VC: Short Project

QP 2022-23

Tracking and Recognition of Objects

Bernat Borràs Civil
Òscar Ramos Nuñez

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Índex

Introducció

Aquest document està dividit en quatre parts: tracking, reconeixement, conclusions i annex. Les parts de tracking i reconeixement contenen els mètodes implementats en el treball, la conclusió reflecta els pensaments un cop realitzat el treball i l'annex conté codi útil i/o referenciat durant el document.

Cada mètode consta d'un text explicatiu del procés d'aprenentatge i prova d'error, l'explicació del mètode final, els resultats obtinguts i els punts forts i febles del mètode. Els resultats de cada mètode venen acompanyats d'un parell de gràfiques on es pot veure, per cada vídeo, la mitjana de detecció de l'objecte (que mostra la precisió de detecció) i el rati de frames amb un Overlapping Ratio major a 0.2 (és a dir quan es solapa la capsa de la predicció amb la real) respecte del total (que mostra si manté l'objecte durant el vídeo o el perd). Són importants els dos gràfics ja que se solen complementar, per exemple en la part de reconeixement podem observar dades baixes en el primer gràfic en el vídeo de Breakfast Club, mentre que aquest mateix vídeo té un rati elevat de frames amb ORatio > 0.2.

Explicarem ara resumidament els apartats de tracking i de reconeixement.

La part de tracking, a més de tenir una petita introducció, conté la implementació del tracking nostre i del tracking d'internet. Anem a veure'ls una mica per sobre:

- Tracking nostre: donada una box (predicció de l'objecte en un frame), agafem la primera amb el groundTruth.txt, busquem la següent dintre de totes les que, amb la mateixa amplada i alçada, tenen el centre dintre de la box anterior, fent que només busquem la següent en una regió d'un 50% més de la longitud de la box original. D'entre totes les possibles ens quedem amb la que té una distància d'histograma menor, amb la box actual.
  Resultats: bons per gran quantitat dels vídeos.

- Tracking d'internet: donada una box corresponent a un frame agafem els punts singulars d'aquesta i, mitjançant SIFT i RANSAC, busquem de trobar-ne la següent.
  Resultats: defectuosos (SIFT acostuma a detectar punts singulars del background).

Pel que fa a la part de reconeixement aquesta també té una introducció, una mica més llarga ja que ens hem trobat amb més problemes, a més de l'explicació del mètode propi i el mètode d'internet. Els mètodes són els següents:

- Reconeixement propi: entrenament d'una xarxa amb un DataSet personalitzat, que conté imatges dels vídeos amb el seu respectiu *label* que identifica quin tipus d'objecte apareix a l'escena i on està.
  Resultats: depenent del vídeo molt bons o mediocres.

- Reconeixement d'internet: a partir d'una xarxa ja entrenada per detectar objectes quotidians (com persones, avions, peixos…) selecció de tipus d'objecte a cercar en cada vídeo i anàlisis de les seves prediccions.
  Resultats: quelcom bons per vídeos on l'objecte es veu clarament i més dolents quan no està tan nítid (com en el *Boxing1* on, malgrat voler cercar una persona no es veu a vegades clar on està amb els colors del background).

Per aconseguir els gràfics per cada mètode i tenir una visió general dels resultats de cadascun d'ells, així com per desenvolupar els dos gràfics de les conclusions (que comparen els quatre mètodes alhora), s'han anat enregistrant els resultats de cada execució final (és a dir de la versió final) de cada mètode.

Tracking

Implementació de codi propi

Introducció

Per a fer el nostre mètode de tracking vam pensar que la millor manera seria buscar per tot el frame quin és el retall de la imatge més similar al corresponent al retall inicial de l'objecte a reconèixer.

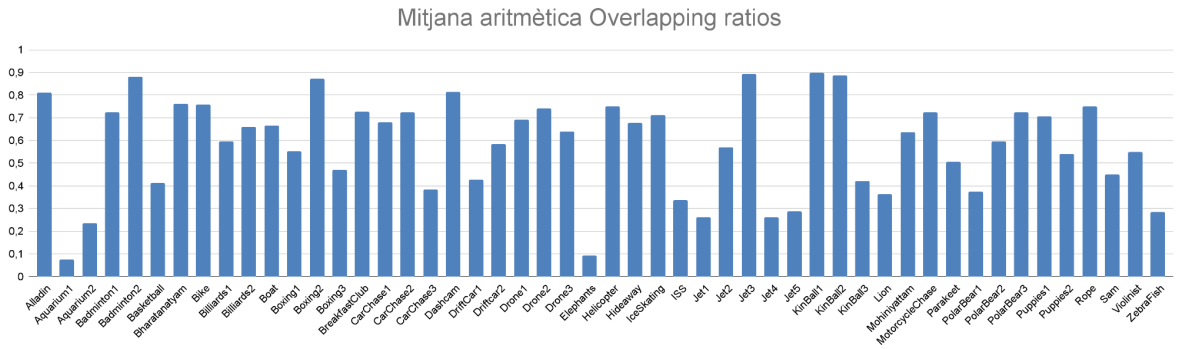
Per a comprovar com de similar eren dos retalls d'imatges, hem optat per calcular la distància de chi-quadrada dels histogrames de colors en dues dimensions. Per tal que ens donés un bon resultat ens calia normalitzar la mida i el color dels dos retalls. La mida dels retalls sempre són de la mateixa mida, i el color es normalitza quan els representem amb una matriu de 2 dimensions de mida 16x16. A part de la distància chi-quadrada vam valorar també la distància euclídea, però els resultats eren un xic inferiors, i el temps d'execució una mica millor.

A priori la idea sembla bona, però abans d'implementar-la sabíem que seria un procés molt lent al fer tantes operacions. Sabíem que hauríem de sacrificar precisió per tal que s'executés en un temps raonable.

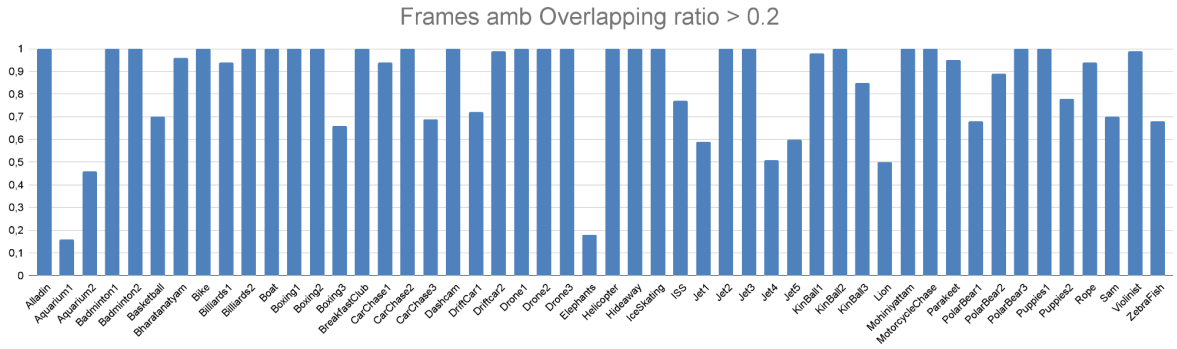
La primera implementació que vam fer era només cercar les possibles caixes tal que es solapessin amb l'anterior, i que estiguessin dins del frame. A més a més, vam reduir el nombre de comprovacions, fent que només se'n produeixi una cada x píxels. Mitjançant prova i error vam trobar un valor que donava un resultat decent i trigava el voltant d'un segon per frame.

Fent proves amb altres vídeos vam veure que el valor que deia quan fer una comprovació, no anava bé per a objectes on la caixa fos petita. Per això la segona versió vam afegir que el nombre de píxels entre retall i retall, varies en funció de la mida de la caixa. Les caixes més petites fan comprovacions més freqüents al necessitar una precisió major, i les caixes més grans fan comprovacions cada més píxels ja que necessiten menys precisió i no volem un temps d'execució gaire gran.

Resultats



Gràfic 1: Mitjana aritmètica de detecció per vídeo mètode tracking propi



Gràfic 2: Rati de frames amb OR major a 0.2 perr vídeo mètode tracking propi

Podem veure que el seguiment de l'objecte al llarg del vídeo es manté bastant en gran part dels vídeos (queda representat per *Frames amb Overlapping ratio > 0.2*). Només en *Aquarium1* i *Elephants* s'ha perdut el tracking significativament, i només *Aquarium2* (a part dels 2 anteriors) no arriba al 50% dels frames traquejats correctament.

Vídeos com *Jet3* i *KinBall2* obtenen un molt bon resultat gràcies al contrast que hi ha entre l'objecte i el fons. En canvi, els 3 vídeos comentats anteriorment obtenen un resultat pèssim al haver-hi un menor contrast amb el fons o una superposició de l'objecte a rastrejar, i un altre molt similar.

En el gràfic *Mitjana aritmètica Overlapping Ratios* podem veure que el percentatge mitjà d'overlapping de la caixa és baix tenint en compte que s'ha pogut traquejar en molts casos l'objecte durant tot el vídeo. Això es deu al fet que la localització de la caixa que dona el nostre algorisme no és gaire precisa.

Defectes i punts positius

El punt fort d'aquest mètode és la capacitat de trobar imatges similars pel voltant d'objectes. Això fa que els vídeos on l'objecte a rastrejar no canvia gaire l'estètica, l'aconsegueix traquejar la majoria de frames.

Sense les optimitzacions fetes, el nostre mètode té tres grans defectes. El primer és el gran cost computacional. Sense aquestes, l'algorisme calcularia la distància entre els histogrames de tots els retalls possibles de la imatge.

El segon gran defecte és que en el cas que l'objecte a traquejar canviï significativament d'estètica, el nostre algorisme el pot perdre fàcilment ja que sempre calcula la distància respecte al retall del frame inicial.

El tercer apareix quan el vídeo té un fons amb molt pes, o es produeix una sobreposició de l'objecte amb un altre. Sense les optimitzacions fetes, una vegada l'objecte deixi de ser oclòs, continuarua fent el rastreig correctament.

Les optimitzacions fetes per reduir dràsticament el temps d'execució ens ha portat més defectes.

L'optimització de cercar només per la zona propera a l'objecte, fa afegir dos defectes. El primer és que si l'objecte a rastrejar, d'un frame a l'altre, produeix un canvi de posició tan brusc, llavors aquest objecte quedaria fora de la regió on busquem similituds, i el perdrem. A més a més, aquesta optimització fa, que si durant el vídeo perds l'objecte, és molt poc probable recuperar-lo, ja que quedaria fora dels límits de la zona de comparacions.

Una altra optimització que ens porta problemes és la de no fer totes les comparacions possibles de la zona, i fer-ne cada x nombre de píxels. Això fa que la localització de la caixa no sigui tan precisa, i sigui més probable perdre l'objecte.

Implementació amb mètodes ja existents (clàssics)

Introducció

Per al mètode de tracking ja existent hem decidit combinar dues tècniques estudiades a classe, la detecció i descripció de punts singulars per SIFT, i l'emparellament d'aquests mitjançant RANSAC.

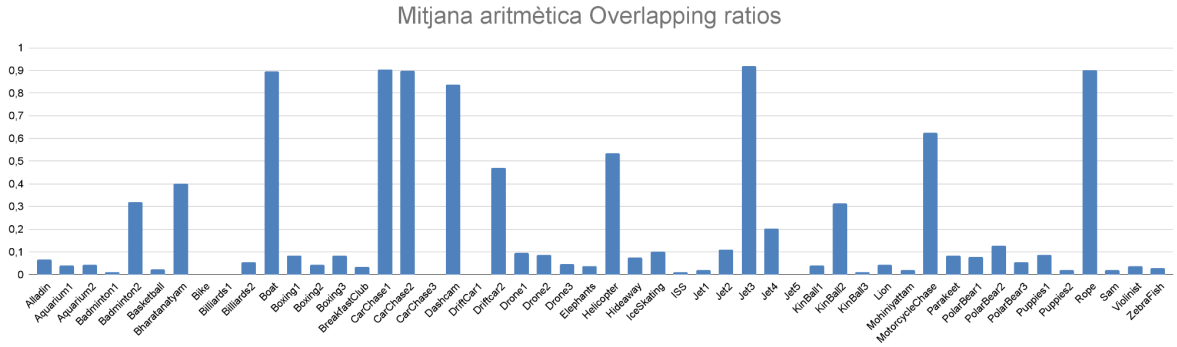
El codi fet itera per a cada vídeo i fa els passos explicats a continuació. Per al primer frame ens desem el retall obtingut amb la bounding box, i ens servirà per a la comparació entre imatges.

Per a cada frame del vídeo, extraiem els descriptors de SIFT del retall de l'objecte del primer frame, i de tot el frame que estem tractant. Amb els punts i els seus descriptors de les dues imatges, els emparallem tal com es va explicar a classe. En el cas que tinguem menys de 3 parelles, descartem el frame al no tenir prou parelles per aplicar RANSAC.

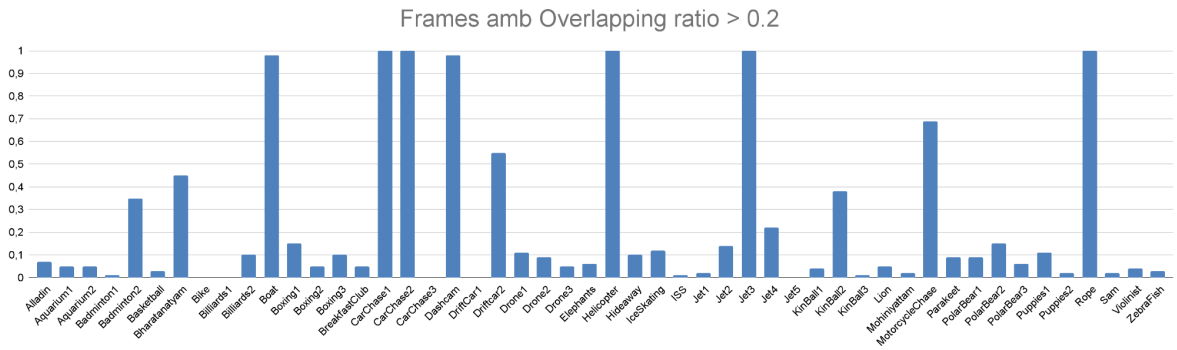
En el cas que tinguem prou parelles, apliquem l'algorisme de RANSAC calculant la millor transformació afi amb aquestes parelles. Finalment calculem l'overlapping del frame comparant la caixa obtinguda per RANSAC amb la caixa òptima.

Hem de dir que el codi es pot optimitzar, no cal calcular cada vegada els descriptors de SIFT del retall inicial. Però com que no teníem problemes greus d'eficiència de temps, vam optar per a la millor llegibilitat del codi.

Resultats



Gràfic 3: Mitjana aritmètica de detecció per vídeo mètode tracking clàssic



Gràfic 4: Rati de frames amb OR major a 0.2 perr vídeo mètode tracking clàssic

Abans de comentar els resultats, cal indicar que en els vídeos *Bike, Billiards1, CarChase3, DrifCar1* i *Jet5* tenen un resultat de 0 en els dos gràfics per la incapacitat de l'algorisme de continuar executant-lo. En aquests vídeos, en un frame, RANSAC és incapaç de trobar la transformació afi tot i tenir almenys 3 emparellaments. Matlab retorna un error, i no ens pot retornar els resultats calculats fins a l'anterior frame.

En aquests dos gràfics veiem que els resultats són molt baixos, excepte 7 vídeos que tenen un molt bon resultat. Els bons resultats individuals d'aquests 7 vídeos són els resultats que ens esperaríem per a cadascun dels 50. En aquests vídeos, SIFT

ha pogut trobar bons punts singulars de l'objecte a cercar i el fons, fent que l'emparellament d'aquests tingui gran quantitat de punts, i la localització de l'objecte sigui gairebé l'òptima.

Malauradament en la resta dels vídeos, els resultats no són els que ens esperàvem. Majoritàriament, hem detectat 2 motius per la qual aquests obtenen un resultat gairebé nul. En vídeos on el fons és detallat i té molt pes, SIFT detecta punts en el background del retall inicial, i a cada frame el matching es produeix en la posició inicial. El segon gran problema obtingut és que la manca d'emparellaments fa que la bounding box trobada per RANSAC estigui molt deformada i dispersa, fent que l'overlapping ratio sigui pèssim.

Imatge 1: Frame 16 de vídeo Bike amb la capsa molt deformada

Defectes i punts positius

El punt positiu que hem trobat en aquest mètode és que en els casos que hi hagi una quantitat important de detalls a l'objecte, i el fons tingui poc pes, el matching produit és gairebé perfecte. Un matching que es produeix constantment durant el vídeo, i just al punt idoni.

Hem detectat uns quants defectes en aquest mètode, i com hem comentat per sobre a l'apartat de resultats, el major és que en les imatges que tenen un fons amb molt pes, a cada frame intenta fer el matching a la mateixa posició. Això també pot comportar que el matching es fa amb punts del fons i punts de l'objecte, fent que la caixa resultant estigui deformada, i obtenint un valor pèssim d'overlapping ratio.

Un altre defecte és la incapacitat de calcular la transofrmació afí mitjançant RANSAC tot i tenir almenys 3 emparellaments. Com bé hem dit anteriorment, Matlab ens retornava un error, i no érem capaços d'aconseguir els resultats anteriors.

També, en objectes poc detallats, SIFT pot trobar molt pocs punts, i al fer l'emparellament pot produir que la bounding box estigui influïda per un *outlier* (Amb 3 punts, un *outlier* té molt pes). Fins i tot, si no s'arriba als 3 emparellements, no es pot calcular la transformació afi, resultant a un overlapping ratio de 0.

Finalment, es produeix un defecte que també apareixia en el mètode anterior. És en el cas que l'objecte a traquejar canviï significativament d'estètica, que el nostre algorisme el pot perdre fàcilment, ja que sempre fa el matching respecte al retall del frame inicial.

Reconeixement

Abans de ficar-nos en les implementacions, tant com pròpies com d'internet, és important destacar que per fer aquesta part del treball s'ha utilitzat Python (en concret python3) com a eina principal.

En un primer moment vam començar a desenvolupar diverses xarxes neuronals dintre de l'aplicatiu de Matlab, modificant paràmetres per ajustar la precisió al màxim, dintre dels nostres coneixements. Un cop vam decidir quins paràmetres i *layers* havia de tenir la nostra xarxa vam començar l'entrenament. Aquest va consistir en fer un entrenament de la xarxa agafant com a imatges objecte 5 retalls de 5 frames on només sortís l'objecte per cada vídeo, formant 250 imatges. Per altra banda, com a imatges no objecte, vam agafar d'internet una sèrie de 1000 imatges random, fruites, escenes, mapes, fotografies, etc.

Un cop obtinguda la xarxa ens vam trobar amb un problema, desenvolupar un codi que ens permetés, amb una xarxa classificadora com la nostra, aconseguir una xarxa de detecció, que ens busqui en la imatge l'objecte pel qual l'hem entrenada.

Imatge 2: Resultat d'una predicció en Matlab amb la xarxa classificadora

En la imatge es pot veure el producte de la nostra xarxa. Degut a que només detecta com a objecte l'Alladin, doncs hem entrenat la xarxa amb els retalls dels frames emmarcant l'objecte de l'escena, el frame sencer ho classifica, correctament, com no objecte.

Però aquest no és el resultat que volem, l'objectiu és que la xarxa ens indiqui on és l'objecte. Després d'una cerca exhaustiva no vam trobar cap mètode viable per fer-ho, tots els codis d'exemple eren poc intel·ligibles o demanaven "ToolBoxes" bastant complexes les quals desconeixíem.

Per aquest motiu vam decidir provar sort amb Python. Ara el nostre objectiu va passar a aconseguir entrenar una xarxa amb el nostre DataSet personalitzat de crops de fotogrames dels vídeos per tal de, amb la xarxa, aconseguir el reconeixement.

Implementació de codi propi

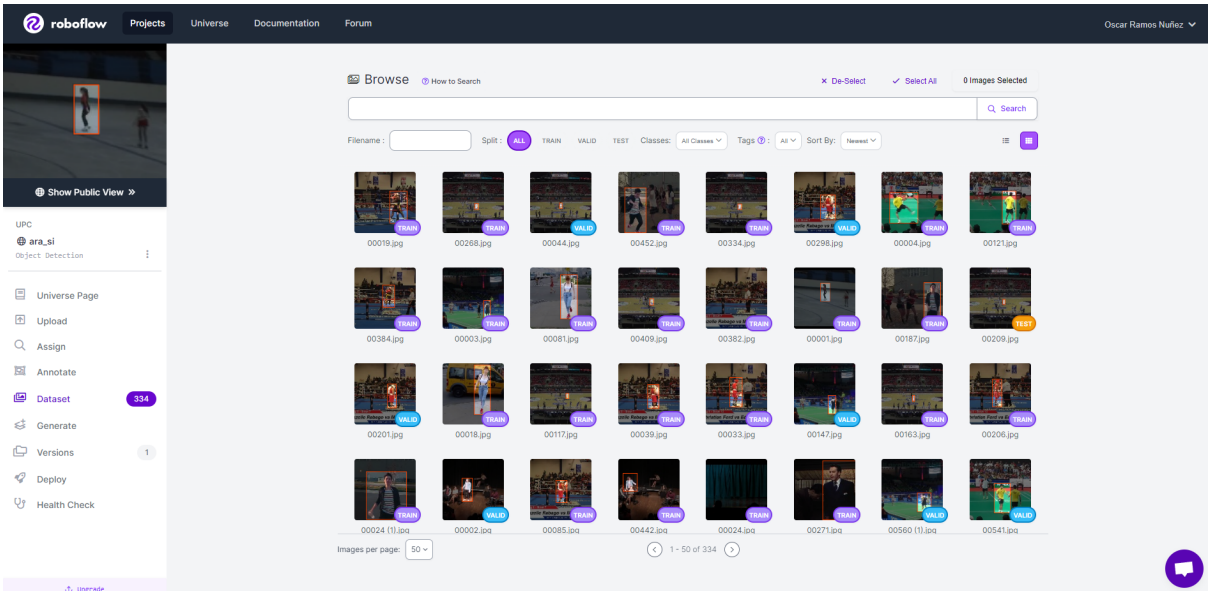
Aquesta implementació ha estat la part més difícil del treball, ja que no hem obtingut resultats fiables fàcilment.

Des d'un primer moment volíem desenvolupar una xarxa en Matlab, malgrat com s'ha comentat vam haver de canviar la metodologia i provar amb Python. A Matlab aconseguíem entrenar la xarxa per classificar retalls de frames per determinar si aquests retalls eren objecte, o no, però no sabíem ben bé com convertir la xarxa en una detectora d'objectes (no volem que ens digui si la imatge és un objecte, volem que ens marqui on està).

L'objectiu en Python va ser, donada una xarxa, intentar entrenar-la per la detecció de l'objecte a un vídeo en concret, Alladin, per tal de no haver de fer un training molt llarg per tots els vídeos alhora, sense la certesa de que ens funcionés un cop acabada l'execució (només el traininig per el vídeo de Alladin ens podia trigar fins a 5 hores depenent dels "epochs" del training).

Després de cada entrenament d'una xarxa, observàvem uns resultats molt dolents (moltes vegades si seleccionàvem un frame del vídeo per tal que la xarxa ens fes una predicció de l'objecte en el frame ens marcava tota la imatge com a objecte).
Vam provar de canviar el script d'entrenament, els epochs, el DataSet de train, etc. Però sempre acabàvem amb uns resultats similars.

Durant la cerca de la resposta del mal funcionament de les nostres xarxes vam descobrir Roboflow (https://app.roboflow.com/). Aquesta web ens va permetre fer un *labeling* més precís (podent seleccionar el tipus d'objecte seleccionat en una imatge), deixant enrere a els 250 retalls de frames centrats en l'objecte del vídeo utilitzats fins al moment. La web també permet exportar el DataSet, un cop les imatges han estat pujades i anotades (selecció d'objecte dintre la imatge + anomenar quin tipus d'objecte és, com persona, pilota,etc), per fer un training "més professional".



Imatge 3: DataSet creat en Roboflow

Des de la pròpia web es recomana utilitzar la xarxa Yolov8 per fer el training amb el DataSet exportat. Vam començar a pujar imatges dels vídeos per tal de detectar l'Alladin. Malauradament, i després de canviar aproximadament 10 vegades el DataSet (afegint o eliminant imatges), i després del posterior training a la xarxa, aconseguíem resultats pèssims, ens detectava 2 o 3 objectes per frame, i cap d'ells era l'Alladin, pobre.

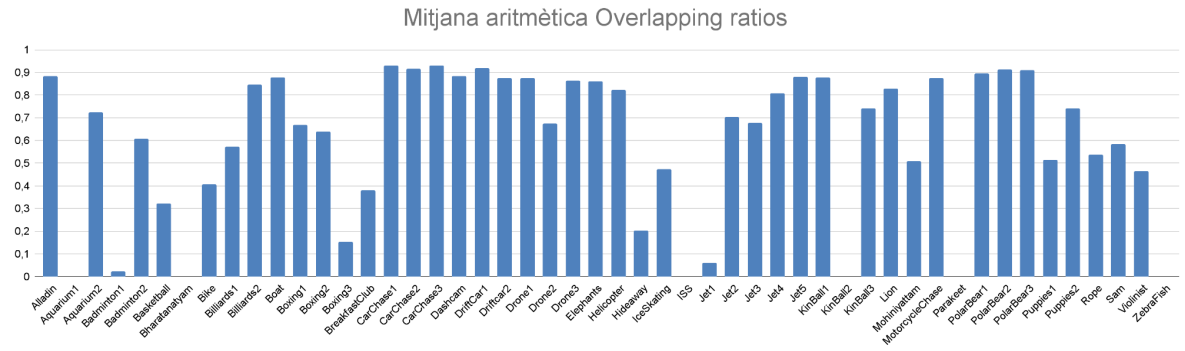
Imatge 4: Resultat predicció xarxa intentant identificar a Alladin en la imatge

Posteriorment, vam descobrir que des de la pròpia pàgina s'ofereix una opció per fer el training de la xarxa, amb un DataSet que hagis creat dintre la mateixa pàgina. I aquesta vegada si vam obtenir uns bons resultats (al voltant del 90% dels frames de

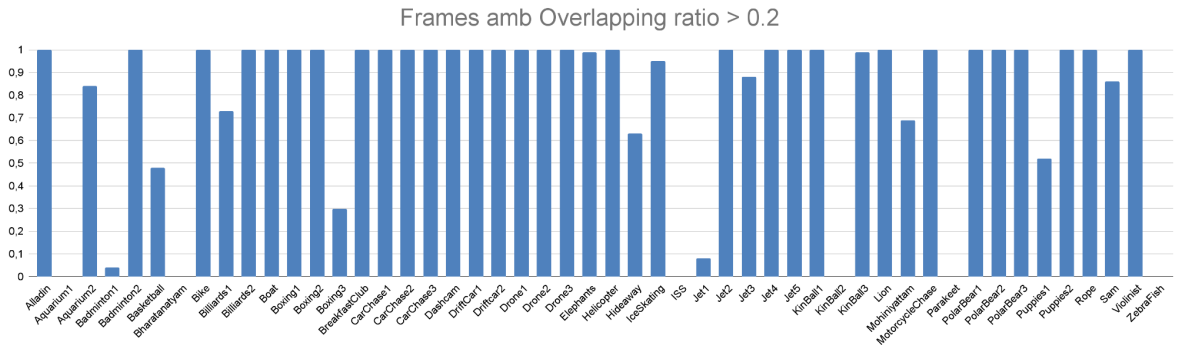
Alladin els detectava correctament). Així doncs, vam ampliar el DataSet amb 6-8 imatges per cada vídeo. Per cadascuna es va seleccionar manualment l'objecte corresponent del vídeo i es va etiquetar amb la classe corresponent (no volíem que detectés "objecte", si estem en el vídeo Aquarium, per exemple, volem que detecti "Fish"), arribant fins a X classes diferents (llista classes). Un cop ampliat el DataSet vam procedir a tornar a entrenar la xarxa en la web. Vam aconseguir resultats satisfactoris, després de la quantitat de xarxes defectuoses provades vam aconseguir una que realment funcionava, per quasi tots els vídeos. Ens vam trobar, però, amb un últim problema, la web no permet descarregar una xarxa entrenada. El que si permet, però, és fer un "deploy local" de la xarxa, és a dir no la tens físicament com un arxiu, però pots utilitzar-la en els teus scripts i codis si tens la clau privada (la qual teníem, ja que la havíem creat nosaltres). Mitjançant aquest petit script podíem "jugar" amb la xarxa i provar frames de diferents vídeos, a més de guardar el resultat de la predicció en un arxiu.

El codi fet per nosaltres, simplement itera per a cada vídeo i frame, i fa una crida a l'api de la xarxa, i tracta els resultats obtinguts. Dels objectes detectats per la xarxa, descartàvem els que no estiguessin etiquetats amb l'objecte que volíem. Per exemple, en el vídeo *Alladin* descartàvem tots els objectes no identificats com a persona. A més a més, hem de dir que en el cas que es detectessin vàries instàncies d'una mateixa classe d'objecte, només seleccionem la detecció amb el major *confidence level*. Finalment el nostre codi calcula l'*overlapping ratio* de cada frame, i l'imprimeix en un fitxer.

Resultats



Gràfic 5: Mitjana aritmètica de detecció per vídeo mètode reconeixement propi



Gràfic 6: Rati de frames amb OR major a 0.2 per vídeo mètode reconeixement propi

En aquests gràfics podem observar que en gran part dels vídeos, la xarxa ha fet una bona feina en reconèixer els objectes. La majoria d'aquests vídeos són captures on l'objecte a reconèixer apareix nítidament, i només una sola instància d'aquest tipus. Per exemple, *MotorcycleChase*, la xarxa és capaç de reconèixer sense cap problema la moto, assignant-li un gran *confidence level*. Els vídeos de *CarChase*, tot i reconèixer múltiples instàncies de cotxes, el que volem reconèixer és el que apareix més nítidament, i precisament és el que seleccionem al tenir un major valor de *confidence level*.

Tot i així, els vídeos *ISS* i *Bharatanatyam* no s'ha pogut reconèixer correctament l'objecte. En aquest cas calia reconèixer la cara d'una persona, classe d'objecte que no hem creat. Per exemple, en el vídeo *Bharatanatyam*, la xarxa detectava la persona correctament, i centrava la caixa al centre de la persona, i al ser tan petita feia que les dues caixes no se solapessin.

Altres vídeos com *Aquarium1, ZebraFish* i *Parakeet*, la manca de detecció de l'objecte era degut a la múltiple aparició d'objectes de la mateixa classe. Per exemple, en els vídeos *AquariumX*, apareixen múltiples peixos. La xarxa els detecta correctament (incloent-hi el que volem trackejar, però al assignar un nivell menor de *confidence level*, el nostre codi selecciona un peix incorrecte.

Defectes i punts positius

El millor punt d'aquest mètode és la gran capacitat de detectar objectes. Com hem vist en els resultats, gran part dels vídeos s'ha aconseguit un valor d'*overlapping* mitjà alt. Entrenar la xarxa amb retalls d'imatges on apareixen aquests objectes fa disparar el confidence level dels objectes reconeguts.

Possiblement, el major defecte de la xarxa és la presència d'interferències entre objectes de la mateixa classe de diferents vídeos. És a dir, la classe *persona* és present a una gran part dels vídeos, i la xarxa busca característiques genèriques

d'una persona. Això fa, per exemple, que en el vídeo *Boxing3* es reconeguin diverses persones, on algunes d'aquestes tenen un major *confidence level*, fent que la detecció sigui baixa.

També voldria comentar que es podria millorar la precisió del reconeixement de la xarxa, entrenant-la amb una quantitat de mostres significativament major.

Implementació amb mètodes ja existents

Introducció

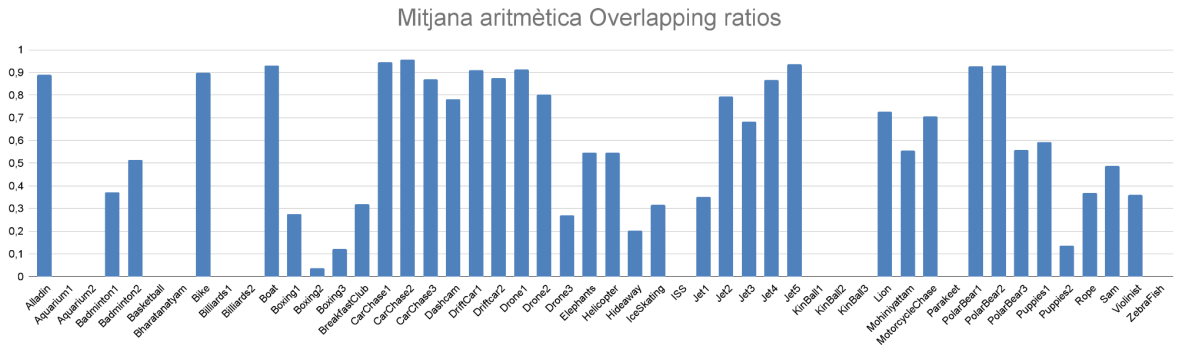
Per al reconeixement mitjançant mètodes ja implementats vam cercar diferents xarxes ja entrenades que permetessin reconèixer una selecció d'objectes habituals. D'aquestes xarxes cercades vam trobar una anomenada Yolov8, una xarxa dissenyada específicament per a ser utilitzada amb Python. La xarxa és capaç de reconèixer 80 objectes diferents, dels que s'inclouen persona, automòbil, bicicleta, diferents animals, i els objectes quotidians més comuns.
El nostre codi s'encarrega de proveir a la xarxa els frames dels vídeos, i tractar les dades obtingudes per la xarxa. D'aquestes dades, descartàvem els objectes que no formen part d'una classe que indicàvem per a cada vídeo. És a dir, per a cada objecte que la xarxa obtenia, et retornava el nom de l'objecte que creia que era.

Per a cada vídeo hem seleccionat manualment l'objecte que cal cercar. Hi ha casos on ha sigut senzill seleccionar-lo, ja que l'objecte a cercar estava suportat per a la xarxa. D'altres, com el vídeo de *Lion*, al no haver-hi un objecte marcat com a lleó, la xarxa es confonia pensant que realment era una vaca. En aquests casos hem optat per seleccionar la classe objecte que la xarxa detectava. Tot i això hi ha alguns vídeos on ens ha sigut impossible seleccionar cap classe d'objecte que fes detectar l'objecte que ens interessava. *Aquarium1, Aquarium2* i *ZebraFish* hem optat per no analitzar-los, ja que la xarxa no té cap classe *peix*, i cap de les classes presents feia reconèixer el peix que ens interessava.

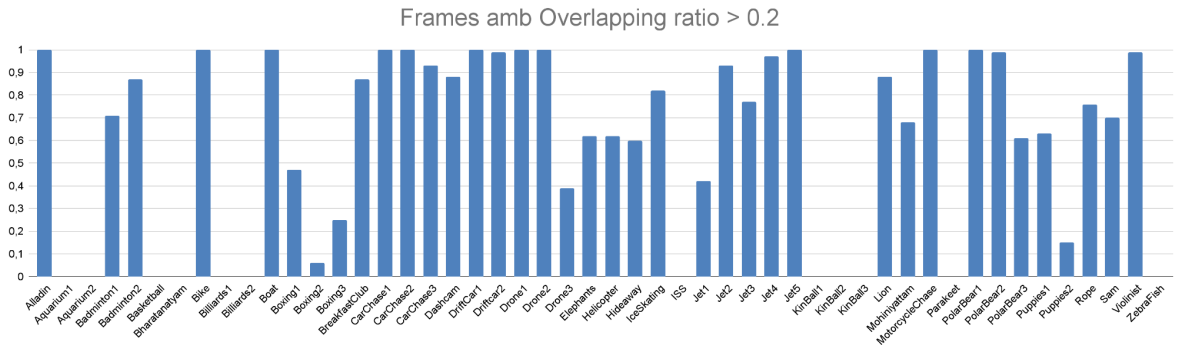
A més a més, hem de dir que en el cas que es detectessin vàries instàncies d'una mateixa classe d'objecte, només seleccionem la detecció amb el major *confidence level*.

Finalment el nostre codi calcula l'*overlapping ratio* de cada frame, i imprimeix en un fitxer.

Resultats



Gràfic 7: Mitjana aritmètica de detecció per vídeo mètode reconeixement xarxa internet



Gràfic 8: Rati de frames amb OR major a 0.2 per vídeo mètode reconeixement xarxa internet

Com podem veure a les gràfiques, els resultats són dispersos. Hi ha vídeos com *CarChase2* i *Jet5* que els resultats són molt pròxims a 1 (hem pogut reconèixer amb bastant precisió l'objecte), i resultats pròxims a 0, o directament 0 (no s'ha pogut reconèixer l'objecte).

Els vídeos amb resultats elevats són els vídeos on els objectes a reconèixer coincidien perfectament amb una classe que la xarxa era capaç de reconèixer. A més a més, els millors resultats obtinguts venen de vídeos on només podem observar una sola instància de l'objecte, com per exemple *Boat* o *Alladin*.

Vídeos com *Boxing2* o *Badminton1*, calia reconèixer la classe *persona*, classe disponible a la xarxa. El fet que a cada imatge apareguessin diverses persones, se selecciona la que la xarxa creu que té més possibilitats de ser persona, i en molts casos se seleccionava una que no és la que hem de reconèixer. Això fa que aquests resultats siguin mediocres.

Finalment, vídeos com *Bharatanatyam* tenen un resultat de 0, ja l'objecte a reconèixer no està inclòs en la xarxa. En aquest cas específic, el que volíem

reconèixer era la cara de la persona però la xarxa detectava la persona, correctament, i centrava la caixa al centre d'aquesta.

Defectes i punts positius

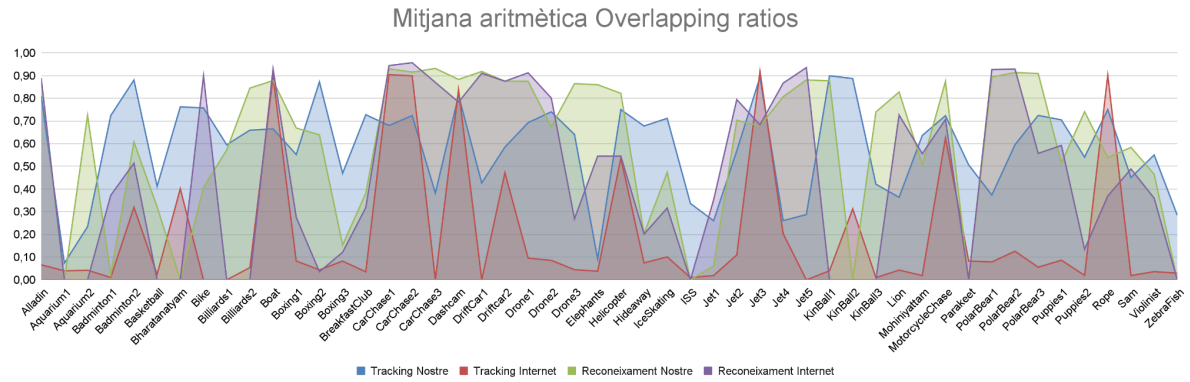
Aquest mètode reconeix amb molta precisió els objectes que suporta la xarxa, i que només n'apareix una sola instància. Això fa que reconegués sense problema l'objecte i, en tenir només una sola instància, és la que seleccionàvem.

El major defecte del mètode és el de reconèixer objectes que la xarxa no suporta, fent que els resultats siguin nefastos.

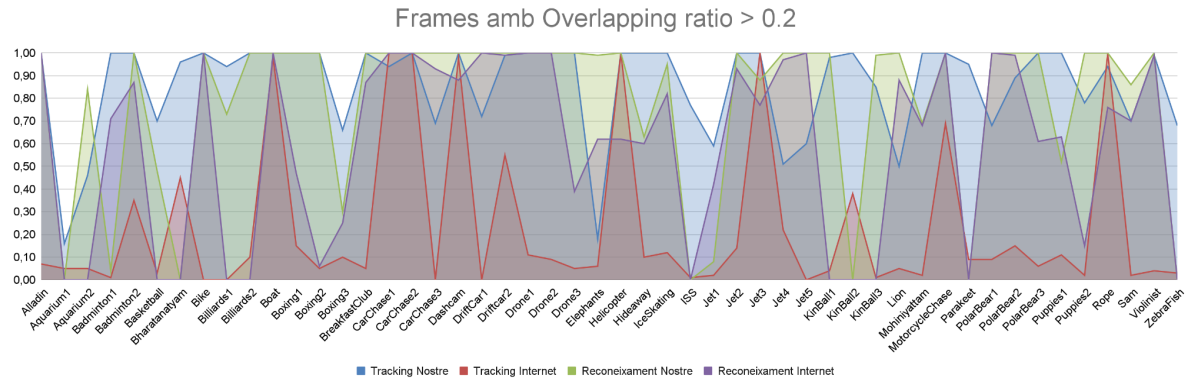
El segon major defecte és la selecció de l'objecte en el cas que reconeixes múltiples instàncies. Com hem dit abans, el nostre mètode seleccionem l'objecte amb més *confidence level*, i en molts casos l'objecte seleccionat no era el que ens interessava, tot i ser detectat per la xarxa.

Conclusions

Els gràfics que es mostren a continuació és la superposició dels gràfics mostrats anteriorment de cada mètode, per a fer una millor comparació. Hem optat per canviar el gràfic de barres per un d'àrees per a una millor visualització.



Gràfic 9: Comparació mitjanes aritmètiques d'overlapping ratios dels quatre mètodes



Gràfic 10: Comparació de ratis de frames amb overlapping ratio > 0.2 dels quatre mètodes

La conclusió més òbvia que podem extreure d'aquests gràfics és que el mètode *Tracking d'Internet* (línia vermella) és el que pitjors resultats treu. Aquesta comparació també es veu clarament comparant els gràfics de barres obtinguts entre els diferents mètodes. El principal motiu és que en bastants vídeos l'execució es talla en un cert frame i provoca overlapping ratios de 0.

També podem observar que les línies blaves i verdes (corresponents a *Tracking nostre* i *Reconeixement nostre*) són les que tendeixen a estar més amunt. És a dir que tenen els overlapping ratios majors i que es poden trackejar durant més frames dels vídeos. Mentre que el *Reconeixement nostre* deu a la bona precisió la xarxa ja entrenada juntament amb les noves classes afegides, *Tracking nostre* deu a la bona precisió la constància en detecció d'imatges similars.

Podem observar que la línia lila (corresponent a *Reconeixement d'Internet*) no té uns resultats tan baixos com *Tracking d'Internet*, però tampoc s'arriba a la precisió i constància dels mètodes *Tracking nostre* i *Reconeixement nostre*.

Principalment, el reconeixement ja entrenat ha estat perjudicat per la manca de tipus d'objectes a reconèixer, com la classe peix, que provoca una mala detecció d'objectes en els frames de certs vídeos. A més, la xarxa d'Internet, al estar ja entrenada, té un altre desavantatge respecte a una xarxa nova sense dades. La xarxa sense entrenar, és a dir la del mètode propi, "coneixerà" l'objecte *persona* a partir dels retalls dels vídeos, com l'*Alladin* o els jugadors de Bàdminton, que són justament el que volem buscar en els vídeos. En canvi, la xarxa d'Internet ha après les característiques d'una persona amb altres imatges. Potser fora dels vídeos la xarxa d'Internet funciona millor, però pels vídeos de la pràctica la xarxa pròpia, entrenada explícitament pels vídeos, funciona millor.

Finalment, hem de comentar que no ens quedaríem amb cap d'aquests 4 mètodes. Creiem que la combinació dels dos mètodes amb millors resultats (*Tracking nostre* i *Reconeixement nostre*) fan un cinquè mètode que superaria els resultats d'aquests 4. Aquest cinquè mètode tractaria d'utilitzar la xarxa de *Reconeixement nostre*. Si només apareix una detecció de la classe de l'objecte que ens interessa, l'escolliríem, tal com hem fet amb aquest mètode. En el cas que hi haguessin més d'un objecte reconegut, es calcularia la distància chi-quadrada respecte al retall inicial, tal com s'ha fet en el mètode *Tracking nostre*. Així evitem el major problema de les xarxes de reconeixement (problema de selecció). A part, la xarxa que entrenaríem tindria la classe *cara*, per a millorar els resultats dels vídeos *ISS* i *Bharatanatyam*.

Annex

Codi referenciat durant l'informe o utilitzat en el treball.

Codi font tracking nostre (comparació histogrames)

```
% Per a tots els vídeos
videoNames =dir("TinyTLP\");
nFrames = 100;
OverlappingRates = zeros([50, nFrames]);
for i=1:50
   name = videoNames(i).name;
   OverlappingRates(i,:) = ORatioCompHistogrames(name,nFrames);
end

% Per a una selecció de vídeos
videoNames = {'Alladin','PolarBear1'};
nFrames = 100;
OverlappingRates = zeros([size(videoNames,2), nFrames]);
for i=1:size(videoNames,2)
   name = videoNames{i};
   OverlappingRates(i,:) = ORatioCompHistogrames(name,nFrames);
end

function ORatio = ORatioCompHistogrames(videoName, nFrames)
   close all
   hold off

   % Obtenir nom de les imatges i del fitxer groundtruth_rect
   path1 = strcat('./TinyTLP/', videoName);
   path11 = strcat(path1,'/groundtruth_rect.txt');
   path2 = strcat(path1,'/img/*.jpg');

   % Obtenir Bounding Boxes Òptimes
   BB = importdata(path11);
   Idir = dir(path2);

   figure
   hold on

   % Retall de l'objecte a rastrejar
   previousBB = BB(1,2:5);
   filename = horzcat(Idir(1).folder,'/',Idir(1).name);
   I = imread(filename);
   previousCrop = imcrop(I,previousBB);

   % Vector de l'overlapping ratio de cada frame del vídeo
   ORatio = zeros(1,nFrames);
```

```
   for i=1:nFrames
       filename = horzcat(Idir(i).folder,'/',Idir(i).name);
       I = imread(filename);

       % Inicialitzem amb la posició actual
       bestBB = previousBB;
       bestCrop = imcrop(I, bestBB);
       min_diff = computeDiff(previousCrop,bestCrop);

       % Càlcul del nombre de píxels a comprovar
       i_step = floor(BB(i,4)/25);
       j_step = floor(BB(i,5)/25);

       % Iterar pels punts de la bounding box
       for i_cord = 1:i_step:previousBB(3)
           for j_cord = 1:j_step:previousBB(4)

               % Calcular distància respecte el retall inicial
               [tlx,tly] =
topLeftCoordinate([previousBB(1)+i_cord,previousBB(2)+j_cord],previousBB(3),previousBB(4));
               possibleBB = [tlx,tly,previousBB(3),previousBB(4)];
               Crop = imcrop(I, possibleBB);
               diff = computeDiff(previousCrop,Crop);

               % Comprovar que la caixa sigui la de menys distància
               if diff < min_diff
                   min_diff = diff;
                   bestBB = possibleBB;
               end
           end
       end

       % Càlcul overlapping ratio de la millor Bounding Box
       ORatio(1,i) = bboxOverlapRatio([bestBB(1),bestBB(2),BB(i,4),BB(i,5)],BB(i,2:5));

       %Mostrar les imatges amb les caixes
       imshow(I)
       rectangle('Position',[bestBB(1),bestBB(2),BB(i,4),BB(i,5)],'EdgeColor','red');
       rectangle('Position',BB(i,2:5),'EdgeColor','yellow');
       drawnow

       previousBB = bestBB;
   end
end

% Retorna la distància entre les dues imatges
function diff = computeDiff(I1,I2)
   if size(I1) == size(I2)
       hist1 = calculateHistogram(I1);
       hist2 = calculateHistogram(I2);
       diff = compareHistograms(hist1,hist2);
   else
```

```
        diff = 10000000000;
    end
end

% Retorna les coordenades de la cantonada superior esquerra
function [minx, miny] = topLeftCoordinate(centralPoint, width, height)
    minx = centralPoint(1) - width/2;
    miny = centralPoint(2) - height/2;
end

% Retorna l'histograma de color normalitzat en dues dimensions
function [hist] = calculateHistogram(im)
    I = sum(rgb2gray(im),'all')/(size(im,1)*size(im,2))/255;
    im_norm = (double(im)/255)/(3*I);
    hist = zeros([16,16]);
    for i = 1:size(im, 1)
        for j = 1:size(im, 2)
            r = min(floor(15*im_norm(i,j,1)) + 1, 16);
            g = min(floor(15*im_norm(i,j,2)) + 1, 16);
            hist(r,g) = hist(r,g) + 1;
        end
    end
end

% Retorna la distància chi-quadrada entre els dos histogrames
function [s] = compareHistograms(hist1, hist2)
    s = 0;
    for i = 1:16
        for j = 1:16
            if (hist1(i,j) + hist2(i,j) > 0)
                s = s + (hist1(i,j) - hist2(i,j))*(hist1(i,j) - hist2(i,j))/(hist1(i,j) +
hist2(i,j));
            end
        end
    end
end
```

Codi font tracking ja existent (SIFT + RANSAC)

```
% Per a tots els vídeos
videoNames =dir("TinyTLP\");
nFrames = 100;
OverlappingRates = zeros([50, nFrames]);
for i=1:50
   name = videoNames(i).name;
   OverlappingRates(i,:) = ORatioSiftRansac(name,nFrames);
end

% Per a una selecció de vídeos
videoNames = {'Alladin','PolarBear1'};
nFrames = 100;
OverlappingRates = zeros([size(videoNames,2), nFrames]);
for i=1:size(videoNames,2)
   name = videoNames{i};
   OverlappingRates(i,:) = ORatioSiftRansac(name,nFrames);
end

function [ORatio] = ORatioSiftRansac(nom_fitxer, nFrames)
   close all
   hold off

   % Obtenir nom de les imatges i del fitxer groundtruth_rect
   path1 = strcat('./TinyTLP/', nom_fitxer);
   path11 = strcat(path1,'/groundtruth_rect.txt');
   path2 = strcat(path1,'/img/*.jpg');

   % Obtenir Bounding Boxes Òptimes
   BB = importdata(path11);
   Idir = dir(path2);

   % Retall de l'objecte inicial
   filename = horzcat(Idir(1).folder,'/',Idir(1).name);
   I = imread(filename);
   B1 = BB(1,2:5);
   im_obje = rgb2gray(imcrop(I,B1));
   im_obj = im_obje(:,:,1);
   imshow(im_obj);

   % Vector de l'overlapping ratio de cada frame del vídeo
   ORatio = zeros(1,nFrames);

   for i = 1:nFrames
       filename = horzcat(Idir(i).folder,'/',Idir(i).name);
       I2 = imread(filename);

       im_esce = rgb2gray(I2);
       im_esc = im_esce(:,:,1);

       % SIFT
```

```
        kp_obj = detectSIFTFeatures(im_obj);
        kp_esc = detectSIFTFeatures(im_esc);
        [feat_obj, kp_obj] = extractFeatures(im_obj,kp_obj);
        [feat_esc, kp_esc] = extractFeatures(im_esc,kp_esc);

        % Emparallament
        pairs = matchFeatures(feat_obj, feat_esc);
        m_kp_obj = kp_obj(pairs(:,1));
        m_kp_esc = kp_esc(pairs(:,2));

        % Ransac
        if (m_kp_obj.Count ≥ 3) && (m_kp_esc.Count ≥ 3)
            % Càlcul transformació afi
            [T] = estimateGeometricTransform2D(m_kp_obj,m_kp_esc, "affine");

            % Càlcul Bounding Box
            Bi = BB(i,2:5);
            fo = Bi(4);
            co = Bi(3);
            box = [1,1; co,1; co,fo; 1,fo; 1,1];
            nbox = transformPointsForward(T,box);
            B2 = [nbox(1,1),nbox(1,2),Bi(3),Bi(4)];

            % Càlcul overlapping ratio
            ORatio(1,i) = bboxOverlapRatio(Bi,B2);

            % Mostrar imatges amb bounding box
            imshow(im_esc)
            hold on
            line(nbox(:,1),nbox(:,2));
            drawnow
        end
    end
end
```

Codi font reconeixement nostre (xarxa entrenada per nosaltres)

```
import sys
import roboflow

# Diccionari clau: nom-video, valor: objecte a detectar
objectNames = {
   'Alladin':'Person',
   'Aquarium1':'Fish',
   'Aquarium2':'Fish',
   'Badminton1':'Person',
   'Badminton2':'Person',
   'Basketball':'Person',
   'Bharatanatyam':'Person',
   'Bike':'Bike',
   'Billiards1':'ball',
   'Billiards2':'ball',
   'Boat':'Boat',
   'Boxing1':'Person',
   'Boxing2':'Person',
   'Boxing3':'Person',
   'BreakfastClub':'Person',
   'CarChase1':'Car',
   'CarChase2':'Car',
   'CarChase3':'Car',
   'Dashcam':'Car',
   'DriftCar1':'Car',
   'DriftCar2':'Car',
   'Drone1':'Person',
   'Drone2':'Person',
   'Drone3':'Person',
   'Elephants':'Elephant',
   'Helicopter':'Helicopter',
   'Hideaway':'Person',
   'IceSkating':'Person',
   'ISS':'Person',
   'Jet1':'Jet',
   'Jet2':'Jet',
   'Jet3':'Jet',
   'Jet4':'Jet',
   'Jet5':'Jet',
   'KinBall1':'ball',
   'KinBall2':'ball',
   'KinBall3':'ball',
   'Lion':'Lion',
   'Mohiniyattam':'Person',
   'MotorcycleChase':'MotorBike',
   'Parakeet':'Bird',
   'PolarBear1':'PolarBear',
   'PolarBear2':'PolarBear',
   'PolarBear3':'PolarBear',
```

```
    'Puppies1':'Dog',
    'Puppies2':'Dog',
    'Rope':'Person',
    'Sam':'Person',
    'Violinist':'Person',
    'ZebraFish':'Fish'
}

# Retorna cantonada de dalt a l'esquerra de la caixa
def centerToTopLeft(coordinates, width, height):
    (cx, cy) = coordinates
    minx = cx - width/2
    miny = cy - height/2
    return (minx,miny)

# Retorna Overlapping Ratio
def overlapRatio(xmin1, ymin1, xmin2, ymin2, width, height):
    xmax1 = xmin1 + width
    xmax2 = xmin2 + width
    ymax1 = ymin1 + height
    ymax2 = ymin2 + height

    xmin = min(xmax1,xmax2) - max(xmin1,xmin2)
    if xmin < 0:
        xmin = 0
    ymin = min(ymax1,ymax2) - max(ymin1,ymin2)
    if ymin < 0:
        ymin = 0
    area_interseccio = xmin*ymin
    area_unio = 2*width*height - area_interseccio
    return area_interseccio/area_unio

# Càlcul del overlapping del frame segons oclusió i multiple selecció
def selectOverlapRatio(points, xmin, ymin, width, height, isLost):
    if isLost:
        if len(points):
            return 0
        else:
            return 1

    if not len(points):
        return 0

    # Escollir caixa amb millor confidence level
    max_c = 0
    max_p = (0,0)
    for (conf, point) in points:
        if conf > max_c:
            max_c = conf
            max_p = point

    (x,y) = max_p
```

```
    return overlapRatio(xmin, ymin, x, y, width, height)

def runVideo(videoname, model):
    # Llegir fitxer dades
    with open('./TinyTLP/' + videoname + '/groundtruth_rect.txt', 'r') as f:
        BBS = [[int(num) for num in line.split(',')] for line in f]

    overlappingRatios = []

    # Càlcul millor caixa
    for i in range(1,101):
        # Nom de les imatges segons valor de i
        name = "./TinyTLP/" + videoname + "/img/000" + str(i) + ".jpg"
        if i < 10:
            name = "./TinyTLP/" + videoname + "/img/0000" + str(i) + ".jpg"
        if i ⩾ 100:
            name = "./TinyTLP/" + videoname + "/img/00" + str(i) + ".jpg"

        prediction = model.predict(name)

        [_,xmin,ymin,width,height,isLost] = BBS[i]
        points = []
        for p in prediction:
            # Descartar objectes
            if p["class"] ≠ objectNames[videoname]:
                continue
            coords = (p["x"], p["y"])
            # Afegir punt com a possible candidat
            points.append((r["confidence"], centerToTopLeft(coords, width, height)))

        # Afegir overlapping ratio de la millor caixa
        overlappingRatios.append(selectOverlapRatio(points, xmin, ymin, width, height, isLost))

    # Escriptura de valors al fitxer
    with open('./' + videoname + 'Nostre.txt', 'w') as k:
        for o in overlappingRatios:
            k.write(str(o) + '\n')


rf = roboflow.Roboflow(api_key="7ryBC8sKb0QeK9S2EXmK")
project = rf.workspace().project("upc-3sj4d/ara_si")
model = project.version("1").model

#per a un sol video rebut com a paràmetre
#videoname = sys.argv[1]
#runVideo(videoname, model)

#per a fer-los tots
for v in objectNames.keys():
    runVideo(v, model)
```

Codi font reconeixement ja existent (xarxa ja entrenada)

```
from ultralytics import YOLO
import sys

# Diccionari clau: nom-video, valor: objecte a detectar
objectNames = {
   'Alladin':'person',
   #'Aquarium1':'person',
   #'Aquarium2':'person',
   'Badminton1':'person',
   'Badminton2':'person',
   'Basketball':'person',
   'Bharatanatyam':'person',
   'Bike':'person',
   'Billiards1':'sports ball',
   'Billiards2':'sports ball',
   'Boat':'boat',
   'Boxing1':'person',
   'Boxing2':'person',
   'Boxing3':'person',
   'BreakfastClub':'person',
   'CarChase1':'car',
   'CarChase2':'car',
   'CarChase3':'car',
   'Dashcam':'car',
   'DriftCar1':'car',
   'DriftCar2':'car',
   'Drone1':'person',
   'Drone2':'person',
   'Drone3':'person',
   'Elephants':'elephant',
   'Helicopter':'airplane',
   'Hideaway':'person',
   'IceSkating':'person',
   'ISS':'person',
   'Jet1':'airplane',
   'Jet2':'airplane',
   'Jet3':'airplane',
   'Jet4':'airplane',
   'Jet5':'airplane',
   'KinBall1':'sports ball',
   'KinBall2':'sports ball',
   'KinBall3':'sports ball',
   'Lion':'cow',
   'Mohiniyattam':'person',
   'MotorcycleChase':'motorcycle',
   'Parakeet':'bird',
   'PolarBear1':'bear',
   'PolarBear2':'bear',
   'PolarBear3':'bear',
```

```
    'Puppies1':'dog',
    'Puppies2':'dog',
    'Rope':'person',
    'Sam':'person',
    'Violonist':'person',
    #'ZebraFish':'person'
}

# Retorna cantonada de dalt a l'esquerra de la caixa
def yoloBoxToTopLeft(coordinates, width, height):
    (cx, cy) = ((coordinates[0]+coordinates[2])/2,(coordinates[1]+coordinates[3])/2)
    minx = cx - width/2
    miny = cy - height/2
    return (minx,miny)

# Retorna Overlapping Ratio
def overlapRatio(xmin1, ymin1, xmin2, ymin2, width, height):
    xmax1 = xmin1 + width
    xmax2 = xmin2 + width
    ymax1 = ymin1 + height
    ymax2 = ymin2 + height

    xmin = min(xmax1,xmax2) - max(xmin1,xmin2)
    if xmin < 0:
        xmin = 0
    ymin = min(ymax1,ymax2) - max(ymin1,ymin2)
    if ymin < 0:
        ymin = 0
    area_interseccio = xmin*ymin
    area_unio = 2*width*height - area_interseccio
    return area_interseccio/area_unio

# Càlcul del overlapping del frame segons oclusió i multiple selecció
def selectOverlapRatio(points, xmin, ymin, width, height, isLost):
    if isLost:
        if len(points):
            return 0
        else:
            return 1

    if not len(points):
        return 0

    # Escollir caixa amb millor confidence level
    max_c = 0
    max_p = (0,0)
    for (conf, point) in points:
        if conf > max_c:
            max_c = conf
            max_p = point

    (x,y) = max_p
```

```
    return overlapRatio(xmin, ymin, x, y, width, height)

def runVideo(videoname, model):
    # Llegir fitxer dades
    with open('TinyTLP/' + videoname + '/groundtruth_rect.txt', 'r') as f:
        BBS = [[int(num) for num in line.split(',')] for line in f]

    results = model.predict(source="TinyTLP/" + videoname + "/img")
    overlappingRatios = []

    # Càlcul millor caixa
    i = 0
    for result in results:
        [_,xmin,ymin,width,height,isLost] = BBS[i]
        points = []
        for r in result.boxes:
            # Descartar objectes
            if result.names[r.cls.item()] != objectNames[videoname]:
                continue
            [coords] = r.xyxy.tolist()
            # Afegir punt com a possible candidat
            points.append((r.conf.item(), yoloBoxToTopLeft(coords, width, height)))

        # Afegir overlapping ratio de la millor caixa
        overlappingRatios.append(selectOverlapRatio(points, xmin, ymin, width, height, isLost))
        i += 1

    # Escriptura de valors al fitxer
    with open(videoname + 'Internet.txt', 'w') as k:
        for o in overlappingRatios:
            k.write(str(o) + '\n')


model = YOLO("yolov8m.pt")

#per a un sol video rebut com a paràmetre
#videoname = sys.argv[1]
#runVideo(videoname, model)

#per a fer-los tots
for v in objectNames.keys():
    runVideo(v, model, method)
```