UAB
Universitat Autònoma de Barcelona

Repte 2: Seguiment d'objectes

Entregat per:

1639080 - Carlota Cortés

1636358 - Paula Macías

1636671 - Marc Puigbó

1636486 - Pol Riubrogent

Mètodes Avançats de Processament de Senyal, Imatge i Vídeo

1/11/2024

1. Introducció

La detecció i seguiment d'objectes és una tasca clau en el camp de la visió artificial. Els algoritmes d'aquest tipus són essencials per a nombroses aplicacions de visió per computador. Primer de tot, farem una petita part teòrica on analitzarem els algoritmes de seguiment d'objectes.

El codi del treball es troba a: https://github.com/NIU1636486/PSIV_Repte2

Si bé el seguiment visual d'un objecte en el temps i l'espai, per a nosaltres es una feina fàcil, és un procés complex que involucra múltiples subprocessos. Abans que puguem començar a rastrejar el moviment de l'objecte l'hem de detectar i distingir entre d'altres objectes.

La detecció d'objectes localitza la presència d'objectes en una imatge amb quadres delimitadors (localització) i indica els tipus d'objectes localitzats (classificació).

Existeixen diferents tipus de seguiment d'objectes, el que utilitzarem en aquest projecte es el seguiment de múltiples objectes; MOT rastreja simultàniament múltiples objectes d'interès en el vídeo. En el nostre cas detectarem múltiples cotxes a la vegada.

La majoria d'algoritmes de seguiment multiobjectes tenen les següents etapes:

ETAPA 1 – DETECCIÓ: Els objectius d'interès en aquesta fase s'anoten i ressalten. S'identifica els objectes que pertanyen a les classes designades.

ETAPA 2 – MOVIMENT: S'analitzen les deteccions i s'extreuen característiques d'aparença i d'interacció a través dels algorismes. Es fa servir un predictor de moviment per predir les posicions posteriors de cada objectiu rastrejat.

***ETAPA 3 – RECORDATORI*:** Les prediccions de característiques es fan servir per calcular les puntuacions de similitud entre les parelles de detecció. Aquestes puntuacions es fan servir després per associar deteccions que pertanyen al mateix objectiu. S'assignen ID a deteccions similars i s'apliquen ID diferents de deteccions que no formen part de parells. Alguns models de seguiment d'objectes es creen fent servir aquests passos per separat, mentre que altres combinen i usen els passos en conjunt.

1.1 Objectiu

L'objectiu principal del projecte és, a partir d'un vídeo de l'entrada d'enginyeria, saber quants automòbils es mouen en cada un dels carrils d'entrada i sortida. Específicament quants cotxes pugen i baixen en la durada del vídeo.

2. Metodologia

En aquest repte hem dividit l'algorisme en tres etapes:

1 - Detecció

2 - Seguiment

3 - Recompte

2.1 Detecció

Després del primer repte, ens vam familiaritzar amb YOLO i vam experimentar de primera mà els seus avantatges en termes de velocitat i precisió en la detecció d'objectes. Per això, vam optar per tornar a treballar amb YOLO. Inicialment, vam utilitzar la versió 8, amb la qual ja havíem treballat, però, com s'explica en les millores de rendiment, vam acabar canviant a la versió 5 per la seva major rapidesa, un factor clau per a nosaltres, ja que treballem amb vídeos de llarga durada i busquem detecció a temps real.

Per aquest projecte, vam utilitzar un model YOLO preentrenat sobre el dataset COCO, un conjunt de dades popular que inclou diversos tipus d'objectes, entre ells vehicles (cotxes, camions, etc.).

El codi llegeix els fotogrames del vídeo redimensionant-los a 1020x500 píxels, i en cada fotograma el model YOLO executa la predicció per detectar i identificar objectes. Després de cada predicció, es processen les "bounding boxes" de cada objecte detectat, amb les coordenades de les cantonades superior esquerra i inferior dreta, el nivell de confiança en la detecció i la classe de l'objecte.

Com que en el nostre cas només ens interessen les deteccions que corresponen a vehicles, vam descartar altres tipus d'objectes per centrar-nos específicament en el seguiment de cotxes en concret.

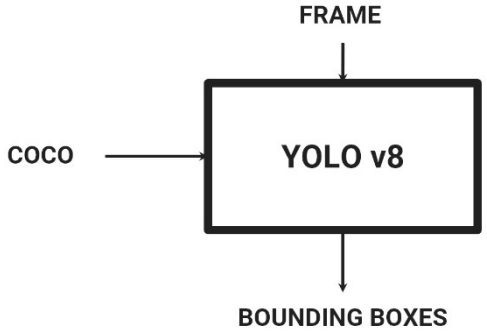


Figura 1. Diagrama del sistema de detecció

2.2 Seguiment

Un cop feta la detecció, el següent pas és el seguiment dels objectes detectats, és a dir, el "tracking". Aquesta fase consisteix a monitoritzar els vehicles detectats al llarg dels fotogrames consecutius del vídeo, assignant-los un identificador únic que permet distingir-los i rastrejar-ne la trajectòria en temps real.

Després de la detecció, tenim una llista de totes les "bounding boxes" i nivells de confiança associats, la qual passem al sistema de seguiment. Aquest Tracker és una classe dissenyada per gestionar múltiples objectes detectats al llarg de diversos fotogrames d'un vídeo.

Aquesta classe emmagatzema les coordenades centrals de cada objecte detectat en un diccionari, associant a cada un identificador únic. Per a la generació d'aquests identificadors, el tracker utilitza un comptador per generar IDs únics per als nous objectes detectats. Quan es crida el mètode d'actualització, se li passa la llista de bounding boxes, i a partir d'aquestes es calcula el punt central de cada objecte.

A partir del punt central el tracker comprova si ja existeix en el diccionari. A més, si la distància entre el punt nou i un punt central existent és inferior a 35 píxels, es considera que correspon al mateix objecte, i s'actualitza la seva posició. En cas contrari, si no es troba cap coincidència, es genera un nou ID per al punt central i s'afegeix al diccionari. Al final de cada actualització, el tracker neteja el diccionari de punts centrals per eliminar els objectes que ja no es detecten.

La sortida del tracker és una llista que inclou els identificadors i les coordenades de les "bounding boxes" actualitzades de cada vehicle, en altres paraules la posició actual de cada objecte. Aquestes dades es fan servir per comptabilitzar els vehicles que pugen o baixen.

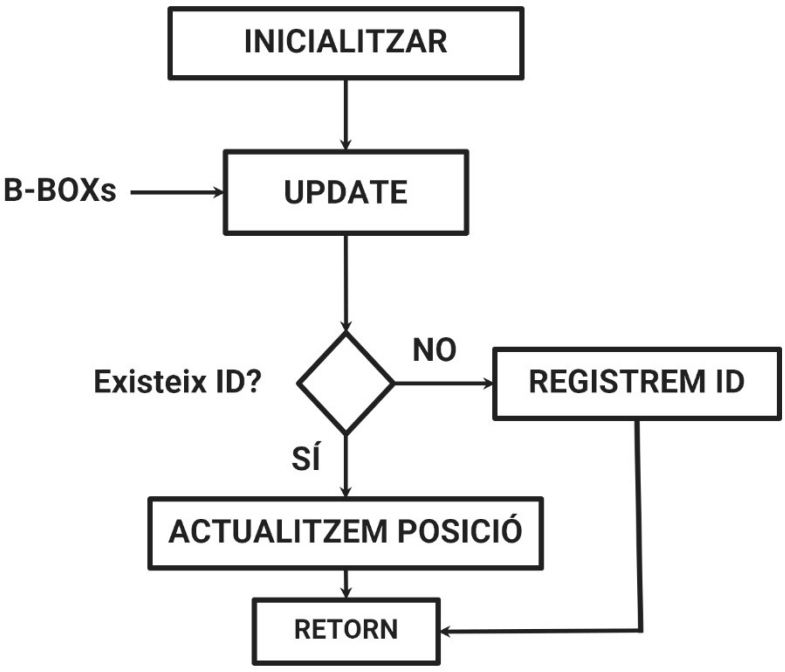


Figura 2. Diagrama del sistema de seguiment

2.3 Recompte

Per acabar, la fase de recompte de vehicles. Per comptabilitzar-los, vam definir dues línies horitzontal al fotograma:

Una línia vermella a y=400, que representa el límit inicial per detectar vehicles que estan baixant.

Una línia blava a y=450, que és la segona línia per detectar vehicles que es desplacen en direcció ascendent.

Llavors, les condicions de recompte es divideixen en dues categories: vehicles que baixen i vehicles que pugen. Per als que baixen, la condició és que el vehicle travessi primer la línia vermella i després la blava. De manera similar, si un vehicle travessa la línia blava abans que la vermella, es considera que està pujant. Quan es compleix alguna d'aquestes condicions, s'afegeix l'identificador del vehicle al comptador que li correspon. A més, per garantir precisió amb certa flexibilitat, s'aplica un marge de tolerància (offset) de 7 píxels a la posició.

Pel que fa a la visualització, es dibuixa un cercle al centre de cada vehicle detectat, i s'indica el seu identificador per facilitar el seguiment visual. A més, es mostren les línies vermella i blava al fotograma.

Finalment, el nombre de vehicles que pugen i baixen es mostra al vídeo analitzat.

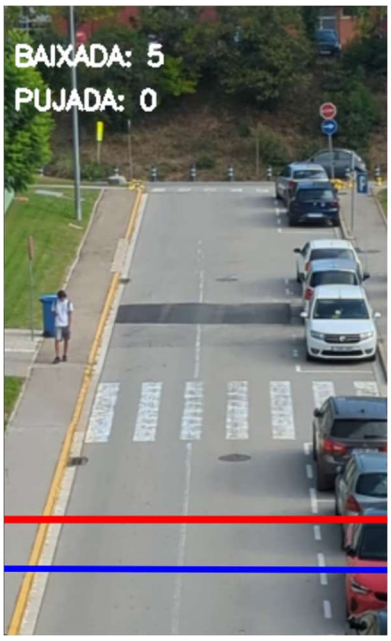


Figura 3. Captura de la sortida del sistema de recompte.

3. Disseny Experimental

3.1 Descripció dels datasets

Per realitzar les proves amb els diferents models que hem fet, tenim a disposició diferents vídeos de la rampa de la entrada d'enginyeria proporcionats pel professorat, a continuació detallem els que hem utilitzat:

Seqüencia short:

TEMPS	4:02 min
FPS	30 fps
COTXES PUJADA	6 cotxes
COTXES BAIXADA	2 cotxes

Seqüencia middle:

TEMPS	7:00 min
FPS	30 fps
COTXES PUJADA	5 cotxes
COTXES BAIXADA	7 cotxes

Seqüencia shadow :

TEMPS	12:03 min
FPS	30 fps
COTXES PUJADA	3 cotxes
COTXES BAIXADA	10 cotxes

Seqüencia long 1:

TEMPS	25:21 min
FPS	30 fps
COTXES PUJADA	8 cotxes
COTXES BAIXADA	24 cotxes

Seqüencia long 2:

TEMPS	1:04:54 h
FPS	30 fps
COTXES PUJADA	6 cotxes
COTXES BAIXADA	133 cotxes

3.2 Experiments i mètriques

En aquesta fase del projecte, vam explorar diverses estratègies per optimitzar el rendiment de detecció i seguiment de vehicles. A continuació es detallen els experiments realitzats.

Experiment 1: Model amb una única línia de recompte

En aquest model simplificat per a la detecció de vehicles, s'ha implementat un sistema de comptatge de moviments basat en una única línia de referència. Primer, es dibuixa una línia horitzontal en el frame del vídeo en una posició específica, a la coordenada y=400. Aquesta línia serveix com a llindar, de manera que els vehicles han de creuar-la perquè el sistema en registri el moviment de pujada o baixada.

El model detecta i registra la posició vertical y de cada cotxe quan es detecta el seu moviment en el frame. Aquest registre permet un seguiment continu del desplaçament de cada vehicle respecte a la línia de referència. Quan la coordenada y d'un cotxe es troba per sobre de la línia i es detecta el seu moviment cap avall, s'emmagatzema l'ID del vehicle i la seva coordenada y en un diccionari de "baixades". Si, en un frame posterior, es detecta que el mateix vehicle ha creuat la línia en direcció cap a la part inferior del frame, el model incrementarà el comptatge de baixades per a aquell vehicle.

Pel moviment de pujada, el procediment és invers: el sistema detecta quan un vehicle es mou cap amunt a través de la línia, registrant-ne així les pujades. Aquest sistema permet un seguiment més eficient dels vehicles amb menys complexitat en el model, ja que només cal una línia de referència per obtenir dades de moviments rellevants per al comptatge de vehicles.

Utilitzarem la aquest model en la resta d'experiments.

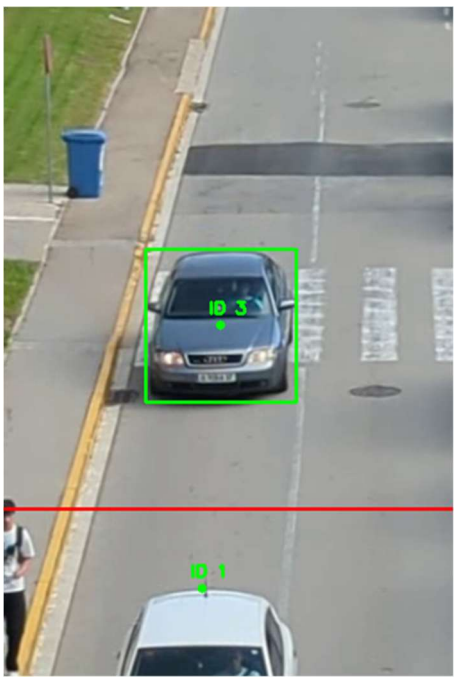


Figura 4. Captura de la sortida del sistema de recompte amb 1 línia.

Experiment 2: Ús de YOLOv5

Vam optar per canviar de YOLOv8 a YOLOv5, ja que, després de revisar les dades d'Ultralytics, vam observar que YOLOv5 destacava en termes de velocitat de detecció, una característica clau per al nostre projecte. Usem YOLOv5 per a la resta d'experiments.

Experiment 3: Reducció de la resolució

El segon pas per millorar el rendiment va ser disminuir la resolució de la imatge. La imatge original era de 960x540 píxels i la vam reduir a 480x270 per accelerar el procés. A més, vam eliminar el retall innecessari que s'aplicava a cada detecció amb YOLO, ja que afegia temps de processament sense aportar beneficis i alterava el format original del vídeo, que volíem mantenir intacte.

Experiment 4: Salt de fotogrames

Per millorar el rendiment, vam decidir saltar fotogrames, ja que no cal processar tots els fotogrames per aconseguir un seguiment efectiu. Inicialment, vam provar de saltar un de cada dos fotogrames, i després vam experimentar amb salts més grans com un de cada tres o quatre fotogrames. Aquesta tècnica va oferir un increment significatiu en l'eficiència.

Experiment 5: Escala de grisos per a YOLO

Una altra idea de millora va ser convertir els fotogrames a escala de grisos per accelerar el processament. Tanmateix, YOLO requereix imatges en RGB, per la qual cosa aquesta modificació no es va poder implementar.

Experiment 6: ROI

Aquest experiment va ser triat per poder accelerar la velocitat del processament i a més per detectar millor els cotxes. Abans d'aplicar el ROI els cotxes aparcats a la rampa els detectava i després ja no. A més la velocitat també ha augmentat.

La zona d'interès que em retallat no es tota la carretera, sinó únicament la meitat inferior d'aquesta.

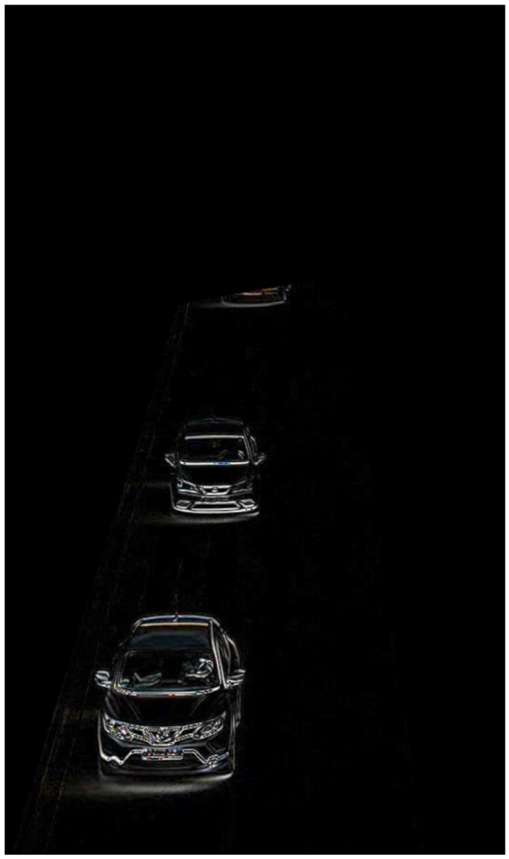
Figura 5. Regió d'interès (ROI)

Experiment 7: Detecció de moviment

Al mirar els vídeos vam poder veure que hi ha molts fragments els quals no hi ha pràcticament moviment de cotxes a la carretera. És per això que vam pensar que potser podríem buscar cotxes amb el YOLO només quan sabem que hi ha moviment, així estalviar moltes execucions del model, quan sabem que no hi haurà res a detectar.

Per fer-ho, utilitzant la ROI, simplement mirem la diferència absoluta entre el frame i l'anterior. Si la mitjana d'aquesta diferència és major que un threshold, apliquem el yolo i el tracker en aquest frame.

A continuació es pot veure un exemple de com es veu aquest frame de diferència, amb el frame següent al costat.



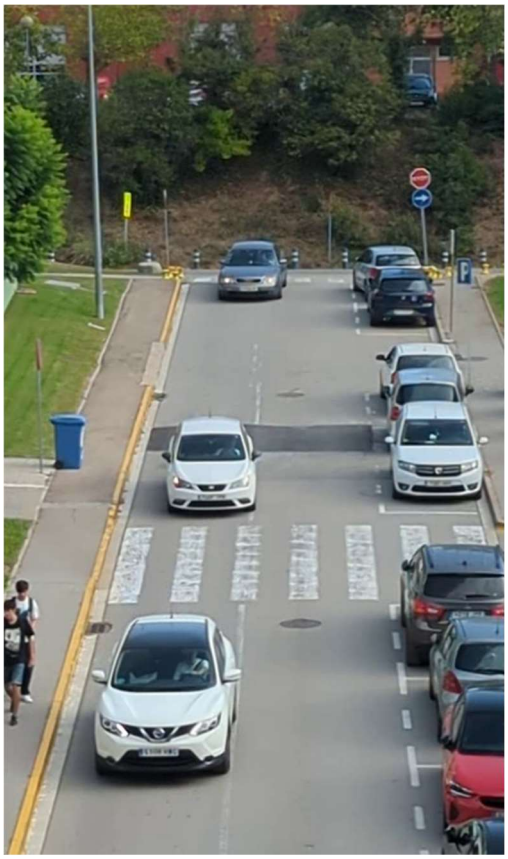


Figura 6 i 7. A la dreta, la diferència entre el frame actual i el frame anterior. A l'esquerra el frame actual.

4. Resultats

Per dur a terme una anàlisi exhaustiva del model implementat i determinar quin mètode és més eficient i obté els resultats més precisos, s'han realitzat diversos experiments acompanyats dels seus respectius resultats. Aquesta metodologia ha permès comparar el rendiment de les diferents optimitzacions i identificar aquelles que ofereixen la millor combinació de precisió i eficiència, amb l'objectiu de configurar el model final més òptim. A continuació, es presenten els detalls dels experiments i els resultats obtinguts. En tots els experiments s'ha utilitzat el vídeo més curt, el qual té 6 cotxes de pujada i 2 de baixada.

4.1 Experiment 1

Usant el model amb una sola línia per a la detecció obtenim els següents resultats amb el la seqüència short.

TEMPS REAL	4:02 min
TEMPS D'EXECUCIÓ	50:52min
FPS	2.39 fps
COTXES PUJADA	6 cotxes
COTXES BAIXADA	2 cotxes

Com veiem en la taula, obtenim els resultats de pujada i baixada correcte, però el temps d'execució és extremadament elevat en comparació amb el temps real del vídeo original. En els següents experiments ens centrarem en millorar el rendiment d'aquest model.

4.2 Experiment 2

Usant el model amb una línia juntament amb la versió 5 de YOLO, obtenim aquests resultats:

TEMPS REAL	4:02 min
TEMPS D'EXECUCIÓ	18:07min
FPS	8.81 fps
COTXES PUJADA	6 cotxes
COTXES BAIXADA	2 cotxes

Es pot apreciar que amb aquest experiment, els resultats segueixen sent correctes i el temps d'execució s'ha reduït notablement a 18min aproximadament, el qual ens indica que l'experiment ha sigut exitós. Per això els següents experiments es realitzen sobre aquest model, ja que presenta una velocitat més fàcil de tractar.

4.3 Experiment 3

Reduint notablement la resolució de la imatge però mantenint la proporció, obtenim un bon recompte de pujada i baixada.

TEMPS REAL	4:02 min
TEMPS D'EXECUCIÓ	15:05min
FPS	8.11 fps
COTXES PUJADA	6 cotxes
COTXES BAIXADA	2 cotxes

En la taula veiem que el temps d'execució ha disminuït en 3min aproximadament, amb 8.11 frames per segon. Es pot afirmar que aquest experiment és eficient.

4.4 Experiment 4

Utilitzar el salt de fotogrames, per reduir la quantitat de vegades que apliquem el YOLO i agilitzar la velocitat de la detecció. Obtenim els següents resultats:

TEMPS REAL	4:02 min
TEMPS D'EXECUCIÓ	7:22 min
FPS	16.74 fps
COTXES PUJADA	6 cotxes
COTXES BAIXADA	2 cotxes

Com veiem a la taula de resultats, el temps de execució es redueix considerablement al passar menys fotogrames.

4.6 Experiment 6

Arribem al penúltim experiment que hem fet, en aquest experiment pensàvem tenir més bons resultats que en els anteriors. Degut a que no ens centrem en el rendiment, sinó en millorar la detecció de cotxes i el seu seguiment. Al reduir el espai on el YOLO detecta els cotxes eliminem els cotxes aparcats i evitem que detecti aquests cotxes com falsos positius; reduint així els errors de detecció.

TEMPS REAL	4:02 min
TEMPS D'EXECUCIÓ	46:48 min
FPS	2.56 fps
COTXES PUJADA	6 cotxes
COTXES BAIXADA	1 cotxes

En aquest cas, ens va sorprendre els resultats, perquè veiem que aplicant aquest mètode sobre el model inicial, tarda molt més que en els altres experiments. I no millora gaire. Sobretot en el temps. Però en canvi ens assegura detectar els cotxes correctament, excepte en alguns vídeos que hi ha cotxes que baixen per la rampa per una zona on no hauria.

4.7 Experiment 7

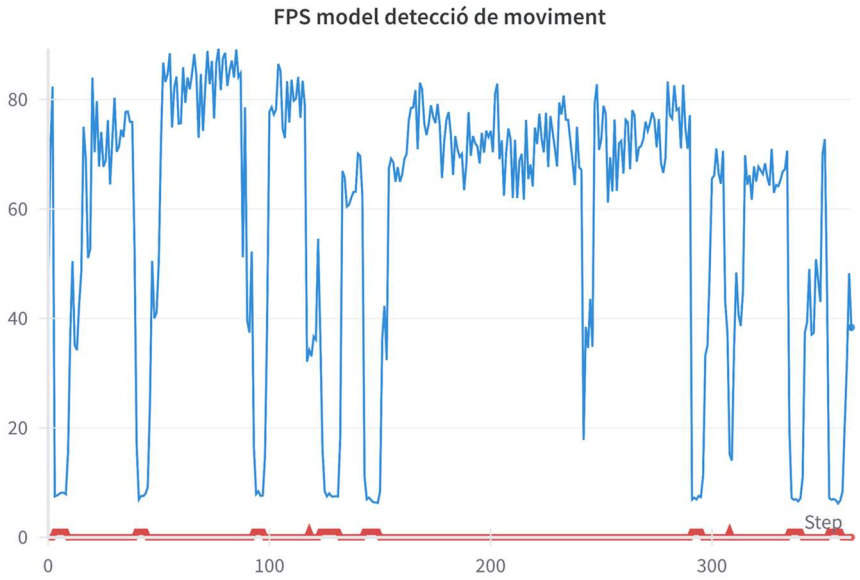


Figura 7. FPS al llarg de l'execució de l'experiment 7. En blau, els FPS, en vermell indica quan apliquem YOLO (1-SÍ, 0-NO)

TEMPS REAL	4:02 min
TEMPS D'EXECUCIÓ	3:04 min
FPS	58.36 fps
COTXES PUJADA	6 cotxes
COTXES BAIXADA	1 cotxes

Aquest experiment es el que millor resultats ens ha donat, no només el temps d'execució s'ha reduït, sinó que s'executa més ràpid que el propi vídeo. Podem observar en la Figura 7 quan s'activa el YOLO i quan no, veiem que hi ha moltes vegades que no s'activa perquè no es necessari, cosa que millora molt l'execució.

Generalment veiem que els FPS es mantenen en un nivell alt entre 50 i 80, indicant estabilitat, i que té caigudes quan s'aplica el YOLO, però es recupera ràpidament.

Veiem també que té un error de detecció com passava amb el ROI, que no detecta el cotxe que surt el vídeo que baixa per la rampa per un lloc on no hauria.

4.8 Resultats finals

Com que hem vist que tots els experiments han millorat el rendiment del nostre model, els hem combinat per crear el model final. A continuació es poden veure els resultats obtinguts amb el model final.

VIDEO	PUJADA (REALS)	PUJADA (PREDITS)	BAIXADA (REALS)	BAIXADA (PREDITS)	TEMPS (REAL)	TEMPS (EXECUCIÓ)
Short	**6**	**6**	**2**	**1**	**4:02 min**	**2:34 min**
Middle	**5**	**5**	**7**	**7**	**7:00 min**	**4:43 min**
Shadow	**3**	**3**	**10**	**10**	**12:03 min**	**6:55 min**
Long 1	**8**	**9**	**25**	**25**	**25:01 min**	**19:14 min**
Long 2	**6**	**6**	**133**	**131**	**1:04:54 h**	**48:03 min**

En aquesta taula podem veure els resultats de l'execució del model final on hem combinat les millores provades a tots els experiments que hem fet amb tots els vídeos que hem explicat a l'apartat de dataset.

Veiem en general que el temps d'execució es més ràpid que el temps real dels vídeos, i que detecta els cotxes correctament, a excepció dels vídeos on ho ha cotxes que pujen o baixen per una zona de la rampa que no està permès fer-ho.

5. Discussions i Conclusions

Podem concloure que hem aconseguit un sistema molt ràpid i eficient per contar cotxes en la rampa de l'Escola d'Enginyeria. Tot i perdre algun cotxe, els hem "sacrificat" per tal de fer el model més eficient. Igualment, el model inicial és capaç de contar tots els cotxes sense error, per tant hem aconseguit un model eficient i un model que els conta tots.

Millores

La millora principal que podem fer en el nostre model principal (el final) seria tenir en compte els casos especials els quals ens fan perdre algun cotxe, com per exemple què fer si un cotxe en comptes de baixar pel carril que toca, baixa pel carril on normalment hi ha cotxes aparcats (cas real del vídeo més curt). També podríem millorar el sistema de detecció de moviment per només aplicar yolo al fragment de la imatge on detectem moviment.

6. Referencies

[1] A. Singla. Accedido el 20 de octubre de 2024. [En línea]. Disponible: https://github.com/AarohiSingla/Detect--track-and-Count-using-YOLOv8

[2] "Faster video file FPS with cv2.VideoCapture and OpenCV - PyImageSearch". PyImageSearch.. [En línea]. Disponible: https://pyimagesearch.com/2017/02/06/faster-video-file-fps-with-cv2-videocapture-and-opencv/

[3] "Simple object tracking with OpenCV - PyImageSearch". PyImageSearch. [En línea]. Disponible: https://pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/