TESE DE DOUTORAMENTO

DESIGN OF REAL-TIME MULTIPLE OBJECT VISUAL DETECTION AND TRACKING SYSTEMS

Mauro Fernández Sanjurjo

ESCOLA DE DOUTORAMENTO INTERNACIONAL DA UNIVERSIDADE DE SANTIAGO DE COMPOSTELA
PROGRAMA DE DOUTORAMENTO EN INVESTIGACIÓN EN TECNOLOXÍAS DA INFORMACIÓN

SANTIAGO DE COMPOSTELA

2020

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

DECLARACIÓN DO AUTOR DA TESE

Design of Real-Time Multiple Object Visual Detection and Tracking Systems

Don Mauro Fernández Sanjurjo

Presento a miña tese, seguindo o procedemento adecuado ao Regulamento, e declaro que:

1. *A tese abarca os resultados da elaboración do meu traballo.*
2. *No seu caso, na tese faise referencia ás colaboracións que tivo este traballo.*
3. *A tese é a versión definitiva presentada para a súa defensa e coincide coa versión enviada en formato electrónico.*
4. *Confirmo que a tese non incorre en ningún tipo de plaxio de outros autores nin de traballos presentados por min para a obtención de outros títulos.*

En Santiago de Compostela, 8 de xaneiro de 2021

Asdo. Mauro Fernández Sanjurjo

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

AUTORIZACIÓN DO DIRECTOR/TITOR DA TESE

Design of Real-Time Multiple Object Visual Detection and Tracking Systems

Dr. Manuel Mucientes Molina, Profesor Contratado Doutor da Área de Ciencia da Computación e Intelixencia Artificial da Universidade de Santiago de Compostela

Dr. Víctor M. Brea Sánchez, Profesor Titular da Área de Electrónica da Universidade de Santiago de Compostela

INFORMAN:

Que autorizamos a presentación desta tese, que se corresponde co traballo realizado por ***Don Mauro Fernández Sanjurjo*** *baixo a nosa dirección, considerando que reúne os requisitos esixidos no Regulamento de Estudos de Doutoramento da USC, e que como directores desta non incorre nas causas de abstención establecidas na Lei 40/2015.*

En Santiago de Compostela, 8 de xaneiro de 2021

Asdo. Manuel Mucientes Molina
Director da tese

Asdo. Víctor M. Brea Sánchez
Director da tese

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Aos meus pais,
aos que non lles podo estar máis agradecido.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

No man is more unhappy than he who never faces adversity. For he is not permitted to prove himself.

Seneca

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Agradecementos

En primeiro lugar, quero expresar o meu máis sincero agradecemento aos meus directores Manuel Mucientes Molina e Víctor M. Brea Sánchez, por toda a axuda que me prestaron durante estes anos.

Quixera agradecer tamén ao Departamento de Electrónica e Computación, así como ao Centro de Investigación en Tecnoloxías Intelixentes (CiTIUS), da Universidade de Santiago de Compostela, pola excelente contorna de traballo e os recursos que fixeron posible o desenvolvemento desta tese. Ao director adxunto do CiTIUS durante a maior parte da miña etapa, Prof. Paulo Félix, polo seu apoio e proximidade, á actual dirección do centro, Prof. Paula López e Prof. Senén Barro, así como ás unidades de apoio do centro pola súa constante dispoñibilidade. Tamén gustaríame mencionar ao Prof. Jorge Batista e a todo o grupo do ISR (*Instituto de Sistemas e Robótica*) da Universidade de Coimbra por acollerme durante a miña estadía.

É imporante agradecer en especial as amizades que fixen durante esta etapa no CiTIUS, destacando a Brais, Bea, Dani e Alberto, por estar sempre para todo o necesario e por as risas que pasamos.

Doutra banda, gustaríame agradecer a todas as entidades que fixeron posible o desenvolvemento desta tese co seu financiamento. En primeiro lugar ao Ministerio de Ciencia, Innovación y Universidades pola axuda de Formacion de Personal Investigador (ref. BES-2015-071889). Tamén á Consellería de Educación, Universidades y Formación Profesional de la Xunta de Galicia (ref. ED431C 2018/29, ED431C 2017/69 e acreditación 2016-2019, ED431G/08), ambas co-financiadas polo Fondo Europeo de Desenvolvemento Rexional do programa ERDF/FEDER. Tamén expresar o meu agradecemento ao Ministerio de Economía, Industria y Competitividad (ref. TIN2017-84796-C2-1-R e TEC2015-66878-C3-3-R) e ao Ministerio de Ciencia, Innovación y Universidades (ref. RTI2018-097088-B-C32).

8 de xaneiro de 2021

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Resumo

A detección e o seguimento de múltiples obxectos é unha das principais liñas de investigación no campo da visión por computador. A tarefa consiste en identificar todos os obxectos de interese nunha imaxe ou fotograma de vídeo e asignar un identificador único a aquelas instancias que correspondan ao mesmo obxecto mentres este apareza na escena. O primeiro paso no proceso dun sistema de detección e seguimento de múltiples obxectos (DSMO) é a detección, que establece as posicións e categorías de todos os obxectos de interese na imaxe actual. O seguimento mantén a identidade do obxecto ao longo do vídeo, e a súa función é tripla: i) diferenciar instancias de obxectos que pertencen á mesma categoría; ii) estimar a nova posición dos obxectos para mellorar a asociación de datos; iii) cando o detector non pode ser executado en todos os fotogramas do vídeo debido ás necesidades de tempo real, manter as posicións de todos os obxectos que están a ser seguidos (ou trackers) para axudar á asociación de datos cando se chame de novo ao detector. Por último, a asociación de datos emparella as deteccións e os trackers, crea novos obxectos e elimina os obxectos pertinentes da escena.

Os sistemas de detección e seguimento de múltiples obxectos forman a base de solucións sofisticadas que realizan automaticamente tarefas de visión por computador. Un exemplo diso poderían ser as aplicacións de reconto de vehículos, nas que, unha vez detectado o tipo de vehículo (categoría), o sistema mantén un rexistro de cantos vehículos de cada tipo circularon durante un período de tempo determinado. Aplicacións como a detección de espazos ocupados nun aparcamento e o tempo que cada coche pasou dentro da instalación son tamén variantes deste tipo de aplicación. Outro uso nas aplicacións de tráfico é a detección de anomalías, é dicir, unha vez que se identificou o fluxo habitual de tráfico, o sistema lanza unha alarma cando detecta un evento anómalo como vehículos que se desprazan na dirección equivocada ou accidentes. Estas aplicacións requiren o coñecemento de onde se atopan os

obxectos de interese en cada momento, é dicir, estes sistemas deben ser capaces de detectar os obxectos de interese e seguilos mantendo a súa identidade mentres aparecen na escena.

Desde o punto de vista do tipo de información que debe transferirse ao usuario final, as aplicacións de visión por computador poden dividirse en dous grandes grupos: i) as que envían un fluxo continuo de vídeo, e ii) as que só necesitan enviar datos calculados a partir da análise de vídeo. Estas últimas son especialmente adecuadas para a computación no bordo: o sistema de visión fai a maior parte ou toda a computación no bordo, antes de enviar o resultado —que normalmente non inclúe vídeo, senón datos— ao usuario final. Algúns exemplos deste tipo de sistemas son: peaxes de autoestrada; aplicacións de cálculo de persoas para determinar a capacidade de asentos nunha instalación; estimación da velocidade dos vehículos; vixilancia por vídeo que busca a detección de anomalías no comportamento das persoas como a entrada en áreas restrinxidas; aplicacións para detectar e evitar colisións a bordo de vehículos aéreos non tripulados; automóbiles sen condutor; ou a avaliación para a análise da distancia social —unha aplicación oportuna en tempos do COVID-19— só para mencionar algúns exemplos. Estas solucións requiren menores anchos de banda, poñendo menos restricións de hardware na rede de comunicación, favorecendo así a computación no bordo. De feito, este tipo de computación fai que o sistema sexa máis robusto ante erros de comunicación e, ao mesmo tempo, aínda que a incomodidade causada polo despregamento de cámaras nos espazos públicos non pode ser mitigada, polo menos os problemas de privacidade nos ataques aos datos poden verse reducidos —os datos transmitidos son a maioría das veces datos procesados en lugar de vídeo en bruto.

A detección de obxectos nun sistema de DSMO pode realizarse de varias formas, sendo as máis relevantes os algoritmos de subtracción de fondo, os detectores clásicos baseados en características deseñadas manualmente ou os detectores de aprendizaxe profunda. Ademais, o seguimento pode abordarse de dúas maneiras principais. A primeira, coñecida como Seguimento Visual de Obxectos (SVO), consiste en estimar a posición actual do obxecto a partir da súa posición na imaxe do instante anterior e a imaxe actual, utilizando unicamente información visual para realizar a tarefa. A segunda, coñecida como Seguimento de Múltiples Obxectos (SMO), consiste en estimar a nova posición do obxecto a través da asociación de detección correspondentes ao mesmo obxectivo ao longo do tempo. Os algoritmos de SMO poden opcionalmente facer uso de solucións de SVO ou outras solucións para determinar a nova posición dos obxectos e ter unha representación de características dos mesmos, pero diferéncianse dos enfoques de SVO en que todas estas tarefas realízanse para mellorar a pos-

terior asociación de datos, é dicir, asociar detecciónss con trackers, que é a tarefa mínima que debe cumprir calquera algoritmo de SMO para ser considerado como tal.

En canto á diferenza entre sistemas de DSMO e enfoques de SMO é importante resaltar que nesta tese centrámonos no desenvolvemento de sistemas de DSMO, é dicir, trátase dun sistema completo que inclúe todos os compoñentes que permiten identificar e seguir os obxectos, dando como resultado unha solución integral. Doutra banda, os enfoques de SMO resolven o problema do seguimento mediante a correcta asociación das detecciónss. Polo xeral, as solucións de SMO asumen detecciónss dispoñibles en todos os fotogramas de vídeo e non teñen en conta o tempo de detección nin a arquitectura do detector. Por tanto, a maioría das solucións de SMO non son sistemas de DSMO xa que estes últimos inclúen todos os compoñentes necesarios, non asumen que as detecciónss están sempre dispoñibles e deben ter en conta o tempo de detección para determinar a súa velocidade de procesamento. Esta distinción é particularmente importante no caso das solucións que requiren unha velocidade de funcionamento en tempo real, é dicir, un sistema de detección e seguimento de múltiples obxectos en tempo real (DSMOTR). Estes enfoques deben funcionar en tempo real, tendo en conta todos os compoñentes, mentres que no caso das solucións de SMO, só a asociación de datos debe funcionar en tempo real para que se considere un enfoque de seguimento de múltiples obxectos en tempo real (SMORT). Por tanto, o desenvolvemento dun sistema de DSMOTR, como os que se abordan nesta tese, é moito máis complexo que o dos enfoques de SMO e de SMORT.

A pesar dos numerosos avances dos últimos anos nos enfoques de detección e seguimento, o desenvolvemento de sistemas de DSMOTR aínda non se explorou suficientemente. Iso débese a que tanto a detección como o seguimento de obxectos son correntes de investigación activas separadas, cuxo principal obxectivo en ambos os casos é mellorar a precisión dos algoritmos, a miúdo sen prestar tanta atención ao rendemento en tempo real na súa integración conxunta. Isto tradúcese nunha gran dificultade para proporcionar solucións integrais para aplicacións con velocidade de procesamento en tempo real, o que fai que moitos sistemas de DSMO actuais sigan intercambiando precisión por velocidade para cumprir cos requirimentos de tempo de cómputo. Para comprender o desafío, a continuación, preséntase unha análise dos principais compoñentes dun sistema de DSMO, é dicir, detección, seguimento, e a súa combinación.

A tarefa de detección de obxectos consiste en identificar nunha imaxe estática os obxectos de interese que proporcionan a súa caixa delimitadora (ou bounding box). Dúas solucións

clásicas a este problema foron os algoritmos de subtracción de fondo ou os detectores baseados en características deseñadas manualmente. A subtracción de fondo, a partir da análise dos fotogramas iniciais dun vídeo, crea un modelo do fondo que permite diferenciar a parte estática da imaxe dos obxectos en movemento que aparecen nela. As solucións desta natureza presentan alta velocidade e baixas necesidades de computación, pero a súa precisión é limitada, polo que o seu uso restrínxese a escenarios controlados.

Doutra banda, os detectores clásicos utilizan características de deseño manual, como SIFT para puntos característicos ou HOG para orientacións de gradiente, para establecer unha representación dos obxectos que se queren detectar. Unha vez establecida esta representación do obxecto, a tarefa de detección redúcese á procura desas características na imaxe. Un dos primeiros métodos para propoñer rexións onde poderían estar os obxectos de interese, foron as xanelas deslizantes, que son moi custosas computacionalmente porque exploran todas as posibles localizacións dentro da imaxe. Ademais, a súa precisión na estimación da escala é limitada. Máis tarde, xurdiron métodos máis complexos para xerar propostas de rexións, como a procura selectiva, que fusiona píxeles similares en rexións. Tamén se introduciron técnicas como a Análise de Compoñentes Principais (PCA) para reducir a dimensionalidade das características e acelerar o cálculo de modelos de aprendizaxe supervisada como as máquinas de vectores de soporte (SVM) para clasificar as diferentes categorías de obxectos.

A dirección do campo de investigación na detección de obxectos cambiou radicalmente en 2012 cando Krizhevsky e outros propuxeron AlexNet, unha rede de clasificación baseada en redes neuronais convolucionais. Aínda que as redes neuronais xa foran propostas en 1989 por Yan LeCun, este rexurdimento produciuse en parte debido a unha maior potencia de cálculo das unidades de procesamento gráfico (GPU) e a conxuntos de datos máis grandes. O auxe da aprendizaxe profunda representou un cambio de paradigma, xa que xa non era necesario deseñar descritores para obxectos particulares, senón que se podían extraer automaticamente características máis robustas dun gran conxunto de datos directamente a través da rede. Esta aprendizaxe automática de características permitiu adaptar os algoritmos de detección a diferentes dominios cuxo deseño manual era sempre un desafío.

Desde a introdución de Alexnet, fixéronse numerosos avances en visión por computador a través da aprendizaxe profunda. Especificamente, para a tarefa de detección de obxectos, destaca R-CNN, unha rede de detección que traballa sobre rexións propostas por procura selectiva e cada unha delas tómase como entrada para que a rede produza un vector de características. Este vector utilízase para clasificación e regresión da caixa delimitadora dos

obxectos. R-CNN é unha mellora con respecto aos métodos clásicos anteriores, pero aínda ten algunhas limitacións, por exemplo, o cálculo de características para cada rexión realízase de maneira illada, o que fai que o algoritmo sexa ineficiente desde o punto de vista computacional. Unha mellora deste enfoque é Fast- RCNN, que xera propostas de rexións da mesma maneira que R-CNN pero aplica a extracción de características a toda a imaxe, e procesa as rexións directamente no mapa de características profundas. A partir de aí, a investigación de detección de obxectos dividiuse en dúas correntes principais: detectores de dúas etapas e dunha etapa.

Os detectores de dúas etapas na primeira fase integran unha Rede de Propostas de Rexións (RPN) para extraer rexións de interese con mapas de características profundas, e logo, na segunda etapa, realizan a clasificación de obxectos e a regresión de caixas delimitadoras nestas rexións. A primeira RPN foi proposta en Faster R-CNN, mellorando o traballo previo de Fast R-CNN. Por outra banda, os detectores dunha etapa, en lugar dun problema de clasificación baseado en propostas de rexións, propoñen caixas de predición a partir de imaxes de entrada directamente sen un paso de proposta de rexión. Un dos primeiros deseños foi a rede YOLO que divide a imaxe nunha cuadrícula, onde unha cela da cuadrícula é responsable de detectar o obxecto cuxo centro está contido na mesma. O detector SSD adopta un enfoque similar, pero con múltiples relacións de aspecto por cela de cuadrícula e varias escalas de mapa de características para mellorar a predición.

Desde este punto en diante, houbo numerosos avances en ambas correntes. Un dos avances máis relevantes foi FPN. Esta arquitectura está deseñada para detectar obxectos a diferentes escalas a través dunha aproximación de arriba a abaixo con conexións laterais que permite construír unha semántica de alto nivel en todas as escalas. Actualmente, FPN utilízase de forma habitual con detectores de dúas etapas, e a súa influencia pódese atopar tamén nos novos enfoques dunha soa etapa como YOLOv3 ou RetinaNet, e como parte de solucións de alto nivel en concursos como o MS COCO.

A pesar de todos os avances no campo da detección de obxectos, a gran maioría das solucións non son adecuadas para un sistema de DSMOTR. Os algoritmos de subtracción de fondo, a pesar da súa alta velocidade e baixos requirimentos computacionais, teñen unha baixa precisión, non poden identificar a categoría do obxecto detectado, presentan problemas na segmentación de obxectos e artefactos como sombras, e limítanse a cámaras estáticas e obxectos en movemento. Por outra banda, o uso de detectores clásicos descártanse para un enfoque de DSMO, xa que o seu rendemento está moi lonxe dos detectores baseados en

aprendizaxe profunda de última xeración, e isto comprometería a precisión do sistema. En canto ás solucións de aprendizaxe profunda para a detección de obxectos, a maioría delas non alcanzan unha velocidade de procesamento en tempo real, independentemente do hardware utilizado. Este problema agrávase cando se require traballar con solucións en sistemas embebidos. Aínda así, as solucións lixeiras como TinyYOLO son posibles á conta dunha menor precisión. A calidade das detección ten un gran impacto na calidade do proceso de seguimento, por tanto, a selección dun detector máis rápido, pero menos preciso podería comprometer o rendemento do sistema. Ademais, garantir o tempo real no módulo de detección non significa necesariamente facelo para o sistema completo de DSMO.

A tarefa de seguimento mantén a identidade das instancias identificadas polo detector. Hai dúas tendencias principais para realizar esta tarefa: O Seguimento Visual de Obxectos (SVO) e o Seguimento de Múltiples Obxectos (SMO). Os algoritmos de SVO tentan calcular as posicións sucesivas dun obxecto a partir dunha detección inicial utilizando só a información visual dispoñible. É dicir, no caso ideal, unha soa primeira detección sería suficiente para seguir ao obxecto ata que este desapareza. Por outra banda, as solucións de SMO tratan a tarefa como un problema de asociación de datos, é dicir, o seguimento redúcese á correcta asociación de detección pertencentes ao mesmo obxecto ao longo do tempo.

Aínda que os enfoques de SVO tamén se coñecen como enfoques de seguimento dun único obxecto, son moi relevantes para un sistema de DSMOTR, xa que permiten establecer a posición dos obxectos sen necesidade de realizar detección en todos as fotogramas. A redución da dependencia das detección periódicas é unha característica importante, xa que os detectores de aprendizaxe profunda requiren moito tempo e adoitan representar a parte máis lenta do sistema. De feito, só hai unhas poucas solucións para a detección de obxectos que poderían executarse en todos as fotogramas de vídeo mantendo unha velocidade de cómputo en tempo real.

Unha das solucións de SVO máis relevantes son os filtros de correlación discriminativos (DCF). Un DCF é unha técnica supervisada para aprender un clasificador lineal explotando as propiedades de correlación circular para adestrar a un regresor deslizante. Nos últimos anos, os enfoques baseados en DCF aplicáronse con éxito para o seguimento. Un dos primeiros foi MOSSE que usaba só exemplos en escala de grises para adestrar o filtro. O marco dos DCF ampliouse máis tarde a mapas de características multicanle, que permiten incorporar características de alta dimensión, tales como HOG para mellorar a precisión do seguimento. Máis tarde, o marco do DCF foi mellorado significativamente, por exemplo, incluíndo a estimación

de escala, memoria a longo prazo, e engadindo a regularización espacial para aliviar os efectos periódicos da convolución circular.

O éxito das CNNs para a tarefa de detección tamén se trasladou ao seguimento para aumentar a robustez e a precisión dos algoritmos mediante o uso de características profundas con máis información que as deseñadas manualmente. Varias solucións inclúen CNNs dentro dun marco DCF como CCOT, que introduce mapas de características multi-resolución, e ECO, a súa versión mellorada. Recentemente, os mesmos autores, continuaron esta liña de investigación cunha nova arquitectura de seguimento, que consiste en compoñentes de estimación e clasificación dedicados a mellorar a precisión das bounding boxes.

Outro feito relevante nas arquitecturas de aprendizaxe profunda de SVO son as redes siamesas. Os algoritmos de seguimento baseados en redes siamesas resolven o problema mediante correlación cruzada entre un exemplar do obxecto e unha rexión de procura. SiamFC é un dos primeiros enfoques deste tipo. Debido ao seu éxito como tracker, fixéronse numerosas modificacións a esta arquitectura, por exemplo, engadindo unha Rede de Propostas de Rexións (RPN) para xerar propostas de caixas delimitadoras que pasan polas ramas de clasificación e regresión, centrando o adestramento en distractores semánticos, e engadindo unha estratexia de procura de rexións para o seguimento a longo prazo, ou adestrando o tracker con arquitecturas CNN máis complexas.

A maioría das solucións actuais de mellor rendemento, como se mostra nos últimos desafíos de seguimento de obxectos visuais, baséanse nunha formulación CNN do DCF ou en redes siamesas. A mellora na precisión dos algoritmos de SVO nos últimos anos foi significativa. Con todo, o seu uso para un sistema de DSMOTR segue presentando desafíos. Por unha banda, a medida que se desenvolveron solucións máis completas, o tempo de computación tamén se incrementou. Nos últimos anos, con todo, esta tendencia está a empezar a invertirse, dando lugar a algoritmos como ATOM ou SiamFC que alcanzan a velocidade de procesamento en tempo real para un obxecto. Con todo, estas arquitecturas de aprendizaxe profunda non proporcionan un marco para tratar con múltiples obxectos, e a súa escalabilidade a través de múltiples instancias é limitada, non alcanzando a velocidade de procesamento en tempo real.

Os algoritmos de SMO realizan a tarefa de seguimento de múltiples obxectos asociando unha identificación a cada detección —representada por unha caixa delimitadora— en cada fotograma. É dicir, asumen que hai detección dispoñibles en cada instante de tempo e, con isto, o seguimento redúcese a asociar detección que pertencen ao mesmo obxecto en fotogra-

mas sucesivos. Esta é unha tarefa difícil debido ás numerosas oclusións e interaccións entre obxectos que tamén poden ser moi similares entre si. Esta é unha das razóns polas que a aplicación de modelos de SVO para o seguimento de múltiples obxectos non é unha solución adecuada, xa que incorre en numerosos cambios de identidade, debido a que estes modelos non proporcionan un marco para a asociación de datos para incluír, eliminar ou combinar obxectos en seguimento. Os enfoques de SMO tenden a presentar as seguintes etapas: detección, estimación do movemento/extracción de características, afinidade e asociación. Na maioría dos enfoques de SMO a etapa de detección preséntase xeralmente por separado do algoritmo. Na etapa de estimación do movemento, calcúlase a nova posición do obxecto mentres que na etapa de extracción de características establécese unha representación visual do mesmo. En ambos os casos, isto faise para mellorar a asociación de datos. Na etapa de afinidade, establécese unha medida da similitude entre os obxectos, a través da aparencia, a distancia ou ambas. Por último, na etapa de asociación, emparéllanse as detección e os trackers, xeralmente buscando a mellor solución global que optimice a medida de afinidade seleccionada.

Diferentes algoritmos de SMO poderían empregar diferentes etapas. Se o enfoque inclúe a etapa de detección como parte do algoritmo, considérase un sistema de DSMO. Algunhas solucións de SMO non fan uso da estimación do movemento ou das etapas de representación de características, mentres que outras só utilizan unha das dúas. Isto permítelles alcanzar unha alta velocidade á conta dunha menor precisión. Con todo, todos os algoritmos de SMO teñen polo menos unha medida de afinidade e etapas de asociación de datos para resolver o problema. Os sistemas de DSMO son solucións completas e, como tales, deben incluír tamén o detector; nos casos en que non se dispoña de detección en todos as fotogramas, tamén será necesaria a estimación do movemento para evitar a perda da posición dos obxectos ata a chegada da seguinte detección.

Unha forma de mellorar a asociación entre as detección e os obxectos en seguimento é cunha representación adecuada dos mesmos. Por iso, desde a entrada da aprendizaxe profunda, os algoritmos de SMO incluíron características CNN para modelar a aparencia dos obxectos. Estes algoritmos separan as etapas de detección e de extracción de características de seguimento, que xeralmente son redundantes, xa que para detectar obxectos é necesario extraer características. O detector adoita ser a parte máis custosa do sistema, polo que engadir unha etapa adicional de extracción de características adoita facelos restritivos para aplicacións en tempo real. As arquitecturas de detección e extracción de características que permiten unir estas dúas etapas nunha mesma CNN producen unha representación robusta dos obxectos nun

tempo de cálculo menor. Ademais, estes enfoques de SMO son tamén sistemas de DSMO xa que son solucións integrais.

Para a predición do movemento, xurdiron solucións sofisticadas baseadas na aprendizaxe profunda, como as redes de memoria a longo e curto prazo (LSTM), que permiten a predición de modelos non lineais de movemento, aínda que o seu custo computacional segue sendo elevado, facendo que os predictores bayesianos clásicos como o Filtro de Kalman sigan sendo unha das solucións máis utilizadas para os algoritmos de última xeración.

O obxectivo da asociación de datos é a optimización da medida de afinidade, que ten en conta as características dos obxectos, así como a súa predición de movemento para realizar a asignación. Hai varias solucións a este problema, aínda que a máis utilizada é o método húngaro.

Fixéronse moitos avances no campo de SMO, o que se reflicte no gran número de algoritmos desenvolvidos cada ano. Con todo, pasar dunha solución de SMO a un sistema de DSMOTR non é tan sinxelo como parece. Analizando os mellores algoritmos actuais, podemos observar unha serie de inconvenientes para o deseño dun sistema de DSMOTR. En primeiro lugar, a maioría das solucións enumeradas non funcionan en tempo real. Ademais, os seus tempos de cálculo non teñen en conta as restricións de hardware, e nin sequera consideran o tempo do detector. Por último, un gran número de sistemas non son solucións en liña, senón que poden utilizar a información futura para corrixir os erros de seguimento pasados, algo que non é posible nun sistema de DSMOTR.

Como se dixo anteriormente, a maioría das aplicacións que esixen un sistema de DSMO requiren unha velocidade de procesamento en tempo real. O deseño dun sistema de DSMOTR é unha tarefa complexa debido a unha serie de desafíos: a gran maioría das solucións desenvolvidas nos últimos anos tanto para a detección como para o seguimento non alcanzan a velocidade de procesamento en tempo real; solucións máis rápidas e menos precisas poden comprometer a utilidade do sistema en escenarios reais, facendo que a aplicación xa non sexa útil para o propósito previsto; o tempo real debe alcanzarse para todo o sistema, é dicir, incluíndo a detección e o seguimento. Isto fai que as solucións que ofrecen unha velocidade adecuada por separado xa non sexan válidas cando se utilizan en conxunto.

Todos estes desafíos agrávanse cando se trata de computación embebida debido á menor capacidade de computación dos dispositivos. A visión por computador no bordo expón moitos desafíos científicos e técnicos. Como xa se dixo, os algoritmos modernos de visión por computador baséanse principalmente na aprendizaxe profunda, onde xeralmente, máis profundo

significa máis precisión, pero tamén tempos de procesamento máis elevados e unha maior capacidade de computación subxacente para soportar grandes requisitos de memoria. Así pois, os algoritmos de aprendizaxe profunda de visión por computador de última xeración execútanse en GPUs de servidor de gama alta, que en moitas ocasións non son unha opción como dispositivos embebidos debido ao seu prezo, tamaño e consumo de enerxía, por exemplo, a bordo dun UAV. A solución máis sinxela é o mapeo dos algoritmos nas GPU de baixo consumo, que teñen menos capacidade de cálculo que as GPU dos servidores. Isto adoita vir acompañado de penalizacións de tempo e precisión. Por tanto, o reto é deseñar algoritmos de análise de vídeo en dispositivos embebidos mantendo a precisión e a velocidade.

O obxectivo principal desta tese doutoral é deseñar sistemas de DSMOTR co fin de obter resultados que permitan a súa utilización para aplicacións reais con alta precisión, tanto en arquitecturas de servidores como en sistemas embebidos. Para iso perseguíronse os seguintes obxectivos:

1. **Un sistema de DSMOTR para unha arquitectura de servidor**. O sistema debe ter un módulo de detección baseado nunha arquitectura de aprendizaxe profunda que identifique os obxectos de interese. Posteriormente, o módulo de seguimento estimará a posición dos obxectos de dúas maneiras: usando características visuais por unha banda e usando un filtro bayesiano pola outra. A combinación de información de aparencia e movemento permitirá ao sistema detectar oclusións e posteriormente recuperar a identidade dos obxectos. Finalmente, o módulo de asociación de datos encargarase de emparellar as detección e os obxectos en seguimento para engadir, eliminar e refinar a posición dos mesmos. Deseñado desta maneira, o sistema non dependerá do detector en cada fotograma e, por tanto, será posible alcanzar unha velocidade de procesamento superior a 25 FPS con ducias de obxectos.

2. **Un sistema de DSMOTR para arquitecturas embebidas**. O sistema debería incluír unha arquitectura conxunta de detección e modelado de obxectos para proporcionar, ademais da posición dos obxectos, un vector de características de aprendizaxe profunda que represente a súa aparencia e permita unha asociación de datos máis robusta. Para mitigar a dependencia do sistema do módulo de detección, será necesario realizar un seguimento visual dos obxectos para manter a súa posición entre chamadas ao detector, e empregar un seguimento baseado en filtros bayesianos para permitir a posterior reidentificación dos obxectos en caso de oclusións. O obxectivo do sistema é propor-

cionar resultados en tempo real mentres se desprega nun dispositivo embebido de baixo consumo.

3. **Desenvolvemento de aplicacións de monitoraxe de tráfico e peóns en situacións reais baseadas nas arquitecturas propostas**. En primeiro lugar, desenvolveranse dúas aplicacións de monitoraxe do tráfico. A primeira será unha aplicación de análise de entrada/saída en rotondas na que, a partir dun vídeo gravado por un vehículo aéreo non tripulado (UAV), supervisarase a entrada e saída que toma cada vehículo. A segunda será unha aplicación de detección de anomalías en estradas na que, a partir dun vídeo, identificará o fluxo de tráfico, tendo en conta a dirección do movemento e a velocidade. A continuación, lanzarase unha alarma cando ocorra un evento anormal como accidentes ou vehículos en sentido contrario. En segundo lugar, medirase o rendemento da arquitectura para sistemas embebido na tarefa de vixilancia de peóns. Para iso, utilizarase un conxunto de datos de última xeración con situacións heteroxéneas.

Nesta memoria detállanse as contribucións do traballo desenvolvido e os resultados máis relevantes dá investigación.

En concreto, o Capítulo 2 introduce a evolución do estado da arte das diferentes partes que poden formar un sistema de DSMOTR. Isto inclúe unha revisión dos métodos de detección de obxectos en imaxes, pasando dos clásicos algoritmos de subtracción de fondo e baseados en características de deseño manual a solucións máis recentes de aprendizaxe profunda. No caso do seguimento, faise un estudo das dúas principais tendencias actuais. Os algoritmos de SVO seguen un obxecto desde unha soa detección, explotando só a información visual, mentres que as solucións de SMO tratan con múltiples obxectos na escena a través dun enfoque de seguimento por detección.

No Capítulo 3 preséntase un sistema de DSMOTR para arquitectura de servidor que resolve dous dos desafíos máis importantes para unha aplicación de seguimento: o funcionamento en tempo real con centos de obxectos e as oclusións totais, que penalizan o rendemento do sistema. O sistema baséase en tres módulos: detección, seguimento e asociación de datos. En primeiro lugar, os obxectos identifícanse mediante un detector baseado na aprendizaxe profunda. Segundo, o seguimento realízase cunha combinación dun DCF e un filtro de Kalman. Isto permite estimar o erro para que o seguimento sexa máis robusto e fiable. Por último, a asociación de datos mediante o algoritmo Húngaro combina a información dos pasos anteriores. Todo isto permite que o sistema alcance unha velocidade de procesamento en tempo

real, á vez que é robusto fronte a oclusións totais e permite procesar máis de 400 obxectos simultaneamente. Para demostrar a utilidade do sistema en contornas reais, desenvolvéronse dúas aplicacións de tráfico: a detección de anomalías e a análise de entrada/saída de rotondas. O sistema foi avaliado con máis de dous mil vehículos en contornas reais.

No Capítulo 4 preséntase unha solución portátil para a detección e o seguimento en dispositivos embebidos. Este traballo é o noso primeiro achegamento ás arquitecturas integradas. O sistema utiliza un algoritmo de subtracción de fondo para detectar obxectos en movemento e varias instancias do algoritmo de seguimento visual de obxectos de aprendizaxe profunda GOTURN para seguir os obxectos previamente detectados. O rendemento do algoritmo proposto móstrase sobre unha colección de secuencias representativas capturadas coa cámara incorporada, así como un estudo en termos de consumo de enerxía e velocidade.

No Capítulo 5 mellóranse as limitacións dos traballos anteriores para propoñer un novo sistema de DSMOTR para dispositivos de GPU embebidos que amplía unha arquitectura conxunta de seguimento e detección engadindo un módulo composto por algoritmos de seguimento baseados na aparencia e no movemento que permite manter a identidade dos obxectos de interese durante períodos de tempo máis longos, á vez que alivia a carga do detector. O noso sistema está mapeado nunha plataforma de GPU embebida, o que reduce significativamente o consumo de enerxía con respecto a unha GPU de servidor. As métricas de rendemento de seguimento mostran un 51,1% en Precisión de Seguimento de Obxectos Múltiples (MOTA) no conxunto de datos MOT16. Isto, xunto cunha velocidade de procesamento en tempo real de 25,2 FPS para ata 45 obxectos simultáneos e un baixo consumo de enerxía de 15 W, fan do noso sistema unha solución ideal para unha ampla gama de aplicacións de análises de vídeo.

Finalmente, no Capítulo 6 resúmense as conclusións da investigación desenvolvida nesta tese. Ademais, neste capítulo discútense as futuras liñas de traballo.

Contents

1 Introduction **1**
1.1 Object Detection . . . 5
1.2 Object Tracking . . . 7
1.3 Components Integration . . . 11
1.4 Hardware and Real-time Processing . . . 12
1.5 Objectives . . . 13
1.6 Contributions . . . 14
1.7 Dissertation Structure . . . 17

2 Related Work **21**
2.1 Detection . . . 21
2.2 Tracking . . . 28
2.3 Computer Vision on the Edge . . . 41

3 Real-Time Multiple Object Detection and Tracking System for Server Architectures **45**
3.1 System Overview . . . 46
3.2 Results . . . 53
3.3 Final Remarks . . . 63

4 Deep Learning-Based Multiple Object Visual Detection and Tracking on Embedded System for IoT and Mobile Edge Computing Applications **65**
4.1 Multiple Object Tracking Approach . . . 66
4.2 Results . . . 71
4.3 Final Remarks . . . 79

5 Real-Time Multiple Object Detection and Tracking System for Embedded GPUs **81**
5.1 RTMODT System Overview 82
5.2 Experimental Study 89
5.3 Final Remarks 95

6 Conclusions **97**

Bibliography **103**

List of Figures **119**

List of Tables **125**

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

CHAPTER 1

INTRODUCTION

Multiple object detection and tracking is one of the main topics in computer vision. The task is to identify all the objects of interest in a frame of video and to assign a unique ID to those instances that correspond to the same object while it appears in the scene. The first step in the pipeline of a multiple object detection and tracking (MODT) system is detection, which establishes the positions and categories of all the objects of interest in the current frame. Tracking keeps the identity of the object throughout the video, and its role is triple: (i) differentiate between intra-class objects; (ii) estimate the new position of objects to enhance data association; (iii) when the detector cannot be run in all the frames due to real-time requirements, keep the positions of all the tracked objects to help data association when the detector is called again. Finally, given a set of detections and trackers, data association matches detections and trackers, initializes new objects, and deletes objects out of the frame. Figure 1.1 shows an example of the steps of the system.

Multiple object detection and tracking systems form the basis of sophisticated solutions that automatically perform computer vision tasks —Figure 1.2. An example of this could be vehicle counting applications, where, once detected the type of vehicle (category), the system keeps a record of how many vehicles of each type have run during a determined period of time. Applications such as the detection of occupied spaces in a parking lot and the time each car has spent inside the facility are also variants of this type of application. Another use in traffic applications is the detection of anomalies, that is, once the usual flow of a traffic intersection has been identified, the system launches an alarm when it detects an anomalous event such as vehicles moving in the wrong direction or accidents. These applications require

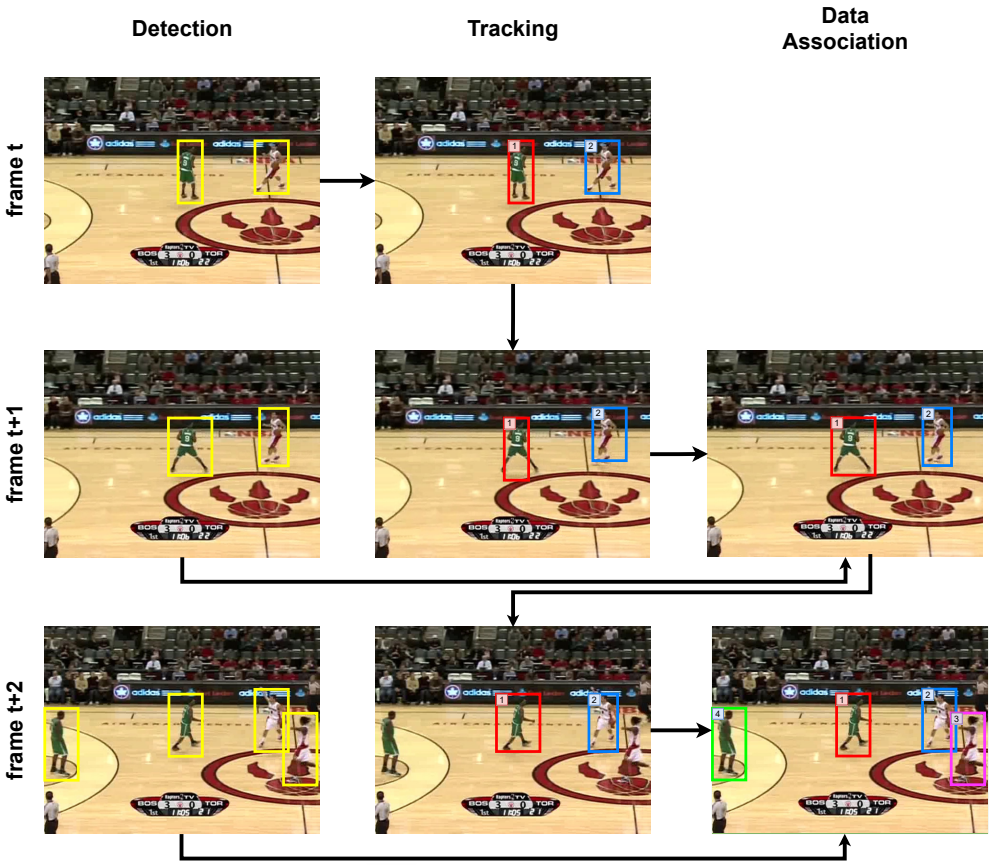


Figure 1.1: Example of multiple object detection and tracking (MODT) for basketball players. The detection module identifies the objects of interest. The tracking module calculates the position of previous trackers in the current frame. The data association module matches detections and trackers to correct, add, and remove trackers.

the knowledge of where the objects of interest are at each time instant, i.e., these systems must be able to detect the objects of interest and track them maintaining their identity while they appear on the scene.

From the point of view of the type of information to be transferred to the end-user, computer vision applications can be divided into two large groups: (i) those that send a continuous video stream, and (ii) those that only need to send data calculated from video analysis. The

(a) (b) (c) (d)

Figure 1.2: Some examples of applications of MODT systems. (a) assisted surveillance application in airports. (b) and (c) show applications of detection and tracking of vehicles and pedestrians, respectively. (d) drone surveillance in restricted aerial spaces. In all cases it is necessary to identify the object (bounding box), its category and to assign it a unique ID (represented by the color of the bounding box). Images with Pixabay License [138].

latter are especially suitable for edge computing: the computer vision system makes most or all the computing on the edge, before sending the result —which usually does not include video, but data— to the end user [35, 125, 139]. Some examples of this kind of systems are: highway tolls; people counting applications to determine seating capacity; vehicle speed estimation; video surveillance searching for human anomaly detection like the entrance in restricted areas; sense and avoid on board of UAVs; self-driving cars; or video surveillance assessing of social distance observation [128] —a timely application in COVID-19 times— just to mention a few examples. These solutions require lower bandwidths, putting fewer hardware constraints on the communication network, thus favoring edge computing. Indeed,

computing at the edge device makes the system more robust to communication breakdowns and, at the same time, although the uneasiness caused by the deployment of cameras in public spaces cannot be assuaged, at least privacy issues from attacks on data can be mitigated —broadcasted data is most of the time processed data instead of raw video.

The detection of objects in an MODT system can be performed in various forms, the most relevant being background subtraction algorithms, hand-crafted feature-based detectors, or deep learning detectors. Moreover, tracking can be addressed in two main ways. The first one, known as Visual Object Tracking (VOT), consists of estimating the position of the object in the current frame from its position in the previous frame and the current image, using only visual information to perform the task. The second one, known as Multiple Object Tracking (MOT), consists of estimating the new position of the object through the association of detections corresponding to the same target over time. MOT algorithms can optionally make use of VOT or other solutions to determine the new position of objects and have a feature representation of them, but they differ from VOT approaches in that all these tasks are performed to improve the subsequent association of data, i.e. associate detections with trackers, which is the minimum task that must meet any algorithm to be considered an MOT.

Regarding the difference between MODT systems and MOT approaches, in this thesis we focus on the development of MODT, i.e., this refers to a complete system that includes all the components that allow to identify and track objects, resulting in an end-to-end solution. On the other hand, MOT approaches solve the problem of tracking multiple objects through the correct association of detections. Generally, MOT solutions assume detections available in all video frames and do not take into account the detection time nor architecture. Therefore, most of the MOT solutions are not MODT systems since the latter includes all the necessary components, do not assume that detections are always available and they must take into account the detector time to determine their processing speed. This distinction is particularly important in the case of solutions that require real-time operating speed, i.e. a Real-Time Multiple Object Detection and Tracking (RTMODT) system. These approaches need to operate in real-time, taking into account all the components, while in the case of MOT solutions, only the data association must operate in real-time for it to be considered a Real-Time Multiple Object Tracking (RTMOT) approach. Therefore, the development of an RTMODT system, like those addressed in this thesis, is much more complex than both MOT and RTMOT approaches.

Despite the numerous advances in recent years in detection and tracking approaches, the development of RTMODT systems is still insufficiently explored. This is so because both

object detection and object tracking are separate active research streams, whose main objective in both cases is to improve the accuracy of the algorithms, often without paying so much attention to real-time performance in their joint integration. This translates into a great difficulty to provide end-to-end solutions for real-life applications with real-time speed, making many current MODT systems continue to trade accuracy for speed to meet computation time requirements. To understand the challenge, an analysis of the main components of an MODT system, namely, detection and tracking, and their combination are introduced below.

1.1 Object Detection

The object detection task consists of identifying in a static image the objects of interest providing their bounding box. Two classic solutions to this problem were background subtraction algorithms or detectors based on hand-crafted features. Background subtraction, from the analysis of the initial frames of a video, creates a model of the background that allows differentiating the static part of the image from the moving objects that appear in it. Solutions of this nature present high speed and low computational needs, but their precision is limited, so their use is restricted to controlled scenarios.

On the other, hand classic detectors use features, like scale-invariant feature transform (SIFT) descriptors [75] or Histogram of Oriented Gradients (HOG) [21], to establish a representation of the objects to be detected. Once this representation of the object is set, the detection task is reduced to the search of those features in the image. One of the first methods to propose regions, where the objects of interest might be, was sliding windows [20, 107, 108], which are very computationally expensive because they explore all possible locations within the image. In addition, their accuracy in scale estimation was limited. Later, more complex methods to generate region proposals emerged, such as selective search [104], which merges similar pixels into regions. Techniques such as Principal Component Analysis (PCA) were also introduced to reduce the dimensionality of features and speed up the computation of supervised learning models like support vector machines (SVM) or multilayer perceptrons (MLP) to classify the different object categories.

The direction of the research field in object detection changed radically in 2012 when Krizhevsky et al. proposed AlexNet [62], a classification network based on convolutional neural networks. Although neural networks had already been proposed in 1989 by Yan LeCun [64], this re-emergence occurred in part because of a greater computational power of graph-

ics processing units (GPUs) and larger data sets. The advent of deep learning represented a paradigm shift, as it was no longer necessary to design manual descriptors for particular objects, but rather more robust features could be automatically extracted from a large data set directly through the network.

Since the introduction of Alexnet, numerous advances have been made in computer vision through deep learning. Specifically, for the object detection task, R-CNN [39] stands out as an object detection network working on regions proposed by [104], i.e., selective search was used to propose possible locations of the objects, and each of the regions was taken as input for the network to produce a feature vector. This vector was used for classification and bounding box regression. R-CNN is an improvement concerning previous hand-crafted methods but it still has some limitations, for example, calculating features for each region in an isolated way making the algorithm computationally inefficient. An improvement to this approach is Fast-RCNN [38], which generates region proposals in the same manner as R-CNN but applies feature extraction to the entire image, and processes the regions directly on the deep feature map. From that point on, the object detection research has been divided into two major streams: two-stage and one-stage detectors.

- **Two-stage detectors:** in the first stage, these detectors integrate a Region Proposal Network (RPN) to extract regions of interest with deep feature maps, and then, in the second stage, they perform object classification and bounding-box regression on these regions. The first RPN was proposed in Faster R-CNN [90], improving the previous work of Fast R-CNN.

- **One-stage (One-shot) detectors:** instead of a classification problem relying on region proposals, these methods propose predicted boxes from input images directly without a region proposal step. Redmon et al. [86] first suggested this design with the YOLO network that divides the image into a grid, where a grid cell is responsible for detecting the object whose center falls into. The Single Shot Detector (SSD)[74], adopts a similar approach but with multiple aspect ratios per grid cell and various feature map scales to improve prediction.

From this point onwards, there have been numerous advances in both streams. One of the most relevant developments has been FPN [69]. This backbone is designed to detect objects at different scales through a top-down architecture with lateral connections that allows building high-level semantics at all scales. FPN is currently commonly used with two-stage detectors,

and its influence can be found in the new one-stage approaches such as YOLOv3 [88] or RetinaNet [70], and as a part of top-ranked solutions in contests like MS. COCO [71].

Despite all the progress made in the field of object detection, the vast majority of solutions are not suitable for an RTMODT system. Background subtraction algorithms, despite their high speed and low computational requirements, have low accuracy, cannot identify the category of the detected object, present problems in the segmentation of objects and artifacts like shadows, and are limited to static cameras and moving objects. On the other hand, classic hand-crafted-based detectors are ruled out in an MODT approach, as their performance is far away from state-of-the-art deep learning detectors, thus compromising the accuracy of the system. Regarding state-of-the-art deep learning solutions for object detection, most of them do not reach a real-time processing speed regardless of the hardware used. This problem is aggravated when attempting to work on low-powered embedded solutions. Still, lightweight solutions like TinyYOLO [88] are possible at the cost of lower accuracy. The quality of the detections has a great impact on the quality of the tracking process, therefore the selection of a faster but less accurate detector might compromise the performance of the system. Moreover, guaranteeing real-time in the detection module does not necessarily mean doing so for the complete MODT system.

1.2 Object Tracking

The tracking task maintains the identity of the objects identified by the detector. There are two major trends in the state-of-the-art: Visual Object Tracking (VOT) and Multiple Object Tracking (MOT). VOT algorithms try to calculate the successive positions of an object from an initial detection using only the available visual information. That is, in the ideal case, a single first detection would be enough to track the object until it disappears. On the other hand, MOT solutions treat the task as a data association problem, that is, the tracking is reduced to the correct association of detections belonging to the same object over time.

1.2.1 Visual Object Tracking

Although VOT approaches are also known as single object trackers, they are very relevant for an RTMODT system, since they allow to establish the position of objects without the need for detections in all frames. Reducing the dependence on periodic detections is an important feature as deep learning detectors are very time-consuming, and they usually represent the

system's bottleneck. Indeed, there are only a few solutions for object detection that could be run in all video frames maintaining the real-time constraint.

One of the most relevant VOT solutions are Discriminative correlation filters (DCFs). A DCF is a supervised technique for learning a linear classifier by exploiting the properties of circular correlation for training a regressor in a sliding-window fashion. In recent years, DCF based approaches have been successfully applied for tracking. Bolme et al. [13] first introduced the MOSSE tracker, using only grayscale samples to train the filter. The DCF framework was later extended to multi-channel feature maps [28], that allow to incorporate high-dimensional features, such as HOG [21] to improve the tracking accuracy. Later, the DCF framework has been significantly improved by, e.g., including scale estimation [24], non-linear kernels [49], a long-term memory [77], and by adding spatial regularization for alleviating the periodic effects of circular convolution [26].

The success of the CNNs for the detection task was also transferred to tracking for increasing the robustness and accuracy of the algorithms by using richer deep features instead of hand-crafted ones. Several solutions include CNNs within a DCF framework such as CCOT [27], which introduces multi-resolution feature maps, and ECO [22], its enhanced version. Recently, the same authors, continued this line of research with a novel tracking architecture, consisting of dedicated target estimation and classification components to improve bounding box accuracy [23].

Another milestone in VOT deep learning architectures are Siamese networks. Siamese network-based trackers formulate tracking as convolutional feature cross-correlation between a target template and a search region. SiamFC [8] is one of the first approaches of this kind. Due to its success as a tracker, numerous modifications have been made to this architecture by, e.g., adding a Region Proposal Network (RPN) to generate bounding box proposals that go through classification and regression branches [67], focusing the training on semantic distractors [126], and adding a search region strategy for long-term tracking, or training the tracker with more complex CNN architectures [66].

The majority of the best current performing solutions, as shown by the latest VOT tracking challenges [61], are based either on a CNN formulation of the DCF or on Siamese Networks. The improvement in the accuracy of VOT algorithms in recent years has been significant. However, their use for an RTMODT system continues to present challenges. On the one hand, as more complete solutions were developed, the computing time has also been increased, going from 172 FPS in Henriques et al. [49] to 0.3 FPS for the 2017 Visual Object

Tracking Challenge[60] winner C-COT [27]. In recent years, however, this trend is beginning to reverse, giving rise to trackers such as ATOM [23] or SiamFC [8] that do reach real-time processing speed for one target. Yet, these deep learning architectures do not provide a framework to deal with multiple objects, and their scalability through multiple VOT instances is limited, failing to reach real-time processing speed.

1.2.2 Multiple Object Tracking

MOT algorithms perform the tracking task for multiple objects by associating an ID to every detection —represented by a bounding box— in each frame. That is, they assume that there are detections available at each time instant and, with this, tracking is reduced to associate detections that belong to the same object in successive frames. This is a difficult task due to the numerous occlusions and interactions between objects that can also be very similar to each other. This is one of the reasons why applying VOT models for multiple object tracking is not an adequate solution, as it incurs in numerous ID switches and target drift errors, since these models do not provide a framework for data association to include, eliminate or combine trackers. MOT approaches might have the following stages: detection, motion estimation/feature extraction, affinity, and association. In most MOT approaches the detection stage is generally presented separately from the algorithm. In the motion estimation stage, the new position of the object is calculated while in the feature extraction stage a visual representation of the object is set. In both cases, this is done to improve data association. In the affinity stage, a measure of the similarity between objects is established, through appearance, distance, or both. Finally, in the association stage, detections and trackers are matched, generally looking for the best global solution that optimizes the selected affinity measure between detections and trackers.

Different MOT algorithms might employ different stages as shown in Figure 1.3. If the approach includes the detection stage as part of the algorithm, it is considered an MODT system. Some MOT solutions do not make use of motion estimation or feature representation stages such as [11], while others only use one of the two [10]. This allows them to reach a high speed at the cost of lower accuracy. However, all MOT algorithms have at least affinity measure and data association stages to solve the problem. MODT systems are stand-alone solutions, and, as such, they must include the detector as well; in those cases where detections are not available in all frames, motion estimation will also be necessary to avoid losing the position of the objects until the arrival of the next detection.

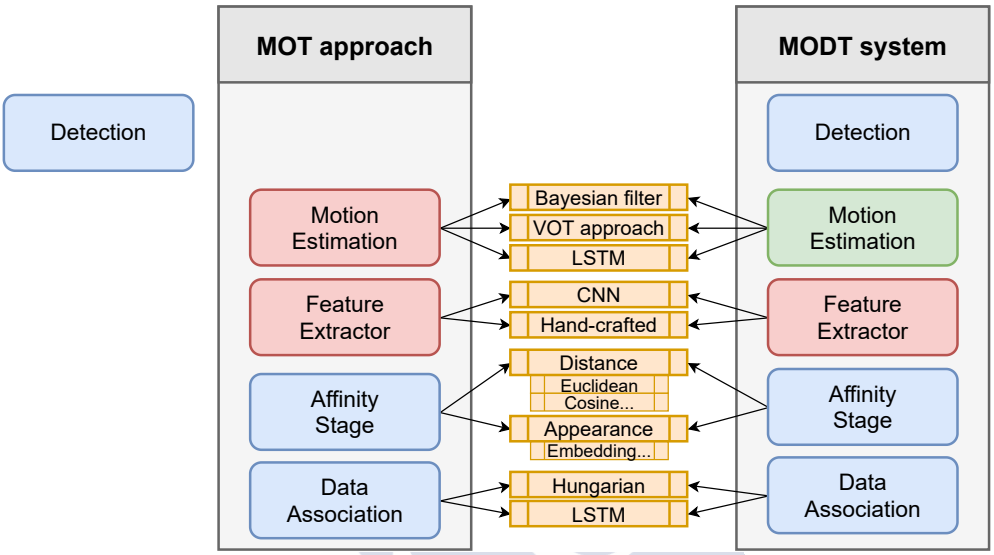


Figure 1.3: Differences between an MOT approach and an MODT system. An MOT approach (left) can have the detection module detached from the solution, whereas an MODT system (right) must include it within the system. Both solutions should have the stages of detection, affinity, and association (represented in blue) while motion estimation and feature extraction (represented in red) are optional. In the case of MODT systems, the motion estimation stage (represented in green) becomes necessary when detections can not be guaranteed in all frames —which frequently occurs for real-time operation.

One way to improve the association between detections and trackers is with a proper representation of the objects. That is why since the entry of deep learning, MOT algorithms included CNN features to model the appearance of objects [15, 17, 114]. These algorithms separate the detection and the tracking feature extraction stages, which are generally redundant, since to detect objects it is necessary to extract features. The detector is usually the bottleneck of the system, so adding an additional deep tracking feature extraction step usually makes its use restrictive for real-time applications. Architectures of detection and feature extraction that enable to join these two stages in the same forward pass of the CNN lead to a robust object representation in less computation time [113, 122]. In addition, these MOT approaches are also MODT systems as they are end-to-end solutions.

For motion prediction, sophisticated solutions based on deep learning such as Long Short-Term Memory (LSTM) networks have emerged, allowing the prediction of non-linear models

of motion, although their computational cost is still high, making classical Bayesian predictors such as the Kalman Filter [53] one of the most widely used solutions for state-of-the-art algorithms [6, 10, 68, 91, 114].

The goal of data association is the optimization of the affinity measure, which takes into account the features of the objects as well as their motion prediction to make the assignment. There are several solutions to this problem, although the most widely used is the Hungarian method [63].

Many advances have been made in recent years in the field of MOT, which is reflected in the large number of algorithms submitted to the MOTChallenge [136], the most widely used benchmark for this research topic. However, going from an MOT solution to an RTMODT system is not as straightforward as it appears. Analyzing the best algorithms reported [136], we can observe a series of drawbacks for the design of an RTMODT system. First, most of the listed solutions do not operate in real-time. Moreover, their computation times do not take into account hardware restrictions and do not even consider the detector time. Finally, a large number of systems are not online solutions but can use future information to correct tracking errors, something that is not possible for an RTMODT system.

1.3 Components Integration

All the above gives an idea of the difficulty in designing an RTMODT system with high precision. There are three basic options for its design:

1. **Detector + VOT tracker adapted for multiple objects:** although this is not strictly an MODT system because it does not present data association, this solution has the advantage of its simplicity, and that it does not require detections in all the frames since the VOT tracker can estimate the position of the objects from the visual information. The detector would be called every given number of frames, set beforehand. This means that more precise detectors can be used and therefore will have a positive impact on the accuracy of the system. Its main drawbacks are that it does not provide a framework to combine detections with trackers, to add new objects, to remove those that come out of the scene, to re-identify them after occlusions, or to reset their position with the detector information. This will yield many ID switches, making it unsuitable for crowded scenarios. In addition, the adaptation of many of the VOT algorithms to multiple objects is not possible in a straightforward way due to scalability issues.

2. **Detector + MOT solution:** this case presents the advantages of having an appropriate framework to deal with multiple objects to add, remove and re-identify the objects. The main problem is that the system becomes dependent on the detector, i.e., detections are needed in all frames. This restricts the type of solutions that can be adopted since it becomes strictly necessary that both solutions achieve real-time processing speed together. If the latter is met, in most cases, this will result in faster but less accurate detectors, as is the case of MOT solutions that do not use image information, which affects the overall performance of the system.

3. **Detector + VOT + MOT:** this is the approach proposed in this thesis. In this case, the advantages of all the components can be exploited while minimizing their shortcomings. On the one hand, thanks to the use of a VOT approach, a slower but more accurate detector can be employed, since it is not necessary to execute it in all frames, making the system less dependent on it. On the other hand, the MOT solution provides a framework to deal with multiple objects and its execution is only required in those frames where detections are made. Finally, the VOT algorithm can be faster and less accurate since the MOT approach will allow it to reset its position with the detector information, and therefore, it has to follow the objects only between detector calls.

1.4 Hardware and Real-time Processing

Most of the applications that demand an MODT system require real-time processing speed. Designing an RTMODT system is a complex task due to a series of challenges:

- The vast majority of solutions developed in recent years for both detection and tracking do not reach real-time processing speed.
- Faster and less accurate solutions can compromise the usability of the system in real scenarios, making the application no longer useful for the intended purpose.
- Real-time must be achieved for the entire system, i.e. including detection and tracking. This makes solutions that offer adequate speed separately no longer valid when used in conjunction.

All these challenges are aggravated when it comes to edge computing due to the lower computing capacity of embedded systems. Computer vision on the edge poses many scientific

and technical challenges. As stated above, modern computer vision algorithms mainly rely on deep learning, where generally, deeper means more accuracy, but also longer processing times and more underlying computing capacity to support large memory requirements [47, 102]. So, state-of-the-art computer vision deep learning algorithms run on high-end server GPUs, which on many occasions are not an option as edge devices due to their price, size, and power consumption, e.g. onboard of a quadcopter. The most straightforward solution is the mapping of algorithms onto embedded GPUs, which feature less computing capacity than that of server GPUs. This usually comes in with time and accuracy penalties from a redesign of accurate but deep models into shorter pipelines to fit on an embedded GPU. The challenge is to design video analytics algorithms on embedded GPUs while keeping accuracy and frame rate.

1.5 Objectives

The main objective of this Ph.D. thesis is to design RTMODT systems with the aim of obtaining state-of-the-art results that allow their use for real applications with high precision, both in server architectures and on embedded systems. To achieve this, the following objectives have been pursued:

O1. An RTMODT system for a server architecture

The system must have a detection module based on a deep learning architecture that will identify the objects of interest. Then, the tracking module will estimate the position of the objects in two ways: using visual features on the one hand and a Bayesian filter on the other. Combining appearance and motion information will allow the system to detect occlusions and later recover the identity of the objects. Finally, the data association module will be in charge of matching detections and trackers to add, remove, and refine the position of the objects. Designed in this way, the system will not depend on the detector in every frame and, therefore, it will be possible to reach a processing speed higher than 25 FPS with dozens of objects.

O2. An RTMODT system for embedded architectures

The system should include a joint architecture of detection and embedding to provide, in addition to the position of the objects, a vector of deep learning features that models their appearance and allows a more robust data association. To mitigate the system's

dependence on the detection module, it will be necessary to perform a fast visual object tracking to maintain the objects' position between detector calls, and to employ Bayesian filter-based tracking to enable subsequent re-identification of objects in the case of occlusions. The objective of the system is to provide state-of-the-art results in real-time while being deployed on an embedded device with low power consumption.

O3. Development of real-life traffic and pedestrian monitoring applications based on the proposed architectures

First, using the proposed architecture in O1, two computer traffic monitoring applications will be developed. The first one will be a roundabout I/O analysis application in which, from a video recorded by a UAV, the entry and exit taken by each vehicle will be monitored. The second one will be a traffic anomaly detection application in which, from a video, the traffic flow will be identified, taking into account the direction of movement and speed. Then, an alarm will be launched when an abnormal event occurs such as accidents or opposite direction vehicles. Secondly, the performance of the proposed architecture in O2 will be measured in the case of pedestrian monitoring. To do so, a state-of-the-art dataset with heterogeneous situations will be used.

1.6 Contributions

The research developed in this thesis has led to the following contributions:

C1. An RTMODT system for server architecture

C1.1 The system is composed of a deep-learning-based detector, a low-level Discriminative Correlation Filter (DCF) tracker, a high-level Kalman Filter tracker, and data association based on the Hungarian algorithm. The system can process more than 400 objects simultaneously in videos with HD resolution in real-time.

C1.2 To handle occlusions, we provide a metric for on-line tracking failure detection by estimating the distance between two independent tracking methods, allowing to update the system's tracking error accordingly. This error is used to determine where the objects might be by defining a Search ROI (Region-Of-Interest). When an object is occluded, the error, and therefore the search ROI size, increases allowing to re-identify the object when it reappears by searching for it in a larger area.

C2. An RTMODT system for edge computing

C2.1 The detector incorporates a joint architecture of detection and embedding. The tracking is done by combining a DCF-based tracker and a Kalman filter, allowing to calculate the position of the objects and re-identify them in case of occlusion. Finally, the data association is done in two steps, first, giving priority to the visual features of the objects and, second, to their movement predictions.

C2.2 The adaptation of the proposed algorithm for its execution on a low-power embedded system, an NVIDIA Jetson TX2, achieving an optimal balance between speed and precision. The system performs at a rate of 25 FPS on average with up to 45 simultaneous objects, thus making the application suitable for real-time processing.

C3. Extension of the algorithms for traffic and pedestrian monitoring applications

C3.1 We extend the system proposed in C1 for solving two real-life traffic applications: roundabout I/O (Input/Output) analysis and traffic anomalies detection. For roundabout monitoring, the system identifies the entry and exit taken by a vehicle and generates the I/O matrix. Metrics have been conducted with eight videos of roundabouts recorded from an Unmanned Aerial Vehicle (UAV) at 30 FPS with HD resolution and more than 1,000 vehicles in total with a success rate of 93%. To determine traffic anomalies the system generates the movement map of the road. The system triggers an alarm when an unusual event —low probability— occurs. The system is able to track successfully (without IDs losses) more than 90% of the objects and launch the corresponding alarm in more than 90% of the tested cases.

C3.2 To validate the system proposed in C2 for real-life pedestrian monitoring, we perform exhaustive metrics in the MOT16 dataset in which our system obtains a 51.1% in Multiple Object Tracking Accuracy (MOTA), delivering state-of-the-art results for embedded systems at an average rate of 25 FPS with up to 45 simultaneous objects.

Publications

All the contributions detailed above are included in the following publications, which comprise the scientific output of the research during this thesis:

Journals

- **M. Fernández-Sanjurjo**, B. Bosquet, M. Mucientes, and V. M. Brea. Real-time visual detection and tracking system for traffic monitoring. Engineering Applications of Artificial Intelligence, Vol. 85, pp. 410-420, 2019.
  Impact factor: 4.201 (JCR 2019).
  Journal ranked Q1 in JCR 2019.
  Category: COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE (33/136).
  Thesis contribution: C1, C3.

- **M. Fernández-Sanjurjo**, M. Mucientes, and V. M. Brea. Real-Time Multiple Object Visual Tracking for Embedded GPU Systems. IEEE Internet of Things Journal. Accepted.
  Impact factor: 9.936 (JCR 2019).
  Journal ranked Q1 in JCR 2019.
  Category: COMPUTER SCIENCE, INFORMATION SYSTEMS (3/156).
  Thesis contribution: C2,C3.

- B. Blanco-Filgueira, D. García-Lesta, **M. Fernández-Sanjurjo**, V. M. Brea and P. López. Deep Learning-Based Multiple Object Visual Tracking on Embedded System for IoT and Mobile Edge Computing Applications. IEEE Internet of Things Journal, Vol. 6, No. 3, pp. 5423-5431, 2019.
  Impact factor: 9.936 (JCR 2019).
  Journal ranked Q1 in JCR 2019.
  Category: COMPUTER SCIENCE, INFORMATION SYSTEMS (3/156).
  Thesis contribution: C2,C3.

Conferences

- **M. Fernández-Sanjurjo**, M. Mucientes, and V. M. Brea. A Real-Time Processing Stand-Alone Multiple Object Visual Tracking System *In Proceedings of the 18th Inter-*

national Conference on Computer Analysis of Images and Patterns (CAIP), p. 64-74, Salerno (Italy), 2019.
Conference Ranking (GGS Rating): B, Class 3.
Thesis contribution: C1.

- **M. Fernández-Sanjurjo**, M. Mucientes, and V. M. Brea. Real-Time Traffic Monitoring with Occlusion Handling *In Proceedings of the 9th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, p. 273-284, Madrid (Spain), 2019.
  Conference Ranking (GGS Rating): Work in Progress.
  Thesis contribution: C1.

- B. Blanco-Filgueira, D. García-Lesta, **M. Fernández-Sanjurjo**, V. M. Brea and P. López. Live Demonstration: Deep Learning-Based Visual Tracking of Multiple Objects on a Low-Power Embedded System. *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1, Sapporo (Japan), 2019.
  Conference Ranking (GGS Rating): B, Class 3.
  Thesis contribution: C2,C3.

Others

- Copyrighted Software: **M. Fernández-Sanjurjo**, M. Mucientes, V. M. Brea, D. Cabello, P. López. vTrack4VC: Tracking system for vehicle counting and classification, Reference: SC-210-2016, 2016.

1.7 Dissertation Structure

This dissertation is divided into six chapters. Chapter 2 analyzes the related prior work. Chapter 3 describes our RTMODT system for server architectures. Chapter 4 shows a first approach with embedded devices in which we propose a detection and tracking system. Chapter 5 explains our RTMODT system for embedded architectures taking as a starting point the work presented in the previous chapter. Finally, Chapter 6 presents the conclusions of this research. More specifically, the document structure is as follows:

- Chapter 2 introduces the evolution in the state-of-the-art of the different parts that may form an RTMODT system. This includes a review of object detection methods in images, going from classic background subtraction and hand-crafted-based algorithms to

more recent deep learning solutions. In the case of tracking, a study of the two main current trends is made. VOT algorithms follow an object from a single detection, exploiting only visual information, while MOT solutions deal with multiple objects in the scene through a tracking-by-detection approach.

- Chapter 3 introduces an RTMODT system that solves two of the most important challenges for a real-life tracking application: real-time operation with hundreds of objects and the total occlusions, which hinder the tracking performance. The system is based on three modules: detection, tracking, and data association. First, objects are identified through a deep learning-based detector. Second, tracking is performed with a combination of a Discriminative Correlation Filter and a Kalman Filter. This allows estimating the tracking error in order to make tracking more robust and reliable. Finally, the data association through the Hungarian algorithm combines the information of the previous steps. All this allows the system to reach a real-time processing speed while being robust to total occlusions and enabling more than 400 objects to be processed simultaneously. To demonstrate the usefulness of the system in real environments, the software has been extended for two traffic applications: anomaly detection and roundabout input/output analysis. The system has been evaluated with more than two thousand vehicles in real-life videos.

- Chapter 4 presents a portable solution for detection and tracking on embedded devices. This work is our first approach with embedded architectures. The system uses a background subtraction algorithm to detect moving objects and several instances of the GOTURN deep learning visual object tracker to follow the previously detected objects. The performance of the proposed algorithm is shown over a collection of representative sequences captured with the onboard camera as well as a study in terms of power consumption and velocity.

- Chapter 5 improves the limitations of the previous work to propose a novel RTMODT system for embedded GPU devices that extends a joint architecture of tracking and detection by adding a module comprised of appearance-based and movement-based trackers that allows maintaining the identity of the objects of interest for longer periods of time while alleviating the burden of the detector. Our system is mapped onto an embedded GPU platform, cutting down power consumption significantly with respect to a server GPU. Tracking performance metrics show a 51.1% in Multiple Object

Tracking Accuracy (MOTA) on the MOT16 dataset. This, in conjunction with a real-time processing speed of 25.2 FPS for up to 45 simultaneous objects and low power consumption of 15W, make our system an ideal solution for a wide range of video analytics applications.

- Finally, in Chapter 6 the conclusions of the research developed in this thesis are summarized. Moreover, this chapter discusses the future lines of research.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 2

RELATED WORK

Multiple object detection and tracking (MODT) systems are made up of two main modules: detection and tracking. In this chapter, we will review the state-of-the-art and the evolution of the major solutions to these fields. In Section 2.1 the three most relevant methods for identifying the objects of interest in a video image are presented: (i) background subtraction algorithms; (ii) hand-crafted-based detectors; (iii) deep learning detectors. Section 2.2 reviews the existing approaches that addressed object tracking. A review of Visual Object Tracking (VOT) algorithms is provided in Section 2.2.1. Section 2.2.2 tackles some of the most relevant Multiple Object Tracking (MOT) approaches. Finally, in Section 2.3 a review of hardware architectures for tracking is presented.

2.1 Detection

Every MODT system needs a mechanism to identify the objects of interest in an image. This is a well-studied problem in the field of computer vision and known as object detection. Over the years different solutions have appeared. To make the explanation more clear we will divide them into three different groups: background subtraction algorithms, detectors based on hand-crafted features, and detectors based on deep features.

2.1.1 Background Subtraction Algorithms

Background subtraction is a technique for detecting moving objects from a video stream. The general idea is, from an input video, for each frame, to be able to generate a binary

mask that divides each of the image pixels into two groups: foreground and background — Figure 2.1. The most trivial solution would be a simple frame difference between the current frame and the static frame of the background: those pixels that present a deviation above a given threshold will be identified as foreground and the remaining ones as background. This simplistic approach can give acceptable results in some controlled scenarios. However, several problems must be addressed if a better solution is to be provided [83]:

- Rapid changes in scene illumination: different times of day or clouds.
- Movement changes: camera oscillations.
- Different scenarios: outdoor and indoor.
- Accurate background modelling even if there are always foreground objects in all frames.
- Integration of new changes in the background model: e.g. parked cars.
- Background objects with dynamic movement: e.g. tree leaves.
- Camouflaged foreground objects: the color of an object similar to the background.

To address all these challenges a multitude of more sophisticated methods have been developed in the recent past. One of the best-known classical background subtraction solutions is Gaussian Mixture Models (GMM) [98]. Instead of attempting to model all the pixels of the image as a single distribution, each pixel is modeled as a mixture of Gaussians thus, providing a more robust representation. Determining which Gaussians may correspond to background processes is done by seeking out those with the greatest persistence and least variance. Once determined, those pixel values that do not fit the background distributions are considered foreground, and those that are detected as background are used to improve the model by an iterative update rule. This parametric approach has a very low memory requirement.

However, this solution has some disadvantages. Firstly, it considers that the background is always more frequently visible than the foreground and presents less variance, which is not always the case. On the other hand, the initial estimate of the parameters can be inaccurate and the value of the learning rate is difficult to establish correctly. Another problem is that in certain scenarios, it is not efficient to model the background with a small number of Gaussians.

An approach for dealing with some of the problems of parametric methods is SACON (SAmple COnsensus) [9], in which the background model is defined in a non-parametric

fashion. The value of each background model pixel is defined as a history of the most recent N values of that pixel using a FIFO strategy. To determine the value of a pixel the notion of "consensus" is applied, that is, a new pixel value will be classified as background if it matches most of the values stored in the pixel's model history.

In this way, the algorithm eliminates many of the parameters required by GMM-based solutions, making it more robust. However, the fact that the pixels are replaced using a FIFO strategy leads to problems in those situations where there are slow and fast movements in the background unless a high value of N is used, which increases the training time.

Another non-parametric approach that solves the problems of the FIFO replacement strategy is ViBe [5]. This method also stores for each pixel a set of past values at that same position or in the neighborhood. This set is used to determine if a new pixel belongs to the background and if so, the model is updated by randomly choosing which values to replace from the background model. Finally, when a pixel is identified as background it is propagated into the background model of a neighbor by another random selection method.

The PBAS [51] method improves some aspects of ViBe. Specifically, in ViBe the decision threshold and randomness parameters are set for all pixels. PBAS treats these parameters as state variables that can change over time separately for each pixel and introduces dynamic controllers for each of them.

Figure 2.1: Example of background subtraction. From an input video (left) the algorithm generates a model (right) that allows identifying the static part of the image or background (black) and the mobile objects or foreground (white). This allows the subsequent framing of the objects through the minimum bounding box that encloses each white blob. Images with Pixabay License [138].

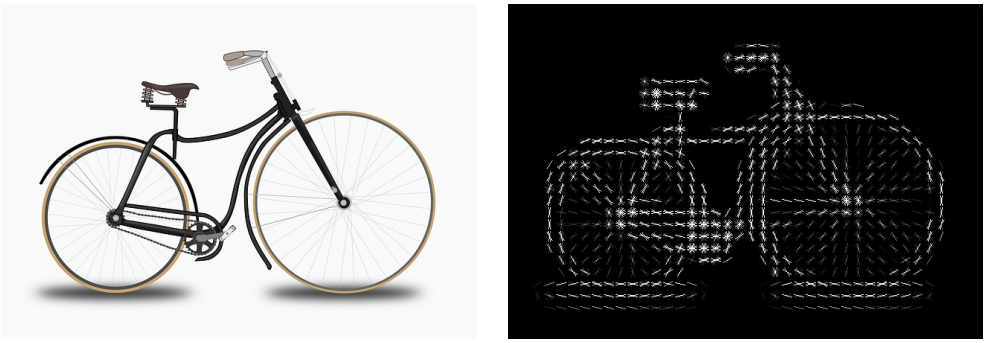


Figure 2.2: Example of HOG feature extraction. From an input image (left) the distribution of intensity gradients for each cell is calculated (right). Images with Pixabay License [138].

2.1.2 Object Detection with Hand-Crafted Features

In the field of category-specific object detection, two distinct time periods can be distinguished. On the one hand, before 2014 most of the solutions used hand-crafted features, and from that point, deep learning-based solutions dominate the state-of-the-art.

In the absence of deep robust features (such as those that would be obtained later through the use of neural networks), object detectors built complex feature representations for those objects that were to be identified and then detected objects by searching for those features in the image.

One of the first methods of this type was the Viola & Jones algorithm [107, 108]. This detector uses sliding windows to identify the objects in the image. This allowed testing all possible positions and scales in the image to determine where the objects of interest were located. The algorithm was focused on human faces. The authors used the Adaboost [36] algorithm to select the most representative features of the whole set to identify faces and exploited the concept of cascade detection, a multi-stage detection paradigm to reduce the computational overhead.

Another milestone in terms of feature representation was the Histogram of Oriented Gradients (HOG) algorithm [21]. This descriptor allows representing the appearance of objects from the distribution of intensity gradients or edge directions. This technique counts the number of occurrences of the gradient orientation in a series of locations in the image —Figure 2.2. This method differs from other solutions such as scale-invariant feature transform (SIFT) descriptors [75], edge orientation histograms, and shape contexts in that it is computed in a

grid of cells and uses overlapping normalization at the block level for improved accuracy. It was successfully used in detection and classification tasks in conjunction with Support Vector Machines (SVMs).

Probably the most advanced solution of the traditional object detectors was Deformable Part-based Model (DPM) [32]. This algorithm proposes a different direction to perform object detection. Instead of trying to find an effective representation of the object class and search it in the image, it proposes the detection as an ensemble of detections on different object parts, so the detector training consists of learning how to decompose an object in those parts. Since the original proposal, many improvements have been made [33, 34, 40]. Although the performance of DPM is much lower than current deep learning algorithms, DPM has influenced current detection methods with concepts like bounding box regression, mixture models, etc.

2.1.3 Deep Learning Object Detection

The improvement in the performance of hand-crafted features based detectors began to flatten out. However, in 2012 the re-introduction of Convolutional Neural Networks or CNNs [62] took place. Neural networks as object detectors in images are mainly based on consecutive convolution operations that extract features from the images, and a final classification in one of the object categories. As the convolution layers become deeper, the extracted features become more complex. They allow extracting more robust features than those obtained by traditional detectors. The first neural networks were adapted to classification problems triggered by the increasing performance of Graphics Processing Units (GPUs). The pioneering architecture was AlexNet [62]. The high performance of the classification task of CNNs was transferred to the object detection task. Deep learning object detectors can be divided into two categories: two-stage detection and one-stage detection. Two-stage object detectors perform a two-step approach: features extraction and region proposal. The first paper following this scheme was R-CNN [39]. Since then, the field of object detection through deep learning has become one of the most active in computer vision. R-CNN [39] includes a region proposal algorithm (such as selective search [104] or edge boxes [127]) and applies a classification network to each of them to extract features. Then, it runs a linear SVM stage to predict the presence of an object and its category. Since R-CNN calculates the features for each region proposal independently, this operation was redundant for each overlapped bounding box, affecting the detection time, and making the solution very slow.

Improving the previous approach, Fast-RCNN [38] introduces the regions in an intermediate stage of the network. While R-CNN inputs each region proposal to the CNN, Fast-RCNN extracts features from the entire image. It also applies a Region-of-Interest (RoI) pooling layer to obtain a feature map of fixed size from region proposals of diverse sizes. All these improvements resulted in a decrease in time and storage needed. However, Fast-RCNN continues resorting to selective search to propose RoIs resulting in a slow solution.

Faster-RCNN [90] introduces a region proposal algorithm based entirely on a neural network called the Region Proposal Network (RPN), a fully convolutional network that uses the information from intermediate layers of a standard classification network to provide different locations in which an object may appear. RPN increases the speed of producing region proposals because it shares convolutional features and layers with the detection network.

To improve the performance of the proposal of regions in all possible scales, Lin et al. [69] replicate the RPN from Faster-RCNN in several layers of the network in which deeper feature maps are combined with shallower ones by up-sampling and merging. The shallower the layer the smaller the object it will locate. This approach, called Feature Pyramid Network (FPN) provides outstanding results as shown in the MS. COCO detection challenge 2016 [71].

Many new solutions have emerged from this point. As an example, Grid R-CNN [76] implements a grid guided localization approach to capture the spatial information explicitly and exploit much more the position-sensitive property of convolutions. The authors design a multi-point supervision formulation and a two-stage information fusion strategy to this end.

Despite all the solutions proposed in recent years, architectures such as Faster-RCNN with FPN, despite having been well established in the field for some years, are still among the most used and most accurate in the task of object detection. They are usually combined with different backbones for feature extraction, like: VGG [96], ResNet [47] or ResNeXt [118] among others.

The other type of detectors are one-stage detectors, which directly propose boxes from input images without including a specific region proposal method. One of the first approaches of this type was You Only Look Once (YOLO) [86]. Unlike the two-stage solutions applies a single neural network to the full image by dividing the image into regions and predicting bounding boxes and probabilities for each region simultaneously. Improvements were made in successive versions [87], especially in YOLO v3 [88] where an FPN-like architecture allows detection of smaller objects. Another algorithm of this type is Single Shot MultiBox Detector (SSD) [74], which proposes multi-reference and multi-resolution detection techniques. One

of the most successful one-stage approaches is RetinaNet [70], which obtains comparable results to those of the two-stage detectors due to its new loss function named focal loss that changes the standard cross-entropy loss, putting more focus on hard misclassified examples during training. Some of the most recent solutions are RefineDet, FCOS, and YOLOv4.

RefineDet [121] comprises two modules: (i) the anchor refinement module that discards negative anchors to reduce search space for the classification step, and also adjust the locations and sizes of anchors to provide better initialization in the regression step; and (ii) the object detection module that further improves the regression accuracy and predicts multi-class label. In addition, RefineDet performs a transfer connection block to connect the features of the two modules.

The Fully Convolutional One-Stage (FCOS) [103] object detector employs an anchor free implementation to avoid the computation related to anchor boxes and their hyperparameters. In addition, FCOS overcomes previous one-stage detectors in MS COCO with the bonus of being considerably lighter.

YOLOv4 [12] is the latest version of the classic one-shot YOLO detector. In this version, it uses a better backbone —CSPDarknet53 [110]— that enhances the learning capability of the network. It also includes Path Aggregation Network (PANet) [72] instead of FPN (YOLOv3), and adds spatial pyramid pooling [46] to increase the receptive field and separate out the most significant context features.

As a summary of this Subsection, Figure 2.3 shows a roadmap with some of the most salient architectures throughout the last years.

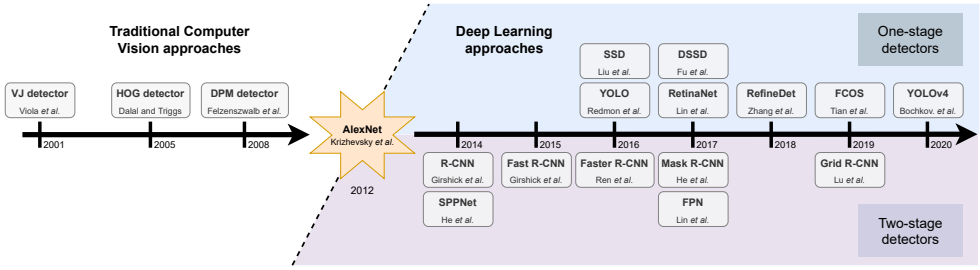


Figure 2.3: An object detection timeline showing some of the most relevant methods for each period. Classic hand-crafted milestones: VJ [107, 108], HOG [21], DPM [32]. Deep learning methods: AlexNet [62], R-CNN [39], Fast R-CNN [38], Faster R-CNN [90], YOLO [86], SSD[74], RetinaNet [70], Mask R-CNN [45], FPN [69], RefineDet [121], FCOS [103], YOLOv4 [12].

2.1.4 Analysis for a Real-Time Operation

Not all the solutions for object detection in images are equally feasible for an RTMODT system. Background subtraction algorithms have the advantage that they are fast and computationally inexpensive. At the same time, they do not require data to be available for training since they work on the video frames. They also detect any type of mobile object regardless of its category. However, as seen in Chapter 1 they present many problems in real-life applications due to their inaccuracy, and are limited to static cameras and detect only moving objects. This is the reason why background subtraction for the task of object detection is often relegated to controlled scenarios or situations where a lightweight solution is required.

Regarding the traditional algorithms for object detection based on hand-crafted features, these have been gradually replaced by deep learning approaches due to their superior performance. However, it is worth mentioning that not all convolutional networks are valid if real-time processing speed is required. Many of the solutions are too computationally expensive to reach this type of requirement, and those that offer enough speed usually do so at the expense of accuracy. To illustrate this problem, data extracted on state-of-the-art deep learning object detection has been used. At the time of this writing, the computation times and memory requirements of the top five Detectron solutions —which is a deep-learning-based image analysis platform by Facebook [117]— can be seen in Table 2.1. The results show that none of the approaches reaches a speed of at least 25 FPS despite the fact that this processing rate is met with high-end hardware. This is a limitation in the use of accurate solutions for real-time computing. In addition, as explained before, the detector would only be one piece of the system. This problem is aggravated in the case of developing MODT systems for embedded architectures. The data in Table 2.1 has been taken with an NVIDIA V100 GPU with a power consumption of 300 watts and 32GB of memory. This contrasts greatly with the 15 watts that would consume an embedded system like NVIDIA Jetson TX2. Therefore, the mapping of these algorithms to embedded systems is not possible in a direct manner if real-time is a requirement. This gives an idea of how far current research is from being fully exploited by low-power embedded systems, which are more suitable for edge IoT devices.

2.2 Tracking

An MODT system has to track the objects previously detected. This implies being able to maintain an estimation of their position and identity while they appear on the scene. Two

Architecture	Backbone	inf_t (s/im)	FPS	train_mem (GB)	boxAP
RetinaNet	R101	0.054	18.52	5.2	40.4
Faster R-CNN	R101-DC5	0.086	11.63	6.1	40.6
Faster R-CNN	R101-C4	0.139	7.19	5.9	41.1
Faster R-CNN	R101-FPN	0.051	19.61	4.1	42.0
Faster R-CNN	X101-FPN	0.098	10.20	6.7	43.0

Table 2.1: Top five configurations for object detection in Detectron2 [117] (Facebook). Architecture: Network used —RetinaNet [70] or Faster R-CNN [90]. Backbone: R101 —ResNet-101; X101 —ResNeXt-101; FPN —use a ResNet+FPN backbone with standard convolution and FC heads for mask and box prediction, respectively; C4 —use a ResNet conv4 backbone with conv5 head; DC5 (Dilated-C5) —use a ResNet conv5 backbone with dilations in conv5, and standard convolution and FC heads for mask and box prediction, respectively. inf_t (s/im): inference time in seconds per image. FPS: inference rate in frames per second. train_mem (GB): Memory needed for training in GB. boxAP: box Average Precision (%) in MS. COCO Object Detection Challenge [71].

main trends in the computer vision field try to solve this problem with different approaches: Visual Object Tracking (VOT) and Multiple Object Tracking (MOT).

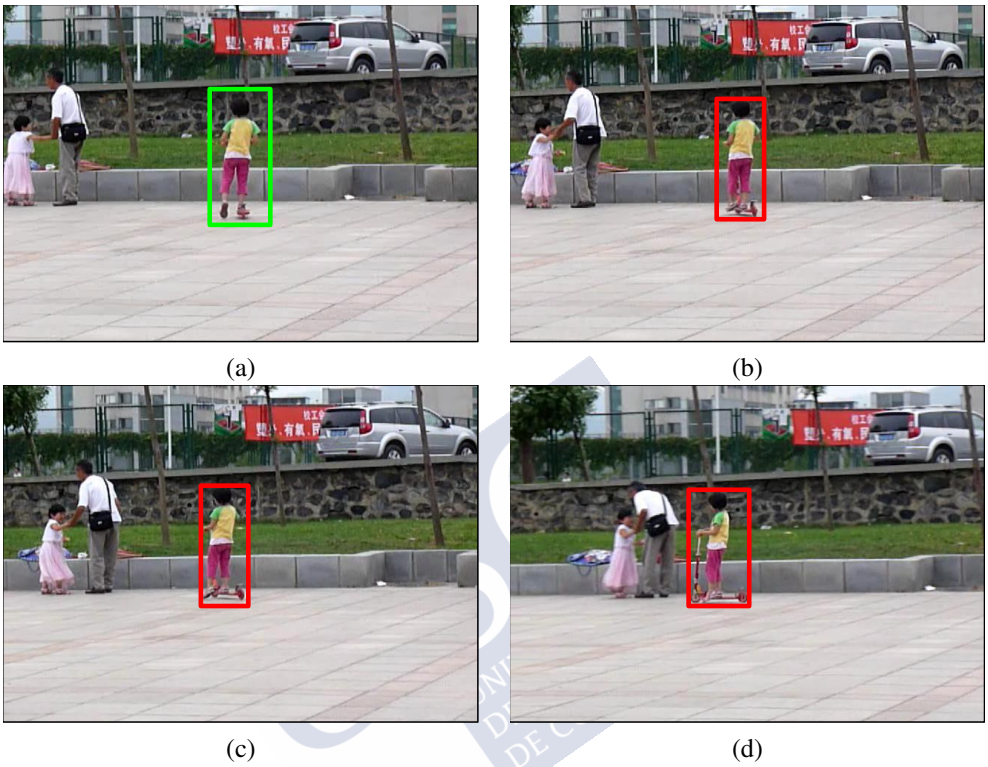
Both approaches have advantages and disadvantages when used in a Multiple Object Detection and Tracking System. In Section 2.2.1, we discuss the state of the art of VOT in recent years and their strengths and limitations for being used in an MODT system. Section 2.2.2 will present the state-of-the-art for MOT.

2.2.1 Visual Object Tracking

VOT algorithms, once initiated with a detection bounding box, estimate the position of the object in the new frame, exploiting just the visual information. They have an appearance model composed of visual features of the object. The target location is estimated in subsequent frames without the need for further detections. These approaches are focused on a single target. An example of the operation of an VOT algorithm can be seen in Figure 2.4.

A significant part of the research in VOT algorithms is present in the VOT challenge [57, 58, 59, 60, 61]. Most of the algorithms in this contest are model-free, short term, and causal:

- Model-free: the only training of the tracker will be a target bounding box at the initial frame.



(a) (b)

(c) (d)

Figure 2.4: VOT example. In (a) the tracking is initialized with a provided bounding box (in green). In the following frames (b), (c) and (d) the tracker estimates the position of the object adjusting its bounding box accordingly.

- Short term: re-detection of the object once lost is not possible. Therefore the task of the tracker is to report the bounding box until the target is lost or disappears from the scene.
- Causal: the tracker will not be able to use future information to predict the position of the target.

One of the most important trends in VOT has been trackers based on Discriminative Correlation Filters (DCFs). The general idea of this type of algorithms is that, in each frame, the tracker will estimate the new position of the object from its position in the previous frame. Therefore, the input of a DCF is the bounding box of the object in the initial frame, which is

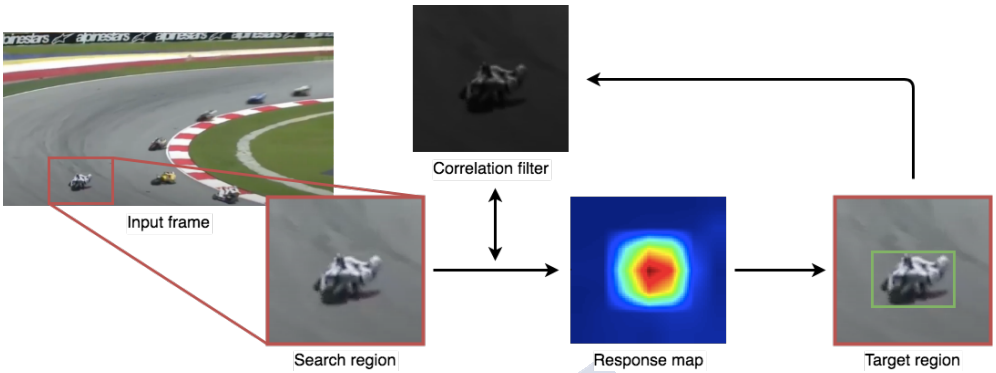


Figure 2.5: DCF tracking example. The response map is calculated from the maximum value of the correlation between the search region and the filter. The target center will be represented by the maximum value of the response map.

known as the target patch. From this patch, a feature representation of the object is extracted —which varies according to the algorithm—, then a correlation operation is applied, and the response map between the initial filter and the extracted features is calculated. This operation is done in the frequency domain using a Fast Fourier Transform (FFT). From the response map, a confidence map is obtained by applying the Inverse FFT (spatial domain). The maximum value of this response map represents the new position of the object (new center of the patch). Once the new position of the object is established, the new features are extracted and the correlation filters are modified to progressively update the appearance of the object —Figure 2.5. The challenges of this type of tracker include: being robust to sudden changes in object appearance, robust feature selection, changes in scale, filter size selection, etc.

One of the first DCF-based proposals was MOSSE [13], which performed the tracking task by attempting to minimize the sum of the squared error between the desired output and the actual output. This filter uses grayscale values of the image and performs random transformations on the original image patch in order to increase robustness against appearance changes.

Since MOSSE, many tracking algorithms have been based on DCFs. One of them is KCF [49], which has more robust features than the intensity pixel values, i.e. Histogram of Oriented Gradients (HOG) features, combined with a Gaussian kernel function to differentiate object and background. KCF has good accuracy and high speed due to exploiting the properties of circular arrays and kernel functions. KCF has served as the foundation for many subsequent

tracking approaches [19, 31, 77].

The trackers discussed so far only solve the problem of estimating the translation of the object in the successive frames, ignoring completely its change in scale. This imposes limitations on the use of these algorithms in videos, where objects undergo a considerable scale change. The first tracker that addressed this problem was Discriminative Scale Space Tracking (DSST) [24]. In this approach, translation and scaling are estimated by different filters. First, the translation of the object is calculated, then a trained scale filter is applied to the newly calculated position. This allows approximating the bounding box of the object, reflecting the relative change in scale compared to the previous frame.

Subsequently, regularization strategies have been brought to fix a number of the issues of the preceding algorithms which, include:

- Limited detection range: identical size for the filter and the object patch.
- Errors in irregular objects: they can learn the model from the background rather than the object.
- They are restricted to a fixed search region.
- Errors with deformable objects due to training samples.

One of the ways to enhance these algorithms is to use regularization. An example of this is the Spatially Regularized DCF (SRDCF) algorithm [26], which introduces spatial regularization that weakens the background, giving additional strength to the object.

The success of CNNs for detection and classification tasks led to a great interest in their use for the tracking task. CNNs allow computing more robust features than the hand-crafted ones employed in classical DCFs. For example, SRDF was modified by using deep CNN features in deepSRDCF [25]. However, the first solutions that used DCFs in conjunction with deep learning features did not experience the expected performance improvement. The main reason was due to not combining feature maps of different resolutions, something solved later by the Continuous Convolutional Operators for Tracking (CCOT) algorithm [27]. CCOT introduced multi-resolution feature maps. It learns a series of convolution filters to create the target's confidence map in the continuous-domain. This allows an integration of multi-resolution feature maps to estimate the location of the target.

Efficient Convolution Operators (ECO) [22] is the upgraded CCOT variant. Instead of learning a representation from a large range of filters, ECO comprises a smaller set of fil-

ters through matrix factorization. ECO includes a Gaussian Mixture Model (GMM) to reflect different appearances rather than learning consecutive appearances as CCOT, therefore preventing overfitting.

There are also some recent solutions such as the ATOM [23] tracker that works by alternating between a target classification module that is learned online, a target estimation module that is learned offline, and an updating step, and it delivers state-of-the-art results.

In parallel with DCF-based trackers, another type of tracking that has been gaining a lot of presence in the state of the art in the last years are those based on siamese convolutional neural network architectures. The goal is to decide if the two image patches that are entered into the network contain similar objects or not. The network measures the similarity between the two inputs and has the ability to jointly learn features and similarity. At first, the concept of the Siamese network was used for signature authentication and recognition of fingerprints and later was introduced to the tracking task.

SiamFC [8] is one of the first approaches of this kind. This tracker consists of two branches that apply an identical transformation —deep features extractor— to two inputs: the search image and the exemplar. Then, both representations are combined through cross-correlation, generating a score map that indicates the most probable position of the object —Figure 2.6. In Li et al. [67] SiamRPN is proposed, adding a Region Proposal Network (RPN) to a Siamese network in order to generate bounding box proposals that go through classification and regression branches. DaSiamRPN [126] improves SiamRPN, focusing the training on semantic distractors, and adding a search region strategy for long-term tracking. These trackers showed excellent performance in tracking metrics in the VOT challenge [60, 61].

Analysis for a real-time operation

When using an VOT algorithm for real-time multiple object detection and tracking systems there are a number of limitations. First, many of the state-of-the-art algorithms do not work in real-time. The inclusion of a section on real-time since the 2017 VOT Challenge emphasizes the importance of the real-time operation. It should also be noted that many of those solutions that achieve a high frame rate do not provide a framework for dealing with multiple objects. That is, computation times do not scale optimally when adding more than one target. Finally, many deep learning solutions require GPU computing, which can be a problem in the development of complete systems by having the detector and tracker competing for GPU resources.

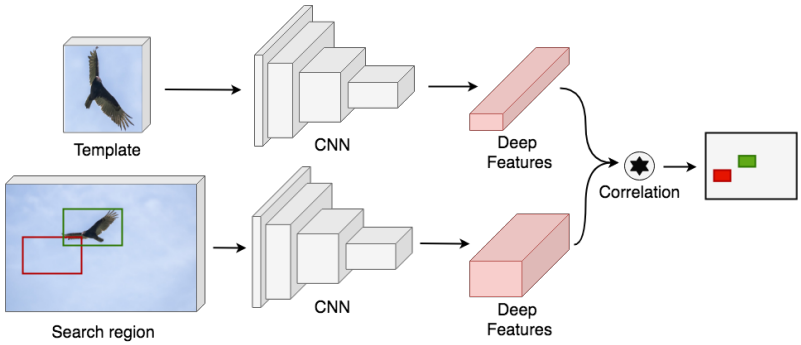


Figure 2.6: Basic framework for object tracking with siamese networks [8]. The algorithm attempts to locate a smaller template image within a search region. This is done by first producing learned embeddings for both these images and then cross-correlating these features. The resultant is a spatial score map containing confidences of presence of the template within the search region.

For all these reasons, in many scenarios for complete systems, classic solutions such as KCF or DSST trackers are still being used due to their great speed and versatility, which allows to integrate them in a full pipeline, as well as to adapt them for the tracking of multiple objects.

2.2.2 Multiple Object Tracking

Multiple Object Tracking (MOT) is defined as the task of identifying and tracking objects without any prior knowledge of the number of targets or their appearance. Unlike algorithms for object detection, which provide a series of rectangular bounding boxes (detections) defined by their coordinates, height, and width, MOT algorithms also associate a target ID to every detection bounding box, so as to differentiate among intra-class objects –Figure 2.7.

Tracking-by-detection is the most straightforward technique for MOT algorithms: a series of detection bounding boxes are derived from the video frames and form the input to the tracking process, which assigns the same ID to bounding boxes representing the same object. In order to ease the comparison between different MOT solutions, several MOT datasets have a standard collection of detections that allow evaluating the efficiency of the algorithm that associates detections exclusively, as the performance of the detector can have a substantial effect on the outcomes of the tracking. The most widely used benchmark for multiple object

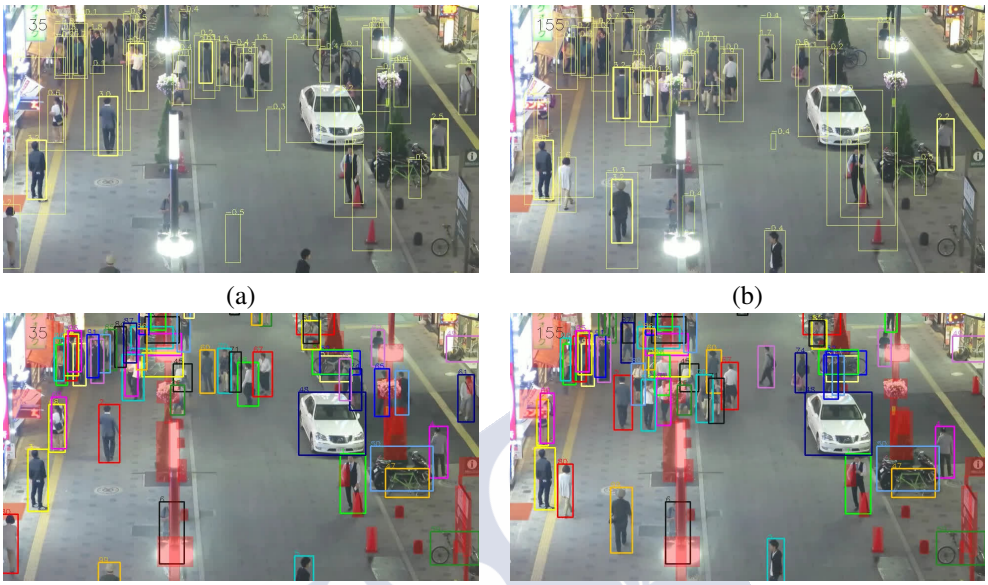


Figure 2.7: Example of how MOT algorithms perform the tracking task. In (a) and (b) the result of the object detection process is shown —numbers at the top of the bounding represent the detector confidence. In (c) and (d) the outcome of an ideal tracking is displayed, where each object has a unique ID and it is kept as long as it is visible.

tracking is MOTChallenge [136]. It provides a large number of videos with detections for the task of pedestrian tracking.

There is a great variety of MOT algorithms in the state of the art. In most of them, it is possible to observe the following stages: detection stage, motion estimation/feature extraction stage, affinity stage, and association stage.

The detection stage provides the bounding box of the objects present in each frame of the video. In the motion estimation and/or feature extraction phase the algorithm extracts information about the movement or the appearance of the objects and makes the necessary predictions. The affinity phase establishes a measure to associate detections and trackers. This measure can be either similarity (appearance) or distance (motion). Finally, in the association stage, it is necessary to make the assignment between detections and trackers to put the same ID to the detections of the same object while it appears on the scene. This thesis focuses on the stages where the greatest advances have been made in the MOT task. These are motion

estimation and feature extraction. The affinity phase is directly linked to these since the new extracted information is the one used to establish a similarity metric between different objects. The detection stage has already been addressed in Section 2.1 and it is usually independent of the tracking algorithm, as these are usually developed to be measured in benchmarks such as the MOTChallenge, and the detections are given in advance. Keeping the same detections for all the algorithms allows to correctly evaluate the tracking in isolation. The association stage usually resorts to classic solutions as the Hungarian algorithm [63]. An example of the different stages of an MOT algorithm can be seen in Figure 2.8.

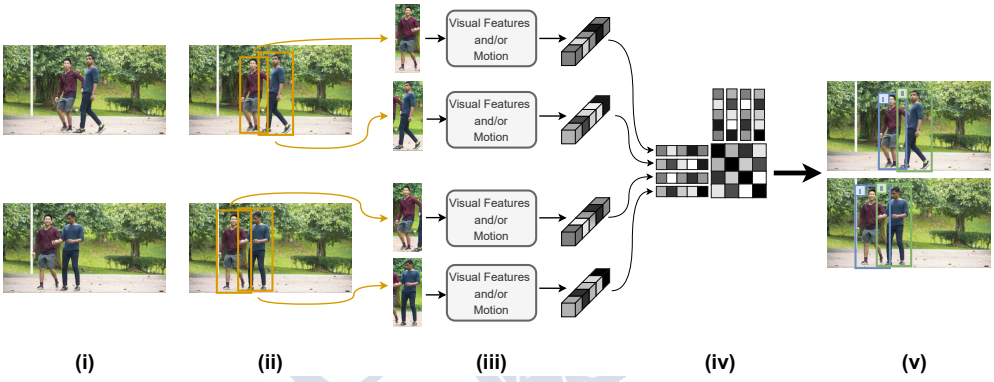


Figure 2.8: The usual operation of an MOT algorithm. Given the raw frames of a video (i), object detection is performed to obtain the bounding boxes of the objects of interest (ii) —usually, these detections are provided by an MOT benchmark like MOTChallenge [136]. Then, for each detection visual features and/or motion information is obtained (iii). Using this information an affinity measure is established, e.g. distance function, to estimate the probability of two detections corresponding to the same target (iv). Finally, the data association assigns the corresponding ID to each object.

Motion prediction

One of the most commonly used solutions in the field for motion estimation are those based on Bayesian filters. Despite not supporting multi-modal distributions, the most widely used is the Kalman filter [53], which produces an estimate of the position of the object based on its previous state and the current measurement [6, 10, 68, 91, 114].

A more complex Bayesian solution that allows working with multi-modal distributions is the Particle Filter [1]. Their drawback, however, is a higher computational load. Thus, for

high-dimensional state spaces, particle filters are not ideal as their computational cost grows exponentially with the number of state parameters [16, 119].

Nowadays, with the advent of deep learning, new alternatives for motion estimation have emerged. One of these is recurrent neural networks (RNN), like the Long Short-Term Memory (LSTM), which are often used to learn more sophisticated, non-linear motion models from data. In [93], a model for each detection based on three RNNs to compute different features is proposed. The second of these was an LSTM trained to predict the motion model of each object, i.e., their velocity vector. The other two RNNs learned the appearance of the objects and the interactions between them. Another algorithm with an LSTM for motion estimation is the one proposed in [2]. The LSTM is responsible for predicting the new position and size of the bounding box for each object in the image. This is done by taking into account the information about the position and the speed of the objects in the previous time instants.

Another solution to estimate the movement of objects is by using a DCF. A proposal of this type is shown in [111], where classical DCF affinity was combined with optical flow, motion, and scale affinities. Another alternative is shown in [123], where the DCF inputs are features extracted from a CNN previously reduced in dimensionality with Principal Component Analysis (PCA).

Feature extraction

One of the issues that adds robustness to an MOT system is a robust feature representation of the tracked objects. This allows to create more powerful similarity metrics and avoid mismatches, enabling subsequent re-identification of the object once it is lost or occluded. Although there are algorithms that provide satisfactory results without a model of the appearance of the objects [10, 11] , using visual features improves the final results.

The feature extraction phase has evolved, as have other areas of computer vision, to reflect the advances in deep learning. Some of these algorithms focus on extending traditional MOT approaches using CNNs. In this line, Kim et al. [56] and Chen et al. [15] propose extensions to the classical multiple hypotheses tracking (MHT) [89]. The former introduces on-line appearance representations, while the latter enhances the detection model of classical MHT.

SORT [10] is one of the most widely used algorithms despite its simplicity due to its high speed. It does not use any feature representation, resulting in many ID switches. This has been addressed with DeepSORT [114], where an appearance model of the objects is added by means of a CNN which provides a 128-component vector representation. This allows for the

calculation of the affinity between detections and trackers using deep features, considerably reducing the number of ID switches.

The solution proposed in [17] also includes a CNN to model the appearance of the objects. In this case, a region-based fully convolutional neural network (R-FCN) [20] is used to predict potential candidates for detection using information from existing trackers. Then, features are extracted at those positions through a trained GoogLeNet [102], and these are used for data association to increase the robustness of the association and the reidentification (re-ID) of objects.

Another algorithm that provides excellent results in the MOTchallenge benchmark with CNNs to model object appearance is JDE [113]. It proposes an architecture in which the task of detection and embedding appearance are learned in a single network. Specifically, this appearance modeling is integrated into a single shot detector that provides both the bounding boxes of the detected objects and their embedding vector. This vector is also used for object association and re-ID tasks, obtaining great performance at a high video rate. This solution employs an anchor-based approach —a variation of YOLOv3 [88] architecture. Some problems of this approach are that it gives priority to the detection task over the re-ID since one occurs only after the other. Also, an anchor could contain multiple object instances, so it will be forced to predict conflicting classes. Finally, different anchors with different image patches could be responsible for predicting the same identity. To solve this problem FairMOT [122] proposes an anchor-free approach that extracts re-ID features only at the object center unlike JDE, which extracts re-ID features at the centers of all positive anchors. It also gives the same importance to the detection and re-ID tasks, resulting in improved performance.

Analysis for a real-time operation

Despite the large number of MOT approaches that appear each year, many of them are far from being able to be integrated into an RTMODT system. MOT algorithms are divided into batch and online methods. Batch methods can obtain information from future frames to determine the identity of objects, getting better results. Online methods, however, do not have access to this information, but can only access past and present information. Their performance is often worse since they cannot correct past errors using future information. For an RTMODT system, only online solutions are suitable, since the only information available at any given moment is the current frame and the tracking information of previous frames.

Indeed, analyzing the results from the MOTChallenge [136] —the most important MOT

Ref.	Name	MOTA	IDF1	MOTP	Hz.	Detector	Online
[122]	FairMOT_V2	74,9	72,8	81,2	25,4	priv.	yes
[122]	Fair	69,3	72,3	80,2	26,0	priv.	yes
[124]	TAP	64,8	73,5	78,7	39,4	priv.	yes
[65]	MCMOT_HDM	62,4	51,6	78,3	34,9	priv.	no
[79]	smartSORT	60,4	56,1	78,9	28,0	priv.	yes
[10]	SORTwHPD16	59,8	53,8	79,6	59,5	priv.	yes
[11]	IOU	57,1	46,9	77,1	3004,6	priv.	no

Table 2.2: Algorithms submitted to MOT16 Challenge that operate in real-time –i.e. equal to or greater than 25FPS. For each algorithm it is displayed the scientific publication on which it is based, its name in the MOT16 leaderboard, the main MOT metrics (MOTA, ID F1 Score and MOTP), its processing speed (in frames per second excluding the detector) and, finally, we show if the detector uses public or private detections, and if it is an online method.

Ref.	Name	MOTA	IDF1	MOTP	Hz.	Detector	Online
[122]	Fair	73,7	72,3	81,3	25,9	priv.	yes
[97]	GMPHDOGM17	49,9	47,1	77,0	30,7	publ.	yes
[3]	GMPHD_Rd17	46,8	54,1	76,4	30,8	publ.	yes
[54]	PHD_LMP	45,9	42,5	76,6	29,4	publ.	yes
[11]	IOU17	45,5	39,4	76,9	1522,9	publ.	no
[10]	SORT17	43,1	39,8	77,8	143,3	publ.	yes
[30]	GM_PHD	36,4	33,9	76,2	38,4	publ.	yes

Table 2.3: Algorithms submitted to MOT17 Challenge that operate in real-time –i.e. equal to or greater than 25FPS. For each algorithm it is displayed the scientific publication on which it is based, its name in the MOT17 leaderboard, the main MOT metrics (MOTA, ID F1 Score and MOTP), its processing speed (in frames per second excluding the detector) and, finally, we show if the detector uses public or private detections, and if it is an online method.

dataset, the vast majority of the solutions do not reach a real-time processing speed of 25 FPS. At the date of writing this doctoral thesis, the algorithms that do work in real-time in the MOT leaderboard for the datasets MOT16, MOT17, and MOT20 are shown in Tables 2.2, 2.3 and 2.4. Those submissions lacking a scientific publication have been discarded due to the impossibility to verify their reliability or the methods they use. The results show that only 10 different algorithms reach real-time computation speed.

Of these algorithms that reach a computation speed higher or equal to 25 FPS, it should be mentioned that:

Ref.	Name	MOTA	IDF1	MOTP	Hz.	Detector	Online
–	MOTSOT	68,6	71,4	79,5	224,0	priv.	yes
–	GMPHD_Rd20	44,7	43,5	77,5	25,2	publ.	yes
–	Surveily	44,6	42,5	76,1	29,8	publ.	yes
[10]	SORT20	42,7	45,1	78,5	57,3	publ.	yes

Table 2.4: Algorithms submitted to MOT20 Challenge that operate in real-time –i.e. equal to or greater than 25FPS. For each algorithm it is displayed the scientific publication on which it is based, its name in the MOT20 leaderboard, the main MOT metrics (MOTA, ID F1 Score and MOTP), its processing speed (in frames per second excluding the detector) and, finally, we show if the detector uses public or private detections, and if it is an online method.

- The reported times of systems using public detections do not include the time of the detector but simply the times of the tracking part in isolation. This practically rules them out for real-time use because most CNN detectors do not work in real-time, and the MOT solutions that do it almost all of them propose a joint tracking and detection architecture. Therefore, the fact that both parts are considered separately will suppose a considerable decrease in the number of FPS regardless of the detector.

- References [3, 10, 11, 30, 54, 97, 122] use private detections, something that does not affect the real-time requirement but makes their results not comparable with solutions that use public detections, since it is known that the quality of the detections has a great impact on the tracking performance.

- Some approaches such as [10, 11] are trackers that do not employ visual information and do not exploit the potential of the CNNs. Therefore, their performance is lower than other solutions with a model of the appearance of the objects for affinity computation.

- References [11, 65] are not online solutions, so their use for a real-time system is ruled out.

- None of the solutions presented has hardware restrictions. This implies that the reported times might decrease drastically with a device with fewer resources such as a low consumption embedded system and, therefore, no longer be a viable real-time solution.

Taking into account the above considerations, it is clear the difficulty in designing a complete system for detecting and tracking multiple objects in real-time, which becomes even more difficult on the edge, on devices with limited resources.

2.3 Computer Vision on the Edge

CNNs offer increasing accuracy for visual tasks at the expense of huge computation and memory resources. Current hardware solutions can satisfy these demands at the cost of high energy consumption, especially when real-time processing is also a requirement, understood as the capability of solving the application problem using at any time only the available frame provided by the live camera without intermediate storage for delayed processing. That is the case of edge computing solutions, where the computation is performed close to the data source in order to avoid cloud transfer and thus improve response time [95]. Edge computing also benefits from energy savings due to no cloud transfer need, but it also requires embedded low-power consumption nodes. Thus, in order to exploit the state-of-the-art CNNs for real applications at IoT end-nodes, low-power embedded solutions must be explored whereas maintaining reasonable accuracy and performance.

Diverse hardware solutions for CNN inference acceleration have been proposed in recent years. They range from standalone solutions to heterogeneous systems and Systems-on-Chip (SoC), which can include field-programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), CPUs, and graphics processing units (GPUs). Focusing on visual tasks-oriented proposals, those based on FPGAs stand out in terms of energy efficiency but not in performance [43, 82]. Additionally, some of them are supported in other elements as CPUs or external DRAM memory [41, 78, 85, 94], the networks used as benchmark are not always representative [109, 120] or their price limits the application range [9, 84]. In terms of ASICs, several accelerators for CNNs as an alternative to CPUs and GPUs for computer vision tasks as image classification [18], face detection [4] and recognition [29], recently have been presented. They deliver a very low power consumption but long development cycles, high cost, and little flexibility to adapt to the rapid progress of the algorithms. Regarding CPUs, Intel Knights Landing [133] and Intel Knights Mill [134] generations of Xeon Phi x86 CPUs are optimized for deep learning but conceived for training in supercomputers, servers, and high-end workstations. Intel also offers the Aero Compute Board, a purpose-built unmanned aerial vehicle (UAV) developer kit powered by a quad-core Intel Atom processor.

NVIDIA, one of the most important GPU manufacturers, has also increased its offering for deep learning solutions rapidly [129]. NVIDIA GPUs for deep learning cover data center solutions (DGX systems and Tesla solutions), desktop development (Titan Xp, Quadro GV100, Titan V), and embedded applications (Jetson family). NVIDIA DGX systems are fully integrated solutions built on the NVIDIA Volta GPU platform but server-oriented. NVIDIA

DGX-2 contains 16 Tesla V100 GPUs consuming 10 kW whereas NVIDIA DGX-1 is based on 8 Tesla V100 GPUs with 3.5 kW consumption. Facebook's Big Sur [132] and more recent Big Basin [135] custom deep learning server, with 8 NVIDIA Tesla P100 GPU accelerators, is similar to NVIDIA DGX-1.

Although the application at hand will ultimately dictate the hardware platform to run computer vision on the edge of the internet of things, some issues must be considered. As commented above, stand-alone multiple object tracker systems typically [35, 114] comprise object detection, one-object tracking, predictive filters like Kalman solutions [53] to handle occlusions, and data association to deal with several objects on the scene [63, 115]. State-of-the-art object detectors and one-object trackers are based on deep learning [14, 73, 126], and as such demand large memory capacity, along with high-end GPUs not to yield very long detection or tracking times that might render the system useless for video-rate processing. Predictive filters and data association, however, are not so demanding, and they can run on CPUs. Still, many core architectures provided with many threads are desirable to parallelize tasks, and track as many objects as possible at a time at video rate processing. In this line, [35, 114] are solutions implemented on server GPUs and many core CPU architectures for tracking several objects at video rate processing. Nevertheless, such a performance cannot be reached on low footprint and power embedded devices, more suitable for edge computing. ASICs and FPGA devices are the best option for embedded computer vision with low footprint and power consumption [43, 92, 94, 106, 112]. Both, however, suffer from long design cycles and less on-chip programming flexibility, which might make not to take advantage of the latest contributions from the very rapidly evolving field of deep learning for computer vision. This delay between state-of-the-art computer vision solutions and hardware implementations leads to a loss of tracking performance of an embedded approach of this type.

Off-the-shelf single board computers do not have the capacity to hold a stand-alone multiple object detection and tracking system at video frame rate, and they have mainly been used for classification tasks [105]. Thus, embedded GPUs naturally emerge as the best candidate to run our stand-alone multiple object tracker at video rate processing among the plethora of platforms and devices for computer vision on the edge, with the benefits of low footprint and power consumption.

As a result of the previous comments, we have decided to use as embedded hardware for our development the NVIDIA Jetson TX2, a Pascal GPU architecture which consists of a quad-core 2.0 GHz 64 bit ARMv8 A57 processor, a dual-core 2.0 GHz superscalar ARMv8

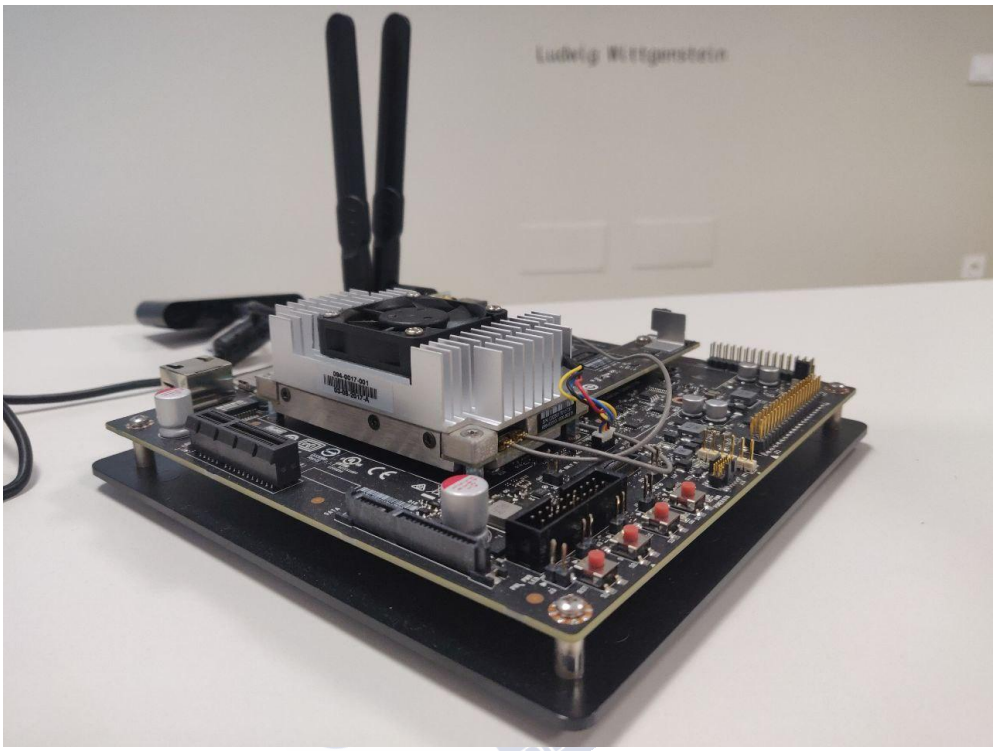


Figure 2.9: Jetson TX2 development kit with its integrated GPU was selected as our embedded device for integrating our embedded developments.

Denver processor, and an integrated Pascal GPU 1.3 GHz with 256 cores —Figure 2.9. The six CPU cores and the GPU share 8 GB DRAM memory. It is power-efficient, with small dimensions and high throughput for embedded applications. Whereas discrete GPUs consumption ranges between 150 and 250 W, integrated GPU as on the Jetson TX2 swings between 5 and 15 W. Additional advantages are the reduced size and needless active cooling. It is also commercially available as a development kit [137], which also includes a 5MP CSI camera and WLAN and Bluetooth connectivity among other peripherals.

As seen in Table 2.5, some other alternatives of the Jetson family were also possible. Nevertheless, throughout the time of this Thesis, Jetson TX2 presented the best trade-off between price and hardware performance metrics.

Feature	Jetson Nano	Jetson TX1	Jetson TX2/TX2i	Jetson AGX Xavier
CPU (ARM)	4-core ARM A57 @ 1.43 GHz	4-core ARM Cortex A57 @ 1.73 GHz	4-core ARM Cortex-A57 @ 2 GHz, 2-core Denver2 @ 2 GHz	8-core ARM Carmel v.8.2 @ 2.26 GHz
GPU	128-core Maxwell @ 921 MHz	256-core Maxwell @ 998 MHz	256-core Pascal @ 1.3 GHz	512-core Volta @ 1.37 GHz
Memory	4 GB LPDDR4, 25.6 GB/s	4 GB LPDDR4, 25.6 GB/s	8 GB 128-bit LPDDR4, 58.3 GB/s	16 GB 256-bit LPDDR4, 137 GB/s
Storage	MicroSD	16 GB eMMC 5.1	32 GB eMMC 5.1	32 GB eMMC 5.1
Tensor cores	–	–	–	64
PCI-Express lanes	4 lanes PCIe Gen 2	5 lanes PCIe Gen 2	5 lanes PCIe Gen 2	16 lanes PCIe Gen 4
Power	5W / 10W	10W	7.5W / 15W	10W / 15W / 30W

Table 2.5: Hardware comparison for Jetson modules.

Chapter 3

Real-Time Multiple Object Detection and Tracking System for Server Architectures

RTMODT systems represent an essential tool for a broad range of surveillance applications. Two of the most noteworthy challenges for these systems are the real-time operation with hundreds of targets and the total occlusions which hinder the tracking of the objects. In this chapter, we present an RTMODT approach for server architectures that deals with these two challenges based on three modules: detection, tracking and data association. First, objects are identified through a deep learning based detector. Second, tracking is performed with a combination of a Discriminative Correlation Filter and a Kalman Filter. This permits to estimate the tracking error in order to make tracking more robust and reliable. Finally, the data association through the Hungarian algorithm combines the information of the previous steps. To show the applicability of the system in real-life scenarios, we present two traffic application scenarios: anomaly detection in traffic monitoring and roundabout input/output analysis. The evaluation of the system has been performed with real-life video data sets with over 2,000 vehicles. The contents of this chapter are extracted from the following publications:

> M. Fernández-Sanjurjo, B. Bosquet, M. Mucientes, and V. M. Brea. Real-time visual detection and tracking system for traffic monitoring. Engineering Applications of Artificial Intelligence, Vol. 85, pp. 410-420, 2019.

M. Fernández-Sanjurjo, M. Mucientes, and V. M. Brea. A Real-Time Processing Stand-Alone Multiple Object Visual Tracking System *In Proceedings of the 18th International Conference on Computer Analysis of Images and Patterns (CAIP)*, p. 64-74, Salerno (Italy), 2019.

M. Fernández-Sanjurjo, M. Mucientes, and V. M. Brea. Real-Time Traffic Monitoring with Occlusion Handling *In Proceedings of the 9th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, p. 273-284, Madrid (Spain), 2019.

3.1 System Overview

Our system is made up of three blocks (Figure 3.1): detection, tracking, and data association. To detect objects in an image, we use a deep learning-based detector. For tracking, we combine a DCF-based tracker with a Kalman-based one, which enables to calculate a failure detection metric to identify occluded objects. Finally, in the data association module, we assign each detection with its correspondent tracker through the Hungarian method [63, 115] and perform an update of the trackers.

Algorithm 1: Steps of MODT system.

Require:

(a) Im_t: Image frame at current time t

(b) $\Phi_{t-1} = \{\varphi^1_{t-1}, \varphi^2_{t-1}, \ldots, \varphi^n_{t-1}\}$

Function `Main(`Im_t, Φ_{t-1}`)`**:**
  $\overline{\Phi}_t$ = `Tracking_Prediction(`Φ_{t-1},Im_t`)`
  if $time_elapsed > \tau$ **then**
    Ψ_t = `ConvNet_Detect()`
    Φ_t = `Tracking_Update(`$\overline{\Phi}_t$,Ψ_t,Im_t`)`
  else
    $\Phi_t = \overline{\Phi}_t$
  return Φ_t

Algorithm 1 presents the main steps of the system. The inputs to the system at every time

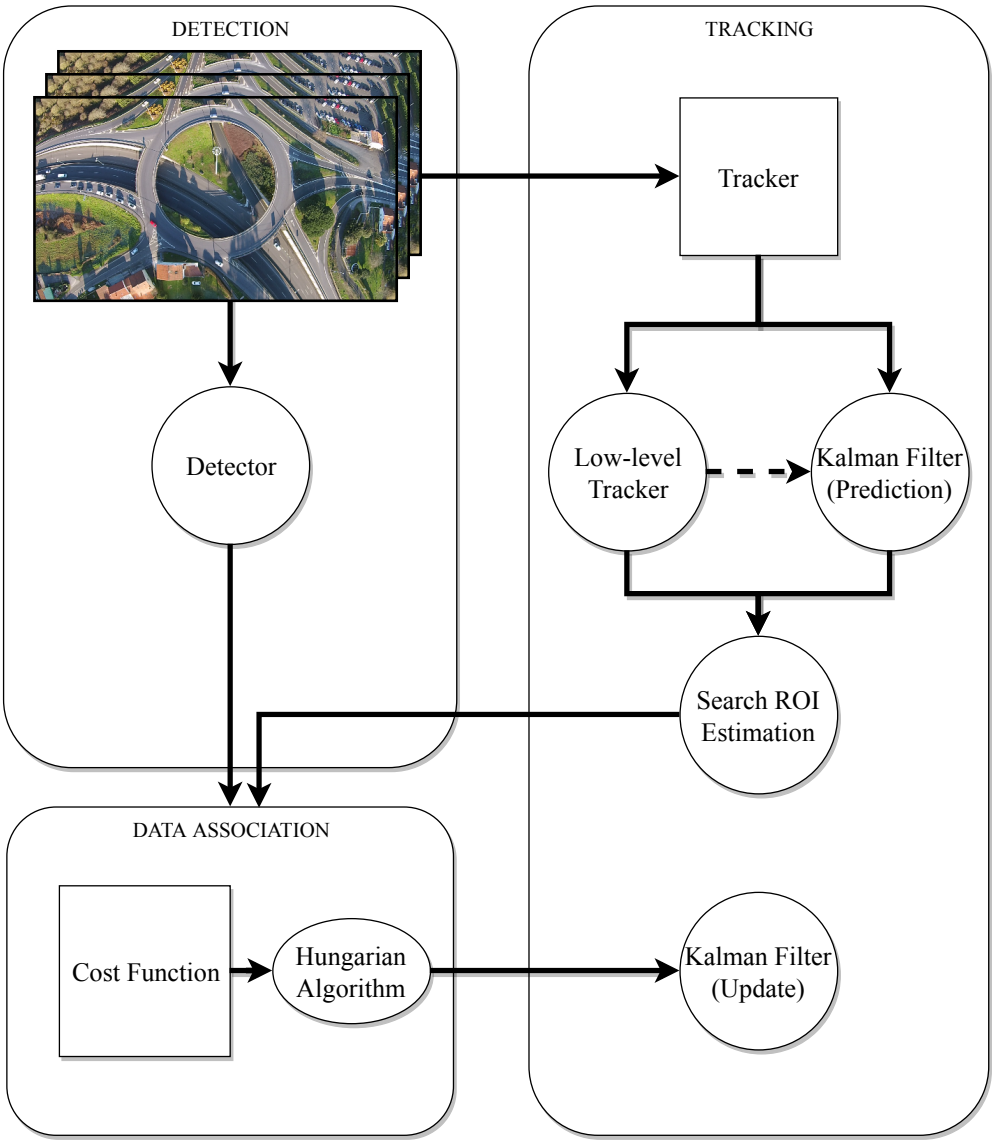


Figure 3.1: Architecture of our MODT system.

instant t are the new frame (Im_t) of the video, and the set of trackers in the previous time instant (Φ_{t-1}). First, the trackers positions in the new image (Im_t) are calculated (Algorithm

1, line 2 —Alg. 1:2—). This is done by combining two trackers: a DCF and a Kalman one (Section 3.1.1). Then, if the time elapsed between detections is over a certain threshold τ (Alg. 1:3), the detection of the objects of interest (Ψ_t) in the current image Im_t is performed through a ConvNet (Alg. 1:4). In practice, τ is set to the minimum time value that allows the system to achieve real-time performance. Detection is performed with a fully convolutional network called FPN [69], which uses feature maps information at different scales to locate from small to large objects, through a pyramidal architecture with lateral connections between them. The FPN provides high precision at a high computational cost, taking about 130 ms to perform a full detection in an HD image. Thus, clearly, a deep learning solution for detection does not suffice to comply with real-time requirements, which makes tracking necessary. After detection, trackers are updated (Alg. 1:5, Section 3.1.2). If no detection is performed at current time t, tracking prediction alone ($\overline{\Phi}_t$) determines the current trackers state (Φ_t, Alg. 1:7).

3.1.1 Tracking Prediction

Alg. 2 describes the trackers prediction. First, the system estimates the position of the trackers in the new frame (Im_t) using DCF tracking. Our tracker is based on the Discriminative Scale Space Tracker (DSST) [24], which is a correlation-filter-based tracker [13]. It uses HOG [21] and color as features for the correlation filter that models the tracked object.

In Alg. 2:3 the correlation scores Y_t are computed considering a sample S_t, extracted from Im_t and the previous filter information formed by W_{t-1} and X_{t-1}. λ is a weight parameter and d is the feature dimension of the correlation filter. The maximum value of Y_t —taking the inverse Discrete Fourier Transform (DFT)— will be the center of the object in the new image (ζ_t) as shown in Alg. 2:4. Then, the filter is updated at Alg. 2:5-6, where F is a target sample extracted from Im_t at position ζ_t, G is the desired correlation output and η is a learning rate parameter. In practice, we use two DCFs, one for estimating translation and another one for scale change. Tracking based just on DCF trackers has two limitations. First, we cannot handle occlusions (Figure 3.2). Second, it does not provide a robust tracking failure detection (i.e. knowing when the tracking fails) as the PSR (Peak to Sidelobe Ratio) value [13], which measures the spread degree of the convolution operation of the correlation filter, is not a reliable measure. As shown in Figure 3.3, the PSR takes different threshold values for different videos and scenarios, which makes difficult to identify when a tracker is lost.

Algorithm 2: Tracking Prediction

Require:

(a) Im_t: Image frame at current time t

(b) $\Phi_{t-1} = \{\varphi_{t-1}^1, \varphi_{t-1}^2, \ldots, \varphi_{t-1}^n\}$

Function `Tracking_Prediction` (Φ_{t-1}, Im_t) **:**

  for $i = 1$ *to* n **do**

    $Y_t = \frac{\sum_{l=1}^{d} \overline{W_{t-1}^l} S_t^l}{X_{t-1}+\lambda}$

    $\zeta_t = max(\mathscr{F}^{-1}\{Y_t\})$

    $W_t^l = (1-\eta)W_{t-1}^l + \eta \overline{G} F_t^l, \; l = 1,..,d$

    $X_t = (1-\eta)X_{t-1} + \eta \sum_{k=1}^{d} \overline{F_t^k} F_t^k$

    $\bar{\mu}_t = A_t \, \mu_{t-1}$

    $\bar{\Sigma}_t = A_t \, \Sigma_{t-1} \, A_t^T + R_t$

    $\kappa = \delta[1 - exp\{-(\zeta_t - c(\bar{\mu}))^T$

    $(C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} \, (\zeta_t - c(\bar{\mu})\}]$

    $Q_t = Q_t \, \kappa$

    $K_t = \bar{\Sigma}_t \, C_t^T \, (C_t \, \bar{\Sigma}_t \, C_t^T + Q_t)^{-1}$

    $\Sigma_t = (I - K_t \, C_t)\bar{\Sigma}_t$

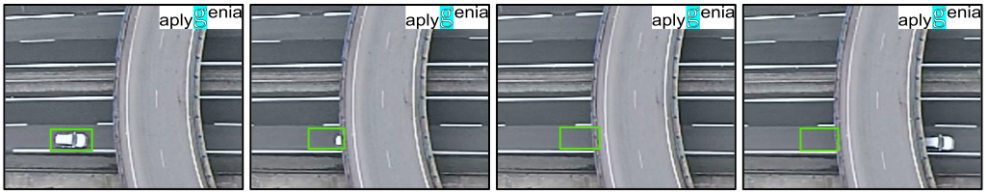
    $\overline{\varphi}_t^i \leftarrow < W_t, X_t, \bar{\mu}_t, \Sigma_t, Q_t >$

  return $\overline{\Phi}_t$

To provide a solution to both problems, we introduce a Kalman Filter (KF) tracker that, by modeling the movement of the object can handle occlusions and, in combination with the DCF tracker, can estimate the error in the tracking process. So, once the object's new position is calculated by the DCF tracker (ζ_t), we estimate the position using the Kalman filter. We use a linear constant velocity model in the KF, so the state of each object is modeled as:

$$\mu := [x, y, v_x, v_y] \tag{3.1}$$

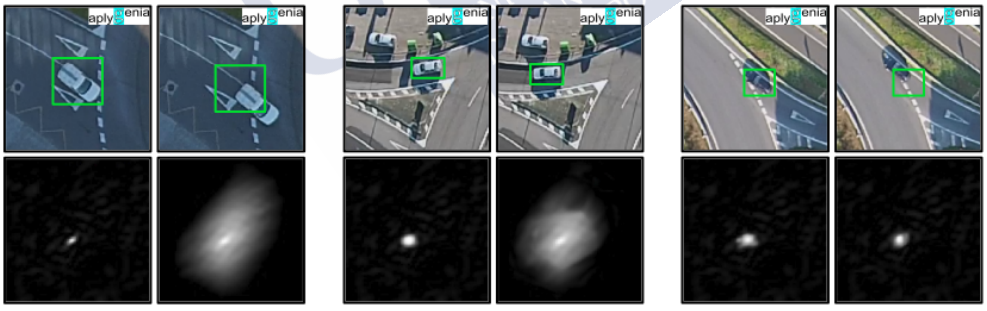
Here x and y represent the position of the object, and v_x and v_y represent the linear velocity in both axes. We perform Kalman prediction in Alg. 2 (7-8): $\bar{\mu}_t$ is the predicted state mean, A is the transition matrix, $\bar{\Sigma}$ is the predicted covariance of the system and R is the process noise covariance matrix. At this point we have two independent estimations of the new object position: DCF (ζ_t) and Kalman ($\bar{\mu}_t$).



(a) DCF Tracker

(b) DCF+KF Tracker

Figure 3.2: (a) The low-level DCF tracker (in green) cannot recover the identity of the object once occluded as it only relies on appearance. (b) The combination of a DCF and a KF manages occlusions, as it also takes into account the object motion model. Images courtesy of Aplygenia S.L.



(a) (b) (c)

Figure 3.3: PSR values are poor predictors of tracking failures for the DCF tracker.(a) An ideal case in which PSR values predict correctly a tracking failure. (b) A vehicle is being tracked correctly but the PSR distribution indicates a tracking loss. (c) Tracking loss is not detected by PSR as the values do not change in both situations. Images courtesy of Aplygenia S.L.

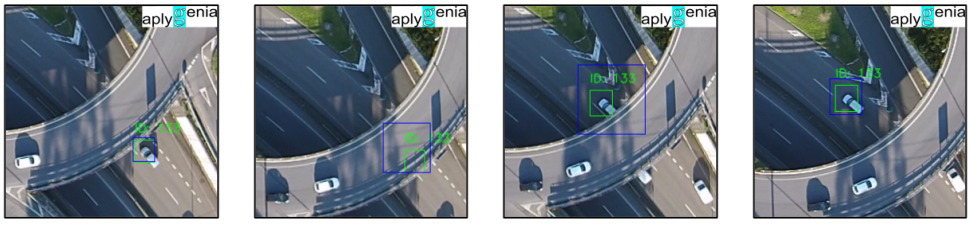
In order to measure the agreement degree of both estimators, we use the Mahalanobis distance, which measures the distance between a point and a distribution. We express this

distance as the uncertainty that represents how unlikely both positions correspond to the same tracker (Alg. 2:10), where $c(\bar{\mu})$ is the expected position of the object estimated by the KF, C_t is the Jacobian matrix of c (measurement model), Q_t is the measurement noise covariance matrix, and δ is a normalization factor. With this uncertainty (κ), we update accordingly the covariance of the measure Q_t (Alg. 2:11): the higher the uncertainty in both positions corresponding to the same object, the higher the covariance. When one of the estimators fails to track an object (i.e. occlusion), the distance between the positions of the trackers increases and so does the uncertainty. This is an important feature because it permits the detection of tracking failures, thus, maintaining a more reliable estimation of the object position during the time elapsed through a larger covariance matrix. Then, we perform a partial correction (Alg. 2:12-13) of the covariance Σ not only every time we integrate a measure performing a full correction (Alg. 3), but with every prediction as well.

At this point we integrate in every tracker $\overline{\varphi}_t^i$ the new calculated information, which is represented in a new set $\overline{\Phi}_t$ (Alg. 2:14-15).

3.1.2 Tracking Update

The tracking update process is shown in Alg. 3. First, the algorithm estimates the area in which the vehicle might be in the current frame. We call this area our search ROI (Region of Interest). This search ROI is a rectangle centered at $\bar{\mu}_t$ and with a size proportional to Σ_t (Alg. 3:3, Figure 3.4).



(a) (b) (c) (d)

Figure 3.4: Creation of a search ROI for occlusion handling. (a) Both tracking methods agree on the object position. (b) As the DCF fails to track the occluded object, the distance between both estimations increases and so it does the search ROI. This process continues in (c) and finally in (d), when the detector finds the vehicle at the other side of the road and the tracker recovers. Images courtesy of Aplygenia S.L.

Algorithm 3: Data Association and Tracking Update

Require:

(a) Im_t: Image frame at current time t

(b) $\overline{\Phi}_t = \{\overline{\varphi}_t^1, \overline{\varphi}_t^2, \ldots, \overline{\varphi}_t^n\}$

(c) $\Psi_t = \{\psi_t^1, \psi_t^2, \ldots, \psi_t^m\}$

Function `Tracking_Update` $(\overline{\Phi}_t, \Psi_t, Im_t)$ **:**
  for *i=1 to n* **do**
    $SROI_t^i$ =`Calc_SearchROI` $(\bar{\mu}_t, \Sigma_t)$
    for *j=1 to m* **do**
      if $\frac{SROI_t^i \cap \psi_t^j}{SROI_t^i \cup \psi_t^j} > 0$ **then**
        $z_t = \psi_t^j.center$
        $\omega(i,j) = [(z_t - c(\bar{\mu}_t))^T$
        $(C_t \Sigma_t C_t^T + Q_t)^{-1} (z_t - c(\bar{\mu}_t))]$
      else
        $\omega(i,j) = \infty$
  $\{< \overline{\varphi}_t^{\alpha}, \psi_t^{\beta} >\}$ =`Hungarian` (ω)
  for *every* α, β *in* $\{< \overline{\varphi}_t^{\alpha}, \psi_t^{\beta} >\}$ **do**
    `DCF_Reset` (ψ_t^{β})
    $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
    $\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$
    $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$
    $\varphi_t^i \leftarrow < W_t, X_t, \mu_t, \Sigma_t, Q_t >$
  return Φ_t

Next, data association assigns every detection to its corresponding tracker. The first step is to identify the possible detections that are candidates to be assigned to a particular tracker. In our case, a detection (ψ_t^j) that has an IoU (Intersection over Union) > 0 [116] with a tracker's search ROI ($SROI_t^i$) is considered a candidate for the assignment (Alg. 3§5). Once a detection has already been established as a candidate, we calculate the cost of association $\omega(i,j)$ as the Mahalanobis distance between the position predicted by the tracker ($c(\bar{\mu}_t)$) and the center of the measurement provided by the ConvNet (z_t) (Alg. 3§6-8). If there is no overlap (IoU) the association between the tracker $\bar{\varphi}_t^i$ and a detection ψ_t^j is not possible, and

the cost of association is set to infinite (Alg. 3§10). With this information, a cost matrix is generated. Every element of the matrix represents the cost of associating a detection (ψ_t^j) with a tracker (φ_t^i). That results in an assignation problem that is solved in polynomial time by the Hungarian Method (Alg. 3§11). For every successful assignation ($< \varphi_t^\alpha, \psi_t^\beta >$) we reset the DCF tracker of φ_t^α at position ψ_t^β, and perform an update of its KF, obtaining the new mean (μ_t) and covariance (Σ_t) of the filter (Alg. 3§13-16). Finally, every tracker φ_t^i in Φ_t (Alg. 3§17-18) is constructed with W_t, X_t and Q_t from the previous prediction step, and μ_t and Σ_t from the current update step.

3.2 Results

In this section, we evaluate our system in three different scenarios. Section 3.2.1 presents a comparison between a single DCF tracker and a combination between a DCF and a Kalman filter trackers (DCF + KF). In Section 3.2.2, we explore the computational cost of the different parts of our system and its implementation for real-time performance. In Section 3.2.3, we evaluate our system in a real-life scenario for roundabout monitoring. Finally, in Section 3.2.4 we introduce our solution for anomaly detection in traffic roads.

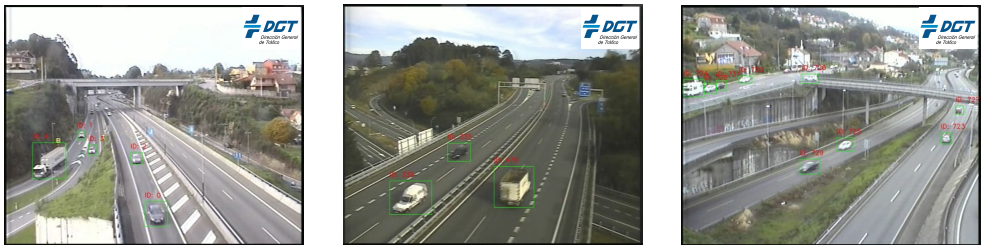
3.2.1 Comparison between DCF and DCF+KF

In this section, we compare the core of our tracking system — the DCF tracker plus the Kalman Filter — with a DCF baseline, in order to show the increase in robustness metrics by adding a motion predictor even in scenarios that do not present occlusions. In order to guarantee a fair comparison we will remove the following components from our complete system: (i) the ConvNet detector — we will use the ground truth detections; (ii) the tracking detection failure module; (iii) the data association module — association is based on IoU [7].

The dataset has three real-life videos provided by the *DGT* [1] with more than 1,000 vehicles with a frame rate of 15 FPS and without total occlusions, for a fair comparison (Figure 3.5 and Table 3.1). We measure the performance with standard multiple object tracking metrics [7].

In every frame, we have a set of hypotheses (our trackers) and a set of objects (labeled ground truth). We assign to every hypothesis its nearest object with IoU > 0. For every successfully assigned pair, we measure the IoU between the two bounding boxes to estimate

[1] Dirección General de Tráfico (DGT) is the Spanish Traffic Authority [131].



(a) (b) (c)

Figure 3.5: Example frames of our video dataset to evaluate tracking metrics. These videos are from traffic monitoring cameras. Images courtesy of DGT.

Video	Frames	Vehicles	Time
DGT_34001	4,900	221	0:05:26
DGT_39132	15,050	575	0:16:43
DGT_39151	2,345	244	0:02:36
Total	***22,295***	***1,040***	***0:24:46***

Table 3.1: Dataset with more than 1,000 different vehicles in more than 20 minutes of video to evaluate tracking metrics.

the multiple object precision (MOTP). To estimate the multiple object accuracy (MOTA) or robustness we count every hypothesis without an associated object as a *false positive*, every object that has no associated hypothesis as a *miss* (or false negative) and, finally, every assignation of identity (ID) that differs from the last frame's ID as a *mismatch* (or ID switch).

Also, in order to simulate a real-life scenario in which we cannot perform detection in every frame to achieve real-time performance, we varied the frequency of the detections (obtained from the ground truth) from once every 5 frames to once every 30 frames. To identify tracking failure, we use the PSR value. As the PSR is not a reliable metric, we repeated the experiments for different PSR thresholds. Finally, when detections are available, all trackers are reinitialized.

Results for MOTP are shown in Figure 3.6. As we can see, DCF+KF achieves more precision than DCF independently of the PSR threshold, but the differences are negligible. This is an expected result as the objective of adding a KF is to increase robustness.

MOTA metrics are shown in Figure 3.7. DCF+KF shows more robustness than DCF for every combination of PSR value and number of frames between detector calls. The higher the

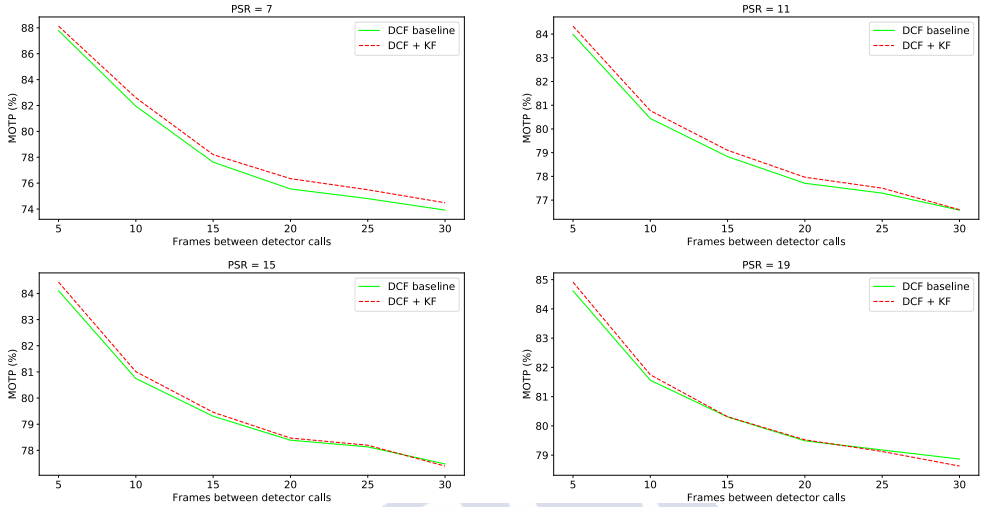


Figure 3.6: MOTP results for DCF and DCF+KF with several PSR thresholds and different number of frames between detector calls for the videos outlined in Table 3.1.

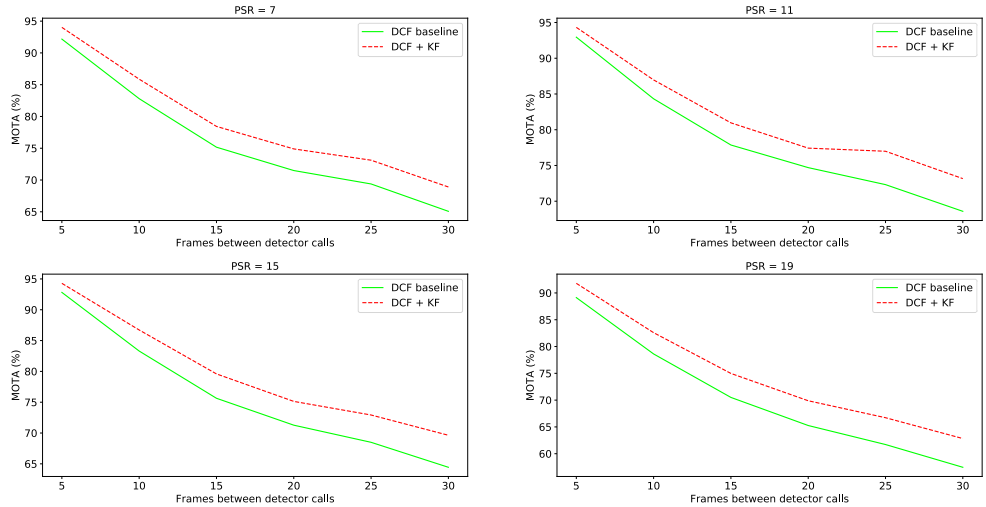


Figure 3.7: MOTA results for DCF and DCF+KF with several PSR thresholds and different number of frames between detector calls.

number of frames between detector calls, the higher the improvement of DCF+KF over DCF

Version	Reinit.	M	FP	MM	M(%)	FP(%)	MM(%)
DCF	5	168	1,091	84	17.2%	30.9%	402.9%
DCF+KF	5	143	834	17			
DCF	10	639	2,102	164	34.9%	24.0%	381.3%
DCF+KF	10	474	1,695	34			
DCF	15	1,285	2,540	205	40.7%	12.5%	387.2%
DCF+KF	15	913	2,258	42			
DCF	20	1,949	2,806	209	45.8%	10.0%	353.4%
DCF+KF	20	1,336	2,551	46			
DCF	25	2,704	2,788	201	42.9%	13.0%	277.5%
DCF+KF	25	1,892	2,468	53			
DCF	30	3,309	2,951	197	43.7%	3.7%	205.6%
DCF+KF	30	2,302	2,846	65			

Table 3.2: Reinitializations (Reinit), Misses (M), false positives (FP) and mismatches (MM) in the videos of Table 3.1 for a PSR threshold of 13 (standard). The right columns show the improvement of DCF+KF over DCF.

— 5% of improvement for 25 and 30 frames. This is important for a real-time application because the deep learning detector can be executed only at some time instants.

This increase in robustness is detailed in Table 3.2, analyzing the misses, false positives, and mismatches. The largest reduction of failures is in the number of *mismatches*. This is because the KF allows keeping the identities of the objects for longer periods, which results in an improvement between 200% and 400% compared with DCF. This improvement is of key importance for roundabout monitoring as each mismatch represents a failure and has a negative impact on the success rate in the final system. Also, for detecting anomalies on traffic roads, we need to maintain the identity of the vehicles as long as possible to ensure that the correct alarm is triggered. In conclusion, the results obtained support the addition of a Kalman filter to a DCF tracker as the core of our tracking system due to the increase in robustness while keeping precision.

3.2.2 Implementation Details

The proposed system (Figure 3.1) runs on a server with an Intel Xeon E52623v4 2.60 GHz CPU, 128 GB RAM and an Nvidia GP102GL 24GB [Tesla P40] as GPU. Figure 3.8 shows how the system works for a 30 FPS HD video. The system performs tracking in one of every 3 frames and detection in one of every 6 frames. Also, these two tasks are completely parallelized by threads. With these frequencies, the robustness of our system is not affected.

Table 3.3 shows the times of the two most computationally expensive operations of our system: detection and tracking — computing times of other tasks are negligible. As explained before, for a 30 FPS HD video, the tracking module processes 1 of 3 frames, which gives 0,1 seconds per frame. Using 15 threads for parallelization, the system is able to process more than 400 objects in the image while maintaining real-time performance, *i.e.* 30 FPS — the maximum number of objects at any given time in our videos was 60. As mentioned before, detection is the slowest part of our system, taking an average 0.135 seconds in an HD image and 0.075 seconds in VGA resolution. These values are below the 0.2 threshold required by the system for the detection module.

3.2.3 Roundabout Monitoring

In this section, we analyze our complete system (Figure 3.1) for roundabout monitoring. The objective of the system is to identify the entry and the exit a vehicle takes, maintaining its

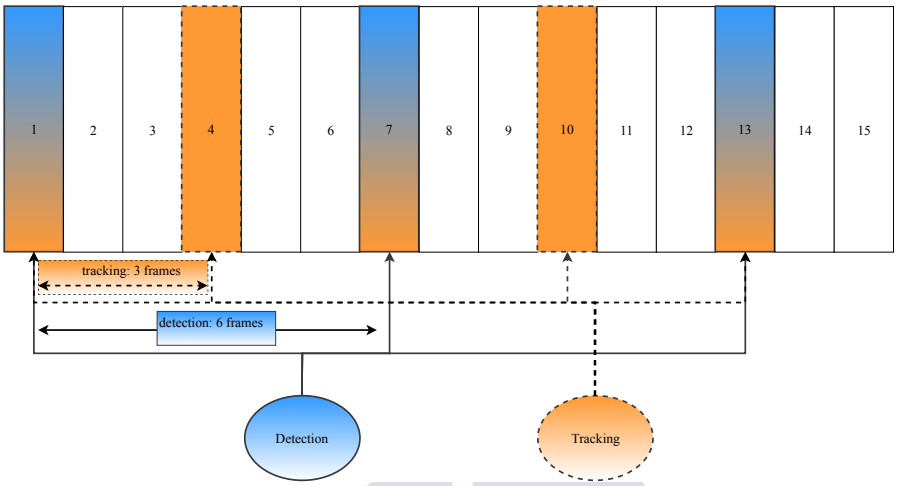


Figure 3.8: Frame processing of our system with an input video. We show the first 15 frames as an example. The system performs tracking one of every 3 frames (orange), and detection one of every 6 frames (blue).

Tracking	
Frames processed per second	10 frames of 30
Total max. time with parallel computing	0.0121 sec (60 objects, 15 threads)
Max. number of objects in 0.1 sec	492 objects
Detection	
Frames processed by second	5 frames of 30
Average time per HD image	0.135 sec

Table 3.3: Computational times for the tracking and detection modules of the RTMODT system.

identity while it remains in the roundabout. The final goal is to provide the I/O matrix R, in which every element $(R(i, j))$ represents the number of vehicles that joined the roundabout taking entry i and exit j. If a vehicle enters the roundabout and exits it with the same ID we

count that as a tracking success. On the contrary, if the identity changes along the video, then we count that vehicle as a tracking failure.

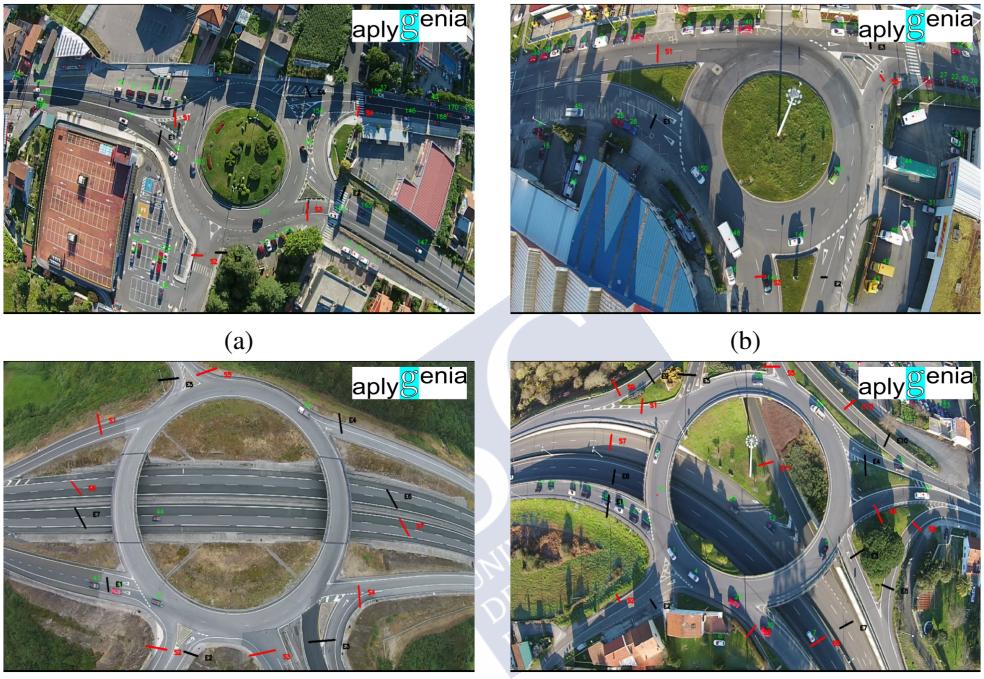


Figure 3.9: Example frames of some videos of the roundabout monitoring dataset. These videos are recorded from a UAV flying over a roundabout. Images courtesy of Aplygenia S.L.

For performing the metrics, we use a video dataset that consists of eight videos of roundabouts recorded from an Unmanned Aerial Vehicle (UAV) at 30 FPS with HD resolution. The videos have different conditions that are challenging for traffic monitoring: shadows, total occlusions (two-level roads), camera movement, etc. Figure 3.9 shows a snapshot of some of these videos [2].

Table 3.4 shows the results obtained from processing the I/O matrix of eight videos with more than 1,000 vehicles in total. An average success rate of 93% is obtained. Results also show our system's ability to handle occlusions as two of the videos are scenarios with a high

[2] A demonstration video can be downloaded from: http://bit.ly/roundabout_sample_video

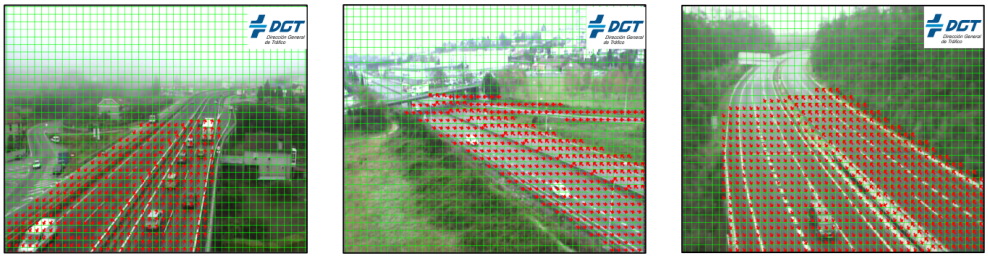
rate of total occlusions: in one of them, the 50% of the vehicles are totally occluded nearly twice on average.

Video	#occ	#vocc	Time (min:sec)	#vehicles	Success
usc_vr_1	308	160	05:11	320	86,50%
usc_yt_1	–	–	01:43	13	100%
usc_yt_2	–	–	00:30	15	93%
usc_yt_3	–	–	00:45	14	100%
usc_pl_1	–	–	11:12	138	88%
usc_rb_1	–	–	11:48	230	95%
usc_sx_1	–	–	09:26	255	91%
usc_ou_1	22	11	02:49	52	96%
Total	***330***	***171***	***43:40***	***1,037***	***93,36%***

Table 3.4: Results in the video dataset for roundabout monitoring. The columns are: video, number of occlusions (#*occ*), number of vehicles occluded (#*vocc*), duration of video, total number of vehicles (#*vehicles*) and success rate obtained by our tracking system.

3.2.4 Anomaly Detection in Traffic Monitoring

Another traffic monitoring application that we developed using our systems as a base is anomaly detection. The objective is to identify possible anomalies on a road, like sudden stops (sometimes produced by crashes), vehicles that circulate at an inappropriate velocity, or vehicles that move in a direction different from the usual on that road.



(a) (b) (c)

Figure 3.10: Examples of the movement map generated in traffic videos for accident detection. For each cell we show just the mean angle. Images courtesy of DGT.

(a) (b) (c)

Figure 3.11: Example of an alarm for a vehicle moving in a wrong direction. In (a) the system detects an object moving with a direction contrary to the usual one in that lane, so an alarm is launched. In (b) the system keeps tracking the object despite being totally occluded. In (c) the system continues the tracking after the occlusion, keeping the same ID and the alarm. Images courtesy of DGT.

(a) (b) (c)

Figure 3.12: Example of an alarm for an accident. In (a) a motorcycle crashes with a car. In (b) the system detects both vehicles as static in a lane that has movement, so it triggers an alarm. Finally in (c) the system keeps triggering alarms for every car that stops in that area. Images courtesy of DGT.

First, the system has to generate a movement map of the road. We do that by dividing the image into grid cells. Each one of the cells has two moving averages: one for the angle and the other for the velocity. We update the movement map with the tracking information that our RTMODT system provides. Figure 3.10 shows the movement map calculated for three traffic videos. The alarms are fired based on a probabilistic Gaussian model of the velocity and direction in the cells of the map occupied by the vehicle. Thus, when the velocity or direction of the vehicle differs from the standard ones, an alarm is triggered. The model also takes into account that the velocity and/or direction in the area can change, and an alarm might be

(a) (b) (c)

Figure 3.13: Frames of a video that has both alarms. In (a) a truck starts to derail. In (b) the system detects an anomaly in the second lane when the truck invades it so it triggers the wrong direction alarm. In (c) when the crashed truck remains static the system triggers the velocity alarm and it does the same for every truck that moves with abnormal velocity (reduced speed). Images courtesy of DGT.

switched off due to this. Figures 3.11, 3.12 and 3.13 show detected alarms in three different traffic scenarios.

3.3 Final Remarks

We have presented an RTMODT system that combines ConvNet detection, DCF and Kalman trackers, and a Hungarian data association. The system is able to track hundreds of objects in real-time while being robust to occlusions. The combination of the DCF and Kalman filters allows to: (i) improve both the precision and, especially, the robustness, obtaining an improvement of up to 402,9% in the number of tracking mismatches; (ii) estimate the error of each tracker, thus increasing the robustness and reliability of the system. We have applied our RTMODT system to two different real-life traffic monitoring applications. First, in roundabout monitoring, our system achieves a 93% success rate for the I/O matrix, even in cases with high occlusion rates, shadows, and movement of the UAV onboard camera. Second, for anomaly detection, the system identifies accidents on roads, sudden stops, abnormal speeds, and vehicles that move in the wrong direction, triggering an alarm in any of these scenarios.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 4

DEEP LEARNING-BASED MULTIPLE OBJECT VISUAL DETECTION AND TRACKING ON EMBEDDED SYSTEM FOR IOT AND MOBILE EDGE COMPUTING APPLICATIONS

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Nowadays an increasing number of applications have certain hardware restrictions for their execution. This is shown in the increasing demand of solutions for embedded architectures, however, compute and memory demands of state-of-the-art deep learning methods are still a shortcoming that must be addressed to make them useful at IoT end-nodes. In particular, recent results depict a hopeful prospect for image processing using Convolutional Neural Networks, CNNs, but the gap between software and hardware implementations is already considerable for IoT and mobile edge computing applications due to their high power consumption. In this chapter, we present our first approach for embedded architectures. This is a portable solution for detecting and tracking multiple objects. Since the system processes videos obtained through the integrated camera of the embedded device, real-time processing speed will be met as long as the rate of the solution is equal to or higher than the video recording rate ,i.e. 10 FPS. This proposal represents a first step in the development of embedded architectures and will form the basis of the RTMODT system proposed in the following chapter. The solution is implemented on an NVIDIA Jetson TX2 development kit. It includes a camera and wireless connection capability and it is battery powered for mobile and outdoor

applications. A collection of representative sequences captured with the onboard camera, dETRUSC video dataset, is used to exemplify the performance of the proposed algorithm and to facilitate benchmarking. The results in terms of power consumption and frame rate demonstrate the feasibility of deep learning algorithms on embedded platforms although more effort in the joint algorithm and hardware design of CNNs is needed. The contents of this chapter are extracted from the following publications [1]:

> B. Blanco-Filgueira, D. García-Lesta, M. Fernández-Sanjurjo, V. M. Brea and P. López. Deep Learning- Based Multiple Object Visual Tracking on Embedded System for IoT and Mobile Edge Computing Applications. IEEE Internet of Things Journal, Vol. 6, No. 3, pp. 5423-5431, 2019.

> B. Blanco-Filgueira, D. García-Lesta, M. Fernández-Sanjurjo, V. M. Brea and P. López. Live Demonstration: Deep Learning-Based Visual Tracking of Multiple Objects on a Low-Power Embedded System. *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1, Sapporo (Japan), 2019.

4.1 Multiple Object Tracking Approach

Figure 4.1, shows the experimental setup with a Jetson TX2 powered by a 3S LiPo battery and remotely controlled by a tablet using WiFi connection. The images shown on the tablet correspond to live performance, where the black and white image on the left represents the detection of the foreground and the camera capture with green and red boxes on the right depicts the tracking.

4.1.1 Overview

A proposal for multiple object tracking was developed making use of the Hardware-Oriented PBAS (HO-PBAS) algorithm [37] to detect moving objects integrated with the GOTURN CNN based tracker [48]. The HO-PBAS detector [37] is a hardware oriented foreground segmentation method based on the pixel-based adaptive segmenter (PBAS) algorithm [51]. The HO-PBAS is oriented to focal-plane processing, [99], with the benefit of less memory

[1] We would like to thank B. Blanco-Filgueira, D. García-Lesta and P. López for their work and collaboration.

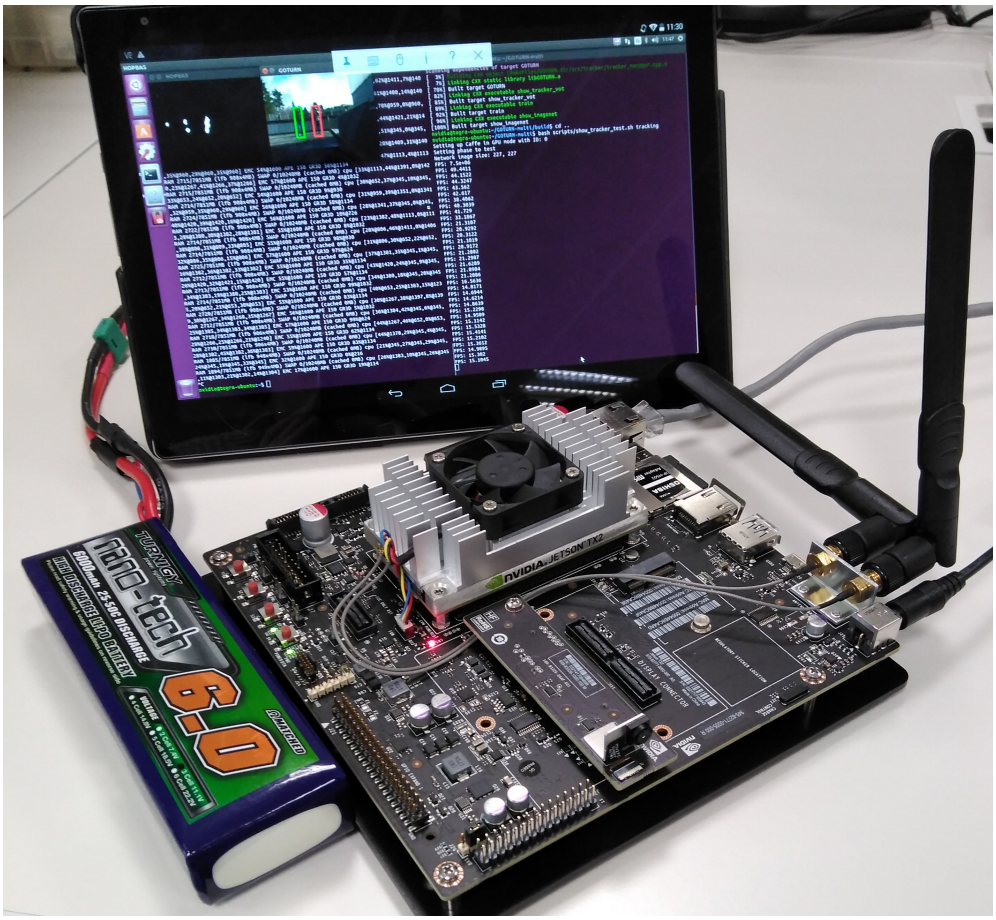


Figure 4.1: Embedded solution: Jetson TX2 development kit is battery powered and remotely controlled using a tablet and WiFi connection.

usage, becoming appropriate for future on-chip implementation of the whole algorithm and also for its use on embedded solutions such as the Jetson platform. Both CPU- and GPU-based implementations of the HO-PBAS were developed. The latter was written with the CUDA version of the OpenCV library functions, using some custom-made low-level CUDA functions as well. However, due to the simplicity of the algorithm, the speedup obtained from the GPU parallel execution does not compensate for the required time for the data transfer from the global memory to the GPU memory. Thus, the CPU version of the HO-PBAS was

chosen, leaving the GPU computational resources available for the tracking algorithm. On the other side, GOTURN [48], that was originally designed for 1-object tracking, has been applied in this work to multiple object tracking. It was tested on the VOT2014 dataset [57] and it is one of the few that can achieve real-time performance with the additional advantage of reasonable hardware requirements. The GOTURN algorithm was implemented using Caffe [52], a deep learning framework which allows a high level of abstraction. Caffe libraries are optimized to use CUDA so low level development is not needed to maximize performance.

Both algorithms have been tested on publicly available datasets and present good results separately. In this work, they were integrated to address an end-to-end solution over the NVIDIA Jetson TX2 embedded platform. It consists in multiple object tracking of real-time detected objects. For this purpose, both algorithms must run jointly, obtaining precise object detections from HO-PBAS which are used as inputs for the GOTURN tracker. Regardless of the accuracy of the used detector, its output can not be as precise as manual annotation of the object to track, which is the approach commonly used for tracker validation such as in VOT competition. Thus, merely integration of both algorithms is not enough and additional high level development was needed to overcome limitations of a non ideal input annotation for the tracker, as will be explained in the next subsections.

A diagram representation of the proposed algorithm is depicted in Figure 4.2. The model and the camera are initialized and then detection and tracking of multiple objects is carried out as it is explained in the following subsections.

4.1.2 Detection

The output of the HO-PBAS detector are the bounding boxes of the moving objects in the scene which have an area between desired maximum and minimum values. As new objects are detected in the scene, their bounding boxes are sent to the tracker. Typically, trackers are tested on annotated videos where the first frame bounding box is used as input, that is, the object to track. Thus, the object is completely at the scene in the first frame and the manual annotation is ideal. This means that the object can be well characterized by the CNN and the tracker algorithms can be compared in the same terms for challenge purpose, regardless of the detection. However, in a real situation objects come into the scene from the edges and they are detected as soon as they occupy a minimum area, what can occur before they enter the image completely. If a partially cropped object is sent to the tracker, it probably does not manage enough information to treat it as the same object when it appears completely. Therefore,

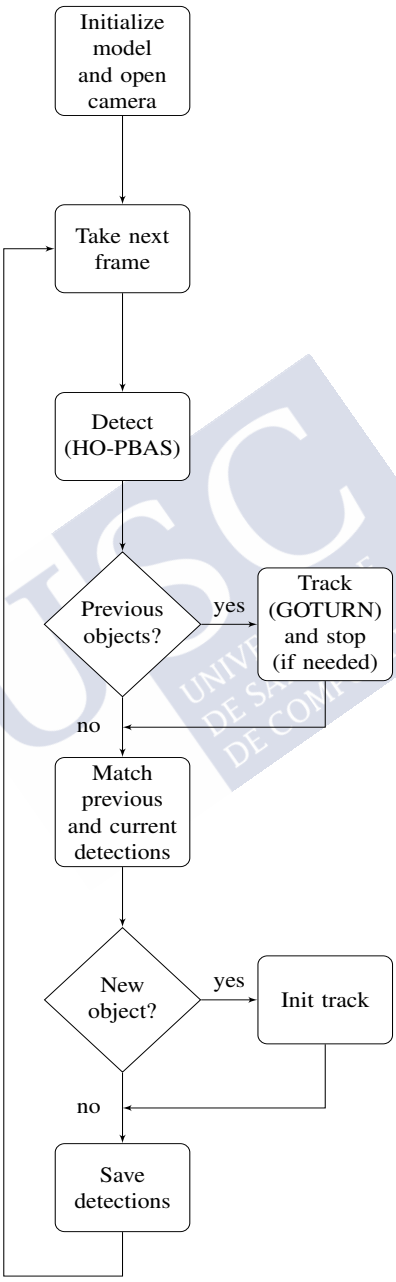


Figure 4.2: Conceptual diagram of the multiple object tracking algorithm proposed.

partial object detection must be prevented by only considering completely detected objects. In our implementation, this is accomplished by requiring the bounding boxes to be separated from the image borders by a certain number of pixels.

4.1.3 Tracking of Previous Objects and New Candidates

1. After the detection, if moving objects were already detected in previous frames ("Previous objects?", Figure 4.2), the GOTURN algorithm continues tracking them independently of HO-PBAS performance. To do that, the CNN itself estimates the new position of every tracked object. Additionally, in order to stop the tracking of those that leave the scene, the algorithm checks if any of the bounding boxes are close to the edges of the image ("Track (GOTURN) and stop (if needed)", Figure 4.2).

2. Then, it must be checked whether there is a new object in the scene. The minimum matching between all pairs of previous and current bounding boxes, calculated as the intersection over union [116], is selected as a potential new detection ("Match previous and current detections", Figure 4.2). In the particular case that more than one new object appeared on the scene, they would be identified and their tracking initialized in subsequent frames.

3. After that, it must be determined whether the new detection is really a new object ("New object?", Figure 4.2). This is solved by comparing the new detection with all the bounding boxes previously estimated by the tracker. This prevents false positives derived, for instance, from the following circumstances:

- The occasional detector failure over the same moving object which would result in a new object every time the detector fails and restores its performance later.

- Objects that stop for a while, long enough to be treated as background, and are thus identified as new objects by the detector when they move again.

- Objects that move away from the foreground can become too small to be detected by HO-PBAS but they can be tracked by GOTURN and be recognized as previous objects if they come back to the foreground, see Figure 4.3(a).

Next, if the new detection is identified as a new object after the checking, a new tracker is initialized ("Init track", Figure 4.2). Finally, current detections are saved as previous detections for the next frame ("Save detections", Figure 4.2).

Although there are more robust state-of-the-art techniques to solve the problem of data association between trackers and detections, such as those well ranked at the MOTChallenge benchmark [136], sometimes their low speed is a drawback for real-time applications [50, 55]. Owing to the requirements to develop an end-to-end solution over a low-power embedded platform for real-time performance, the more straightforward solution described here is used instead.

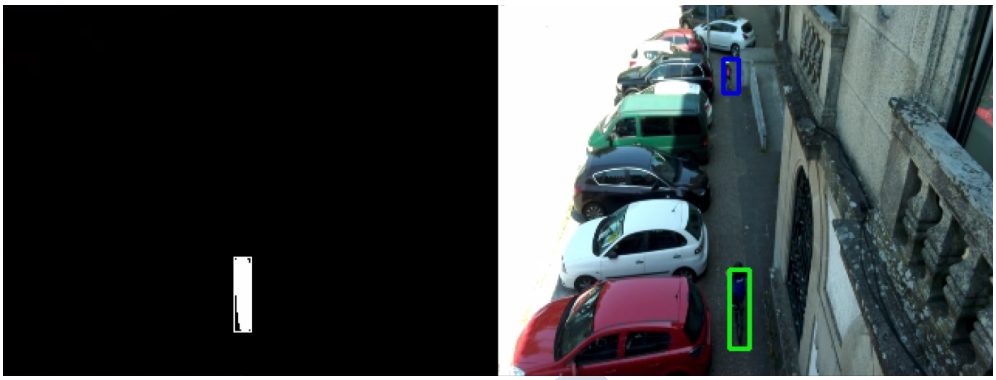
4.2 Results

The proposed algorithm was written in C++, QVGA resolution frames (320×240 pixels) are processed and the maximum RAM usage is 1.6 GB. Following recent trends and suggestions of other authors, which claim a jointly software and hardware design, the results of the present work focus on the frame rate and power consumption of the algorithm, which are hardware dependent [100, 101].

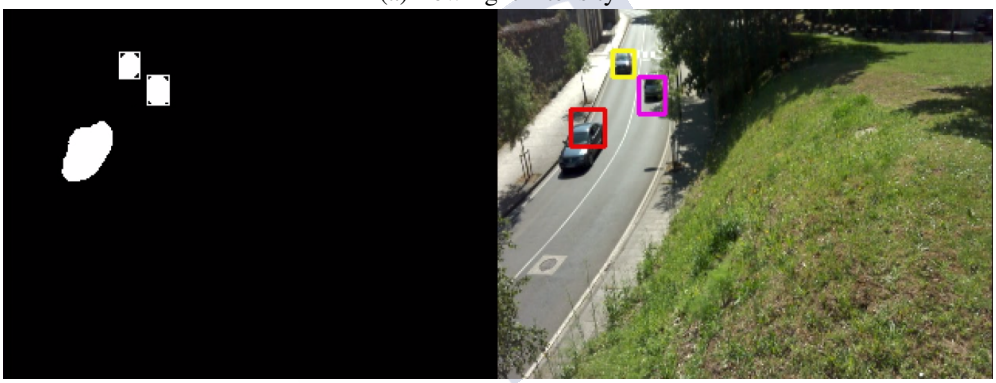
4.2.1 dETRUSC Database

The accuracy of a model should be measured on widespread used datasets. The HO-PBAS detector was evaluated on the 2014 updated version of changedetection.net (CDnet) public dataset [42] whereas the GOTURN tracker was tested on VOT2014 dataset [57]. However, in this work the detector and tracking algorithms were integrated and thus no benchmark for joint performance is available. Additionally, videos are captured with the camera of the Jetson TX2 development kit in order to demonstrate live processing capability for real-time decision making. That is, the complete end-node performance aims to demonstrate the feasibility of deep learning techniques for visual tasks on embedded solutions for IoT end-nodes.

Due to the aforementioned reasons, a collection of representative sequences captured with the onboard camera were taken and used to exemplify the performance of the proposed algorithm. The dETRUSC video dataset and the results are publicly available and can be accessed at [130]. The videos represent real scenarios, including diverse challenging situations such as low light and low contrast conditions, glass reflections, high velocity and shadows. Figure 4.3 shows a pair of captures of the multiple object tracking performance. The black and



(a) Low light intensity



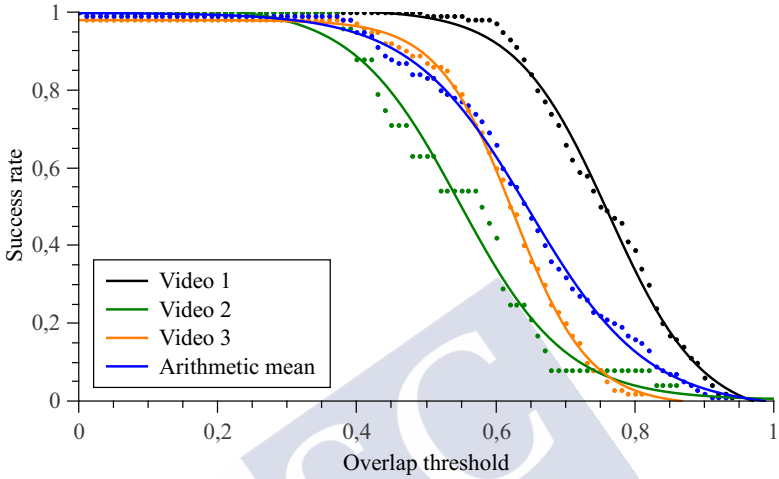
(b) High velocity

Figure 4.3: Example of multiple object tracking under different challenging conditions.

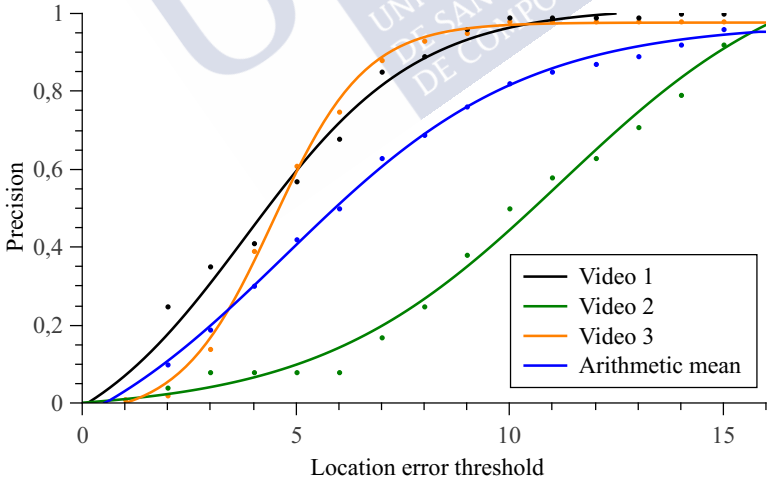
white image on the left is the HO-PBAS segmentation whereas tracking is depicted on the right. Figure 4.3(a) exemplifies the correct tracker performance under low light intensity and high contrast conditions. In Figure 4.3(b) the capability of the tracker to follow vehicles is illustrated, although GOTURN does not handle large movements in order to not increase the complexity of the network and thus achieve high frame rate.

4.2.2 Detection and tracking performance

The performance of both HO-PBAS detector and GOTURN tracker separately have been previously given in [37, 48], respectively. Regarding robustness and precision, it must be con-



(a) Success plots



(b) Precision plots

Figure 4.4: Success and precision plots for some videos of the dETRUSC dataset and arithmetic mean.

sidered that they are usually calculated for single object tracking. Additionally, the ground truth detection is used to initialize the algorithm and to reinitialize it when precision decreases under a certain threshold. This differs from our joint detection and tracking solution, which initializes the GOTURN tracker with the detection given by the HO-PBAS and is never reinitialized.

Considering the aforementioned aspects, precision and success plots of the proposed complete system using some videos of the dETRUSC dataset, Section 4.2.1, are depicted in Figure 4.4 for completeness, but no comparison with other trackers in challenges like VOT is possible in the same terms as we do not use the ground truth bounding box to initialize and reinitialize the tracker. Figure 4.4(a) shows the success plots for three videos. The success plot represents the percentage of successful frames whose overlap between the tracker and the ground truth bounding boxes is larger than the given threshold [116]. The arithmetic mean of the success rate value at 0.5 overlap threshold is 90% in our complete system. Figure 4.4(b) represents the precision plots for the same videos. The precision plot is based on the center location error metric, which is defined as the average Euclidean distance between the center locations of the tracked target and the ground truth [116]. Thus, precision plot shows the percentage of frames whose estimated location is within the given threshold distance of the ground truth. A representative precision score is for 20 pixels threshold, which is 100% in our complete system.

4.2.3 Frame rate

VOT2017 challenge introduced a real-time experiment [60] but, as explained in the report, tracking speed depends not only on the programming effort and skill but also on the used hardware. If no power constraint nor small size are needed, the latency reduces as a more powerful hardware is chosen. Thus, comparison of different algorithms in terms of throughput (GOPS), latency (s) or frame rate (FPS) takes on meaning only when they all run on the same hardware.

To the best of our knowledge, there are not previous solutions for multiple object detection and tracking running on Jetson TX2, thus this proposal can not be compared with others in terms of velocity. Just in a recent work [44], the latency and throughput of different CNNs only for object recognition or detection are characterized, achieving a maximum frame rate of 4.3 FPS in full clocks mode for object detection, what it is not enough for real-time inference at the edge.

Table 4.1: NVIDIA Jetson TX2 Operation Modes

Mode	Denver (GHz)	A57 (GHz)	GPU (GHz)
Max-N	2.0	2.0	1.30
Max-Q	-	1.2	0.85
Max-P Core-All	1.4	1.4	1.12
Max-P ARM	-	2.0	1.12
Max-P Denver	2.0	-	1.12

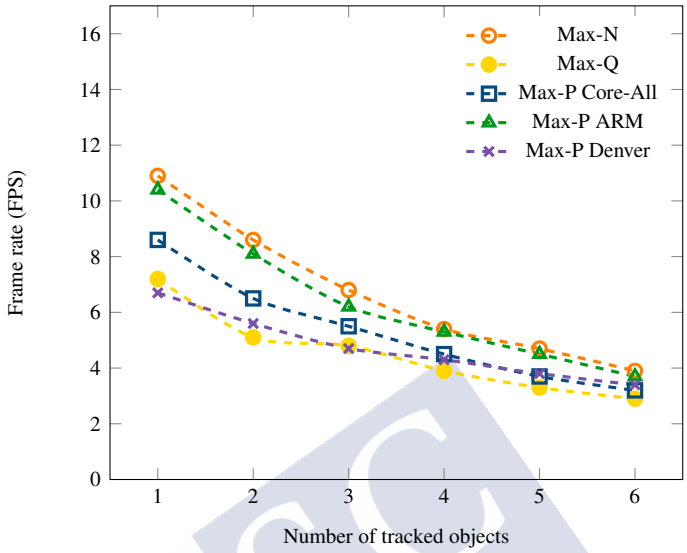
In order to measure the maximum algorithm speed over the Jetson, regardless of the capture frame rate, prerecorded video processing is performed instead of live capture. The metrics for the frame rate of the proposed model as a function of the number of tracked objects were measured for all the available operation modes and full clocks mode, Figure 4.5. As the number of objects increases, the algorithm processes the frames more slowly, although a saturation tendency is observed. The most efficient operation mode, Max-Q, also offers the worst performance in terms of velocity. Regarding the full clocks mode, it improves the one object tracking performance but no significant profit is observed for multiple object purpose.

In view of these results, real-time performance was evaluated at Max-N operation mode and 10 FPS video capture, using at any time only the available frame from the live camera without storing intermediate frames for delayed processing. Satisfactory results were found in outdoor and indoor scenarios, under low light intensity and high contrast conditions and fast moving objects such as vehicles. Video results can be accessed at the dETRUSC database site [130].

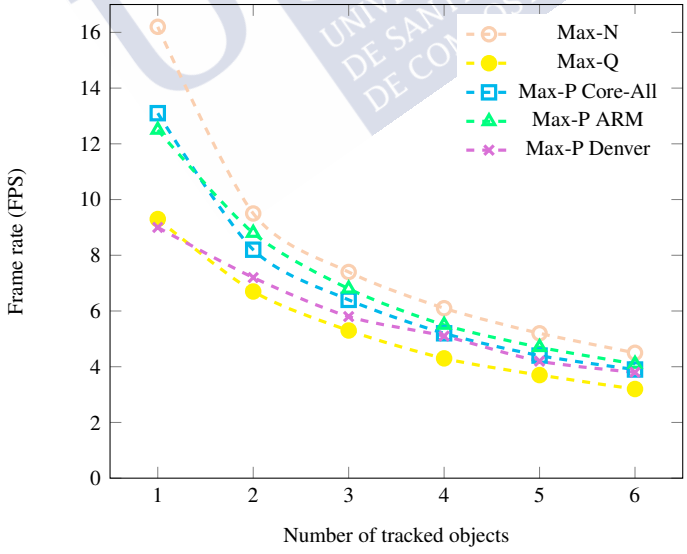
4.2.4 Power Consumption

Consumption is one of the main limiting factors for deep learning application on embedded IoT end-nodes. The opportunities that CNNs offer for image processing are undeniable but their practical use and expansion will highly depend on the hardware design and the development of hardware-oriented algorithms [100]. The energy consumption of CNNs is dominated by data movement instead of the computation itself [101]. Fortunately, the most costly operations in terms of data movement are highly parallel.

The total power consumption of the proposed multiple object tracking algorithm running



(a) Normal operation



(b) Full clocks mode

Figure 4.5: Frame rate of the multiple object tracking algorithm for different number of tracked objects and Jetson TX2 operation modes.

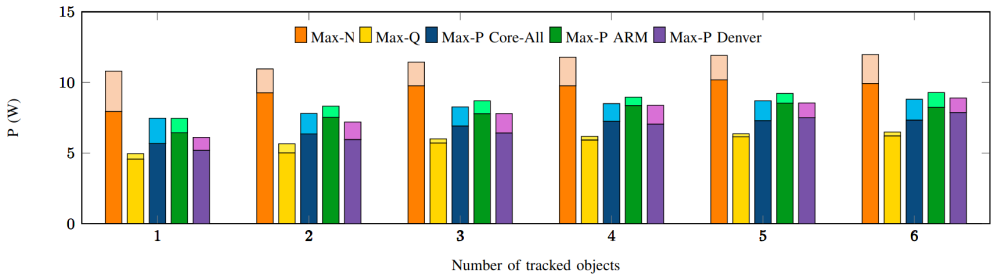
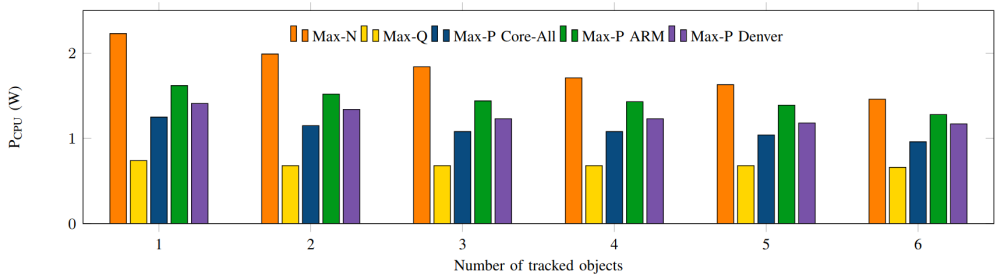


Figure 4.6: Total power consumption of the system for different number of tracked objects and Jetson TX2 operation modes. The bar increment represents the additional energy consumption on full clocks mode.

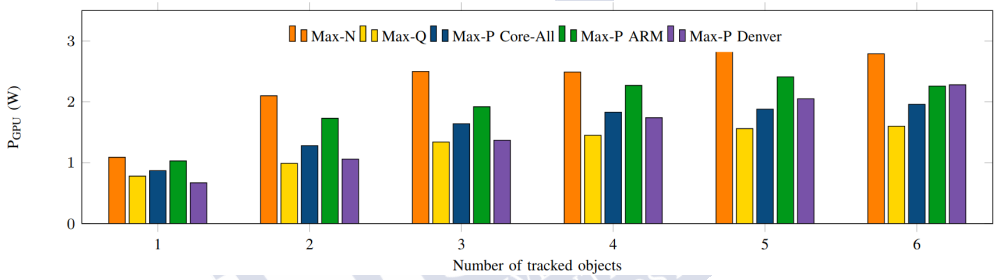
on NVIDIA Jetson TX2 development kit is shown in Figure 4.6. It was measured for the different operation modes of the board summarized in Table 4.1. The increments of the bars represent the additional consumption due to full clocks mode activation, which forces to run all cores at the maximum speed all time. As can be observed, the latter has little effect for Max-Q mode since it limits the clocks to ensure operation in the most efficient range. The power was also measured as a function of the number of tracked objects, finding that power consumption softly increases with the presence of more moving objects in the scene. An analogous representation of the power consumed by the CPUs (P_{CPU}) and GPU (P_{GPU}) is depicted in Figure 4.7. An increment of the GPU utilization is observed through a greater power consumption as the number of tracked objects increases. On the contrary, CPUs consumption diminishes as GPU workload raises because they must await GPU completion.

As expected, the Max-Q operation mode is the most cost-effective in terms of energy, regardless of metric, ranging from 4.57 to 6.21 W. On the contrary, Max-N is the most expensive with 12 W maximum consumption since all CPUs and GPU run at maximum clock speeds, see Table 4.1. Max-P modes are a balance between both, running all CPUs, only ARM A57 processors, or only Denver processors. That is to say, using a 3S 6000 mAh LiPo battery as in our experimental set-up, an autonomy of six hours under continuous performance at maximum power consumption is guaranteed.

If we compare our proposal with the state-of-the-art detectors and trackers we can observe that improving the accuracy has a great impact on velocity and power consumption. For instance, Faster R-CNN [90] and R-FCN [20] detectors can run at 5-17 FPS and 6 FPS, respectively, on a K40 GPU, which has a power consumption of 235 W. That is, a similar frame rate



(a) CPUs



(b) GPU

Figure 4.7: Power consumption of the CPUs and GPU for different number of tracked objects and Jetson TX2 operation modes.

as ours is achieved but performing only the detection and with 20 times more power consumption. In turn, MDNet [81] tracker achieves a speed of 1 FPS running on a K20 GPU, 225 W power consumption, whereas high frame rate of 58-86 FPS is obtained with SiameseFC [8] tracker but using a GeForce GTX Titan X, which consumes 250 W. Moreover, those data correspond to only 1-object tracking and do not consider the detection phase. Additionally, we are not considering CPUs consumption in these detectors and trackers results, nor comparing dimensions and price of the hardware, nor memory usage, which are also penalized. Summarizing, what we propose is an algorithm that solves the application problem in real-time, with low memory usage, and implemented on a small dimensions hardware platform, which also has an affordable price and with a low-power consumption of 12 W.

4.3 Final Remarks

An end-to-end solution for real-time deep learning-based multiple object tracking in an embedded and low-power IoT oriented platform is presented. NVIDIA Jetson TX2 [137] development kit was chosen because it allows development from beginning to end from the early stage, including camera and wireless connection. It was powered by a LiPo battery and remotely controlled using a tablet. Another strengths of the platform are price and dimensions.

The Hardware-Oriented PBAS (HO-PBAS) algorithm [37] to detect moving objects was integrated with the GOTURN CNN based tracker [48] in order to perform multiple object tracking. Whereas both algorithms present good results separately, additional development was needed to overcome some limitations due to the jointly operation.

Qualitative results over the dETRUSC video dataset captured with the onboard camera can be found at [130]. Good results were found under real challenging scenarios including low light and high contrast conditions, glass reflections, high velocity and shadows. Moreover, the good performance of the proposed algorithm was also exemplified in terms of robustness and precision over videos of the same dataset. Regarding velocity and power consumption, real-time performance at 10 FPS video capture, using at any time only the available frame from the live camera without intermediate storage for delayed processing, with a total power consumption of only 12 W is achieved.

The memory and computational resources are sufficient to run the complete solution as in a standard platform, which results in a lower power consumption at the expense of a reduced frame rate. However, if more complexity was needed or more vision algorithms were added, the embedded system could be limited. Thus, a joint algorithm and hardware design is essential in the future of IoT, specially for those applications such as visual tasks which require live performance and thus cloud computing is not an alternative due to latency or poor coverage.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Chapter 5

Real-Time Multiple Object Detection and Tracking System for Embedded GPUs

From our first development for embedded platforms we have continued this line of research to develop a complete RTMODT system for this type of architecture. The development of RTMODT systems on low-power edge devices as IoT nodes, without compromising accuracy, is a challenge due to the limited computing capacity of said devices. This might rule out the best in-class computer vision solutions, which nowadays are based on deep learning, and thus, they are very hardware demanding. This chapter meets this challenge with an RTMODT system that employs cutting-edge deep learning architectures on an embedded GPU while operating in real-time. For this purpose, a system has been designed that extends a joint architecture of tracking and detection by adding a module comprised of appearance-based and movement-based trackers that allow to maintain the identity of the objects of interest for longer periods of time while alleviating the burden of the detector. Our system is mapped onto an embedded GPU platform, cutting down power consumption significantly with respect to a server GPU. The contents of this chapter are extracted from the following publication:

> M. Fernández-Sanjurjo, M. Mucientes, and V. M. Brea. Real-Time Multiple Object Visual Tracking for Embedded GPU Systems. IEEE Internet of Things Journal. Accepted.

5.1 RTMODT System Overview

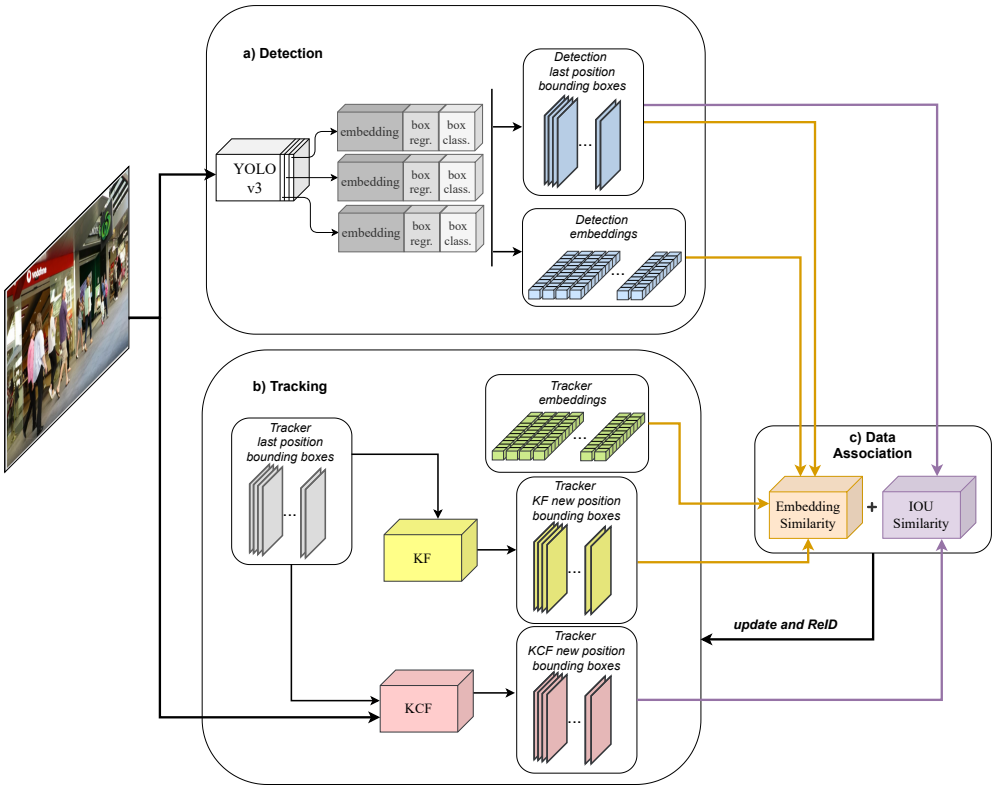


Figure 5.1: Different components of the RTMODT system. Object detection is carried out by a JDE-based detector —module (a)—, which provides a bounding box for each detected object as well as a feature vector that models its appearance. Tracking —module (b)— is performed by a combination of two different trackers: an appearance-based tracker —Kernelized Correlation Filter (KCF)— and a motion-based Kalman filter (KF). Data association —module (c)— is solved in two steps: (i) associations based on visual features (embedding) which also use motion information; (ii) association by position. Real-time processing speed is achieved through an optimal distribution of the components among the available hardware: detection on GPU, and tracking and data association on CPU.

The objective of our RTMODT system is to identify the objects of interest that appear in a video and track them while they remain on the scene. In case any of these objects becomes occluded the system would try to recover its identity when it reappears. The whole system

must reach real-time processing speed. The pipeline of our system is shown in Figure 5.1. It consists of three main components: detection, tracking, and data association. In order to guarantee a real-time processing speed, our system will use tracking in all the frames of the video, while detection and data association will be performed only at certain time intervals. This reduces the dependency of our system on the detector, which is the most computationally expensive component. We denote the time between successive detections by τ and, in practice, it is set to the minimum value that allows this real-time speed to be maintained.

When the system receives the first frame of a video, detection is performed and all the detected objects are added as new trackers. The detection (Section 5.1.2) is done using a Joint Detection and Embedding (JDE) based approach [113] —module (a) in Figure 5.1. This solution combines a deep learning detector model for target localization and an appearance embedding model for data association within the same architecture. Thus, it provides for each detected object a bounding box and an embedding vector that models its appearance.

From this point on, in each new video frame the first task is to estimate the position of the trackers in the new video image through the tracking module —module (b) in Figure 5.1. This module determines the position of the objects of interest in the successive frames until the next call to the detector, allowing to maintain their identity. The tracking module is composed of a combination of an appearance-based correlation filter tracker —Kernelized Correlation Filter (KCF) [49]— and a motion-based Kalman filter tracker (KF) [53] (Section 5.1.1). The combination of these two trackers provides increased robustness by making accurate tracking when the object is visible (KCF), and allowing re-identification of occluded objects (KF). Both solutions have the advantage that they are computationally very fast and are fully parallelizable on CPU.

The final process —module (c) in Figure 5.1— is data association between detections and trackers (Section 5.1.3). Its main objective is to integrate the information from the other two components to preserve the correct identity of the objects over time. Data association is conducted following two strategies: based on the appearance or the position of the objects. The association based on the appearance of the objects is done through the embeddings provided by the detector, while the one based on position is carried out using the object bounding box. The association is used for: refining the position and features of currently visible objects, adding new objects that appear in the scene, and removing those that leave the scene or become momentarily occluded. For those frames for which the detector is not run, the system will update the position of the objects using only the information coming from the tracking

module.

Algorithm 4: Multiple Object Tracking System

Require:

(a) Im_t: Image frame at current time t

(b) Φ_{t-1}: Set of current visible trackers in last frame $t-1$

(c) Φ_{t-1}^{lost}: Set of lost trackers

Function `Main` (Im_t, Φ_{t-1}) **:**
  Φ_t = `Track_KCF` (Φ_{t-1}, Im_t)
  Φ_t^{lost} = `Track_KF` (Φ_{t-1}^{lost}, Im_t)
  if $time_elapsed > \tau$ **then**
    Ψ_t = `JDE_Detect` (Im_t)
    $\Phi_t^{\alpha^1}, \Phi_t^{\beta^1}, \Psi_t^{\gamma^1}$ = `Hung` (`Emb` (Φ_t, Ψ_t))
    $\Phi_t^{\alpha^2}, \Phi_t^{\beta^2}, \Psi_t^{\gamma^2}$ = `Hung` (`IoU` ($\Phi_t^{\beta^1}, \Psi_t^{\gamma^1}$))
    $\Phi_t^{\alpha^3}, \Phi_t^{\beta^3}, \Psi_t^{\gamma^3}$ = `Hung` (`IoU` ($\Phi_t^{lost}, \Psi_t^{\gamma^2}$))
    $\Phi_t = \{\Phi_t^{\alpha^1}, \Phi_t^{\alpha^2}, \Phi_t^{\alpha^3}, \Psi_t^{\gamma 3}\}$
    $\Phi_t^{lost} = \{\Phi_t^{\beta^2}, \Phi_t^{\beta^3}\}$
  return Φ_t

Algorithm 4 shows the main steps of the proposed system. In the following sections, we describe in detail each of these steps.

5.1.1 Tracking

The goal of the tracking module is to calculate the position of the previously detected objects in the new video frame. Object tracking is carried out by a combination of two trackers: KCF —appearance-based tracker— and KF —motion-based tracker. On the one hand, the KCF algorithm presents a variant of the classic correlation filter. It identifies the direction in which the object is moving from the maximum correlation between the future patch and any of the translated patches proposed. The model of the object being tracked is updated online using a linear ridge regression model. The main steps of the algorithm KCF can be summarized as:

- With the initial position of the object the model for the tracker is trained in the first frame.

- For a new frame the last bounding box position defines a test image patch.
- The maximum score location over the test image patch and the model represents the new center of the target, and the bounding box is updated with that position.
- The new model is trained at the new position.

This type of tracker has good accuracy and high speed due to exploiting the properties of circular arrays and kernel functions.

On the other hand, Kalman filter-based tracking produces an estimate of the position of the object based on its previous state and the current measurement. The state of the tracked objects is represented by:

$$s = [x, y, a, h, vx, vy, va, vh] \tag{5.1}$$

where (x, y) represents the bounding box center coordinates, a represents the aspect ratio, h represents the object height and (vx, vy) represents the speeds of the object center shift, and (va, vh) the speeds of the aspect ratio and height change. The KF uses a constant velocity motion model. The state of the object is updated with the measurement (from the matched detection) represented by:

$$z = [x, y, a, h] \tag{5.2}$$

In the first video frame, the trackers —formed by a KCF and a KF— are initialized from the first detections. From that point on, at each time t the inputs to the system are: the current frame of the video (Im_t) and the sets of visible (Φ_{t-1}) and lost trackers (Φ_{t-1}^{lost}) —trackers that could not be matched with previous detections— in the previous frame $t-1$. Each one of the trackers in these sets contains a bounding box that represents its last position, and an embedding vector representing its appearance. The embedding vector is provided by the JDE detector and contains 512-d features to model the appearance of the object. At current time t, the new position of the visible trackers is calculated (Φ_t) in the current frame with the KCF tracker —Algorithm 4, line 2 (Alg. 4:2). For the lost trackers the position in the current frame (Φ_t^{lost}) is calculated using the KF tracker —Alg. 4:3. The purpose of the motion-based tracker is to support the appearance-based tracker when the later fails due to occlusions, distractors or detector failures, thus allowing the eventual re-identification of objects through position. This combination of trackers has the advantages of speed and robustness, making it possible to estimate an accurate position of objects by visual features when they are available and to predict their movement when they are occluded.

5.1.2 Detection

Object detection is done through a JDE approach —module (a) in Figure 5.1. This solution tackles object detection and embedding modelling within the same architecture, making both parts share the same low-level features and avoiding the re-computation so common in most tracking-by-detection algorithms, which results in a limitation for real-time performance.

The underlying architecture of the detector is a modification of YOLOv3 [88] single shot detector. It has the advantage of being faster than its two-stage detection counterparts and by incorporating an FPN-based architecture [69] for the feature extraction within the network at different scales, which significantly improves the detection of small objects.

To calculate jointly detections and embeddings, once the video image has made a forward pass through the network, a feature map is obtained for each of the three down-sampling scales and these are combined in the way established by FPN. The JDE detection network works as follows: first, the corresponding input frame goes through the network backbone, generating a feature map. In a second step, that feature map enters the header of the network. In a conventional CNN for object detection, the header has two heads: one for the classification of each proposal and another one for refining the proposal bounding box through regression. However, as seen in Figure 5.1, the JDE detection adds a third header to generate the embeddings of each detected object.

The frame rate of the system is controlled through parameter τ, which indicates the time elapsed between detections. The challenge is to select the shortest time τ that allows maintaining high frame rate values without affecting tracking metrics with several objects in the scene. In operation, when the time elapsed since the last detection exceeds τ, a JDE-based detection will be performed in the current frame, which will yield a set of object detections (Ψ_t), each with a bounding box and an embedding —Alg. 4&4-5. Once the trackers (Φ_t) and detections (Ψ_t) in the current frame have been calculated, it is necessary to perform data association in order to integrate the information of both sources.

5.1.3 Data Association

Data association is done in three steps. First, the assignment between detections and trackers is made using embedding similarity because it provides greater robustness than location-based association. This type of association is based on the appearance of the objects through the use of the deep features provided by the detection network. To carry out the process, a cost matrix

is created using the Euclidean distance between trackers and detections embeddings vectors. Given the embedding vectors (em^a, em^b) of size n corresponding to tracker a and detection b respectively, the corresponding matrix element (a,b) —representing row a and column b of the cost matrix C_v— is calculated as:

$$C_v(a,b) = \sqrt{\sum_{i=1}^{n} \left(em_i^b - em_i^a\right)^2} \tag{5.3}$$

The value of the matrix C_v contains each possible combination between trackers and detections, that is, the matrix will have as many rows as trackers and as many columns as detections.

To filter out those associations that are not probable, such as two very separate objects, we calculate the Mahalanobis distance. The Mahalanobis distance between a tracker a with mean μ_a and covariance Σ_a —given by its Kalman filter— and a detection b that represents an observation z_b is defined as:

$$C_d(a,b) = \sqrt{(z_b - j(\mu_a))^T (J\Sigma_a J^T + Q_a)^{-1} (z_b - j(\mu_a))} \tag{5.4}$$

where $j(\mu_a)$ is the prediction of the measurement for the track with predicted state μ_a, J is the Jacobian matrix of j and Q_a is the covariance of the measurement noise.

This is calculated for each possible combination between trackers and detections. For those gating distances that are greater than a threshold η, the cost of association in the matrix $(C_d(a,b))$ is set to infinity. In practice, this threshold would be the 0.95 quantile of the chi-square distribution with 4 degrees of freedom (9.4877). Finally, the association value of the tracker a and detection b will be a combination of both cost matrices —C_v and C_d—:

$$C_{emb}(a,b) = \lambda * C_v(a,b) + (1-\lambda) * C_d(a,b) \tag{5.5}$$

The parameter λ determines the trade-off between both costs. In practice, λ is set to 0.98.

Later on, the associations are determined by the Hungarian method, which is a well-known algorithm for solving the assignment problem in polynomial time, yielding three sets as a result: pairs of detections and trackers successfully associated, represented by a set of current updated trackers $\Phi_t^{\alpha^1}$, unmatched trackers $\Phi_t^{\beta^1}$ and unmatched detections $\Psi_t^{\gamma^1}$ —Alg. 1 6. For each successful assignment $\Phi_t^{\alpha^1}$, the tracker is updated with its corresponding detection in two ways. On the one hand, updating the new bounding box of the tracker with the one of the detection, resetting the KCF, and updating the KF in this new position. On the other hand,

the feature vector of the last detection (em^b) is used to update the one of the tracker (em^a) by making a weighted sum of them, calculated as:

$$em^a = \delta \times em^a + (1-\delta) \times em^b \tag{5.6}$$

where the hyper-parameter δ determines the update rate. This way of updating the features allows to progressively refine the embedding vector of the tracker with the new detections.

Unmatched pairings are due to the fact that the minimum similarity threshold between a tracker and a detection is not reached. Those trackers and detections not matched by embedding ($\Phi_t^{\beta 1}, \Psi_t^{\gamma 1}$) will go to the second step of the data association, which is guided by IoU (Intersection over Union) [11]. To make the association by IoU, giving the bounding boxes of tracker a ($bbox^a$) and a detection b ($bbox^b$) the cost is defined as:

$$C_{iou}(a,b) = 1 - \left(\frac{bbox^a \bigcap bbox^b}{bbox^a \bigcup bbox^b}\right) \tag{5.7}$$

Matrix C_{iou} is again solved by the Hungarian method, which generates the properly updated trackers with their corresponding detections ($\Phi_t^{\alpha^2}$), unmatched trackers ($\Phi_t^{\beta^2}$) and unmatched detections ($\Psi_t^{\gamma^2}$) — Alg. 4.7. In the third step the process is repeated once again with the remaining unmatched detections ($\Psi_t^{\gamma^2}$) and the vector of previously lost trackers (Φ_t^{lost}). This vector contains those trackers that could not be matched in previous time instants, not necessarily only in the previous frame, either due to detector failures or because the object was occluded. This last check allows the re-identification of those objects that reappear on the scene after one of these two situations. The result of this last matching produces the successfully matched trackers ($\Phi_t^{\alpha^3}$), unmatched trackers ($\Phi_t^{\beta^3}$), and unmatched detections ($\Psi_t^{\gamma^3}$) —Alg. 4.8.

Finally, the new set of trackers will be formed by every successfully updated tracker ($\Phi_t^{\alpha^1}, \Phi_t^{\alpha^2}, \Phi_t^{\alpha^3}$) and by those detections that have not been matched up to this point ($\Psi_t^{\gamma^3}$), which will be added as new objects (Alg. 4.9). Those trackers that could not be matched throughout the whole process ($\Phi_t^{\beta^2}, \Phi_t^{\beta^3}$) will form the set of lost trackers (Φ_t^{lost}) —Alg. 4.10. An track remains for a predetermined time t_{delete} in vector Φ_t^{lost} before being completely removed.

By performing the data association in three steps, priority is given to matching based on features learned from the neural network. This implies that in situations where several objects appear together, fewer association errors will be made by taking into account their appearance first, instead of their position.

5.2 Experimental Study

In this section, we will present an analysis of the results obtained by our algorithm in different conditions. In Section 5.2.1, we perform a comparison of our system with other three RTMODT solutions. In Section 5.2.2 we perform a time analysis discussion about the improvements of our proposal for an embedded architecture.

5.2.1 Experimental Results and Comparison

As quality metrics of our system, we will use the two most common in the state-of-the-art: Multiple object tracking precision (MOTP) and Multiple object tracking accuracy (MOTA) [7]. These are calculated between hypothesis (trackers) and objects (ground truth). MOTP reflects the average precision, defined by the average Intersection over Union (IoU) between the hypotheses and the correctly matched objects. MOTA reflects the accuracy of the tracking algorithm by combining three metrics: False Positives (FP), False Negatives (FN), and ID Switches (ID Sw). False positives are those cases in which a hypothesis has no associated object. False negatives occur when an object does not have an associated hypothesis. ID Switches occur when the hypothesis associated with the same object changes. MOTA is the most important of the two metrics and is used to rank multiple object tracking solutions.

In order to guarantee an accurate measurement of the quality and the execution time of the system, it is necessary to use a widespread dataset with a considerable number of objects. In this case, we have selected the MOT16 dataset [80] for several reasons. Firstly, it is the dataset used for JDE evaluation [113], which leads to a fairer comparison with our system. Secondly, it is a standard dataset for measuring MOT metrics and presents a large number of objects in different scenarios. For pedestrian detection, we use the model JDE-576×320 provided by JDE authors. We also use the MOT-16 test set for performing the measurements. Figure 5.2 shows the visual result of our system in some videos of this test set. As usual in a machine learning approach, we have separate training and test sets, so our selected model has been trained with several pedestrian datasets: Caltech, Citypersons, CUHK-SYSU, PRW, ETHZ, and MOT17.

To make a fair comparison with [113], it is necessary that the systems to be measured reach a real-time processing speed, i.e. 25 FPS, on the embedded device. The method proposed in [113] works in real-time for high-end hardware but once integrated on the NVIDIA Jetson TX2 its speed decreases to 3.4 FPS. One of the reasons for this deep fall in processing speed

Table 5.1: Tracking results of our system in MOT16 test set.

Sequence	MOTA	MOTP	FP	FN	ID Sw,
MOT16-01	29.2	73.2	562	3,893	72
MOT16-03	67.3	74.7	9,307	23,673	1,184
MOT16-06	25.5	71.5	2,979	5,186	432
MOT16-07	33.4	73.2	2,100	8,459	317
MOT16-08	32.0	74.6	1,244	9,936	194
MOT16-12	43.9	73.7	863	3,663	124
MOT16-14	18.6	69.6	2,525	11,836	684
OVERALL	51.1	74.1	19,580	66,646	3,007

is that the JDE model calls for deep learning detection in every frame. Thus, making straight numbers, video-rate processing at 25 FPS with this approach leads to a drop in the number of times the detector is called from once per frame to once every 8 frames on average. That would be the realistic scenario in which a satisfactory processing rate would be reached on the embedded system, we will call this approach JDE-RT.

We have also compared our approach with other two real-time solutions. Both of these approaches are based on an implementation of Simple Online Real-Time Tracking (SORT) [10]. This tracker includes a Kalman filter to estimate the position of the objects and the Hungarian method to make the association by Intersection over Union (IOU). For the detection module, these approaches employ Tiny-YOLOv3 [88] and Tiny-YOLOv4, [12] which are the more recent lightweight versions of this single shot detector. These YOLO versions represent the state-of-the-art in real-time object detection on embedded systems. To train both detectors we have used the same dataset as with the other two systems —our approach and JDE-RT. We have called these systems Tiny-YOLOv3-SORT and Tiny-YOLOv4-SORT. Detections have been made in all frames since both approaches work in real-time on Jetson TX2, specifically, Tiny-YOLOv3-SORT at 33 FPS and Tiny-YOLOv4-SORT at 30 FPS.

The results of our system (as described in Section 5.1), which uses detections every 11 frames to achieve the target processing rate, are shown in Table 5.1. Results of JDE-RT —with detections every 8 frames— are shown in Table 5.2. Tables 5.3 and 5.4 show the results for the YOLO-SORT based approaches, and finally Table 5.5 presents the overall comparison.

In view of the results, we can notice that our system achieves a 5.2 point rise in overall MOTA compared with JDE-RT — or a 11.3% increase. This boost in the accuracy of the system is due to the management of trackers between detector calls. Our solution uses a low-

Table 5.2: Tracking results of JDE-RT in MOT16 test set.

Sequence	MOTA	MOTP	FP	FN	ID Sw
MOT16-01	25.5	72.9	399	4,327	38
MOT16-03	67.9	74.6	6,709	26,395	466
MOT16-06	8.4	70.4	1,887	8,400	283
MOT16-07	17.9	71.5	1,813	11,363	230
MOT16-08	29.0	73.4	632	11,116	131
MOT16-12	24.9	71.1	727	5,390	115
MOT16-14	1.6	66.9	1,639	15,897	660
OVERALL	45.9	73.9	13,806	82,888	1,923

Table 5.3: Tracking results of Tiny-YOLOv4-SORT in MOT16 test set.

Sequence	MOTA	MOTP	FP	FN	ID Sw
MOT16-01	27.5	79.0	114	4,473	47
MOT16-03	52.9	74.9	3,573	44,893	729
MOT16-06	50.1	78.6	332	5,241	186
MOT16-07	24.6	76.9	334	11,857	119
MOT16-08	32.3	79.2	529	10,645	149
MOT16-12	43.8	78.5	185	4,422	55
MOT16-14	11.6	72.6	225	15,862	260
OVERALL	42.8	75.8	5,292	97,393	1,545

Table 5.4: Tracking results of Tiny-YOLOv3-SORT in MOT16 test set.

Sequence	MOTA	MOTP	FP	FN	ID Sw
MOT16-01	33.9	74.6	139	4,015	71
MOT16-03	51.5	76.5	3,803	46,050	882
MOT16-06	46.5	78.7	300	5,652	219
MOT16-07	26.1	76.4	565	11,333	161
MOT16-08	32.4	80.1	352	10,818	152
MOT16-12	42.2	79.8	125	4,616	57
MOT16-14	13.6	71.4	653	14,943	368
OVERALL	42.3	76.8	5,937	97,427	1,910

Table 5.5: Comparative results of all solutions.

System	MOTA	MOTP	FP	FN	ID Sw
Ours	**51.1**	74.1	19,580	**66,646**	3,007
JDE-RT	45.9	73.9	13,806	82,888	1,923
Tiny-YOLOv4-SORT	42.8	75.8	**5,292**	97,393	**1,545**
Tiny-YOLOv3-SORT	42.3	**76.8**	5,937	97,427	1,910

Figure 5.2: Sample frames of our system running on some of the MOT 16 test set videos on the NVIDIA Jetson TX2 embedded device.

level tracker that adds more robustness than the simple Kalman filter used in the JDE-RT. This implies that the tracked objects will remain overlapped with the ground truth for longer periods of time. Thus, we get an overall better tracking performance by reducing the number of false negatives cases (FN) in 16,242.

Both solutions present limitations due to the impossibility of adding a new object or removing one that no longer remains in the scene until the next detector call. Our approach calls the detector less frequently than JDE-RT, which explains the increase in false positives (FP). This lower number of detector calls should have an equally negative impact on the number of false negatives but this is offset by the overall greater robustness of our system. With respect

to the number of ID switches (ID Sw), they also increase simply because by being able to track more objects, more associations are performed, and therefore, more errors are made. Regarding multiple object precision (MOTP), our system also improves JDE-RT although the gain is not very significant and both results are high.

Comparing our system with Tiny-YOLOv4-SORT and Tiny-YOLOv3-SORT, we see an improvement in MOTA of 8.3 and 8.8 points, respectively. This is a big difference, especially if we take into account that both approaches use detections in all frames, while our system does it only every 11 frames.

As we can see, the main advantage of our solution is the significant reduction of the number of false negatives (FN), by being able to keep the position of the objects for a longer time thanks to the tracking module. YOLO-SORT approaches have fewer false positives (FP) because they perform detections on all frames, thus resetting the bounding boxes of the objects while our system can continue tracking incorrect objects (FP) until the next detection arrives. They also present fewer ID switches (ID Sw) because it is easier to associate objects from consecutive frames than with a separation of 11 frames between them.

Both Tiny-YOLOv4-SORT and Tiny-YOLOv3-SORT outperform our system in MOTP by a small margin as both of them always rely on the detector to establish the position of the objects while our system uses tracking most of the time to do so.

As MOTA is the main performance metric to rank tracking methods in the MOT challenge, the increase in MOTA achieved by our solution outperforms the other methods in the comparison (Table 5.5). This makes it an excellent option as an RTMODT system to be used on an embedded low-power device to solve real-life video surveillance applications. The results obtained in the MOT16 test dataset are more than satisfactory given the time and hardware constraints as this dataset is proposed as a real challenge with many objects and occlusions.

5.2.2 Time Analysis and Optimization

In this subsection, we will give quantitative evidence of the times obtained by the different solutions, as well as a justification of the improvement obtained. First, we compare the complete JDE system as proposed in [113] without any modification using its faster model 576×320, with our system (as explained in Section 4.1) using the same configuration. These measures have been performed on the NVIDIA Jetson TX2 embedded device and over the same MOT16 test set. Table 5.6 shows the results obtained. The results show that our so-

Table 5.6: Average FPS on NVIDIA Jetson TX2

Solution	Avg. FPS MOT16	Speed up
JDE	3.4	–
Ours	25.2	7.4×

lution obtains a speed up of 7.4× with respect to JDE. This makes our system suitable for real-time processing.

One of the reasons why it has been possible to achieve this speedup has been the parallelization of the KCF algorithm on the ARM architecture. To illustrate the effect of this optimization, in Figure 5.3 we show a comparison of this isolated part of the system, that is, only the KCF, comparing the sequential version proposed in [49] against our own optimization on the ARM architecture. This is the bottleneck of our system and will determine the maximum number of objects that our system is able to manage in real-time. For the sequential KCF implementation, as the number of objects being tracked increases, it can be seen that for more than 12 objects (point A) the system is no longer able to reach real-time processing (25 FPS). However, our optimization is able to track up to 70 objects (point B) at 25 FPS or more. Thus, our system is able to operate with a large number of objects without sacrificing the application's computing time. The speedup achieved when the workload is maximum (100 objects) is 5.25×. Taking into account that the NVIDIA Jetson TX2 has 6 cores, it is close to the ideal speedup of 6. For the MOT16 dataset, the average speed up is 4.03×.

It is worth commenting on the difference in computing capability between high-end GPUs, for which many algorithms are designed, and low-power embedded systems. Table 5.7 shows the comparison between the hardware resources used by the original JDE [113] and our solution. On the one hand, JDE [113] employs an NVIDIA Titan XP GPU, while our system works in real-time on an NVIDIA Jetson TX2 system that has an NVIDIA Pascal GPU. The NVIDIA Jetson TX2 has 8 times less single-precision floating-point performance (FP32) and 16 times less power consumption (Max. PC) than the NVIDIA Titan XP. This gives a rough estimate of the challenge of achieving e processing speed on hardware with limited resources. That power consumption of 15 W would allow for an energy autonomy of more than 5 hours with a 6.000 mA.h LiPo battery. This allows the execution of the system in environments where it would not be possible if a higher consumption were required.

Table 5.7: Comparison of hardware resources between NVIDIA Titan XP GPU used in [113] and NVIDIA Jetson TX2 used in our approach. The table shows max power consumption (Max. PC) and single-precision floating-point performance (FP32).

Device	Max. PC (W)	FP32 (TFLOPS)
NVIDIA Titan XP GPU	250	12.15
NVIDIA Jetson TX2	15	1.5

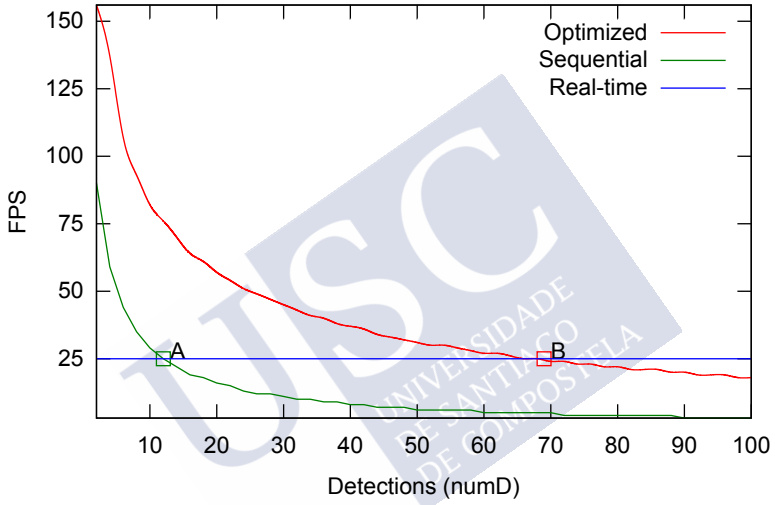


Figure 5.3: Comparison of two KCF implementations: sequential (green) and our optimization (red). Real-time speed is represented by the blue horizontal line. The KCF tracker is the bottleneck of our system.

5.3 Final Remarks

We have presented an RTMODT system for an embedded platform that uses a cutting edge deep learning architecture in conjunction with an additional tracking module —optimized for the embedded architecture— that alleviates the load of the detector and allows for better tracking between widespread detections over time.

The system achieves a real-time processing speed of 25.2 FPS while following up to 45 objects simultaneously. A comparison of our solution with previous art JDE adapted for real-time processing on an embedded system and two YOLO-SORT solutions has been made using standard multiple object tracking metrics in the MOT16 dataset. Results show an improve-

ment in MOTA ranging from 5.2 to 8.8 points. Also, our parallelization of the low-level KCF tracking algorithm for an ARM architecture has been analyzed, obtaining a maximum speed up of 5.25× with respect to the sequential version.

The system has been deployed on the NVIDIA Jetson TX2 embedded device. Based on the results obtained and taking into account the processing speed and a maximum power consumption of only 15W, our solution is a suitable option for a large number of real-life applications in which video analytics is required.

Chapter 6

Conclusions

The number of applications that can benefit from an RTMODT system is increasing every day due to the evolution of more robust detection and tracking algorithms that allow addressing problems that previously seemed unsolvable by computer vision solutions. Some examples of this type of applications are vehicle counting, pedestrian monitoring in crowded scenarios, access control, traffic accident detection, scene interpretation, and assisted surveillance. In these contexts, assigning each object with a bounding box and a unique ID is the basis for the solution to the problem and to do so at a rate equal or greater to the input video is a requirement for the successful implementation of the application.

There are several alternatives available to solve the problem, whose strengths and drawbacks have been analyzed in this thesis in Chapter 2. In this review of prior research, the need of developing real-time high accurate solutions has been identified. Detecting and tracking multiple objects in real-time is a difficult task and requires a careful design of the different parts to ensure that an adequate processing speed is achieved without compromising the accuracy and/or usability of the system.

In this Ph.D. Thesis the topic of RTMODT systems has been addressed through state-of-the-art computer vision techniques. Particularly, the work has focused on designing systems that are able to detect and assign a bounding box with an identity to the objects of interest as well as to recover them in the case of momentary occlusions. RTMODT systems have been developed both for server architectures and for embedded low-powered systems. Besides, the feasibility of the solutions developed in real-life applications has been demonstrated, both for traffic and pedestrian monitoring.

In the tracking field, we have proposed a new architecture for real-time processing. These contributions have been detailed in Chapter 3 and can be summarized as follows. First, an RTMODT system that combines a deep learning detection through a ConvNet, visual feature tracking via a DCF algorithm, motion estimation using a Kalman filter, and data association with the Hungarian method. Each of these components provides a crucial function. Deep learning detectors offer much higher accuracy than classical solutions such as background subtraction or hand-crafted feature detectors but their speed is limited. Solving the system dependency on the detector will be the task of the tracking module. DCF-based tracking allows following the objects previously detected with high accuracy for a couple of frames, so there is no need for detections in all the frames, thus increasing the speed of the system by alleviating the burden of the detector. In those cases where there are no visual features to detect the object, such as occlusions, it will not be possible to continue the tracking with the DCF, so the Kalman filter will maintain a prediction of the position and speed of the object for its later re-identification when it reappears. The combination of the DCF and Kalman filter allows using both visual features and motion prediction to estimate the error of each tracker, thus improving the number of tracking mismatches up to a 402,9% compared with just a DCF tracker, increasing in this way the reliability of the system. Finally, the Hungarian algorithm allows combining the information between detections and trackers to add new objects that appear on the screen, remove those that have left the scene and correct the position of the trackers with their corresponding detections. Thanks to this design the system can track hundreds of objects in real-time while being robust to occlusions. We have applied the RTMODT system to two different real-life applications. Firstly, we have implemented a solution of roundabout I/O analysis, whose aim is, from a video of a traffic roundabout recorded by a UAV, to detect, for each vehicle, the input and output it takes. Our system achieves a 93% success rate for the I/O matrix, even in cases with high occlusion rates, shadows, and movement of the UAV onboard camera. Secondly, a traffic anomaly detection system has been developed. From a traffic surveillance video, the system detects the direction and speed of movement of each lane. From that information, when a low-probability event occurs in any of these areas, the system launches an alarm. This allows detecting objects that circulate in the opposite direction of the road, accidents, etc. Both applications have been validated with actual traffic application datasets showing satisfactory results.

Given that many tracking applications need to be executed on embedded devices, we have deepened the study of this type of architecture. Our first contact with embedded systems has

been detailed in Chapter 4 and can be summarized as follows. First, a careful analysis has been made to select the NVIDIA Jetson TX2 as a device for our embedded implementations. It has been selected for several reasons: (i) it is a device that has an integrated GPU, which means that despite being an embedded system, it is possible to make use of deep learning solutions by using CNN-based approaches; (ii) it is also a low price and small dimension product, which is very attractive for application environments that demand to do computing on the edge while keeping the cost of the infrastructure to a minimum; (iii) it provides a wireless connection as well as a camera to capture videos, which makes it good for use at different locations; (iv) the device allows portability thanks to its small size and its compatibility with a LiPo battery. On this device, we have developed a detection and tracking approach which uses a background subtraction method for detection, namely the Hardware-Oriented PBAS (HO-PBAS) algorithm [37] and multiple instances of the GOTURN CNN based tracker [48] to track multiple objects. In this first contact with embedded architectures, object tracking is performed separately for each detected object. The dETRUSC dataset captured with the onboard camera is available online for research purposes at [130]. This dataset has been used to evaluate detection and tracking performance, frame rate, and power consumption.

With the knowledge acquired after the development of the previous solution, this line of research has been further explored in order to design an RTMODT system that is able to track dozens of objects with state-of-the-art results while maintaining real-time processing speed. These contributions have been detailed in Chapter 5. The system uses a joint architecture of detection and embedding. This allows extracting, in addition to the bounding box of the objects of interest, a vector of deep features that models their appearance. This adds robustness to the data association stage and allows for better matching and object re-identification. For the tracking module, the system uses a combination of a DCF tracker optimized for an embedded architecture and a Kalman filter whose function is to maintain the identity of the objects between widespread detections. This alleviates the load of the detector by reducing the number of detections needed and increasing the overall speed of the system. Also when an occlusion occurs the system is able to re-identify the object. Finally, the Hungarian method allows the combination of the information of the two modules -detection and tracking- taking into account firstly the embeddings and secondly the position of the objects to do the matching. The system has also been deployed on the NVIDIA Jetson TX2 embedded device, on which it reaches an average processing speed of 25.2 FPS while following up to 45 objects simultaneously with a consumption of only 15W. This makes the system a suitable option for

a large number of real-life applications in which video analytics is required. To assess the performance of the proposed solution the MOT16 has been used as the dataset to perform the metrics. Our system has been compared with other three state-of-the-art RTMODT solutions, showing an increase in MOTA ranging from 5.2 to 8.8 points. Considering the results obtained, and that it is an end-to-end solution, our system can be used satisfactorily for a large number of real-life pedestrian monitoring applications.

As a result of the work developed in this thesis, some directions of research that might be worthy to explore in the future have been identified. These ideas focus on further improving the efficiency of the proposed algorithms. As a starting point for future improvements, the solution proposed in Chapter 5 will be used, as it represents the most advanced stage of our research in RTMODT systems. One point that could considerably improve the metrics achieved by our solution for embedded systems is the quality of the detector, as it directly affects the tracking accuracy. Not only that, but by proposing a joint architecture for detection and appearance, better detections necessarily imply better feature representation (embeddings). It would be interesting to explore the integration of new one-shot detection solutions such as YOLOv4 [12] or follow an approach like the one proposed in [122] in the case of two-stage detectors.

Another factor to be improved in this type of solution is the tracking module, both visual and motion estimation. More advanced solutions based on Siamese networks like [126] and [66], or solutions using a CNN formulation from DCF such as [23], would provide greater accuracy in the visual tracking of the system. However, this task requires extensive research, since in order to be applicable in RTMODT systems these solutions should scale properly with the number of objects. That is, these visual tracking architectures should be modified so that a network forward pass computes the position of all objects present in the image instead of doing it one by one in a sequential fashion.

Another point that can improve the tracking results is more advanced motion estimators. Solutions like LSTM networks can be exploited for this task, but extensive optimization is required to perform this task in a reasonable time. However, in all cases, it is essential to take into account the trade-off between the accuracy of the solutions included and their impact on the overall computation time.

As an add-on, application-dependent modules would certainly improve metrics too. Finally, when it comes to computer vision on the edge, different research lines can be followed. On the one hand, lightweight versions of high quality or accurate RTMODT systems can be

designed, trying to lose as low accuracy as possible. On the other hand, implementations on either solutions with lower power consumption and more affordable embedded GPUs as Jetson Nano, or more powerful and expensive devices as Jetson Xavier, on the other side of the spectrum should also be explored. Embedded devices as ASICs or FPGAs would be discarded because of their very different underlying hardware from that of a GPU, resulting in very long design cycles.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Bibliography

[1] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[2] Maryam Babaee, Zimu Li, and Gerhard Rigoll. Occlusion handling in tracking multiple people using rnn. In *IEEE International Conference on Image Processing (ICIP)*, pages 2715–2719. IEEE, 2018.

[3] Nathanael L Baisa. Occlusion-robust online multi-object visual tracking using a gm-phd filter with a cnn-based re-identification. *arXiv preprint arXiv:1912.05949*, 2019.

[4] Suyoung Bang, Jingcheng Wang, Ziyun Li, Cao Gao, Yejoong Kim, Qing Dong, Yen-Po Chen, Laura Fick, Xun Sun, Ron Dreslinski, Trevor Mudge, Hun Seok Kim, David Blaauw, and Dennis Sylvester. A 288μW programmable deep-learning processor with 270KB on-chip weight storage using non-uniform memory hierarchy for mobile intelligence. In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 250–251, 2017.

[5] Olivier Barnich and Marc Van Droogenbroeck. ViBe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image Processing*, 20(6):1709–1724, 2011.

[6] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1806–1819, 2011.

[7] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

[8] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-Convolutional Siamese Networks for Object Tracking. In *European Conference on Computer Vision (ECCV)*, pages 850–865, 2016.

[9] Marco Bettoni, Gianvito Urgese, Yuki Kobayashi, Enrico Macii, and Andrea Acquaviva. A Convolutional Neural Network Fully Implemented on FPGA for Embedded Platforms. In *New Generation of CAS (NGCAS)*, pages 49–52, 2017.

[10] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016.

[11] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-speed tracking-by-detection without using image information. In *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, 2017.

[12] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[13] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2544–2550, 2010.

[14] Brais Bosquet, Manuel Mucientes, and Víctor M. Brea. STDnet: exploiting high resolution feature maps for small object detection. *Engineering Applications of Artificial Intelligence*, 91(103615), 2020.

[15] Jiahui Chen, Hao Sheng, Yang Zhang, and Zhang Xiong. Enhancing detection model for multiple hypothesis tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.

[16] Long Chen, Haizhou Ai, Chong Shang, Zijie Zhuang, and Bo Bai. Online multi-object tracking with convolutional neural networks. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 645–649. IEEE, 2017.

[17] Long Chen, Haizhou Ai, Zijie Zhuang, and Chong Shang. Real-time multiple people tracking with deeply learned candidate selection and person re-identification. *2018 IEEE International Conference on Multimedia and Expo (ICME)*, 2018.

[18] Yu-Hsin Chen, Tushar Krishna, Joel S. Emer, and Vivienne Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 52(1):127–138, 2017.

[19] Zhizhen Chi, Hongyang Li, Huchuan Lu, and Ming-Hsuan Yang. Dual deep network for visual tracking. *IEEE Transactions on Image Processing*, 26(4):2005–2015, 2017.

[20] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 379–387, 2016.

[21] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893, 2005.

[22] Martin Danelljan, Goutam Bhat, F Shahbaz Khan, and Michael Felsberg. ECO: Efficient convolution operators for tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6638–6646, 2017.

[23] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Atom: Accurate tracking by overlap maximization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4660–4669, 2019.

[24] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Discriminative scale space tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8):1561–1575, 2016.

[25] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Convolutional features for correlation filter based visual tracking. In *IEEE International Conference on Computer Vision (ICCV) Workshops*, pages 58–66, 2015.

[26] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 4310–4318, 2015.

[27] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *European Conference on Computer Vision (ECCV)*, pages 472–488, 2016.

[28] Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg, and Joost Van de Weijer. Adaptive color attributes for real-time visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1090–1097, 2014.

[29] Zidong Du, Shaoli Liu, Robert Fasthuber, Tianshi Chen, Paolo Ienne, Ling Li, Tao Luo, Qi Guo, Xiaobing Feng, Yunji Chen, and Olivier Temam. An accelerator for high efficient vision processing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(2):227–240, 2017.

[30] Volker Eiselein, Daniel Arp, Michael Pätzold, and Thomas Sikora. Real-time multi-human tracking using a probability hypothesis density filter and multiple detectors. In *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 325–330. IEEE, 2012.

[31] Heng Fan and Haibin Ling. Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking. In *IEEE International Conference on Computer Vision (ICCV)*, pages 5486–5494, 2017.

[32] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[33] Pedro F Felzenszwalb, Ross B Girshick, and David McAllester. Cascade object detection with deformable part models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2241–2248, 2010.

[34] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2009.

[35] Mauro Fernández-Sanjurjo, Brais Bosquet, Manuel Mucientes, and Víctor M Brea. Real-time visual detection and tracking system for traffic monitoring. *Engineering Applications of Artificial Intelligence*, 85:410–420, 2019.

[36] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[37] Daniel García-Lesta, Paula López, Víctor Manuel Brea, and Diego Cabello. In-pixel analog memories for a pixel-based background subtraction algorithm on CMOS vision sensors. *International Journal of Circuit Theory and Applications*, 46(9):1631–1647, 2018.

[38] Ross Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[39] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[40] Ross B Girshick, Pedro F Felzenszwalb, and David A Mcallester. Object detection with grammar models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 442–450, 2011.

[41] Vinayak Gokhale, Jonghoon Jin, Aysegul Dundar, Berin Martini, and Eugenio Culurciello. A 240 G-ops/s mobile coprocessor for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 682–687, 2014.

[42] Nil Goyette, Pierre-Marc Jodoin, Fatih Porikli, Janusz Konrad, and Prakash Ishwar. Changedetection.net: A new change detection benchmark dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop*, pages 1–8, 2012.

[43] Kaiyuan Guo, Lingzhi Sui, Jiantao Qiu, Jincheng Yu, Junbin Wang, Song Yao, Song Han, Yu Wang, and Huazhong Yang. Angel-Eye: A Complete Design Flow for Mapping CNN onto Embedded FPGA. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(1):35–47, 2017.

[44] Jussi Hanhirova, Teemu Kämäräinen, Sipi Seppälä, Matti Siekkinen, Vesa Hirvisalo, and Antti Ylä-Jääski. Latency and throughput characterization of convolutional neural networks for mobile computer vision. In *ACM Multimedia Systems Conference*, pages 204–215, 2018.

[45] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.

[46] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.

[47] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[48] David Held, Sebastian Thrun, and Silvio Savarese. Learning to Track at 100 FPS with Deep Regression Networks. In *European Conference on Computer Vision (ECCV)*, pages 749–765, 2016.

[49] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2014.

[50] Roberto Henschel, Laura Leal-Taixé2 Daniel Cremers, and Bodo Rosenhahn. Fusion of head and full-body detectors for multi-object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1428–1437, 2017.

[51] Martin Hofmann, Philipp Tiefenbacher, and Gerhard Rigoll. Background segmentation with feedback: The pixel-based adaptive segmenter. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2012.

[52] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia Systems Conference*, pages 675–678, 2014.

[53] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[54] Hasith Karunasekera, Han Wang, and Handuo Zhang. Multiple object tracking with attention to appearance, structure, motion and size. *IEEE Access*, 7:104423–104434, 2019.

[55] Margret Keuper, Siyu Tang, Yu Zhongjie, Bjoern Andres, Thomas Brox, and Bernt Schiele. A multi-cut formulation for joint segmentation and tracking of multiple objects. *arXiv:1607.06317*, 2016.

[56] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. Multiple hypothesis tracking revisited. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[57] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernandez, Tomas Vojir, Gustav Hager, Georg Nebehay, and Roman Pflugfelder. The visual object tracking vot2014 challenge results. In *European Conference on Computer Vision (ECCV) Workshops*, 2014.

[58] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernandez, Tomas Vojir, Gustav Hager, Georg Nebehay, and Roman Pflugfelder. The visual object tracking vot2015 challenge results. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2015.

[59] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernandez, Tomas Vojir, Gustav Hager, Georg Nebehay, and Roman Pflugfelder. The visual object tracking vot2016 challenge results. In *European Conference on Computer Vision (ECCV) Workshops*, 2016.

[60] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernandez, Tomas Vojir, Gustav Hager, Georg Nebehay, and Roman Pflugfelder. The visual object tracking vot2017 challenge results. In *IEEE International Conference on Computer Vision (ICCV) Workshops*, 2017.

[61] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Roman Pflugfelder, Joni-Kristian Kamarainen, Luka Cehovin Zajc, Ondrej Drbohlav, Alan Lukezic, Amanda Berg, et al. The seventh visual object tracking vot2019 challenge results. In *IEEE International Conference on Computer Vision (ICCV) Workshops*, 2019.

[62] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[63] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[64] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[65] Byungjae Lee, Enkhbayar Erdenee, Songguo Jin, Mi Young Nam, Young Giu Jung, and Phill Kyu Rhee. Multi-class multi-object tracking using changing point detection. In *European Conference on Computer Vision*, pages 68–83. Springer, 2016.

[66] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. SiamRPN++: Evolution of siamese visual tracking with very deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4282–4291, 2019.

[67] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8971–8980, 2018.

[68] Xin Li, Kejun Wang, Wei Wang, and Yang Li. A multiple object tracking method using kalman filter. In *IEEE international Conference on Information and Automation*, pages 1862–1866. IEEE, 2010.

[69] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2117–2125, 2017.

[70] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[71] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

[72] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8759–8768, 2018.

[73] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[74] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37, 2016.

[75] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[76] Xin Lu, Buyu Li, Yuxin Yue, Quanquan Li, and Junjie Yan. Grid R-CNN. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7363–7372, 2019.

[77] Chao Ma, Xiaokang Yang, Chongyang Zhang, and Ming-Hsuan Yang. Long-term correlation tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5388–5396, 2015.

[78] Yufei Ma, Yu Cao, Sarma Vrudhula, and Jae-sun Seo. Optimizing the convolution operation to accelerate deep neural networks on fpga. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pages 1–14, 2018.

[79] Michel Meneses, Leonardo Matos, Bruno Prado, André de Carvalho, and Hendrik Macedo. Learning to associate detections for real-time multiple object tracking. *arXiv preprint arXiv:2007.06041*, 2020.

[80] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016.

[81] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4293–4302, 2016.

[82] Eriko Nurvitadhi, Suchit Subhaschandra, Guy Boudoukh, Ganesh Venkatesh, Jaewoong Sim, Debbie Marr, Randy Huang, Jason Ong Gee Hock, Yeong Tat Liew, Krishnan Srivatsan, and Duncan Moss. Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks? In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays - FPGA '17*, pages 5–14, 2017.

[83] Massimo Piccardi. Background subtraction techniques: a review. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 3099–3104. IEEE, 2004.

[84] Yuran Qiao, Junzhong Shen, Tao Xiao, Qianming Yang, Mei Wen, and Chunyuan Zhang. Fpga-accelerated deep convolutional neural networks for high throughput and energy efficiency. *Concurrency and Computation: Practice and Experience*, 29(20), 2017.

[85] Jiantao Qiu, Sen Song, Yu Wang, Huazhong Yang, Jie Wang, Song Yao, Kaiyuan Guo, Boxun Li, Erjin Zhou, Jincheng Yu, Tianqi Tang, and Ningyi Xu. Going deeper with embedded FPGA platform for convolutional neural network. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays - FPGA '16*, pages 26–35, 2016.

[86] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[87] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7263–7271, 2017.

[88] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[89] Donald Reid et al. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, 1979.

[90] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

[91] Pol Rosello and Mykel J Kochenderfer. Multi-agent reinforcement learning for multi-object tracking. In *International Conference on Autonomous Agents and MultiAgent Systems*, pages 1397–1404, 2018.

[92] Manuele Rusci, Davide Rossi, Elisabetta Farella, and Luca Benini. A sub-mw iot-endnode for always-on visual monitoring and smart triggering. *IEEE Internet of Things Journal*, 4(5):1284–1295, 2017.

[93] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *IEEE International Conference on Computer Vision (CVPR)*, pages 300–311, 2017.

[94] Nimish Shah, Paragkumar Chaudhari, and Kuruvilla Varghese. Runtime programmable and memory bandwidth optimized fpga-based coprocessor for deep convolutional neural network. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2018.

[95] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[96] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[97] Young-min Song, Kwangjin Yoon, Young-Chul Yoon, Kin-Choong Yow, and Moongu Jeon. Online multi-object tracking framework with the gmphd filter and occlusion group management. *arXiv preprint arXiv:1907.13347*, 2019.

[98] Chris Stauffer and W Eric L Grimson. Adaptive background mixture models for real-time tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 246–252, 1999.

[99] Manuel Suárez, Víctor Manuel Brea, Jorge Fernández-Berni, Ricardo Carmona-Galán, Diego Cabello, and Ángel Rodríguez-Vázquez. Low-power CMOS vision sensor for Gaussian pyramid extraction. *IEEE Journal of Solid-State Circuits*, 52(2):483–495, 2017.

[100] Vivienne Sze, Yu-Hsin Chen, Joel Emer, Amr Suleiman, and Zhengdong Zhang. Hardware for Machine Learning: Challenges and Opportunities. In *IEEE Custom Integrated Circuits Conference (CICC)*, 2017.

[101] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel Emer. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proccedings of the IEEE*, 105(12):2295–2329, 2017.

[102] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper

with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[103] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 9627–9636, 2019.

[104] Jasper R. R. Uijlings, Koen E. A. Van De Sande, Theo Gevers, and Arnold W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[105] Delia Velasco-Montero, Jorge Fernández-Berni, Ricardo Carmona-Galán, and Ángel Rodríguez-Vázquez. Optimum selection of DNN model and framework for edge inference. *IEEE Access*, 6:51680–51692, 2018.

[106] Delia Velasco-Montero, Jorge Fernández-Berni, Ricardo Carmona-Galán, and Ángel Rodríguez-Vázquez. Previous: A methodology for prediction of visual inference performance on iot devices. *IEEE Internet of Things Journal*, 2020.

[107] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[108] Paul Viola and Michael J Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[109] Chao Wang, Lei Gong, Qi Yu, Xi Li, Yuan Xie, and Xuehai Zhou. DLAU: A Scalable Deep Learning Accelerator Unit on FPGA. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(3):513–517, 2017.

[110] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. CSPNet: A new backbone that can enhance learning capability of cnn. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 390–391, 2020.

[111] Lu Wang, Lisheng Xu, Min Young Kim, Luca Rigazico, and Ming-Hsuan Yang. Online multiple object tracking via flow and convolutional features. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3630–3634. IEEE, 2017.

[112] Xiaying Wang, Michele Magno, Lukas Cavigelli, and Luca Benini. Fann-on-mcu: An open-source toolkit for energy-efficient neural network inference at the edge of the internet of things. *IEEE Internet of Things Journal*, 7(5):4403–4417, 2020.

[113] Zhongdao Wang, Liang Zheng, Yixuan Liu, and Shengjin Wang. Towards real-time multi-object tracking. *arXiv:1909.12605*, 2019.

[114] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *IEEE international conference on image processing (ICIP)*, pages 3645–3649, 2017.

[115] Bo Wu and Ram Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75(2):247–266, 2007.

[116] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2411–2418, 2013.

[117] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. Accessed: 2020-12-20.

[118] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1492–1500, 2017.

[119] Changjiang Yang, Ramani Duraiswami, and Larry Davis. Fast multiple object tracking via a hierarchical particle filter. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 212–219. IEEE, 2005.

[120] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks. In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 161–170, 2015.

[121] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Single-shot refinement neural network for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4203–4212, 2018.

[122] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *arXiv preprint arXiv:2004.01888*, 2, 2020.

[123] Dawei Zhao, Hao Fu, Liang Xiao, Tao Wu, and Bin Dai. Multi-object tracking with correlation filter for autonomous vehicle. *Sensors*, 18(7):2004, 2018.

[124] Z. Zhou, J. Xing, M. Zhang, and W. Hu. Online multi-target tracking with tensor-based high-order graph matching. In *International Conference on Pattern Recognition (ICPR)*, pages 1809–1814, 2018.

[125] Sijie Zhu, Chen Chen, and Waqas Sultani. Video anomaly detection for smart surveillance. *arXiv preprint arXiv:2004.00222*, 2020.

[126] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *European Conference on Computer Vision (ECCV)*, pages 101–117, 2018.

[127] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision (ECCV)*, pages 391–405, 2014.

[128] Andrew Ng Web site. https://www.andrewng.org/, 2020. Accessed: 2020-05-18.

[129] Deep learning AI Developer. https://www.nvidia.com/en-us/deep-learning-ai/developer/, 2020. Accessed: 2020-12-20.

[130] dETRUSC video dataset. https://citius.usc.es/investigacion/datasets/detrusc, 2020. Accessed: 2020-12-20.

[131] DGT: Dirección General de Tráfico. http://www.dgt.es/es/, 2020. Accessed: 2020-12-20.

[132] Facebook to open-source AI hardware design. https://code.facebook.com/posts/1687861518126048/facebook-to-open-source-ai-hardware-design/, 2020. Accessed: 2020-12-20.

[133] Intel Xeon Phi Processor 7290F. https://ark.intel.com/products/95831/Intel-Xeon-Phi-Processor-7290F-16GB-1_50-GHz-72-core, 2020. Accessed: 2020-12-20.

[134] Intel Xeon Phi Processor 7295. https://ark.intel.com/products/128690/Intel-Xeon-Phi-Processor-7295-16GB-1_5-GHz-72-Core, 2020. Accessed: 2020-12-20.

[135] Introducing Big Basin: Our next-generation AI hardware. https://code.facebook.com/posts/1835166200089399/introducing-big-basin-our-next-generation-ai-hardware/, 2020. Accessed: 2020-12-20.

[136] MOTChallenge the multiple object tracking benchmark. https://motchallenge.net/, 2020. Accessed: 2020-12-20.

[137] NVIDIA Jetson. The embedded platform for autonomous everything. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems-dev-kits-modules/, 2020. Accessed: 2020-12-20.

[138] Pixabay License. https://pixabay.com/es/service/license/, 2020. Accessed: 2020-05-18.

[139] UCF-Crime Dataset. https://webpages.uncc.edu/cchen62/dataset.html, 2020. Accessed: 2020-12-20.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

List of Figures

Fig. 1.1 Example of multiple object detection and tracking (MODT) for basketball players. The detection module identifies the objects of interest. The tracking module calculates the position of previous trackers in the current frame. The data association module matches detections and trackers to correct, add, and remove trackers. 2

Fig. 1.2 Some examples of applications of MODT systems. (a) assisted surveillance application in airports. (b) and (c) show applications of detection and tracking of vehicles and pedestrians, respectively. (d) drone surveillance in restricted aerial spaces. In all cases it is necessary to identify the object (bounding box), its category and to assign it a unique ID (represented by the color of the bounding box). Images with Pixabay License [138]. 3

Fig. 1.3 Differences between an MOT approach and an MODT system. An MOT approach (left) can have the detection module detached from the solution, whereas an MODT system (right) must include it within the system. Both solutions should have the stages of detection, affinity, and association (represented in blue) while motion estimation and feature extraction (represented in red) are optional. In the case of MODT systems, the motion estimation stage (represented in green) becomes necessary when detections can not be guaranteed in all frames —which frequently occurs for real-time operation. . 10

Fig. 2.1 Example of background subtraction. From an input video (left) the algorithm generates a model (right) that allows identifying the static part of the image or background (black) and the mobile objects or foreground (white). This allows the subsequent framing of the objects through the minimum bounding box that encloses each white blob. Images with Pixabay License [138]. . . . 23

Fig. 2.2 Example of HOG feature extraction. From an input image (left) the distribution of intensity gradients for each cell is calculated (right). Images with Pixabay License [138]. 24

Fig. 2.3 An object detection timeline showing some of the most relevant methods for each period. Classic hand-crafted milestones: VJ [107, 108], HOG [21], DPM [32]. Deep learning methods: AlexNet [62], R-CNN [39], Fast R-CNN [38], Faster R-CNN [90], YOLO [86], SSD [74], RetinaNet [70], Mask R-CNN [45], FPN [69], RefineDet [121], FCOS [103], YOLOv4 [12]. . . . 27

Fig. 2.4 VOT example. In (a) the tracking is initialized with a provided bounding box (in green). In the following frames (b), (c) and (d) the tracker estimates the position of the object adjusting its bounding box accordingly. 30

Fig. 2.5 DCF tracking example. The response map is calculated from the maximum value of the correlation between the search region and the filter. The target center will be represented by the maximum value of the response map. . . . 31

Fig. 2.6 Basic framework for object tracking with siamese networks [8]. The algorithm attempts to locate a smaller template image within a search region. This is done by first producing learned embeddings for both these images and then cross-correlating these features. The resultant is a spatial score map containing confidences of presence of the template within the search region. 34

Fig. 2.7 Example of how MOT algorithms perform the tracking task. In (a) and (b) the result of the object detection process is shown —numbers at the top of the bounding represent the detector confidence. In (c) and (d) the outcome of an ideal tracking is displayed, where each object has a unique ID and it is kept as long as it is visible. 35

Fig. 2.8 The usual operation of an MOT algorithm. Given the raw frames of a video (i), object detection is performed to obtain the bounding boxes of the objects of interest (ii) —usually, these detections are provided by an MOT benchmark like MOTChallenge [136]. Then, for each detection visual features and/or motion information is obtained (iii). Using this information an affinity measure is established, e.g. distance function, to estimate the probability of two detections corresponding to the same target (iv). Finally, the data association assigns the corresponding ID to each object. 36

Fig. 2.9 Jetson TX2 development kit with its integrated GPU was selected as our embedded device for integrating our embedded developments. 43

Fig. 3.1 Architecture of our MODT system. 47

Fig. 3.2 (a) The low-level DCF tracker (in green) cannot recover the identity of the object once occluded as it only relies on appearance. (b) The combination of a DCF and a KF manages occlusions, as it also takes into account the object motion model. Images courtesy of Aplygenia S.L. 50

Fig. 3.3 PSR values are poor predictors of tracking failures for the DCF tracker.(a) An ideal case in which PSR values predict correctly a tracking failure. (b) A vehicle is being tracked correctly but the PSR distribution indicates a tracking loss. (c) Tracking loss is not detected by PSR as the values do not change in both situations. Images courtesy of Aplygenia S.L. 50

Fig. 3.4 Creation of a search ROI for occlusion handling. (a) Both tracking methods agree on the object position. (b) As the DCF fails to track the occluded object, the distance between both estimations increases and so it does the search ROI. This process continues in (c) and finally in (d), when the detector finds the vehicle at the other side of the road and the tracker recovers. Images courtesy of Aplygenia S.L. 51

Fig. 3.5 Example frames of our video dataset to evaluate tracking metrics. These videos are from traffic monitoring cameras. Images courtesy of DGT. 54

Fig. 3.6 MOTP results for DCF and DCF+KF with several PSR thresholds and different number of frames between detector calls for the videos outlined in Table 3.1. 55
Fig. 3.7 MOTA results for DCF and DCF+KF with several PSR thresholds and different number of frames between detector calls. 55
Fig. 3.8 Frame processing of our system with an input video. We show the first 15 frames as an example. The system performs tracking one of every 3 frames (orange), and detection one of every 6 frames (blue). 58
Fig. 3.9 Example frames of some videos of the roundabout monitoring dataset. These videos are recorded from a UAV flying over a roundabout. Images courtesy of Aplygenia S.L. 59
Fig. 3.10 Examples of the movement map generated in traffic videos for accident detection. For each cell we show just the mean angle. Images courtesy of DGT. 60
Fig. 3.11 Example of an alarm for a vehicle moving in a wrong direction. In (a) the system detects an object moving with a direction contrary to the usual one in that lane, so an alarm is launched. In (b) the system keeps tracking the object despite being totally occluded. In (c) the system continues the tracking after the occlusion, keeping the same ID and the alarm. Images courtesy of DGT. 61
Fig. 3.12 Example of an alarm for an accident. In (a) a motorcycle crashes with a car. In (b) the system detects both vehicles as static in a lane that has movement, so it triggers an alarm. Finally in (c) the system keeps triggering alarms for every car that stops in that area. Images courtesy of DGT. 61
Fig. 3.13 Frames of a video that has both alarms. In (a) a truck starts to derail. In (b) the system detects an anomaly in the second lane when the truck invades it so it triggers the wrong direction alarm. In (c) when the crashed truck remains static the system triggers the velocity alarm and it does the same for every truck that moves with abnormal velocity (reduced speed). Images courtesy of DGT. 62

Fig. 4.1 Embedded solution: Jetson TX2 development kit is battery powered and remotely controlled using a tablet and WiFi connection. 67
Fig. 4.2 Conceptual diagram of the multiple object tracking algorithm proposed. . . . 69
Fig. 4.3 Example of multiple object tracking under different challenging conditions. . 72

Fig. 4.4 Success and precision plots for some videos of the dETRUSC dataset and arithmetic mean. 73
Fig. 4.5 Frame rate of the multiple object tracking algorithm for different number of tracked objects and Jetson TX2 operation modes. 76
Fig. 4.6 Total power consumption of the system for different number of tracked objects and Jetson TX2 operation modes. The bar increment represents the additional energy consumption on full clocks mode. 77
Fig. 4.7 Power consumption of the CPUs and GPU for different number of tracked objects and Jetson TX2 operation modes. 78

Fig. 5.1 Different components of the RTMODT system. Object detection is carried out by a JDE-based detector —module (a)—, which provides a bounding box for each detected object as well as a feature vector that models its appearance. Tracking —module (b)— is performed by a combination of two different trackers: an appearance-based tracker —Kernelized Correlation Filter (KCF)— and a motion-based Kalman filter (KF). Data association —module (c)— is solved in two steps: (i) associations based on visual features (embedding) which also use motion information; (ii) association by position. Real-time processing speed is achieved through an optimal distribution of the components among the available hardware: detection on GPU, and tracking and data association on CPU. 82
Fig. 5.2 Sample frames of our system running on some of the MOT 16 test set videos on the NVIDIA Jetson TX2 embedded device. 92
Fig. 5.3 Comparison of two KCF implementations: sequential (green) and our optimization (red). Real-time speed is represented by the blue horizontal line. The KCF tracker is the bottleneck of our system. 95

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

List of Tables

Tab. 2.1 Top five configurations for object detection in Detectron2 [117] (Facebook). Architecture: Network used —RetinaNet [70] or Faster R-CNN [90]. Backbone: R101 —ResNet-101; X101 —ResNeXt-101; FPN —use a ResNet+FPN backbone with standard convolution and FC heads for mask and box prediction, respectively; C4 —use a ResNet conv4 backbone with conv5 head; DC5 (Dilated-C5) —use a ResNet conv5 backbone with dilations in conv5, and standard convolution and FC heads for mask and box prediction, respectively. inf_t (s/im): inference time in seconds per image. FPS: inference rate in frames per second. train_mem (GB): Memory needed for training in GB. boxAP: box Average Precision (%) in MS. COCO Object Detection Challenge [71]. 29

Tab. 2.2 Algorithms submitted to MOT16 Challenge that operate in real-time –i.e. equal to or greater than 25FPS. For each algorithm it is displayed the scientific publication on which it is based, its name in the MOT16 leaderboard, the main MOT metrics (MOTA, ID F1 Score and MOTP), its processing speed (in frames per second excluding the detector) and, finally, we show if the detector uses public or private detections, and if it is an online method. . . . 39

Tab. 2.3 Algorithms submitted to MOT17 Challenge that operate in real-time –i.e. equal to or greater than 25FPS. For each algorithm it is displayed the scientific publication on which it is based, its name in the MOT17 leaderboard, the main MOT metrics (MOTA, ID F1 Score and MOTP), its processing speed (in frames per second excluding the detector) and, finally, we show if the detector uses public or private detections, and if it is an online method. . . . 39
Tab. 2.4 Algorithms submitted to MOT20 Challenge that operate in real-time –i.e. equal to or greater than 25FPS. For each algorithm it is displayed the scientific publication on which it is based, its name in the MOT20 leaderboard, the main MOT metrics (MOTA, ID F1 Score and MOTP), its processing speed (in frames per second excluding the detector) and, finally, we show if the detector uses public or private detections, and if it is an online method. . . . 40
Tab. 2.5 Hardware comparison for Jetson modules. 44

Tab. 3.1 Dataset with more than 1,000 different vehicles in more than 20 minutes of video to evaluate tracking metrics. 54
Tab. 3.2 Reinitializations (Reinit), Misses (M), false positives (FP) and mismatches (MM) in the videos of Table 3.1 for a PSR threshold of 13 (standard). The right columns show the improvement of DCF+KF over DCF. 56
Tab. 3.3 Computational times for the tracking and detection modules of the RTMODT system. 58
Tab. 3.4 Results in the video dataset for roundabout monitoring. The columns are: video, number of occlusions (#*occ*), number of vehicles occluded (#*vocc*), duration of video, total number of vehicles (#*vehicles*) and success rate obtained by our tracking system. 60

Tab. 4.1 NVIDIA Jetson TX2 Operation Modes 75

Tab. 5.1 Tracking results of our system in MOT16 test set. 90
Tab. 5.2 Tracking results of JDE-RT in MOT16 test set. 91
Tab. 5.3 Tracking results of Tiny-YOLOv4-SORT in MOT16 test set. 91
Tab. 5.4 Tracking results of Tiny-YOLOv3-SORT in MOT16 test set. 91
Tab. 5.5 Comparative results of all solutions. 92
Tab. 5.6 Average FPS on NVIDIA Jetson TX2 94

Tab. 5.7 Comparison of hardware resources between NVIDIA Titan XP GPU used in [113] and NVIDIA Jetson TX2 used in our approach. The table shows max power consumption (Max. PC) and single-precision floating-point performance (FP32). 95

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA