



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona



Universitat Politècnica de Catalunya

FACULTAT D'INFORMÀTICA DE BARCELONA

Informe Short Project VC

Autors: Jia Long Ji Qiu i Jiabo Wang

2022 - 2023 Q2
22 de juny de 2023

Índex

1	Tracking	2
1.1	Algorisme clàssic	2
1.1.1	Mètode	2
1.1.2	Experimentació	4
1.1.3	Conclusions	14
1.2	Algorisme basat en l'aprenentatge profund	15
1.2.1	Mètode	15
1.2.2	Experimentació	16
1.2.3	Conclusions	21
2	Reconeixement	22
2.1	Model propi	22
2.1.1	Model	22
2.1.2	Experimentació	23
2.1.3	Conclusions	27
2.2	Model estat de l'art de l'actualitat	28
3	Annex	29
3.1	Codi de l'algorisme clàssic (matlab)	29
3.2	Codi del model SiamRPN++ (python)	33
3.3	Codi del model de detecció per reconeixement (matlab)	35
	Referències	37

1 Tracking

El tracking és una tècnica que consisteix en detectar un objecte i seguir el seu moviment al llarg d'una seqüència d'imatges, on l'objecte pot tenir variacions en el moviment, la rotació o la escala. Tot i que el tracking es pot fer tant de múltiples objectes com d'un sol objecte, en aquesta secció ens centrarem en fer tracking d'un sol objecte.

1.1 Algorisme clàssic

1.1.1 Mètode

Un primer enfocament ha sigut utilitzar algorismes clàssics, on hem provat els algorismes d'extracció de característiques *Scale-Invariant Feature Transform* (SIFT) i *Speeded Up Robust Features* (SURF). Tots dos algorismes consisteixen en detectar i descriure *keypoints* a imatges i segueixen els mateixos principis de detecció invariant a la escala, rotació, translació, il·luminació... La diferència és que SURF va sorgir com una alternativa més ràpida que SIFT: mentre que SIFT es basa en la diferència gaussiana, SURF utilitza filtres d'aproximació.

Com que l'objectiu final és trobar una transformació que permeti mapejar els keypoints de l'objecte a seguir als keypoints de cada frame, no seria desgavellat concatenar els punts trobats amb ambdós algorismes i calcular la transformació a partir d'aquesta concatenació, obtenint d'aquesta manera una combinació SIFT+SURF, un altre mètode amb el qual també experimentarem.

Així doncs, l'esquelet inicial d'aquest algorisme serà el següent:

1. Inicialitzar la capsà contenidora amb els valors de la capsà contenidora del ground truth (o bé seleccionar un rectangle manualment).
2. Per un frame, aplicar l'algorisme d'extracció de característiques SIFT, SURF o SIFT+SURF tant al fragment corresponent a la capsà contenidora com al frame.
3. A partir dels *keypoints* trobats, s'ha de trobar una transformació geomètrica que faci que els punts del fragment retallat coincideixin amb els punts del frame (emparellament).
4. Definir una nova capsà contenidora que serà calculada amb la transformació geomètrica obtinguda aplicada als punts dels cantons del fragment retallat.
5. Com que aquesta capsà contenidora no serà ben bé un rectangle, s'ha de fer un processament dels punts obtinguts per obtenir un punt inicial $P = (x, y)$, l'amplada i l'alçada del rectangle.
6. Tornar al punt 2 amb aquesta nova capsà contenidora i passant al següent frame.

Com es pot observar a l'esquelet, el tracking s'està fent bassant-se en el frame anterior. En altres paraules, la detecció d'un objecte al frame n es fa a partir de la capsà contenidora calculada al frame n-1. Un avantatge d'aquesta metodologia és que els canvis en l'objecte, ja sigui per una variació en el punt de vista de la càmera o moviments locals del propi objecte, no seran massa sobtats ja que el moviment que hi ha entre un frame i un altre és mínim. Detectar el frame n = 100 a partir del fragment del frame n = 0, per exemple, costa més que fer-ho a partir del fragment del frame n = 99.

Tot i així, com que tant SIFT com SURF son molt propensos a errors quan es tracta de fer detecció, ja sigui per no haver pogut detectar suficients punts coincidents o bé perquè s'ha trobat una transformació adient, però la capsà contenidora obtinguda després d'aplicar-li la transformació no té res a veure amb el que hauria de ser, s'ha afegit a l'algorisme algunes comprovacions que el fa més robust a aquests errors:

- Pel primer cas, quan no es troben suficients punts coincidents, es manté l'última capsà contenidora vàlida.
- Pel segon cas, parlem exactament del punt 5 de l'esquelet del nostre algorisme. Moltes vegades, tot i trobar-ne una transformació geomètrica afí, la capsà contenidora després d'aplicar la transformació pot acabar sent, per exemple, un paral·lelogram finíssim on no es pot apreciar gens l'objecte que es vol detectar. En aquests casos, com que podem suposar que el canvi en una capsà contenidora d'un frame a un altre és mínim, ens basem en el ratio de les dues capses, de manera que si hi ha una variació d'un 10% en la mida, es descartarà.

Cal destacar que descartar noves capses contenidores o mantenir les anteriors implica una pèrdua d'informació que, si es produceix durant molts frames seguits, pot resultar en una detecció amb pèrdua total. Això és degut a què, com que l'objecte a seguir normalment és dinàmic a l'espai de detecció i la capsà contenidora es manté fixe durant uns quants frames, l'objecte a seguir pot acabar sortint-se d'aquesta capsà, i s'acabaria fent tracking d'un objecte totalment diferent a l'objectiu original.

1.1.2 Experimentació

Per tal de mesurar el rendiment d'aquest model, s'han fet proves en diferents datasets de diferents dificultats. Hem classificat els datasets en 4 dificultats diferents: fàcil, mitjà, difícil i molt difícil. Pels tres primers dificultats, hem escollit 3 datasets respectivament i només hem escollit un dataset classificat com a molt difícil.

Datasets fàcils:

Els datasets fàcils els farem servir per comprovar la bondat dels tres mètodes que hem plantejat: SIFT, SURF i la combinació dels dos SIFT+SURF. Això servirà per determinar quins mètodes valen la pena i descartarem aquells que no donin bons resultats.

- Nivell 1, Boat:

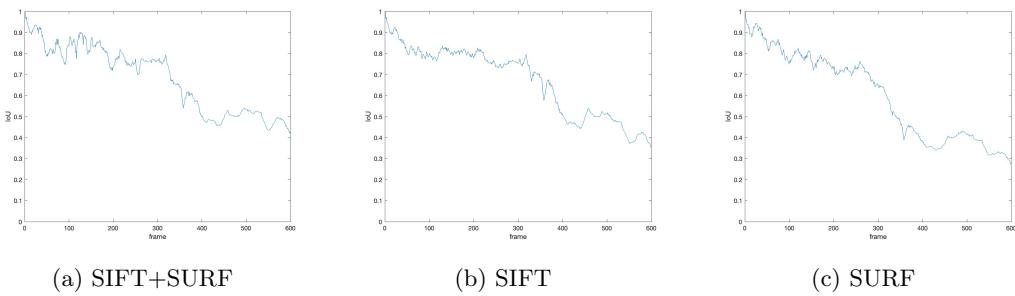


Figura 1: Evolució de l'IoU durant la seqüència de frames del dataset Boat

Amb el dataset de Boat es pot observar que més o menys es manté la capsula contenidora a on hi és l'objecte, sense perdre-ho mai. El problema que hi ha amb aquest dataset és que la capsula contenidora obtinguda s'engrandeix massa, fent que l'IoU no sigui massa bona a mesura que la seqüència avança.

Cal esmentar que la capsula s'engrandeix degut a què a cada iteració s'agafen punts que es troben al voltant de l'objecte, fent que la transformació afí escali cada vegada més. Aquest fet es pot observar a la Figura 2.



Figura 2: Exemple d'escalat en la seqüència Boat

- Nivell 2, Drone3:

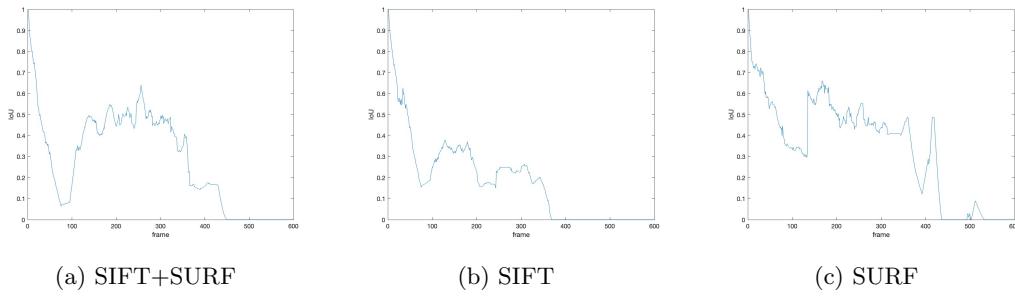


Figura 3: Evolució de l'IoU durant la seqüència de frames del dataset Drone3

En el cas del dataset de Drone3, la capsula contenidora deixa de ser capaç de fer un seguiment de l'objecte. És molt possible que sigui degut a la rotació que fa l'objecte: és quan el dron fa un gir molt ràpid quan l'algorisme deixa de detectar suficients punts, i acaba donant més importància a la gespa.



Figura 4: Exemple d'escalat en la seqüència Drone3

- Nivell 3, Bike:

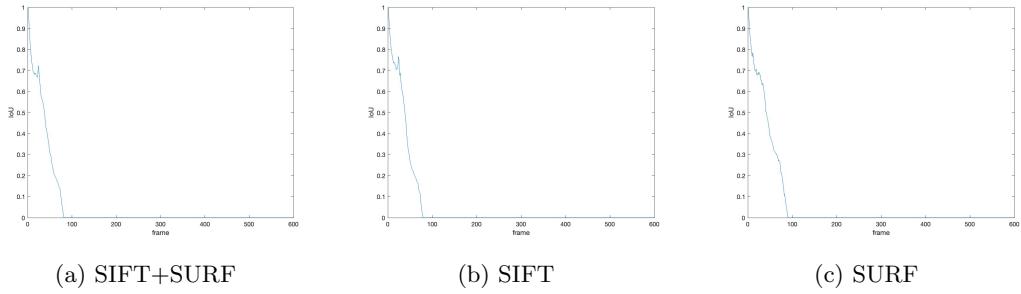


Figura 5: Evolució de l'IoU durant la seqüència de frames del dataset Bike

En aquest últim dataset de la dificultat fàcil es pot observar que el seguiment que es fa és molt pobre, ja que la pèrdua total de l'IoU es produeix pràcticament al començament de la seqüència. En aquest cas, com es pot apreciar a la Figura 6, sembla que sigui degut a que els *keypoints* que troba al paisatge acaben sent més rellevants que el propi objecte que ha de seguir.



Figura 6: Frame de la seqüència Bike, on s'ha acabat donant més importància al paisatge

Com s'ha pogut observar a les Figures 1, 3 i 5, els resultats obtinguts amb els algorismes SIFT, SURF i SIFT+SURF son molt semblants en quant a proporció de l'IoU.

Una excepció ha sigut en el dataset de Drone3, a la Figura 3, on es pot veure que el SURF ha aconseguit un millor rendiment. Tot i així, aquesta diferència no és cap fet conclusiu, ja que aquests algorismes son aleatoris i pot haver sigut casualitat.

D'altra banda, però, sembla que la combinació SIFT+SURF dona resultats més consistents, molt probablement perquè amb la unió de *keypoints* es dona més ènfasi a l'objecte a seguir, fent que sigui més fàcil de detectar. Per tant, per la resta de proves ens cenyirem a aquesta combinació.

Datasets normals:

- Nivell 1, Alladin:

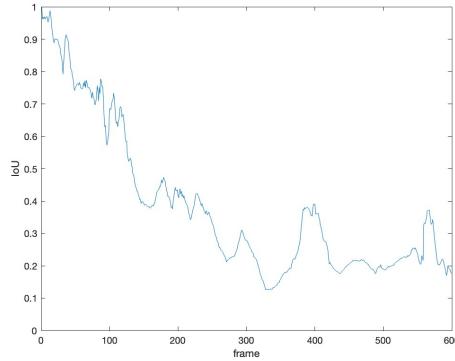


Figura 7: Evolució de l'IoU durant la seqüència de frames del dataset Alladin

En el primer dataset de dificultat normal tenim el d'Alladin, amb el qual tot i que la proporció IoU deixa força a desitjar, ja que baixa del 50% des del primer quart de la seqüència, els resultats son força bons.

En aquesta seqüència, el problema ha estat que, a més de detectar l'objectiu, s'ha donat importància també a altres elements de l'escenari, com ara un calaix, fet que ha provocat un augment a la mida de la capsula contenidora, com podem observar a la Figura 8.



Figura 8: Frame de la seqüència d'Alladin, on a més de detectar-se l'objecte a seguir, es detecta també un altre element de l'escenari

- Nivell 2, Badminton1:

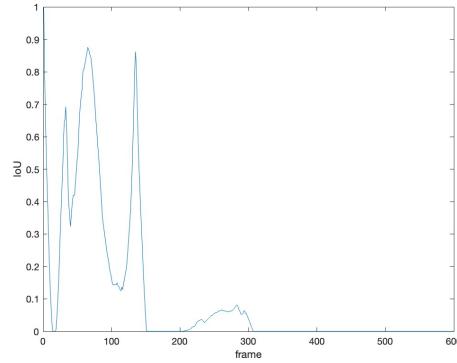


Figura 9: Evolució de l'IoU durant la seqüència de frames del dataset Badminton1

El dataset de Badminton1 és el que més problemes ha donat dins la dificultat normal. En aquesta seqüència sembla ser que no hi ha suficient contrast entre el jugador i la pista de bàdminton, ja que l'algorisme gairebé mai trobava suficients *keypoints*. Això va provocar que la capsà contenidora es mantingués fixa durant massa frames, ja que no hi havia manera de calcular una nova, fins a arribar a un punt en el que ja no hi havia manera de recuperar el seguiment a l'objecte.



Figura 10: Frame de la seqüència Badminton, on s'observa que la capsà contenidora gairebé s'ha mogut de la posició inicial

- Nivell 3, Puppies1:

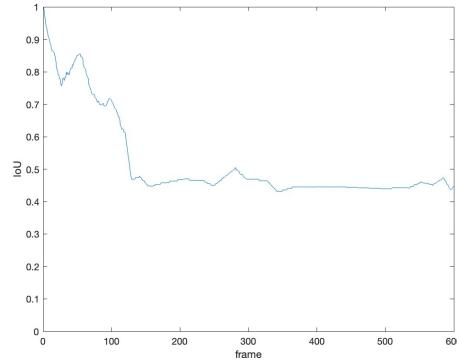


Figura 11: Evolució de l'IoU durant la seqüència de frames del dataset Puppies1

Per últim tenim la seqüència Puppies1. Aquesta seqüència és relativament fàcil, ja que l'objecte gairebé no es mou. La principal dificultat resideix en poder mantenir una bona proporció IoU, evitant que el model acabés dirigint la seva atenció a les reixetes, i no confondre's massa amb els girs de l'objecte.

En el cas d'aquest model, tot i haver mantingut una proporció força bona, realment no és tan bona si tenim en consideració la falta de moviment de l'objecte a seguir. A més a més, aquest moviment, tot i haver sigut mínim, ha provocat que s'engrandeixi la capsula contenidora i, com ja s'ha comentat abans, corregir aquest augment en el tamany és molt complicat amb aquest algorisme.



Figura 12: Frame de la seqüència Puppies1, on la capsula contenidora calculada és massa gran

Datasets difícils:

- Nivell 1, Helicopter:

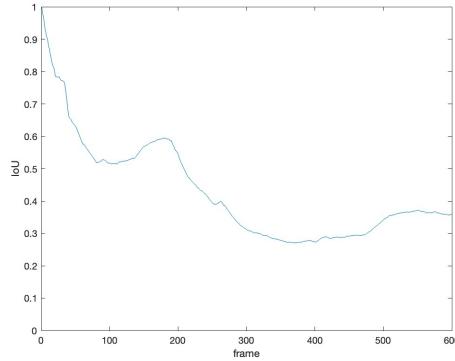


Figura 13: Evolució de l'IoU durant la seqüència de frames del dataset Helicopter

El seguiment al dataset Helicopter ha estat fàcil perquè el moviment és molt lent i l'objecte molt gran. El problema, però, ha estat en el gir que realitza, el qual fa que l'objecte canviï totalment de dimensions (la capsula contenidora real comença sent un rectangle, després passa a ser un quadrat i, finalment, torna a ser un rectangle). El nostre model no ha sigut capaç d'adaptar-se a aquests canvis en la mida, i això ha resultat en una proporció IoU massa pobre. La diferència en mides al final de la seqüència es pot apreciar a la Figura 14



Figura 14: Diferència de mida entre la capsula contenidora real i la calculada

- Nivell 2, DriftCar1:

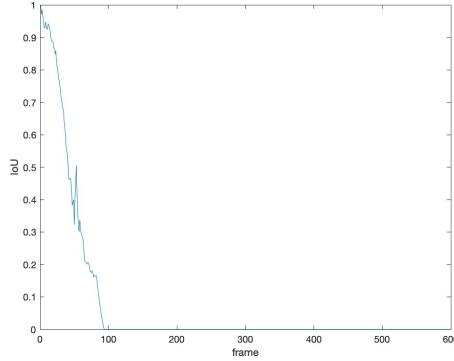


Figura 15: Evolució de l’IoU durant la seqüència de frames del dataset DriftCar1

El dataset DriftCar1 ha donat problemes degut als girs que fa l’objecte justament al començament de la seqüència. Això ha provocat que la capsula contenidora es quedés fixa seguint una carretera, la qual impossibilitava trobar nous *keypoints* i, per tant, va deixar d’actualitzar-se completament.



Figura 16: Moment en el qual el gir de l’objecte a seguir de la seqüència DriftCar1 fa que la capsula contenidora es perdi

- Nivell 3, ISS:

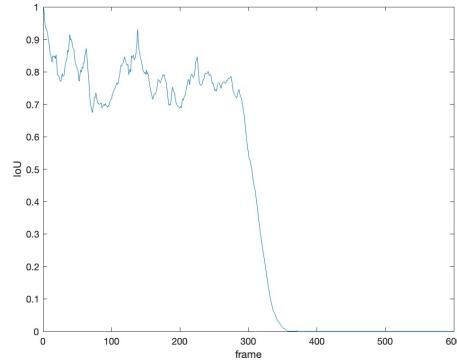


Figura 17: Evolució de l'IoU durant la seqüència de frames del dataset ISS

El tracking de la seqüència ISS ha anat força bé, però ha començat a tenir problemes un cop l'objecte a seguir ha començat a fer un gir. Semblant al cas del dataset DriftCar1, aquest gir ha provocat que l'algorisme detectés més *keypoints* al fons i es quedés fixe durant la resta de la seqüència. També esmentar que l'objecte a seguir desapareix parcialment i, tot i no afectar-nos perquè la capsula contenidora ja s'havia perdut, també pot suposar un problema bastant important.



Figura 18: Moment en el que l'objecte a seguir de la seqüència ISS gira i es perd el seguiment

Datasets molt difícils:

- ZebraFish:

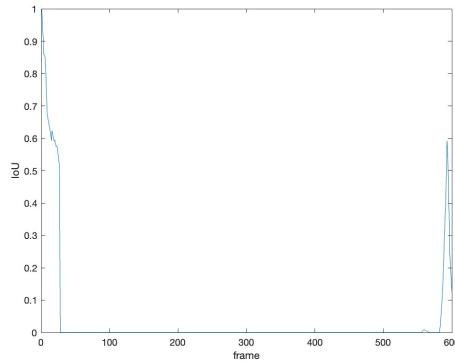


Figura 19: Evolució de l'IoU durant la seqüència de frames del dataset ZebraFish

La dificultat en el dataset ZebraFish resideix principalment en la oclusió. Només començar la seqüència, es pot observar a la figura 20 com un peix cirugià tapa completament durant uns quants frames al peix pallasso que s'havia de seguir.

Una oclusió pot provocar que hi hagi la possibilitat de què els *keypoints* de l'objecte a seguir s'emparellin amb uns punts d'un altre objecte del frame que no té res a veure amb l'objectiu original, com ha passat en aquest cas. A més a més, com que l'escenari en aquest cas és totalment fixe, ha sigut impossible recuperar el seguiment del peix pallasso.



Figura 20: Moment en el qual un peix cirugià tapa el peix a seguir i es detecta un altre objecte

1.1.3 Conclusions

Com hem pogut veure al llarg de la experimentació amb el model basat en un algorisme clàssic, combinant SIFT i SURF, és un model que no s'adapta massa bé a datasets que tenen un grau de complexitat elevat.

Els problemes principals que ens hem trobat han sigut la falta de contrast, la oclusió, el canvi d'escala i els girs locals:

- La falta de contrast entre l'objecte a seguir i l'escenari pot provocar que es detectin massa objectes, fent que la capsula contenidora s'engrandeixi i faci seguiment d'altres objectes.
- La oclusió pot provocar que o bé no es trobin suficients *keypoints* (en aquells casos on l'objecte a seguir és tapat per una superfície gairebé llisa, monòtona, sense suficients característiques), o bé que el seguiment es faci basant-se en la entitat que ha provocat la oclusió, fent que es faci un seguiment d'un objecte que no té res a veure amb l'objectiu original.
- Els canvis d'escala tenen un impacte sobretot a la proporció de l'IoU. Els algoritmes com SIFT o SURF, tot i ser veritat que son invariables a l'escalat, en un contexte de tracking és molt diferent: detectar objectes més grans és fàcil, però això provoca que cada vegada s'engrandeixi més la capsula contenidora, i minimitzar aquesta mida és una tasca molt complicada.
- Els girs dels objectes fan que siguin molt difícils de seguir durant una seqüència, ja que la forma de l'objecte pot canviar dràsticament. És molt més fàcil fer el seguiment d'un objecte que només es va traslladant, mantenint la mateixa forma, que fer el seguiment d'un objecte que es mou i gira a la vegada.

Podem conoure, doncs, que els algoritmes clàssics que hem fet servir no son prou robustos, i que calen models més complexos que puguin fer front als problemes esmentats.

1.2 Algorisme basat en l'aprenentatge profund

1.2.1 Mètode

Un altre enfocament del problema és tractar el seguiment amb tècniques d'aprenentatge profund. Cercant per internet, hem trobat un model que dona un rendiment molt prometedor, SiamRPN++. Aquest model és una extensió de SiamRPN, que tal com es pot deduir del seu nom, combina una xarxa siamesa amb una xarxa de proposta de regions d'interès (RPN) per predir la posició de l'objecte.

Per una banda, l'arquitectura d'una xarxa siamesa consisteix en dues xarxes convolucionals idèntiques i serveix per calcular la similitud entre dues imatges. S'extrauen les característiques d'ambdues imatges mitjançant les dues xarxes convolucionals i posteriorment es calcula una ponderació de semblança a partir d'aquestes característiques.

Per l'altra banda, una xarxa de proposta de regions d'interès és utilitzat per calcular les regions on té més possibilitats d'estar ubicat l'objecte.

Així doncs, el model fa servir la xarxa de proposta de regions d'interès (RPN) per calcular el conjunt de regions amb més possibilitats de contenir l'objecte a seguir i cada regió és comparat amb la plantilla de l'objecte al frame anterior mitjançant la xarxa siamesa i se selecciona la regió que obtingui la puntuació de semblança més alta.

La gran diferència entre SiamRPN++ i SiamRPN és una millora en l'extracció de característiques de les imatges, incorpora una arquitectura siamesa més profunda i complexa que permet capturar característiques amb més precisió.

L'esquelet de l'algorisme de seguiment amb el model SiamRPN++ consisteix en els següents passos:

1. Inicialització de la bounding box: en aquest primer pas, s'ha de proporcionar la ubicació inicial de l'objecte a seguir.
2. Generació de propostes de regions d'interès: mitjançant la xarxa RPN, es generen un conjunt de propostes de regions d'interès.
3. Extracció de característiques: s'agafa com a plantilla la regió inicial on residia l'objecte i juntament amb cada una de les regions d'interès, s'extrauen les seves característiques mitjançant la xarxa siamesa.
4. Càlcul de la similitud: per cada parella (plantilla - regió d'interès) es calcula una puntuació de similitut i es selecciona la regió més similar.
5. Estimació de la bounding box: mitjançant un model de regressió es calcula les coordenades de la bounding box en la regió més similar.
6. Actualització: la bounding box calculat anteriorment és utilitzat per l'estimació en el frame següent i es torna al pas 2.

Aquest algorisme sempre agafa com a plantilla les coordenades de la bounding box del frame anterior (excepte del primer frame), i es compara les regions d'interès actuals amb aquesta plantilla.

La implementació d'aquest algoritme es troba en el framework de codi obert de MMTracking [1], s'ha inferit el model ja entrenat de SiamRPN++. El codi de la inferència es troba en la secció 3.2, on hi ha una funció anomenada `get_iou` per calcular el valor d'intersecció sobre unió de dos

bounding box i una altra funció anomenada inferència que se li ha de proporcionar el nom del dataset a fer el tracking i retorna una llista dels valors d'intersecció sobre unió de la bounding box estimada amb el de ground truth en totes els frames.

1.2.2 Experimentació

Per mesurar el rendiment d'aquest altre model de tracking utilitzarem la mateixa metodologia que hem fet servir amb el model anterior.

Datasets fàcils:

- Nivell 1, Boat:

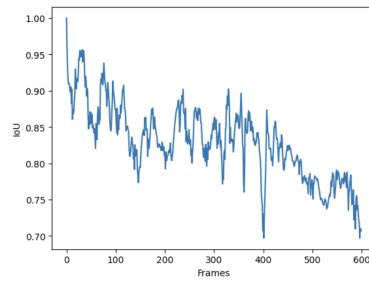


Figura 21: Evolució del valor de Intersecció sobre Unió (IoU) durant la seqüència de frames del dataset Boat

Tal com es pot veure en la figura 21, no es perd l'objecte en cap moment del vídeo i els valors d'intersecció sobre unió amb les capses contenidores del ground truth són en tot moment superiors a 0,7. La dificultat d'aquest conjunt de dades es troba en els canvis d'orientació de la càmera i el model és capaç de solventar-ho amb èxit.

- Nivell 2, Drone3:

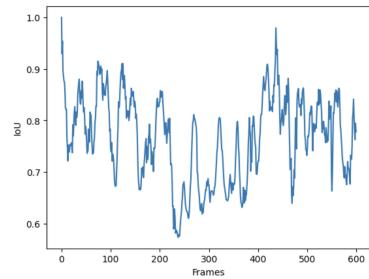


Figura 22: Evolució del valor de Intersecció sobre Unió (IoU) durant la seqüència de frames del dataset Drone3

Per aquest conjunt de dades, a més dels canvis d'orientacions de la càmera, també fa zoom de l'objecte a seguir. Tot i això, se continua obtenint bons resultats: no es perd l'objecte en cap moment i la ràtio d'intersecció sobre unió és superior a 0,6.

- Nivell 3, Bike:

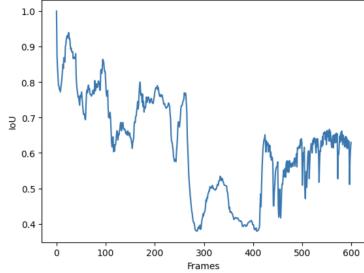


Figura 23: Evolució del valor de Intersecció sobre Unió (IoU) durant la seqüència de frames del dataset Bike

La dificultat d'aquest conjunt de dades es troba en el canvi de fons de l'objecte a seguir, en aquest cas la persona. Al principi el fons és la gespa (color verd, frames 0 - 200 aprox.), però després la persona es mou i el fons canvia a ser de color blau (cel, a partir del frame 300). El model és capaç de seguir la persona, tot i que al moment en què es fa la transició del fons (entre els fotogrames 300 i 400) l'encert de les coordenades de la bounding disminueix fins a 0,4. Son resultats acceptables, no perd l'objecte a seguir i manté les ràtios de la IoU per sobre de 0,4, com es pot observar en la figura 23.

En general, aquest model ha pogut seguir els objectes exitosament en aquests datasets que considerem fàcils.

Datasets normals:

- Nivell 1, Alladin:

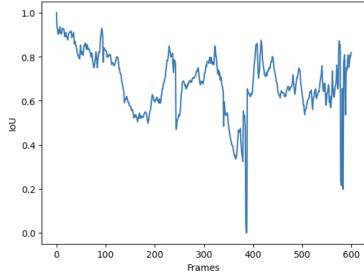


Figura 24: Evolució del valor de Intersecció sobre Unió (IoU) durant la seqüència de frames del dataset Alladin

La part complicada d'aquest dataset es troba en la il·luminació del vídeo. La persona a seguir “canvia” de color per la il·luminació que rep. Això suposa una dificultat a l'hora d'extreure les característiques de la persona, però el model sap solucionar-ho. Tot i que perd la persona en una seqüència de 2 frames a causa del desenfocament de la persona en el moment que es posa a córrer, és capaç de tornar a detectar la persona i continuar amb seguiment. Per tant, no es considera que ha perdut a l'objecte. A part d'aquests dos frames, el model és capaç de seguir persona i obtenir unes ràtios de IoU per sobre de 0,3.

- Nivell 2, Badminton1:

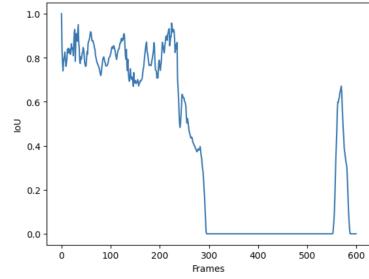


Figura 25: Evolució del valor de Intersecció sobre Unió (IoU) durant la seqüència de frames del dataset Badminton1

En aquest cas, el model no és capaç de seguir al jugador a partir del frame 300. Això és degut al fet que es creua el jugador a seguir amb un company d'equip i com que es vesteixen iguals i tenen una aparença molt semblant, el model no és capaç de distingir-los i comença a seguir el company. Hi ha un pic al final, perquè els dos jugadors es tornen a creuar. No s'ha pogut completar el seguiment del jugador.

- Nivell 3, Puppies1:

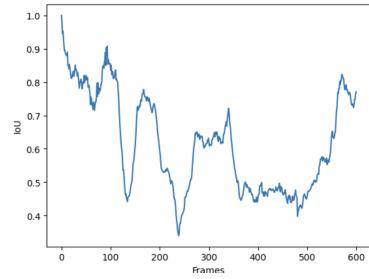


Figura 26: Evolució del valor de Intersecció sobre Unió (IoU) durant la seqüència de frames del dataset Puppies1

La dificultat d'aquest cas és que l'objecte a seguir té un color semblant al de fons, i per tant pot ocasionar problemes en la determinació de les coordenades de la bounding box. Això no

obstant, el model dona uns resultats bastant bons (veure figura 26: no perd l'objecte en cap moment i manté les ràtios de la IoU per sobre de 0,3).

Mitjançant aquests datasets de dificultat mitjana, s'ha trobat una debilitat del model: no és capaç de tractar de manera eficient les oclusions i no té una capacitat de reidentificació amb l'objecte inicial, tal com s'ha pogut veure en el dataset del Badminton1. Quan perd l'objecte a seguir, el substitueix per l'objecte més semblant.

Datasets difícils:

- Nivell 1, Helicopter:

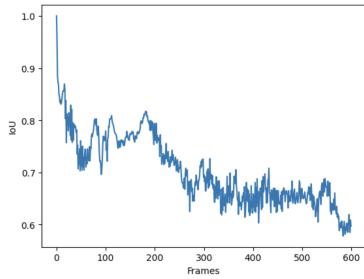


Figura 27: Evolució del valor de Intersecció sobre Unió (IoU) durant la seqüència de frames del dataset Helicopter

El model no ha tingut gaires dificultats per tractar aquest dataset, on la dificultat més es troba que l'objecte a seguir té un color semblant al color de fons. No perd l'objecte i la IoU es manté per sobre de 0,6.

- Nivell 2, DriftCar1:

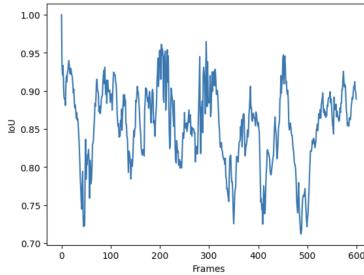


Figura 28: Evolució del valor de Intersecció sobre Unió (IoU) durant la seqüència de frames del dataset DriftCar1

També ha pogut seguir amb èxit l'objecte en aquest dataset, amb ràtios de la IoU sempre per sobre de 0,7. El dataset presenta canvis d'orientació i augmentos de la càmera i l'objecte a seguir es mou ràpidament durant tota la seqüència. Aquests obstacles han sigut solucionats de manera exitosa per part del model.

- Nivell 3, ISS:

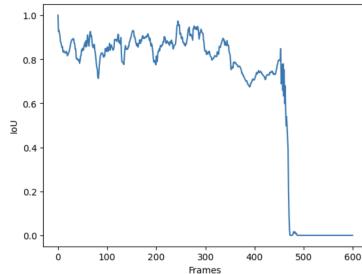


Figura 29: Evolució del valor de Intersecció sobre Unió (IoU) durant la seqüència de frames del dataset ISS

Finalment, en aquest dataset l'objecte a seguir desapareix als instants finals. En aquest cas, el model no és capaç d'adonar-se i comença a seguir els objectes que considera que són els més semblants a l'objecte perdut. Això no obstant, abans de la desaparició és capaç de seguir l'objecte amb una IoU per sobre de 0,7.

El model, per una banda, és robust als canvis d'orientacions i augmentos de la càmera, també pot detectar objectes que es mouen ràpidament. Per una altra banda, quan l'objecte a seguir surt a la imatge (desapareix), no és capaç d'adonar-se d'aquest fet i comença a seguir els objectes més semblants a l'objecte desaparegut.

Datasets molt difícils:

- ZebraFish:

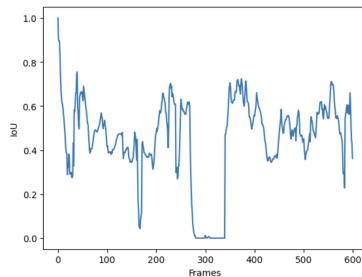


Figura 30: Evolució del valor de Intersecció sobre Unió (IoU) durant la seqüència de frames del dataset ZebraFish

El repte d'aquest dataset és que hi ha un peix exactament igual que el peix a seguir, a més es troben en espais molt propers. A l'inici quan no es creuen, és capaç de seguir-lo, però un cop creuen, el model no és capaç de distingir el peix que ha de seguir i comença a seguir l'equivocat. Hi ha moments que pot tornar a seguir, però el torna a perdre. En definitiva, no és capaç de distingir aquests dos peixos i els confon constantment.

1.2.3 Conclusions

Els resultats d'aquest model són bastant prometedors, és capaç de fer el tracking dels objectes en quasi tots els datasets. És robust als canvis d'orientació de la càmera, com també als augmentos i disminucions de mida de l'objecte. A més, pot fer front als canvis d'il·luminació del dataset i pot captar la presència dels objectes que es mouen ràpidament. La gran limitació d'aquest model és que no tracta bé les oclusions i desaparicions, és a dir, si l'objecte a seguir desapareix i torna a aparèixer posteriorment, en el moment de la desaparició el model canvia d'objectiu i comença a seguir un altre objecte (el que considera més semblant al perdut) i en el moment que torna a aparèixer, el model ja no el pot reidentificar. El mateix passa quan hi ha oclusions parcials de l'objecte que el fan difícil de detectar. Finalment, l'altra dificultat que presenta aquest model és quan hi ha objectes molt semblants a l'objecte a seguir i es troben molt propers, el model es confon i no pot distingir-los, ja que presenten característiques quasi idèntiques.

Una millora que es podria afegir és la millora de la xarxa siamesa per calcular la semblança de dos imatges i poder distingir dos objectes molt semblants. També es podria introduir un mecanisme que distingir els casos en els que l'objecte surt de la imatge i aturi el model fins que torna a aparèixer l'objecte desaparegut.

2 Reconeixement

2.1 Model propi

La idea en aquesta segona part del projecte és implementar un model que permeti fer la detecció de l'objecte amb tècniques de reconeixement.

2.1.1 Model

S'ha entrenat el model de tal manera que és capaç de reconèixer quines característiques pertanyen a l'objecte a seguir i quines pertanyen al fons. A partir d'això, per cada imatge s'extreuen les seves característiques SURF i el model decideix si és una característica que pertany al fons o a l'objecte. Un cop es troben totes les característiques de l'objecte, es calcula la bounding box.

El model implementat segueix els següents passos:

1. Fase d'entrenament: s'extreuen els 100 keypoints SURF més forts per cada frame del dataset, i s'etiqueten segons si pertanyen a l'objecte (si es troben dins de les coordenades del bounding box del ground truth) o pertanyen al fons de l'escena. S'extreuen les característiques d'aquests punts clau i són proporcionades a una xarxa neuronal perquè aprengui a classificar-los. L'entrenament d'aquesta xarxa neuronal s'ha fet amb l'aplicació Classification Learner que proporciona Matlab.
2. Fase d'inferència: per cada frame es calculen tots els punts claus i s'extreuen les seves característiques. Tot seguit, mitjançant el classificador se sap si pertanyen o no a l'objecte. Finalment, es troben mitjançant els punts clau que pertanyen a l'objecte es calcula les coordenades de la bounding box.

2.1.2 Experimentació

Per mesurar el rendiment d'aquest model, hem fet servir 4 datasets diferents: el de la moto (fàcil), Alladin (normal), Helicopter (difícil) i ZebraFish (molt difícil).

1. Moto (fàcil): a continuació (a la figura 31) es mostra la matriu de confusió que s'ha obtingut en la fase d'entrenament. Les característiques que s'han etiquetat amb 0 són les que pertanyen al fons i amb 1 els que pertanyen a l'objecte. El percentatge d'encert del model és molt alt. Per aquest dataset, el model és capaç de seguir l'objecte sense pèrdua. Els valors de la IoU són bastant elevats com es pot observar en la figura 32, valors que es troben al voltant de 0,5, excepte un cas que és molt baix a causa de l'oclosió. Durant una oclusió no es pot aconseguir molts punts clau de l'objecte i, per tant, no pot construir una bounding box amb coordenades precises.

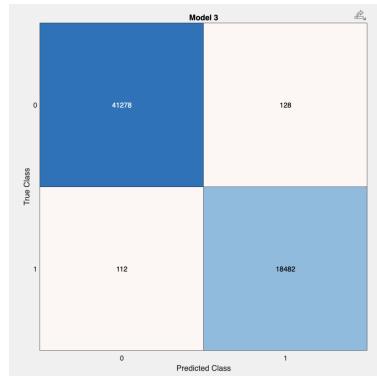


Figura 31: Matriu de confusió entre les característiques que pertanyen al fons (0) i les que pertanyen a l'objecte (1) del dataset Moto

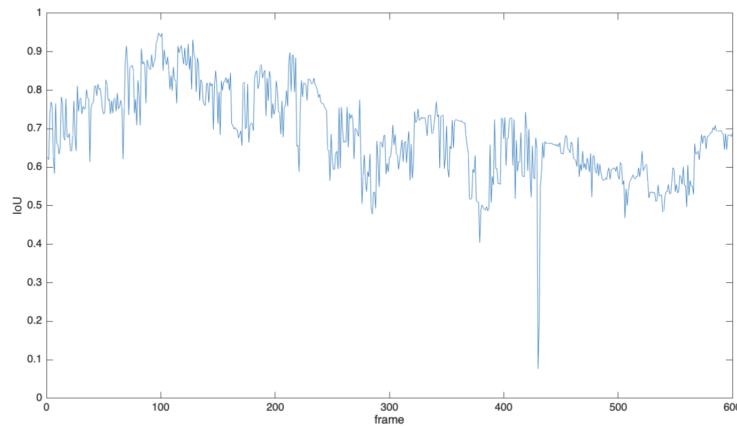


Figura 32: Evolució del valor de Intersecció sobre Unió (IoU) durant la seqüència de frames del dataset Moto

2. Alladin (normal): a la figura 33 es mostra la matriu de confusió obtinguda en l'entrenament de la classificació de les característiques dels punts clau de les imatges del dataset Alladin, el percentatge d'encert és del 95% en la validació. Posteriorment, s'ha calculat les coordenades de les bounding box i s'han aconseguit les ràtios d'Intersecció sobre Unió amb les bounding box del ground truth (Figura 34). Per aquest conjunt de dades, la precisió ha baixat perquè no s'han pogut aconseguir prou punts clau de l'objecte a seguir, no obstant això, no ha perdut l'objecte a seguir.

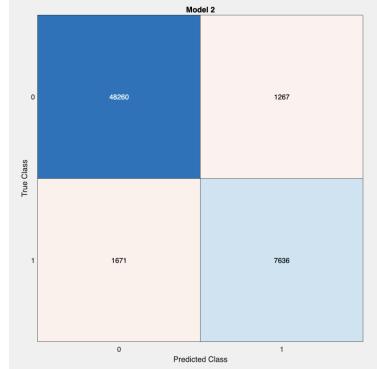


Figura 33: Matriu de confusió entre les característiques que pertanyen al fons (0) i les que pertanyen a l'objecte (1) del dataset Alladin

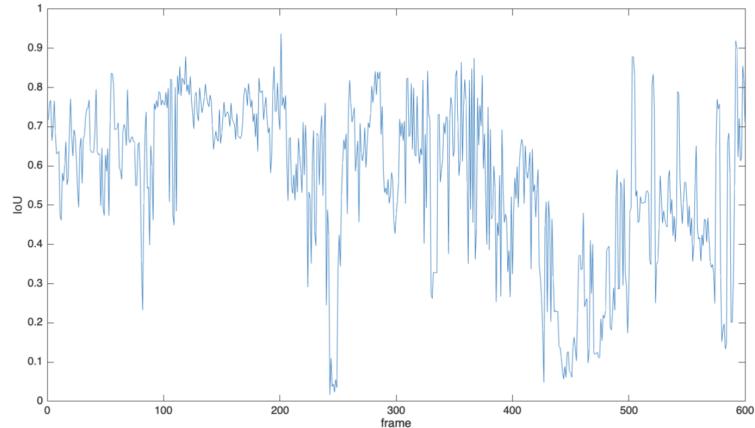


Figura 34: Evolució del valor de Intersecció sobre Unió (IoU) durant la seqüència de frames del dataset Alladin

3. Helicopter (difícil): com en els casos anteriors, al principi s'ha fet un entrenament del classificador (veure figura 35), on s'obté un 95% d'encert en la validació. El classificador torna a oferir resultats molt bons. Posteriorment, s'han calculat els bounding box i s'ha comparat amb els reals assolint les ràtios d'Intersecció sobre Unió. Encara que aquest dataset és considerat d'una dificultat avançada pels autors de la seva creació, el nostre model és capaç de seguir l'objecte sense ocasionar cap pèrdua en cap moment i aconseguir unes ràtios d'IoU molt elevat (per sobre de 0,75), com es pot veure en la figura 36. Això és degut al fet que l'objecte es mou lentament i apareix sempre en la seqüència del vídeo, per tant, és fàcil detectar una gran quantitat de punts clau. Doncs, la dificultat d'aquest dataset resideix a què l'objecte té un color semblant al del fons, però això no suposa cap dificultat pel nostre model, ja que és capaç de distingir el fons i l'objecte d'una manera molt eficient amb el classificador.

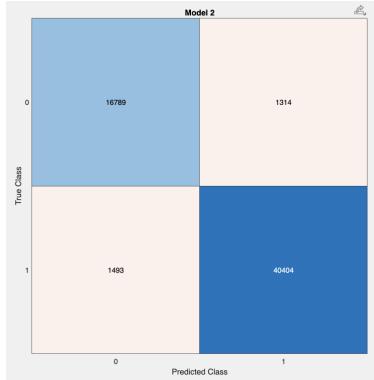


Figura 35: Matriu de confusió entre les característiques que pertanyen al fons (0) i les que pertanyen a l'objecte (1) del dataset Helicopter

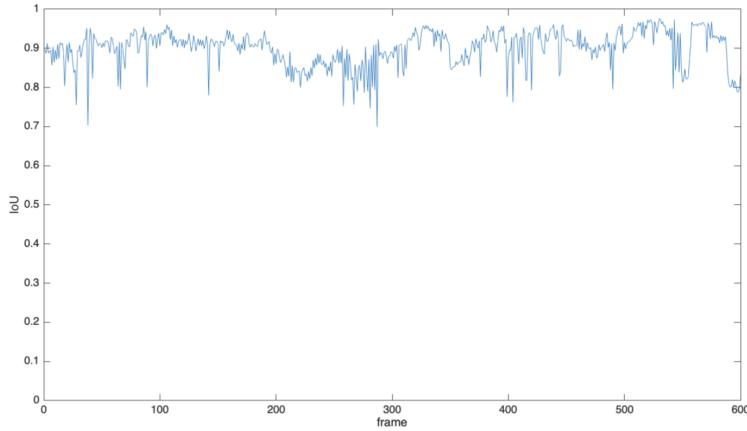


Figura 36: Evolució del valor de Intersecció sobre Unió (IoU) durant la seqüència de frames del dataset Helicopter

4. ZebraFish (molt difícil): per aquest últim cas, el classificador dona bons resultats, però no el seguiment. Només és capaç de seguir el peix durant 35 frames perquè hi ha oclusions constants que no deixen obtenir els punts clau. Sense punts clau, el model no és capaç de construir la bounding box i per tant, es perd l'objecte.

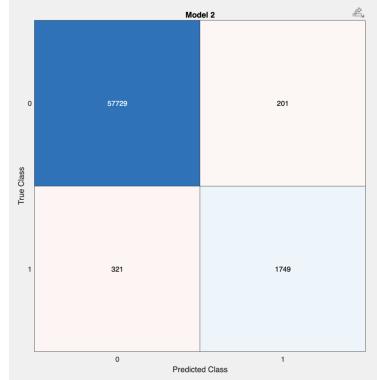


Figura 37: Matriu de confusió entre les característiques que pertanyen al fons (0) i les que pertanyen a l'objecte (1) del dataset ZebraFish

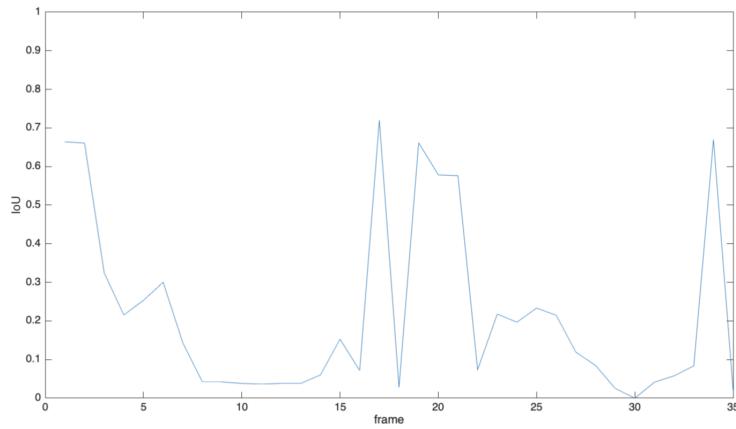


Figura 38: Evolució del valor de Intersecció sobre Unió (IoU) durant la seqüència de frames del dataset ZebraFish

2.1.3 Conclusions

El model que hem desenvolupat, el classificador dona uns resultats molt prometedors en la tasca de classificar punts clau que pertanyen a l'objecte o al fons. Això no obstant, quan hi ha pocs punts clau de l'objecte en la imatge, no pot estimar les coordenades de la bounding box amb precisió i, per tant, les ràtios d'intersecció sobre unió amb la bounding box real és molt petita, no és un model robust a oclusions.

Per una altra banda, considerant que el classificador dona molt bons resultats, es podria utilitzar el model per classificar si en una imatge apareix o no l'objecte a seguir. És a dir, una tasca merament de classificació sense incloure l'estimació de les coordenades de la bounding box.

2.2 Model estat de l'art de l'actualitat

R-CNN [2]: model de xarxa neuronal convolucional basat en regions. El concepte clau d'aquest model, com indica en el seu nom, són les propostes de regió, utilitzades per localitzar els objectes dins d'una imatge. És un dels models més emprats per a tasques de detecció d'objectes en la actualitat. Aquest model consisteix en els següents passos per a la detecció:

1. Extracció de regions d'interès: mitjançant l'algorisme de cerca selectiva¹ s'obté una llista d'aproximadament 2000 regions d'interès possibles. Les regions d'interès tenen mides diferents i l'arquitectura de la xarxa neuronal CNN (Alexnet²) accepta entrades de la mida 227 x 227 píxels, per la qual cosa es necessari redimensionar-ho a aquesta mida.
2. Extracció de característiques: totes les regions d'interès són sotmeses a la xarxa neuronal convolucional per calcular un vector de característiques de dimensió 4096.
3. Classificació de les regions: mitjançant un model de SVM³ preentrenat i usant el vector de característiques s'obté la probabilitat que té aquesta regió d'interès de pertànyer a una classe concreta. I després d'obtenir totes les probabilitats, s'aplica la supressió no màxima (NMS) per a cada regió i es descarta els que obtenen un valor alt d'Intersecció sobre Unió (IoU) amb una regió que té una probabilitat més alta de pertànyer a la classe .
4. Predicció del quadre delimitador (bounding box) mitjançant regressió: per precisar les coordenades del quadre delimitador de l'objecte, s'entrena un model de regressió per a això. Cada quadre delimitador està compost per quatre paràmetres (x, y, w, h) , (x, y) representen les coordenades de la part esquerra avall del quadre delimitador, w representa l'amplada i h representa l'alçada. L'objectiu és trobar uns pesos w tals que multiplicats a les coordenades del quadre delimitador siguin semblants a les coordenades del veritable quadre delimitador.

R-CNN és un model basat en la generació de regions d'interès que posteriorment són tractats per un classificador, seleccionant la regió amb més probabilitat de pertànyer a l'objecte que es vol seguir. La tasca de reconeixement és efectuat pel classificador. La gran diferència d'aquest model comparat amb el que hem desenvolupat nosaltres és que extreu les característiques de les imatges mitjançant xarxes neuronals convolucionals, mentre que nosaltres obtenim les característiques SURF de les imatges. A més, utilitza un model de regressió per calcular les bounding boxes, aconseguint resultats molt més precisos. S'han fet extensions d'aquest model, com per exemple Faster R-CNN, que fa servir xarxes neuronals per la generació de regions d'interès i és molt més eficient i ràpid.

No hem implementat aquest model, ja que hem emprat més temps en entendre-ho que utilitzar-ho i hi ha multituds d'exemples d'implementacions que es pot trobar a internet. El que podem estar segurs, és que aquest model obtindrà uns resultats i un rendiment superior que el model que hem construit nosaltres.

¹Algorisme dissenyat per a ser ràpid en la generació de regions d'interès i que es basa en el càcul de l'agrupació jeràrquica de regions similars en funció del color, la textura, la mida i forma.

²Xarxa neuronal convolucional amb 8 capes de profunditat.

³Algorisme d'aprenentatge supervisat entrenat per predicir la classe d'una mostra

3 Annex

3.1 Codi de l'algorisme clàssic (matlab)

```
close all
objects = [ "Boat" , "Drone3" , "Bike" , "Alladin" , "Badminton1" , "Puppies1" , "Helicopter" ,
debug_mode = false ;

Execute_Algorithm( objects , "SS" , debug_mode );

function Execute_Algorithm( objects , method , debug_mode )
    n_objects = size( objects , 2 );
    for n = 1:n_objects
        object = objects(n);
        BB = importdata("./TinyTLP/" + object + "/groundtruth_rect.txt");
        Idir = dir("./TinyTLP/" + object + "/img/*.jpg");
        initBB = [BB(1,2) , BB(1,3) , BB(1,4) , BB(1,5)];
        nf = size( Idir );
        filename = horzcat(Idir(1).folder , '/' , Idir(1).name);
        frame = imread(filename);
        frame_size = [ size(frame , 1) , size(frame , 2) ];
        currentBB = initBB;
        current_area = (initBB(3) * initBB(4)) / 2;
        first_frame_with_total_loss = -1;
        frames_with_good_ratio = 0;
        IoU_ratios = zeros(nf);
        figure
        for i = 1:nf
            objectToTrack = imcrop(frame , currentBB);
            filename = horzcat(Idir(i).folder , '/' , Idir(i).name);
            frame = imread(filename);

            if (method == "SS")
                [ m_kp_obj , m_kp_esc ] = Get_Matched_Points(objectToTrack , frame);
            elseif (method == "SIFT")
                [ m_kp_obj , m_kp_esc ] = Get_Matched_SIFT_Points(objectToTrack , frame);
            else
                [ m_kp_obj , m_kp_esc ] = Get_Matched_SURF_Points(objectToTrack , frame);
            end

            [T, ~, status] = estimateGeometricTransform2D(m_kp_obj , m_kp_esc , "affine")
            if (status == 0)
                [f, c] = size(rgb2gray(objectToTrack));
                points = [1,1; 1,f; c,f; c,1];
                transformed_points = transformPointsForward(T, points);
                transformed_rectangle = Calculate_Rectangle(transformed_points);
```

```

if (Is_Valid_Rectangle(transformed_rectangle , frame_size))
    new_area = (transformed_rectangle(3) * transformed_rectangle(4)) /
    area_ratio = new_area / current_area;
    if (area_ratio < 0.9)
        if (debug_mode)
            fprintf('Ratio too small! Ignoring new bounding box, ratio = %f\n', area_ratio);
        end
    elseif (area_ratio > 1.1)
        if (debug_mode)
            fprintf('Ratio too big! Ignoring new bounding box, ratio = %f\n', area_ratio);
        end
    else
        currentBB = transformed_rectangle;
        current_area = new_area;
    end
end
else
    if (status == 1)
        if (debug_mode)
            fprintf('matchedPoints1 and matchedPoints2 inputs do not contain enough points\n');
        end
    else
        if (debug_mode)
            fprintf('Not enough inliers found\n');
        end
    end
end
imshow(frame);
hold on
rectangle('Position' , currentBB , 'EdgeColor' , 'red');
rectangle('Position' , BB(i , 2:5) , 'EdgeColor' , 'yellow');
hold off
drawnow

overlap_ratio = bboxOverlapRatio(currentBB , BB(i , 2:5));
IoU_ratios(i) = overlap_ratio;
if (overlap_ratio == 0 && first_frame_with_total_loss == -1)
    first_frame_with_total_loss = i;
end
if (overlap_ratio > 0.5)
    frames_with_good_ratio = frames_with_good_ratio + 1;
end
end
figure
plot(1:600 , IoU_ratios)

```

```

    ylim([0 , 1])
    xlabel('frame')
    ylabel('IoU');
    fprintf('First frame with total loss: %d\n', first_frame_with_total_loss);
    fprintf('Number of frames with good overlap ratio: %d\n', frames_with_good_ratio);
end
end

function [ rectangle ] = Calculate_Rectangle( points )
    [ upper_left , upper_right , bottom_left , bottom_right ] = Get_Corner_Points( points );

    y = ( upper_left(2) + upper_right(2) ) / 2;
    x = ( upper_left(1) + bottom_left(1) ) / 2;
    w = ( upper_right(1) + bottom_right(1) ) / 2 - x;
    h = ( bottom_left(2) + bottom_right(2) ) / 2 - y;

    rectangle = [x y w h];
end

function valid = Is_Valid_Rectangle( rectangle , frame_size )
    x = rectangle(1);
    y = rectangle(2);
    w = rectangle(3);
    h = rectangle(4);
    frame_height = frame_size(1);
    frame_width = frame_size(2);
    valid = (x >= 1) && (y >= 1) && (x + w - 1 <= frame_width) && (y + h - 1 <= frame_height);
end

function [ upper_left , upper_right , bottom_left , bottom_right ] = Get_Corner_Points( points )
    sorted_points = sortrows(points , 1);
    leftmost = sorted_points(1:2 , :);
    rightmost = sorted_points(3:4 , :);
    sorted_left = sortrows(leftmost , 2);
    sorted_right = sortrows(rightmost , 2);

    upper_left = sorted_left(1 , :);
    bottom_left = sorted_left(2 , :);
    upper_right = sorted_right(1 , :);
    bottom_right = sorted_right(2 , :);
end

function [ m_kp_obj , m_kp_esc ] = Get_Matched_SIFT_Points( objectToTrack , frame )
    im_obj = rgb2gray(objectToTrack);
    im_esc = rgb2gray(frame);
    kp_obj = detectSIFTFeatures(im_obj);

```

```

kp_obj = selectStrongest(kp_obj, 500);
kp_esc = detectSIFTFeatures(im_esc);
kp_esc = selectStrongest(kp_esc, 500);
[feat_obj, kp_obj] = extractFeatures(im_obj, kp_obj, 'Method', 'SIFT');
[feat_esc, kp_esc] = extractFeatures(im_esc, kp_esc, 'Method', 'SIFT');
pairs = matchFeatures(feat_obj, feat_esc, 'MatchThreshold', 10);
m_kp_obj = kp_obj(pairs(:,1),:);
m_kp_esc = kp_esc(pairs(:,2),:);
end

function [m_kp_obj, m_kp_esc] = Get_Matched_SURF_Points(objectToTrack, frame)
im_obj = rgb2gray(objectToTrack);
im_esc = rgb2gray(frame);
kp_obj = detectSURFFeatures(im_obj);
kp_obj = selectStrongest(kp_obj, 500);
kp_esc = detectSURFFeatures(im_esc);
kp_esc = selectStrongest(kp_esc, 500);
[feat_obj, kp_obj] = extractFeatures(im_obj, kp_obj, 'Method', 'SURF');
[feat_esc, kp_esc] = extractFeatures(im_esc, kp_esc, 'Method', 'SURF');
pairs = matchFeatures(feat_obj, feat_esc, 'MatchThreshold', 10);
m_kp_obj = kp_obj(pairs(:,1),:);
m_kp_esc = kp_esc(pairs(:,2),:);
end

function [m_kp_obj, m_kp_esc] = Get_Matched_Points(objectToTrack, frame)
[sift_m_kp_obj, sift_m_kp_esc] = Get_Matched_SIFT_Points(objectToTrack, frame);
[surf_m_kp_obj, surf_m_kp_esc] = Get_Matched_SURF_Points(objectToTrack, frame);

m_kp_obj = [sift_m_kp_obj.Location; surf_m_kp_obj.Location];
m_kp_esc = [sift_m_kp_esc.Location; surf_m_kp_esc.Location];
end

```

3.2 Codi del model SiamRPN++ (python)

```

import mmcv
import tempfile
import os
import numpy as np
from mmtrack.apis import inference_sot, init_model

def get_iou(ground_truth, pred):
    # coordinates of the area of intersection.
    ix1 = np.maximum(ground_truth[0], pred[0])
    iy1 = np.maximum(ground_truth[1], pred[1])
    ix2 = np.minimum(ground_truth[2], pred[2])
    iy2 = np.minimum(ground_truth[3], pred[3])

    # Intersection height and width.
    i_height = np.maximum(iy2 - iy1 + 1, np.array(0.))
    i_width = np.maximum(ix2 - ix1 + 1, np.array(0.))

    area_of_intersection = i_height * i_width

    # Ground Truth dimensions.
    gt_height = ground_truth[3] - ground_truth[1] + 1
    gt_width = ground_truth[2] - ground_truth[0] + 1

    # Prediction dimensions.
    pd_height = pred[3] - pred[1] + 1
    pd_width = pred[2] - pred[0] + 1

    area_of_union = gt_height * gt_width + pd_height * pd_width - area_of_intersection

    iou = area_of_intersection / area_of_union

    return iou

def inferencia(dataset):
    os.system('ffmpeg -f image2 -r 60 -i ./' + dataset +
              '/img/%05d.jpg -vcodec mpeg4 -y ./' + dataset + '.mp4')
    sot_config = './mmtracking/configs/sot/siamese_rpn/siamese_rpn_r50_20e_lasot.py'
    sot_checkpoint = './siamese_rpn_r50_1x_lasot_20211203_151612-da4b3c66.pth'
    input_video = './' + dataset + '.mp4'
    bb_file = './' + dataset + '/groundtruth_rect.txt'
    with open(bb_file) as iostream:
        content = iostream.read()
    bbox = dict()
    for line in content.split("\n"):

```

```

frame, xmin, ymin, xmax, ymax, isLost = line.split(",")
bbox[frame] = dict([( 'x1', float(xmin)), ('y1', float(ymin)),
                    ('x2', float(xmax)), ('y2', float(ymax))])
# build the model from a config file and a checkpoint file
sot_model = init_model(sot_config, sot_checkpoint, device='cpu')
init_bbox = [bbox['1'][ 'x1'], bbox['1'][ 'y1'],
bbox['1'][ 'x1'] + bbox['1'][ 'x2'], bbox['1'][ 'y1'] + bbox['1'][ 'y2']]
imgs = mmcv.VideoReader(input_video)
prog_bar = mmcv.ProgressBar(len(imgs))
out_dir = tempfile.TemporaryDirectory()
out_path = out_dir.name
error = list()
for i, img in enumerate(imgs):
    result = inference_sot(sot_model, img, init_bbox, frame_id=i)
#ground-truth bbox
    bbox_gt = np.array([bbox[str(i+1)][ 'x1'], bbox[str(i+1)][ 'y1'],
bbox[str(i+1)][ 'x1'] + bbox[str(i+1)][ 'x2'], bbox[str(i+1)][ 'y1'] + bbox[str(i+1)][ 'y2'],
+ bbox[str(i+1)][ 'y1']], dtype=np.float32)
    bbox_pr = np.array([result[ 'track_bboxes'][0], result[ 'track_bboxes'][1],
result[ 'track_bboxes'][2], result[ 'track_bboxes'][3]], dtype=np.float32)
#predicted bbox
#intersection over union
    io = get_iou(bbox_gt, bbox_pr)
    error.append(io)
    sot_model.show_result(
        img,
        result,
        wait_time=int(1000. / imgs.fps),
        out_file=f'{out_path}/{i:06d}.jpg')
    prog_bar.update()
output = './' + dataset + 'inference.mp4'
print(f'\nmaking the output video at {output} with a FPS of {imgs.fps}')
mmcv.frames2video(out_path, output, fps=imgs.fps, fourcc='mp4v')
out_dir.cleanup()
return error

```

3.3 Codi del model de detecció per reconeixement (matlab)

```

BB = importdata( './MotorcycleChase/groundtruth_rect.txt' );
Idir = dir( './MotorcycleChase/img/*.jpg' );

descriptores = [];

%% Entrenament del classificador

nf = size(Idir); % nombre total de fitxers imatges
for i = 1:min(600, nf)
    boundbox = BB(i,:);
    filename = horzcat(Idir(i).folder, '/', Idir(i).name);
    image = rgb2gray(imread(filename));

    puntosClave = selectStrongest(detectSURFFeatures(image), 100);

    puntosClaveFondo = puntosClave( puntosClave.Location(:,1) < boundbox(2) |
    ...
        puntosClave.Location(:,1) > (boundbox(2) + boundbox(4)) | ...
        puntosClave.Location(:,2) < boundbox(3) | ...
        puntosClave.Location(:,2) > (boundbox(3) + boundbox(5)) );

    puntosClaveObjeto = puntosClave( puntosClave.Location(:,1) >= boundbox(2)
& ...
        puntosClave.Location(:,1) <= (boundbox(2) + boundbox(4)) & ...
        puntosClave.Location(:,2) >= boundbox(3) & ...
        puntosClave.Location(:,2) <= (boundbox(3) + boundbox(5)) );

    [descriptoresFondo, ~] = extractFeatures(image, puntosClaveFondo);
    [descriptoresObjeto, ~] = extractFeatures(image, puntosClaveObjeto);
    [f, ~] = size(descriptoresFondo);
    descriptoresFondo = [descriptoresFondo, zeros(1,f)'];

    %% [o, ~] = size(descriptoresObjeto);
    %% descriptoresObjeto = [descriptoresObjeto, ones(1,o)'];

    descriptores = [descriptores; descriptoresFondo; descriptoresObjeto];
end

iou = [];
figure
%hold on % mantenim sempre oberta la mateixa figura
nf = size(Idir); % nombre total de fitxers imatges
for i = 1:min(600, nf)

```

```

filename = horzcat(Idir(i).folder , '/' , Idir(i).name);
image = rgb2gray(imread(filename));

puntos = selectStrongest(detectSURFFeatures(image) , 100);
[ features , ~ ] = extractFeatures(image , puntos);
ynew = Model.predictFcn(features);
puntosObj = puntos(ynew == 1, :);

[x, y, w, h] = calcularBB(puntosObj.Location , puntosObj.Scale);

overlapRatio = bboxOverlapRatio([x, y, w, h],BB(i,2:5));

iou = [iou, overlapRatio];

imshow(image);
hold on
rectangle('Position',[x, y, w, h], 'EdgeColor', 'red');
drawnow
plot(puntosObj);
hold off
end
plot(1:600, iou)
ylim([0, 1])
xlabel('frame')
ylabel('IoU');

function [x, y, w, h] = calcularBB(puntos, scale)
[x2, idx] = min(puntos(:,1));
if scale(idx) > 4
    x = x2-3*scale(idx);
else
    x = x2-6*scale(idx);
end
[y2, idx] = min(puntos(:,2));
if scale(idx) > 4
    y = y2-3*scale(idx);
else
    y = y2-6*scale(idx);
end
[w, idx] = max(puntos(:,1));
w = abs(x - w);
[h, idx] = max(puntos(:,2));
h = abs(y - h);
end

```

Referències

- [1] *MMTracking*. URL: <https://github.com/open-mmlab/mmtracking>
- [2] Girshick, R., Donahue, J., Darrell, T. y Malik, J. (Octubre de 2014). *Rich feature hierarchies for accurate object detection and semantic segmentation*. URL: <https://arxiv.org/abs/1311.2524>