Universidad de Alcalá
Escuela Politécnica Superior

Máster Universitario en Ingeniería Industrial

Trabajo Fin de Máster

Detección, Segmentación y Seguimiento Multi-objeto utilizando un procedimiento semiautomático de anotaciones a nivel de píxel empleando el conjunto de datos KITTI MOTS

Autor: Andrea Pérez Alonso

Tutor: Manuel Ocaña Miguel

2022

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Máster Universitario en Ingeniería Industrial

Trabajo Fin de Máster

Detección, Segmentación y Seguimiento Multi-objeto utilizando un procedimiento semiautomático de anotaciones a nivel de píxel empleando el conjunto de datos KITTI MOTS

Autor: Andrea Pérez Alonso

Director: Manuel Ocaña Miguel

Tribunal:

Presidente: Luis Miguel Bergasa Pascual

Vocal 1º: Hilario Gómez Moreno

Vocal 2º: Manuel Ocaña Miguel

Calificación: .....

Fecha: .....

A mis padres...

"La familia no es algo importante. Lo es todo."
George Santayana

Agradecimientos

Todos los triunfos nacen cuando nos atrevemos a comenzar.

Eugene Fitch Ware

A mis padres Alicia y Juan Carlos, quienes con su amor, dedicación y esfuerzo me han permitido llegar a cumplir hoy un sueño más. Gracias por luchar conmigo siempre.

A toda mi familia, por su apoyo incondicional durante todo este proceso. Gracias a sus consejos hacen de mi una mejor persona y sé que me acompañan en todas mis metas.

A mi pareja Daniel, porque cada etapa que cierro hace que se abra una nueva juntos. Gracias por tanto ánimo, consuelo y motivación, pero sobre todo gracias por tanto amor. Mis ganas de comerme el mundo son las tuyas.

A todos mis amigos, por alegrarse de mis logros y brindarme su mano en mis fracasos. Nos quedan muchas cosas buenas por vivir.

A mis profesores, ya que gracias a sus conocimientos hacen que pueda crecer día a día profesionalmente.

Por último, y no por ello menos importante, a mi tutor Manuel Ocaña, quien con su dirección, enseñanza, atención y colaboración permitió la realización de este trabajo.

Resumen

En este Trabajo Fin de Máster (TFM) se van a estudiar distintos métodos de detección, segmentación y seguimiento multi-objeto existentes recientemente que utilizan el conjunto de datos de entrenamiento y prueba de la base de datos Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago (KITTI), tomando como partida la técnica propuesta en el artículo de investigación "MOTS: Multi-Object Tracking and Segmentation", presentado en el congreso "Conference on Computer Vision and Pattern Recognition (CVPR)" del año 2019. Los métodos estudiados participan en Multi-object Tracking And Segmentation (MOTS) Evaluation de "The KITTI Vision Benchmark Suite" y sus códigos son abiertos para poder entrenar con ellos. Por lo tanto, se reproducirán los resultados obtenidos de cada método en KITTI MOTS Validation.

Palabras clave: MOTS, KITTI.

Abstract

In this Master's Thesis, I'm going to study differents recently existing multi-object detection, segmentation and tracking methods that use the training and testing dataset of the KITTI database, taking as a starting point the technique proposed in the research paper "MOTS: Multi-Object Tracking and Segmentation", presented at the "CVPR" conference in 2019. The studied methods participate in MOTS Evaluation of "The KITTI Vision Benchmark Suite" and their codes are open for training with them. Therefore, the results obtained by each method in KITTI MOTS Validation will be reproduced.

Keywords: MOTS, KITTI.

Índice general

Índice de figuras

Índice de tablas

Lista de acrónimos

AssA	Association Accuracy.
BDD	Berkeley Deep Drive.
CAMOT	Class-Agnostic Multi-Object Tracker.
CDTS	Collaborative Detection, Tracking, and Segmentation.
CIWT	Combined Image- and World-Space Tracking.
CNN	Convolutional Neuronal Network.
CVPR	Conference on Computer Vision and Pattern Recognition.
DAVIS	Densely Annotated VIdeo Segmentation.
DetA	Detection Accuracy.
FAST	Funciones de Accelerated Segment Test.
FBMS	Freiburg-Berkeley Motion Segmentation.
FPN	Feature Pyramid Network.
GPS	Global Positioning System.
HOTA	Higher Order Tracking Accuracy.
IDS	Id SwitcheS.
IMU	Inertial Measurement Unit.
IoU	Intersection-over-Union.
JDE	Joint Detection and Embedding.
KINS	KITTI INStance.
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago.
KNN	K-Nearest-Neighbor.
LIDAR	Laser Imaging Detection and Ranging.
LSTM	Long Short-Term Memory.
MLP	Multi-Layer Perceptron.

MOT	Multi-object Tracking.
MOTA	Multi-Object Tracking Accuracy.
MOTP	Multi-Object Tracking Precision.
MOTS	Multi-object Tracking And Segmentation.
MOTSA	Multi-Object Tracking and Segmentation Accuracy.
MOTSP	Multi-Object Tracking and Segmentation Precision.
MS COCO	Microsoft Common Objects in COntext.
NMS	Non Maximum Suppression.
NuScenes	NuTonomy Scenes.
PWC	Pyramid, Warping, and Cost.
R-CNN	Regions with CNN features.
RADAR	Radio Detection And Ranging.
ReLU	Rectified Linear Unit.
RGB	Red Green Blue.
RLE	Run-Length Encoding.
RNN	Recurrent Neural Network.
RoI	Region of interest pooling.
ROLO	Recurrent YOLO.
RPN	Region Proposal Network.
RRC	Recurrent Rolling Convolution.
SFM	Structure from motion.
SIFT	Scale-invariant feature transform.
SLAM	Simultaneous Localization And Mapping.
sMOTSA	Soft Multi-Object Tracking and Segmentation Accuracy.
SORT	Simple Online and Realtime Tracking.
SSD	Single Shot MultiBox Detector.
SURF	Speeded-Up Robust Features.
SVM	Support Vector Machine.
TFM	Trabajo Fin de Máster.
TMR	Trusted Motion Region.
UA-DETRAC	University at Albany DETection and tRACking.
VCN	Volumetric Correspondence Networks.
VOS	Video Object Segmentation.
YOLO	You Only Look Once.

Lista de símbolos

h	Altura.
w	Anchura.
α_d	Asociación de vectores.
D	Conjunto de detecciones de un vídeo.
FN	Conjunto de falsos negativos.
FP	Conjunto de falsos positivos.
t_m	Tiempo frame.
M	Número de Ground Truth.
H	Hipótesis enmascaradas.
K	Número de Hipótesis enmascaradas.
id_h	Hipótesis track.
id_d	Identificador.
N	Número de máscaras de píxeles.
$mask_d$	Máscara.
i_d	Seguimiento de Ground Truth.
t_d	Marco de tiempo de detección de máscaras.
T	Marco de tiempo de un vídeo.
t_h	Tiempo de Hipótesis track.
TP	Conjunto de verdaderos positivos.

Capítulo 1

Introducción

1.1 Seguimiento de objetos

En la actualidad, el seguimiento de objetos es de gran aplicación en diversos campos [33]. Estos pueden ser los medios de comunicación y realidad aumentada (industrias del cine y la televisión), aplicaciones médicas e investigación biológica (ayuda en el diagnóstico, estimar la posición de determinados tejidos blandos o de instrumentos como por ejemplo agujas durante al cirugía), vigilancia e inteligencia de negocios, tele-colaboración y juegos interactivos, instalaciones de arte y espectáculos, etc.

Uno de los grandes retos de la ingeniería es el desarrollo de sistemas autónomos capaces de ayudar a los humanos en las tareas cotidianas. Un ejemplo de ello son los sistemas de conducción autónoma, que pueden contribuir a reducir las muertes causadas por accidentes de tráfico. Aunque en los últimos años se han utilizado diversos sensores novedosos para tareas como el reconocimiento, la navegación y la manipulación de objetos, los sensores visuales rara vez se aprovechan en aplicaciones robóticas: Los sistemas de conducción autónoma sobre todo están basados Global Positioning System (GPS), telémetros láser y radar, así como en mapas muy precisos del entorno. En los últimos años se ha desarrollado un número cada vez mayor de puntos de referencia para impulsar el rendimiento de los sistemas de reconocimiento visual.

El seguimiento de objetos permite estimar en el tiempo la posición de uno o varios objetos móviles mediante el uso de una cámara. La forma en la que las cámaras de vídeo capturan información sobre los objetos es en forma de conjunto de píxeles.

A lo largo del tiempo se han creado distintos métodos y aplicaciones para el seguimiento de objetos. En la última década, estos se han visto mejorados gracias al aumento de calidad y resolución de los sensores de imagen junto al incremento de la potencia de cálculo.

La gran cantidad de datos que contiene un vídeo y las técnicas que se deben aplicar para el reconocimiento de objetos provoca que sea un proceso lento y complejo.

Además, el seguimiento puede verse afectado por diferentes factores que hacen que sea una tarea difícil. Uno de ellos es el *clutter*, que ocurre cuando el aspecto del objeto de interés dentro de la imagen puede ser similar al resto de objetos o al fondo. Esto provoca que en las áreas no deseadas se obtengan características que son muy difíciles de comparar con las que se espera que realmente genere el objeto. Otro de los factores es el cambio de aspecto del objeto en el plano de la imagen que puede sufrir debido a cambios de posición (por ejemplo, al girar), a la iluminación ambiente o global (dirección, intensidad o color de la luz), al ruido (en función de la calidad del sensor) o a las oclusiones (parcial o totalmente tapado por otro objeto).

El seguimiento de objetos sigue siendo un desafío, especialmente cuando se trata de múltiples objetos. En particular, los resultados de las evaluaciones de seguimiento recientes muestran que el rendimiento del rastreo a nivel de caja delimitadora, *Bounding Box*, está saturado [1].

Trabajando al nivel de los píxeles se pueden conseguir mejoras de lo anterior. Para ello, es necesario que las tres tareas de detección, segmentación y seguimiento de objetos se evalúen de manera conjunta.

Los conjuntos de datos permiten entrenar y evaluar modelos, en este caso algoritmos de visión artificial en escenarios de conducción autónoma. Éstos contienen datos de imágenes reales recopilados de escenas como áreas urbanas, pueblos y carreteras.

La tarea de segmentación no suele proporcionar anotaciones sobre datos de vídeo o información sobre identidades de objetos en diferentes imágenes. Por otro lado, los conjuntos de datos más comunes para el seguimiento de múltiples objetos sólo proporcionan anotaciones de objetos a nivel de *Bounding Boxes*. Estas *boxes* pueden ser demasiado gruesas, por ejemplo, cuando los objetos están parcialmente ocluidos de tal manera que sus *Bounding Boxes* contienen más información de otros objetos que de ellos mismos. En la Figura 1.2 se puede contemplar la comparativa de la tarea de segmentación frente al rastreo a nivel de *Bounding Box*. Cuando los objetos se cruzan, grandes partes de la *Bounding Box* de un objeto pueden pertenecer a otra instancia, mientras que las máscaras de segmentación por píxel localizan los objetos con precisión.

Es en estos casos donde la segmentación de los objetos a nivel de píxeles da como resultado una descripción más natural de la escena y puede proporcionar información adicional para los pasos de procesamiento subsiguientes. Por el contrario, los seguimientos con *Bounding Boxes* superpuestas crean ambigüedades cuando se comparan con la *Gound Truth* que normalmente deben ser resueltas en el momento de la evaluación mediante procedimientos de emparejamiento heurístico. Los resultados del seguimiento basado en la segmentación por definición no se superponen y por lo tanto pueden compararse con la *Gound Truth* para fundamentar la verdad de una manera directa.

En el artículo de investigación "MOTS: Multi-Object Tracking and Segmentation" [1] presentado en el congreso "CVPR" del año 2019, proponen ampliar la conocida tarea de Seguimiento de múltiples objetos al Seguimiento de la Segmentación de la instancia. Este artículo representa la base de trabajo de este proyecto.

1.2 Contexto

La población de adultos mayores aumenta año tras año en los países desarrollados, según la OCDE. En 2030, uno de cada tres conductores españoles tendrá más de 65 años. Conducir es clave para la independencia y la movilidad de nuestros mayores, pero existe cierta preocupación por el deterioro de sus habilidades a medida que envejecen. La automatización de vehículos puede contribuir a este objetivo incorporando Sistemas Avanzados de Asistencia al Conductor (ADAS). La conducción autónoma se ha convertido recientemente en un tema candente entre los fabricantes de automóviles y los investigadores. Existen numerosos proyectos diferentes de vehículos totalmente autónomos en diversas etapas de desarrollo. Sin embargo, la tecnología de vehículos autónomos está lejos de estar lista para implementarse a gran escala. Además, existe poca investigación en el desarrollo de sistemas diseñados para el sector de la tercera edad que analicen los factores humanos. Los conductores mayores no exigen sistemas de conducción totalmente autónomos, lo que quieren son sistemas que les permitan conducir de forma segura. Consciente de este nicho de investigación, la UAH ha estado trabajando en el proyecto SmartElderlyCar (2016-2018), desarrollando un primer prototipo de coche con algunas funciones autónomas. Sobre la base de nuestros resultados anteriores en este proyecto, el objetivo principal de la propuesta de Techs4AgeCar

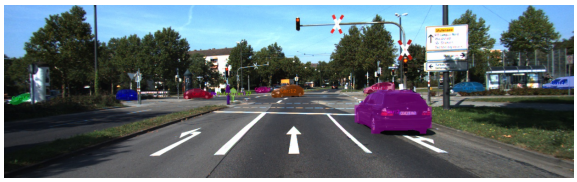
(a) 000087

(b) 000088

(c) 000089

(d) 000090

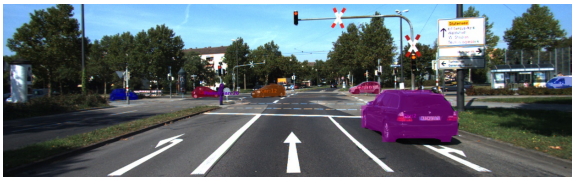
(e) 000091



(f) 000092

(g) 000093

(h) 000094



(i) 000095

(j) 000096

Figura 1.1: Visualización del seguimiento multi-objeto del video 0002 de la secuencia 000087 a la 000096 del conjunto de datos KITTI MOTS

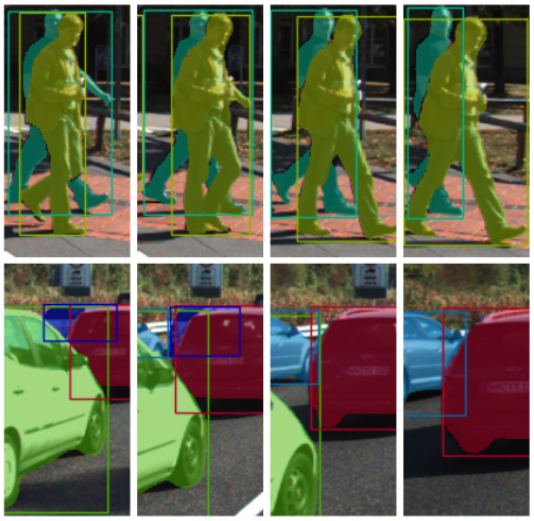


Figura 1.2: Segmentación frente a *Bounding Boxes* del conjunto de datos KITTI MOTS [1]

es investigar técnicas para un nuevo concepto de automóvil eléctrico automático (AgeCar), capaz de ayudar a los conductores mayores con diferentes niveles de automatización, de acuerdo con las necesidades de los conductores y teniendo en cuenta la seguridad y la aceptación de los usuarios como puntos clave de desarrollo. La propuesta es disruptiva tanto en el concepto como en las nuevas técnicas aplicadas. Nuestra apuesta es por un vehículo eléctrico personal, donde la automatización actúe como guardián y no como chofer, y que puede ser de interés no solo para los grandes fabricantes de automóviles, sino también para las empresas españolas emergentes de vehículos eléctricos (Comarth, Little Cars, Irikar, etc). El proyecto pretende contribuir en diferentes campos de investigación. En cuanto a los sistemas de percepción y HMI, nuestra propuesta se centrará, por un lado, en el desarrollo de técnicas robustas para tener un conocimiento completo del entorno del coche, aplicando conceptos como predicción e incertidumbre. Por otro lado, investigamos en un protocolo de comunicación vehículo-usuario (V2U) para mejorar la interpretabilidad y la transferencia segura de control entre niveles de automatización. Las contribuciones en planificación y control tienen como objetivo desarrollar una capa de navegación local robusta que tenga en cuenta la incertidumbre, desarrollar una capa de control ejecutivo capaz de tomar decisiones seguras para los diferentes casos de uso y proporcionar información desde la arquitectura de control hasta la interfaz V2U. mejorar la interpretabilidad de las acciones autónomas. Finalmente, Techs4AgeCar tiene como objetivo allanar el camino para una buena adopción de nuestro concepto AgeCar avanzando en los protocolos de validación. De esta forma, se implementará un conjunto de nuevos escenarios tipo Euro NCAP para la conducción automatizada en presencia de usuarios vulnerables (VRU) y otros vehículos realizando un análisis repetitivo utilizando diferentes conductores senior, con especial foco en la aceptación del usuario. Este TFM se sitúa dentro de este proyecto de investigación, Techs4AgeCar del grupo RobeSafe [34].

1.3 Objetivos

El objetivo de este trabajo consiste en estudiar varias de las técnicas recientes para la detección, seguimiento y segmentación multi-objeto entrenadas con el conjunto de datos KITTI MOTS, reproducir sus resultados obtenidos en KITTI MOTS Validation ejecutando su correspondiente código y realizar una comparativa entre ellas en base a unas métricas de evaluación. Como base se parte de analizar la técnica utilizada en [1] y se seleccionan varias técnicas que consiguen mayor puntuación que ésta en la evaluación de la tarea MOTS en "The KITTI Vision Benchmark Suite" [35] y que ofrecen su código abierto para

poder entrenarlo.

1.4 Estructura del documento

El documento presenta la siguiente estructura:

- El Capítulo 1 de introducción, recoge el resumen del TFM, define sus correspondientes objetivos y ofrece una visión global del trabajo.
- El Capítulo 2 revisa la base teórica o estado del arte, así como los conceptos básicos para la evaluación de la misma
- El Capítulo 3 aborda el desarrollo de los distintos códigos utilizados en este proyecto, ofreciendo resultados y conclusiones obtenidas por los autores.
- En el Capítulo 4 se presentan los resultados obtenidos para cada uno de los métodos analizados, después de llevar a cabo la implementación de cada uno de ellos en diferentes propuestas de configuración hardware y software. Por último, se realiza una comparativa entre todos ellos.
- Finalmente, en el Capítulo 5, se presentan las conclusiones y trabajos futuros.

Capítulo 2

Estado del Arte

2.1 Introducción

Este capítulo recoge los conocimientos teóricos adquiridos para la implementación e interpretación de los resultados de las distintas técnicas, que resultan importantes para llevar a cabo el resto del proyecto. Fundamentalmente, se revisarán las bases de la visión artificial y reconocimiento de objetos y el aprendizaje supervisado basado en redes neuronales, que representan el núcleo del trabajo desarrollado a lo largo de este trabajo.

2.2 La visión artificial como base para el reconocimiento y seguimiento de objetos

La visión artificial es una ciencia que incluye métodos para adquirir, procesar, analizar y comprender las imágenes del mundo real con el objetivo de producir información numérica o simbólica para que puedan ser tratados por un ordenador [36]. Trata de producir el mismo efecto que el uso de los ojos y cerebros humanos para que los ordenadores puedan percibir y comprender una imagen o secuencia de imágenes y actuar según convenga en una determinada situación.

El reconocimiento de objetos es una de las tecnologías que utilizan visión artificial. Se basa en detectar la presencia de objetos en una imagen sobre la base de su apariencia visual, bien sea atendiendo al tipo de objeto (coche, peatón) o a la instancia del objeto (mi coche, tu coche). Esto hace que se distingan dos partes en el proceso de detección: la extracción de descriptores de una imagen y la búsqueda de objetos basada en dichas características.

Los mayores retos tanto de la extracción de características como la clasificación es encontrar descriptores y clasificadores que sean invariantes a los cambios que pueda tener un objeto, como su posición o iluminación.

La visión artificial define cinco principales fases que se muestran en la Figura 2.1. En la primera fase, con ayuda de un sensor, se capturan imágenes digitales (muestreo, discretización y almacenamiento digital de la imagen). En la segunda fase se preprocesan las imágenes con el fin de facilitar las siguientes fases, aplicando filtros o transformaciones geométricas que eliminan zonas indeseables de la imagen o realzan zonas interesantes. La tercera fase consiste en dividir la escena en regiones de atributos similares (segmentación) para aislar los distintos elementos y extraer los que sean de interés. La siguiente fase

extrae las características del objeto para poder clasificarlo adecuadamente. La última fase es la de clasificación, donde se pretende distinguir los objetos segmentados en función de las características extraídas previamente.

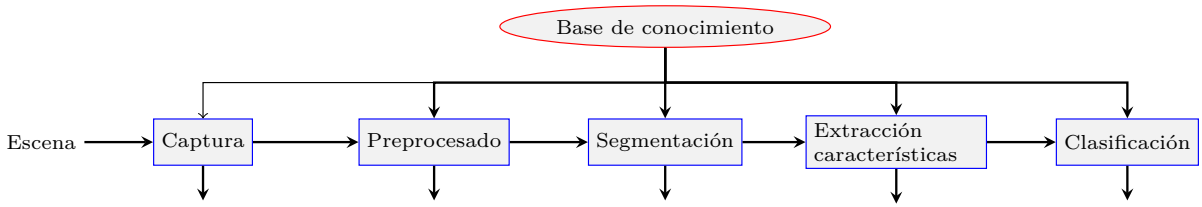


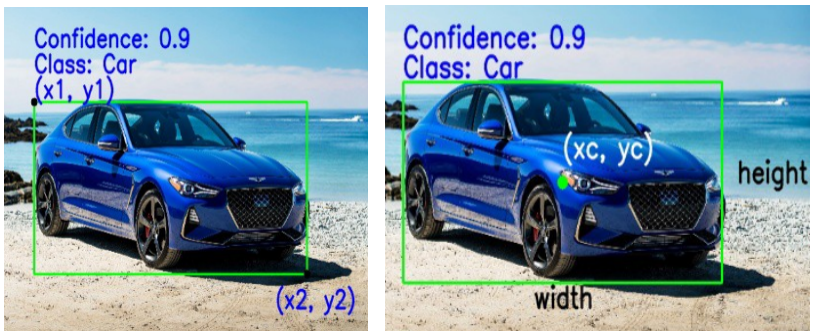
Figura 2.1: Procedimiento para el tratamiento digital de imágenes

En un escenario de seguimiento, la forma de un objeto se puede representar mediante [33]:

- Puntos. Es decir, se representa a través de un centroide o un conjunto de puntos. Esto resulta útil cuando los objetos ocupan una pequeña región de la imagen.
- Formas geométricas (rectángulos, elipses, etc). El movimiento de estas representaciones se realiza a través de modelar su traslación, afinidad u homografía. Se utiliza tanto para objetos rígidos simples como para objetos no rígidos.

  Un cuadro delimitador o *bounding box* es un rectángulo que rodea a un objeto y especifica su posición, su clase (un vehículo o un peatón) y su confianza (probabilidad de que se encuentre en ese lugar) [37]. Es utilizado principalmente para la detección de objetos en una imagen.

  Los dos modos habituales para especificar un *bounding box* se definen en la figura 2.2.



(a) *Bounding box* especificado con respecto a sus puntos superior izquierdo e inferior derecho

(b) *Bounding box* especificado con respecto a sus coordenadas centrales, anchura y altura

Figura 2.2: Modos de especificar un *Bounding Box*

En función del modo en el que se defina un *box*, existen distintos parámetros. *Class* indica el tipo de objeto que se encuentra dentro del *box*. En el ejemplo de la figura 2.2 es un coche. Las coordenadas marcan la esquina superior izquierda y la esquina inferior derecha en el caso 2.2a (x1, y1), y centro del rectángulo en el caso de 2.2b (xc, yc). *Width* es el ancho del *Bounding Box*. *Height* es el alto del *Bounding Box*. *Confidence* es la probabilidad de que el objeto esté realmente en ese *box*. En el ejemplo, indica que hay un 90 % de posibilidades de que el coche exista realmente en ese *box*.

Los *Bounding Boxes* son una de las técnicas de anotación de imágenes más populares en el aprendizaje profundo [38] y además, en comparación con otros métodos de procesamiento de imágenes, reduce los costes y aumenta la eficiencia de la anotación. Los vehículos autónomos dependen en gran medida de los *Bounding Boxes* para hacerlo posible.

- Siluetas y contornos. El contorno representa el límite del objeto y la región que queda dentro es la silueta. Es adecuado para el seguimiento de formas complejas no rígidas.
- Modelos articulados de forma. Se pueden representar las distintas "articulaciones" mediante cilindros o elipses. Un objeto articulado puede ser el cuerpo humano.
- Modelos esqueléticos. El esqueleto de un objeto se puede extraer a través de la transformación del eje medio de la silueta del objeto. Se puede utilizar para modelar objetos articulados y rígidos.

En la Figura 2.3 se representa el objeto con los distintos métodos comentados.

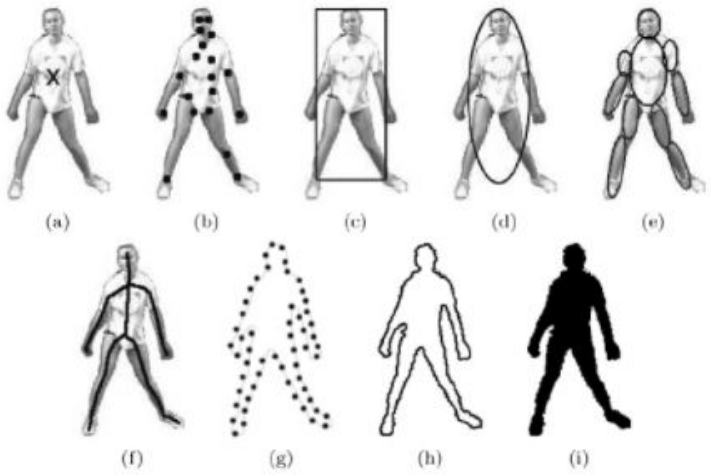


Figura 2.3: Formas de representar un objeto. (a) Centroide, (b) Múltiples puntos, (c) Rectángulo, (d) Elipse, (e) Modelo articulado de forma, (f) Modelo esquelético, (g) Puntos de control en el contorno del objeto, (h) Contorno, (i) Silueta [2]

A su vez, también existen distintas formas de representar las características de aspecto del objeto. Las formas más comunes son las siguientes:

- Densidad de probabilidad (color, textura) del aspecto de los objetos. Estas pueden ser paramétricas o no paramétricas, y se pueden obtener a partir de las regiones de la imagen especificada por los modelos de forma (por ejemplo, la región interior de una elipse o un contorno). En la Figura 2.4 se muestra un ejemplo de representación.

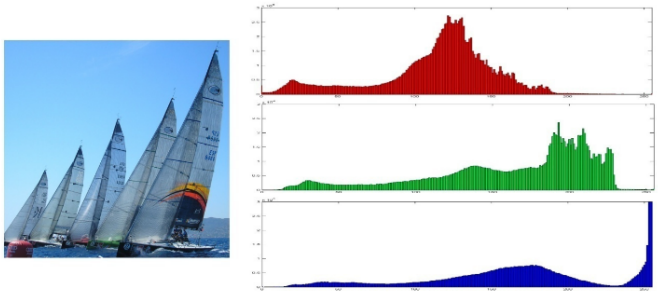


Figura 2.4: Histograma de color RGB de una imagen. Gran parte de los píxeles son azul intenso (valores cercanos a 255)

- Plantillas. Están formadas con formas geométricas simples o siluetas. Su ventaja es que aporta tanto la información espacial como la de aspecto. Su desventaja es que solamente codifican el aspecto de

los objetos generados a partir de una única vista. Por tanto, solo son adecuadas para seguir objetos donde las posiciones no varíen considerablemente al largo del seguimiento.

- Modelos activos de aspecto. Generados mediante el modelado simultáneo de la forma del objeto, definido por un conjunto de puntos de referencia, y su aspecto. Por cada punto de referencia se guarda un vector de aspecto en forma de color, textura o magnitud del gradiente. Es necesario una fase de entrenamiento de estos modelos donde tanto la forma como su aspecto asociados se conocen a partir de un conjunto de muestras. En la Figura 2.5 se muestra un ejemplo de representación.

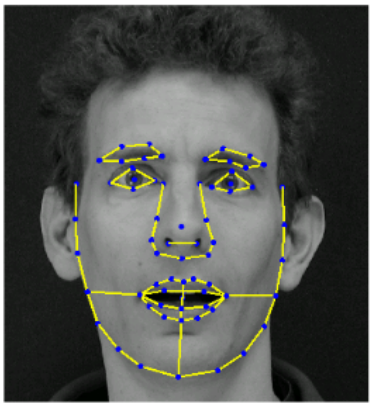


Figura 2.5: Modelo de apariencia activa

- Modelos de aspecto multivista. Codifican diferentes puntos de vista de un objeto, generando un subespacio de las proyecciones que se han dado. En la Figura 2.6 se muestra un ejemplo de representación.

Figura 2.6: Modelo de apariencia multivista

En el seguimiento de objetos, la selección de características juega un papel fundamental. La característica visual más deseada es la singularidad, que hace que al objeto se le pueda distinguir fácilmente en el espacio de características [33].

Las características más comunes son el color, los márgenes, el flujo óptico y la textura. Para representar el color del objeto, el procesamiento de imágenes debe utilizar el espacio de color RGB. Esta característica puede verse afectado por la distribución de energía espectral de la fuente o por las propiedades de reflectancia de la superficie del objeto. Mediante la detección de márgenes se pueden identificar los fuertes cambios de intensidad de la imagen que se pueden generar en los límites de los objetos. Frente al color, son menos sensibles a los cambios de iluminación. El flujo óptico es un campo denso de desplazamiento de vectores que define la translación de cada píxel en una región. Se calcula mediante la restricción de brillantez constante y se utiliza generalmente como característica en el seguimiento de la segmentación basada en movimiento, así como en aplicaciones de seguimiento. La textura es una medida de la variación

de intensidad de una superficie que cuantifica las propiedades como por ejemplo la suavidad y la regularidad. Frente al color, la textura requiere una etapa de procesamiento para generar los descriptores, y es menos sensible a los cambios de iluminación. Los descriptores de características utilizan una serie de aproximaciones matemáticas para aprender una representación de la imagen que no varía en escala.

Cada método de seguimiento requiere un mecanismo de detección de objetos. Se pueden detectar objetos utilizando la información de un solo fotograma o haciendo uso de la información temporal calculada a partir de una secuencia de imágenes, reduciendo así el número de falsas detecciones [33]. Generalmente, la información temporal es obtenida a partir de la técnica *frame differencing*, que comprueba la diferencia entre dos fotogramas de vídeo. Si los píxeles han cambiado, aparentemente ha habido algún cambio en la imagen (por ejemplo, movimiento).

La mayoría de las técnicas funcionan con un cierto desenfoque y umbral, para distinguir el movimiento real del ruido. Los fotogramas también pueden variar cuando cambian las condiciones de luz en una habitación (y el enfoque automático de la cámara, la corrección del brillo, etc.).

Una vez se han establecido las regiones del objeto dentro de la imagen, comienza la tarea de seguimiento donde hay que realizar la correspondencia del objeto de un fotograma a otro. Los métodos más populares para ello son los detectores de puntos, la sustracción del fondo o la segmentación. El caso más simple se daría cuando el objeto y su entorno sólo han sido trasladados de forma alineada al plano de la cámara y no han sufrido variaciones de apariencia ni se han movido a alta velocidad. Sin embargo, la realidad es que el objeto puede sufrir distintas variaciones como pueden ser un cambio de posición, de escala o de color; desaparecer unos instantes de la escena y volver a entrar; o quedar total o parcialmente ocluido hasta llegar a un punto que su apariencia ha cambiado completamente con respecto al primer cuadro. Algunos ejemplo de esto se muestran en la Figura 2.7.

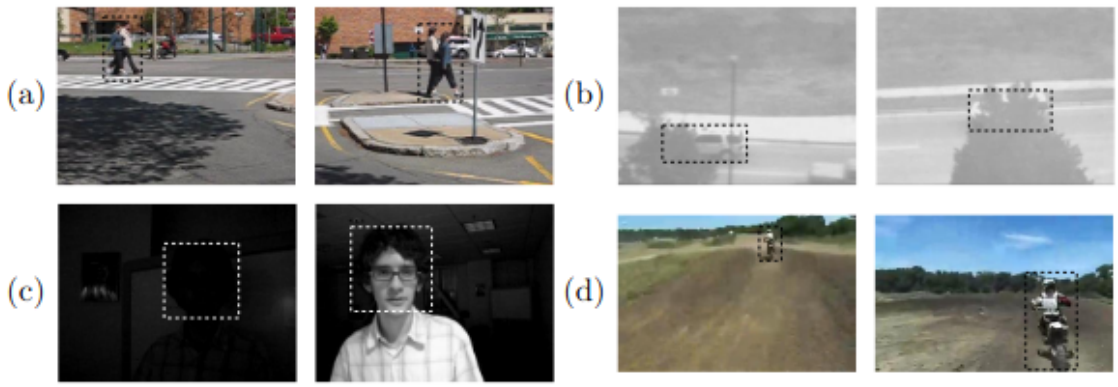


Figura 2.7: Problemas típicos a tener en cuenta en el seguimiento de objetos: (a) Cambios de pose, (b) oclusiones parciales y totales, (c) cambios de iluminación y (d) cambios de escala (https://rdu.unc.edu.ar/bitstream/handle/11086/2829/TF17327.pdf;sequence=1)

Los detectores de puntos localizan los puntos de interés en imágenes con una textura expresiva en sus respectivas localidades. Una característica deseable en cuanto a los puntos de interés es su invariación en los cambios de iluminación y en el punto de vista de la cámara.

La sustracción del fondo consiste en la construcción de una representación de la escena, o modelo de fondo, para después localizar las desviaciones del modelo para cada fotograma entrante. Un objeto en movimiento se representa cuando se da cualquier cambio significativo en una región de la imagen del modelo de fondo.

La segmentación consiste en dividir una imagen en varias regiones o grupos de píxeles denominadas segmentos [39]. Más concretamente, la segmentación es un proceso de clasificación por píxel que asigna

una categoría a cada píxel de la imagen analizada. Su objetivo es localizar regiones con significado. Se utiliza tanto para localizar objetos como para encontrar sus bordes dentro de una imagen. El resultado de la segmentación de una imagen es un conjunto de segmentos que cubren toda la imagen sin superponerse. Se puede representar como una imagen de etiquetas (una etiqueta para cada píxel) o como un conjunto de contornos. Las técnicas de segmentación pueden utilizar información de movimiento (basadas en movimiento) y/o complementar el movimiento mediante el uso de información especial (basadas en características espacio temporales).

Las técnicas basadas en movimiento pueden ser tanto en 2D, que son simples pero menos realistas, como en 3D, que son las más utilizadas en la práctica. Los métodos 2D pueden utilizar las discontinuidades del flujo óptico, donde el desplazamiento de un píxel es un vector de movimiento que representa el movimiento entre el píxel en una imagen y el píxel correspondiente en la imagen posterior; o la detección de cambios (detección de los píxeles del objeto y de los píxeles del fondo, asumiendo que el fondo es normalmente estacionario). Como métodos 3D, existen los Structure from motion (SFM), que generalmente manejan escenas 3D con información relevante de profundidad y se asume un movimiento rígido, y los algoritmos paramétricos, que frente a SFM, no asumen esa profundidad y sólo asumen rigidez de movimiento en partes de la escena.

Las técnicas basadas en características espacio temporales a su vez se pueden basar en límites (extracción de márgenes muy prominentes para segmentar los objetos de interés) o en regiones (se centra en un conjunto de regiones generalmente definidas con características espaciales o espacio temporales).

Uno de los casos más sofisticados de la segmentación es la segmentación semántica que clasifica objetos diversos. Mask Regions with CNN features (R-CNN) es un segmentador semántico que clasifica 90 categorías del conjunto de datos Microsoft Common Objects in COntext (MS COCO) [25], reconociendo personas, vehículos, vegetación, calle, vereda, edificios, y demás elementos típicos en escenas urbanas.

Las técnicas de seguimiento tienen como objetivo generar la trayectoria de un objeto a través del tiempo, posicionándolo dentro de la imagen. Estas técnicas se pueden clasificar en tres grandes grupos mostrados en la Figura 2.8.

En las técnicas de seguimiento de puntos, la asociación de los puntos que representan los objetos detectados en imágenes consecutivas está basada en el estado del objeto en la imagen anterior, donde se puede incluir posición y movimiento. En un escenario donde existan oclusiones y entradas y salidas de los objetos puede presentar problemas. Estas técnicas pueden ser determinísticas, donde una predicción futura del comportamiento del objeto a partir del objeto anterior determina el coste de correspondencia. Este coste viene definido por una serie de restricciones: proximidad, velocidad máxima, cambios de velocidad pequeños, movimiento común, rigidez y uniformidad por proximidad. También pueden ser técnicas probabilísticas, que consideran las observaciones y las incertidumbres del modelo para la valoración del estado del objeto que se está siguiendo, entendiendo como observación la posición del objeto dentro de la imagen. Modelan propiedades como la posición, velocidad y aceleración del objeto dentro de la imagen utilizando el espacio de estados. Algunos de los métodos utilizados son los filtros de Kalman, filtros de partículas, filtros para la probabilidad conjunta de los datos y seguimiento de múltiples hipótesis.

Un filtro recursivo es una solución conveniente cuando se requiere de una estimación cada vez que se recibe una nueva medida. La aproximación mediante filtros recursivos permite procesar datos secuencialmente en lugar de en bloque, por lo que no es necesario almacenar todo el conjunto de datos o reprocesar datos existentes cuando una nueva medida está disponible. Un filtro recursivo consta de dos etapas: la primera de predicción y la segunda de actualización. La etapa de predicción predice el estado de la f.d.p. de un instante al siguiente utilizando el modelo de movimiento. Dado que el estado normalmente está sujeto a perturbaciones desconocidas (modeladas como ruido Gaussiano), la predicción generalmente traslada,

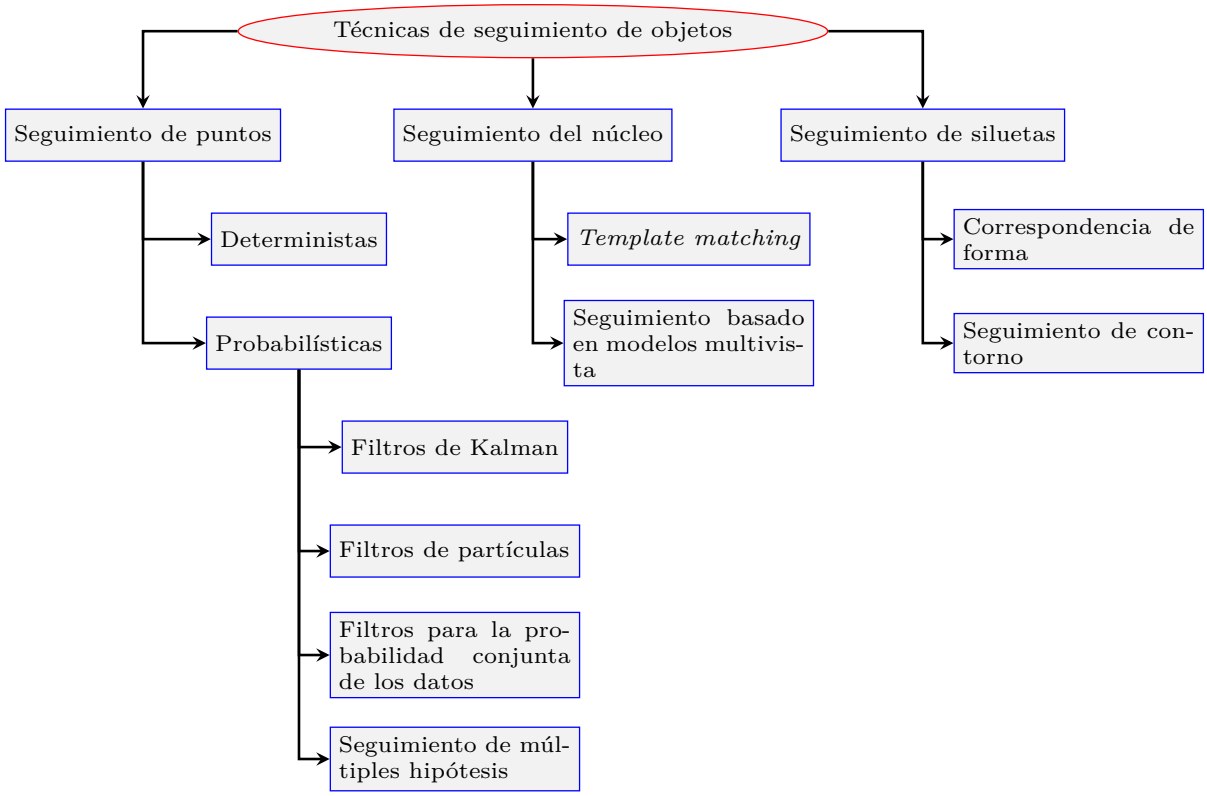


Figura 2.8: Técnicas de seguimiento de objetos

deforma y expande la f.d.p.. La etapa de actualización modifica la predicción de la f.d.p. utilizando la última medida.

El filtro de Kalman provee una solución recursiva óptima al problema de filtrado lineal de datos discretos, mediante el método de mínimos cuadrados. El objetivo de este filtro es calcular un estimador lineal, insesgado y óptimo del estado de un sistema en t con base en la información disponible en $t-1$, y actualizar con la información adicional disponible en t dichas estimaciones. Se presupone que el sistema puede ser descrito a través de un modelo estocástico lineal donde el estado tiene una distribución Gaussiana. Este filtro es interesante por su habilidad para predecir el estado de un sistema en el pasado, presente y futuro, aún cuando la naturaleza precisa del sistema modelado es desconocida.

El filtro de partículas es un método que se emplea para estimar el estado de un sistema que cambia a lo largo del tiempo. Este filtro se compone de un conjunto de muestras (partículas) y unos valores (pesos) asociados a cada una de esas muestras. Las partículas representan muestras del espacio de estado (estados posibles) del proceso, y los pesos representan muestras de la f.d.p. a posteriori del estado, dadas las observaciones. En la primera fase de inicialización, el filtro "lanza" al azar un conjunto de puntos (para el seguimiento de un objeto en una secuencia de imágenes, se "lanzan" sobre el plano de la imagen) y crea un estado aleatorio donde opcionalmente se puede dar algún tipo de información a priori como el tamaño del objeto o la posición aproximada. En la fase de actualización en función de la similitud del estado de cada partícula respecto al estado de referencia se le asignará un peso a cada uno de ellas. En la tercera fase de estimación se crea un nuevo conjunto de partículas que constituye la estimación a priori del estado en el siguiente instante de tiempo empleando métodos de remuestreo probabilísticos que den lugar a nuevas partículas con mayor probabilidad. La última fase de predicción realiza una ligera modificación del estado de cada uno de ellos introduciendo ruido aditivo aportando variabilidad al sistema con el fin de estimar el estado del objeto en el instante siguiente. Al terminar esta etapa se obtiene un nuevo conjunto de partículas al que se le vuelve a aplicar las mismas etapas y se repite esto continuamente hasta terminar

la secuencia de datos.

El filtro de Kalman tiene la limitación de que asume que las variables de estado tienen una distribución Gaussiana. Es por ello que sus estimaciones son pobres para variables de estado que no sigan esa distribución. Sin embargo esa limitación se solventa con el filtro de partículas.

El filtrado de Kalman entró en el salón de la fama cuando se utilizó en el Apollo PGNCS para producir una estimación de posición óptima para la nave espacial, basada en mediciones de posición pasadas y nuevos datos

Las técnicas de seguimiento del núcleo (kernel) realizan un cálculo del movimiento del objeto, el cual está representado por una región inicial, de una imagen a la siguiente. El seguimiento puede realizarse utilizando plantillas y modelos de apariencia basados en densidad de probabilidad (*template matching*)o basándose en modelos multivista. Estos últimos se utilizan cuando el aspecto del objeto cambia drásticamente y como consecuencia se pierde el seguimiento de este objeto.

Las técnicas de seguimiento de siluetas se realizan mediante la valoración de la región del objeto en cada imagen utilizando la información que contiene. Esta información puede ser en forma de densidad de aspecto o de modelos de forma, generalmente presentados con mapas de márgenes. Uno de los métodos de estas técnicas es la correspondencia de forma, donde se busca la silueta del objeto y su modelo asociado dentro de la imagen actual. Otro de los métodos es el seguimiento del contorno, donde evolucionan un contorno inicial en un fotograma anterior a la nueva posición en el fotograma actual.

2.3 Aprendizaje supervisado, redes neuronales

La detección de objetos en una imagen se puede resolver a través de los métodos de aprendizaje supervisado. Estos consisten en un aprendizaje automático de distintas vistas del objeto obtenidas de un conjunto de ejemplos. Por lo tanto, a través de un conjunto de ejemplos de aprendizaje (diferentes clases de objetos y sus características asociadas), estos métodos generan una función de mapeado o relación de las entradas con las salidas deseadas.

Para facilitar el trabajo del algoritmo, es importante elegir un conjunto de descriptores que permitan discriminar entre una clase y otra. Si se quiere conseguir que la red neuronal sea capaz de generalizar e identificar un tipo de objeto determinado en cualquier imagen, es importante utilizar un elevado número de imágenes para realizar el entrenamiento, tanto de imágenes que son ese objeto (etiquetadas como 1) como de imágenes que no lo son (etiquetadas como 0), incluyendo la mayor variabilidad posible. Con esto, la red es capaz de ajustar sus parámetros para satisfacer en la medida de lo posible todas las imágenes, y por lo tanto es capaz de extraer de manera precisa las características que identifican la presencia de un objeto determinado en una imagen.

La detección de caras humanas es un ejemplo clásico de aplicación de estos algoritmos en detección de objetos. En este caso, el conjunto de aprendizaje está compuesto de caras humanas de todo tipo y de objetos y texturas que no representan caras. De esta forma el algoritmo puede establecer un criterio adecuado sobre lo que es una cara y lo que no.

El principal inconveniente de estos métodos es la extensa colección de muestras de cada clase de objeto que necesitan para el aprendizaje. Además, esta colección debe ser etiquetada manualmente.

Las técnicas de aprendizaje de patrones más utilizadas son Support Vector Machine (SVM), mejora adaptativa (*Adaboost*) o redes neuronales.

La red neuronal se define como un algoritmo que simula el comportamiento del cerebro a través de modelos matemáticos. Se compone de miles de neuronas artificiales interconectadas que se almacenan en

filas, denominadas capas, formando miles de conexiones [40]. Una capa es un conjunto de neuronas cuyas entradas provienen de una capa anterior (o de los datos de entrada en el caso de la primera capa) y cuyas salidas son la entrada de una capa posterior. La salida de la capa viene dada por tres funciones:

- Una función de propagación o excitación que es el sumatorio de cada entrada multiplicada por el peso de su interconexión (valor neto). La conexión se denomina excitatoria si el peso resulta positivo, e inhibitoria si resulta negativo.
- Una función de activación que modifica a la anterior. Puede no existir (véase sección 2.3.1.
- Una función de transferencia, que se aplica al valor devuelto por la función de activación. Es utilizada para acotar la salida de la neurona y generalmente viene dada por la interpretación que se quiera dar a las salidas. Algunas de las más utilizadas son la función sigmoidea (obtener valores en el intervalo [0,1]) y la tangente hiperbólica (obtener valores en el intervalo [-1,1]).

A nivel esquemático, una neurona artificial se representa del siguiente modo [3]:

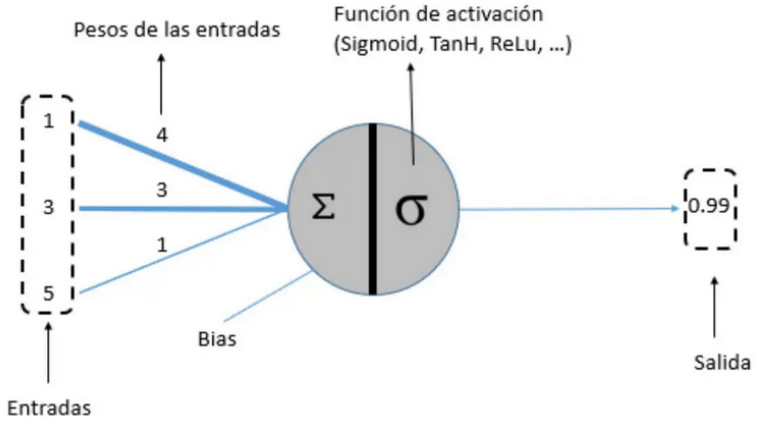


Figura 2.9: Representción de una red neuronal [3]

Las redes neuronales obtienen resultados con una alta precisión gracias a ser modelos de vanguardia que capturan características complejas de una forma óptima y efectiva. Además, todas las unidades individuales de procesamiento proporcionan una respuesta al mismo tiempo por lo que todas las neuronas de una capa trabajan en forma paralela. Es necesario un aprendizaje en la red antes de emplearla y los pesos de cada neurona son ajustados basándose en la experiencia. Si parte de la red no trabaja, solo deja de funcionar la parte para que dicha neurona sea significativa, el resto sigue su comportamiento normal. Las neuronas pueden reconocer patrones que no han sido aprendidos, sólo deben tener cierto parecido con el conocimiento previo que tenga la red.

Sin embargo, las redes neuronales suelen necesitar mayor volumen de datos para el entrenamiento del modelo y requieren de alta capacidad de recursos computacionales. Además existe complejidad de aprendizaje para grandes tareas ya que cuanto más se necesita que aprenda una red, más complicado resulta enseñarle. El tiempo de aprendizaje es elevado y esto depende de dos factores: primero si se incrementa la cantidad de patrones a identificar o clasificar, y segundo, si se requiere mayor flexibilidad o capacidad de adaptación de la red neuronal para reconocer patrones que sean sumamente parecidos. Las redes no son fácilmente explicables, conocer las reglas o motivos por los que la red devuelve esos resultados no suele ser fácil y precisa de otras analíticas.

Las redes neuronales se pueden clasificar según el número de capas (monocapa o multicapa), o según su tipo de conexión (convolucionales, recurrentes, de retroalimentación o de base radial). Un ejemplo del primer caso es el de la Figura 2.10: una red monocapa y una red multicapa formada por cuatro capas.

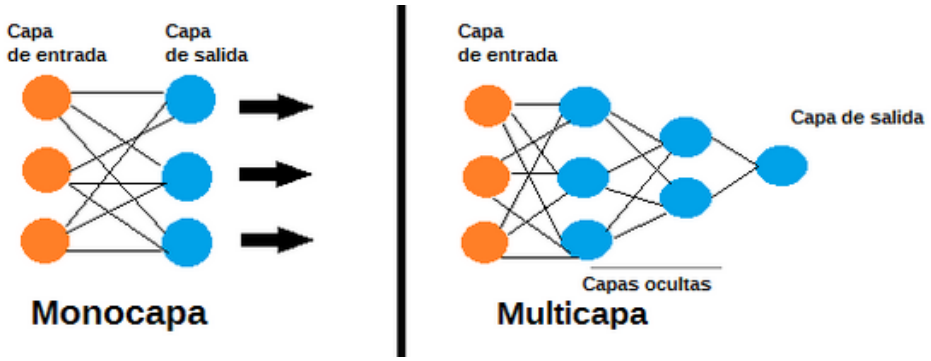


Figura 2.10: Clasificación de redes neuronales según el número de capas. En la red multicapa las características aprendidas en la primera capa puede ser la aparición o no de ejes en una parte concreta de la imagen. La segunda capa detecta uniones de ejes. La tercera capa aprende combinaciones que correspondería a partes de objetos. Así sucesivamente [4].

Las neuronas de la primera capa (capa de entrada) reciben como entrada los datos reales que alimentan a la red neuronal. La salida de la última capa (capa de salida) es el resultado visible de la red. Las capas que se sitúan entre la capa de entrada y la capa de salida se conocen como capas ocultas ya que se desconocen tanto los valores de entrada como los de salida. El concepto de *Deep Learning*, o aprendizaje profundo, nace a raíz de utilizar un gran número de capas ocultas en las redes.

Una de las modelizaciones de redes neuronales empleadas en las arquitecturas *Deep Learning* es la CNN o red neuronal convolucional. Este tipo de arquitectura resulta muy efectiva en la detección y categorización de objetos, y en la clasificación y segmentación de imágenes. Las neuronas de sus redes corresponden a campos receptivos.

La convolución consiste en filtrar una imagen utilizando una máscara. Diferentes máscaras producen distintos resultados. Las máscaras representan las conexiones entre neuronas de capas anteriores. El objetivo de CNN es aprender características de orden superior utilizando la operación de convolución. Cada píxel de salida es una combinación lineal de los píxeles de entrada. Estas capas aprenden progresivamente las características de orden superior de la entrada sin procesar. El proceso de una CNN se muestra en la Figura 2.11.

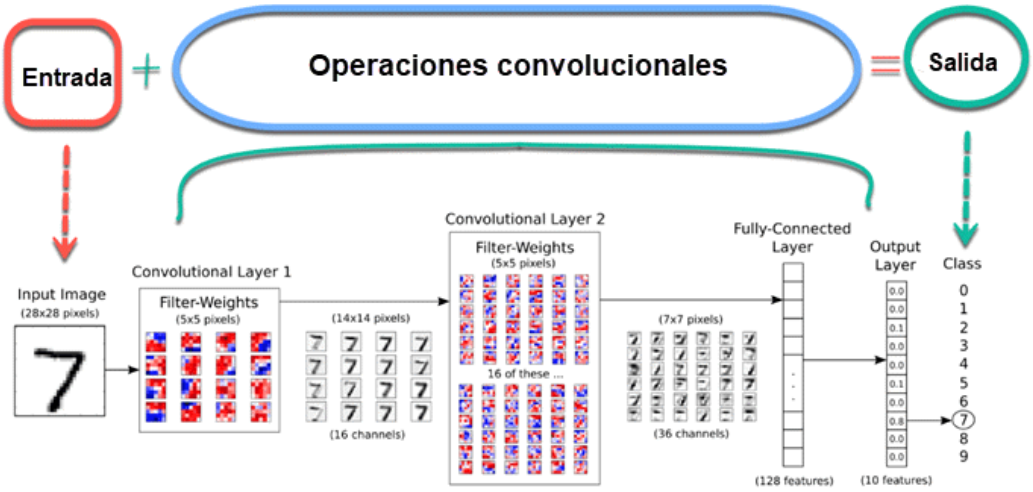


Figura 2.11: Proceso de una CNN. En primer lugar, la imagen de entrada es empujada a la red. A continuación, la imagen de entrada pasa por un número infinito de pasos; esta es la parte convolucional de la red. Finalmente, la red neuronal puede predecir el dígito en la imagen.[5].

La principal diferencia de la CNN con la red de percepción multicapa viene en que cada neurona no se une con todas y cada una de las capas siguientes, sino que solo con un subgrupo de ellas (se especializa), con esto se consigue reducir el número de neuronas necesarias y la complejidad computacional necesaria para su ejecución [41].

Una Region Proposal Network (RPN) toma una imagen como entrada y devuelve las propuestas de regiones rectangulares, donde cada propuesta tiene una puntuación de objetividad (grado de confianza en que la propuesta contiene un objeto) [9]. Esta red predice los límites de las propuestas de regiones desde algunas *boxes* de referencia predefinidas y de tamaños fijos denominadas *Anchor Boxes*. Las *Anchor Boxes* se extienden por toda la imagen y al venir en diferentes tamaños y diferentes relaciones de aspecto ayudan a capturar diferentes tipos de objetos.

La RPN trabaja sobre el mapa de características resultante de la CNN y define las *anchors* en el mapa de características, pero las *Anchor Boxes* finales se crean con respecto a la imagen original. Cada característica de este mapa se denomina *Anchor Point*. Para cada punto se colocan *Anchor Boxes* con diferentes tamaños y proporciones sobre la imagen centradas en dicho punto. La red toma todas las *Anchor Boxes* como entrada y, a continuación, genera la puntuación de objetividad para cada *Box* y realiza una regresión para encontrar una *Boundary Box* más precisa.

La RPN puede implementarse como una red de convolución completa. En primer lugar, realiza una convolución de 3×3 en el mapa de características ($H \times W$) utilizando 512 unidades, y para cada ubicación (ventana), genera 9 *Anchor Boxes* en la imagen. Esta convolución devuelve un mapa de características de 512 -d para cada ubicación. La salida de la convolución anterior se introduce en 2 capas de convolución paralelas de 1×1, una para la clasificación y otra para la regresión de las *Anchor Boxes*. La capa de clasificación devuelve las probabilidades de estar en primer plano (objeto presente) y en segundo plano para cada *Anchor Boxes*. Para cada ubicación del mapa de características, el tamaño de la salida de esta capa sería de $2k$. La capa de regresión devuelve los 4 desplazamientos predichos para la *box* límite, y el tamaño de la salida sería de $4k$.

Dentro de las 9 *Anchor Boxes* para cada ubicación del mapa de características, no todas son relevantes. Por ello, a cada *box* se le asigna una etiqueta, que se basa en la Intersection-over-Union (IoU) sobre el *Ground Truth*. Se definen 3 etiquetas:

- Etiqueta = 1 (Primer plano): Un *anchor* puede tener la etiqueta 1 si el *anchor* tiene el mayor IoU con el *Ground Truth* o si el IoU con la *Ground Truth* es mayor que 0,7. (IoU >0,7).
- Etiqueta = -1 (Fondo): Un *anchor* se asigna con -1 si IoU < 0,3.
- Etiqueta = 0: Si no entra en ninguna de las condiciones anteriores, este tipo de *anchors* no contribuyen al entrenamiento, se ignoran.

Durante la fase de prueba, la RPN crea múltiples propuestas de región para un objeto, pero no todas son las mejores, por lo que para encontrar la mejor propuesta y eliminar todas las propuestas redundantes utiliza un método llamado Supresión No Máxima. En la Supresión No Máxima, se ordenan todas las propuestas de regiones según su puntuación de objetividad (de alta a baja) y se itera a través de todas las propuestas. Se selecciona la propuesta con la puntuación de objetividad más alta y se eliminan todas las propuestas que tienen IoU > 0,5 con la propuesta elegida actualmente. De esta manera se eliminan todas las propuestas desfavorables para un objeto y se mantiene la mejor. Las propuestas eliminadas no se procesarán en la siguiente iteración. Se repite el paso anterior para todas las propuestas.

Otro tipo de redes son las Recurrent Neural Network (RNN). Estas no tienen una estructura de capas, sino que permiten conexiones arbitrarias entre las neuronas, incluso pudiendo crear ciclos. Con esto se

consigue crear la temporalidad, permitiendo que la red tenga memoria [41]. Los datos introducidos en el momento t en la entrada, son transformados y van circulando por la red incluso en los instantes de tiempo siguientes $t+1$, $t+2$, ...

Las Long Short-Term Memory (LSTM) son un tipo especial de redes recurrentes [6]. Las redes recurrentes introducen bucles en el diagrama de la red provocando que la información pueda persistir. Una de sus limitaciones es que pueden modelar dependencias a corto plazo, mientras que las LSTM tienen memoria a más largo plazo. En la Figura 2.12 se puede observar esquemáticamente esta diferencia.

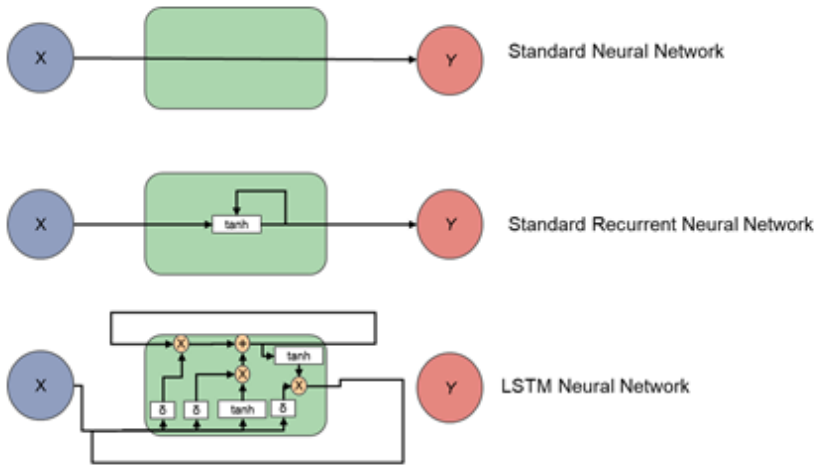


Figura 2.12: Tipos de esquemas de redes neuronales. Una salida Y se calcula a partir de una entrada X. En A), Y depende únicamente de X. En B), Y depende de X y del valor anterior de Y. En C), la lógica es más compleja para mejorar el efecto de instancias anteriores de Y (y, por consiguiente, de X) [6].

2.3.1 Función de activación ReLU

Una función de activación se encarga de devolver una salida (0,1) a partir de un valor de entrada ReLU. Para minimizar el coste computacional, la derivada de la función debe ser simple. La función ReLU se muestra en la ecuación 2.1, donde anula los valores negativos de entrada y deja tal cual entran los positivos.

$$ReLU(x) = max(0, x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \tag{2.1}$$

Las características de la función ReLU son las siguientes: sólo se activa si a la entrada hay valores positivos (Activación *Sparse*), no está acotada, se pueden "morir" demasiadas neuronas , se comporta bien con imágenes, tiene buen desempeño en redes convolucionales y tiene rapidez computacional.

En las CNN cuando se procesa una imagen, cada capa de convolución debe capturar algún patrón en la imagen y pasarla en la siguiente capa de convolución. La función ReLU se utiliza en este tipo de redes ya que los valores negativos no son importantes en el procesamiento de imágenes y, por lo tanto, se establecen en 0, y los valores positivos después de la convolución deben pasar a la siguiente capa. Si se utiliza la función sigmoidea o tangente hiperbólica, la información se pierde ya que ambas funciones modificarán las entradas a un rango muy cerrado.

2.4 Detección y seguimiento de objetos en una imagen

La visión artificial y el aprendizaje profundo han realizado grandes avances en los últimos años. Sus técnicas ahora demuestran un rendimiento impresionante en la detección de objetos, así como en la imagen y en la segmentación de instancias. Además, han hecho que hoy en día la clasificación de objetos dentro de una imagen esté bastante resuelta [42]. Los modelos que se encuentran disponibles públicamente han sido entrenados en grandes cantidades de datos.

La propuesta de Seguimiento y Segmentación Multi-Objeto, llevada a cabo en el artículo de referencia, conocida como MOTS, demuestra que, con conjuntos de datos más potentes, la capacidad de los procesos de segmentación y seguimiento funcionan mejor que los actuales métodos de seguimiento con *Bounding Boxes*. Para ello, MOTS requiere de la combinación de claves temporales y de máscaras.

Casi todos los enfoques de seguimiento multi-objeto basados en la visión representan los objetos rastreados utilizando *Bounding Boxes* en el dominio de la imagen. En el ámbito del seguimiento de un solo objeto basado en la imagen, recientemente se ha producido un cambio hacia representaciones de máscara (con precisión de píxel). Los enfoques de seguimiento que utilizan datos RGB, estereoscópicos o de profundidad (como Laser Imaging Detection and Ranging (LIDAR)) suelen representar los objetos con centroides o *Bounding Boxes* 3D. Entre las representaciones más precisas se encuentran las nubes de puntos o las representaciones tridimensionales fijas que permiten integrar las mediciones 3D a lo largo del tiempo. Sin embargo, estas representaciones tienen el coste de un tiempo de procesamiento y un consumo de memoria adicionales, que son las principales preocupaciones para los rastreadores multiobjeto que mantienen un gran espacio de hipótesis. En este caso, los objetos se representan rastreados mediante sus posiciones 3D en el espacio-mundo y sus máscaras de píxeles en el dominio de la imagen. Las máscaras se almacenan de forma eficiente con la codificación Run-Length Encoding (RLE), que suele reducir el tamaño de almacenamiento a O(✓n), donde n corresponde al número de píxeles/puntos que representan la máscara.

Los primeros métodos que existieron para la detección de objetos aparecieron a finales de los años 90. Estos métodos utilizan la detección de características clásica (Scale-invariant feature transform (SIFT) [43] y Speeded-Up Robust Features (SURF) [44] como descriptores de características y Funciones de Accelerated Segment Test (FAST) [45] para detectar esquinas), en combinación con un algoritmo de aprendizaje automático como K-Nearest-Neighbor (KNN) [46] o SVM [47] para la clasificación, o con un comparador de descriptores como FLANN [48] para la detección de objetos.

En cuanto al seguimiento de objetos, existen métodos que todavía perduran como el filtro de Kalman y el flujo óptico. Todavía en muchos algoritmos se puede ver la influencia del filtro de Kalman, como por ejemplo en Simple Online and Realtime Tracking (SORT) [49], que utiliza una combinación del algoritmo húngaro y el filtro Kalman para lograr un seguimiento de objetos decente.

2.4.1 Detección de objetos

En el año 2014 aparece el clasificador R-CNN [7], un sistema compuesto por tres módulos. El módulo principal emplea un algoritmo de segmentación llamado búsqueda selectiva que extrae en torno a 2000 propuestas de regiones para seleccionar qué partes de una imagen tienen más probabilidades de contener un objeto. El algoritmo escanea la imagen con ventanas de varias escalas y busca píxeles adyacentes que comparten colores y texturas, al mismo tiempo que tiene en cuenta las condiciones de iluminación. Otro módulo es una CNN que extrae de la búsqueda selectiva un vector de características de longitud fija de cada propuesta. El módulo final clasifica cada región con SVM lineales específicas de categoría. En la Figura 2.13 se representa este método.

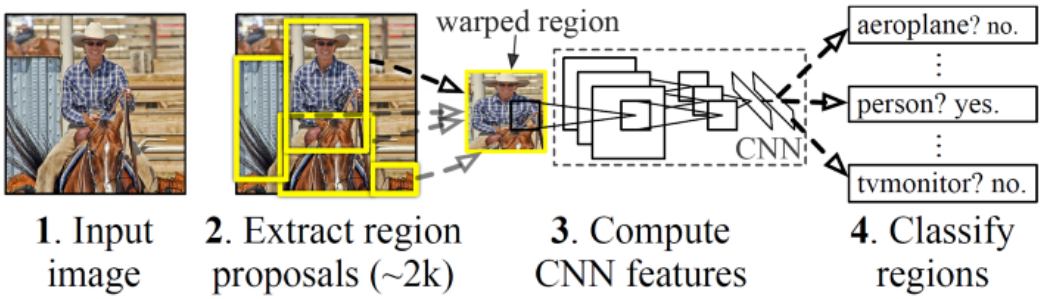


Figura 2.13: Arquitectura R-CNN [7]

R-CNN en comparación con los métodos actuales, es muy lento de entrenar y poco preciso. Su sucesor fue Fast R-CNN [8]. A diferencia del anterior, este método es más rápido ya que calcula un mapa de características convolucional para toda la imagen de entrada en tan solo un pase de la red. También resulta un entrenamiento más simple debido a que se entrena de un extremo a otro con una pérdida de múltiples tareas.

Está constituido por dos etapas. En la primera se obtiene el mapa de características convolucional pasando una imagen con un conjunto de propuestas de objetos a través de una red completamente convolucional. A continuación, se extrae un vector de características de longitud fija para cada propuesta del mapa utilizando una Region of interest pooling (RoI) mediante capas completamente conectadas para generar la probabilidad *softmax* (la clase) y el *Bounding Box* (la posición del objeto). La Figura 2.14 representa este método.

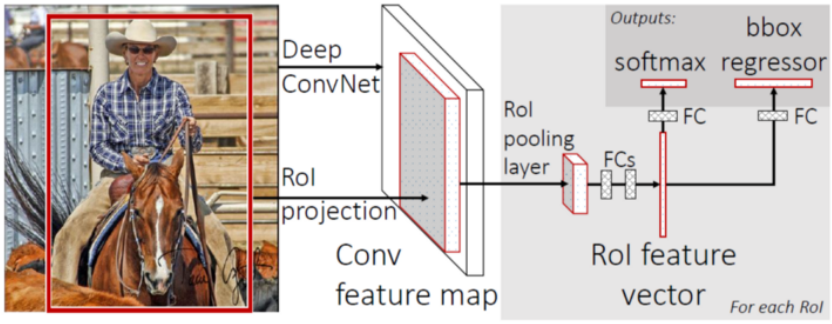


Figura 2.14: Arquitectura Fast R-CNN [8]

Este método sigue siendo bastante lento debido a que la CNN está bloqueada por la búsqueda selectiva. De este modo surge Faster R-CNN, aportando un enfoque de aprendizaje profundo [9]. Consta de dos módulos que se fusionan en una sola red y se capacitan de extremo a extremo: una CNN denominada RPN y el detector Fast R-CNN.

Faster R-CNN consta de dos etapas. La primera etapa es la RPN que propone *Bounding Box* de objetos candidatos. La segunda etapa, extrae características utilizando RoIPool de cada *Box* candidata y realiza la clasificación y la regresión de los *Boxes*. Las características utilizadas por ambas etapas pueden ser compartidas para una inferencia más rápida. Por defecto, utiliza tres escalas y tres relaciones de aspecto, lo que da como resultado nueve *anchors* en cada ventana deslizante. Este método se muestra en la Figura 2.15.

Este método es una de las mejores opciones para la detección de objetos. Su principal inconveniente es que no es capaz de localizar el objeto a nivel de píxeles, solamente detecta el *Bounding Box* que lo rodea.

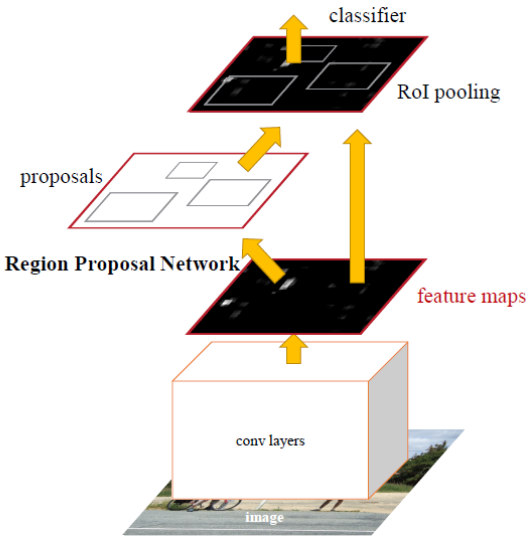


Figura 2.15: Arquitectura Faster R-CNN [9]

Mask R-CNN es el método presentado en el artículo de referencia, que amplía Faster R-CNN añadiendo una rama para predecir una máscara de objeto (segmentación de instancias) en paralelo con la rama existente para el reconocimiento de *Bounding Boxes* [10] (véase Figura 2.16.

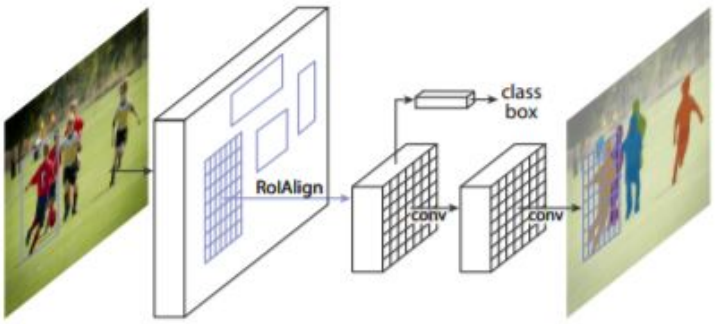


Figura 2.16: Arquitectura Mask R-CNN [10]

Mask R-CNN es fácil de entrenar y sólo añade una pequeña sobrecarga a Faster R-CNN, funcionando a 5 fps. Además, Mask R-CNN es fácil de generalizar a otras tareas, por ejemplo, permitiendo estimar poses humanas en el mismo marco. Faster R-CNN tiene dos salidas para cada objeto, una etiqueta de clase y un *Bounding Box*. A esto se le añade una tercera rama que da como resultado la máscara del objeto. Mask R-CNN es, por tanto, una idea natural e intuitiva. Pero la salida adicional de la máscara es distinta de las salidas de clase y *Box*, lo que requiere la extracción de una distribución espacial mucho más fina de un objeto.

Otra diferencia frente a Faster RCNN es que en la segunda etapa la capa de agrupación de RoIPool se reemplaza por RoIAlign. RoIPool realiza la cuantificación de las regiones de interés, redondeando los valores de punto flotante a valores decimales en el mapa de características resultante provocando que la segmentación de instancias dé como resultado muchas inexactitudes en cuanto a píxeles. RoIAlign evita cualquier cuantificación por completo y alinea correctamente las características extraídas con la entrada.

En la actualidad, el método de detección de objetos más popular es You Only Look Once (YOLO) [11]. Este sistema procesa vídeos en tiempo real con un retraso mínimo conservando una precisión respetable necesitando solo una propagación hacia delante para detectar todos los objetos en una imagen. La última

iteración de YOLO es YOLOv3 [50]. Esta es más precisa en objetos pequeños y un poco peor en objetos más grandes comparado con sus anteriores versiones.

YOLO genera el *bounding box*, la confianza y la clase para cada *box*, mientras que YOLOv3 predice *bounding box* a tres escalas diferentes en distintas profundidades de la red. A través del método de Non Maximum Suppression (NMS) [51], se descartan todos los *bounding box* que se superponen entre sí más de un umbral de IoU previamente decidido, dejando así el *box* de mayor confianza. El método YOLO se representa en la Figura 2.17.

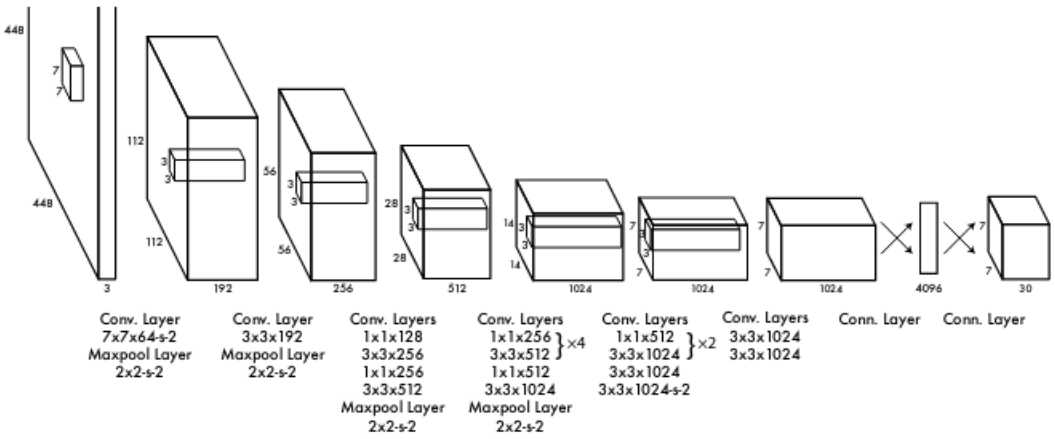


Figura 2.17: Arquitectura YOLO [11]

El método Single Shot MultiBox Detector (SSD) [12] sale a la par que YOLO. De forma similar, sólo necesita una única propagación de la red hacia delante. En este modelo se generan *bounding box* candidatos de diferentes escalas a lo largo del proceso pasando la imagen de entrada a través de una serie de capas convolucionales. Hoy en día se puede ver SSD con una red troncal VGG-16 [52] tal y como se muestra en la Figura 2.18, ResNet [53], Inception [54] e incluso MobileNet [55].

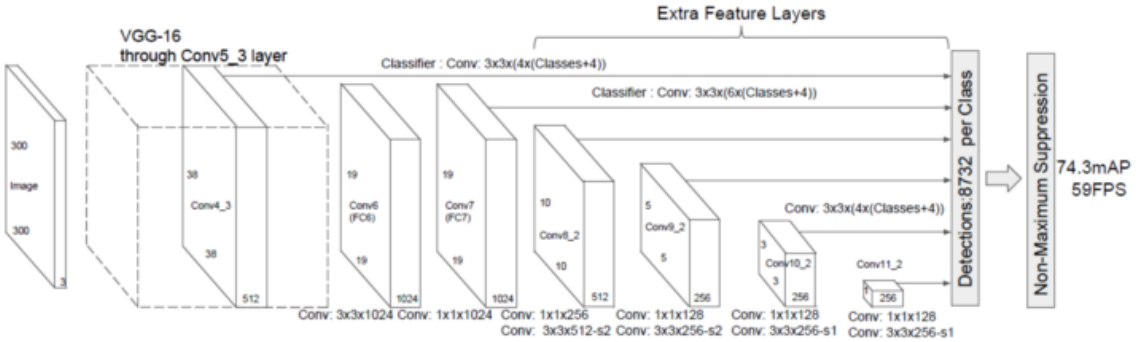


Figura 2.18: Arquitectura SSD [12]

Los objetos etiquetados se consideran ejemplos positivos, mientras que los *box* que no se superpongan con los positivos son ejemplos negativos. SSD aplica la minoría negativa justo después de realizar la NMS Con la minoría negativa se seleccionan solo los ejemplos negativos que tengan la mayor pérdida de confianza (relación entre los positivos y negativos de máximo 1:3).

RetinaNet [13] (véase Figura 2.19) fue propuesto en 2017 y también necesita una única etapa. Como red troncal utiliza ResNet [53] + Feature Pyramid Network (FPN) [56] generando una pirámide de características convolucionales de múltiples escalas. Tiene dos subredes en la capa superior: una clasifica las *boxes* y otra genera el desplazamiento desde las *boxes* hasta las *boxes* del *ground truth*.

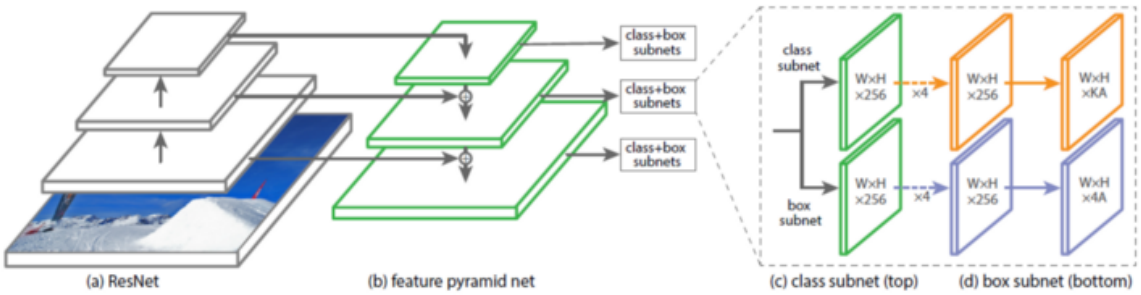


Figura 2.19: Arquitectura RetinaNet [13]

2.4.2 Seguimiento de objetos

Recurrent YOLO (ROLO) [14] es un método de seguimiento de un solo objeto que combina la detección de objetos y RNN (véase Figura 2.20). Utiliza YOLO, para recopilar características visuales, con LSTM [57] para devolver la ubicación del objeto rastreado.

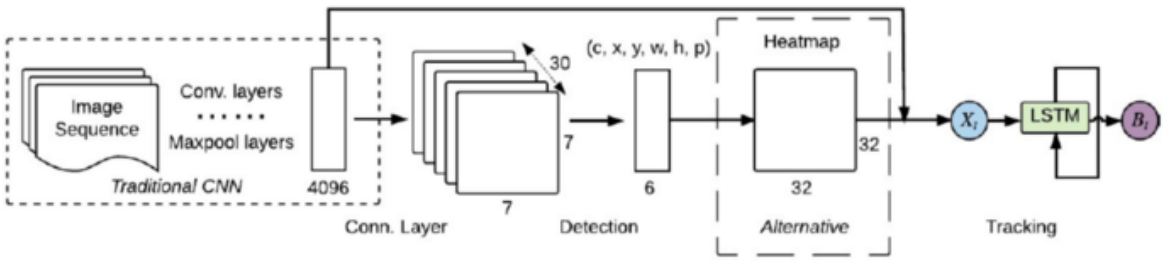


Figura 2.20: Arquitectura ROLO [14]

Otro método de seguimiento de un solo objeto es SiamMask [15]. Produce *bounding box* rotados a 55 fps y proporciona máscaras de segmentación de objetos independientes de la clase. Para lograrlo, debe inicializarse con un solo *bounding box* para que pueda rastrear el objeto deseado. Este método se representa en la Figura 2.21.

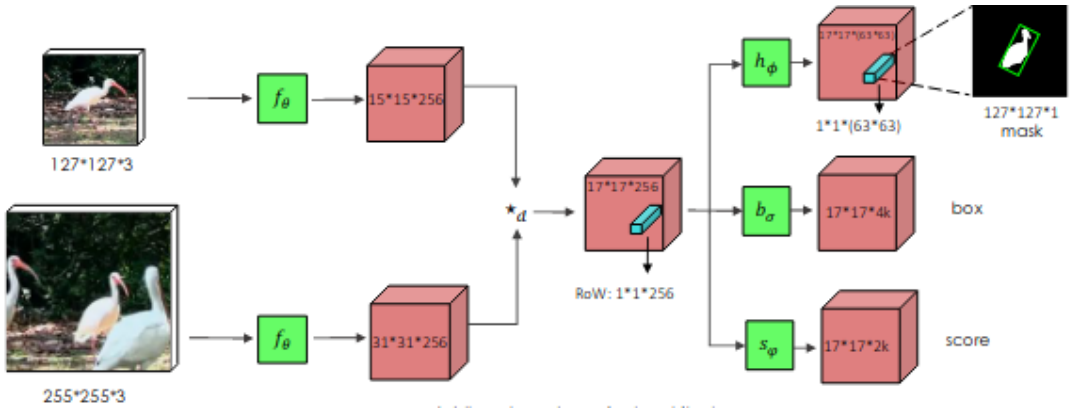


Figura 2.21: Arquitectura SiamMask [15]

Deep SORT [58] mejora a SORT con el remplazo de la métrica de asociación por un nuevo aprendizaje de métricas de coseno (reparametrización del régimen *softmax*). Son métodos muy similares, la diferencia se encuentra en que los *bounding box* se calculan utilizando una CNN previamente entrenada en un conjunto de datos de reidentificación de personas a gran escala. Se caracteriza por ser simple de

implementar, ofrece una precisión sólida y se ejecuta en tiempo real.

TrackR-CNN (véase Figura 2.22) se introdujo como base para el desafío de MOTS y se presenta en el artículo de referencia de este trabajo [1]. Para la detección de objetos a lo largo del tiempo se utiliza el módulo Mask R-CNN sobre una red troncal ResNet-101. Esta capa completamente conectada recibe propuestas de región y genera un vector de asociación para cada propuesta. Para producir el resultado final, el sistema debe decidir qué detecciones deben notificarse. La coincidencia entre las detecciones de fotogramas anteriores y las propuestas actuales se realiza mediante el algoritmo húngaro, mientras que solo permite pares de detecciones con vectores de asociación menores que algún umbral. El seguimiento se crea integrando convoluciones 3D que se aplican a las características de la red troncal, incorporando el contexto temporal del vídeo. Como alternativa, también se considera LSTM convolucional.

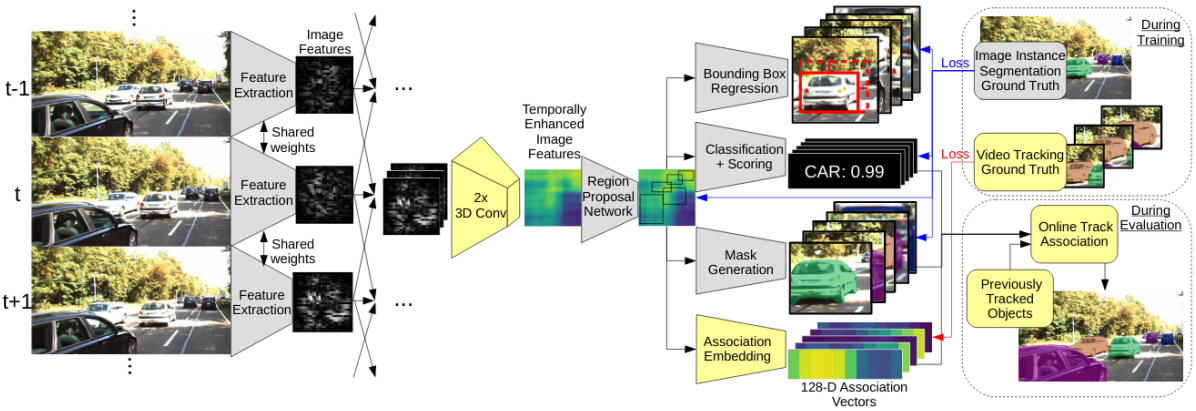


Figura 2.22: Procedimiento TrackR-CNN [1]

Tracktor++ [16] predice la posición de un objeto en el siguiente fotograma calculando la regresión del *bounding box*, sin necesidad de entrenar u optimizar los datos de seguimiento. El detector de objetos para Tracktor++ es Faster R-CNN, ResNet y FPN de 101 capas, entrenado en el conjunto de datos de detección de peatones MOT17Det [21] (véase Figura 2.23). Tracktor++ utiliza la rama de regresión de Faster R-CNN para el seguimiento de fotograma a fotograma extrayendo características del fotograma actual y luego utilizando las ubicaciones de los objetos del fotograma anterior como entrada para que el proceso de agrupación de RoI retroceda su ubicación en el marco actual. También emplea algunos modelos de movimiento tales como la compensación de movimiento de cámara basado en el registro de imágenes, y a corto plazo reidentificación. El método de reidentificación almacena en caché las pistas desactivadas para un número fijo de fotogramas y luego compara las pistas recién detectadas con ellas para una posible reidentificación. La distancia entre pistas se mide mediante una red neuronal siamesa.

Joint Detection and Embedding (JDE) [17] es similar a RetinaNet (véase Figura 2.24). Utiliza una red troncal para obtener mapas de características de la entrada en tres escalas. Estos mapas se fusionan utilizando conexiones residuales y de muestreo ascendente. Finalmente, los cabezales de predicción se adjuntan en la parte superior de los mapas de características fusionados, que generan un mapa de predicción denso para abordar la clasificación de *anchors*, la regresión de *bounding box* y el aprendizaje integrado. JDE también genera vectores de incrustación de apariencia al procesar los marcos. Estas incrustaciones de apariencia se comparan con incrustaciones de objetos detectados previamente utilizando una matriz de afinidad. Finalmente, el viejo algoritmo húngaro y el filtro de Kalman se utilizan para suavizar las trayectorias y predecir las ubicaciones de los objetos detectados previamente en el cuadro actual.

Además de operar a nivel de píxel, existen otras propuestas que operan a nivel de superpíxel como puede ser el caso de [59] o de [60].

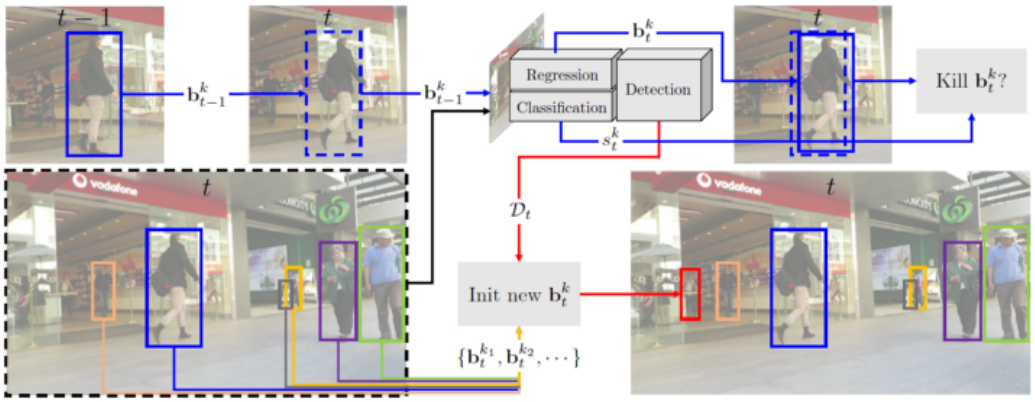


Figura 2.23: Arquitectura Tracktor ++ [16]

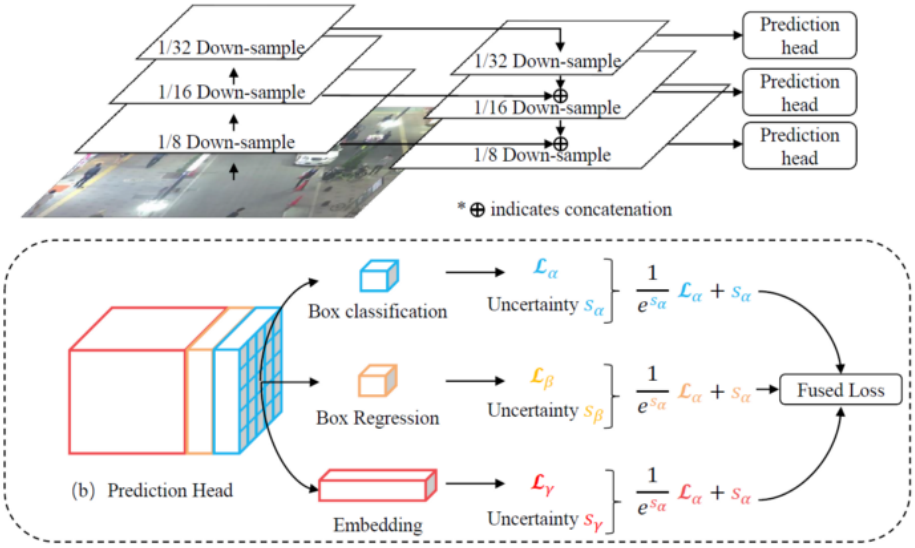


Figura 2.24: Arquitectura JDE [17]

Por otro lado, Class-Agnostic Multi-Object Tracker (CAMOT) [18] realiza un seguimiento basado en máscaras de objetos genéricos en el conjunto de datos KITTI utilizando información estereoscópica. Esto tiene un inconveniente, y es que limita su precisión para objetos distantes. Este método permite rastrear tanto objetos de clases predefinidas como objetos arbitrarios basados en propuestas de categoría agnóstica utilizando la información 3D de la configuración estéreo de KITTI. Esto hace que sea capaz de operar en entornos ricos, en los que un robot puede captar objetos no vistos anteriormente, para los que los detectores pueden ser difíciles de obtener.

El enfoque de este método se muestra en el esquema de la figura 2.25. Extrae un gran número de propuestas de regiones de las imágenes de entrada representadas como máscaras de segmentación con precisión de píxeles, una por cada instancia de objeto propuesto. Las máscaras se almacenan y comparan de forma eficiente utilizando una representación comprimida con codificación de longitud de ejecución RLE. A continuación, utilizando la profundidad del estéreo (nube de puntos) (véase Figura 2.26, se localizan dichas propuestas en 3D. Proporciona como salida una trayectoria hipotética para cada uno de esos objetos. Opcionalmente, se aplica un clasificador de regiones Faster R-CNN a cada trayectoria hipotética para proporcionar una etiqueta semántica para las categorías de objetos conocidos, ayudando a resolver las ambigüedades entre entre hipótesis de objetos que se solapan.

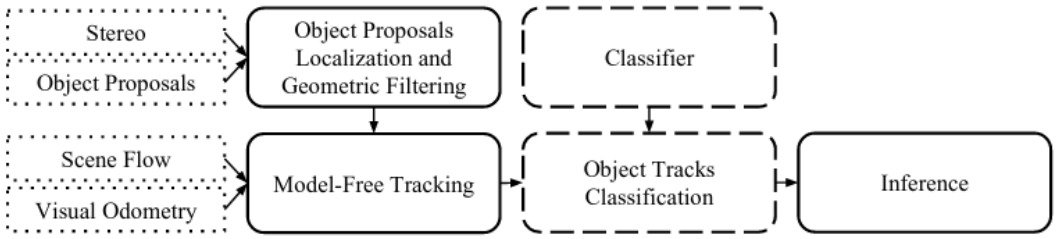


Figura 2.25: Recibe un gran conjunto de propuestas de regiones, crea un conjunto de hipótesis de objetos "propuestos" mediante el rastreo y, finalmente, selecciona aquellos que son la explicación más probable de las señales observadas. Opcionalmente, se puede utilizar un clasificador para identificar la categoría del objeto [18].

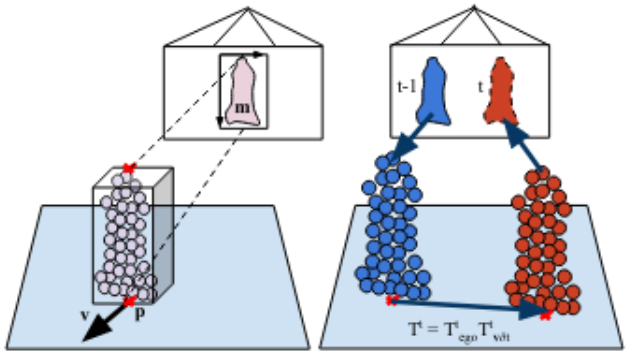


Figura 2.26: (izquierda) Representación de las hipótesis. Cada hipótesis está representada por vectores de posición y velocidad en el espacio 3D y la máscara de píxeles de la imagen. Los *Bounding Boxes* pueden derivarse trivialmente de la máscara de píxeles en el dominio de la imagen, y del espacio 3D dada la nube de puntos. (derecha) Predicción de la máscara visualización. Aprovechando la información sobre el movimiento del ego y la velocidad (dada en 3D), obtenemos predicciones de apariencia precisas en píxeles de los objetos objetos rastreados en los fotogramas futuros [18]

No existen muchos rastreadores basados en máscaras con código fuente disponible, pero existen métodos de rastreo basados en *Bounding Boxes*. Combined Image- and World-Space Tracking (CIWT) [61] combina la información basada en la imagen con la información 3D de la estereoscopía para el seguimiento conjunto en el espacio de la imagen y del mundo. BeyondPixels [62] es uno de los métodos de seguimiento más potentes para coches en el conjunto de datos de seguimiento original de KITTI. Combina la información de la apariencia con pistas 3D. Recurrent Rolling Convolution (RRC) [63] consigue una localización precisa en KITTI en particular, mientras que el detector más convencional Mask R-CNN fue diseñado para la detección general de objetos. Estos métodos siguen mostrando que hay límites para las técnicas de seguimiento de *Bounding Boxes* del estado del arte en MOTS cuando simplemente se segmentan las *boxes* utilizando Mask R-CNN.

Collaborative Detection, Tracking, and Segmentation (CDTS) [64] realiza un Video Object Segmentation (VOS) no supervisado, es decir, sin utilizar la información del primer fotograma. Sólo considera *clips* de vídeo cortos con pocas apariciones y desapariciones de objetos. Sin embargo, en MOTS, muchos objetos entran o salen con frecuencia de una escena llena de gente. Los métodos mencionados anteriormente son capaces de producir resultados de seguimiento con máscaras de segmentación pero su rendimiento no puede ser evaluado de forma exhaustiva, ya que no existe ningún conjunto de datos con anotaciones de MOTS.

Capítulo 3

Desarrollo

3.1 Introducción

En este capítulo se mostrarán los conjuntos de datos necesarios para poder trabajar con los sistemas de detección multi-objeto y las métricas de evaluación que se emplean para comparar los diferentes métodos. Además, se mostrarán varios métodos que abordan las tareas Multi-object Tracking (MOT) y MOTS, los cuales aportan su código para poder entrenarlo, y que obtienen resultados del conjunto de datos KITTI MOTS. Para todos ellos, se realiza un resumen del experimento, indicando las técnicas utilizadas y los resultados obtenidos.

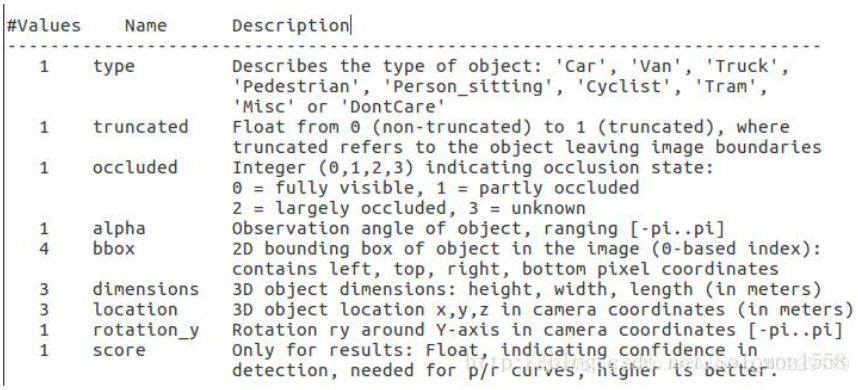
3.2 Conjuntos de Datos

En la actualidad no existen conjuntos de datos para la tarea MOTS. Su disponibilidad es muy limitada debido a que las anotaciones de máscaras a nivel de píxeles para cada *box* en un vídeo es una tarea extremadamente difícil. Sin embargo, sí que hay algunos conjuntos de datos con anotaciones MOT, es decir, seguimientos anotados a nivel de *Bounding Box*. En la tarea de MOT, un número inicialmente desconocido de objetivos de un conjunto conocido de clases deben ser rastreados como *Bounding Boxes* en un vídeo. En particular, los objetivos pueden entrar y salir de la escena en cualquier momento y deben ser recuperados después de una larga oclusión y bajo cambios de apariencia.

Figura 3.1: Plataforma de conducción autónoma Annieway [19]

KITTI Vision Benchmark Suite [19] desarrolla nuevos retos de visión artificial en el mundo real a través de su plataforma de conducción autónoma *Annieway* (véase Figura 3.1). El conjunto de datos KITTI, como bien indica su nombre, fue cofundado por el Instituto de Tecnología de Karlsruhe en Alemania y el Instituto Americano de Tecnología de Toyota [20] . En la actualidad es el conjunto de datos más grande del mundo dentro de la ciencia de la visión artificial para la evaluación de algoritmos en escenarios de conducción autónoma. Sus tareas de interés son: estéreo, flujo óptico, odometría visual, detección de objetos en 3D y seguimiento en 3D. Para ello, equipan una camioneta estándar con dos cámaras de vídeo de alta resolución en color y en escala de grises. Un escáner láser *Velodyne* y un sistema de localización GPS proporcionan el *ground truth*. Sus conjuntos de datos se capturan conduciendo por la ciudad mediana de Karlsruhe, en zonas rurales y en autopistas. Además de proporcionar todos los datos en formato bruto, extraen puntos de referencia para cada tarea. Para cada uno de sus puntos de referencia, también proporcionan una métrica de evaluación y el sitio web de evaluación. Su objetivo es reducir el rendimiento de los datos cuando se trasladan del laboratorio al mundo real. Cada imagen contiene hasta 15 automóviles y 30 peatones, con varios grados de oclusión y truncamiento. El conjunto de datos completo se compone de 389 pares de imágenes estéreo y diagramas de flujo ópticos, secuencias de distancia visual de 39,2 km e imágenes de más de 200.000 objetos etiquetados en 3D, muestreados y sincronizados a una frecuencia de 10 Hz.

Figura 3.2: Ejemplo de etiquetado en una imagen del conjunto de datos KITTI [20]



#Values	Name	Description
1	type	Describes the type of object: 'Car', 'Van', 'Truck', 'Pedestrian', 'Person_sitting', 'Cyclist', 'Tram', 'Misc' or 'DontCare'
1	truncated	Float from 0 (non-truncated) to 1 (truncated), where truncated refers to the object leaving image boundaries
1	occluded	Integer (0,1,2,3) indicating occlusion state: 0 = fully visible, 1 = partly occluded 2 = largely occluded, 3 = unknown
1	alpha	Observation angle of object, ranging [-pi..pi]
4	bbox	2D bounding box of object in the image (0-based index): contains left, top, right, bottom pixel coordinates
3	dimensions	3D object dimensions: height, width, length (in meters)
3	location	3D object location x,y,z in camera coordinates (in meters)
1	rotation_y	Rotation ry around Y-axis in camera coordinates [-pi..pi]
1	score	Only for results: Float, indicating confidence in detection, needed for p/r curves, higher is better.

Figura 3.3: Descripción de la etiqueta de datos [20]

En general, el conjunto de datos original se clasifica en "Carretera", "Ciudad", "Residencial", "Campus" y "Persona". Para la detección de objetos 3D, la etiqueta se subdivide en coche, furgoneta, camión, peatón, peatón (sentado), ciclista y tranvía. El formato de etiqueta del conjunto de datos KITTI se muestra en la imagen 3.2. La descripción de cada dato se encuentra en la figura 3.3.

Cuando en la etiqueta señala *'DontCare'* es debido a que el área no está marcada, por ejemplo, porque el objeto de destino está demasiado lejos del LIDAR. Para evitar que durante el proceso de evaluación el área que originalmente era el objeto de destino pero que no se etiquetó por alguna razón se cuenta como falsos positivos, el *script* de evaluación ignorará automáticamente el resultado de la predicción del área *'DontCare'*.

Los conjuntos de datos MOTChallenge (Figura 3.4) [21] muestran a los peatones desde una variedad de puntos de vista diferentes. University at Albany DETection and tRACking (UA-DETRAC) (Figura 3.5) [22] también presenta escenas de la calle, pero contiene anotaciones sólo para los vehículos. Otro conjunto de datos de MOT es PathTrack (Figura 3.6) [23], que proporciona anotaciones de trayectorias humanas en diversas escenas. PoseTrack (Figura 3.7) [24] contiene contiene anotaciones de posiciones conjuntas para múltiples personas en vídeos.

Figura 3.4: Anotaciones en el conjunto de datos MOTChallenge [21]

Figura 3.5: Muestra de fotogramas anotados en el conjunto de datos UA-DETRAC [22]

Ninguno de estos conjuntos de datos proporciona máscaras de segmentación para los objetos anotados y, por lo tanto, no describen interacciones complejas con suficiente detalle.

En el artículo de referencia de este trabajo, la red trocal de ResNet-101 [65] se utiliza para MaskR-CNN. Esta se entrena previamente con el conjunto de datos MS COCO [25] y Mapillary Vistas [26].

MS COCO (Figura 3.8) reúne imágenes de escenas cotidianas complejas las cuales contienen objetos comunes en su contexto natural. Para localizar el objeto de forma precisa, se etiqueta utilizando segmen-

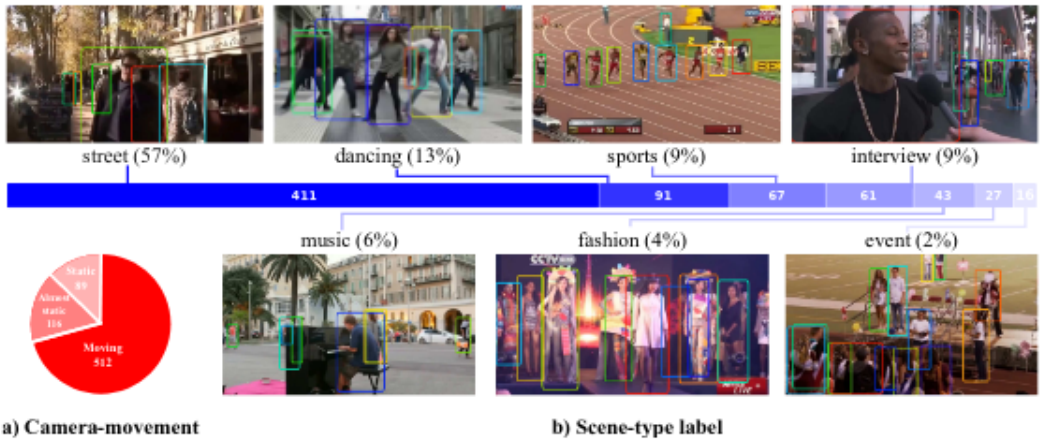


Figura 3.6: Distribución de las etiquetas de la escena en PathTrack. a) distribución de las etiquetas de movimiento de la cámara. Casi tres cuartas partes de las secuencias se han grabado con una cámara en movimiento. b) la distribución de las etiquetas de tipo de escena y los ejemplos correspondientes. Más de la mitad de las secuencias son escenas callejeras [23]

Figura 3.7: Muestra de fotogramas anotados en el conjunto de datos PoseTrack [24]

tación de instancias. Su conjunto de datos contiene fotos de 91 tipos de objetos que serían fácilmente reconocibles por un niño de 4 años. Tiene un total de 2,5 millones de instancias etiquetadas en 328.000 imágenes obtenidas a través de nuevas interfaces de usuario para la detección de categorías, la localización de instancias y la segmentación de instancias. Su conjunto de datos aborda tres problemas centrales de investigación en la comprensión de escenas: la detección de vistas no icónicas de los objetos, el razonamiento contextual entre objetos y la localización precisa de objetos en 2D. Se entiende por vista icónica (véase Figura 3.8), por ejemplo, cuando el objeto aparece de perfil, sin obstáculos, cerca del centro de una foto bien compuesta. Los sistemas de reconocimiento actuales funcionan bastante bien en las vistas icónicas, pero tienen dificultades para reconocer objetos de otro modo (en el fondo, parcialmente ocluidos, en medio del desorden), lo que refleja la composición de las escenas cotidianas reales. La identidad de muchos objetos sólo puede resolverse utilizando el contexto, debido a su pequeño tamaño o a su aspecto ambiguo en la imagen. Para impulsar la investigación sobre el razonamiento contextual, son necesarias imágenes que representen escenas en lugar de objetos aislados.

Para crear un conjunto de datos a gran escala que cumpla estos tres objetivos (véase Figura 3.9), se emplea un novedoso proceso de recopilación de datos con un amplio uso de Amazon Mechanical Turk. En primer lugar, recogen un gran conjunto de imágenes que contenían relaciones contextuales y vistas de

Figura 3.8: Ejemplo de (a) imágenes de objetos icónicos, (b) imágenes de escenas icónicas y (c) imágenes no icónicas [25]

objetos no icónicos. Para ello buscan pares de objetos junto con imágenes recuperadas mediante consultas basadas en escenas. A continuación, se etiqueta cada imagen como si contuviera determinadas categorías de objetos mediante un enfoque de etiquetado jerárquico. Para cada categoría encontrada, las instancias individuales son etiquetadas, verificadas y finalmente segmentadas.

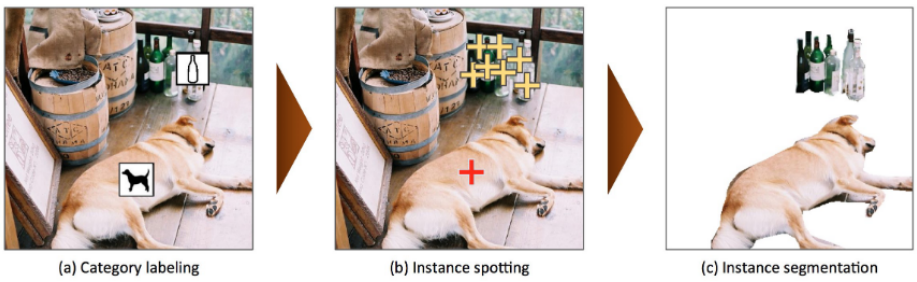


Figura 3.9: Proceso de anotación del conjunto de datos COCO: (a) etiquetar las categorías presentes en la imagen, (b) localizar y marcar todas las instancias de las categorías etiquetadas y (c) segmentar cada instancia de objeto [25]

Mapillary Vistas es un conjunto de datos de imágenes de calles a gran escala que contiene 25000 imágenes de alta resolución anotadas en 66 categorías de objetos con etiquetas adicionales específicas para 37 clases. La anotación de objetos se realiza mediante el uso de polígonos para delinearlos. Contiene imágenes de todo el mundo, capturadas en diversas condiciones climáticas, estacionales y diurnas (véase Figura 3.10). Las imágenes proceden de diferentes dispositivos de imagen (teléfonos móviles, tabletas, cámaras de acción, equipos de captura profesionales) y de fotógrafos con distinta experiencia. De este modo, su conjunto de datos ha sido diseñado y recopilado para cubrir la diversidad, la riqueza de detalles y la extensión geográfica.

En la tarea de VOS, las segmentaciones de instancia para uno o múltiples objetos genéricos se proporcionan en el primer fotograma de un vídeo y deben segmentarse con precisión de píxeles en todos los fotogramas posteriores. Los conjuntos de datos VOS existentes sólo contienen unos pocos objetos que también están presentes en la mayoría de los fotogramas. Además, las métricas de evaluación habituales para esta tarea (índice de *Jaccard* de la región y medida F de los límites) no tienen en cuenta los casos de error, como los cambios de *id*, que pueden producirse al rastrear varios objetos. Por el contrario, MOTS se centra en un conjunto de clases predefinidas y considera escenas abarrotadas con muchos objetos que interactúan. MOTS también añade la dificultad de descubrir y seguir un número variable de objetos nuevos a medida que aparecen y desaparecen en una escena.

El conjunto de datos Densely Annotated VIdeo Segmentation (DAVIS) 2016 (Figura 3.11) [27], para

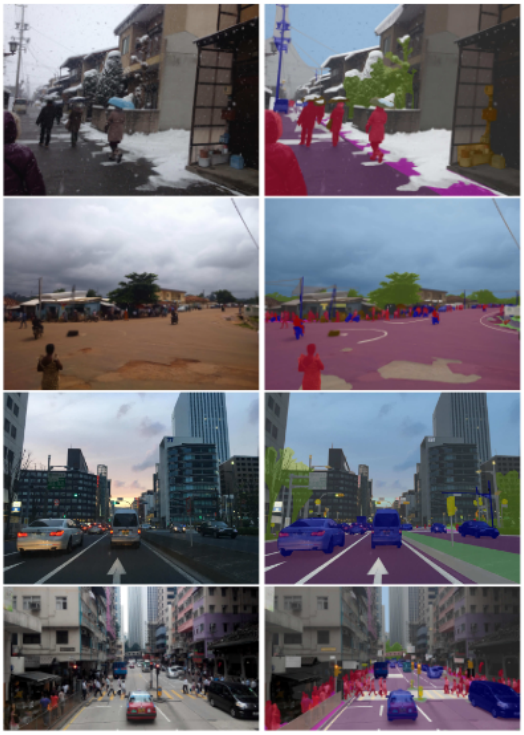


Figura 3.10: Ejemplos de etiquetado cualitativo en Mapillary Vistas. Muestran cuatro pares de imágenes de diferentes regiones geográficas y con diversas condiciones de luz y clima, originales con sus correspondientes etiquetas superpuestas codificadas por colores [26]

el caso de un solo objeto, y el conjunto de datos DAVIS 2017 (Figura 3.12) [28], para varios objetos son conjuntos de datos para la tarea VOS. El conjunto de datos YouTube-VOS (Figura 3.13) [29] es un orden de magnitud mayor que DAVIS. Segtrackv2 [66], Freiburg-Berkeley Motion Segmentation (FBMS) [67] y un subconjunto anotado del conjunto de datos YouTube-Objects [68] también se consideran para esta tarea.

Figura 3.11: Secuencias de muestra del conjunto de datos DAVIS 2016, con máscaras de segmentación de *ground truth* superpuestas [27]

Los conjuntos de datos de segmentación de instancias de vídeo no pueden utilizarse para el enfoque de seguimiento a nivel de píxel ya que las anotaciones de instancia se proporcionan únicamente para un pequeño subconjunto de fotogramas adyacentes. Se pueden encontrar conjuntos de datos como Cityscapes [69], Berkeley Deep Drive (BDD) [70] y ApolloScape [71]. Este último proporciona anotaciones de instancia

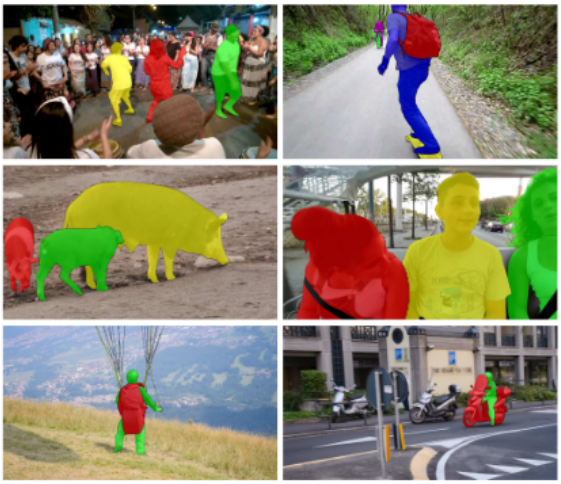


Figura 3.12: Secuencias de muestra del conjunto de datos DAVIS 2017, con anotaciones [28]

Figura 3.13: Secuencias de muestra del conjunto de datos Youtube-VOS, con anotaciones [29]

para cada fotograma pero sin identidades de objeto a lo largo del tiempo, por lo que sigue sin ser válido.

El artículo de referencia proporciona dos nuevos conjuntos de datos basados en los populares conjuntos de datos KITTI y MOTChallenge para entrenar y evaluar métodos que abordan la tarea de MOTS. La utilidad de estos nuevos conjuntos está demostrada para la formación integral de rastreadores multi-objeto a nivel de píxeles. El procedimiento de anotación de MOTS, añade máscaras de segmentación en los *Bounding Boxes*. En total, se han anotado 65.213 máscaras de segmentación.

KITTI MOTS realiza un proceso semiautomático para ampliar las anotaciones a nivel de *Bounding Box* del conjunto de datos KITTI mediante máscaras de segmentación. Se utiliza una CNN que produce de forma automática máscaras de segmentación a partir de *bounding boxes*, seguida de un proceso de corrección a través de anotaciones manuales que afinan la red inicial. Se repite este proceso hasta alcanzar una precisión a nivel de píxel de todas las máscaras de anotación. Para facilitar el entrenamiento y la evaluación, se dividen las 21 secuencias de entrenamiento del conjunto KITTI en un conjunto de entrenamiento y validación, respectivamente. MOTSChallenge se centra en escenas con multitud de peatones, donde es muy complicado debido a que surgen muchos casos de oclusión para los cuales un píxel aporta una descripción especialmente beneficiosa. La Figura 3.14 muestra las anotaciones a partir del conjunto de datos KITTI MOTS y MOTSChallenge.

Figura 3.14: Ejemplo de anotaciones en imágenes, KITTI MOTS (arriba) y MOTSChallenge (abajo)

3.3 Métricas de Evaluación

Un rastreador de objetos ideal debe encontrar en todo momento el número correcto de objetos presentes, estimar la posición de cada objeto con la mayor precisión posible y mantener un seguimiento coherente de cada objeto a lo largo el tiempo (a cada objeto se le debe asignar un único ID de seguimiento que se mantiene constante a lo largo de la secuencia incluso después de una oclusión temporal). Esto lleva a que las métricas de rendimiento deban permitir juzgar la precisión de un rastreador para determinar la ubicación exacta de los objetos; reflejen su capacidad para seguir de forma consistente las configuraciones de los objetos a lo largo del tiempo produciendo exactamente una trayectoria por objeto; tener el menor número posible de parámetros libres, umbrales ajustables, etc., para ayudar a que las evaluaciones sean sencillas y a que los resultados sean comparables; ser claras, fácilmente comprensibles y que se comporten de acuerdo con la intuición humana; ser lo suficientemente general para permitir la comparación de la mayoría de tipos de rastreadores (rastreadores 2D, 3D, rastreadores del centro del objeto, rastreadores del área del objeto, etc.); y ser poco numerosos y, sin embargo, expresivos, para que puedan utilizarse, por ejemplo, en grandes evaluaciones en las que se comparan muchos sistemas.

Teniendo en cuenta los criterios anteriores, un procedimiento para la evaluación sistemática y objetiva de las características de un rastreador puede ser el siguiente. Suponiendo que para cada marco temporal t, un rastreador multi-objeto emite un conjunto de hipótesis $\{h_1, ..., h_m\}$ para un conjunto de objetos visibles $\{o_1, ..., o_n\}$, el procedimiento de evaluación comprende los siguientes pasos. Para cada marco temporal t debe establecer la mejor correspondencia posible entre hipótesis h_j y los objetos o_i. Para cada correspondencia encontrada, calcular el error en la estimación de la posición del objeto. Para cada marco temporal t debe acumular todos los errores de correspondencia: contar todos los objetos para los que no se ha obtenido ninguna hipótesis como fallos, contar todas las hipótesis del rastreador para las que no existe ningún objeto real como falsos positivos y contar todos los casos en los que la hipótesis de seguimiento de un objeto ha cambiado en comparación con los fotogramas anteriores como errores de coincidencia. Esto podría ocurrir, por ejemplo, cuando dos o más objetos se intercambian al pasar cerca el uno del otro, o cuando la pista de un objeto se reinicia con un con un ID de rastreo diferente, después de haberse perdido por oclusión.

Entonces, el rendimiento del seguimiento puede expresarse intuitivamente en dos números: la "precisión del seguimiento", que expresa lo bien que se estiman las posiciones exactas de los objetos, y la "precisión

del seguimiento", que muestra cuántos errores cometió el rastreador en términos de fallos, falsos positivos, desajustes fallos en la recuperación de rastreos, etc.

El interés por realizar evaluaciones sistemáticas con bases de datos y métricas comunes en el ámbito del seguimiento ha incrementado en los últimos años. Un ejemplo de ello es CLEAR [72].

En el artículo de referencia, como medidas de evaluación se adapta a la tarea de MOTS la métrica CLEAR MOT para el seguimiento de múltiples objetos. Las máscaras de segmentación por objeto deben incluirse en la métrica de evaluación. Se requiere que tanto las máscaras *Ground Truth* de los objetos como las máscaras producidas por un método MOTS no se superpongan, es decir, que cada píxel pueda asignarse como máximo a un objeto.

CLEAR propone un procedimiento exhaustivo para detectar los tipos básicos de errores producidos por los rastreadores de múlti-objeto e introduce dos métricas novedosas, la Multi-Object Tracking Precision (MOTP), y la Multi-Object Tracking Accuracy (MOTA), que expresan intuitivamente las fortalezas generales de un rastreador y que son adecuadas para su uso en evaluaciones generales de rendimiento. Estas dos nuevas medidas de rendimiento global permiten una visión clara e intuitiva de las principales características del rastreador: su precisión en la estimación de las posiciones de los objetos, su capacidad para determinar el número de objetos y su configuración, y su habilidad para mantener un seguimiento consistente a lo largo del tiempo.

Formalmente, el *Ground Truth* de un vídeo con marcos de tiempo T, altura h, y anchura w consiste en un conjunto de N máscaras de píxeles de M *Ground Truth* con $M = \{m_1, ..., m_N\}$ con $m_i \in \{0,1\}^{h \times w}$, cada uno de los cuales pertenece a un tiempo correspondiente frame $t_m \in \{1, ..., T\}$ y se le asigna un seguimiento de Ground Truth i_d $id_m \in N$. El resultado de un método MOTS es un conjunto de K hipótesis enmascaradas $H = \{h_1, ..., h_K\}$ con $h_i \in \{0,1\}^{h \times w}$, a cada uno de los cuales se le asigna una hipótesis *track* i_d $id_h \in N$ y un marco de tiempo $t_h \in \{1, ..., T\}$.

Se deben establecer correspondencias entre los objetos *Ground Truth* y las hipótesis del rastreador. En la configuración basada en *Bounding Box*, el establecimiento de las correspondencias no es trivial y se realiza mediante la coincidencia bipartita [73], ya que los *Ground Truth Boxes* pueden solaparse y varios *boxes* hipotéticos pueden encajar bien en un *Ground Truth Box* determinado. En el caso de MOTS, el establecimiento de correspondencias se simplifica, ya que cada píxel se asigna de forma exclusiva a un objeto como máximo en el *Ground Truth* y en las hipótesis, respectivamente. Así, como máximo una máscara de predicción puede tener una IoU de más de 0,5 con una determinada máscara *Ground Truth* [74].

Por lo tanto, el mapeo $c : H \rightarrow M \cup \{\emptyset\}$ de las máscaras de hipótesis al *Ground Truth*, puede ser definido simplemente usando IoU basado en máscaras como

$$c(h) = \begin{cases} argmax_{m \in M} IoU(h, m) & \text{si } maxIoU(h, m)_{m \in M} > 0.5 \\ 0 & \text{si } otherwise \end{cases} \tag{3.1}$$

El conjunto de verdaderos positivos $TP = \{h \in H \mid c(h) = \emptyset\}$ está compuesto por máscaras hipotéticas que se mapean para una máscara verdadera.

$$\tilde{TP} = \sum_{h \in TP} IoU(h, c(h)) \tag{3.2}$$

De manera similar, los falsos positivos son máscaras con hipótesis que no están mapeadas a ninguna verdad fundamental, es decir, $FP = \{h \in H \mid c(h) = \emptyset\}$. Finalmente, el conjunto $FN = \{m \in M \mid$

$c^{-1}(m) = \emptyset\}$ de falsos negativos contiene las máscaras de *Ground Truth* que no están cubiertas por ninguna máscara hipotética.

El conjunto Id SwitcheS (IDS) es entonces definido como el conjunto de máscaras de *Ground Truth* cuyo predecesor fue rastreado con una identificación diferente.

$$IDS = \{m \in M \mid c^{-1}(m) \neq \emptyset^{pred(m)} \neq id_{c^{-1}(m)} \neq id_{c^{-1}(pred(m))}\} \tag{3.3}$$

Dadas las definiciones anteriores, se definen nuevas variantes basadas en la métrica original de CLEAR MOT.

Multi-Object Tracking and Segmentation Accuracy (MOTSA) como una versión basada en el IoU de máscara de la métrica MOTA:

$$MOTSA = 1 - \frac{|FN| + |FP| + |IDS|}{|M|} = \frac{|TP| - |FP| - |IDS|}{|M|} \tag{3.4}$$

También, Multi-Object Tracking and Segmentation Precision (MOTSP) como:

$$MOTSP = \frac{\tilde{TP}}{|TP|} \tag{3.5}$$

Finalmente, se introduce sMOTSA, la cual acumula el número moderado TP de verdaderos positivos en lugar de contar cuántas máscaras alcanzan un IoU de más de 0,5:

$$sMOTSA = \frac{\tilde{TP} - |FP| - |IDS|}{|M|} \tag{3.6}$$

Por lo tanto, sMOTSA mide tanto la segmentación como la calidad de la detección y el seguimiento.

3.4 Métodos evaluados

Durante la elaboración de este TFM, y gracias a la gran repercusión que tienen este tipo de métodos, han aparecido otros proyectos enfocados en la misma temática y utilizando el mismo conjunto de datos KITTI que han conseguido una mayor puntuación que el artículo propuesto en *The KITTI Vision Benchmark Suite* en el apartado de *MOTS Evaluation*.

En las tablas 3.1 y 3.2 se muestran únicamente las puntuaciones de los métodos que se hicieron públicos y en negrita se marcan los que se han analizado para este trabajo gracias a la aportación del código para poder ejecutarse.

Method	HOTA	DetA	AssA	DetRe	DetPr	AssRe	AssPr	LocA	sMOTSA
ViP-DeepLab [75]	76.38 %	82.70 %	70.93 %	88.70 %	88.77 %	75.86 %	86.00 %	90.75 %	81.03 %
EagerMOT [76]	74.66 %	76.11 %	73.75 %	79.59 %	90.24 %	76.27 %	92.70 %	90.46 %	74.53 %
MOTSFusion [31]	73.63 %	75.44 %	72.39 %	78.32 %	90.78 %	75.53 %	89.97 %	90.29 %	74.98 %
ReMOTS [77]	71.61 %	78.32 %	65.98 %	83.51 %	87.42 %	68.03 %	92.61 %	89.33 %	75.92 %
PointTrack [78]	61.95 %	79.38 %	48.83 %	85.77 %	85.66 %	79.07 %	56.35 %	88.52 %	78.50 %
TrackR-CNN [1]	56.63 %	69.90 %	46.53 %	74.63 %	84.18 %	63.13 %	62.33 %	86.60 %	66.97 %
GMPHD_SAF [79]	55.14 %	77.01 %	39.76 %	81.57 %	87.29 %	69.22 %	49.42 %	88.72 %	75.39 %
STMask++ [80]	51.90 %	52.52 %	52.40 %	61.41 %	69.77 %	58.55 %	75.76 %	82.12 %	42.06 %

Tabla 3.1: MOTS Evaluation Cars: Only Published Methods

A continuación, se realizará un estudio y evaluación de resultados de los diferentes métodos enunciados y que consiguen mejores puntuaciones.

Method	HOTA	DetA	AssA	DetRe	DetPr	AssRe	AssPr	LocA	sMOTSA
ViP-DeepLab [75]	64.31 %	70.69 %	59.48 %	75.71 %	81.77 %	67.52 %	74.92 %	84.40 %	68.76 %
ReMOTS [77]	58.81 %	67.96 %	52.38 %	71.86 %	82.22 %	54.40 %	88.23 %	84.18 %	65.97 %
EagerMOT [76]	57.65 %	60.30 %	56.19 %	63.45 %	81.58 %	60.19 %	83.35 %	83.65 %	58.08 %
PointTrack [78]	54.44 %	62.29 %	48.08 %	65.49 %	81.17 %	64.97 %	58.66 %	83.28 %	61.47 %
MOTSFusion [31]	54.04 %	60.83 %	49.45 %	64.13 %	81.47 %	56.68 %	70.44 %	83.71 %	58.75 %
GMPHD_SAF [79]	49.33 %	65.45 %	38.32 %	69.62 %	80.98 %	57.88 %	49.77 %	83.82 %	62.87 %
TrackR-CNN [1]	41.93 %	53.75 %	33.84 %	57.85 %	72.51 %	45.30 %	50.74 %	78.03 %	47.31 %
STMask++ [80]	27.11 %	25.07 %	31.00 %	28.26 %	54.57 %	38.12 %	54.79 %	72.43 %	7.28 %

Tabla 3.2: MOTS Evaluation Pedestrians: Only Published Methods

3.5 TrackR-CNN

En el trabajo [1], el sistema MOTS se enfoca mediante una red neuronal y se consigue gracias al uso de TrackR-CNN y MaskR-CNN.

TrackR-CNN (véase la Figura 2.22) puede asociar las detecciones a lo largo del tiempo y manejar la dinámica temporal al haber incorporado una unidad de asociación y dos capas convolucionales 3D separables en profundidad con kernels de filtro de 3x3x3 cada uno (dos dimensiones son espaciales y la tercera es el tiempo), función de activación ReLU y 1024 mapas de características entre la red neuronal y la RPN. Las convoluciones 3D se aplican a las características de la red neuronal para aumentarlas con el contexto temporal. Estas características aumentadas son utilizadas por la RPN. Como alternativa, también se consideran las capas LSTM convolucionales, que conservan la estructura espacial de la entrada calculando sus activaciones mediante convoluciones en lugar de productos matriciales. Las convoluciones 3D se inicializan con una función de identidad, tras lo cual se aplica la ReLU. Cuando se utiliza una LSTM convolucional, los pesos se inicializan de forma aleatoria y se añade una conexión de salto para preservar las activaciones de los pesos preentrenados de las capas posteriores durante los pasos iniciales del entrenamiento. La unidad de asociación es una capa totalmente conectada que recibe RPN como entradas y predice un vector de asociación para cada propuesta. Esto se inspira en los vectores de incrustación utilizados en la reidentificación de personas [81]. Como resultado proporciona detecciones basadas en máscaras junto con características de asociación, donde ambas se introducen en un algoritmo de seguimiento que decide qué detecciones seleccionar y cómo vincularlas a lo largo del tiempo.

MaskR-CNN extrae el contexto temporal del vídeo de entrada integrando convoluciones 3D (donde la tercera dimensión adicional es el tiempo) sobre una red neuronal residual ResNet-101 [65]. Se utiliza este tipo de red ya que las redes neuronales muy profundas son difíciles de entrenar debido al problema del gradiente de desaparición [82]. Aplicar múltiples capas en una CNN provoca un aumento en el error de entrenamiento: si se aumenta la profundidad de la red, la precisión se satura y degrada rápidamente (se produce un aumento del error). Las redes neuronales residuales buscan la manera de que el error de entrenamiento disminuya con una red más profunda. Están compuestas de bloques residuales donde la entrada se agrega directamente a la salida de la red a través de una ruta conocida como conexión de salto. En el artículo [65] ResNet aborda el problema de optimización con una red más profunda omitiendo ciertas capas con el uso de bloques residuales (véase Figura 3.15). Logra aumentar la profundidad sin perjudicar el rendimiento del modelo y, además, reduce el número de parámetros.

Cada vector de asociación representa la identidad de un coche o una persona. Se entrenan de forma que los vectores que pertenecen a la misma instancia estén cerca unos de otros y los vectores que pertenecen a instancias diferentes estén lejos unos de otros. Se define la distancia $d(\vec{v}, \vec{w})$ entre dos vectores de asociación $\vec{v}$ y $\vec{w}$ como su distancia euclídea, es decir:

$$d(\vec{v}, \vec{w}) := ||\vec{v} - \vec{w}|| \tag{3.7}$$

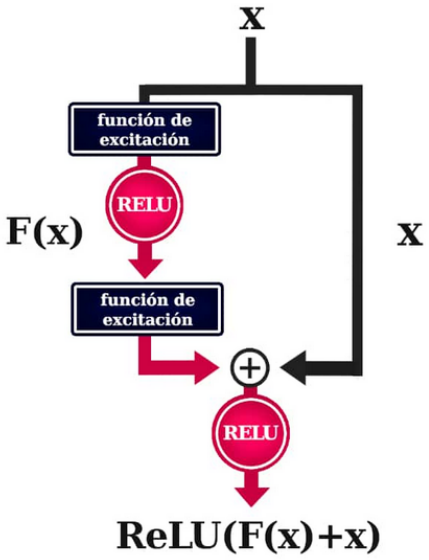


Figura 3.15: Un bloque residual se compone de una ruta residual (izquierda) y otra conexión atajo (derecha) que la soslaya. La ruta residual $F(x)$ se compone de dos capas de pesos sinápticos (ya sean densas o convolucionales) intercalados por una función rectificadora. El resultado se suma a la información que atraviesa la conexión atajo x "identidad". Y tras ello se aplica de nuevo la función rectificadora. La información (y también el gradiente, en sentido inverso) puede atravesar con ello dos caminos diferentes: el de la función identidad x o el de la ruta residual $F(x)$.

D denota el conjunto de detecciones para un vídeo. Cada detección $d \in D$ consiste en una máscara $mask_d$ y una asociación de vectores α_d, que vienen del marco de tiempo t_d, y se le asigna un id id_d determinado por su superposición con los objetos *Ground Truth*. Para una secuencia de vídeo de pasos de tiempo T, la pérdida de asociación con margen α viene dada por

$$\frac{1}{|D|}\sum_{d\in D} max(max_{e\in Di,id_e=id_d}||a_e - a_d|| - min_{e\in Di,id_e\neq id_d}||a_e - a_d|| + \alpha, 0)) \tag{3.8}$$

El IoU basado en la máscara junto con la deformación del flujo óptico es clave para asociar máscaras de píxeles a lo largo del tiempo.

Por lo tanto, una alternativa a las similitudes del vector de asociación es experimentar con la deformación de la máscara. Para una detección $d \in D$ en el tiempo $t-1$ con máscara d y una detección $e \in D$ con máscara $mask_e$ en el tiempo t, se define la puntuación de propagación de la máscara como:

$$maskprop(mask_d, mask_e) = IoU(W(mask_d), mask_e) \tag{3.9}$$

donde $W(m)$ denota la deformación de la máscara m por el flujo óptico entre los fotogramas $t-1$ y t.

En el resultado final se deben decidir las detecciones a notificar y cómo enlazarlas en trayectorias a lo largo del tiempo. Para ello, se amplían las trayectorias existentes con nuevas detecciones basadas en la similitud de su vector de asociación con la detección más reciente de esa trayectoria.

Para cada clase y fotograma t, se enlazan las detecciones en el fotograma actual que tienen una confianza del detector mayor que un umbral γ con las detecciones seleccionadas en los fotogramas anteriores utilizando las distancias del vector de asociación de la ecuación 3.7. Sólo se elige la detección más reciente para las trayectorias de hasta un umbral de β fotogramas en el pasado. A través del algoritmo húngaro se realiza el emparejamiento, mientras que sólo se permiten pares de detecciones con una distancia

menor que un umbral δ. Por último, todas las detecciones de alta confianza no asignadas inician nuevas trayectorias.

Las trayectorias resultantes pueden contener máscaras superpuestas que no se permiten en la tarea MOTS. En tal caso, los píxeles pertenecientes a las detecciones con una mayor confianza tienen prioridad sobre las detecciones con menor confianza.

En este trabajo, se enriquecen directamente las detecciones utilizando la información temporal y se aprenden las características de asociación conjuntamente con el detector, en lugar de limitarse a "postprocesar" las detecciones dadas como en otros métodos. En el trabajo [83] abordan el seguimiento agregando características de localización y apariencia por fotograma y combinándolas a lo largo del tiempo utilizando LSTM. En el trabajo [84] también combinan características de apariencia obtenidas por detecciones recortadas con información de velocidad e interacción utilizando una combinación de LSTM. En ambos casos, las características combinadas se introducen en un procedimiento tradicional de correspondencia húngara.

El procedimiento de anotación de este trabajo puede segmentar muchos objetos de forma totalmente automática, dejando que los anotadores se centren en mejorar los resultados para los casos difíciles (segmentación semiautomática). La decisión de qué objetos anotar se deja a los anotadores humanos para garantizar que todas las anotaciones alcancen la calidad necesaria para un conjunto de datos de referencia a largo plazo.

El método propuesto extiende las anotaciones a nivel de *Bounding Boxes* a máscaras de segmentación. Para convertir las *Bounding Boxes* en máscaras de segmentación, se utiliza una red de refinamiento totalmente convolutiva [85] basada en DeepLabv3+ [86] que toma como entrada un recorte de la imagen de entrada especificada por el *Bounding Box* con una pequeña región de contexto añadida, junto con un canal de entrada adicional que codifica el *Bounding Box* como una máscara. Se itera el proceso de generación y corrección manual de máscaras hasta el nivel de píxeles de precisión.

En otros métodos se requiere de la intervención del usuario para cada objeto a segmentar. Polygon-RNN [87] es otra técnica de anotación semiautomática que predice automáticamente una segmentación en forma de polígono cuyos vértices pueden ser corregidos por el anotador. Fluid Annotation [88] permite al anotador manipular los segmentos predichos por Mask R-CNN [10] para anotar imágenes completas. Los métodos anteriormente mencionados aceleran la creación de máscaras de segmentación de objetos en fotogramas aislados. Sin embargo tienen una serie de inconvenientes: no operan a nivel de rastreo, no hacen uso de las anotaciones existentes de los *Bounding Boxes* y no explotan las máscaras de segmentación que han sido anotadas para el mismo objeto en otros fotogramas de vídeo.

TrackR-CNN se entrena en el conjunto de datos KITTI MOTS o MOTSChallenge, durante 40 épocas con una tasa de aprendizaje de $5 \cdot 10^{-7}$ utilizando el optimizador Adam [89]. Durante el entrenamiento, se utilizan minilotes que constan de 8 fotogramas adyacentes de un solo vídeo, siendo 8 el número máximo de fotogramas que caben en la memoria con una tarjeta gráfica Titan X (Pascal). En los límites de los lotes, la entrada de la capa de convolución 3D se rellena con cero en el tiempo. Cuando se utiliza la LSTM convolucional, los gradientes se propagan hacia atrás a través de los 8 fotogramas durante el entrenamiento y en el momento de la prueba el estado recurrente se propaga por toda la secuencia. Los vectores producidos por la unidad de asociación tienen 128 dimensiones y la pérdida de asociación definida en la ecuación 3.8 se calcula sobre las detecciones obtenidas en un lote. Se elige un margen de $\alpha = 0,2$, que resultó útil en [90]. Para los experimentos de propagación de la máscara, se calcula el flujo óptico entre todos los pares de fotogramas adyacentes utilizando Pyramid, Warping, and Cost (PWC)-Net.

El rastreador alcanza una velocidad alrededor de 2 fotogramas por segundo en el momento de la prueba. Cuando se utiliza la LSTM convolucional, se ejecuta en línea y cuando se utilizan convoluciones

3D, se ejecuta casi en línea gracias a la anticipación de los dos fotogramas los de las convoluciones 3D. Se ajustan los umbrales del sistema de seguimiento (δ, β, γ) para cada clase por separado en el conjunto de entrenamiento con búsqueda aleatoria utilizando 1000 iteraciones por experimento.

3.5.1 Cálculo del flujo óptico: PWC-Net

PWC-Net [30] es un modelo de CNN compacto pero eficaz para el flujo óptico. Está diseñado de acuerdo al procesamiento piramidal, la deformación y el uso de un volumen de costes. PWC-Net utiliza la estimación del flujo óptico actual para deformar las características de la CNN de la segunda imagen. A continuación, utiliza las características deformadas y las de la primera imagen para construir un volumen de costes, que es procesado por una CNN para estimar el flujo óptico. Funciona a unos 35 fps en imágenes con resolución Sintel (1024×436).

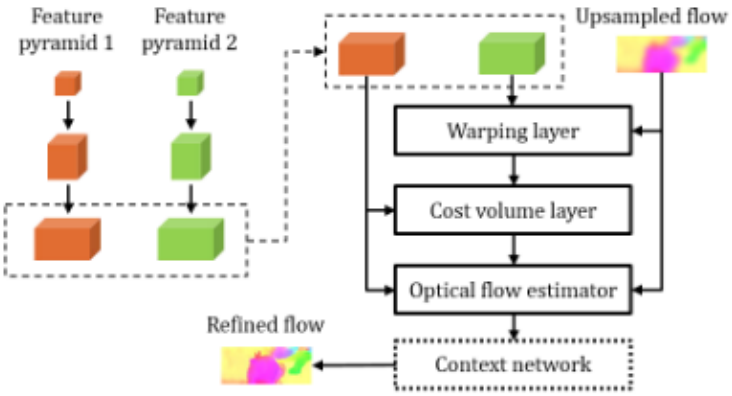


Figura 3.16: Componentes clave de PWC-Net [30]

La figura 3.16 resume los componentes clave de PWC-Net. En primer lugar, dado que las imágenes sin procesar varían en función de las sombras y los cambios de iluminación, se sustituye la pirámide de imágenes fija por pirámides de características aprendibles. En segundo lugar, se toma la operación de *warping* del enfoque tradicional como una capa de la red para estimar el movimiento grande. En tercer lugar, dado que el volumen de costes es una representación más discriminatoria del flujo óptico que las imágenes en bruto, la red tiene una capa para construir el volumen de costes, que luego es procesado por las capas de la CNN para estimar el flujo. Las capas de *warping* y de volumen de costes no tienen parámetros aprendibles y reducen el tamaño del modelo. Por último, PWC-Net utiliza una red de contexto para explotar la información contextual para refinar el flujo óptico. En comparación con la minimización de la energía, las capas de *warping*, volumen de costes y CNN son computacionalmente ligeras.

Extractor de la pirámide de características. Dadas dos imágenes de entrada I_1 e I_2, se generan pirámides de L niveles de representaciones de características, siendo el nivel inferior (*zeroth*) las imágenes de entrada, es decir, ${c_t}^0 = I_t$. Para generar la representación de características en la l^a capa, ${c_t}^l$, se utilizan capas de filtros convolucionales para reducir la muestra de las características en el $l-1^o$ nivel de la pirámide, ${c_t}^{l-1}$, con un factor de 2. Del primer al sexto nivel, el número de canales de características son, respectivamente, 16, 32, 64, 96, 128 y 196.

Capa de deformación. En el l^o nivel, se deforman las características de la segunda imagen hacia la primera imagen utilizando el flujo de subida $\times 2$ del nivel $l+1^o$. Se utiliza la interpolación bilineal para implementar la operación de *warping* y se calculan los gradientes de las características de la CNN de entrada y el flujo para la retropropagación. Para el movimiento no translacional, el *warping* puede compensar algunas distorsiones geométricas y poner los parches de imagen en la escala correcta.

Capa de volumen de costes. A continuación, se utilizan las características para construir un volumen de costes que almacena los costes de correspondencia para asociar un píxel con sus correspondientes píxeles en el siguiente fotograma. En el caso de una pirámide de L niveles, sólo hay que calcular un volumen de coste parcial con un rango limitado de d píxeles. Un movimiento de un píxel en el nivel superior corresponde a 2^{L-1} píxeles en las imágenes de resolución completa. Por lo tanto, se puede establecer que d sea pequeño. La dimensión del volumen de costes 3D es $d^2 x H^l x W^l$, donde H^l y W^l denotan la altura y la anchura del l^o nivel de la pirámide, respectivamente.

Estimador de flujo óptico. Se trata de una CNN multicapa. Su entrada es el volumen de costes, las características de la primera imagen y el flujo óptico sobremuestreado, y su salida es el flujo w^l en el nivel l. El número de canales de características en cada capa convolucional es respectivamente de 128, 128, 96, 64 y 32, que se mantienen fijos en todos los niveles de la pirámide. Los estimadores de los distintos niveles tienen sus propios parámetros en lugar de compartir los mismos. Este proceso de estimación se repite hasta el nivel deseado, l_0.

Red de contexto. Se emplea una subred, denominada red de contexto, para ampliar eficazmente el tamaño del campo receptivo de cada unidad de salida en el nivel de pirámide deseado. Toma el flujo estimado y las características de la penúltima capa del estimador de flujo óptico y produce un flujo refinado. La red de contexto es una CNN feed-forward y su diseño se basa en convoluciones dilatadas. Consta de 7 capas convolucionales. El núcleo espacial de cada capa convolucional es de 3x3. Estas capas tienen diferentes constantes de dilatación. Una capa convolucional con una constante de dilatación k significa que una unidad de entrada a un filtro en la capa está separada por una unidad k de las otras unidades de entrada al filtro en la capa, tanto en dirección vertical como horizontal. Las capas convolucionales con constantes de dilatación grandes amplían el campo receptivo de cada unidad de salida sin incurrir en una gran carga computacional. De abajo a arriba, las constantes de dilatación son 1,2,4,8,16,1, y 1.

3.5.2 Principales resultados obtenidos por los autores

La tabla 3.3 muestra los resultados obtenidos en el artículo en el conjunto de validación KITTI MOTS. Los resultados que se consiguen son competitivos superando varias líneas de base como puede ser Mask R-CNN + *maskprop*, que denota una línea de base simple para la que afina el MS COCO y Mapillary preentrenado Mask R-CNN en los fotogramas del conjunto de entrenamiento de KITTI MOTS. Evalúan en el conjunto de validación y vinculan las detecciones basadas en la máscara a lo largo del tiempo utilizando las puntuaciones de propagación de la máscara. En comparación con esta línea de base, TrackR-CNN logra puntuaciones sMOTSA y MOTSA más altas, lo que implica que las capas de convolución 3D y la unidad de asociación ayudan a identificar objetos en vídeo. Las puntuaciones de MOTSP siguen siendo similares.

	sMOTSA		MOTSA		MOTSP	
	Car	Ped	Car	Ped	Car	Ped
TrackR-CNN (ours)	76.2	46.8	87.8	65.1	**87.2**	**75.7**
Mask R-CNN + maskprop	75.1	45.0	86.6	63.5	87.1	75.6
TrackR-CNN (box-orig) + MG	75.0	41.2	87.0	57.9	86.8	76.3
TrackR-CNN (ours) + MG	76.2	**47.1**	87.8	**65.5**	**87.2**	**75.7**
CAMOT [18] (our det)	67.4	39.5	78.6	57.6	86.5	73.1
CIWT [61] (our det) + MG	68.1	42.9	79.4	61.0	86.7	75.7
BeyondPixels [62] + MG	**76.9**	-	**89.7**	-	86.5	-
GT Boxes (orig) + MG	77.3	36.5	90.4	55.7	86.3	75.3
GT Boxes (tight) + MG	**82.5**	**50.0**	**95.3**	**71.1**	**86.9**	**75.4**

Tabla 3.3: Resultados en KITTI MOTS

TrackR-CNN (*box-orig*) es una versión de su modelo entrenado sin unidad de máscara en las anotaciones originales del *bounding box* de KITTI. Esta línea base se evalúa en la configuración MOTS añadiendo máscaras de segmentación como un paso de post-entrenamiento (+MG) con la unidad de máscara de KITTI afinada R-CNN (puntuaciones MOTA ajustadas según las anotaciones de seguimiento originales de KITTI). Las puntuaciones de sMOTSA y MOTSA para esta configuración son peores que para su método y la línea de base anterior, especialmente cuando se consideran peatones. Esto supone que los *Bounding Boxes* que no son estrechos no son una referencia ideal para el seguimiento y que usar simplemente un método de segmentación de instancias sobre las predicciones de *Bounding Boxes* no es suficiente para resolver la tarea de MOTS. En la figura 3.17 se muestran los resultados cualitativos de esta línea de base. El modelo basado en *boxes* a menudo confunde objetos ocluidos similares entre sí, lo que hace que se pierdan máscaras e identificaciones. Por el contrario, el modelo de este artículo plantea una hipótesis de máscaras coherentes.

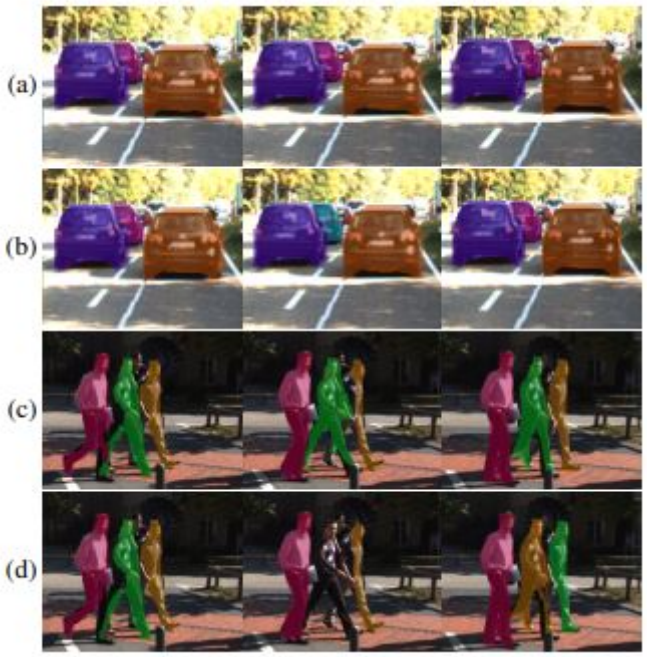


Figura 3.17: Resultados cualitativos de KITTI MOTS. (a)+(c) Su modelo TrackR-CNN evaluado en secuencias de validación de KITTI MOTS. (b)+(d) TrackR-CNN (box orig) + MG evaluados en las mismas secuencias. El entrenamiento con máscaras en sus datos evita la confusión entre objetos similares cercanos

La adición de máscaras de segmentación no es una desventaja. Esto lo demuestran utilizando la unidad de Mask R-CNN para sustituir las máscaras generadas por su método en TrackR-CNN (*ours*) + MG. Ya que los resultados son bastante similares, el utilizar máscaras de segmentación no es una (des)ventaja importante.

Adicional a esto, sus experimentos de referencia muestran las ventajas que ofrece entrenar en datos de segmentación de instancias temporalmente consistentes en vídeo sobre el entrenamiento en datos de segmentación de instancias sin información temporal y sobre el entrenamiento en datos de segmentación de *Bounding Boxes*. Sus propuestas de conjuntos de datos hacen que se puedan entrenar estas dos vertientes conjuntamente, mientras que antes no era posible.

En la versión original, CAMOT toma como entrada propuestas de objetos genéricos de SharpMask [91]. Para una mejor comparabilidad, utilizan en su lugar las detecciones de su TrackR-CNN (obtenidas al

ejecutarlo como un detector normal sin asociación) como entradas. Hay que tener en cuenta que CAMOT sólo puede rastrear regiones para las que se dispone de la profundidad de la imagen estereoscópica, lo que limita su capacidad de recuperación. Los resultados muestran que su método de seguimiento propuesto es significativamente mejor que CAMOT con el mismo conjunto de detecciones de entrada. Dado que no hay muchos rastreadores basados en máscaras con código fuente disponible, también consideran los métodos de rastreo basados en *Bounding Boxes*. CIWT y BeyondPixels y, de nuevo, convierten sus resultados en máscaras de segmentación utilizando el cabezal de Mask R-CNN afinado de KITTI. Hay que tener en cuenta que estos métodos se ajustaron para que tuvieran un buen rendimiento en la tarea original basada en el *Bounding Box*. El sistema de seguimiento propuesto, que aborda el seguimiento y la generación de máscaras de forma conjunta, obtiene mejores resultados que CIWT cuando genera máscaras post-hoc. En cuanto a BeyondPixels, las puntuaciones resultantes de sMOTSA y MOTSA son más altas que las de su método, pero siguen mostrando que hay límites para los métodos de seguimiento de *Bounding Boxes* del estado del arte en MOTS cuando simplemente se segmentan las cajas utilizando Mask R-CNN.

También para comparar, obtuvieron resultados de segmentación basados en *Ground Truth Bounding Box* y los evaluaron con sus nuevas anotaciones. En este caso, consideraron dos variantes de *Ground Truth*: los *Bounding Box* originales de KITTI (*orig*), que son amodales, es decir, si sólo es visible la parte superior del cuerpo de una persona, el *box* se extenderá hasta el suelo; y los *Bounding Boxes* ajustados (*tight*) derivados de sus máscaras de segmentación. Una vez más, generan las máscaras utilizando la Mask R-CNN ajustada de KITTI MOTS. Sus resultados muestran que, incluso con hipótesis de ruta perfectas, generar máscaras precisas sigue siendo un reto, especialmente para los peatones. Esto es aún más cierto cuando se utilizan *boxes* amodales, que a menudo contienen grandes regiones que no muestran el objeto. Esto valida aún más su afirmación de que las tareas MOT pueden beneficiarse de la evaluación por píxeles.

En la tabla 3.4, comparan diferentes variantes de componentes temporales para TrackR-CNN. 1xConv3D y 2xConv3D quiere decir utilizar una o dos capas convolucionales 3D separables en profundidad entre la red troncal y la RPN, cada una con 1024 dimensiones. Del mismo modo, 1xConvLSTM y 2xConvLSTM denota una o dos capas convolucionales LSTM apiladas en la misma etapa con 128 canales de características cada una. El número de parámetros por canal de características en una LSTM convolucional es mayor debido al *gating*. El uso de más canales de características no pareció ser útil durante los experimentos iniciales. Por último, *None* indica que no se añaden capas adicionales como componente temporal. En comparación con la línea de base de *None*, añadir dos convoluciones 3D mejora significativamente las puntuaciones de sMOTSA y MOTSA para los peatones, mientras que el rendimiento para los coches sigue siendo comparable. Sorprendentemente, el uso de LSTM convolucionales no produce ninguna ganancia significativa con respecto a la línea de base.

Temporal component	sMOTSA		MOTSA		MOTSP	
	Car	Ped	Car	Ped	Car	Ped
1xConv3D	76.1	46.3	87.8	64.5	87.1	**75.7**
2xConv3D	76.2	**46.8**	87.8	**65.1**	87.2	**75.7**
1xConvLSTM	75.7	45.0	87.3	63.4	87.2	75.6
2xConvLSTM	76.1	44.8	**87.9**	63.3	87.0	75.2
None	**76.4**	44.8	**87.9**	63.2	**87.3**	75.5

Tabla 3.4: Diferentes componentes temporales para TrackRCNN. Comparación de resultados en KITTI MOTS

En la figura 3.5, comparan diferentes mecanismos utilizados para la asociación entre detecciones. Cada línea sigue el sistema de seguimiento propuesto, pero se utilizan diferentes puntuaciones para el paso de asociación húngaro. Cuando se utiliza la unidad de asociación, los vectores de asociación pueden coincidir con detecciones de hasta β fotogramas en el pasado. Para el resto de mecanismos de asociación, sólo es

lógico el emparejamiento entre fotogramas adyacentes.

Association Mechanism	sMOTSA		MOTSA		MOTSP	
	Car	Ped	Car	Ped	Car	Ped
Association head	**76.2**	**46.8**	**87.8**	**65.1**	**87.2**	**75.7**
Mask IoU	75.5	46.1	87.1	64.4	**87.2**	**75.7**
Mask IoU (train w/o assoc.)	74.9	44.9	86.5	63.3	87.1	75.6
Bbox IoU	75.4	45.9	87.0	64.3	**87.2**	**75.7**
Bbox Center	74.3	43.3	86.0	61.7	**87.2**	**75.7**

Tabla 3.5: Diferentes mecanismos de asociación para TrackRCNN. Comparación de resultados en KITTI MOTS

Para Mask IoU sólo utilizan las puntuaciones de propagación de la máscara de la ecuación 3.9, lo que degrada las puntuaciones de sMOTSA y MOTSA. Esto subraya la utilidad de su unidad de asociación, que puede superar a una ruta basada en el flujo óptico utilizando incrustaciones proporcionadas por una única red neuronal. En este caso, también prueban el entrenamiento sin la pérdida de asociación (Mask IoU (*train w/o assoc.)*), que degrada aún más las puntuaciones de MOTSA. Por tanto, la pérdida por asociación también tiene un efecto positivo en el propio detector. Sorprendentemente, utilizando el IoU de *Bounding Box* (donde las *boxes* se deforman con la mediana de los valores de flujo óptico dentro de la *box*, *Bbox IoU*) tiene casi el mismo que el IoU de la máscara. Por último, utilizar sólo las distancias de los centros de las *boxes* (*Bbox Center*) para la asociación, es decir, hacer una búsqueda del vecino más cercano, degrada significativamente el rendimiento.

En conclusión, demuestran que mediante su método de entrenamiento es capaz de superar a otros métodos comparables que sólo se entrenan con seguimiento de *Bounding Boxes* y máscaras de segmentación de una sola imagen. Sus nuevos conjuntos de datos hacen posible este entrenamiento conjunto, lo que que abre muchas oportunidades para la investigación futura.

3.6 MOTSFusion

El método MOTSFusion propone utilizar la reconstrucción 3D para mejorar el seguimiento de objetos [31]: primero se rastrea para reconstruir y después reconstruye para rastrear. Aprovecha el movimiento 3D extraído de las reconstrucciones dinámicas de los objetos para rastrearlos durante largos periodos de oclusión completa y recuperar los detectores perdidos. Primero construye *tracklets* cortos utilizando el flujo óptico 2D, y luego los fusiona en reconstrucciones dinámicas de objetos en 3D. El movimiento preciso del objeto en 3D de estas reconstrucciones se utiliza para fusionar los *tracklets* de oclusión en rastros a largo plazo, y para localizar objetos cuando faltan detecciones. En KITTI, reduce el número de cambios de identificación de los *tracklets* iniciales en más de un 50 %. Además, tiene entre un 60 % y un 70 % menos de menos cambios de ID que los mejores métodos anteriores de la competencia utilizando el mismo conjunto de detecciones.

Muchos de los enfoques actuales de MOT rastrean con éxito los objetos cuando son visibles de forma constante, pero no logran rastrear los objetos a largo plazo a través de la desaparición y la oclusión, asignando identificaciones incorrectas a los objetos cuando vuelven a aparecer. Para que un sistema sea totalmente autónomo es crucial que el seguimiento a largo plazo sea muy preciso. Su propuesta para abordar este problema es utilizar las reconstrucciones 3D dinámicas a la hora de estimar el movimiento 3D de los objetos. Con la información de movimiento que se obtiene, es posible rastrear objetos a través de la oclusión y recuperar las detecciones perdidas. MOTSFusion construye reconstrucciones a partir de los rastros y mejora el seguimiento con estas reconstrucciones.

MOTSFusion se evalúa utilizando el conjunto de datos KITTI. Utilizan la referencia de seguimiento KITTI para evaluar sus resultados de seguimiento tanto de coches como de peatones en escenas de conducción del mundo real. Se han ampliado las anotaciones de máscara a nivel de píxel para evaluar la tarea MOTS. Para la evaluación utilizan el servidor de pruebas oficial de KITTI, así como la división de validación de [1] para la evaluación.

MOTSFusion se basa en cuatro ideas clave. La primera es que se pueden construir *tracklets* cortos precisos utilizando la consistencia de movimiento 2D obtenida de las máscaras de segmentación y el flujo óptico. Esto da lugar a *tracklets* cortos muy precisos para cuando los objetos son visibles de forma contigua. La segunda clave es que para un objeto que sufre transformaciones de cuerpo rígido, existe un conjunto de tales transformaciones que fusiona todas las nubes de puntos de este objeto de cada paso de tiempo en una reconstrucción 3D consistente, y estas transformaciones definen el movimiento 3D de ese objeto. La tercera es que incluso para los objetos no rígidos, si la transformación de cuerpo rígido que mejor se ajusta se calcula de forma robusta, se obtienen estimaciones de movimiento 3D lo suficientemente precisas como para determinar el movimiento 3D global de un objeto. Por último, la cuarta clave es que la consistencia entre el movimiento 3D estimado de dos *tracklets* contiene suficiente información para determinar si estos *tracklets* pertenecen al mismo objeto y deben ser fusionados en un *track* más largo.

Siguiendo estas ideas, desarrollan un algoritmo de dos etapas. La primera etapa es una asociación de las detecciones en *tracklets* cortos y coherentes desde el punto de vista espacio-temporal en el dominio de la imagen calculando una máscara de segmentación para cada detección y midiendo la consistencia de estas máscaras bajo una deformación definida por el flujo óptico. La segunda etapa proyecta esos *tracklets* en un dominio 3D global utilizando el *ego-motion* de la cámara y una estimación de la profundidad por píxel. Para cada *tracklet*, se calcula un conjunto de transformaciones homogéneas que alinean las representaciones del objeto en cada paso de tiempo en una reconstrucción dinámica, definiendo el movimiento 3D preciso del objeto. Extrapolan las trayectorias 3D de cada *tracklet* y fusionan los *tracklets* en pistas de objetos a largo plazo midiendo la consistencia de las trayectorias 3D estimadas. Esta fusión se basa en la suposición de que los objetos se mueven con transformaciones de cuerpo rígido (no válido para peatones ya que pueden moverse de forma articulada. Aun así, funciona muy bien ya que captura con precisión el movimiento que es lo que necesita para el seguimiento y no los detalles del movimiento articulado). De este modo, se pueden salvar los largos periodos de oclusión y las detecciones que faltan. Por último, rellenan las detecciones que faltan y las máscaras de segmentación utilizando las trayectorias estimadas.

Este rastreador explota tanto la consistencia del movimiento en el espacio de la imagen 2D (utilizando el flujo óptico y las máscaras de segmentación), como la consistencia del movimiento en el espacio del mundo 3D (utilizando reconstrucciones 3D dinámicas a partir de estimaciones de profundidad y movimiento del ego) para un seguimiento preciso a largo plazo.

En la figura 3.18 se puede observar la estructura del método MOTSFusion. Como entradas, utiliza fotogramas de vídeo, así como estimaciones de *ego-motion* y profundidad por fotograma. Para sus principales experimentos, utilizan profundidad estéreo (DispNet3 [92]), y *ego-motion* de un algoritmo Simultaneous Localization And Mapping (SLAM) (ORB-SLAM2 [93]). Sin embargo, este método está diseñado para trabajar con cualquier estimación de profundidad disponible (estéreo, LIDAR, Radio Detection And Ranging (RADAR), SFM o monocular) y estimaciones de *ego-motion* (SLAM, GPS, Inertial Measurement Unit (IMU)). Para la detección de objetos, utilizan tanto RRC como TrackR-CNN. Utilizan el flujo óptico obtenido de [92]. Para cada detección de *Bounding Boxes* estiman una máscara de segmentación. Para ello, utilizan una red neuronal totalmente convolucional de [85] denominada BB2SegNet, que recorta y redimensiona una región de la imagen dada por un *Bounding Box* a un parche de 385x385 dando como resultado una máscara de segmentación para cada *box*. Deformando los píxeles de cada máscara de segmentación generan *tracklets* 2D utilizando los valores de flujo óptico en estos píxeles en el

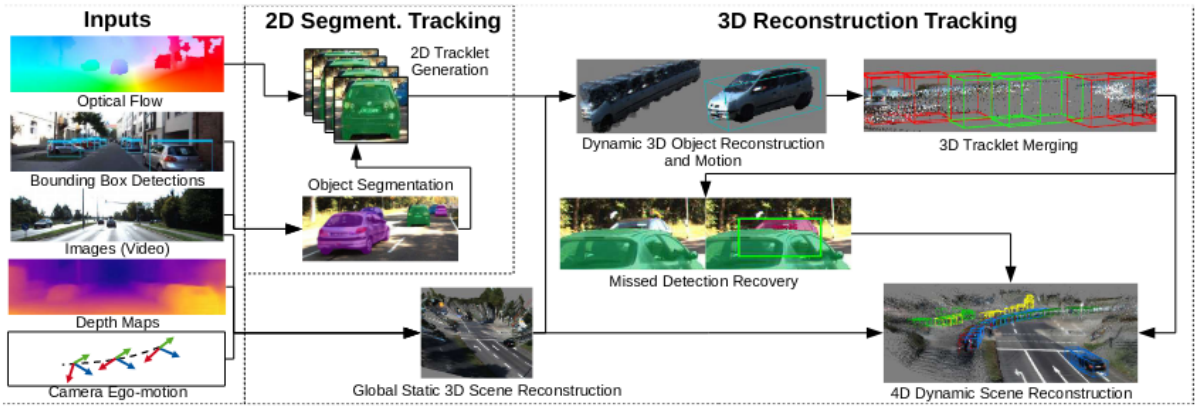


Figura 3.18: Método MOTSFusion [31]

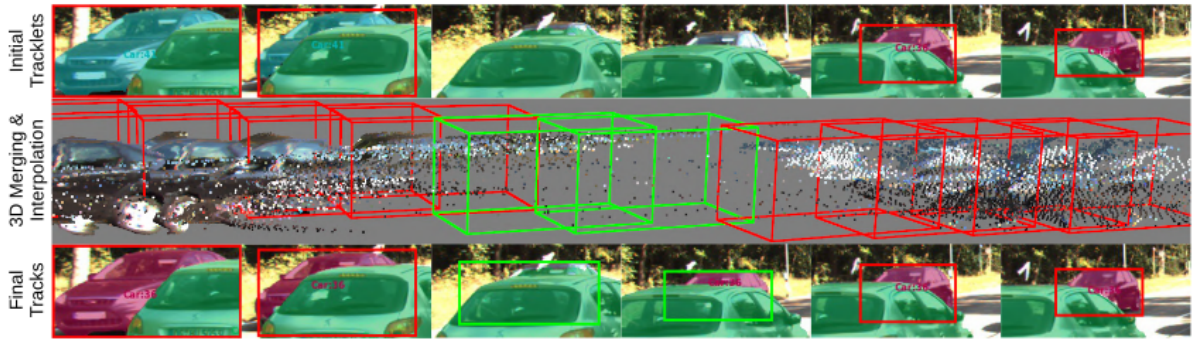


Figura 3.19: Arriba: Detecciones iniciales de *Bounding Boxes* en 2D (rojo) y resultados de la segmentación basada en resultados de los *tracklets*. Centro: Resultado de *Bounding Boxes* 3D interpolados entre los dos *tracklets*, mostrando la consistencia temporal del movimiento 3D de los dos *tracklets* a través de los fotogramas sin detecciones. Abajo: Detección de los *Bounding Boxes* 2D originales (rojo) y los nuevos *Bounding Boxes* 2D interpolados (verde) junto con la máscara de segmentación rellenada y el ID del *track* fusionado [31].

siguiente fotograma. Para crear similitudes de asociación, se calcula el IoU de estas máscaras deformadas y el conjunto de máscaras de segmentación en el siguiente fotograma. Para asignar las máscaras a los *tracklets* previamente existentes se utiliza el algoritmo húngaro. Todas las máscaras que no se fusionan en *tracklets* anteriores comienzan nuevos *tracklets*. Para ello, se requiere un umbral mínimo de IoU para fusionar las máscaras.

Es necesaria la localización 3D de cada objeto en cada paso de tiempo en un marco mundial común para poder obtener el movimiento de los objetos en 3D. Para ese mundo-marco se utiliza la posición de la cámara en el primer fotograma y se crea una nube de puntos 4D (3D + tiempo) de la escena utilizando la profundidad, los intrínsecos de la cámara y la matriz de posición de la cámara. Esta reconstrucción de la escena en 4D se utiliza para estimar el movimiento de los objetos independientemente del movimiento de la cámara del *ego*, y para visualizar el seguimiento 3D.

Para realizar la fusión dinámica de objetos en 3D, se calcula un conjunto de transformaciones de cuerpo rígido que deforman el conjunto de puntos 3D de un *tracklet* en cada paso de tiempo en una reconstrucción consistente del objeto 3D. Se ajusta una transformación homogénea a partir de los puntos del espacio del mundo dentro de una máscara de píxeles en el tiempo t a los puntos del espacio mundial dentro de la máscara de píxeles del mismo *tracklet* en el tiempo $t + 1$. Esta transformación se acumula a lo largo de todos los pasos de tiempo, de modo que todos los puntos de un *tracklet* del objeto es una

reconstrucción 3D coherente. El conjunto de puntos 3D utilizados para ajustar la transformación se filtra a través del cálculo del valor atípico local utilizando la densidad de alcance local de cada punto con sus vecinos más cercanos. Esto se hace para minimizar la influencia de las estimaciones de profundidad incorrectas y de las máscaras de objetos incorrectas. Los puntos que tengan un valor atípico mayor que la mediana del valor atípico de todos los puntos son filtrados. De los puntos 3D restantes en el tiempo t y $t+1$ sólo son utilizados aquellos para los que existen correspondencias entre los dos tiempos. Estos se correlacionan entre los dos tiempos utilizando el vector de flujo óptico. A continuación, se muestrea un máximo de 200 puntos correspondientes sobre la máscara. Sobre estos puntos se realiza una optimización no lineal por mínimos cuadrados minimizando la distancia entre cada par de puntos correspondientes en las dos nubes de puntos para determinar la transformación homogénea que mejor alinea las dos nubes de puntos 3D. Esta transformación está restringida a una transformación homogénea 3-DoF (traslación X y Z, y rotación alrededor de la normal del plano del suelo). Esta transformación proporciona la alineación precisa (movimiento) de un objeto entre pasos de tiempo en coordenadas mundiales. Para cada *tracklet*, se calcula esta transformación para todos los pares de fotogramas vecinos y se acumulan las transformaciones, fusionando las nubes de puntos de cada paso de tiempo en una reconstrucción 3D en un marco de referencia consistente centrado en el objeto.

Los *tracklets* se fusionan analizando su consistencia 3D bajo el movimiento dado por la fusión de objetos. Se examinan los candidatos a la fusión de un *tracklet* terminado hasta una número determinado de fotogramas más allá de la extensión temporal del *tracklet* terminado. Para sus principales experimentos eligen un umbral de 20 fotogramas. La incertidumbre inherente a las estimaciones de movimiento se mide por la precisión de la alineación de las transformaciones del cuerpo rígido utilizando el error residual del ajuste no lineal. Se define como una Trusted Motion Region (TMR) como el conjunto de transformaciones de movimiento contiguas más cercanas al final de un *tracklet* que están todas por debajo de un umbral de error residual. Se mapean cada una de las transformaciones en la TMR de coordenadas globales a coordenadas objeto-céntricas que están centradas en el centro del objeto actual. Esta parametrización de movimiento centrada en el objeto codifica la misma transformación que la original, pero ahora está centrada en el centro del objeto en movimiento. Esto permite promediar significativamente estas transformaciones, así como extrapolarlas en el futuro, siempre en relación con la estimación actual del centro del objeto.

Se utiliza la ubicación mediana del conjunto de puntos 3D filtrados como el centro del objeto. También se modela la incertidumbre de la posición del objeto a partir de la estimación estereoscópica. La ubicación del objeto en cada paso de tiempo se modela mediante una distribución normal multivariante.

La coherencia del movimiento 3D entre dos *tracklets* se calcula extrapolando el movimiento 3D de ambos *tracklets* entre sí utilizando la transformación relativa media de las TMR en todos los pasos de tiempo entre las TMR de los dos *tracklets*, incluyendo el último fotograma de la TMR anterior y el primer fotograma de la última. A partir de estas estimaciones de posición 3D y sus incertidumbres, la coherencia del movimiento 3D de dos *tracklets* viene dada por la distancia media de Mahalanobis de estas estimaciones de posición 3D ponderada por sus respectivas incertidumbres de las covarianzas. Si el movimiento extrapolado para un *tracklet* no puede estimarse de forma robusta debido a la falta de una TMR, se determina la consistencia del movimiento 3D extrapolando sólo uno hacia el otro y calculándolo en el paso de tiempo del último fotograma del *tracklet* para el que no tenemos estimación de movimiento. Si ambos *tracklets* no tienen una estimación de movimiento robusta, entonces se asume que ambos son estacionarios. Se fusionan dos *tracklets* si la consistencia del movimiento 3D es superior a un umbral.

Para cada fotograma entre dos fusionados, se determina si el objeto es visible y no se detectó, o si está completamente ocluido. En primer lugar, se estima un *Bounding Box* 3D para cada *box*. Se utiliza el centro del *Bounding Box* y se asumen unas dimensiones fijas para los peatones y los coches, dadas por

la anchura/altura/longitud de los recuadros 3D en el conjunto de entrenamiento KITTI. Esto simplifica la estimación de los recuadros 3D, y resulta una localización adecuada para producir segmentaciones. Se establece la orientación del *Bounding Box* a la dirección del movimiento si el objeto sufre un movimiento significativo. En caso contrario, se establece la orientación en la dirección del vector propio con el mayor valor propio a vista de pájaro para todos los puntos 3D del objeto en el paso de tiempo actual.

Se proyecta este *Bounding Box* 3D en el espacio de la imagen como un *Bounding Box* 2D y se promedia con las esquinas del *Bounding Box* del fotograma anterior deformadas por el vector de flujo óptico medio de todos los puntos del *Bounding Box*. Se ejecuta BB2SegNet para obtener una máscara de segmentación para esta caja. Se comprueba la validez de esta máscara de segmentación tomando los valores de profundidad de los puntos dentro de la nueva máscara y proyectándolos de nuevo en coordenadas mundiales 3D. Si los puntos 3D de la nueva máscara están lo suficientemente cerca del centro del *Bounding Box* 3D, la máscara pasa la comprobación de consistencia. Esta comprobación determina si el objeto ha sido ocluido, ya que si está ocluido los puntos pertenecientes a la nueva máscara estarán significativamente por delante del *Bounding Box* 3D estimado. Esto se utiliza para rellenar las detecciones que faltan, sin introducir muchos falsos positivos. Esto también se ejecuta al principio y al final de los *tracks* completos, donde se aplica este procedimiento a cada fotograma hasta que se llega a un fotograma en el que falla la comprobación de consistencia, el objeto se mueve fuera del campo de visión de la cámara o se llega al final de la secuencia de vídeo.

3.6.1 Principales resultados obtenidos por los autores

Como métrica de evaluación, adoptan la métrica CLEARMOT, que es el estándar de KITTI MOT. Para el seguimiento de la segmentación adoptan una versión de estos adaptada para las máscaras [1], que tiene en cuenta la precisión de la segmentación incorporando el IoU de las máscaras predichas y de las verdaderas en la puntuación. Para las segmentaciones utilizan la versión IDS de [1]. Para los *bounding boxes* presentan la versión original de KITTI como IDS, aunque se ha señalado a menudo que esta que esta definición no tiene en cuenta correctamente los interruptores ID. Por lo tanto, también presentan resultados en el de validación utilizando la definición de [1], que etiqueta como IDS* y MOTA*, respectivamente. Esta definición cuenta los cambios de ID no sólo cuando la etiqueta ID cambia entre dos fotogramas contiguos, sino también cuando hay un hueco entre un cambio de ID (oclusiones).

En la tabla 3.6 se presentan los resultados de MOTSFusion para el seguimiento de la segmentación en el conjunto de validación de KITTI MOTS. Comparan el uso de tres combinaciones diferentes de detectores y métodos de segmentación. Utilizan tanto las detecciones como las segmentaciones de [1] para una comparación justa con el estado del arte anterior. También utilizan un detector más potente [94] y un método de segmentación más potente [85], para compararlo con el estado del arte anterior del rastreador de *bounding boxes* [62] con máscaras de segmentación añadidas. También se indican los resultados con las detecciones de [1] con las segmentaciones de [85]. Sólo se evalúan los coches cuando utilizan las detecciones de [94]. Utilizando las segmentaciones de [85] en lugar de las de [1], sMOTSA aumenta de 78,2 a 82,8 para los coches y de 50,1 a 59,4 para los peatones (4,6 y 9,3 puntos porcentuales). Utilizando las detecciones de [94] en lugar de [1], sMOTSA aumenta otros 2,9 puntos porcentuales, pasando de 82,8 a 85,7 para los coches.

Presentan resultados de ablación que muestran cómo su seguimiento basado en la reconstrucción 3D mejora los *tracklets* iniciales (MOTSFusion (2D)). En las mejores configuraciones, con fusión de *tracklets* basada en la reconstrucción y la recuperación de la detección de faltas (MOTSFusion) reducen el IDS de 61 a 31 para los coches y de 55 a 35 para los peatones (49 % y 36 % de mejora relativa respectivamente), y reducen el número de FNs de 386 a 364 para los coches y de 814 a 784 para los peatones, mientras

que los FPs aumentan sólo marginalmente de 37 a 44 para los coches y de 94 a 99 para los peatones. Se observan resultados similares en todas las configuraciones, lo que supone un aumento de sMOTSA de entre 0,4 y 0,7 puntos porcentuales. Esto supone una mejora significativa en la capacidad de seguimiento a largo plazo.

Se muestra que en el conjunto de validación MOTSFusion supera a TrackR-CNN con las mismas detecciones y segmentaciones en 2 puntos porcentuales para los coches (78,2 frente a 76,2) y 3,3 para los peatones (50,1 frente a 46,8) en sMOTSA, y tiene 61 % (36 frente a 93) y 56 % (34 frente a 78) menos de IDS para coches y peatones, respectivamente. Si se compara con Beyond-Pixels con segmentaciones añadidas de BB2SegNet, se mejora de 84,9 a 85,7 en sMOTSA y reducen el número de IDS de 97 a 31. Esto supone menos de un tercio de los IDS. Utilizando las segmentaciones de [1], este método supera a [62] en 1,3 puntos porcentuales en sMOTSA (78,2 frente a 76,9), aunque [62] utiliza el detector RRC más potente. También supera significativamente a otros métodos de seguimiento [61] y [18] que utilizan las mismas detecciones y segmentaciones.

Tracking	Det. & Seg.	Speed	Cars					Pedestrians				
			sMOTSA	MOTSA	IDS	FP	FN	sMOTSA	MOTSA	IDS	FP	FN
MOTSFusion	RRC[94]+BB2SegNet[85]	0.44	**85.7**	**94.5**	**31**	44	364	-	-	-	-	-
MOTSFusion(2D)	RRC[94]+BB2SegNet[85]	**0.14**	85.2	94.0	61	**37**	386	-	-	-	-	-
BePix[62]	RRC[94]+BB2SegNet[85]	0.36	84.9	93.8	97	61	**337**	-	-	-	-	-
MOTSFusion	TrRCNN[1]+BB2SegNet[85]	0.44	**82.8**	**90.5**	**51**	51	**661**	**59.4**	**72.6**	**35**	99	**784**
MOTSFusion (2D)	TrRCNN[1]+BB2SegNet[85]	**0.14**	81.9	89.6	102	**41**	695	58.2	71.2	55	**94**	814
MOTSFusion	TrRCNN[1]+TrRCNN[1]	0.44	**78.2**	**90.0**	**36**	94	673	**50.1**	**68.0**	**34**	181	855
MOTSFusion (2D)	TrRCNN[1]+TrRCNN[1]	**0.14**	77.5	89.2	85	**86**	699	48.9	66.6	53	**178**	886
BePix[62]	RRC[94]+TrRCNN[1]	0.36	76.9	89.7	88	280	**458**	-	-	-	-	-
TrRCNN[1]	TrRCNN[1]+TrRCNN[1]	0.50	76.2	87.8	93	134	753	46.8	65.1	78	267	**822**
CIWT[61]	TrRCNN[1]+TrRCNN[1]	0.28	68.1	79.4	106	333	1214	42.9	61.0	42	401	863
CAMOT[18]	TrRCNN[1]+TrRCNN[1]	0.76	67.4	78.6	220	172	1327	39.5	57.6	131	198	1090

Tabla 3.6: Resultados MOTSFusion en KITTI MOTS

MOTSFusion funciona con cualquier entrada de profundidad, pero utiliza estéreo para los principales experimentos. MOTSFusion funciona de forma similar utilizando LIDAR como cuando se utiliza el estéreo. Esto demuestra que su método es robusto a las imprecisiones presentes en la profundidad estereoscópica, y que puede manejar con éxito la entrada dispersa.

MOTSFusion se ejecuta en 0,44 segundos por fotograma, incluido el preprocesamiento de la entrada, utilizando un PC de sobremesa con una CPU Intel Core i7-5930K y una GPU GTX 1080 Ti. La parte más lenta del proceso (0,16 segundos) se encuentra en el ajuste de las transformaciones homogéneas para la fusión de objetos en 3D. Podría acelerarse fácilmente en el futuro utilizando una implementación de optimización por mínimos cuadrados en la GPU.

En comparación con otros rastreadores con el mismo hardware, TrackR-CNN tarda 0,50 segundos por fotograma y Beyond-Pixels tarda 0,30 segundos por fotograma (más 0,06 segundos para la generación de máscaras para MOTS). Por lo tanto, este algoritmo es capaz de funcionar con una eficiencia similar y con un rendimiento mucho mayor, especialmente para el seguimiento a largo plazo. Este método sería más rápido cuando se utiliza LIDAR o GPS como entrada porque la profundidad y/o el movimiento del *ego-motion* provendrían directamente de un sensor.

3.7 PointTrack

Los métodos actuales de MOTS no suelen abordar la forma de extraer las características de instancia de los segmentos. Aunque los métodos actuales de MOTS adoptan redes de segmentación avanzadas para extraer características de la imagen, no consiguen aprender incrustaciones de instancias discriminatorias,

que son esenciales para una asociación de instancias sólida. Esto da lugar a un rendimiento de seguimiento limitado.

En [32] se propone un método muy eficaz para el aprendizaje de incrustaciones de instancias discriminativas en segmentos mediante la división de la representación de imágenes compactas en nubes de puntos 2D no ordenadas. Este método está inspirado en PointNet [95], que permite la agregación de características directamente a partir de nubes de puntos 3D con formato irregular. Concretamente, para cada instancia, se construyen dos nubes de puntos separadas para el segmento del primer plano y el área circundante, respectivamente. En cada nube de puntos, se propone además combinar diferentes modalidades de características puntuales para realizar una incrustación de instancias unificada y consciente del contexto. Para permitir la utilidad práctica de MOTS, mejoran el método de segmentación de instancias de una etapa SpatialEmbedding [96] para la coherencia temporal y construyen un nuevo marco MOTS llamado PointTrack. Además, dado que KITTI MOTS tiene la densidad de instancias limitada y no tiene suficientes escenas abarrotadas, construyen un conjunto de datos MOTS más difícil, llamado APOLLO MOTS, basado en el conjunto de datos ApolloScape [71]. APOLLO MOTS tiene un número de fotogramas similar al de KITTI MOTS pero tiene dos veces más *tracks* y anotaciones de coches. La densidad media de coches de APOLLO MOTS es de 5,65, mucho mayor que la de KITTI MOTS (3,36). Además, como el seguimiento se vuelve más difícil cuando los coches se superponen, se cuenta el número de coches abarrotados tanto para APOLLO MOTS como para KITTI MOTS. Se considera que un coche está abarrotado si y sólo si su segmento es adyacente con cualquier otro coche. APOLLO MOTS tiene 2,5 veces más coches abarrotados que KITTI MOTS.

Las evaluaciones realizadas en tres conjuntos de datos muestran que PointTrack supera a todos los de los métodos MOTS anteriores con un amplio margen (véase Tabla 3.7). PointTrack puede reducir los cambios de *id* y se generaliza bien en la extracción de instancias (véase subfigura derecha de la Figura 3.20). El marco que proponen consigue, en primer lugar, un rendimiento casi en tiempo real y supera a todos los métodos de vanguardia, incluidos los métodos en KITTI MOTS con un amplio margen (véase la subfigura izquierda de la Figura 3.20).

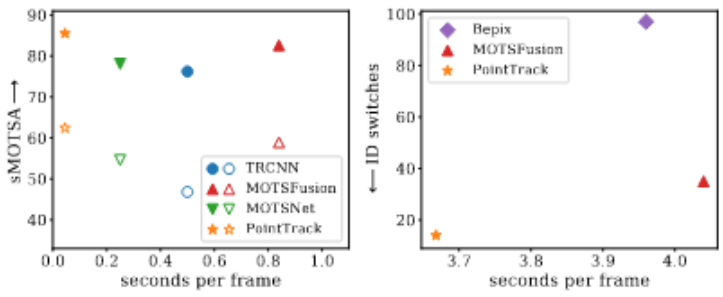


Figura 3.20: Comparación PointTrack y los métodos MOTS de última generación en sMOTSA (izquierda) y en los cambios de identificadores (derecha). En la subfigura de la izquierda, los símbolos rellenos y los símbolos huecos denotan los resultados para coches y para peatones, respectivamente. En la subfigura de la derecha, todos los métodos realizan el seguimiento sobre el mismo resultado de segmentación, que tarda 3,66s [32].

En la Figura 3.21 se puede observar una visión general de PointTrack. Para una instancia C con su segmento C_s y su rectángulo circunscrito más pequeño C_b, amplían C_b a $\hat{C}_b$ extendiendo su borde en las cuatro direcciones (arriba, abajo, izquierda y derecha) por un factor de escala k ($k = 0,2$ por defecto). Tanto C_s como $\hat{C}_b$ se visualizan en verde oscuro en la esquina inferior izquierda de la Figura 3.21. Entonces, consideran el segmento de primer plano como una nube de puntos 2D y lo denotan como F. De forma similar, consideran la otra área en $\hat{C}_b$ como la nube de puntos del entorno y la denotan como

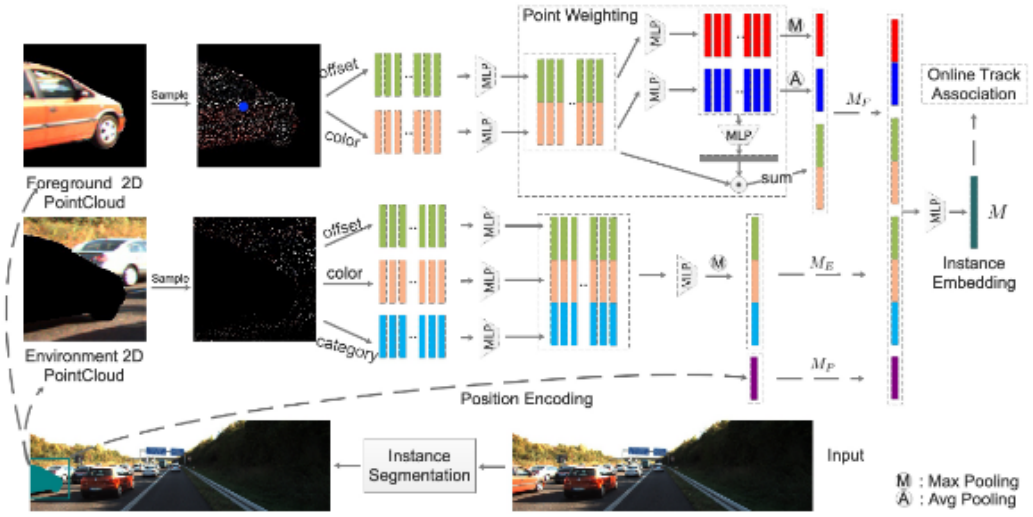


Figura 3.21: Visión general de PointTrack. Para una imagen de entrada, PointTrack obtiene segmentos de instancia mediante una red de segmentación de instancia. A continuación, PointTrack considera el segmento y su entorno circundante como dos nubes de puntos 2D y aprende características sobre ellos por separado. MLP significa Percepción multicapa con *Leaky* ReLU [32].

E. Cada punto dentro de $\hat{C}_b$ tiene un espacio de datos de seis dimensiones (u, v, R, G, B, C) que contiene la coordenada (u, v) en el plano de la imagen, el color del píxel (R, G, B), y a qué clase C pertenece el píxel.

Para la nube de puntos de primer plano F, toman una muestra aleatoria uniforme de puntos N_F ($N_F = 1000$ por defecto) para la extracción de características. Como se muestra en la Figura 3.21, los puntos N_F son suficientes para cubrir uniformemente una instancia relativamente grande. Para la nube de puntos del entorno E, se seleccionan aleatoriamente puntos N_E ($N_E = 500$ por defecto). La coordenada del punto de primer plano F_i se indica como (u_i^F, v_i^F) y la coordenada del punto de entorno E_i se indica como (u_i^E, v_i^E). El punto central $P(u_c^F, v_c^F)$ se calcula promediando las coordenadas de los puntos de primer plano seleccionados $\{F_i | i = 1, ..., N_F\}$ en el plano de la imagen. P se resalta en azul en la nube de puntos de primer plano (véase la Figura 3.21).

PointTrack aprende incrustaciones de instancia consistentes a partir del aprendizaje de una serie de modalidades de datos: *Offset*, Color, Categoría y Posición. Los valores de *Offset* se formulan como vectores desde el centro de la instancia P hasta ellos mismos, representando las ubicaciones relativas dentro del segmento. Los vectores de *Offset* de los puntos del primer plano aportan información sobre la escala y la forma de las instancias. Se definen los datos de *Offset* de cada punto de primer plano F_i y de cada punto del entorno E_i como:

$$O_{Fi} = (u_i^F - u_c^F, v_i^F - v_c^F), O_{Ei} = (u_i^E - u_c^E, v_i^E - v_c^E) \tag{3.10}$$

Las características de apariencia discriminatoria pueden aprenderse a partir de los puntos del primer plano y la distribución del color circundante puede aprenderse a partir de los puntos del entorno. Los datos de color son fundamentales para la asociación precisa de instancias. Los datos de color se formulan como:

$$C_{Fi} = (R_i^F, C_i^F, B_i^F), C_{Ei} = (R_i^E, C_i^E, B_i^E) \tag{3.11}$$

Se codifican todas las etiquetas de clase semántica para incorporar contexto ambiental en las características de los puntos, incluyendo la clase de fondo (suponiendo que las clases Z incluyen el fondo) en vectores de longitud fija de un solo punto $\{H_j|j = 1, ..., Z\}$. El vector categoría de un solo punto también se recoge para la extracción de características. Suponiendo que E_i pertenece a la categoría C_i, los datos de la categoría se formulan como:

$$Y_{Ei} = H_{Ci}, C_i \in [1, Z] \tag{3.12}$$

Cuando la instancia actual es adyacente a otras instancias, para E_i que se encuentra en las instancias cercanas, los datos de categoría Y_{Ei} junto con los datos de desplazamiento O_{Ei} le indican a PointTrack tanto la posición relativa como la clase semántica de las instancias cercanas, que sirven como pistas fuertes para la asociación de instancias. Las tres modalidades de datos anteriores se centran en la extracción de características alrededor de C_b independientemente de la posición de C_b en el plano de la imagen. Por ello, se codifica la posición de C_b en la incrustación de posición M_P. Se incrusta la posición de C_b (4-dim) en un vector de alta dimensión (64-dim) para facilitar el aprendizaje mediante el cálculo de funciones coseno y seno de diferentes longitudes de onda. Basándose en las cuatro modalidades de datos anteriores, PointTrack aprende las incrustaciones del primer plano M_F y las incrustaciones del entorno M_E en ramas separadas. Como se muestra en la Figura 3.21, las incrustaciones del entorno M_E se aprenden fusionando primero (O_E, C_E, Y_E) para todos los E_i y aplicando después la operación de agrupación máxima a las características fusionadas. Para la nube de puntos de primer plano F, M_F se aprende fusionando (O_F, C_F).

Figura 3.22: Visualización de los puntos críticos. Los puntos rojos y amarillos representan los puntos críticos del primer plano y los puntos críticos del entorno, respectivamente [32].

Basándose en la intuición de que los puntos más prominentes deberían tener pesos más altos para diferenciar las instancias, y otros puntos también deberían ser considerados, pero con pesos más bajos, se introduce la capa de ponderación de puntos para ponderar activamente todos los puntos de primer plano y sumar las características de todos los puntos. La capa de ponderación de puntos aprende a resumir las características del primer plano aprendiendo a ponderar los puntos. Las visualizaciones demuestran que la capa de ponderación de puntos aprende a dar mayor peso a las zonas informativas. Después, como se muestra en la Figura 3.22, M_F , M_E , y las incrustaciones de posición M_P se concatenan para predecir las incrustaciones de instancia final M como:

$$M = \mathbf{MLP}(M_F + M_E + M_P) \tag{3.13}$$

donde + representa la concatenación y MLP denota percepción multicapa.

Para cada caso, para validar la consistencia temporal de los puntos críticos, se seleccionan *crops* de tres fotogramas consecutivos. En el caso de los puntos en primer plano, los puntos con un 10 % de pesos superiores predichos por la capa de ponderación de puntos se representan en rojo. Los puntos críticos de primer plano se reúnen alrededor de los cristales de los coches y alrededor de las luces de los coches. Se cree que los desplazamientos de estos puntos son esenciales para aprender la forma y la postura del vehículo. Además, sus colores son importantes para perfilar la apariencia de la instancia y la distribución de la luz. PointTrack mantiene la consistencia de la ponderación puntos en fotogramas consecutivos incluso cuando se ocluyen diferentes partes (la segunda y el quinto de la primera fila de la Figura 3.22), o cuando el coche se mueve hacia el límite de la imagen (el cuarto de la primera fila). La consistencia en la ponderación de puntos a través de los fotogramas muestra la eficacia de la capa de ponderación de puntos. Para los puntos de entorno, se visualizan los cinco puntos más críticos en amarillo. Estos puntos se seleccionan obteniendo primero el tensor con un tamaño de $256 * N_E$ antes de la capa de agrupación máxima en la rama de entorno y recogiendo después el índice con el valor máximo para todas las 256 dimensiones. Entre estos 256 índices, se seleccionan los puntos pertenecientes a los cinco índices más comunes. Al combinar los datos de la Categoría con los datos del *Offset*, los puntos del entorno proporcionan fuertes pistas de contexto para la asociación de instancias. La distribución de los puntos críticos del entorno valida que PointTrack aprende características de contexto discriminativas a partir de los puntos del entorno.

Para obtener el resultado final del rastreo, es necesario realizar una asociación de instancias basada en las similitudes. Dado el segmento C_{si} y el segmento C_{sj}, y sus incrustaciones M_i y M_j, la similitud S se formula como sigue:

$$S(C_{si}, C_{sj}) = -D(M_i, M_j) + \alpha * U(C_{si}, C_{sj}) \tag{3.14}$$

donde D denota la distancia euclidiana y U representa la máscara IoU. α se establece en 0,5 por defecto. Si una pista activa no se actualiza para los últimos β fotogramas, se termina esta pista automáticamente. Para cada fotograma, se calcula la similitud entre las últimas incrustaciones de todas las pistas activas y las incrustaciones de todas las instancias en el fotograma actual según la Ecuación 3.14. Se establece un umbral de similitud γ para la asociación de instancias y sólo se permite la asociación de instancias cuando la similitud es mayor que γ. Se utiliza el algoritmo húngaro para realizar la correspondencia de instancias. Tras la asociación de instancias, los segmentos no asignados iniciarán nuevas pistas. Por defecto, β y γ se fijan en 30 y -8,0 respectivamente.

PointTrack se basa en un método de segmentación de instancias de una etapa llamado SpatialEmbedding [96]. SpatialEmbedding realiza la segmentación de instancias sin propuestas de *Bounding Box*, por lo que se ejecuta mucho más rápido que los métodos de dos etapas. Como se muestra en la Figura 3.23, SpatialEmbedding sigue una estructura de codificador-decodificador con dos decodificadores separados: (i) el decodificador *seed*; (ii) el decodificador *inst*. Dada una imagen de entrada I^T en el momento T, el decodificador *seed* predice *seed maps* S^T para todas las clases semánticas. Además, el decodificador *inst* predice un *sigma map* que denota el margen de agrupaciones por píxel y un *Offset map* que representa el vector que apunta al centro de instancia correspondiente. Después, los centros de instancia se muestrean a partir de S^T y los píxeles se agrupan en segmentos según el margen de agrupación aprendido para cada instancia. Cuando se aplica a MOTS, al estudiar los casos de fallo de segmentación, se encuentra que las predicciones del *seed maps* no son consistentes entre fotogramas consecutivos, lo que da lugar a muchos

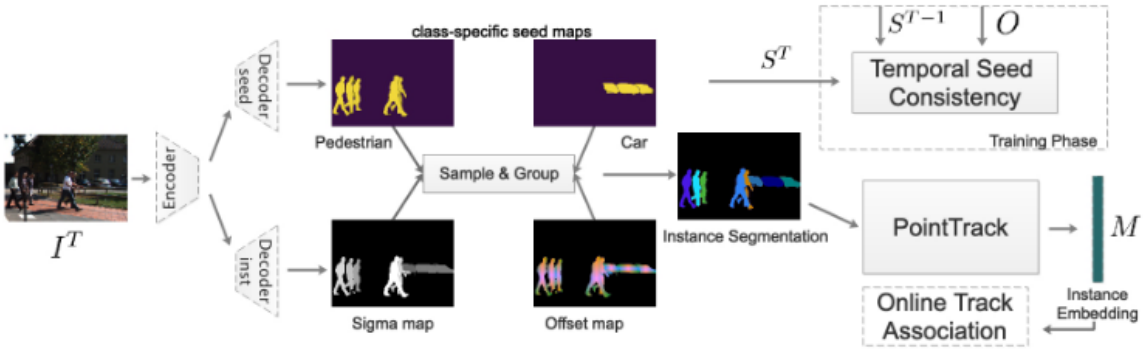


Figura 3.23: SpatialEmbedding [32].

falsos positivos y falsos negativos. Por lo tanto, se introduce la pérdida de consistencia temporal en la fase de entrenamiento para mejorar la calidad de la predicción del *seed maps* de la siguiente manera. En primer lugar, también se introduce la imagen de entrada I^{T-1} en el momento $T-1$ en SpatialEmbedding para predecir los *seed maps* S^{T-1}. A continuación, el flujo óptico O entre I^{T-1} y I^T es estimado por Volumetric Correspondence Networks (VCN)3 [97]. Posteriormente, se sintetizan los *seed maps* deformados $\hat{S}^T = O(S^{T-1})$ explotando O para deformar S^{T-1}. La pérdida de consistencia temporal se formula como:

$$L_{tc} = \frac{1}{N}\sum_{i}^{N} ||\hat{S}_i^T - S_i^T||^2 \tag{3.15}$$

donde N es el número de píxeles de primer plano e i denota el i-ésimo píxel de primer plano.

3.7.1 Principales resultados obtenidos por los autores

PointTrack se evalúa en tres conjuntos de datos: KITTI MOTS [35], MOTSChallenge [21] y su APOLLO MOTS. Los resultados se proporcionan en el conjunto de pruebas oficial KITTI MOTS y como métricas de evaluación, se centran en sMOTSA, MOTSA y en los IDS.

Debido al límite de datos de entrenamiento en KITTI MOTS (sólo 1704 fotogramas contienen Peatón donde sólo 1957 máscaras son anotadas manualmente), la red de segmentación se pre-entrena en el conjunto de datos KITTI INStance (KINS) [98]. Después, SpatialEmbedding se ajusta en KITTI MOTS con la propuesta de pérdida de consistencia de la *seed* durante 50 épocas a una tasa de aprendizaje de $5 \cdot 10^{-6}$. Se construye una base de datos de instancias a partir del conjunto de entrenamiento extrayendo todos los $\hat{C}_b$ de todos los *track ids*. Muestrean D *track ids* como un lote, cada uno con tres *crops*. Estos tres recortes se seleccionan de tres fotogramas igualmente espaciados en lugar de tres fotogramas consecutivos para aumentar la discrepancia *intra-track-id*. El espacio entre fotogramas se elige aleatoriamente entre 1 y 10.

En KITTI MOTS, PointTrack tarda 0,037s por fotograma para la segmentación de la instancia, 8ms por fotograma para el seguimiento, y 3ms por instancia para la extracción de incrustaciones. En el caso de los Coches, el detector que utiliza MOTSFusion consume mucho tiempo y tarda 3,6s por fotograma en realizar la detección. Aunque el seguimiento de objetos se basa exclusivamente en imágenes 2D con una red de segmentación de instancias ligera, PointTrack consigue un rendimiento comparable al del método de seguimiento 3D MOTSFusion (un 0,3 % más que MOTSA) con una mejora significativa de la velocidad (0,045s VS. 4,04s). En el caso de los Peatones, PointTrack supera los enfoques actuales en un 3,5 % y un 5,4 % en sMOTSA y MOTSA respectivamente. Aunque sólo se observan pequeñas mejoras respecto

a MOTSFusion para los Coches en el conjunto de validación de KITTI MOTS, PointTrack supera a MOTSFusion por amplios márgenes en el conjunto de pruebas oficial (véase las Tablas 3.2 y 3.1), lo que demuestra la buena capacidad de generalización. Además, cuando se elimina la pérdida de consistencia temporal (véase la penúltima fila de la Tabla 3.7), las caídas de rendimiento son observables (un 2,6 % y un 0,5 % en sMOTSA para Coches y Peatones, respectivamente). Este demuestra la eficacia de la pérdida de consistencia temporal.

Type	Method	Det. & Seg.	Speed	Cars			Pedestrians		
				sMOTSA	MOTSA	IDS	sMOTSA	MOTSA	IDS
2D	TRCNN [1]	TRCNN	0.5	76.2	87.8	93	46.8	65.1	78
3D	BePix [62]	RCC+TRCNN	3.96	76.9	89.7	88	-	-	-
2D	MOTSNet [99]	MOTSNet	-	78.1	87.2	-	54.6	69.3	-
3D	MOTSFusion [31]	TRCNN+BS	0.84	82.6	90.2	51	58.9	71.9	36
3D	BePix [62]	RCC+BS	3.96	84.9	93.8	97	-	-	-
3D	MOTSFusion [31]	RCC+BS	4.04	**85.5**	94.6	35	-	-	-
2D	PointTrack	PointTrack	**0.045**	**85.5**	**94.9**	**22**	**62.4**	**77.3**	**19**
2D	PointTrack (without TC)	PointTrack	**0.045**	82.9	92.7	25	61.4	76.8	21
2D	PointTrack (on Bbox)	PointTrack	**0.045**	85.3	94.8	36	61.8	76.8	36

Tabla 3.7: Resultados Point Track en KITTI MOTS

En la última fila de la Tabla 3.7 se ha ignorado C_s y en su lugar se han muestreado puntos dentro de C_b. Los puntos $N_E + N_F$ se muestrean aleatoriamente y se elimina la rama de la red para la incrustación del entorno. Para este caso, los IDS de Coches y Peatones aumentan considerablemente, y también sMOTSA en el caso de los Peatones. El aumento del IDS demuestra que los segmentos son importantes para mejorar el rendimiento del seguimiento. Además, la diferencia entre el descenso del rendimiento de los Coches y el de los Peatones demuestra que los segmentos son más eficaces para mejorar el rendimiento del seguimiento de los vehículos no rígidos, en los que la extracción de características a nivel de *Bounding Box* introduce más ambigüedades.

Cuando se aplica a los mismos resultados de segmentación en diferentes conjuntos de datos, PointTrack puede reducir eficazmente el IDS. La reducción constante del IDS en diferentes conjuntos de datos y diferentes resultados de segmentación demuestran la eficacia de PointTrack.

Para examinar el impacto en el rendimiento de las cuatro modalidades, realizan pruebas eliminándolas. Como se muestra en la Tabla 3.8, la mayor caída de rendimiento se produce cuando se eliminan los datos de Color. Por el contrario, la caída del rendimiento es mínima cuando se eliminan los datos de Posición. Esta diferencia en el rendimiento demuestra que PointTrack se centra más en las características de la apariencia y del entorno y se basa menos en la Posición del *Bounding Box* para asociar las instancias, lo que lleva a un mayor rendimiento de seguimiento y a un IDS mucho menor que los enfoques anteriores.

				Cars			Pedestrians		
Color	Offset	Category	Position	sMOTSA	MOTSA	IDS	sMOTSA	MOTSA	IDS
✓	✓	✓	✓	**85.51**	**94.93**	**22**	**62.37**	**77.35**	**19**
x				83.65	93.08	171	61.15	76.13	60
	x			85.32	94.74	37	62.16	77.14	26
		x		85.33	94.40	38	62.13	77.11	27
			x	85.35	94.77	35	62.31	77.29	21

Tabla 3.8: Impacto en el rendimiento de las diferentes modalidades

En las Tablas 3.2 y 3.1 se muestran los resultados de la evaluación en el conjunto de pruebas oficial de KITTI, donde PointTrack ocupa actualmente décimo quinto lugar (en el momento del artículo era el primer lugar). En MOTSA, PointTrack supera a MOTSFusion en un 6,8 % para los Coches y en un 3,6 % para los Peatones.

3.7.2 PointTrack ++

PointTrack++ [78] amplía notablemente el marco PointTrack ofreciendo tres importantes mejoras. En primer lugar, basándose en la observación de que la mala predicción del *seed map* da lugar a muchos falsos positivos y falsos negativos, se sustituye la rama del *seeds map* por la rama de la segmentación semántica y se considera la confianza de la clase semántica como la puntuación de la *seed* para la selección de píxeles en la inferencia. Por lo tanto, se cambia el decodificador de *seeds* por el decodificador de segmentación semántica y se introduce la pérdida focal para tratar el desequilibrio de clases por píxel.

En segundo lugar, para crear escenas más pobladas para el entrenamiento de la segmentación de instancias, se introduce la estrategia de copiar y pegar copiando instancias con luminosidad similar para cubrir instancias en las imágenes de entrenamiento. De esta forma, mejora la calidad de la segmentación, especialmente para los Peatones. Afortunadamente, las anotaciones precisas de las instancias de los píxeles proporcionadas por MOTS hacen que copiar y pegar sea conveniente y eficaz. Para ello, se construye una base de datos de Peatones extrayendo los píxeles y segmentos de todos los Peatones. A continuación, para las instancias de cada imagen de entrenamiento, se colocan aleatoriamente Peatones con una luminosidad similar de la base de datos en una posición razonable. Las imágenes de entrenamiento realistas resultantes tienen escenas más pobladas y ayudan a Point Track++ a lograr una mayor calidad de segmentación.

Por último, basándose en la intuición de que una mayor discrepancia *intra-track-id*, que es beneficiosa para el aprendizaje de las incrustaciones de primer plano, es perjudicial para el aprendizaje de las incrustaciones de entorno y de posición, se propone un entrenamiento multietapa para el aprendizaje de incrustaciones de instancia más discriminativas. El marco resultante, PointTrack++, ocupó el primer puesto en la tabla oficial de KITTI MOTS.

Siguiendo a PointTrack, la red de incrustación de PointTrack++ se entrena de extremo a extremo en lotes de diferentes *track ids*. Cada lote consta de D *track ids*, cada uno con tres *crops*. En PointTrack, estos tres *crops* se seleccionan a partir de tres fotogramas igualmente espaciados en lugar de tres fotogramas consecutivos para aumentar la discrepancia *intra-track-id* y el espacio se elige aleatoriamente entre 1 y S (fijado en 10 por defecto). Sin embargo, si el intervalo entre los fotogramas muestreados es grande, tanto el área del entorno como la posición de la misma instancia pueden cambiar tan drásticamente que resulte difícil diferenciar los distintos *track ids*. Empíricamente, cuando la red de incrustación sólo aprende en la nube de puntos 2D del entorno, establecer S a un valor mayor que 2 hace que la red de incrustación no converja. Sin embargo, cuando la red de incrustación sólo aprende en la nube de puntos 2D del primer plano, establecer S a un valor grande como 12 ayuda a conseguir un mayor rendimiento de seguimiento. Por lo tanto, se propone entrenar M_F, M_E, M_P por separado en diferentes S eliminando las otras dos incrustaciones en el entrenamiento. Después, se fijan los parámetros de las tres ramas, excepto el último MLP, y se aprende la incrustación de instancia agregada M_A añadiendo una capa MLP adicional. La incrustación de instancia final se obtiene concatenando M_F, M_E, M_P, M_A.

Todos los experimentos se llevan a cabo en un servidor GPU con Intel i9-9900X y una TITAN RTX. Como PointTrack++ puede explotar las etiquetas de segmentación de instancias a nivel de imagen para el entrenamiento, se preentrena la red de segmentación en el conjunto de datos KINS. Después, la red de segmentación se ajusta en KITTI MOTS durante 50 épocas con una tasa de aprendizaje de $5 \cdot 10^{-6}$. El factor de modulación de la pérdida focal se fija en 2,0. Durante el entrenamiento de la red de incrustación, se asigna S a 8, 2, 1, 5 para M_F, M_E, M_P, M respectivamente. Para Copiar y Pegar, la probabilidad de pegar un Peatón es de 0,2 y 0,5 para los Coches y los Peatones respectivamente. Por último, en el caso de PointTrack++, la imagen de entrada se muestrea al doble del tamaño original.

Los principales resultados en el conjunto de validación de KITTI MOTS se muestran en la Tabla 3.9, donde se compara PointTrack++ con el estado de la técnica anterior. Como la imagen de entrada

está muestreada, PointTrack++ necesita el doble de tiempo de inferencia que PointTrack. Sin embargo, se observan incrementos obvios de sMOTSA del 1,31 % y el 3,11 % para los Coches y los Peatones, respectivamente. Estas mejoras constantes demuestran la eficacia de las mejoras propuestas.

En la Tabla 3.10, se muestra el impacto de cuatro modificaciones en el rendimiento. *'2X'* significa que el muestreo de la entrada es el doble del tamaño original. *'Sem'* denota la adopción del mapa de segmentación semántica como *seed map*. *'CP'* representa el método de copiar y pegar y *'Sep'* representa el entrenamiento en varias fases de la red de incrustación. La primera fila muestra el rendimiento del PointTrack original. La aplicación de *'2X'* y *'Sem'* aporta una pequeña mejora sMOTSA (0,84 %) para los Coches. Sin embargo, se observa un gran incremento de sMOTSA del 2,15 % para los Peatones. Además, la incorporación de *'CP'* en el entrenamiento aporta un 0,52 % de mejora sMOTSA para los Peatones. Además, al entrenar por separado de la red de incrustación, PointTrack++ consigue un 0,86 % más de MOTSA.

Type	Method	Det. & Seg.	Speed	Cars			Pedestrians		
				sMOTSA	MOTSA	IDS	sMOTSA	MOTSA	IDS
2D	TRCNN	TRCNN	0.5	76.2	87.8	93	46.8	65.1	78
3D	BePix	RCC+TRCNN	3.96	76.9	89.7	88	-	-	-
2D	MOTSNet	MOTSNet	-	78.1	87.2	-	54.6	69.3	-
3D	MOTSFusion	TRCNN+BS	0.84	82.6	90.2	51	58.9	71.9	36
3D	BePix	RCC+BS	3.96	84.9	93.8	97	-	-	-
3D	MOTSFusion	RCC+BS	4.04	**85.5**	94.6	35	-	-	-
2D	PointTrack	PointTrack	**0.045**	85.5	94.9	22	62.4	77.3	**19**
2D	PointTrack++	PointTrack++	0.095	**86.81**	**95.95**	**17**	**65.51**	**81.54**	26

Tabla 3.9: Resultados Point Track ++ en KITTI MOTS

				Cars			Pedestrians		
Color	Offset	Category	Position	sMOTSA	MOTSA	IDS	sMOTSA	MOTSA	IDS
				85.5	94.9	22	62.4	77.3	19
v				86.12	94.87	19	63.78	78.28	23
v	v			86.34	95.14	19	64.65	79.27	23
v	v	v		86.37	95.09	16	65.17	81.21	23
v	v	v	v	86.81	95.95	17	65.51	81.54	26

Tabla 3.10: Impacto de las modificaciones

3.8 EagerMOT

En [76], se presenta EagerMOT, un método sencillo de seguimiento que fusiona todas las observaciones de objetos disponibles procedentes de detectores de objetos 3D y 2D, para obtener una buena interpretación de la dinámica de la escena. Mediante el uso de cámaras, este método identifica y mantiene los *tracks* en el dominio de la imagen, mientras que las detecciones 3D permiten una localización precisa de la trayectoria en 3D tan pronto como los objetos entran en el área de detección LIDAR. Esto se consigue mediante un procedimiento de asociación en dos fases. En primer lugar, se asocian las detecciones de objetos procedentes de diferentes modalidades de sensores. A continuación, se emplea una formulación de seguimiento que permite actualizar los estados de seguimiento incluso cuando sólo se dispone de pruebas parciales del objeto. De este modo, EagerMOT es robusto frente a los falsos negativos procedentes de diferentes modalidades de sensores y puede inicializar los seguimientos de objetos antes de que éstos entren en el rango de detección de profundidad.

Cuando se utiliza un detector de objetos 3D potente basado en LIDAR, el rendimiento de seguimiento

es competitivo incluso en un método sencillo basado en modelos de movimiento lineal y en la asociación de datos de dos fotogramas 3D con solapamiento. Sin embargo estos métodos son más sensibles a las superficies reflectantes y de bajo albedrío, y sólo pueden operar dentro de un rango de detección limitado debido a la escasez de la señal de entrada. Por otro lado, los métodos basados en imágenes aprovechan una señal visual rica para ganar robustez frente a las oclusiones parciales y localizar objetos con precisión de píxel en el dominio de la imagen, incluso cuando los objetos están demasiado lejos para ser localizados de forma fiable en el espacio 3D. Sin embargo, la localización en 3D de los objetos circundantes es vital en escenarios de robots móviles.

Este método es lo suficientemente versátil como para ser aplicado a varias configuraciones sensoriales diferentes, como LIDAR combinado con una cámara frontal (como se utiliza en KITTI [35]), o combinado con múltiples cámaras con frentes de visión no superpuestos (como se emplea en NuTonomy Scenes (NuScenes) [100]). Con EagerMOT, se establece un nuevo estado del arte en la prueba de referencia NuScenes 3D MOT y en la prueba de referencia KITTI para el seguimiento y la segmentación de objetos múltiples en 2D.

EagerMOT supone simplemente una plataforma móvil con una configuración sensorial calibrada equipada con un LIDAR y (posiblemente múltiples) sensores de cámara. Dado un detector de objetos pre-entrenado de objetos para ambas modalidades de sensores, este método puede ser fácilmente en cualquier plataforma móvil sin necesidad de entrenamiento adicional y supone un coste computacional adicional mínimo a la hora de tiempo de inferencia. Sus principales contribuciones son proponer un enfoque de asociación de datos simple pero efectivo que puede aprovechar una variedad de detectores de objetos diferentes, originados en modalidades potencialmente diferentes; mostrar que el enfoque puede aplicarse a una variedad de tareas 2D/3D MOT y MOTS, y en diferentes configuraciones de sensores; y realizar un análisis minucioso del método, demostrando mediante estudios de ablación la eficacia del enfoque propuesto para la asociación de datos y los resultados del estado del arte en tres puntos de referencia diferentes.

El método propuesto combina detectores de objetos LIDAR 3D complementarios que localizan con precisión los objetos en el espacio 3D, y detectores de objetos 2D que son menos susceptibles a las oclusiones parciales y y que siguen siendo fiables incluso cuando los objetos están lejos del sensor. La fusión de las pruebas de objetos en 2D y 3D durante el seguimiento es un área poco explorada. En cambio, este método trata las diferentes modalidades de sensores de forma independiente. Se rastrean los objetivos en ambos dominios simultáneamente, pero no se acoplan explícitamente sus estados 2D-3D.

A diferencia de MOTSFusion, este método se basa únicamente en las detecciones de objetos de las *Bounding Boxes* obtenidas a partir de dos modalidades de sensor complementarias y se adapta bien a diferentes entornos sensoriales.

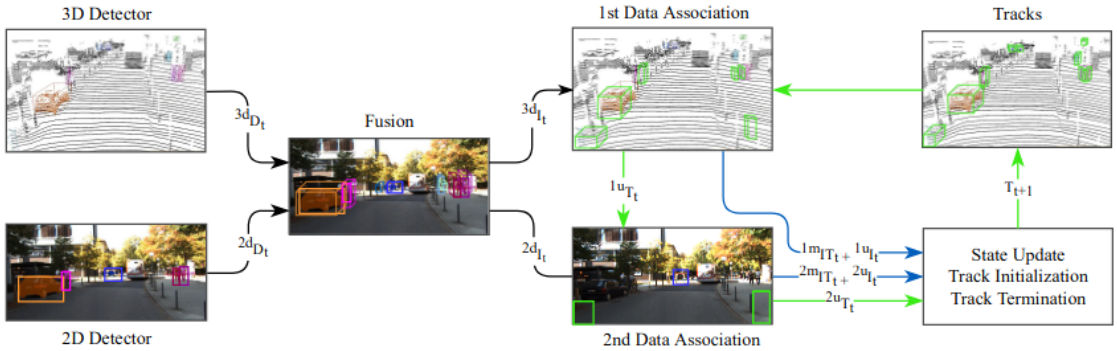


Figura 3.24: Esquema método EagerMOT

EagerMOT combina evidencias complementarias de objetos en 2D y 3D obtenidas de detectores de objetos pre-entrenados. En la Figura 3.24 se ofrece una visión general del método. Como entrada en cada fotograma, se toma un conjunto de detecciones de *Bounding Boxes* 3D ${}^{3d}D_t$ y un conjunto de detecciones 2D ${}^{2d}D_t$.

A continuación, el módulo de fusión de observaciones establece correspondencia entre los conjuntos de detecciones 2D y los conjuntos de detecciones 3D que hay a la entrada (véase Figura 3.25, es decir, trata su solapamiento 2D en el dominio de la imagen y da lugar a un conjunto de instancias de objetos fusionados $I_t = \{I_t^0, ..., I_t^i\}$. Se define el solapamiento 2D de un par ${}^{3d}D_t^i$ y ${}^{2d}D_t^i$ como la IoU entre la proyección 2D de ${}^{3d}D_t^i$ en el plano de la imagen de la cámara y ${}^{2d}D_t^i$. Se clasifican todos los posibles emparejamientos de detección por su solapamiento en orden descendente y se combinan para formar una única instancia fusionada ${}^{both}It^i$ cuando (i) su solapamiento está por encima de un umbral θ_{fusion} y (ii) aún no se ha emparejado ninguna detección 2D, o 3D. Las instancias fusionadas contienen una ubicación 3D precisa del objeto y su *Bounding Box* 2D. Las detecciones restantes (no coincidentes) se consideran observaciones parciales, que contienen información sobre sólo una de las dos modalidades.

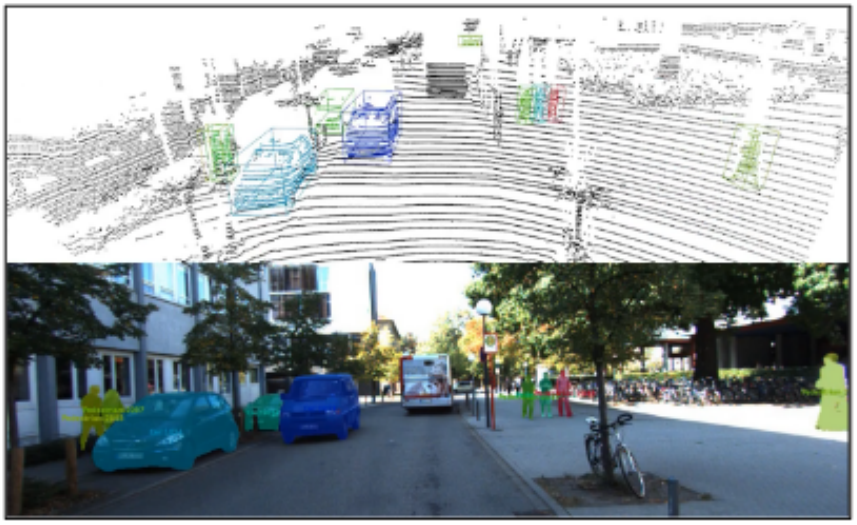


Figura 3.25: Combinación de objetos 2D y 3D en el método EagerMOT

Tras el módulo de fusión, durante cada fotograma t, las instancias fusionadas I_t entran en un modulo de asociación de dos etapas para actualizar los *tracks* existentes T_t con nueva información 3D y/o información 2D. Se mantiene el estado 2D y 3D de los *tracks* T_t en paralelo pero se tratan de forma independiente. Se representa el estado 3D de un *track* mediante un *Bounding Box* orientado a objetos 3D y un vector de velocidad posicional, mientras que un *Bounding Box* 2D representa su estado 2D. Se debe tener en cuenta que estos estados no tienen que ser observados en su totalidad para cada fotograma, los *tracks* pueden ser actualizados usando sólo la información 3D, sólo la información 2D, o ambas. Para las pistas ${}^{3d}T_t$ se mantiene adicionalmente un modelo de movimiento de velocidad constante, modelado por un filtro lineal de Kalman. Para cada nuevo fotograma $t+1$, los *tracks* existentes ${}^{3d}T_t$ predicen su ubicación en el fotograma actual, basándose en observaciones previas y estimaciones de velocidad.

En la primera etapa de asociación, se emparejan con avidez las instancias detectadas ${}^{3d}I_t$ con los *tracks* ${}^{3d}T_t$ basándose en la distancia escalada entre los *boxes* orientados de las instancias y los *boxes* orientados previstos de los *tracks*. La distancia escalada para un par de *Bounding Boxes* 3D orientados se define como la distancia Euclidea entre ellos, multiplicada por la distancia normalizada coseno normalizado entre sus vectores de orientación. De la misma forma que en el módulo de fusión, los pares de instancia-*track* que mejor coinciden (por debajo de un umbral máximo θ_{3d}) forman las ${}^{1m}IT_t = \{(I_t^i, T_t^j), ...\}$. El resto se

etiqueta como no coincidentes. Después de esta primera etapa de emparejamiento, todas las instancias de objetos detectadas en 3D deben asociarse con éxito a los *tracks* existentes o ser etiquetados como no emparejados y no participar en más emparejamientos.

En la segunda etapa de asociación de datos, se asocian con avidez las instancias ${}^{2d}I_t \setminus^{both} I_t$ a los *tracks* restantes ${}^{1u}T_t \cup^{2d} T_t$ basándose en el criterio 2D IoU. Para cada par instancia-*track*, se evalúa el solapamiento entre la instancia en el fotograma actual y la proyección 2D del *Bounding Box* 3D previsto para el *track* o el último *Bounding Box* 2D observado en el caso de que no se disponga de una predicción 3D (véase Figura 3.26. Esta etapa de asociación es idéntica a la primera, salvo que aquí se utiliza el IoU de la *box* 2D como métrica de asociación, junto con su umbral θ_{2d}. Del mismo modo, los resultados de esta etapa son un conjunto de coincidencias ${}^{2m}IT_t = \{(I_t^i, T_t^j), ...\}$, un conjunto de instancias no coincidentes, y *tracks* no coincidentes.

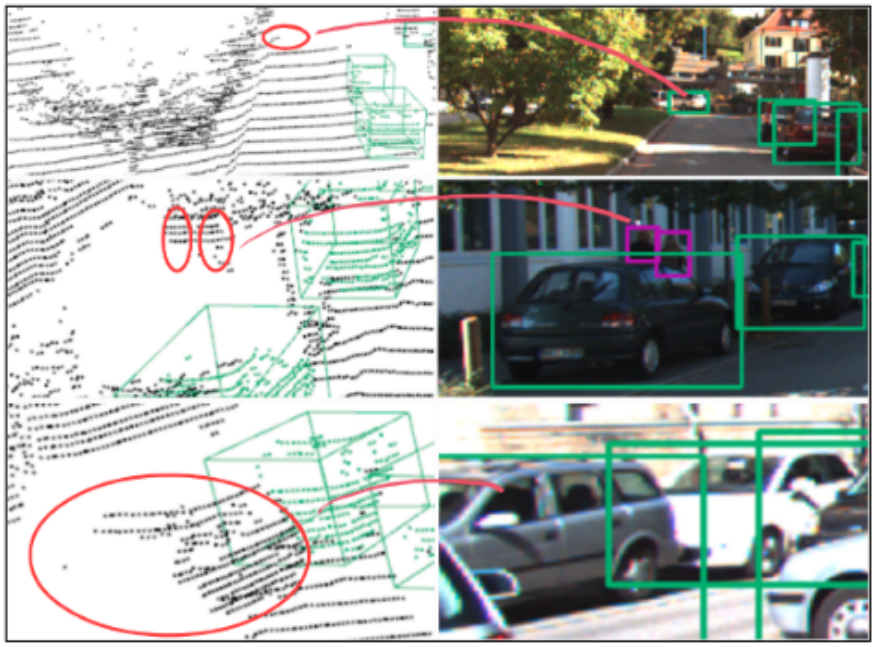


Figura 3.26: Ejemplos de objetos pasados por alto por el detector 3D pero reconocidos por el detector 2D. Desde arriba: fuera de alcance, parcialmente ocluido, fallo del detector

Se utiliza un modelo de movimiento en 3D para obtener predicciones de *Bounding Boxes* en 2D en el dominio de la imagen mediante una operación de proyección de la cámara. El modelo de movimiento no es suficiente para inicializar de forma fiable para ciertos *tracks*, lo que suele ocurrir con los objetos observados fuera del rango de detección del LIDAR. En estos casos, el movimiento aparente del *Bounding Box* suele ser insignificante, y la asociación puede realizarse únicamente en función de los *boxes* 2D observados. La adición de un modelo de predicción para el estado 2D o un modelo de apariencia (aprendido) podría utilizarse para mejorar aún más la segunda etapa de asociación y sigue siendo un trabajo futuro de este método.

Simplemente se actualiza el estado 2D (esquinas del *Bounding Box* superior izquierdo e inferior derecho) sobrescribiendo el estado anterior con el *Bounding Box* 2D recién detectado. Se modela el estado 3D de un *track* como una gaussiana multivariable y se filtran sus parámetros utilizando un filtro lineal de Kalman de velocidad constante. Cuando no se dispone de información de detección de objetos en 3D, sólo se realiza el paso de predicción del filtro de Kalman para extrapolar el estado.

Para gestionar las trayectorias de los objetos y su ciclo de vida, se emplea un sencillo conjunto de reglas. Un *track* se descarta si no se ha actualizado con ninguna instancia (ya sea 3D o 2D) en los

últimos fotogramas de Age_{max}. Como los detectores de objetos 3D no suelen ser tan fiables como los detectores basados en imágenes en términos de precisión, un *track* se considera confirmado si se asocia a una instancia en el fotograma actual y se ha actualizado con información 2D en los últimos fotogramas Age_{2d}. Por último, todas las instancias detectadas ${}^{2u}I_t$ que nunca fueron emparejadas inician nuevos *tracks*.

Esta formulación permite asociar a los *tracks* todos los objetos detectados, aunque no sean detectados ni en el dominio de la imagen ni por un sensor 3D. De este modo, este método puede recuperarse de las oclusiones cortas y mantener una ubicación 3D aproximada cuando falla uno de los detectores y, lo que es más importante puede rastrear objetos lejanos en el dominio de la imagen antes de que los objetos entren en el rango de detección 3D. Una vez que los objetos entran en el rango de detección, se puede inicializar suavemente un modelo de movimiento 3D para cada *track*.

3.8.1 Principales resultados obtenidos por los autores

Este método es evaluado con el conjunto de datos KITTI, utilizando los puntos de referencia de seguimiento multiobjeto KITTI 3D MOT, KITTI 2D MOT y KITTI MOTS. Para KITTI 2D MOT y KITTI MOTS, se utilizan los puntos de referencia oficiales y se compara con los métodos más avanzados publicados y revisados por pares.

Se utilizan las medidas de evaluación estándar de CLEAR-MOT y se centra la discusión en MOTA. Para MOTS, se informa de MOTSA y MOTSP. En los *benchmarks* KITTI 2D MOT y MOTS, además, se informa de la recientemente introducida métrica Higher Order Tracking Accuracy (HOTA). HOTA separa los aspectos de detección y seguimiento de la tarea midiendo por separado la Detection Accuracy (DetA), que evalúa el rendimiento de la detección, y la Association Accuracy (AssA), que evalúa la asociación de la detección.

En KITTI, se sigue a MOTSFusion y se utilizan detecciones 2D de RRC para coches y TrackR-CNN para peatones. Utilizan umbrales de 0,6 y 0,9 para las detecciones de RRC y TrackR-CNN, respectivamente.

Aunque se rastrean los objetos en el espacio 3D, se puede informar de los resultados de rastreo 2D proyectando *Bounding Boxes* 3D en el plano de la imagen utilizando los intrínsecos de la cámara e informando de los *Bounding Boxes* 2D alineados con el eje mínimo que encierran completamente esas proyecciones como posiciones 2D de los *tracks*. MOTS amplía el MOT con la localización precisa de los *tracks* de los objetos en los píxeles. Se puede adaptar este enfoque MOTS fácilmente pasando adicionalmente máscaras de segmentación de las instancias a los *tracks* después de la asociación de datos. En la Tabla 3.11, se informa del rendimiento de su MOTS en el conjunto de pruebas KITTI y se compara con otros métodos publicados. Como puede verse, se obtienen mejores resultados en comparación con MOTSFusion en ambas clases (+1,03 para la clase de coches y +3,61 para la de peatones) a pesar de utilizar el mismo conjunto de máscaras de segmentación 2D. Sin embargo, se observa que EagerMOT utiliza adicionalmente detecciones de objetos en 3D obtenidas del flujo LIDAR, mientras que MOTSFusion se basa en las cámaras estéreo. Este método es aplicable a una amplia variedad de configuraciones sensoriales centradas en LIDAR, a menudo empleadas en conjuntos de datos de automoción modernos, por ejemplo, NuScenes. Además, este método se ejecuta a 90 *fps* en KITTI (LIDAR + una sola cámara) en comparación con MOTSFusion a 2 *fps* (cámaras estéreo). Por último, este método establece nuevos resultados del estado del arte tanto para las clases de coches (74,66) como de peatones (57,65) en términos de HOTA. Se comporta especialmente bien en términos de AssA, confirmando una vez más que el uso de observaciones de sensores adicionales ayuda a mantener la consistencia de la pista.

	Method	HOTA	DetA	ASSA	sMOTSA	IDS	FPS
car	**Ours**	**74.66**	76.11	**73.75**	74.53	458	**90**
	MOTSFusion	73.63	75.44	72.39	74.98	**201**	2
	PointTrack	61.95	**79.38**	48.83	78.50	346	22
	TrackR-CNN	56.63	69.90	46.53	66.97	692	2
ped.	**Ours**	**57.65**	60.30	**56.19**	58.08	270	**90**
	MOTSFusion	54.04	60.83	49.45	58.75	279	2
	PointTrack	54.44	**62.29**	48.08	**61.47**	**176**	22
	TrackR-CNN	41.93	53.75	33.84	47.31	482	2

Tabla 3.11: Resultados en la prueba de referencia 2D KITTI MOTS (para las clases de coches y peatones). EagerMOT utiliza el mismo mismo conjunto de detecciones de objetos y máscaras de segmentación que MOTSFusion

En KITTI, utilizan $\theta_{fusion} = 0,01$, $\theta_{3d} = 0,01$, $\theta_{2d} = 0,3$, $Age_{max} = 3$, y $Age_{2d} = 3$ para ambas clases.

Capítulo 4

Resultados

4.1 Introducción

En este capítulo se introducirán los resultados obtenidos después de trabajar con el código aportado por cada uno de los métodos analizados en el anterior capítulo. Todos ellos se evalúan sobre el conjunto de datos KITTI MOTS. Se explica brevemente su experimento, se ejecuta el código para obtener resultados y se comparan con los resultados del código original.

4.2 Resultados obtenidos para TrackRCNN

En este primer apartado se ha configurado y ejecutado el código del proyecto TrackR-CNN para la tarea de MOTS [1], con el fin de obtener los resultados del mismo sobre la base de datos KITTI MOTS. En el artículo, los autores proponen usar su código con la siguiente configuración: Python 3.6.9 y Tensorflow 1.13.1 corriendo en una sola GPU GTX 1650. Para esta primera prueba, se ha utilizado la siguiente configuración HW/SW:

- Sistema operativo Linux Ubuntu 18.04 LTS. Se ha ejecutado el código sobre el propio sistema en modo nativo.
- Python 3.6.9.
- Tensorflow 1.13.1.
- Una sola GPU GTX 1650.

Previo a su ejecución, es necesario seguir una serie de pasos de configuración detallados a continuación:

1. Descargar el código de *Github* (`https://github.com/VisualComputingInstitute/TrackR-CNN/tree/master`). Para descargar un proyecto de *Github* existen dos opciones:
    - Descargar desde el repositorio. Navegar hasta el repositorio que se quiera clonar, hacer clic en el botón verde y generar el archivo .zip.
    - Descargar desde la consola de comandos. Mediante la consola de comandos, ubicarse en la ruta deseada para clonar. En el repositorio del proyecto, hacer clic en el botón verde y copiar la URL del documento. Usar el siguiente comando:

```
git clone <URL>
```

donde <URL> en este caso en concreto es https://github.com/VisualComputingInstitute/TrackR-CNN.git.

En caso de no tener instalado git u otro paquete, se consigue utilizando esta serie de comandos:

```
sudo apt-get update
sudo apt-get install <package>
<package> --version
```

donde <package> en este caso es git.

2. Compilar e instalar mediante comandos paquetes para Python3 como tensorflow-gpu, numpy, scipy, sklearn, pypng, opencv-python, munkres, etc. Para instalar un paquete para Python3 basta con escribir este comando:

```
pip3 install <module>
```

donde <module> es el nombre del paquete que se quiere instalar.

Un paso previo que se ha realizado para mayor comodidad es renombrar el comando pip3 a pip.

```
sudo apt-get update
sudo apt-get install python3-pip
pip3 --version
alias pip=pip3
echo alias pip=pip3 >> ~/.bashrc
source ~/.bashrc
```

De modo que la instalación de paquetes/librerías quedaría:

```
pip install <module>
```

3. Descargar imágenes e instancias de la base de datos KITTI MOTS. Las imágenes se pueden descargar en http://www.cvlibs.net/download.php?file=data_tracking_image_2.zip y las instancias puedes obtenerlas en dos formatos distintos: en formato .png (https://www.vision.rwth-aachen.de/media/resource_files/instances.zip) o en formato .txt (https://www.vision.rwth-aachen.de/media/resource_files/instances_txt.zip). En este caso se ha optado por el formato .txt. ya que son archivos más pequeños y más rápidos de leer. Cada línea de un archivo .txt de instancias está estructurada de la forma en la que se indica en la Tabla 4.2 (donde RLE significa codificación de longitud de ejecución de MS COCO).

4. Crear la estructura de carpetas definida en el código aportado para su implementación. Se asume que se tiene una carpeta /path/to/kitti_mots con subcarpetas /path/to/kitti_mots/images que contienen las imágenes de entrada (es decir existen subcarpetas /path/to/kitti_mots/images/0000,

/path/to/kitti_mots/images/0001, ...) y /path/to/kitti_mots/instances (de nuevo con subcarpetas 0000, 0001, ...) que contienen las anotaciones. Además, es necesario crear las carpetas /forwarded, /models, /summaries y /logs.

5. Descargar los modelos de pre-entrenamiento de TrackR-CNN. Estos modelos se pueden encontrar en https://omnomnom.vision.rwth-aachen.de/data/trackrcnn/. En este caso se utilizan dos capas convolucionales 3D separables en profundidad entre la red troncal y la RPN, cada una con 1024 dimensiones (*conv3d_sep2*). Estos archivos deben ser copiados a la carpeta creada /models.

6. Modificar los *scripts* de configuración. Esto es, ajustar los indicadores KITTI_segtrack_data_dir y load_init para que apunten al directorio de datos de KITTI MOTS (data/KITTI_MOTS/train/) y a la ruta del modelo preentrenado (models/converted), respectivamente. Estos parámetros se encuentran en /configs/conv3d_sep2.

Una vez realizados los pasos de configuración, se procede a ejecutar los comandos del código para obtener las detecciones y el *tracking* del modelo seleccionado. En esta primera ejecución, no se realiza ningún cambio y los pesos son los que hay por defecto. Los parámetros de configuración del modelo por defecto son los mostrados en la tabla 4.1 y se encuentran en /configs/conv3d_sep2.

Default settings: car			
tracker	reid_comp	detection_confidence_threshold_car	reid_weight_car
hungarian	euclidean	0.8469800990815324	1.0
mask_iou_weight_car	bbox_center_weight_car	bbox_iou_weight_car	association_threshold_car
0.0	0.0	0.0	0.8165986526897969
keep_alive_car	reid_euclidean_offset_car	reid_euclidean_scale_car	
4	8.810218833503743	1.0090931467228708	
Default settings: ped			
tracker	reid_comp	detection_confidence_threshold_ped	reid_weight_ped
hungarian	euclidean	0.9368820089063415	1.0
mask_iou_weight_ped	bbox_center_weight_ped	bbox_iou_weight_ped	association_threshold_ped
0.0	0.0	0.0	0.47985540892434836
keep_alive_ped	reid_euclidean_offset_ped	reid_euclidean_scale_ped	
6	9.447084376750222	1.3437965549876354	

Tabla 4.1: Parámetros iniciales del modelo de entrenamiento

Los pasos que se han seguido para obtener resultados son los siguientes:

1. Obtener las predicciones del modelo o *forwarding*. Este proceso se realiza ejecutando el siguiente comando:

```
python3 main.py configs/conv3d_sep2 "{\"task\":\"forward_tracking\",
\"dataset\":\"KITTI_segtrack_feed\",\"load_epoch_no\":5,\"batch_size
\":5,\"export_detections\":true,\"do_tracking\":false,\"video_tags
_to_load\":[\"0002\",\"0006\",\"0007\",\"0008\",\"0010\",\"0013\",\"
0014\",\"0016\",\"0018\",\"0000\",\"0001\",\"0003\",\"0004\",\"0005\
"\"0009\",\"0011\",\"0012\",\"0015\",\"0017\",\"0019\",\"0020\"]}"
```

La cadena json suministrada como argumento adicional aquí sobrescribe los ajustes del archivo de configuración. En este caso se ha decidido obtener las predicciones de todas las secuencias pero con `video_tags_to_load` se pueden especificar las que se requieran. Los resultados se escriben en el subdirectorio /forwarded. Resulta un archivo .txt para cada secuencia con líneas de la siguiente forma:

0 21 2 375 1242 Ygf<5b;0000O2O1N_;0‘D5L2N1N2N2cFFZ7=eHFT ... 000kU‘0

La estructura de cada línea sigue lo que se indica en la tabla 4.2.

Marco de tiempo	id objeto	clase id	Altura imagen	Anchura imagen	RLE
0	21 (clase id es 2, es decir, coche, y la instancia id es 1)	2 (coche)	375	1242	Ygf<5b ... 000kU‘0

Tabla 4.2: Resultados forwarding

2. Ejecutar el algoritmo de rastreo para vincular las detecciones obtenidas en el tiempo con el siguiente comando:

```
python3 main.py configs/conv3d_sep2 "{\"build_networks\":false,\"
import_detections\":true,\"task\":\"forward_tracking\",\"dataset\":\
"KITTI_segtrack_feed\",\"do_tracking\":true,\"visualize_detections\"
:false,\"visualize_tracks\":true,\"load_epoch_no\":5,\"video_tags_
to_load\":[\"0002\",\"0006\",\"0007\",\"0008\",\"0010\",\"0013\",\"
0014\",\"0016\",\"0018\"]}"
```

Los resultados se escriben en el subdirectorio /forwarded y pueden ser procesados por los scripts de mots_tools. El parámetro `visualize_tracks` se ha puesto a `true` para poder obtener las visualizaciones de los resultados del *tracking*. En la figura 4.1 se muestran 4 secuencias del vídeo 0016 visualizando el *tracking* resultante obtenido con el comando anterior.

Por último, para la evaluación de los resultados, se utiliza mots_tools, una herramienta que evalúa y visualiza los resultados de la tarea MOTS. Con su aportación, se pueden comprobar las puntuaciones obtenidas por el modelo seleccionado y compararse con las puntuaciones del artículo. Los pasos que hay que seguir para la utilización de mots_tools son:

1. Descargar mots_tools desde *Github* (`https://github.com/VisualComputingInstitute/mots_tools`).

   Para utilizar la herramienta es necesario tener instalado pycocotools. Los pasos a seguir son los siguientes:

   - Descargar cocotools desde *Github* (`https://github.com/cocodataset/cocoapi`).
   - Navegar hasta el directorio raíz del archivo y ejecutar el siguiente comando para instalar cocotools para Python:

     ```
     make coco/PythonAPI
     ```

   También es necesario comprobar que el directorio de mots_tools se encuentra en *Python path*.

2. Para evaluar el resultado del *tracking* se ejecuta el siguiente comando:

```
python3 mots_eval/eval.py <tracking_results> <gt_folder> <seqmap>
```

(a) 000019

(b) 000020

(c) 000021

(d) 000022

Figura 4.1: Visualización resultados tracking del video 0016 de TrackRCNN

donde `<tracking_results>` indica el resultado en el que se localizan los resultados del *tracking* (`./TrackR-CNN/forwarded/conv3d_sep2/tracking_data`), en `<gt_folder>` se encuentran las anotaciones *Ground Truth* (`./TrackR-CNN/data/KITTI_MOTS/train/instances`) y `<seqmap>` es un archivo .txt que contiene las secuencias que desea evaluar, en este caso las secuencias del conjunto de validación (`mots_eval/val.seqmap`). La evaluación se imprime en la consola de comandos.

3. Para visualizar el resultado del *tracking* es necesario instalar previamente el paquete FFmpeg (con el comando "pip install ffmpeg"). El comando a utilizar para la visualización es:

```
python3 mots_vis/visualize_mots.py <tracking_results> <img_folder>
<output_folder> <seqmap>
```

donde `<tracking_results>` indica el resultado en el que se localizan los resul-

tados del *tracking* (`./TrackR-CNN/forwarded/conv3d_sep2/tracking_data`), en `<img_folder>` se encuentran las imágenes del conjunto de datos KITTI MOTS (`./TrackR-CNN/data/KITTI_MOTS/train/images`), `<output_folder>` es la carpeta donde se crearán los resultados visualizados, y `<seqmap>` es un archivo .txt que contiene las secuencias que desea evaluar, en este caso las secuencias del conjunto de validación (`mots_eval/val.seqmap`). Este proceso ya lo tiene incorporado TrackR-CNN cuando se indica `visualize_tracks` a `true`, por lo que un ejemplo de la visualización se muestra en la Figura 4.1.

En cuanto a la evaluación de resultados, observando los valores que se aportan en el artículo (ver tabla 4.3), para los parámetros por defecto el resultado se muestra en la tabla 4.4. Se han conseguido los mismos valores que en el artículo.

	sMOTSA		MOTSA		MOTSP	
	Car	Ped	Car	Ped	Car	Ped
TrackR-CNN (ours)	76.2	46.8	87.8	65.1	**87.2**	**75.7**
Mask R-CNN + masjprop	75.1	45.0	86.6	63.5	87.1	75.6
TrackR-CNN (box-orig) + MG	75.0	41.2	87.0	57.9	86.8	76.3
TrackR-CNN (ours) + MG	76.2	**47.1**	87.8	**65.5**	**87.2**	**75.7**

Tabla 4.3: Resultados en KITTI MOTS

Compute KITTI tracking eval with simplified matching and MOTSA																									
Sequence	Object	sMOTSA	MOTSA	MOTSP	MOTSAL	MODSA	MODSP	Recall	Prec	F1	FAR	MT	PT	ML	TP	FP	FN	IDS	Frag	GT Obj	Gt Trk	TR Obj	TR Trk	Ig TR Tck
all	car	**76.2**	**87.8**	**87.2**	88.9	89.0	89.8	90.6	98.2	94.3	4.5	87.4	11.3	1.3	7276	134	753	93	130	8029	151	9155	224	1745
	ped.	**46.8**	**65.1**	**75.7**	67.4	67.5	93.8	75.4	90.4	82.3	9.0	61.8	32.4	5.9	2525	267	822	78	129	3347	68	3630	88	838
0002	car	60.8	74.9	82.7	78.1	78.3	85.4	81.6	96.1	88.3	12.9	73.3	20.0	6.7	737	30	166	31	37	903	15	916	25	149
	ped.	51.9	76.7	68.8	78.1	78.3	80.9	79.4	98.6	88.0	0.9	0.0	100.0	0.0	143	2	37	3	8	180	1	158	4	13
0006	car	85.5	96.1	89.2	96.4	96.5	90.6	97.4	99.1	98.2	1.9	100.0	0.0	0.0	523	5	14	2	6	537	11	647	21	119
	ped.	-inf	-inf	inf	-inf	-inf	100.0	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0	1	0	0	0	0	0	1	1	0
0007	car	84.6	94.6	89.6	94.9	95.0	92.0	96.1	98.80	97.4	3.2	92.5	7.5	0.0	2170	26	88	9	14	2258	53	2794	69	598
	ped.	26.6	40.3	73.8	41.8	41.8	98.9	52.2	83.3	64.2	0.9	0.0	100.0	0.0	35	7	32	1	4	67	2	78	9	36
0008	car	83.4	96.4	86.6	97.0	97.0	86.2	97.2	99.8	98.5	0.5	90.5	9.5	0.0	1013	2	29	6	11	1042	21	1257	28	242
	ped.	-inf	-inf	inf	-inf	-inf	100.0	0.0	0.0	0.0	11.0	0.0	0.0	0.0	0	43	0	0	0	0	0	44	7	1
0010	car	85.1	96.2	88.5	96.3	96.3	89.6	96.3	100.0	98.1	0.0	92.3	7.7	0.0	580	0	22	1	4	602	13	677	16	97
	ped.	19.4	29.1	66.6	29.1	29.1	98.2	29.1	100.0	45.1	0.0	0.0	50.0	50.0	16	0	39	0	1	55	2	17	3	1
0013	car	60.7	75.0	83.4	77.8	77.8	98.5	86.1	91.2	88.6	0.9	100.0	0.0	0.0	31	3	5	1	1	36	2	52	11	18
	ped.	57.1	77.6	76.4	79.7	79.9	82.5	86.5	92.9	89.6	17.9	73.8	21.4	4.8	795	61	124	21	31	919	42	1546	36	690
0014	car	64.7	79.3	82.6	80.2	80.4	82.2	83.9	96.0	89.5	15.1	71.4	28.6	0.0	385	16	74	5	16	459	14	502	13	101
	ped.	-19.3	-0.8	61.6	1.3	1.7	85.4	47.9	50.9	49.4	52.8	0.0	100.0	0.0	58	56	63	3	7	121	2	114	5	0
0016	car	44.7	56.7	81.2	60.5	60.7	82.1	63.9	95.2	76.5	12.9	50.0	25.0	25.0	533	27	301	33	32	834	4	563	17	3
	ped.	49.2	66.4	76.7	68.8	68.9	76.4	73.7	93.8	82.6	46.4	57.9	36.8	5.3	1478	97	527	50	78	2005	19	1672	23	97
0018	car	82.8	93.8	88.5	94.1	94.2	89.9	96.0	98.1	97.1	7.4	88.9	11.1	0.0	1304	25	54	5	9	1358	18	1747	24	418
	ped.	-inf	-inf	inf	-inf	-inf	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0	0	0	0	0	0	0	0	0

Tabla 4.4: Evaluación de los resultados de seguimiento con los parámetros por defecto

Por último, en su código también aportan la opción de obtener los parámetros óptimos del modelo seleccionado a través del *tuning* o ajuste. Este *script* de ajuste aleatorio encuentra la mejor combinación de parámetros de seguimiento en el conjunto de entrenamiento (secuencias 0000, 0001, 0003, 0004, 0005, 0009, 0011, 0012, 0015, 0017, 0019 y 0020) y posteriormente evalúa dichos parámetros en el conjunto de validación (secuencias 0002, 0006, 0007, 0008, 0010, 0013, 0014, 0016 y 0018). Los resultados obtenidos por los autores de este trabajo se realizaron de esta manera.

Dentro del *script* de ajuste (./scripts/eval/segtrack_tune_experiment.py), se pueden encontrar los parámetros con los que se realizan las diferentes iteraciones, así como sus valores mínimos y máximos. Inicialmente parten con los valores indicados en la tabla 4.5. En ella también se indican el rango de valores aleatorio, o el valor fijo, que pueden tomar.

Para ejecutar por primera vez el comando de ajuste, se configuran los parámetros del mismo modo que en el artículo para poder conseguir sus mismos resultados. Esto es `<association_type>` y

Default settings: car				
	tracker	reid_comp	detection_confidence_threshold_car	reid_weight_car
inicial	hungarian	euclidean	0.55	0.0
rango	hungarian	euclidean	(0.5, 0.85)	1.0
	mask_iou_weight_car	bbox_center_weight_car	bbox_iou_weight_car	association_threshold_car
inicial	0.0	0.0	0.0	0.3
rango	1.0	1.0	1.0	(0.0, 0.99)
	keep_alive_car	reid_euclidean_offset_car	reid_euclidean_scale_car	new_reid_threshold_car
inicial	0	5	1	2.0
rango	(0, 4)	(5.0, 10.0)	(0.7, 1.7)	(0.7, 1.7)
	box_offset	box_scale	new_reid	
inicial	50.0	0.02	False	
rango	(30.0, 80.0)	(0.01, 0.1)	False	
Default settings: ped				
	tracker	reid_comp	detection_confidence_threshold_ped	reid_weight_ped
inicial	hungarian	euclidean	0.95	0.0
rango	hungarian	euclidean	(0.8, 0.99)	1.0
	mask_iou_weight_ped	bbox_center_weight_ped	bbox_iou_weight_ped	association_threshold_ped
inicial	0.0	0.0	0.0	0.3
rango	1.0	1.0	1.0	(0.0, 0.99)
	keep_alive_ped	reid_euclidean_offset_ped	reid_euclidean_scale_ped	new_reid_threshold_ped
inicial	0	5	1	2.0
rango	(6, 10)	reid_euclidean_offset_car	reid_euclidean_scale_car	(0.7, 1.7)
	box_offset	box_scale	new_reid	
inicial	50.0	0.02	False	
rango	(30.0, 80.0)	(0.01, 0.1)	False	

Tabla 4.5: Parámetros iniciales para el ajuste

<num_iterations> con un valor de reid (utilizando las incrustaciones de asociación) y 1000 respectivamente:

```
python3 scripts/eval/segtrack_tune_experiment.py </path/to/
detections/> </path/to/groundtruth/> </path/to/precomputed_optical_
flow/> </path/to/output_file> </path/to/tmp_folder/> </path/to/mots_
eval/> <association_type> <num_iterations>
```

donde </path/to/detections/> apunta al directorio donde se encuentran las detecciones obtenidas por el comando *forwarding* (./TrackR-CNN/forwarded/conv3d_sep2/detections/5, </path/to/groundtruth/> a las instancias *Ground Truth* (./TrackR-CNN/data/KITTI_MOTS/train/instances), </path/to/precomputed _optical_flow/> se ignora para este experimento, </path/to/output_file> al archivo de texto editable que contendrá los resultados de las iteraciones individuales de ajuste, </path/to/tmp_folder/> al directorio donde se crearán carpetas temporales durante el experimento, y </path/to/mots_eval/> al directorio donde se encuentre el *script* de evaluación de la herramienta mots_tools.

Los resultados de este primer *tuning* se pueden visualizar en la tabla 4.6 y se extraen de la consola de comandos. En ella se indica la mejor puntuación sMOTSA, tanto para coches como para peatones, que se ha obtenido en el modelo de entrenamiento combinando los parámetros de seguimiento y la puntuación resultante en el modelo de evaluación con dichos parámetros. Además de esto, el resultado proporciona un archivo .txt con todas las combinaciones efectuadas de los parámetros y sus distintas puntuaciones en el modelo de entrenamiento (tuning_results.txt). Los parámetros que se muestran en la tabla para las mejores puntuaciones de coches y peatones no tienen por qué ser de la misma iteración, en este caso no lo son.

Para verificar que con 1000 iteraciones en el proceso de *tuning* es suficiente para detectar los mejores parámetros de configuración, se realizan dos afinamientos más: el primero con 100 iteraciones (véase Tabla 4.7) y el segundo con 2000 iteraciones (véase Tabla 4.8).

Best CAR sMOTSA train 76.7			
Settings			
tracker	reid_comp	detection_confidence_threshold_car	reid_weight_car
hungarian	euclidean	0.8436857796543771 default: 0.8469800990815324	1.0
mask_iou_weight_car	bbox_center_weight_car	bbox_iou_weight_car	association_threshold_car
0.0	0.0	0.0	0.4164745015027633 default: 0.8165986526897969
keep_alive_car	reid_euclidean_offset_car	reid_euclidean_scale_car	new_reid_threshold_car
4 default: 4	8.221494711468113 default: 8.810218833503743	1.5018496324494501 default: 1.0090931467228708	2.0
box_offset	box_scale	new_reid	
50.0	0.02	False	
[('0002', 233), ('0006', 270), ('0007', 800), ('0008', 390), ('0010', 294), ('0013', 340), ('0014', 106), ('0016', 209), ('0018', 339)] False			
Scores			
sMOTSA	**MOTSA**	**MOTSP**	**IDS**
76.2	**87.8**	**87.2**	**97**

Best PED sMOTSA train 57.4			
Settings			
tracker	reid_comp	detection_confidence_threshold_ped	reid_weight_ped
hungarian	euclidean	0.9369257762801722 default: 0.9368820089063415	1.0
mask_iou_weight_ped	bbox_center_weight_ped	bbox_iou_weight_ped	association_threshold_ped
0.0	0.0	0.0	0.12249982741161666 default: 0.47985540892434836
keep_alive_ped	reid_euclidean_offset_ped	reid_euclidean_scale_ped	new_reid_threshold_ped
9 default: 6	6.016631257116043 default: 9.447084376750222	1.2216388038749773 default: 1.3437965549876354	2.0
box_offset	box_scale	new_reid	
50.0	0.02	False	
[('0002', 233), ('0006', 270), ('0007', 800), ('0008', 390), ('0010', 294), ('0013', 340), ('0014', 106), ('0016', 209), ('0018', 339)] False			
Scores			
sMOTSA	**MOTSA**	**MOTSP**	**IDS**
47.0	**65.3**	**75.7**	**73**

Tabla 4.6: Resultados del ajuste con 1000 iteraciones

En la Tabla 4.9 aparecen resumidas las puntuaciones obtenidas en el modelo de entrenamiento de las tres pruebas de afinamiento realizadas. Como se puede apreciar, las puntuaciones de sMOTSA y MOTSA para el caso de los coches mejoran notablemente de 100 a 1000 iteraciones en la prueba de evaluación. Sin embargo, la diferencia es aparentemente nula cuando se pasa de 1000 a 2000 iteraciones.

Para garantizar los hipotéticos resultados que se obtienen configurando el modelo con los parámetros del *tuning*, se realizan dos pruebas más ejecutando *forwarding and tracking*. La primera se realiza con los parámetros que han resultado del *tuning* combinando la mejor puntuación de los coches con la de los peatones de distintas iteraciones. La segunda prueba utiliza los parámetros de una de las iteraciones que ha dado también la mejor puntuación de coches pero no es la misma que la de la tabla (en esta iteración también coincide la puntuación de los peatones con la mejor). Para realizar estos experimentos, se han modificado los pesos en el archivo /configs/conv3d_sep2.

En el primer caso, la puntuación que se obtiene (con mots_tools) es similar a la del *tuning* y casi igual que la que ya se obtenía con los parámetros por defecto. La segunda prueba, aunque tenga la misma puntuación que la mejor en el entrenamiento, en la evaluación no obtiene los mismos resultados al haber utilizado otros parámetros. La primera prueba realizada se muestra en la Tabla 4.10 y la segunda prueba en la Tabla 4.11.

En la Tabla 4.12 se muestran los resultados obtenidos de las tres pruebas realizadas, la primera con los parámetros por defecto y las otras dos con parámetros obtenidos por el *tuning*. Aunque no sea grande la diferencia, se han obtenido mejores resultados modificando los parámetros por defecto por los aportados tras el *tuning* mejorando así al artículo.

Best CAR sMOTSA train 76.6			
Settings			
tracker	reid_comp	detection_confidence_threshold_car	reid_weight_car
hungarian	euclidean	0.848681988976919 default: 0.8469800990815324	1.0
mask_iou_weight_car	bbox_center_weight_car	bbox_iou_weight_car	association_threshold_car
0.0	0.0	0.0	0.15088550740985654 default: 0.8165986526897969
keep_alive_car	reid_euclidean_offset_car	reid_euclidean_scale_car	new_reid_threshold_car
1 default: 4	7.679929358043557 default: 8.810218833503743	1.0738153667661239 default: 1.0090931467228708	2.0
box_offset	box_scale	new_reid	
50.0	0.02	False	
[('0002', 233), ('0006', 270), ('0007', 800), ('0008', 390), ('0010', 294), ('0013', 340), ('0014', 106), ('0016', 209), ('0018', 339)] False			
Scores			
sMOTSA	**MOTSA**	**MOTSP**	**IDS**
75.9	**87.5**	**87.2**	**114**

Best PED sMOTSA train 57.8			
Settings			
tracker	reid_comp	detection_confidence_threshold_ped	reid_weight_ped
hungarian	euclidean	0.9394273908103546 default: 0.9368820089063415	1.0
mask_iou_weight_ped	bbox_center_weight_ped	bbox_iou_weight_ped	association_threshold_ped
0.0	0.0	0.0	0.7128210970340386 default: 0.47985540892434836
keep_alive_ped	reid_euclidean_offset_ped	reid_euclidean_scale_ped	new_reid_threshold_ped
7 default: 6	7.110100684109886 default: 9.447084376750222	1.3799946399717806 default: 1.3437965549876354	2.0
box_offset	box_scale	new_reid	
50.0	0.02	False	
[('0002', 233), ('0006', 270), ('0007', 800), ('0008', 390), ('0010', 294), ('0013', 340), ('0014', 106), ('0016', 209), ('0018', 339)] False			
Scores			
sMOTSA	**MOTSA**	**MOTSP**	**IDS**
47.0	**65.3**	**75.7**	**75**

Tabla 4.7: Resultados del ajuste con 100 iteraciones

4.3 Resultados obtenidos con PointTrack

En este segundo apartado se comprueba el funcionamiento del código del proyecto PointTrack para la tarea de MOTS [32]. En este caso, se desconoce cómo se ha ejecutado el código, solamente indican como requisito que se utilice una versión de Python 3.6 o superior. La configuración HW/SW que se ha empleado es:

- Sistema operativo Linux Ubuntu 18.04 LTS. Se ha ejecutado el código sobre el propio sistema en modo nativo y también sobre Google Colab.
- Python 3.6.9.
- Una sola GPU GTX 1650.

Para poder ejecutar el código en modo nativo, es necesario realizar una serie de pasos previos de configuración:

1. Descargar el código de *Github* (`https://github.com/detectRecog/PointTrack`). Para descargar un proyecto de *Github* existen dos opciones:
   - Descargar desde el repositorio. Navegar hasta el repositorio que se quiera clonar, hacer clic en el botón verde y generar el archivo .zip.
   - Descargar desde la consola de comandos. Mediante la consola de comandos, ubicarse en la ruta deseada para clonar. En el repositorio del proyecto, hacer clic en el botón verde y copiar la URL del documento. Usar el siguiente comando:

Best CAR sMOTSA train 76.7			
Settings			
tracker	reid_comp	detection_confidence_threshold_car	reid_weight_car
hungarian	euclidean	0.8440796306799431 default: 0.8469800990815324	1.0
mask_iou_weight_car	bbox_center_weight_car	bbox_iou_weight_car	association_threshold_car
0.0	0.0	0.0	0.2509414002745895 default: 0.8165986526897969
keep_alive_car	reid_euclidean_offset_car	reid_euclidean_scale_car	new_reid_threshold_car
3 default: 4	9.460887382664694 default: 8.810218833503743	0.9224776399671926 default: 1.0090931467228708	2.0
box_offset	box_scale	new_reid	
50.0	0.02	False	
[('0002', 233), ('0006', 270), ('0007', 800), ('0008', 390), ('0010', 294), ('0013', 340), ('0014', 106), ('0016', 209), ('0018', 339)] False			
Scores			
sMOTSA	**MOTSA**	**MOTSP**	**IDS**
76.2	**87.8**	**87.2**	**96**

Best PED sMOTSA train 57.9			
Settings			
tracker	reid_comp	detection_confidence_threshold_ped	reid_weight_ped
hungarian	euclidean	0.9371264058836783 default: 0.9368820089063415	1.0
mask_iou_weight_ped	bbox_center_weight_ped	bbox_iou_weight_ped	association_threshold_ped
0.0	0.0	0.0	0.1403031067152116 default: 0.47985540892434836
keep_alive_ped	reid_euclidean_offset_ped	reid_euclidean_scale_ped	new_reid_threshold_ped
6 default: 6	8.179160796354198 default: 9.447084376750222	1.2766271183323186 default: 1.3437965549876354	2.0
box_offset	box_scale	new_reid	
50.0	0.02	False	
[('0002', 233), ('0006', 270), ('0007', 800), ('0008', 390), ('0010', 294), ('0013', 340), ('0014', 106), ('0016', 209), ('0018', 339)] False			
Scores			
sMOTSA	**MOTSA**	**MOTSP**	**IDS**
46.8	**65.2**	**75.7**	**78**

Tabla 4.8: Resultados del ajuste con 2000 iteraciones

	Best sMOTSA train		sMOTSA		MOTSA		MOTSP	
	Car	Ped	Car	Ped	Car	Ped	Car	Ped
100 iteraciones	76.6	57.8	75.9	47.0	87.5	65.3	87.2	75.7
1000 iteraciones	76.7	57.4	76.2	47.0	87.8	65.3	87.2	75.7
2000 iteraciones	76.7	57.9	76.2	46.8	87.8	65.2	87.2	75.7
Artículo			76.2	46.8	87.8	65.1	87.2	75.7

Tabla 4.9: Resumen resultados tuning

```
git clone <URL>
```

donde `<URL>` en este caso en concreto es https://github.com/detectRecog/PointTrack.git.

En caso de no tener instalado `git` u otro paquete, se consigue utilizando esta serie de comandos:

```
sudo apt-get update
sudo apt-get install <package>
<package> --version
```

donde `<package>` en este caso es `git`.

2. Compilar e instalar mediante comandos paquetes para Python3. Proporcionan un documento de texto que recoge todos los paquetes necesarios y sus respectivas versiones (requirements.txt). Para instalar un paquete para Python3 basta con escribir este comando:

Compute KITTI tracking eval with simplified matching and MOTSA																									
Sequence	Object	sMOTSA	MOTSA	MOTSP	MOTSAL	MODSA	MODSP	Recall	Prec	F1	FAR	MT	PT	ML	TP	FP	FN	IDS	Frag	GT Obj	Gt Trk	TR Obj	TR Trk	Ig TR Tck
all	car	76.2	87.8	87.2	89.0	89.0	89.8	90.7	98.2	94.3	4.6	87.4	11.3	1.3	7286	137	743	97	131	8029	151	9185	224	1762
	ped.	47.0	65.3	75.7	67.4	67.5	93.8	75.2	90.4	82.3	9.0	61.8	32.4	5.9	2525	267	822	73	128	3347	68	3630	73	838
0002	car	60.6	74.8	82.7	78.0	78.2	85.4	81.7	95.8	88.2	13.7	73.3	20.0	6.7	738	32	165	31	37	903	15	919	24	149
	ped.	51.9	76.7	68.8	78.1	78.3	80.9	79.4	98.6	88.0	0.9	0.0	100.0	0.0	143	2	37	3	8	180	1	158	4	13
0006	car	85.5	96.1	89.2	96.4	96.5	90.6	97.4	99.1	98.2	1.9	100.0	0.0	0.0	523	5	14	2	6	537	11	647	21	119
	ped.	-inf	-inf	inf	-inf	-inf	100.0	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0	1	0	0	0	0	0	1	1	0
0007	car	84.6	94.6	89.6	94.9	95.0	92.0	96.1	98.8	97.4	3.2	92.5	7.5	0.0	2170	26	88	9	14	2258	53	2799	69	603
	ped.	26.6	40.3	73.8	41.8	41.8	98.9	52.2	83.3	64.2	0.9	0.0	100.0	0.0	35	7	32	1	4	67	2	78	9	36
0008	car	83.4	96.4	86.6	97.0	97.0	86.2	97.2	99.8	98.5	0.5	90.5	9.5	0.0	1013	2	29	6	11	1042	21	1259	28	244
	ped.	-inf	-inf	inf	-inf	-inf	100.0	0.0	0.0	0.0	11.0	0.0	0.0	0.0	0	43	0	0	0	0	0	44	6	1
0010	car	84.6	95.7	88.5	96.1	96.2	89.6	96.3	99.8	98.1	0.3	92.3	7.7	0.0	580	1	22	3	6	602	13	679	17	98
	ped.	19.4	29.1	66.6	29.1	29.1	98.2	29.1	100.0	45.1	0.0	0.0	50.0	50.0	16	0	39	0	1	55	2	17	3	1
0013	car	60.7	75.0	83.4	77.8	77.8	98.5	86.1	91.2	88.6	0.9	100.0	0.0	0.0	31	3	5	1	1	36	2	53	11	19
	ped.	57.0	77.5	76.4	79.7	79.9	82.5	86.5	92.9	89.6	17.9	73.8	21.4	4.8	795	61	124	22	32	919	42	1546	27	690
0014	car	64.9	79.5	82.6	80.7	80.8	82.2	84.3	96.0	89.8	15.1	71.4	28.6	0.0	387	16	72	6	15	459	14	505	12	102
	ped.	-19.3	-0.8	61.6	1.3	1.7	85.4	47.9	50.9	49.4	52.8	0.0	100.0	0.0	58	56	63	3	7	121	2	114	4	0
0016	car	45.1	57.3	81.1	61.2	61.4	81.9	64.6	95.2	77.0	12.9	50.0	25.0	25.0	539	27	295	34	32	834	4	569	18	3
	ped.	49.5	66.7	76.7	68.8	68.9	76.4	73.7	93.8	82.6	46.4	57.9	36.8	5.3	1478	97	527	44	76	2005	19	1672	19	97
0018	car	82.8	93.9	88.5	94.2	94.3	89.9	96.1	98.1	97.1	7.4	88.9	11.1	0.0	1305	25	53	5	9	1358	18	1755	24	425
	ped.	-inf	-inf	inf	-inf	-inf	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0	0	0	0	0	0	0	0	0

Tabla 4.10: Evaluación de los resultados de seguimiento con los parámetros de la primera prueba

Compute KITTI tracking eval with simplified matching and MOTSA																									
Sequence	Object	sMOTSA	MOTSA	MOTSP	MOTSAL	MODSA	MODSP	Recall	Prec	F1	FAR	MT	PT	ML	TP	FP	FN	IDS	Frag	GT Obj	Gt Trk	TR Obj	TR Trk	Ig TR Tck
all	car	76.2	87.9	87.1	89.1	89.1	89.8	90.8	98.1	94.3	4.3	87.4	11.3	1.3	7293	140	736	98	130	8029	151	9210	224	1777
	ped.	47.1	65.4	75.7	67.5	67.5	93.8	75.4	90.5	82.3	8.9	60.3	33.8	5.9	2524	264	823	72	126	3347	68	3613	69	825
0002	car	60.7	74.9	82.7	78.2	78.4	85.4	81.9	95.9	88.4	13.7	73.3	20.0	6.7	740	32	163	32	36	903	15	923	24	151
	ped.	51.9	76.7	68.8	78.1	78.3	80.9	79.4	98.6	88.0	0.9	0.0	100.0	0.0	143	2	37	3	8	180	1	158	4	13
0006	car	85.5	96.1	89.2	96.4	96.5	90.6	97.4	99.1	98.2	1.9	100.0	0.0	0.0	523	5	14	2	6	537	11	647	21	119
	ped.	-inf	-inf	inf	-inf	-inf	100.0	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0	1	0	0	0	0	0	1	1	0
0007	car	84.6	94.6	89.6	95.0	95.0	92.0	96.2	98.8	97.5	3.4	92.5	7.5	0.0	2172	27	86	9	13	2258	53	2809	68	610
	ped.	26.6	40.3	73.8	41.8	41.8	98.9	52.2	83.3	64.2	0.9	0.0	100.0	0.0	35	7	32	1	4	67	2	78	9	36
0008	car	83.4	96.4	86.6	97.0	97.0	86.2	97.2	99.8	98.5	0.5	90.5	9.5	0.0	1013	2	29	6	11	1042	21	1260	28	245
	ped.	-inf	-inf	inf	-inf	-inf	100.0	0.0	0.0	0.0	11.0	0.0	0.0	0.0	0	41	0	0	0	0	0	42	6	1
0010	car	84.6	95.7	88.5	96.1	96.2	89.6	96.3	99.8	98.1	0.3	92.3	7.7	0.0	580	0	22	3	6	602	13	680	17	99
	ped.	19.4	29.1	66.6	29.1	29.1	98.2	29.1	100.0	45.1	0.0	0.0	50.0	50.0	16	0	39	0	1	55	2	17	3	1
0013	car	60.7	75.0	83.4	77.8	77.8	98.5	86.1	91.2	88.6	0.9	100.0	0.0	0.0	31	3	5	1	1	36	2	53	11	19
	ped.	57.3	77.7	76.4	79.7	79.9	82.5	86.5	92.9	89.6	17.9	73.8	21.4	4.8	795	61	124	20	30	919	42	1535	24	679
0014	car	64.8	79.5	82.6	80.7	80.8	82.1	84.5	95.8	89.8	16.0	71.4	28.6	0.0	388	17	71	6	15	459	14	508	13	103
	ped.	-18.4	0.0	61.6	2.1	2.5	85.4	47.9	51.3	49.6	51.9	0.0	100.0	0.0	58	55	63	3	7	121	2	113	3	0
0016	car	45.2	57.4	81.1	61.3	61.5	81.8	64.9	95.1	77.1	13.4	50.0	25.0	25.0	541	28	293	34	33	834	4	572	18	3
	ped.	49.4	66.6	76.7	68.7	68.8	76.4	73.7	93.8	82.5	46.4.1	52.6	42.1	5.3	1477	97	528	45	76	2005	19	1669	19	95
0018	car	82.8	93.9	88.5	94.2	94.3	89.9	96.1	98.1	97.1	7.4	88.9	11.1	0.0	1305	25	53	5	9	1358	18	1758	24	428
	ped.	-inf	-inf	inf	-inf	-inf	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0	0	0	0	0	0	0	0	0

Tabla 4.11: Evaluación de los resultados de seguimiento con los parámetros de la segunda prueba

```
pip3 install <module>
```

donde `<module>` es el nombre del paquete que se quiere instalar.

Un paso previo que se ha realizado para mayor comodidad es renombrar el comando pip3 a pip.

```
sudo apt-get update
sudo apt-get install python3-pip
pip3 --version
alias pip=pip3
echo alias pip=pip3 >> ~/.bashrc
source ~/.bashrc
```

De modo que la instalación de paquetes/librerías quedaría:

```
pip install -r requirements.txt
```

	sMOTSA		MOTSA		MOTSP	
	Car	Ped	Car	Ped	Car	Ped
TrackR-CNN (paper)	76.2	46.8	87.8	65.1	87.2	75.7
Original	76.2	46.8	87.8	65.1	87.2	75.7
Prueba 1	76.2	47.0	87.8	65.3	87.2	75.7
Prueba 2	76.2	47.1	87.9	65.4	87.1	75.7

Tabla 4.12: Resumen resultados evaluación del código

3. Descargar imágenes e instancias de la base de datos KITTI MOTS. Las imágenes se pueden descargar en http://www.cvlibs.net/download.php?file=data_tracking_image_2.zip y las instancias puedes obtenerlas en dos formatos distintos: en formato .png (https://www.vision.rwth-aachen.de/media/resource_files/instances.zip) o en formato .txt (https://www.vision.rwth-aachen.de/media/resource_files/instances_txt.zip). En este caso se ha optado por el formato .txt. ya que son archivos más pequeños y más rápidos de leer. Cada línea de un archivo .txt de instancias está estructurada de la forma en la que se indica en la Tabla 4.2 (donde RLE significa codificación de longitud de ejecución de MS COCO).

4. Crear la estructura de carpetas definida en el código aportado para su implementación. Una vez descargadas las imágenes y las instancias, se copian en el directorio raíz y se modifica la ruta en el *script* ./config.py.

5. Descargar los modelos preajustados para coches para PointTrack. Estos modelos se pueden encontrar en https://drive.google.com/file/d/14Hn4ZztfjGUYEjVd-9FRNB5a-CtBkPXc/view. Estos archivos deben ser copiados a la carpeta ./pointTrack_weights.

6. Modificar los *scripts* de configuración. Esto es, modificar los parámetros systemRoot, kittiRoot, rootDir y pythonPath en el archivo ./config.py a las rutas que tengamos establecidas en el sistema.

Una vez realizados los pasos de configuración, se procede a ejecutar los comandos del código para obtener la asociación de instancias. Los parámetros de configuración del modelo por defecto se encuentran en ./pointTrack_weights/PointTrack.pthCar. Los pasos a seguir son los siguientes:

1. Generar el resultado de la segmentación en el conjunto de validación. El comando a utilizar es:

   ```
   python3 -u test_mots_se.py car_test_se_to_save
   ```

   El resultado se almacena donde se haya definido en repoRoot/config_mots/car_test_se_to_save.py, por defecto en ./car_SE_val_prediction/'.

2. Probar PointTrack en los resultados de la segmentación de la instancia. Esto se realiza con el comando:

   ```
   python3 -u test_tracking.py car_test_tracking_val
   ```

   Un problema común que puede aparecer en este apartado es el hecho de que no funcione el comando. Buscando en las *issues* se puede encontrar una entrada en la que se indica que es un fallo común y que todavía no estaba resuelto. La solución que se ha adoptado en este TFM es renombrar los directorios, ya que algunas otras personas lo solucionaron así. La solución pasa por añadir */pointTrack_weights/PointTrack.pthCar* en el path *./car_finetune_tracking/checkpoint.pth* cambiando así el nombre del fichero.

Este comando ha generado los resultados de segmentación en formato .txt en el directorio repoRoot/tracks_car_pointtrack_val para cada secuencia. Estos resultados podrían reproducirse con ayuda de mots_tols para obtener la evaluación y visualizar los *tracks*.

3. Generar la instancia DB a partir de vídeos. El comando a utilizar es:

   ```
   python3 -u datasets/MOTSInstanceMaskPool.py
   ```

   Como resultado se genera una carpeta en el directorio raíz ImgTrackEnvDB con archivos .pkl.

4. Cambiar la línea que carga los pesos del modelo a la ruta que contiene los pesos por defecto.

   Esta modificación se realizó en el paso que genera los resultados de segmentación para poder ejecutarse.

5. Entrenar el modelo. Para realizar este paso se utiliza el comando:

   ```
   python3 -u train_tracker_with_val.py car_finetune_tracking
   ```

   Como resultado, se guardará el mejor seguimiento ajustado del conjunto de validación en la carpeta que se haya especificado en repoRoot/config_mots/car_finetune_tracking.py. Estos valores se utilizaron para realizar de nuevo el primer y segundo paso, sin embargo las puntuaciones obtenidas eran peores por lo que no se tuvieron en cuenta.

   En este apartado apareció un error relacionado con la memoria disponible en el sistema, como se puede ver en el siguiente mensajes de error:

   ```
   RuntimeError: CUDA out of memory. Tried to allocate 142.00 MiB (GPU 0;
   3.82 GiB total capacity; 2.23 GiB already allocated; 93.19 MiB free;
   2.32 GiB reserved in total by PyTorch)
   ```

   Buscando información en los foros disponibles, se encontró la siguiente posible solución para vaciar la caché del procesador e intentar ganar algo de espacio:

   ```
   gc.collect()    \#andrea
   torch.cuda.empty\_cache()    \#andrea
   ```

   Pero aún así, y en base al siguiente mensaje de error, el resultado fue insuficiente:

   ```
   RuntimeError: CUDA out of memory. Tried to allocate 18.00 MiB
   (GPU 0; 3.82 GiB total capacity; 2.34 GiB already allocated;
   11.81 MiB free; 2.41 GiB reserved in total by PyTorch)
   ```

   En vista de que no se pudo resolver el problema, y que en todas las funciones que utilizaban el módulo 'torch', el error siempre aparecía, se decidió probar en *Google Colab*. *Google Colab* permite programar y ejecutar Python en el navegador con las siguientes ventajas: no requiere configuración, da acceso gratuito a GPUs potentes y permite compartir contenido fácilmente. Los cuadernos de *Colab* ejecutan código en los servidores en la nube de *Google*, lo que permite aprovechar la potencia del *hardware* de *Google*, incluidas las GPU y TPU, independientemente de la potencia del equipo

   Para el uso de *Google Colab* y *One Drive* es necesario disponer de una cuenta de *Google*. Para trabajar de manera más óptima, se pueden contratar más espacio en la nube de *Drive* (limitado a 25 GB gratis) y contratar *Colab Pro*. Con *Colab Pro* se tiene una serie de ventajas frente a la versión gratuita:

- Acceso prioritario a sus GPUs más rápidas. Por ejemplo, es posible tener acceso a una GPU T4 o P100, mientras que la mayoría de los usuarios de la versión gratuita de *Colab* solo pueden usar una GPU K80, que es más lenta. Para poder usar una GPU con tu cuaderno, hay que seleccionar el menú Entorno de ejecución > Cambiar tipo de entorno de ejecución y, a continuación, elegir GPU en el menú desplegable.
- Con *Colab Pro*, se tiene la posibilidad de acceder a VMs con alta capacidad de memoria si están disponibles. Si se quieren modificar las preferencias del cuaderno para usar un entorno de ejecución con una alta capacidad de memoria, hay que seleccionar el menú Entorno de ejecución > Cambiar tipo de entorno de ejecución y, a continuación, elegir Alta capacidad de RAM en el menú desplegable Características del entorno de ejecución.
- Todos los entornos de ejecución de *Colab* se restablecen tras un periodo concreto, que es más breve si el entorno no está ejecutando ningún código. Pese a que los suscriptores de *Colab Pro* siguen teniendo límites, estos equivalen aproximadamente al doble que los de los usuarios de la versión gratuita.

Para ejecutar un cuaderno de *Google Colab*, se copia el repositorio con los cambios realizados y los resultados obtenidos hasta el momento en la nube de *Google Drive*. Se le llama cuaderno al entorno interactivo que te permite escribir y ejecutar código. Los cuadernos de *Colab* son cuadernos de *Jupyter* alojados en *Colab*. Si se selecciona la opción dentro del cuaderno de *Colab* de Activar *Drive*, cada vez que se conecte a un entorno de ejecución, se habilitarán los directorios dentro de ./content/drive/MyDrive. Es necesario tener que modificar las rutas en las que se encontraban los directorios en el equipo para ubicarlos dentro del cuaderno. Para su ejecución, se tiene que repetir el punto 2 de los pasos de configuración para disponer de los mismos paquetes y versiones. Para ejecutar las distintas líneas de comando, hay que añadir el símbolo ! al principio de cada línea. Si se desea navegar los directorios o consultar, hay que añadir el símbolo % al principio de cada línea (cd, ls).

Compute KITTI tracking eval with simplified matching and MOTSA																									
Sequence	Object	sMOTSA	MOTSP	MOTSAL	MODSA	MODSP	Recall	Prec	F1	FAR	MT	PT	ML	TP	FP	FN	IDS	Frag	GT Obj	Gt Trk	TR Obj	TR Trk	Ig	TR Tck
all	car	**85.56**	**94.99**	**90.26**	95.19	95.20	92.39	96.86	98.32	97.58	4.46	94.70	5.30	0.00	7777	133	22	17	95	8029	151	10097	109	2187
0002	car	77.38	91.25	85.22	91.63	91.69	87.16	93.91	97.70	95.77	8.58	100.00	0.00	0.00	848	20	55	4	25	903	15	1201	11	333
0006	car	89.75	97.77	91.89	97.77	97.77	92.75	98.88	98.88	98.88	2.22	100.00	0.00	0.00	531	6	6	0	3	537	11	756	6	219
0007	car	91.31	97.96	93.31	98.04	98.05	94.65	99.38	98.68	99.03	3.75	98.11	1.89	0.00	2244	30	14	2	4	2258	53	3159	31	885
0008	car	89.15	97.50	91.48	97.60	97.60	91.32	97.98	99.61	98.79	1.03	90.48	9.52	0.00	1021	4	21	1	9	1042	21	1216	15	191
0010	car	89.04	96.84	92.02	97.13	97.18	92.72	97.84	99.33	98.58	1.36	92.31	7.69	0.00	589	4	13	2	6	602	13	659	12	66
0013	car	-9.34	-2.78	92.84	-2.78	-2.78	99.30	91.67	49.25	64.08	10.00	100.00	0.00	0.00	33	34	3	0	1	36	2	124	5	57
0014	car	83.43	94.55	88.37	94.55	94.55	88.66	95.64	98.87	97.23	4.72	85.71	14.29	0.00	439	5	20	0	4	459	14	552	10	108
0016	car	69.92	88.49	79.62	89.00	89.09	79.58	91.13	97.81	94.35	8.13	100.00	0.00	0.00	760	17	74	5	38	834	4	783	9	6
0018	car	88.29	95.43	92.61	95.62	95.66	93.44	96.61	99.02	97.80	3.83	88.89	11.11	0.00	1312	13	46	3	5	1358	18	1647	10	322

Tabla 4.13: Evaluación de los resultados del código PointTrack para coches

Type	Method	Det. & Seg.	Speed	Cars			Pedestrians		
				sMOTSA	MOTSA	IDS	sMOTSA	MOTSA	IDS
2D	PointTrack	PointTrack	**0.045**	**85.5**	**94.9**	**22**	**62.4**	**77.3**	**19**
2D	PointTrack (without TC)	PointTrack	**0.045**	82.9	92.7	25	61.4	76.8	21
2D	PointTrack (on Bbox)	PointTrack	**0.045**	85.3	94.8	36	61.8	76.8	36

Tabla 4.14: Resultados Point Track en KITTI MOTS

Por último, para la evaluación de los resultados, se utiliza mots_tools, una herramienta que evalúa y visualiza los resultados de la tarea MOTS. Con su aportación, se pueden comprobar las puntuaciones obtenidas por el modelo seleccionado y compararse con las puntuaciones del artículo. Los pasos que hay que seguir para la utilización de mots_tools son:

1. Descargar mots_tools desde *Github* (https://github.com/VisualComputingInstitute/mots_tools).

   Para utilizar la herramienta es necesario tener instalado pycocotools. Los pasos a seguir son los siguientes:

   - Descargar cocotools desde *Github* (https://github.com/cocodataset/cocoapi).
   - Navegar hasta el directorio raíz del archivo y ejecutar el siguiente comando para instalar cocotools para Python:

     ```
     make coco/PythonAPI
     ```

   También es necesario comprobar que el directorio de mots_tools se encuentra en *Python path*.

2. Para evaluar el resultado del *tracking* se ejecuta el siguiente comando:

   ```
   python3 mots_eval/eval.py <tracking_results> <gt_folder> <seqmap>
   ```

   donde `<tracking_results>` indica el resultado en el que se localizan los resultados del *tracking* (`./tracks_car_pointtrack_val`), en `<gt_folder>` se encuentran las anotaciones *Ground Truth* (`./data/KITTI_MOTS/train/instances`) y `<seqmap>` es un archivo .txt que contiene las secuencias que desea evaluar, en este caso las secuencias del conjunto de validación (`mots_eval/val.seqmap`). La evaluación se imprime en la consola de comandos.

3. Para visualizar el resultado del *tracking* es necesario instalar previamente el paquete FFmpeg (con el comando "pip install ffmpeg"). El comando a utilizar para la visualización es:

   ```
   python3 mots_vis/visualize_mots.py <tracking_results> <img_folder>
   <output_folder> <seqmap>
   ```

   donde `<tracking_results>` indica el resultado en el que se localizan los resultados del *tracking* (`./tracks_car_pointtrack_val`), en `<img_folder>` se encuentran las imágenes del conjunto de datos KITTI MOTS (`./data/KITTI_MOTS/train/images`), `<output_folder>` es la carpeta donde se crearán los resultados visualizados, y `<seqmap>` es un archivo .txt que contiene las secuencias que desea evaluar, en este caso las secuencias del conjunto de validación (`mots_eval/val.seqmap`).

En la Tabla 4.13 se muestran los resultados de segmentación obtenidos en el primer y segundo punto, con los pesos por defecto. Sólo proporcionan el *training* para coches ya que el de los peatones no lo tienen del todo ajustado y por ello no está subido todavía. Como se puede apreciar, da el mismo resultado que el artículo. En las figuras 4.2 se visualizan los resultados de segmentación de las mismas secuencias, extraídas del vídeo 0016, que se obtuvieron para TrackR-CNN.

4.4 Resultados obtenidos con MOTSFusion

El tercer código a comprobar es el del proyecto MOTSFusion para la tarea de MOTS [31]. Este código ha sido probado con los siguientes requerimientos:

(a) 000072

(b) 000073

(c) 000074

(d) 000075

Figura 4.2: Visualización resultados tracking del video 0018 de Point Track

- PC de sobremesa con una CPU Intel Core i7-5930K
- Una sola GPU GTX 1080 Ti
- CUDA 9, cuDNN 7
- Tensorflow 1.13
- Python 3.6

La configuración HW/SW que se ha empleado es:

- Sistema operativo Linux Ubuntu 18.04 LTS. Se ha ejecutado el código sobre el propio sistema en modo nativo.
- CUDA 9, cuDNN 7

- Tensorflow 1.13
- Python 3.6.9.
- Una sola GPU GTX 1650.

Para poder ejecutar el código, es necesario realizar una serie de pasos previos de configuración:

1. Descargar el código de *Github* (https://github.com/tobiasfshr/MOTSFusion). Para descargar un proyecto de *Github* existen dos opciones:

   - Descargar desde el repositorio. Navegar hasta el repositorio que se quiera clonar, hacer clic en el botón verde y generar el archivo .zip.
   - Descargar desde la consola de comandos. Mediante la consola de comandos, ubicarse en la ruta deseada para clonar. En el repositorio del proyecto, hacer clic en el botón verde y copiar la URL del documento. Usar el siguiente comando:

     ```
     git clone <URL>
     ```

     donde <URL> en este caso en concreto es https://github.com/tobiasfshr/MOTSFusion.git.

     En caso de no tener instalado git u otro paquete, se consigue utilizando esta serie de comandos:

     ```
     sudo apt-get update
     sudo apt-get install <package>
     <package> --version
     ```

     donde <package> en este caso es git.

2. Descargar imágenes (pares de imágenes estereoscópicas) e instancias de la base de datos KITTI MOTS. Para este caso es necesario descargar tanto las imágenes de la cámara izquierda como las de la derecha. Estas imágenes se pueden descargar en http://www.cvlibs.net/datasets/kitti/eval_tracking.php y para ello es necesario crear un usuario en la página. Las instancias se pueden obtener en dos formatos distintos: en formato .png (https://www.vision.rwth-aachen.de/media/resource_files/instances.zip) o en formato .txt (https://www.vision.rwth-aachen.de/media/resource_files/instances_txt.zip). En este caso se ha optado por el formato .txt. ya que son archivos más pequeños y más rápidos de leer. Cada línea de un archivo .txt de instancias está estructurada de la forma en la que se indica en la Tabla 4.2 (donde RLE significa codificación de longitud de ejecución de MS COCO).

   También es necesario descargar los datos de calibración, aunque no se indique, desde http://www.cvlibs.net/datasets/kitti/eval_tracking.php. La estructura de carpetas que el código va a buscar es ./data_dir/images/'sec' siendo 'sec' cada una de las secuencias que a su vez deben tener los directorios ./'sec'/image_2 y ./'sec'/image_3 y el archivo de calibración ./'sec'/'sec'.txt

3. Adaptar los archivos de configuración en ./configs/config_default.py según la configuración que desee para entrenar el código. Esto es, configurar la sección de directorios a las rutas donde se vayan a establecer en el sistema; configurar los parámetros del modelo preentrenado (detecciones RCC o TrackR-CNN), y los parámetros del rastreador. En este caso la variable use_detections es

RCC, use_flow es FlowNet y use_segmentations es BB2SegNet. Los parámetros del rastreador se dejan por defecto. Se utilizan estos parámetros ya que son con los que mejor resultado obtienen.

En algunos casos se sobreentiende que las carpetas intermedias o resultantes están creadas o se van a crear si no existen cuando se ejecuta el código. En el momento de la ejecución se generan bastantes errores porque no localiza las rutas, por lo es recomendable crear todos los directorios que se indiquen dentro del *script*.

4. Descargar la red de segmentación preentrenada. Esta red se puede descargar desde https://drive.google.com/file/d/1Jj3VpAo7WJ-8Tvr7M3XLTA2WrUivvvNA/view y su contenido se debe extraer en ./external/BB2SegNet.

Previo a ejecutar el *script* principal, es necesario haber obtenido las detecciones, segmentaciones, flujo óptico, disparidad y la correspondiente nube de puntos. Para ello hay que ejecutar el siguiente comando:

```
python3 precompute.py -config ./configs/config_default
```

La ejecución obtiene perfectamente las detecciones y la segmentación, y se detiene cuando intenta obtener el flujo óptico y la disparidad. Para este paso es necesario descargar y compilar una serie de librerías que no están incluidas dentro de este repositorio y que necesitan otro tipo de requerimientos que provoca que llevarlo a funcionar sea bastante complejo: netdef_models (https://github.com/lmb-freiburg/netdef_models), lmbspecialops (https://github.com/lmb-freiburg/lmbspecialops/tree/eccv18) y netdef_slim (https://github.com/lmb-freiburg/netdef_slim).

En el propio repositorio se indica:

Nota: Para algunos códigos externos que se requieren para ejecutarse en el script de precomputación, se necesitan requisitos diferentes (ver referencias). Consulte los repositorios correspondientes para obtener los requisitos de los mismos.

Una vez obtenidos los datos, se puede ejecutar el rastreador utilizando el comando:

```
python3 main.py -config ./configs/config_default
```

Después de que el rastreador haya completado todas las secuencias, los resultados se evalúan automáticamente con ayuda de mots_tools.

4.4.1 Resultados obtenidos con EagerMOT

Por último, se comprueba el código del método EagerMOT [76] para la tarea de MOTS. Para este código se desconocen los requisitos utilizados. En este caso se ha utilizado la siguiente configuración HW/SW:

- Sistema operativo Linux Ubuntu 18.04 LTS. Se ha ejecutado el código sobre Google Colab.
- Python 3.7.13

Para el uso de *Google Colab* y *One Drive* es necesario disponer de una cuenta de *Google*. Para trabajar de manera más óptima, se pueden contratar más espacio en la nube de *Drive* (limitado a 25 GB gratis) y contratar *Colab Pro*. Con *Colab Pro* se tiene una serie de ventajas frente a la versión gratuita:

- Acceso prioritario a sus GPUs más rápidas. Por ejemplo, es posible tener acceso a una GPU T4 o P100, mientras que la mayoría de los usuarios de la versión gratuita de *Colab* solo pueden usar una GPU K80, que es más lenta. Para poder usar una GPU con tu cuaderno, hay que seleccionar el menú Entorno de ejecución > Cambiar tipo de entorno de ejecución y, a continuación, elegir GPU en el menú desplegable.

- Con *Colab Pro*, se tiene la posibilidad de acceder a VMs con alta capacidad de memoria si están disponibles. Si se quieren modificar las preferencias del cuaderno para usar un entorno de ejecución con una alta capacidad de memoria, hay que seleccionar el menú Entorno de ejecución > Cambiar tipo de entorno de ejecución y, a continuación, elegir Alta capacidad de RAM en el menú desplegable Características del entorno de ejecución.

- Todos los entornos de ejecución de *Colab* se restablecen tras un periodo concreto, que es más breve si el entorno no está ejecutando ningún código. Pese a que los suscriptores de *Colab Pro* siguen teniendo límites, estos equivalen aproximadamente al doble que los de los usuarios de la versión gratuita.

Para ejecutar un cuaderno de *Google Colab*, se copia el repositorio con los cambios realizados y los resultados obtenidos hasta el momento en la nube de *Google Drive*. Se le llama cuaderno al entorno interactivo que permite escribir y ejecutar código. Los cuadernos de *Colab* son cuadernos de *Jupyter* alojados en *Colab*. Si se selecciona la opción dentro del cuaderno de *Colab* de Activar *Drive*, cada vez que se conecte a un entorno de ejecución, se habilitarán los directorios dentro de ./content/drive/MyDrive. Es necesario tener que modificar las rutas en las que se encontraban los directorios en el equipo para ubicarlos dentro del cuaderno. Para su ejecución, se tiene que repetir el punto 2 de los pasos de configuración para disponer de los mismos paquetes y versiones. Para ejecutar las distintas líneas de comando, hay que añadir el símbolo ! al principio de cada línea. Si se desea navegar los directorios o consultar, hay que añadir el símbolo % al principio de cada línea (cd, ls).

Para poder ejecutar el código, es necesario realizar una serie de pasos previos de configuración:

1. Descargar el código de *Github* (https://github.com/aleksandrkim61/EagerMOT). Para descargar un proyecto de *Github* existen dos opciones:

   - Descargar desde el repositorio. Navegar hasta el repositorio que se quiera clonar, hacer clic en el botón verde y generar el archivo .zip.
   - Descargar desde la consola de comandos. Mediante la consola de comandos, ubicarse en la ruta deseada para clonar. En el repositorio del proyecto, hacer clic en el botón verde y copiar la URL del documento. Usar el siguiente comando:

     ```
     git clone <URL>
     ```

     donde <URL> en este caso en concreto es https://github.com/aleksandrkim61/EagerMOT.git.

     En caso de no tener instalado `git` u otro paquete, se consigue utilizando esta serie de comandos:

     ```
     sudo apt-get update
     sudo apt-get install <package>
     <package> --version
     ```

donde <package> en este caso es git.

2. Compilar e instalar mediante comandos paquetes para Python3. Proporcionan un documento de texto que recoge todos los paquetes necesarios y sus respectivas versiones (requirements_pip.txt). Para instalar un paquete para Python3 basta con escribir este comando:

```
pip3 install <module>
```

donde <module> es el nombre del paquete que se quiere instalar.

Un paso previo que se ha realizado para mayor comodidad es renombrar el comando pip3 a pip.

```
sudo apt-get update
sudo apt-get install python3-pip
pip3 --version
alias pip=pip3
echo alias pip=pip3 >> ~/.bashrc
source ~/.bashrc
```

De modo que la instalación de paquetes/librerías quedaría:

```
pip install -r requirements_pip.txt
```

3. Descargar imágenes e instancias de la base de datos KITTI MOTS. Las imágenes se pueden descargar en http://www.cvlibs.net/download.php?file=data_tracking_image_2.zip y las instancias puedes obtenerlas en dos formatos distintos: en formato .png (https://www.vision.rwth-aachen.de/media/resource_files/instances.zip) o en formato .txt (https://www.vision.rwth-aachen.de/media/resource_files/instances_txt.zip). En este caso se ha optado por el formato .txt. ya que son archivos más pequeños y más rápidos de leer. Cada línea de un archivo .txt de instancias está estructurada de la forma en la que se indica en la Tabla 4.2 (donde RLE significa codificación de longitud de ejecución de MS COCO).

4. Descargar Ego_motion para establecer coordenadas mundiales en la base de datos KITTI MOTS. Estos archivos se pueden descargar en https://drive.google.com/drive/folders/1MpAa9YErhAZNEJjIrC4Ky21YfNj2jatMS. Una vez descargados, se deben copiar en el directorio de trabajo de KITTI que se especifique en el siguiente paso.

5. Modificar el archivo .configs/local_variables.py según la configuración deseada. Hay que indicar la ruta donde de localizan las imágenes de kitti así como crear un directorio de trabajo para ese conjunto de datos, donde se van a almacenar todos los resultados, e indicar su ruta. La variable SPLIT se configura como *training*.

6. Descargar las detecciones 3D que se quieran ejecutar. Se van a utilizar con las que consiguieron mejores resultados que en este caso ha sido con 3D PointGNN. Estas detecciones se pueden descargar desde https://drive.google.com/drive/folders/1MpAa9YErhAZNEJjIrC4Ky21YfNj2jatM?usp=sharing. Es necesario indicar la ruta de estas detecciones en .inputs/utils.py.

7. Descargar las detecciones 2D que se quieran ejecutar. Se van a utilizar con las que consiguieron mejores resultados que en este caso ha sido con 2D MOTSFusion+RRC. Estas detecciones se pueden descargar desde https://drive.google.com/file/d/194Yj_L9_cc5Yio-Khk6DGFOUQ6RS7PvV/view. Es necesario indicar la ruta de estas detecciones en .inputs/utils.py.

   Para utilizar las detecciones MOTSFusion+RCC, es necesario ejecutar un comando previo para copiar la información de detección necesaria en su archivo de segmentación:

   ```
   python3 adapt_kitti_motsfusion_input.py
   ```

   Para que este comando funcione correctamente, es necesario añadir dos líneas más al principio del archivo adapt_kitti_motsfusion_input.py:

   ```
   import dataset\_classes.kitti.mot\_kitti as mot\_kitti
   import inputs.utils as input\_utils
   ```

   También hay que modificar la línea:

   ```
   add\_detection\_info\_to\_motsfusion\_rrc\_segmentations([str(i).zfill(4)
   for i in range(29)])
   ```

   por:

   ```
   add\_detection\_info\_to\_motsfusion\_rrc\_segmentations([str(i).zfill(4)
   for i in range(21)])
   ```

   Y por último, dejar comentada esta línea si no se van a usar las detecciones de TrackRCNN:

   ```
   add\_detection\_info\_to\_motsfusion\_trackrcnn\_segmentations([str(i).
   zfill(4) for i in range(29)])
   ```

Previo a la ejecución del código, hay que abrir el archivo run_tracking.py y modificar la función a la que llama ese archivo en función del conjunto de datos que se desea ejecutar. Para KITTI, la función que hay que seleccionar es run_on_kitti(), la otra función se queda comentada.

Para comenzar el seguimiento hay que ejecutar el siguiente comando:

```
python3 run_tracking.py
```

Se han generado dos resultados distintos: *tracking* 2D proyectado 3D y *tracking* 2D *cleaning* 3D. Aunque se rastrean los objetos en el espacio 3D, se puede informar de los resultados de rastreo 2D proyectando *Bounding Boxes* 3D en el plano de la imagen utilizando los intrínsecos de la cámara e informando de los *Bounding Boxes* 2D alineados con el eje mínimo que encierran completamente esas proyecciones como posiciones 2D de los *tracks*. Estos resultados están ejecutados bajo la tarea de MOT pero puede extenderse a la tarea MOTS. MOTS amplía el MOT con la localización precisa de los *tracks* de los objetos en los píxeles. Se puede adaptar este enfoque MOTS fácilmente pasando adicionalmente máscaras de segmentación de las instancias a los *tracks* después de la asociación de datos. Además de

no aportar de forma directa los resultados para la tarea MOTS, el formato de salida de los resultados es distinto al resto de métodos evaluados. Estos son de la forma:

0 3 Car 0 0 0.681500 -1 -1 -1 -1 1.516600 1.659100 4.149600 24.899800 0.460300 31.645600 1.348100 0.063463

Donde cada valor se traduce como se indica en la Tabla 4.15.

frame	**track id**	**type**	**truncated**	**occluded**	**alpha**	**bbox**			
0	3	Car	0	0	0.681500	-1	-1	-1	-1
dimensions			**location**			**rotation_y**	**score**		
1.516600	1.659100	4.149600	24.899800	0.460300	31.645600	1.348100	0.063463		

Tabla 4.15: Formato resultados tracking EagerMOT

Este formato de salida es válido para evaluarse en el *Benchmark* de KITTI (MOTS *test set*), y no es compatible con mots_tools para comparar los resultados. El resto de códigos sí lo están y sus formatos de salida son como se indican en la Tabla 4.2.

4.5 Comparación de resultados obtenidos y conclusiones finales

En las Tablas 4.16 y 4.17 se reflejan los resultados que han obtenido estos cuatro métodos en la prueba de referencia 2D KITTI MOTS y en la prueba de Evaluación respectivamente.

	Method	HOTA	DetA	ASSA	sMOTSA	MOTSA	MOTSP	IDS	FPS
car	EagerMOT	**74.66**	76.11	**73.75**	74.53	83.5	**89.6**	458	**90**
	MOTSFusion	73.63	75.44	72.39	**74.98**	**84.11**	89.30	**201**	2
	PointTrack	61.95	**79.38**	48.83	78.50	-	-	346	22
	TrackR-CNN	56.63	69.90	46.53	66.97	79.60	85.10	692	2
ped.	EagerMOT	**57.65**	60.30	**56.19**	58.08	72.00	**81.50**	270	**90**
	MOTSFusion	54.04	60.83	49.45	58.75	**72.90**	**81.50**	279	2
	PointTrack	54.44	**62.29**	48.08	**61.47**	-	-	**176**	22
	TrackR-CNN	41.93	53.75	33.84	47.31	66.10	74.60	482	2

Tabla 4.16: Resultados en la prueba de referencia 2D KITTI MOTS

Resultados del artículo									
Type	Method	Det. & Seg.	Speed	Cars			Pedestrians		
				sMOTSA	MOTSA	IDS	sMOTSA	MOTSA	IDS
2D	TRCNN	TRCNN	0.5	76.2	87.8	93	46.8	65.1	78
2D	PointTrack	PointTrack	**0.045**	**85.5**	**94.9**	**22**	**62.4**	**77.3**	**19**
3D	MOTSFusion	RCC+BS	0.44	**85.5**	94.6	35	-	-	-
3D	MOTSFusion	TRCNN+BS	0.84	82.6	90.2	51	58.9	71.9	36
3D	MOTSFusion	TRCNN+TRCNN	0.44	78.2	90.0	36	50.1	68.0	34
Resultados propios									
Type	Method	Det. & Seg.	Speed	Cars			Pedestrians		
				sMOTSA	MOTSA	IDS	sMOTSA	MOTSA	IDS
2D	TRCNN	TRCNN P1	-	76.2	87.8	97	47.0	65.3	73
2D	TRCNN	TRCNN P2	-	76.2	87.9	98	**47.1**	**65.4**	**72**
2D	PointTrack	PointTrack	-	**85.5**	**94.9**	**22**	-	-	-

Tabla 4.17: Resultados obtenidos en la prueba de Evaluación KITTI MOTS. Comparativa entre los resultados del artículo frente a los propiamente obtenidos

En el método TrackR-CNN se enriquecen directamente las detecciones utilizando la información temporal y se aprenden las características de asociación conjuntamente con el detector, en lugar de limitarse a

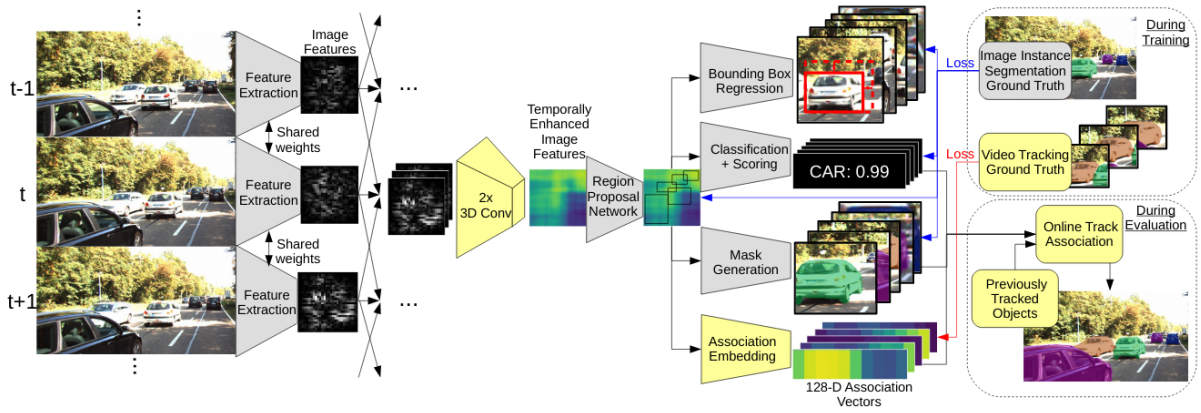


Figura 4.3: Método TrackR-CNN [1]

"postprocesar" las detecciones dadas como en otros métodos. El método propuesto extiende las anotaciones a nivel de *Bounding Boxes* a máscaras de segmentación utilizando una red de refinamiento totalmente convolutiva que toma como entrada un recorte de la imagen de entrada especificada por el *Bounding Box* con una pequeña región de contexto añadida, junto con un canal de entrada adicional que codifica el *Bounding Box* como una máscara. Se itera el proceso de generación y corrección manual de máscaras hasta el nivel de píxeles de precisión.

Los resultados que se consiguen son competitivos superando varias líneas de base donde TrackR-CNN logra puntuaciones sMOTSA y MOTSA más altas, lo que implica que las capas de convolución 3D y la unidad de asociación ayudan a identificar objetos en vídeo. Las puntuaciones de MOTSP siguen siendo similares.

Adicional a esto, sus experimentos de referencia muestran las ventajas que ofrece entrenar en datos de segmentación de instancias temporalmente consistentes en vídeo sobre el entrenamiento en datos de segmentación de instancias sin información temporal y sobre el entrenamiento en datos de segmentación de *Bounding Boxes*. Sus propuestas de conjuntos de datos (entre ellos KITTI MOTS) hacen que se puedan entrenar estas dos vertientes conjuntamente, mientras que antes no era posible. Esto ha abierto muchas oportunidades para la investigación futura.

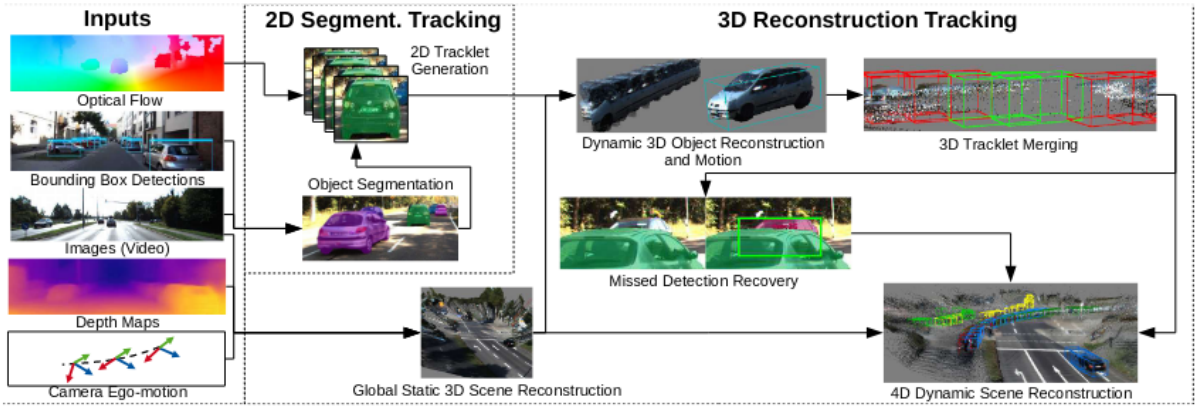


Figura 4.4: Método MOTSFusion [31]

La propuesta de MOTSFusion es abordar el problema del seguimiento de los objetos a largo plazo a través de la desaparición y la oclusión utilizando las reconstrucciones 3D dinámicas a la hora de estimar el movimiento 3D de los objetos. Aprovecha el movimiento 3D extraído de las reconstrucciones dinámicas

de los objetos para rastrearlos durante largos periodos de oclusión completa y recuperar los detectores perdidos. Primero construye *tracklets* cortos utilizando el flujo óptico 2D, y luego los fusiona en reconstrucciones dinámicas de objetos en 3D. El movimiento preciso del objeto en 3D de estas reconstrucciones se utiliza para fusionar los *tracklets* de oclusión en rastros a largo plazo, y para localizar objetos cuando faltan detecciones. Amplía las anotaciones de máscara a nivel de píxel para evaluar la tarea MOTS.

Se muestra que en el conjunto de validación MOTSFusion supera a TrackR-CNN con las mismas detecciones y segmentaciones en 2 puntos porcentuales para los coches (78,2 frente a 76,2) y 3,3 para los peatones (50,1 frente a 46,8) en sMOTSA, y tiene 61 % (36 frente a 93) y 56 % (34 frente a 78) menos de IDS para coches y peatones, respectivamente. Si las detecciones y segmentaciones son RCC+BS, supera a TrackR-CNN en 9,5 puntos porcentuales para los coches (85.5 frente a 76,2) en sMOTSA, y tiene 62 % (35 frente a 93) menos de IDS para coches.

En comparación con otros rastreadores con el mismo hardware, TrackR-CNN tarda 0,50 segundos por fotograma. Por lo tanto, MOTSFusion es capaz de funcionar con una eficiencia similar y con un rendimiento mucho mayor, especialmente para el seguimiento a largo plazo.

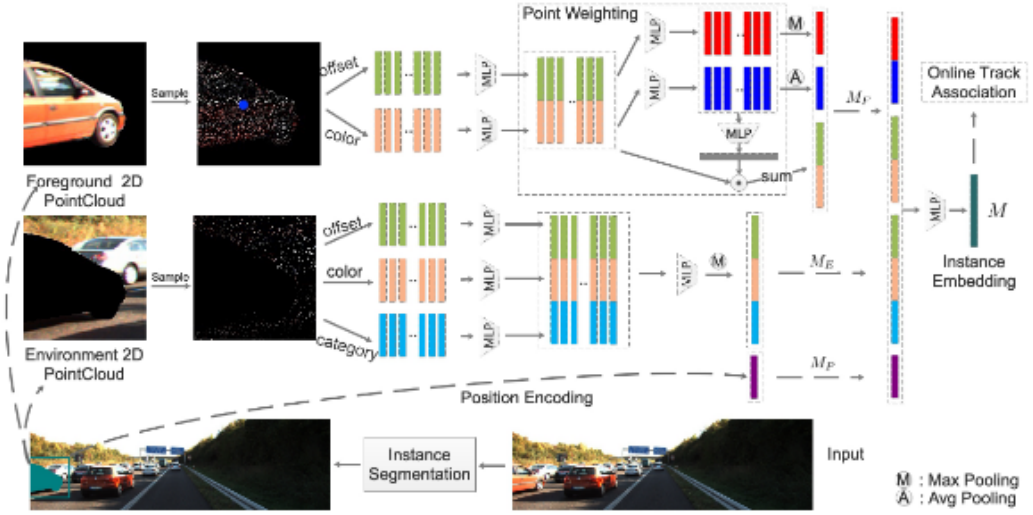


Figura 4.5: Método PointTrack [32].

El método PointTrack propone un método muy eficaz para el aprendizaje de incrustaciones de instancias discriminatorias en segmentos mediante la división de la representación de imágenes compactas en nubes de puntos 2D no ordenadas. Este método permite la agregación de características directamente a partir de nubes de puntos 3D con formato irregular. Concretamente, para cada instancia, se construyen dos nubes de puntos separadas para el segmento del primer plano y el área circundante, respectivamente. En cada nube de puntos, se propone además combinar diferentes modalidades de características puntuales para realizar una incrustación de instancias unificada y consciente del contexto. Para permitir la utilidad práctica de MOTS, mejoran el método de segmentación de instancias de una etapa SpatialEmbedding para la coherencia temporal y construyen un nuevo marco MOTS llamado PointTrack.

Las evaluaciones realizadas en tres conjuntos de datos muestran que PointTrack supera a todos los métodos MOTS anteriores con un amplio margen. PointTrack puede reducir los cambios de *id* y se generaliza bien en la extracción de instancias. El marco que proponen consigue, en primer lugar, un rendimiento casi en tiempo real y supera a todos los métodos de vanguardia, incluidos los métodos en KITTI MOTS con un amplio margen.

PointTrack consigue un rendimiento comparable al del método de seguimiento 3D MOTSFusion (un

0,3 % más que MOTSA) con una mejora significativa de la velocidad (0,045s VS. 0,44s). En el caso de los Peatones, PointTrack supera los enfoques actuales en un 3,5 % y un 5,4 % en sMOTSA y MOTSA respectivamente. Aunque sólo se observan pequeñas mejoras respecto a MOTSFusion para los Coches en el conjunto de validación de KITTI MOTS, PointTrack supera a MOTSFusion por amplios márgenes en el conjunto de pruebas oficial, lo que demuestra la buena capacidad de generalización.

Cuando se aplica a los mismos resultados de segmentación en diferentes conjuntos de datos, PointTrack puede reducir eficazmente el IDS. La reducción constante del IDS en diferentes conjuntos de datos y diferentes resultados de segmentación demuestran la eficacia de PointTrack.

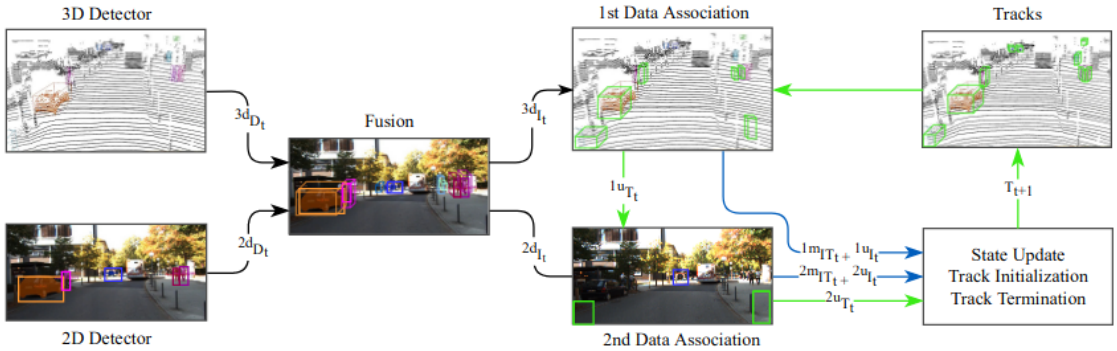


Figura 4.6: Método EagerMOT

EagerMOT combina detectores de objetos LIDAR 3D complementarios que localizan con precisión los objetos en el espacio 3D, y detectores de objetos 2D que son menos susceptibles a las oclusiones parciales y que siguen siendo fiables incluso cuando los objetos están lejos del sensor. Este método trata las diferentes modalidades de sensores de forma independiente. Se rastrean los objetivos en ambos dominios simultáneamente, pero no se acoplan explícitamente sus estados 2D-3D.

A diferencia de MOTSFusion, este método se basa únicamente en las detecciones de objetos de las *Bounding Boxes* obtenidas a partir de dos modalidades de sensor complementarias y se adapta bien a diferentes entornos sensoriales.

Aunque se rastrean los objetos en el espacio 3D, se puede informar de los resultados de rastreo 2D proyectando *Bounding Boxes* 3D en el plano de la imagen utilizando los intrínsecos de la cámara e informando de los *Bounding Boxes* 2D alineados con el eje mínimo que encierran completamente esas proyecciones como posiciones 2D de los *tracks*. MOTS amplía el MOT con la localización precisa de los *tracks* de los objetos en los píxeles. Se puede adaptar este enfoque MOTS fácilmente pasando adicionalmente máscaras de segmentación de las instancias a los *tracks* después de la asociación de datos.

Según la Tabla 4.16, se obtienen mejores resultados en comparación con MOTSFusion en ambas clases (+1,03 para la clase de coches y +3,61 para la de peatones) a pesar de utilizar el mismo conjunto de máscaras de segmentación 2D. Sin embargo, se observa que EagerMOT utiliza adicionalmente detecciones de objetos en 3D obtenidas del flujo LIDAR, mientras que MOTSFusion se basa en las cámaras estéreo. Además, este método se ejecuta a 90 *fps* en KITTI (LIDAR + una sola cámara) en comparación con MOTSFusion a 2 *fps* (cámaras estéreo). Por último, este método establece nuevos resultados del estado del arte tanto para las clases de coches (74,66) como de peatones (57,65) en términos de HOTA. Se comporta especialmente bien en términos de AssA, confirmando una vez más que el uso de observaciones de sensores adicionales ayuda a mantener la consistencia de la pista.

En la Tabla 4.17 también se muestran los resultados obtenidos personalmente y se comparan con los publicados, todos estos obtenidos en la prueba de evaluación (con ayuda de mots_tools), que es la

herramienta utilizada para comparar los distintos métodos en este trabajo. De los cuatro, se han podido completar TrackR-CNN y PointTrack. Ha sido posible reproducir los mismos resultados con el método PointTrack que los publicados. Con TrackR-CNN, además de conseguir los mismos resultados, se han realizado dos pruebas más en las que, cambiando los pesos del modelo gracias al aporte del ajuste de parámetros, obteniendo unos resultados incluso mejores que los presentados en el trabajo de referencia.

Para una mejor comparativa de los resultados entre TrackR-CNN y PointTrack, en las siguientes figuras se muestran dos extractos de la secuencia de vídeo 0018 donde PointTrack logra detectar una furgoneta (Figura 4.7) y donde PointTrack mantiene el seguimiento de un coche durante una oclusión parcial (Figura 4.8).

(a) 000283 PointTrack (b) 000283 TrackRCNN

(c) 000284 PointTrack (d) 000284 TrackRCNN

(e) 000285 PointTrack (f) 000285 TrackRCNN

(g) 000286 PointTrack (h) 000286 TrackRCNN

(i) 000287 PointTrack (j) 000288 TrackRCNN

(k) 000288 PointTrack (l) 000288 TrackRCNN

Figura 4.7: Visualización del seguimiento multi-objeto del vídeo 0018 de la secuencia 000283 a la 000288 del conjunto de datos KITTI MOTS

(a) 000304 PointTrack

(b) 000304 TrackRCNN

(c) 000305 PointTrack

(d) 000305 TrackRCNN

(e) 000306 PointTrack

(f) 000306 TrackRCNN

(g) 000307 PointTrack

(h) 000307 TrackRCNN

(i) 000308 PointTrack

(j) 000308 TrackRCNN

(k) 000309 PointTrack

(l) 000309 TrackRCNN

Figura 4.8: Visualización del seguimiento multi-objeto del vídeo 0018 de la secuencia 000304 a la 000309 del conjunto de datos KITTI MOTS

Capítulo 5

Conclusiones y líneas futuras

5.1 Conclusiones

Como conclusiones finales de este Trabajo Fin de Máster, se puede indicar que se han conseguido los objetivos globales planteados en él.

En concreto, se ha hecho una revisión del estado del arte y de las técnicas a emplear más relevantes. Si bien es cierto que no se ha realizado un estudio en profundidad de las mismas, se ha hecho un gran esfuerzo en la configuración, instalación y puesta en marcha de diferentes técnicas de seguimiento multi-objeto, además de mejorar alguna de ellas. Ya que esto representa el punto base de partida para continuar mejorando y refinando el funcionamiento de estas técnicas de seguimiento.

En total, se han estudiado y ejecutado cuatro métodos que abordan la tarea MOTS que son: TrackR-CNN, PointTrack, MOTSFusion y EagerMOT. En la pruebas de clasificación de referencia empleando la base de datos KITTI MOTS, para coches EagerMOT se encuentra en el tercer puesto del ranking, seguido de MOTSFusion en el cuarto, PointTrack en el noveno y por último TrackR-CNN en el décimo tercer. Para peatones, EagerMOT se encuentra en el quinto puesto, seguido de PointTrack en el noveno, MOTSFusion en el décimo y por último TrackR-CNN en el décimo cuarto

En el capítulo de resultados se han realizado varias configuraciones hardware y software con el objetivo de conseguir el mayor rendimiento de estos cuatro métodos. Hay que tener en cuenta que la mayoría requiere un uso intensivo de capacidades de cómputo del sistema, por lo que en alguna ocasión ha tocado ejecutarlo en una solución de computación en la nube (*Google Colab*) para poder disponer de múltiples unidades de GPU.

Para llevar a cabo una comparación entre los diferentes métodos y similar en cuanto a configuraciones, se han tenido que resolver diferentes problemas de configuración que no estaban resueltos, o bien no documentados, por los propios autores, consiguiendo hacer una comparación lo más fiable posible. Además, después de configurar los diferentes entornos, se ha conseguido mejorar el método TrackR-CNN.

Se ha realizado una comparación tanto teórica como a nivel de resultados obtenidos entre cuatro métodos y se han obtenido unas conclusiones finales.

5.2 Líneas futuras

Las conclusiones obtenidas en este Trabajo Fin de Máster y los resultados del mismo servirán de base para poder ampliar y mejorar el funcionamiento de los cuatro métodos estudiados.

Figura 5.1: Relación de trabajos MOTS

Como mejoras posibles identificadas, se podrían realizar los siguientes:

- Probar diferentes técnicas de seguimiento que se adapten mejor a los modelos de movimiento de los diferentes actores (coches, bicicletas, personas, etc.)
- Realizar un procedimiento de ajuste automático de parámetros que busque mejorar algún parámetro (velocidad de ejecución, porcentaje de aciertos en coches, en peatones, etc.)

En la siguiente imagen se pueden identificar los diferentes trabajos que existen en la materia y que están relacionados entre sí partiendo de [1]. Esta relación se obtiene de la página de referencia: https://www.connectedpapers.com/main/ddb80e2c3e1c2ba012ff33bafaef86f02b7275b0/MOTS-MultiObject-Tracking-and-Segmentation/graph, representando un buen punto de partida para ampliar el estudio a otras técnicas en la materia.

Bibliografía

[1] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe, "Mots: Multi-object tracking and segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[2] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: a survey. acm comput surv," *ACM Comput. Surv.*, vol. 38, 12 2006.

[3] "Redes neuronales artificiales que son y como se entrenan parte i," 2017. [Online]. Available: https://www.xeridia.com/blog/redes-neuronales-artificiales-que-son-y-como-se-entrenan-parte-i

[4] "Tipos de redes neuronales (clasificación)," Ene 2021. [Online]. Available: https://inteligencia-artificial.dev/tipos-redes-neuronales/

[5] "Clasificación de imágenes de tensorflow: Cnn (red neural convolucional)," Mar 2022. [Online]. Available: https://guru99.es/convnet-tensorflow-image-classification/

[6] "Cómo usar redes neuronales (lstm) en la predicción de averías en las máquinas," Nov 2018. [Online]. Available: https://blog.gft.com/es/2018/11/06/como-usar-redes-neuronales-lstm-en-la-prediccion-de-averias-en-las-maquinas/

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.

[8] R. Girshick, "Fast r-cnn," 2015.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[10] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: http://arxiv.org/abs/1703.06870

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015. [Online]. Available: http://arxiv.org/abs/1512.02325

[13] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2018.

[14] G. Ning, Z. Zhang, C. Huang, Z. He, X. Ren, and H. Wang, "Spatially supervised recurrent convolutional neural networks for visual object tracking," 2016.

[15] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr, "Fast online object tracking and segmentation: A unifying approach," 2019.

[16] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2019.00103

[17] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards real-time multi-object tracking," 2020.

[18] A. Osep, W. Mehner, P. Voigtlaender, and B. Leibe, "Track, then decide: Category-agnostic vision-based multi-object tracking," *CoRR*, vol. abs/1712.07920, 2017. [Online]. Available: http://arxiv.org/abs/1712.07920

[19] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[20] "Introducción al conjunto de datos de kitti," 2018. [Online]. Available: https://www.programmerclick.com/article/89471635870/

[21] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, "Mot16: A benchmark for multi-object tracking," 2016.

[22] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, and S. Lyu, "Ua-detrac: A new benchmark and protocol for multi-object detection and tracking," 2020.

[23] S. Manen, M. Gygli, D. Dai, and L. V. Gool, "Pathtrack: Fast trajectory annotation with path supervision," 2017.

[24] M. Andriluka, U. Iqbal, E. Insafutdinov, L. Pishchulin, A. Milan, J. Gall, and B. Schiele, "Posetrack: A benchmark for human pose estimation and tracking," 2018.

[25] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[26] G. Neuhold, T. Ollmann, S. Rota Bulò, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *International Conference on Computer Vision (ICCV)*, 2017. [Online]. Available: https://www.mapillary.com/dataset/vistas

[27] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 724–732.

[28] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. V. Gool, "The 2017 davis challenge on video object segmentation," 2018.

[29] N. Xu, L. Yang, Y. Fan, J. Yang, D. Yue, Y. Liang, B. Price, S. Cohen, and T. Huang, "Youtube-vos: Sequence-to-sequence video object segmentation," 2018.

[30] D. Sun, X. Yang, M. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," *CoRR*, vol. abs/1709.02371, 2017. [Online]. Available: http://arxiv.org/abs/1709.02371

[31] J. Luiten, T. Fischer, and B. Leibe, "Track to reconstruct and reconstruct to track," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, p. 1803–1810, Apr 2020. [Online]. Available: http://dx.doi.org/10.1109/LRA.2020.2969183

[32] Z. Xu, W. Zhang, X. Tan, W. Yang, H. Huang, S. Wen, E. Ding, and L. Huang, "Segment as points for efficient online multi-object tracking and segmentation," *CoRR*, vol. abs/2007.01550, 2020. [Online]. Available: https://arxiv.org/abs/2007.01550

[33] "Seguimiento de objetos," Ene 2021. [Online]. Available: https://es.wikipedia.org/wiki/Seguimiento_de_objetos

[34] "Página del proyecto tech4agecar," http://www.robesafe.uah.es/proyectos/tech4agecar/ [Último acceso 05/abril/2022].

[35] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[36] "Visión artificial)," Sep 2021. [Online]. Available: https://es.wikipedia.org/wiki/Visión_artificial)

[37] "Basics of bounding boxes," Ene 2016. [Online]. Available: https://medium.com/analytics-vidhya/basics-of-bounding-boxes-94e583b5e16c

[38] "Image processing techniques: What are bounding boxes?" Ene 2016. [Online]. Available: https://keymakr.com/blog/what-are-bounding-boxes/

[39] "Segmentación (procesamiento de imágenes)," Mar 2021. [Online]. Available: https://es.wikipedia.org/wiki/Segmentación_(procesamiento_de_imágenes)

[40] "Qué son las redes neuronales y cuál es su aplicación en el marketing," 2021. [Online]. Available: https://artyco.com/que-son-las-redes-neuronales-y-cual-es-su-aplicacion-en-el-marketing/

[41] "Redes neuronales artificiales que son y como se entrenan parte i," 2017. [Online]. Available: https://www.diegocalvo.es/clasificacion-de-redes-neuronales-artificiales/

[42] "Detección y seguimiento de objetos en 2020," 2020. [Online]. Available: https://ichi.pro/es/deteccion-y-seguimiento-de-objetos-en-2020-265049797008115

[43] "Sift," Ene 2021. [Online]. Available: https://es.wikipedia.org/wiki/Scale-invariant_feature_transform

[44] "Surf," Ene 2021. [Online]. Available: https://es.wikipedia.org/wiki/SURF

[45] "Introducción a fast (características de la prueba de segmento acelerada)," 2020. [Online]. Available: https://ichi.pro/es/introduccion-a-fast-caracteristicas-de-la-prueba-de-segmento-acelerada-86669183053157

[46] "Clasificar con k-nearest-neighbor ejemplo en python," 2018. [Online]. Available: https://www.aprendemachinelearning.com/clasificar-con-k-nearest-neighbor-ejemplo-en-python/

[47] "Svm y algoritmos de agrupamiento en algoritmos de aprendizaje automático," 2018. [Online]. Available: https://programmerclick.com/article/5334705809/

[48] "Opencv flann," 2018. [Online]. Available: https://programmerclick.com/article/94052175577/

[49] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, p. 21–37, 2016. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2

[50] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.

[51] "Supresión no máxima (nms)," 2021. [Online]. Available: https://ichi.pro/es/supresion-no-maxima-nms-112304374359350

[52] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[54] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[55] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[56] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2017.

[57] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[58] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," 2017.

[59] G. Seguin, P. Bojanowski, R. Lajugie, and I. Laptev, "Instance-level video segmentation from object tracks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[60] A. Milan, L. Leal-Taixé, and I. Reid, "Joint tracking and segmentation of multiple targets," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2015.

[61] A. Osep, W. Mehner, M. Mathias, and B. Leibe, "Combined image- and world-space tracking in traffic scenes," *CoRR*, vol. abs/1809.07357, 2018. [Online]. Available: http://arxiv.org/abs/1809.07357

[62] S. Sharma, J. A. Ansari, J. K. Murthy, and K. M. Krishna, "Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking," *CoRR*, vol. abs/1802.09298, 2018. [Online]. Available: http://arxiv.org/abs/1802.09298

[63] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu, "Accurate single stage detector using recurrent rolling convolution," 2017.

[64] Y. Jun Koh and C.-S. Kim, "Cdts: Collaborative detection, tracking, and segmentation for online multiple object segmentation in videos," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[65] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[66] F. Li, T. Kim, A. Humayun, D. Tsai, , and J. M. Rehg, "Video segmentation by tracking many figure-ground segments," 2013.

[67] P. Ochs, J. Malik, , and T. Brox, "Segmentation of moving objects by long term video analysis," 2014.

[68] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari, "Learning object class detectors from weakly annotated video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2012, pp. 3282–3289.

[69] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," 2016.

[70] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," 2020.

[71] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The apolloscape open dataset for autonomous driving and its application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, p. 2702–2719, Oct 2020. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2019.2926463

[72] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear mot metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, 2008. [Online]. Available: https://doi.org/10.1155/2008/246309

[73] "Grafo bipartito," https://programmerclick.com/article/4126286151/ [Último acceso 1/septiembre/2021].

[74] A. Kirillov, K. He, R. B. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," *CoRR*, vol. abs/1801.00868, 2018. [Online]. Available: http://arxiv.org/abs/1801.00868

[75] S. Qiao, Y. Zhu, H. Adam, A. Yuille, and L.-C. Chen, "Vip-deeplab: Learning visual perception with depth-aware video panoptic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 3997–4008.

[76] A. Kim, A. Ošep, and L. Leal-Taixé, "Eagermot: 3d multi-object tracking via sensor fusion," 2021.

[77] F. Yang, X. Chang, C. Dang, Z. Zheng, S. Sakti, S. Nakamura, and Y. Wu, "Remots: Self-supervised refining multi-object tracking and segmentation," *CoRR*, vol. abs/2007.03200, 2020. [Online]. Available: https://arxiv.org/abs/2007.03200

[78] Z. Xu, W. Zhang, X. Tan, W. Yang, X. Su, Y. Yuan, H. Zhang, S. Wen, E. Ding, and L. Huang, "Pointtrack++ for effective online multi-object tracking and segmentation," 2020.

[79] Y. Song and M. Jeon, "Online multi-object tracking and segmentation with GMPHD filter and simple affinity fusion," *CoRR*, vol. abs/2009.00100, 2020. [Online]. Available: https://arxiv.org/abs/2009.00100

[80] M. Li, S. Li, L. Li, and L. Zhang, "Spatial feature calibration and temporal fusion for effective one-stage video instance segmentation," 2021.

[81] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," 2017.

[82] "Arquitectura de cnn: ¿cómo funciona resnet y por qué?" 2021. [Online]. Available: https://ichi.pro/es/arquitectura-de-cnn-como-funciona-resnet-y-por-que-30895772711476

[83] Y. Lu, C. Lu, and C.-K. Tang, "Online video object detection using association lstm," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 10 2017, pp. 2363–2371.

[84] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," 2017.

[85] J. Luiten, P. Voigtlaender, and B. Leibe, "Premvos: Proposal-generation, refinement and merging for video object segmentation," *CoRR*, vol. abs/1807.09190, 2018. [Online]. Available: http://arxiv.org/abs/1807.09190

[86] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *CoRR*, vol. abs/1802.02611, 2018. [Online]. Available: http://arxiv.org/abs/1802.02611

[87] L. Castrejon, K. Kundu, R. Urtasun, and S. Fidler, "Annotating object instances with a polygon-rnn," 2017.

[88] M. Andriluka, J. R. R. Uijlings, and V. Ferrari, "Fluid annotation," *Proceedings of the 26th ACM international conference on Multimedia*, Oct 2018. [Online]. Available: http://dx.doi.org/10.1145/3240508.3241916

[89] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[90] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *CoRR*, vol. abs/1703.07737, 2017. [Online]. Available: http://arxiv.org/abs/1703.07737

[91] P. H. O. Pinheiro, T. Lin, R. Collobert, and P. Dollár, "Learning to refine object segments," *CoRR*, vol. abs/1603.08695, 2016. [Online]. Available: http://arxiv.org/abs/1603.08695

[92] E. Ilg, T. Saikia, M. Keuper, and T. Brox, "Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation," 2018.

[93] R. Mur-Artal and J. Tardos, "Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras," *IEEE Transactions on Robotics*, vol. PP, 10 2016.

[94] J. S. J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y. Tai, and L. Xu, "Accurate single stage detector using recurrent rolling convolution," *CoRR*, vol. abs/1704.05776, 2017. [Online]. Available: http://arxiv.org/abs/1704.05776

[95] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," 2017.

[96] D. Neven, B. D. Brabandere, M. Proesmans, and L. V. Gool, "Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth," 2019.

[97] G. Yang and D. Ramanan, "Volumetric correspondence networks for optical flow," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/bbf94b34eb32268ada57a3be5062fe7d-Paper.pdf

[98] L. Qi, L. Jiang, S. Liu, X. Shen, and J. Jia, "Amodal instance segmentation with kins dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[99] L. Porzi, M. Hofinger, I. Ruiz, J. Serrat, S. R. Bulò, and P. Kontschieder, "Learning multi-object tracking and segmentation from automatic annotations," *CoRR*, vol. abs/1912.02096, 2019. [Online]. Available: http://arxiv.org/abs/1912.02096

[100] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *CoRR*, vol. abs/1903.11027, 2019. [Online]. Available: http://arxiv.org/abs/1903.11027

[101] "Información sobre gnu/linux en wikipedia," http://es.wikipedia.org/wiki/GNU/Linux [Último acceso 1/noviembre/2013].

[102] L. Lamport, *LaTeX: A Document Preparation System, 2nd edition.* Addison Wesley Professional, 1994.

Apéndice A

Herramientas y recursos

En general, las herramientas necesarias para la elaboración del proyecto han sido:

- PC compatible
- Sistema operativo GNU/Linux [101]
- Procesador de textos LaTeX[102]

Para TrackR-CNN además ha sido necesario:

- Sistema operativo Linux Ubuntu 18.04 LTS.
- Python 3.6.9.
- Tensorflow 1.13.1.
- Una sola GPU GTX 1650.
- Código de TrackR-CNN obtenido de *Github* (https://github.com/VisualComputingInstitute/TrackR-CNN/tree/master)
- Paquetes y librerias para Python3 como tensorflow-gpu, numpy, scipy, sklearn, pypng, opencv-python, munkres, etc.
- Conjunto de datos KITTI MOTS: imágenes (http://www.cvlibs.net/download.php?file=data_tracking_image_2.zip) e instancias (https://www.vision.rwth-aachen.de/media/resource_files/instances_txt.zip).

Para PointTrack además ha sido necesario:

- Sistema operativo Linux Ubuntu 18.04 LTS.
- Google Colab.
- Python 3.6.9.
- Código de PointTrack obtenido de *Github* (https://github.com/detectRecog/PointTrack)

- Paquetes y librerias para Python3.
- Conjunto de datos KITTI MOTS: imágenes (http://www.cvlibs.net/download.php?file=data_tracking_image_2.zip) e instancias (https://www.vision.rwth-aachen.de/media/resource_files/instances_txt.zip).
- Modelos preajustados de PointTrack (https://drive.google.com/file/d/14Hn4ZztfjGUYEjVd-9FRNB5a-CtBkPXc/view).

Para MOTSFusion además ha sido necesario:

- Sistema operativo Linux Ubuntu 18.04 LTS.
- CUDA 9, cuDNN 7
- Tensorflow 1.13
- Python 3.6
- Una sola GPU GTX 1650.
- Código de MOTSFusion obtenido de *Github* (https://github.com/tobiasfshr/MOTSFusion).
- Paquetes y librerias para Python3.
- Conjunto de datos KITTI MOTS: imágenes (http://www.cvlibs.net/datasets/kitti/eval_tracking.php), instancias (https://www.vision.rwth-aachen.de/media/resource_files/instances_txt.zip) y calibración (http://www.cvlibs.net/datasets/kitti/eval_tracking.php).
- Red de segmentación preentrenada (https://drive.google.com/file/d/1Jj3VpAo7WJ-8Tvr7M3XLTA2WrUivvvNA/view).

Para EagerMOT además ha sido necesario:

- Sistema operativo Linux Ubuntu 18.04 LTS.
- Google Colab.
- Python 3.7.13.
- Código de EagerMOT obtenido de *Github* (https://github.com/aleksandrkim61/EagerMOT).
- Paquetes y librerias para Python3.
- Conjunto de datos KITTI MOTS: imágenes (http://www.cvlibs.net/datasets/kitti/eval_tracking.php) e instancias (https://www.vision.rwth-aachen.de/media/resource_files/instances_txt.zip).
- Ego_motion (https://drive.google.com/drive/folders/1MpAa9YErhAZNEJjIrC4Ky21YfNj2jatMS).
- Detecciones 2D MOTSFusion+RRC (https://drive.google.com/file/d/194Yj_L9_cc5Yio-Khk6DGFOUQ6RS7PvV/view).
- Detecciones 3D PointGNN (https://drive.google.com/drive/folders/1MpAa9YErhAZNEJjIrC4Ky21YfNj2jatM?usp=sharing).

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá