Tracking visual de múltiples objetos con redes convolucionales profundas

Lorenzo Vaquero Otal

Tutores: Manuel Mucientes Molina y Víctor M. Brea Sánchez

Trabajo de Fin de Máster, Universidad de Santiago de Compostela
Máster Universitario en Tecnologías de Análisis de Datos Masivos: Big Data

Resumen

El tracking visual de objetos posee un gran interés en multitud de aplicaciones como la robótica o la videovigilancia. No obstante, mientras que estos campos demandan sistemas capaces de seguir múltiples objetos en tiempo real, gran parte de la investigación en visión por computador se centra en el tracking de un único elemento. Como respuesta a esta necesidad, en este artículo se presenta la arquitectura de un sistema capaz de aplicar eficientemente técnicas de tracking individual a múltiples objetos en tiempo real. Para esto, se propone la extracción global de las características del fotograma mediante una red neuronal convolucional, seguida de un recorte de las distintas áreas de búsqueda de los objetos. La operación de similaridad entre las citadas áreas de búsqueda y la referencia de los objetos a seguir se puede llevar a cabo tanto con una correlación cruzada como mediante una subred de propuestas de regiones. El sistema propuesto ha sido evaluado en distintos conjuntos de datos, reportando tasas de precisión y robustez muy competitivas a la par que alcanza velocidades superiores a las de cualquier otro tracker de múltiples objetos basado en aprendizaje profundo.

1. Introducción

El *tracking visual* de objetos es una disciplina científica perteneciente al campo de la visión por computador que ha recibido una creciente atención durante los últimos años. Su objetivo consiste en mantener la identidad de un objeto a lo largo de los fotogramas de un vídeo, por lo que se emplea en multitud de dispositivos de videovigilancia [53], vehículos autónomos [32] y muchos otros sistemas de análisis de vídeo [66, 36]. No obstante, pese a que recientemente se han realizado numerosos avances, siguen existiendo dificultades a la hora de lidiar con factores como las oclusiones o las deformaciones, resultando en soluciones aún más complejas cuando son varios los objetos a seguir de forma simultánea [59]. Actualmente, para el seguimiento de múltiples objetos simultáneos en vídeos se están utilizando dos aproximaciones muy distintas: el *tracking* mediante detección y la instanciación de múltiples *trackers* individuales basados en métricas de similaridad.

El *tracking* mediante detección consiste en ejecutar un detector en cada fotograma del vídeo que localice los objetos en escena, para después asociar dichas detecciones definiendo trayectorias y asignarles identificadores. Esto se hace posible gracias a los numerosos avances en detección visual de objetos [13, 46, 15, 37], los cuales han permitido la creación de detectores visuales más precisos y veloces. Este proceso de *tracking* se puede realizar tanto en línea (*online*) —empleando los fotogramas previos y el actual, sin la necesidad de consultar imágenes relativas a instantes futuros [58]—, como en diferido (*offline*) —procesando la totalidad del vídeo como un solo lote y obteniendo resultados sólo cuando ha sido analizado al completo—, cada uno con sus ventajas e inconvenientes. Sin embargo, dado que el *tracking* mediante asociación de detecciones tiende a ser lento y requiere que el detector sea previamente entrenado con las categorías a seguir —no se permite el seguimiento de elementos arbitrarios—, se suelen utilizar otras aproximaciones para el *tracking* de objetos individuales.

Los *trackers* visuales individuales se basan en el establecimiento del modelo de aspecto de un objeto arbitrariamente definido, para su comparación con cada uno de los fotogramas del vídeo. Este modelo puede ser generado a partir del propio vídeo, utilizando sus mismos fotogramas como ejemplos [52], o bien de forma general, empleando grandes bases de datos de vídeos etiquetadas [5]. Independientemente de si la similaridad ha sido definida de forma *offline*, *online* o un híbrido de ambas, dos alternativas han sido las que han dominado el estado del arte en *tracking* individual: las basadas en filtros de correlación discriminantes (DCF) y las que emplean técnicas de aprendizaje profundo. Por lo general, los *trackers* visuales individuales destacan por su precisión y velocidad en el seguimiento de objetos arbitrarios. No obstante, sus arquitecturas no permiten el *tracking* de múltiples elementos, por lo que debe instanciarse un *tracker* independiente por cada objeto seguido, provocando un decremento exponencial de la velocidad del sistema.

Para resolver estos problemas y habilitar el *tracking* visual de múltiples objetos arbitrarios simultáneos en tiempo real, se propone el algoritmo SiamMT. SiamMT es un sis-

tema de *tracking* visual totalmente convolucional y siamés, con una extracción de características global en todo el fotograma. La mayor parte de los *trackers* individuales basados en el aprendizaje de métricas de similaridad llevan a cabo el recorte y reescalado de un área de búsqueda sobre el fotograma para localizar el objeto [5, 54, 34, 67, 33, 56]. Esta área de búsqueda de tamaño constante atraviesa una red neuronal convolucional para extracción de características, comúnmente conocida como *backbone*. Tras esto, dichas características se emplean en una capa posterior para la obtención de la posición y dimensiones del objeto. La desventaja de esta aproximación radica en que, en un escenario con múltiples objetos, se deberá realizar este proceso por cada elemento a seguir, lo cual resulta inviable, ya que la extracción de características es la etapa más costosa computacionalmente. De esta forma, aprovechando el hecho de que las áreas de búsqueda de los distintos objetos normalmente se intersecan, este artículo aborda la novedosa aproximación de efectuar una extracción global de las características de todo el fotograma, para realizar un posterior recorte y reescalado de las mismas.

Así, SiamMT se basa en las técnicas de *tracking* visual individual y las modifica para conseguir un seguimiento preciso y en tiempo real de múltiples objetos. Las contribuciones de este artículo se pueden resumir en los siguientes puntos:

1. Se propone SiamMT, una red convolucional siamesa que permite hacer el *tracking* de múltiples objetos en tiempo real y de forma escalable.
2. Para el recorte y reescalado de características obtenidas a través de un *backbone* totalmente convolucional, se define una novedosa formulación basada en RoiAlign [20].
3. Se alcanzan velocidades de inferencia superiores a las de cualquier otro *tracker* de múltiples objetos basado en aprendizaje profundo.
4. El sistema ha sido probado en distintos *benchmarks*, demostrando una robustez y precisión equiparable a las de otros *trackers* individuales.

2. Trabajo relacionado

La contribución principal de este artículo radica en la aplicación de técnicas de *tracking* individual a múltiples objetos de forma eficiente y escalable. Es por esta razón que resulta conveniente realizar un breve análisis previo sobre estas materias.

2.1. Tracking de múltiples objetos

Como se ha mencionado anteriormente, la aproximación por excelencia para el *tracking* de múltiples objetos consiste en la asociación de detecciones [31]. Estos métodos, que por lo general carecen de velocidad debido a los tiempos de ejecución de los detectores y no permiten el seguimiento de objetos arbitrarios, se dividen en sistemas de *tracking offline* y sistemas de *tracking online*.

***Tracking* mediante detección *offline*.** Los *trackers* mediante detección *offline* destacan por su precisión y tolerancia ante errores del detector, siendo la velocidad de procesado la mayor de sus desventajas. Dentro de esta categoría, los métodos más populares a día de hoy son los basados en redes de flujos [44, 3], los basados en modelos probabilísticos en grafos [61, 1, 41] y aquellos que particionan un grafo mediante programación binaria de enteros u optimización de cliques mínimos [29, 65]. También es posible destacar las aproximaciones basadas en discriminación de la apariencia [62, 26], que aprovechan el aspecto del objeto para llevar a cabo las asociaciones, y los algoritmos clásicos de asociación conjunta de datos probabilísticos (JPDA) [48, 14]. En general, este tipo de métodos se suelen aplicar sobre vídeos almacenados en bases de datos con el fin de extraer información adicional, o sobre secuencias capturadas y procesadas a intervalos regulares.

***Tracking* mediante detección *online*.** Por otra parte, los *trackers* mediante detección *online* permiten el procesado de vídeo en *streaming* empleando técnicas de naturaleza más local. Esto les permite alcanzar velocidades de procesamiento cercanas al tiempo real, pero a costa de disminuir su precisión. Dentro de esta categoría, es posible encontrar sistemas que hacen uso de particionamiento de grafos [49], así como de discriminación de la apariencia [42, 8] o de múltiples hipótesis [64]. Además, recientes aproximaciones basadas en aprendizaje profundo están demostrando potentes capacidades de asociación de detecciones, con robustez frente a oclusiones totales prolongadas y reidentificación de los objetivos [40, 63, 57]. Este tipo de *trackers* resultan prometedores para su uso en multitud de aplicaciones, pero aún requieren solventar el problema de la velocidad del detector para permitir su uso en sistemas en tiempo real.

2.2. Tracking de objetos individuales

Los *trackers* visuales individuales definen el modelo de aspecto del objeto arbitrario a seguir, cuya localización y dimensiones son proporcionadas para el primer fotograma de la secuencia. Tras esto, dicho modelo es comparado, fotograma a fotograma, con la región de búsqueda en la que se espera encontrar el objeto, para así actualizar sus coordenadas y tamaño. Esto permite que los *trackers* individuales sean muy precisos y rápidos. Las alternativas predominantes para realizar este *tracking* individual se encuentran basadas en filtros de correlación discriminantes (DCF) o en técnicas de aprendizaje profundo.

***Tracking* individual mediante filtros de correlación discriminantes.** Los trackers basados en DCF predicen la posición del objeto entrenando un filtro que distingue entre el elemento de interés y el fondo de la escena. Sus primeras versiones se centraban en una única característica y un solo filtro por objeto seguido [6]. No obstante, se fueron llevando a cabo mejoras en la precisión con la incorporación de características multidimensionales [11] tales como histogramas de gradientes orientados (HOG) [9] y color, implementando estimadores de escala [10] y otras extensiones como kernels no lineales [22] o componentes de memoria a largo plazo (*long-term*) [38].

***Tracking* individual mediante aprendizaje profundo.** Por otro lado, los *trackers* basados en *deep learning* emplean redes neuronales convolucionales (CNNs) profundas, siendo dos las aproximaciones dominantes: el aprendizaje de métricas de similaridad y el aprendizaje específico del dominio.

Mediante el aprendizaje de métricas de similaridad, se busca entrenar un operador que indique la posición del objeto en un área de búsqueda del fotograma a partir de su aspecto inicial. Una de las primeras arquitecturas basadas en este concepto es SiamFC [5], que utiliza una red siamesa convolucional para la extracción de las características de las imágenes, seguida de un operador de correlación cruzada para su combinación. El principal atractivo de este concepto radica en que permite el seguimiento a altas velocidades de objetos individuales de cualquier clase. Es por esto por lo que han ido apareciendo diversas mejoras sobre la arquitectura original, posibilitando la actualización en línea del modelo [54, 16], empleando técnicas de propuestas de regiones para permitir la regresión del marco delimitador del objeto [34], incluyendo un sistema de supresión de los distractores [67] o incluso incorporando información de segmentación [56].

Relativo al *tracking* mediante aprendizaje específico del dominio, es posible destacar MDNet [43], el cual propone una red unificada con distintas etapas finales adaptadas para cada vídeo de entrenamiento. En el momento de la inferencia, todas las capas específicas del dominio se suprimen, construyéndose una única capa que se ajusta *online* durante el *tracking* para adaptarla al nuevo vídeo.

Cabe subrayar que los avances en la robustez y la exactitud de los sistemas de *tracking* se han ido llevando a cabo a expensas de decrementar su velocidad, partiendo de 172 fps en [9] hasta 2 fps para MFT [2], el ganador del reto Short-Term Visual Object Tracking (VOT) 2018 [27]. Este aumento en el coste computacional presenta serias limitaciones a la hora de usar los *trackers* más avanzados del estado del arte en aplicaciones que requieren de procesamiento en tiempo real. Además, el problema se agrava cuando entre los objetivos se encuentra el de seguir múltiples objetos simultáneamente.

3. Arquitectura de la red

Para el diseño de la arquitectura de la red, se ha tomado como punto de partida el *tracker* SiamFC [5], el cual utiliza métricas de similaridad aprendidas mediante aprendizaje profundo para el seguimiento de objetos individuales a un elevado número de fotogramas por segundo. Para permitir el *tracking* de múltiples objetos, la arquitectura de dicha red ha sido modificada, incluyendo un módulo de recorte y reescalado de las características globales del fotograma. Por último, para ofrecer una solución con una mayor velocidad por objeto, se ha sustituido la operación de fusión de características por la red de propuesta de regiones (RPN) presentada en [34].

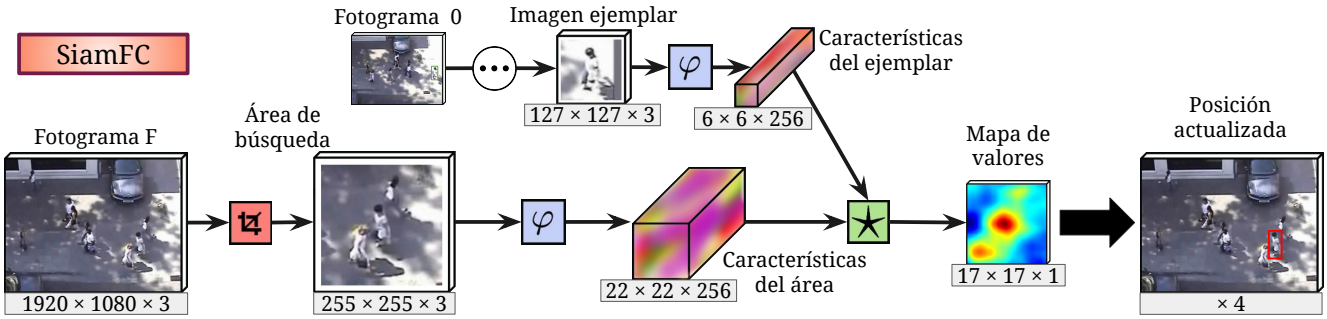
3.1. Arquitectura de red de SiamFC

La arquitectura de red de SiamFC se encuentra descrita en la Figura 1(a). Esta obtiene la posición actual del objeto mediante la comparación de una imagen de ejemplo del mismo con el fotograma actual. Es por esto que la primera acción necesaria para el *tracking* consiste en la creación de dicha imagen ejemplar mediante un operador de recorte y reescalado ⧈. Esta imagen es extraída del primer fotograma en el que se puede reconocer el objeto, y consiste en el área delimitadora que contiene al elemento más un margen de contexto, todo ello escalado a un área de 127×127 píxeles.

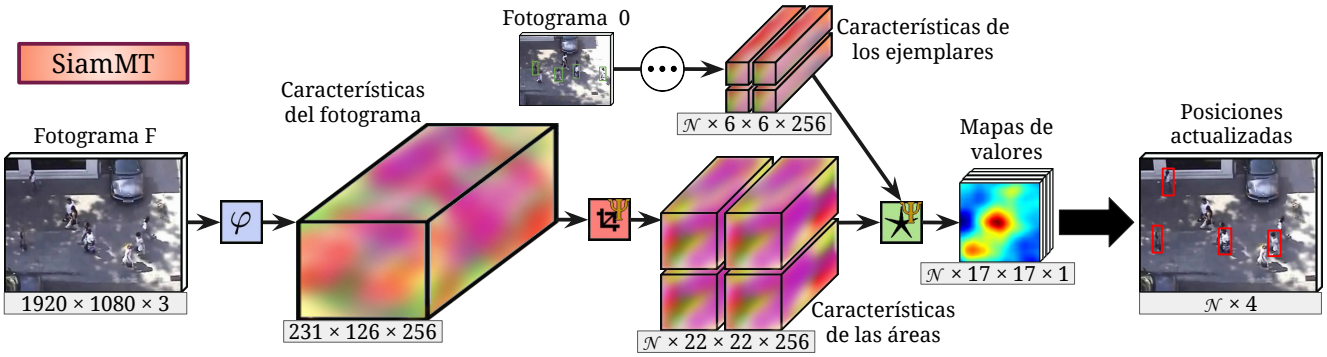
Una vez obtenida la imagen ejemplar, es posible realizar el *tracking* del objeto fotograma a fotograma, el cual se efectúa sobre un área de búsqueda centrada en la última posición conocida del mismo. La extracción de esta área se lleva a cabo de forma muy similar a la obtención de la imagen ejemplar en el primer fotograma, solo que abarcando una zona mayor y siendo escalada a 255×255 píxeles. De este modo, esta imagen reescalada siempre posee la misma proporción con respecto al tamaño del objeto.

Así, partiendo de la imagen ejemplar y del área de búsqueda, se realiza la extracción de sus características mediante φ, una red siamesa y totalmente convolucional —sin *padding*— basada en AlexNet [28]. Los 256 canales de características obtenidos como resultado son comparados mediante un operador de correlación cruzada $\star$, generando un mapa de valores de tamaño 17×17 que indica la probabilidad de que el objeto se encuentre en cada sección del área de búsqueda. El hecho de que no se incluya *padding* en ninguna de las convoluciones es de especial relevancia ya que, si se rellenaran los bordes de los tensores con ceros para que el centro del kernel alcanzara las esquinas, se estaría perdiendo la invarianza estricta frente a traslaciones propia de estas operaciones [33].

Para mejorar la interpretación del mapa de valores, se realiza un sobremuestreo bicúbico del mismo a 272×272 elementos, seguido de una penalización de las zonas más



(a) Arquitectura de red de SiamFC.



(b) Arquitectura de red de SiamMT.

Figura 1: Arquitecturas de red de (a) SiamFC y (b) SiamMT durante la fase de inferencia. SiamMT extrae las características del fotograma al completo mediante un *backbone* φ, lo que le permite aprovechar las computaciones de los objetos cercanos entre sí. Tras esto, se recortan y reescalan las características de las distintas áreas de búsqueda mediante el operador crop^{Ψ}, el cual se basa en RoiAlign [20]. Por último, estas características se combinan con las de sus respectivos ejemplares mediante una correlación cruzada por pares $\star^{\Psi}$, obteniendo unos mapas de valores que indican las nuevas posiciones de los $\mathcal{N}$ objetos. En ambos casos, el alto y el ancho del fotograma de entrada no se encuentran prefijados.

alejadas del centro. De esta forma, seleccionando el elemento de mayor valor de la matriz y trasladando sus coordenadas al fotograma original, es posible obtener la nueva posición del objeto seguido. Con la finalidad de permitir la detección de los cambios de tamaño del objeto, este proceso se lleva a cabo utilizando un área de búsqueda a 3 escalas diferentes.

3.2. Modificación de la arquitectura de SiamFC a múltiples objetos

La arquitectura de red de SiamFC fue modificada para su adaptación a múltiples objetos, tal y como se recoge en la Figura 1(b). A continuación, se destacan los cambios más relevantes.

Extracción global de características. La primera modificación consiste en la supresión del módulo crop de recorte y reescalado de imagen. De esta forma, en SiamMT, el primer paso en el proceso de inferencia consiste en la extracción de todas las características del fotograma mediante el *backbone* φ basado en AlexNet [28]. Este cambio, si bien da lugar a que el coste computacional general sea mayor para un solo objeto —puesto que los fotogramas acostumbran a ser mayores de 255 px^2—, permite la reutilización de características cuando son varios los elementos a seguir. Dicha reutilización resulta crucial para la escalabilidad del sistema, ya que el cálculo de características es la operación más costosa de toda la arquitectura y su realización por objeto resulta inviable.

Recorte y reescalado de características. Tras la extracción de características global, se hace necesario preparar los distintos tensores para su comparación mediante el operador de similaridad. En una primera aproximación, podría parecer suficiente la correlación directa de las características de todo el fotograma con las de cada ejemplar. Sin embargo, esto no resulta factible en un entorno en el que los objetos a seguir sufren cambios de escala. La razón se debe a que los principales *trackers* mediante similaridad [5, 54, 16, 34, 67, 56] poseen una tolerancia relativamente baja frente a las discrepancias entre el tamaño del objeto en la imagen de ejemplo y en el área de búsqueda —

soportando una diferencia máxima de hasta un $15\,\%$ [56]—. Es por esto por lo que se debe ir adaptando la región representada en el área de búsqueda con cada paso de inferencia, incrementándola o decrementándola para que el tamaño del objeto sea siempre constante.

Para permitir una correcta gestión de la escala de los objetos, la aproximación tomada en este artículo consiste en el recorte y reescalado de las características del fotograma que representan el área de búsqueda de cada elemento. Estas operaciones, si bien resultan inmediatas sobre imágenes, poseen una complejidad especial al ser efectuadas sobre características. La primera propuesta para llevar a cabo esta tarea surge con RoiPool [15], que hace el recorte de una región de coordenadas discretas mediante la asignación de un voxel por cada celda —el mayor valor de todos los abarcados en la celda, por ejemplo—. Como sucesora, buscando una mayor precisión, surge RoiAlign [20], que evita las cuantificaciones y calcula los valores de cada celda mediante la agregación —máximo o media— de 4 vóxeles interpolados bilinealmente [25] tomados como muestra. Siguiendo esta línea, la operación crop^{Ψ}, implementada en SiamMT para el recorte y reescalado de características, es una variación de RoiAlign con una única muestra por celda. Esto permite mantener la velocidad de inferencia a la vez que se obtienen valores más precisos y representativos que con RoiPool, ya que el objetivo final es reconstruir la información que se obtendría de la extracción directa de características de cada región reescalada.

Cálculo de coordenadas en características. Para el correcto cálculo de la región de interés, es necesario aplicar una modificación adicional a la operación de recorte de características crop^{Ψ}. En RoiPool y RoiAlign, la transformación entre las coordenadas en la imagen y las coordenadas en características se realiza mediante una simple multiplicación de las coordenadas en píxeles por la escala espacial del *backbone* —calculada como el cociente entre las dimensiones del tensor de salida y el de entrada de la red—. Esto, si bien funciona cuando los extractores de características son arquitecturas como VGG, ResNet o ResNeXt [51, 21, 60], resulta erróneo si la red utilizada carece de *padding*, como es el caso de la AlexNet en [5]. Así, es necesario emplear una transformación adicional, y realizar el cálculo de la escala espacial considerando el tamaño *efectivo* del tensor de entrada.

Denominamos tamaño efectivo de un tensor T a las dimensiones que tendría el tensor de entrada T' que, tras atravesar el *backbone* φ haciendo uso de *padding* tipo *SAME* [12] en todas sus operaciones, produce una salida τ' de las mismas dimensiones que el tensor T al atravesar dicha red φ sin el uso de *padding*. Sabiendo que el *padding* p de tipo *SAME* se calcula siguiendo la lógica mostrada a continuación, para una operación de *stride* s con un kernel k sobre una entrada de tamaño n:

$$p = \begin{cases} max(k-s,0), & \text{si } (n \bmod s) = 0 \\ max(k-(n \bmod s),0), & \text{si } (n \bmod s) \neq 0 \end{cases} \quad (1)$$

es posible demostrar que todo tensor T de tamaño N tendrá un tamaño efectivo de $N-K+S$, siendo K y S los tamaños globales de filtro y *stride* del *backbone* φ, respectivamente. De este modo, dadas unas coordenadas de entrada en píxeles x_i, sus respectivas coordenadas en características x_f se calcularán como se muestra a continuación:

$$x_f = (x_i - \lfloor (K-S)/2 \rfloor) \cdot \frac{(N-K)/S+1}{N-K+S} \quad (2)$$

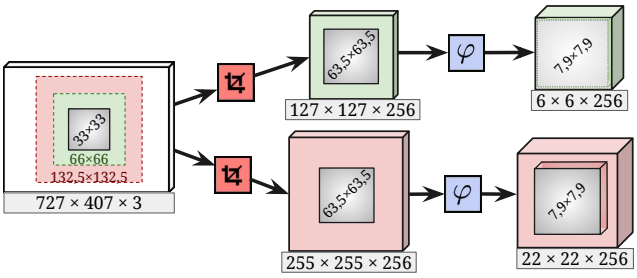
Esta naturaleza en la transformación de coordenadas permite descubrir dos fenómenos que, de lo contrario, pasarían desapercibidos:

- En la arquitectura SiamFC, el tamaño del objeto representado en las características ejemplares es superior al tamaño del tensor que las contiene (Figura 2(a)). Esto significa que, a la hora de realizar las comparaciones, la red únicamente tiene en cuenta una región central del elemento. Si bien los vóxeles de los extremos han tenido acceso a información contextual de las secciones restantes del objeto, su peso podría no ser suficiente.

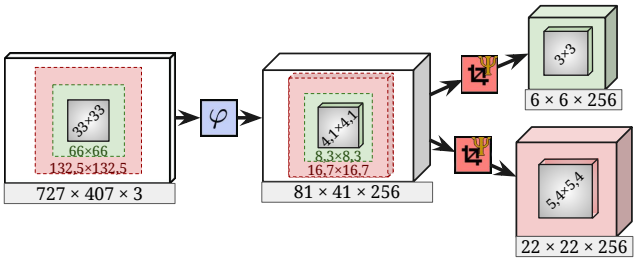
- En la arquitectura SiamMT, no se puede hacer el recorte de las regiones en características utilizando como referencia los tamaños de las mismas en el fotograma (Figura 2(b)). Esto se debe a que el cociente del tamaño de los tensores destino del reescalado es distinto al de SiamFC, lo que haría que en SiamMT el área de búsqueda y el ejemplar tuvieran distintos factores de escala. Como se comentó anteriormente, este tipo de discrepancias en el tamaño de los objetos comparados resultarían altamente perjudiciales para el correcto desempeño de la red.

De este modo, la solución propuesta en SiamMT para resolver estos dos fenómenos consiste en el cálculo de las regiones sobre las características, no sobre el fotograma, y sin utilizar ningún margen para el ejemplar, pues sus vóxeles ya contienen información contextual. Así, tal y como se puede ver en la Figura 2(c), al final del proceso se obtienen 2 objetos del mismo tamaño y sin pérdida de información.

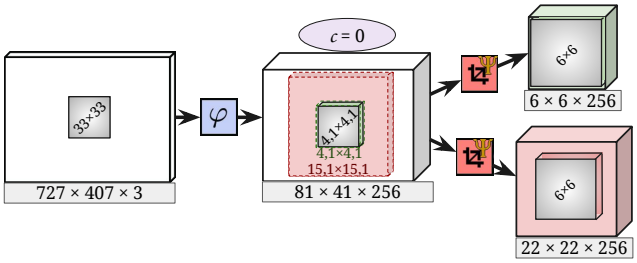
Operador de similaridad. Por último, resulta de especial interés la reformulación de la operación de fusión de características utilizada al final de la arquitectura. Esto se debe a que, tal y como se encuentra definida en la mayor parte de los *frameworks* de *deep learning*, una operación de correlación cruzada va a aplicar un único kernel sobre uno o varios



(a) Evolución del tamaño del objeto en SiamFC.



(b) Evolución del tamaño del objeto en una versión de SiamMT con el cálculo de regiones sobre el fotograma.



(c) Evolución del tamaño del objeto en SiamMT, que calcula las regiones sobre las características del fotograma con un contexto $c = 0$.

Figura 2: Comparativa de la evolución del tamaño de un objeto de 33×33 píxeles para distintas arquitecturas. El objeto y sus características se representan en color gris, mientras que las regiones comprendidas dentro de la imagen ejemplar y del área de búsqueda se representan en verde y rojo, respectivamente.

tensores de entrada. Si se desea llevar a cabo la comparación de varios kernels y tensores 2 a 2, como es el caso en SiamMT, habría que replicar dicha operación a lo largo del tamaño de *batch*, lo cual resulta ineficiente computacionalmente, como se puede ver en la Figura 9(a). Como solución, aprovechando las cualidades de las arquitecturas GPGPU, se propone el uso de la correlación cruzada por pares $\star^{\Psi}$, descrita en el Algoritmo 1.

La operación de similaridad presentada toma como entradas las características de $\mathcal{N}$ ejemplares y las de sus áreas de búsqueda. Cabe destacar que el tensor de ejemplares deberá contar con una dimensión adicional $o_{\mathcal{E}}$. Esta dimensión normalmente tendrá un tamaño 1, pero podrá ser extendida para la obtención de varios canales de salida, lo cual resulta

Algoritmo 1: Correlación cruzada por pares.

Entrada: Un tensor $\mathcal{A}$ con las características de $\mathcal{N}$ áreas de búsqueda, de dimensiones $[\mathcal{N}, h_{\mathcal{A}}, w_{\mathcal{A}}, c_{\mathcal{A}}]$, y un tensor $\mathcal{E}$ con las características de $\mathcal{N}$ ejemplares para obtener $o_{\mathcal{E}}$ canales de salida, de dimensiones $[\mathcal{N}, h_{\mathcal{E}}, w_{\mathcal{E}}, c_{\mathcal{E}}, o_{\mathcal{E}}]$.

Salida: Un tensor $\mathcal{S}$, de dimensiones $[\mathcal{N}, h_{\mathcal{A}} - h_{\mathcal{E}} + 1, w_{\mathcal{A}} - w_{\mathcal{E}} + 1, o_{\mathcal{E}}]$, con el valor de la correlación cruzada entre los tensores de entrada para cada objeto, usando el contenido de $\mathcal{E}$ como kernel.

Algoritmo `XCorrPorPares` $(\mathcal{A}, \mathcal{E})$
- $\mathcal{I} \leftarrow$ `permuta` $(\mathcal{A}, [1, 2, 0, 3])$
- $\mathcal{I} \leftarrow$ `dimensiona` $(\mathcal{I}, [1, h_{\mathcal{A}}, w_{\mathcal{A}}, \mathcal{N} \cdot c_{\mathcal{A}}])$
- $\mathcal{K} \leftarrow$ `permuta` $(\mathcal{E}, [1, 2, 0, 3, 4])$
- $\mathcal{K} \leftarrow$ `dimensiona` $(\mathcal{K}, [h_{\mathcal{E}}, w_{\mathcal{E}}, c_{\mathcal{E}} \cdot \mathcal{N}, o_{\mathcal{E}}])$
- $\mathcal{P} \leftarrow$ `XCorrEnProfundidad` $(\mathcal{I}, \mathcal{K})$
- $\mathcal{S} \leftarrow$ `dimensiona` $(\mathcal{P},$ $[h_{\mathcal{A}} - h_{\mathcal{E}} + 1, w_{\mathcal{A}} - w_{\mathcal{E}} + 1, \mathcal{N}, c_{\mathcal{E}}, o_{\mathcal{E}}])$
- $\mathcal{S} \leftarrow$ `permuta` $(\mathcal{S}, [2, 0, 1, 3, 4])$
- $\mathcal{S} \leftarrow$ `sumaDimension` $(\mathcal{S}, 3)$
- **devolver** $\mathcal{S}$

muy útil cuando se trabaja con *anchors* [47].

La correlación cruzada por pares se hace posible gracias a las propiedades de la correlación cruzada bidimensional en profundidad [17]. Esta toma como entradas un tensor tetradimensional $\mathcal{I}$ y un conjunto de filtros bidimensionales $\mathcal{K}$, y aplica $o_{\mathcal{E}}$ filtros bidimensionales distintos a cada uno de los canales de entrada de cada *batch* de $\mathcal{I}$. Organizando de forma apropiada las características de entrada $\mathcal{A}$ y $\mathcal{E}$ —mediante `permuta`, que reordena las dimensiones de los tensores, y mediante `dimensiona`, que modifica su forma manteniendo sus valores—, es posible obtener unos tensores $\mathcal{I}$ y $\mathcal{K}$ de dimensiones $[1, h_{\mathcal{A}}, w_{\mathcal{A}}, \mathcal{N} \cdot c_{\mathcal{A}}]$ y $[h_{\mathcal{E}}, w_{\mathcal{E}}, c_{\mathcal{E}} \cdot \mathcal{N}, o_{\mathcal{E}}]$, respectivamente. Estos, tras atravesar la operación de correlación cruzada bidimensional en profundidad (`XCorrEnProfundidad`), generarán una salida $\mathcal{P}$ con un único *batch* y $\mathcal{N} \cdot c_{\mathcal{E}} \cdot o_{\mathcal{E}}$ canales, que contendrán el valor de la correlación cruzada por pares sin agregar. Finalmente, tras reorganizar la forma y dimensiones de $\mathcal{P}$, este es sumado a lo largo de su tercera dimensión (`sumaDimension`), lo que se correspondería con la agregación de los $c_{\mathcal{E}}$ canales de cada filtro en una correlación tradicional.

Esta nueva operación, matemáticamente equivalente a aplicar $\star$ por pares, tiene un gran impacto en la velocidad de la arquitectura, permitiendo su escalabilidad a varias decenas de objetos. Si se deseara utilizar *padding* en la correlación, el redimensionado del tensor de salida debería ser

ligeramente modificado.

3.3. Detección de la escala mediante RPN

Puesto que se encuentra basado en la arquitectura de SiamFC, SiamMT hereda su metodología de detección del cambio de escala. Esta consiste en el uso de varias áreas de búsqueda por objeto, cada una de ellas representando al elemento con un tamaño ligeramente diferente. Así, tras la fusión de características, el mapa de valores con las probabilidades más altas indicará la escala en la que se encuentra mejor representado el estado actual de cada objeto.

Dado que se emplean 3 áreas con escalas distintas, todas las operaciones a realizar por cada elemento se verán triplicadas, lo cual supone un coste computacional elevado. Para evitar esto, se propone una variante de SiamMT que sustituye la operación de correlación cruzada final por la red de propuesta de regiones (RPN) presentada en [34]. Con esta arquitectura, se hace innecesaria la creación de múltiples áreas de búsqueda por objeto, pues un regresor se encargará de detectar las variaciones de escala de cada uno de los mismos.

El funcionamiento de una red RPN se basa en el uso de una serie de plantillas denominadas *anchors*. Estas son un conjunto de rectángulos de diferentes escalas y relaciones de aspecto que se deslizan a través del área de búsqueda. De esta manera, el objetivo de la red consiste en predecir la probabilidad de que cada *anchor* contenga o no al objeto y redefinir su tamaño y localización de forma acorde.

Tal y como se puede ver en la Figura 3, la subred RPN utilizada cuenta con 2 ramas: una para la clasificación entre fondo y objeto, y otra para la regresión del cuadro delimitador (*bounding box*). El primer paso a llevar a cabo en cada rama consiste en la aplicación de filtros de convolución $*$ sobre las características extraídas. Siendo k el número de *anchors* considerados por la RPN, se ajusta el número de canales de los ejemplares a $256 \cdot 2k$ para la clasificación y a $256 \cdot 4k$ para la regresión, mientras que el número de canales de las áreas de búsqueda no se varía.

Tras esto, se aplica una operación de correlación cruzada para la fusión de las características de cada rama que, en el caso de SiamMT, se implementa mediante el Algoritmo 1. De este modo, se genera un tensor de $2k$ canales al final de la rama de clasificación —con 2 elementos por *anchor* para codificar la probabilidad de objeto y fondo— y un tensor de $4k$ canales al final de la rama de regresión —con 4 elementos por *anchor* para codificar la posición y tamaño—.

En el extremo de la red, mediante el análisis de los tensores resultantes, se hace posible calcular el nuevo cuadro delimitador de cada objeto seguido. Para esto, primeramente se aplica una función *softmax* sobre los valores de la clasificación, obteniendo las probabilidades de aparición del objeto para cada posición y *anchor*. Seguidamente, los valores de la regresión se combinan con los *anchors* definidos,

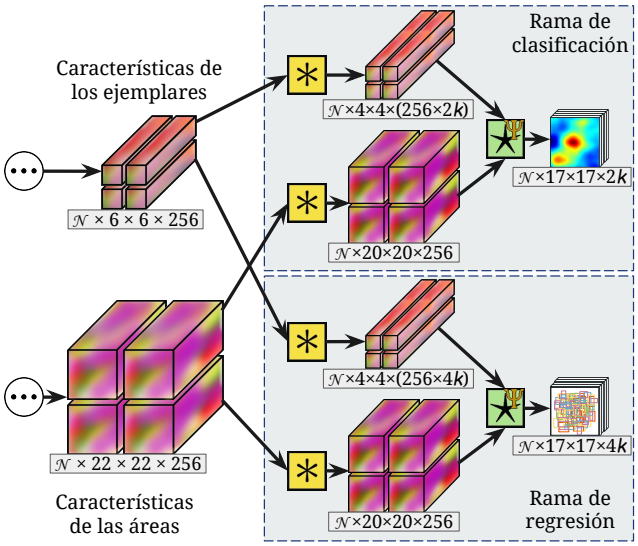


Figura 3: Estructura de la capa RPN. Se compone de dos ramas que toman como entradas las características de $\mathcal{N}$ ejemplares y de sus respectivas áreas de búsqueda. Al final de la rama de clasificación, se obtienen $2k$ valores por cada elemento y posición posible en el área de búsqueda, los cuales codifican la probabilidad para objeto y fondo. Al final de la rama de regresión, son $4k$ los valores reportados por cada localización y objeto, los cuales ajustan la posición y tamaño del *anchor*.

generando las distintas propuestas de la RPN.

Si consideramos el conjunto de los *anchors* como $ANC = \{({x_i}^{an}, {y_i}^{an}, {w_i}^{an}, {h_i}^{an})_{i\in[0,w\cdot h\cdot k)}\}$ y los valores de regresión obtenidos para un objeto como $REG = \{({dx_i}^{reg}, {dy_i}^{reg}, {dw_i}^{reg}, {dh_i}^{reg})_{i\in[0,w\cdot h\cdot k)}\}$, para cada posición y combinación de escala y ratio i, se obtendrá el cuadro delimitador $({x_i}^{pro}, {y_i}^{pro}, {w_i}^{pro}, {h_i}^{pro})$ aplicando las siguientes ecuaciones:

$$\begin{aligned} {x_i}^{pro} &= {x_i}^{an} + {dx_i}^{reg} \cdot {w_i}^{an} \\ {y_i}^{pro} &= {y_i}^{an} + {dy_i}^{reg} \cdot {h_i}^{an} \\ {w_i}^{pro} &= {w_i}^{an} \cdot e^{{dw_i}^{reg}} \\ {h_i}^{pro} &= {h_i}^{an} \cdot e^{{dh_i}^{reg}} \end{aligned} \tag{3}$$

Tras hacer esto y penalizar los grandes desplazamientos y los cambios de escala y forma, se aplica supresión de los no máximos (*non-maximum suppression*) sobre los cuadros delimitadores con la mayor probabilidad de objeto, obteniendo así la nueva posición y tamaño de cada elemento.

3.4. Entrenamiento del sistema

La extracción global de características de un fotograma es un proceso costoso. Es por esto que el entrenamiento del sistema parte de la versión de la arquitectura para *tracking* individual. Así, durante el entrenamiento, se realiza primero

el recorte de las imágenes sobre el fotograma para la posterior extracción de sus características, seguido de su fusión mediante correlación cruzada o RPN. Esto permite no sólo agilizar el proceso de entrenamiento, sino mermar los requisitos computacionales y de almacenamiento, pues se podrá realizar una curación previa de los vídeos y guardar únicamente las imágenes ejemplares y de área de búsqueda ya recortadas y escaladas. En función de la operación de similaridad utilizada, se empleará una función de coste distinta.

Correlación cruzada. Tal y como se describe en [5], la función de coste se calcula como la entropía cruzada sigmoide entre la predicción para un objeto lg y el mapa de valores esperado gt, denominado *groundtruth*:

$$loss = max(lg, 0) - lg \cdot gt + log(1 + e^{-|lg|}) \quad (4)$$

Este *groundtruth* es una matriz bidimensional con los ejemplos positivos en un área de radio R en torno al centro c del tensor. Teniendo en cuenta el *stride* k de la red, el valor de cada elemento de dicha matriz se puede definir de la siguiente forma:

$$y[u] = \begin{cases} +1, & \text{si } k\,\|u - c\| \leq R \\ -1, & \text{si } k\,\|u - c\| > R \end{cases} \quad (5)$$

Puesto que la red es totalmente convolucional, no existe el riesgo de que se vea sesgada hacia el centro del área de búsqueda.

RPN. El cálculo del error cuando se emplea la RPN como operador de similaridad es el descrito en [34]. Para la rama de clasificación, se emplea una función de coste muy similar a la utilizada con la fusión mediante correlación cruzada. En este caso, el *groundtruth* es una matriz tridimensional cuyas clases positivas se corresponden con los *anchors* cuya intersección sobre la unión (IoU) con el objeto es superior a un umbral th_{hi}, fijado a 0,6. Las clases negativas serán aquellas con un IoU inferior a th_{lo}, establecido en 0,3, y el resto de elementos no aportarán valor al coste. La comparación entre *groundtruth* y predicciones se realizará tal y como se ha descrito previamente en la Ecuación 4.

El error en la rama de regresión es el mismo que se utiliza en [47]. Así, siendo A_x, A_y, A_w, A_h el punto central y forma de los *anchors*, y G_x, G_y, G_w, G_h la localización y dimensiones del cuadro delimitador *groundtruth*, los parámetros considerados como correctos se calcularán como se muestra a continuación:

$$\begin{aligned} \delta[0] &= \frac{G_x - A_x}{G_w}, & \delta[1] &= \frac{G_y - A_y}{G_h} \\ \delta[2] &= ln\frac{G_w}{A_w}, & \delta[3] &= ln\frac{G_h}{A_h} \end{aligned} \quad (6)$$

Una vez realizado esto, el error en cada componente se podrá computar como la diferencia x entre los parámetros del *groundtruth* y los valores de predicción, modelado por la función $smooth_{L_1}$ de la Ecuación 7, utilizando σ como hiperparámetro.

$$smooth_{L_1}(x) = \begin{cases} 0{,}5\sigma^2 x^2, & \text{si } |x| < \frac{1}{\sigma^2} \\ |x| - \frac{1}{2\sigma^2}, & \text{si } |x| \geq \frac{1}{\sigma^2} \end{cases} \quad (7)$$

Finalmente, el error total se calcula como la suma del error de clasificación y el error de regresión multiplicado por un parámetro λ de balanceo, que acostumbra a ser 1,2.

***Fine-tuning* de los pesos para inferencia múltiple.** Una vez entrenada la red base, se extraen sus pesos aprendidos y se inyectan en SiamMT, siendo ya posible el *tracking* de múltiples objetos. Sin embargo, puede realizarse un proceso final de ajuste fino —comúnmente conocido como *fine-tuning*— para adaptar las últimas capas de la red a la operación de recorte y reescalado de características, empleando las mismas funciones de coste anteriormente descritas. Para esto, hay que tener en cuenta una consideración adicional, y es que las características representadas dentro del área de búsqueda deberán tener un tamaño con un componente aleatorio. Esto se debe a que, durante el entrenamiento de la arquitectura individual, las imágenes previamente recortadas de áreas de búsqueda y ejemplares pasan por un proceso de *data augmentation*. Este, entre otras transformaciones, las reescala ligeramente, logrando que la red gane tolerancia frente a estos fenómenos. No obstante, durante el entrenamiento de la red final, estas transformaciones se aplican sobre el fotograma al completo, haciendo que la operación de similaridad siempre reciba las características de un área de búsqueda perfectamente recortada y reescalada. Así, de no aplicar una variación aleatoria al proceso de recorte en características, la red aprendería que el objeto a buscar posee un cuadro delimitador con un área constante, cuando durante el *tracking* esto no es así.

4. Experimentos

En esta sección se evalúan las características de SiamMT bajo distintos escenarios. Los experimentos fueron realizados en un sistema con un Intel Xeon E52609, 16 GB de memoria RAM DDR3 y una NVIDIA Quadro P6000. El *framework* de aprendizaje automático escogido fue TensorFlow.

4.1. Detalles de implementación

Extractor de características. La arquitectura concreta del *backbone* utilizado se encuentra recogida en la Tabla 1. Este incorpora una función de activación no lineal ReLU [18] tras cada convolución —a excepción de

Capa	Kernel	Filtros	*Stride*	Canales
input				$\times 3$
conv1	11×11	96	2	$\times 96$
pool1	3×3		2	$\times 96$
conv2	5×5	256	1	$\times 256$
pool2	3×3		2	$\times 256$
conv3	3×3	384	1	$\times 384$
conv4	3×3	384	1	$\times 384$
conv5	3×3	256	1	$\times 256$

Tabla 1: Estructura de capas del extractor de características. Su configuración es similar a la fase convolucional descrita en [28], solo que eliminando los grupos y el *padding*.

la última—, y *batch normalization* [24] durante el entrenamiento tras cada capa. Resulta de especial importancia recalcar que no se introduce *padding* en la red, pues esto daría lugar a que dejara de ser totalmente convolucional [5].

La elección del *backbone* es un aspecto determinante en el rendimiento de la red. Así, si bien otras arquitecturas del estado del arte como ResNet [21] o MobileNet [23] permitirían obtener resultados más precisos, supondrían una velocidad de inferencia mucho inferior, equiparable a la de un detector. Esta es la razón por la que se ha escogido un extractor de características inspirado en AlexNet [28], puesto que ofrece el *backbone* basado en aprendizaje profundo con el menor coste computacional de los propuestos hasta la fecha [7].

Tamaño del ejemplar y del área de búsqueda. Si se denota el tamaño del cuadro delimitador de un objeto como (w, h), la imagen ejemplar del mismo se corresponderá con la región de tamaño $A \times A$ centrada en su localización. Considerando un factor de contexto c, este tamaño se define tal y como se muestra a continuación:

$$(w + c(w + p)) \times (h + c(w + p)) = A^2 \qquad (8)$$

Tras llevar a cabo el recorte de la región del ejemplar, esta es reescalada al tamaño del tensor destino. En los casos en los que el recorte y reescalado se efectúan sobre una imagen, c se fija a $0{,}5$ y el reescalado se hace a 127×127 píxeles. No obstante, cuando los cálculos se hacen directamente sobre características, un c de 0 con un tensor destino de 6×6 unidades son los valores de los hiperparámetros que ofrecen los mejores resultados.

De forma muy similar, el tamaño del área de búsqueda se calcula como el cociente entre su tensor destino y el tensor destino para el ejemplar, multiplicado por la región de recorte del ejemplar. En los casos en los que el recorte se hace sobre una imagen, el tensor destino tiene un tamaño de 255×255, mientras que con recortes sobre características, se escoge un tensor de 22×22 unidades.

Proceso de entrenamiento. Para el entrenamiento del sistema, se ha empleado la base de datos de vídeos facilitada por el ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [50], la cual contiene cerca de 5.000 secuencias, con un total de más de un millón de fotogramas etiquetados. Si bien esta parece una gran cifra, resulta una cantidad de imágenes moderada si se compara con otros *datasets*. Es por esto que el siguiente paso en la evolución del sistema consistirá en incorporar al entrenamiento las bases de datos Youtube-BB [45], ImageNet Detection [50] y COCO Detection [35], multiplicando por 8 el número de ejemplos actual. Durante el entrenamiento, se emplea Descenso de Gradiente Estocástico (SGD) con momento para la minimización de la función de error, a lo largo de 50 épocas con 50.000 ejemplos cada una. En el caso de la operación de similaridad mediante correlación cruzada, se utilizan *batches* de tamaño 8, partiendo de una tasa de aprendizaje de 10^{-2} que es disminuida exponencialmente hasta 10^{-5}. Para la fusión de características mediante RPN, se utilizan *batches* de tamaño 16, con una tasa de aprendizaje disminuida exponencialmente desde 10^{-2} hasta 10^{-6}. En ambos casos, se parte de un *backbone* preentrenado para el etiquetado de ImageNet, lo cual facilita el aprendizaje del modelo.

Proceso de inferencia. La fase de inferencia busca ser lo más minimalista posible, para así alcanzar altas velocidades en el *tracking*. Para lograr esto, las características de cada ejemplar se extraen al comienzo del proceso y se reutilizan a lo largo del mismo. De este modo, se reduce en gran manera la cantidad de operaciones a realizar, además de que esta aproximación permite formular el *tracker* como un *one-shot detector*. Adicionalmente, se ha podido comprobar que la actualización del ejemplar con métodos simples no supone una mejora sustancial en la calidad del *tracking*.

Cuando se emplea la correlación cruzada como método de fusión de características, se realiza la búsqueda del objeto a lo largo de 3 escalas, con una diferencia de un $3{,}75\,\%$ entre sí. En el caso de la RPN, se utilizan *anchors* con ratios de $[0{,}33; 0{,}5; 1; 2; 3]$ a una única escala, pues el tamaño del objeto no cambiará sustancialmente de un fotograma al siguiente. En ambos casos, se emplea un factor de amortiguamiento del $35\,\%$ para suavizar las actualizaciones de tamaño.

4.2. Evaluación de la calidad de *tracking*

El *tracking* por detección se encuentra fuertemente arraigado en el ámbito del seguimiento de múltiples objetos. Tanto es así que todos los *benchmarks* de esta índole ofrecen un conjunto de detecciones como base, siendo el reto la correcta asociación de las mismas [4, 30, 39]. Puesto que SiamMT no hace uso de información de detección para su funcionamiento, su evaluación se ha realizado sobre *benchmarks* de *tracking* individual. Concretamente, se han

considerado OTB-2015 y VOT-2018 [59, 27].

Así, se ha comparado la eficacia de SiamMT utilizando correlación cruzada (SiamMT-FC) y la capa RPN (SiamMT-RPN) frente a las redes SiamFC y SiamRPN [5, 34]. La razón detrás de esta elección se debe a que estas constituyen las arquitecturas base sobre las que se fundamenta SiamMT, y suponen un buen punto de referencia con el que comprobar las implicaciones de la extracción global de características y su reescalado. Para asegurar una comparación justa, todas las redes han sido entrenadas sobre el conjunto de datos ImageNet Video [50].

OTB-2015. El *benchmark* estandarizado OTB [59] se compone de 100 vídeos distintos de situaciones cotidianas, y proporciona un marco idóneo en el que evaluar la robustez de un *tracker*. Principalmente, esto se debe a que no reinicializa el sistema en caso de pérdida del objeto, simulando fielmente un entorno real de seguimiento no supervisado. Por lo tanto, al penalizar tan duramente las distracciones, resulta ideal para el análisis de un sistema de *tracking* de múltiples objetos.

Para el cálculo de la calidad de un modelo, OTB considera la proporción media de aciertos por fotograma en diferentes umbrales: un *tracker* resulta exitoso en un fotograma concreto si la intersección sobre la unión (IoU) entre la predicción y el *groundtruth* se encuentra sobre un determinado umbral. Los *trackers* son comparados en función del área bajo la curva del porcentaje de aciertos para diferentes valores de este umbral.

Los resultados obtenidos para el OPE (*one pass evaluation*) se encuentran recogidos en la Figura 4. Tal y como se puede ver, SiamFC lidera la gráfica con un área bajo la curva (AUC) de $59{,}70$ puntos, seguida de cerca por SiamRPN, con $54{,}15$. Por debajo de estas y con una diferencia relativa similar, se encuentran SiamMT-FC y SiamMT-RPN con un AUC de $44{,}56$ y $40{,}44$ puntos, respectivamente. Como referencia, el *tracker* mejor valorado del OTB-2015 fue Struck [19], con un área bajo la curva de $46{,}15$ puntos.

Si bien SiamMT no es tan precisa como las redes originales, esta diferencia es pequeña, encontrándose dentro de lo aceptable en aplicaciones de *tracking* múltiple. Así, pese a que el ajuste del cuadro delimitador no sea el óptimo, su robustez frente a distractores y velocidad por objeto la hacen una solución muy apta para su aplicación en entornos reales.

VOT-2018. El *benchmark* VOT-2018 [27] es uno de los conjuntos de datos más recientes para la evaluación de *trackers* individuales, incluyendo 60 secuencias distintas en las que se representan variadas situaciones. Dentro del *benchmark*, los *trackers* son automáticamente reinicializados a cinco fotogramas en el futuro cada vez que existe un fallo, es decir, cada vez que la intersección sobre la unión

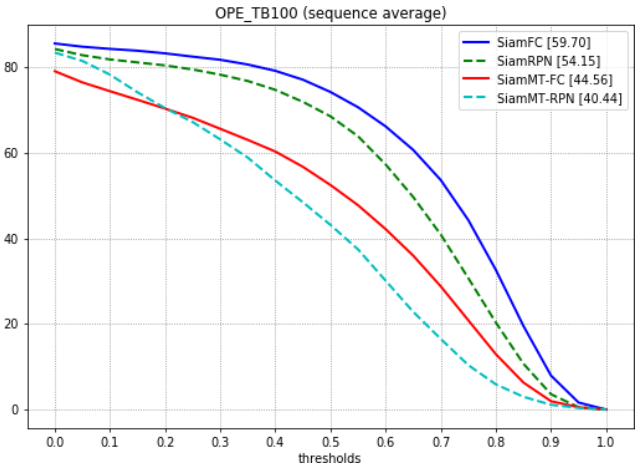


Figura 4: Gráfico OTB-15 de los resultados de acierto para OPE (*one pass evaluation*).

(IoU) entre la predicción y el *groundtruth* es cero. A la hora de presentar los resultados, se tienen en cuenta la precisión (*accuracy*) y la robustez (*robustness*), las cuales son calculadas como la IoU media y el inverso de la razón de fallos, respectivamente. En el VOT, también se valora la superposición esperada promedio (*average expected overlap*), la cual considera la precisión y el número de fallos para obtener la IoU media del *tracker* sin reinicializaciones.

Los resultados obtenidos de precisión-robustez son los mostrados en la Figura 5. Tal y como se puede ver, SiamFC tiene la mayor robustez, mientras que SiamRPN ofrece una excelente precisión, gracias a su regresión del cuadro delimitador. Asimismo, se puede apreciar que, si bien la arquitectura SiamMT presenta mayores dificultades a la hora de ajustar la *bounding box*, posee una alta robustez, muy superior a la de SiamRPN.

La pérdida en precisión de SiamMT se debe principalmente a la extracción global de características, que da lugar a que los detalles más pequeños de la imagen se desvanezcan debido al *stride* global de 8 de la red. Esto podría solventarse parcialmente mediante el establecimiento de un tamaño mínimo de fotograma. Por otra parte, la excelente robustez de la solución demuestra la efectividad de la operación de recorte y reescalado de caractarísticas, la cual genera una información equiparable a la obtenida a partir de una imagen previamente reescalada.

Por último, se puede ver cómo SiamMT-RPN ha obtenido una mayor robustez y menor precisión que SiamMT-FC. Esto viene dado por la función de coste compuesta de SiamMT-RPN, que le permite asignar distintos pesos a la regresión y a la clasificación. En este caso, puesto que la robustez frente a distractores es una de las características más esenciales en *tracking* múltiple, se ha decrementado el valor de λ, haciendo que aumente la importancia de la clasificación a cambio de una pequeña pérdida en la regresión.

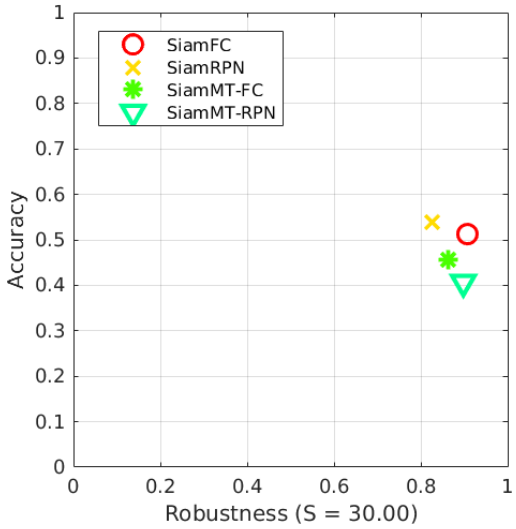


Figura 5: Gráfico VOT-18 de los resultados de precisión (*accuracy*) y robustez (*robustness*).

4.3. Evaluación de la velocidad de *tracking*

En el *tracking* de objetos, la velocidad de inferencia es un aspecto clave, pues es lo que determina si un sistema es viable para su aplicación en entornos reales o no. El objetivo principal de SiamMT es habilitar el seguimiento de un gran número de objetos de forma escalable a altas velocidades, algo que hasta ahora no ha sido logrado a través del *tracking* múltiple mediante detección ni utilizando múltiples instancias de *trackers* individuales. Es por esto que el análisis de los tiempos de computación de las distintas operaciones de la arquitectura de red constituye un ejercicio de gran relevancia.

La evaluación de la velocidad de SiamMT se ha propuesto de forma que se encuentren representados los escenarios más típicos en el *tracking* de múltiples objetos. Así, se han diseñado dos conjuntos de pruebas distintos: el *benchmark* MT-Low, compuesto de vídeos de baja resolución con un tamaño de 489×360 píxeles, y el *benchmark* MT-FHD, con vídeos FHD de 1920×1080 píxeles.

Para demostrar la escalabilidad de la solución propuesta, se compararán las arquitecturas de SiamMT con fusión de características mediante correlación cruzada (SiamMT-FC) y mediante RPN (SiamMT-RPN) frente a una versión modificada del algoritmo SiamFC (SiamTF [55]). Este algoritmo permite el *tracking* de múltiples objetos a través de la ampliación del tamaño de *batch*, sin la reutilización de cómputos, por lo que se podría clasificar como un *tracker* individual especialmente optimizado para su múltiple instanciación. No se ha incluido ninguna variante de SiamRPN en la comparación, debido a que esta no se encuentra diseñada para el *tracking* de múltiples objetos.

Adicionalmente, cabe destacar que SiamMT permite la extracción parcial de las características del fotograma, contemplando únicamente la región mínima en la que se encuentran los objetos seguidos. Esto ofrece grandes ganancias en velocidad cuando los elementos a seguir se encuentran cercanos entre sí, haciendo que con un único objeto en escena se alcancen velocidades superiores a los 40 fps en vídeos FHD. No obstante, si bien esta técnica supone una importante ventaja competitiva, introduce ruido en las mediciones de tiempos al depender de las posiciones relativas de los objetos. Es por esta razón que se ha optado por no utilizarla en los *benchmarks* de tiempos, para así generar unos resultados más estables y fieles, que dejen al descubierto las verdaderas capacidades de escalabilidad de la red.

***Benchmark* MT-Low.** La comparativa de resultados para el *benchmark* MT-Low se encuentra recogida en la Figura 6. Tal y como se puede ver, para un único objeto a seguir, SiamMT-RPN y SiamTF ofrecen velocidades superiores a los 60 fps, con esta última dominando el gráfico, alcanzando los 72 fotogramas por segundo. Esto se debe a que, al tratarse de imágenes de baja resolución, la extracción global de características no difiere mucho de la inferencia utilizando áreas de búsqueda. En el caso de SiamMT-FC, su velocidad de inferencia para un objeto es algo menor, debido a que debe efectuar el recorte y reescalado de 3 áreas de búsqueda sobre las características del fotograma.

Conforme se van agregando objetos, la velocidad de *tracking* de SiamTF se reduce exponencialmente, pasando a algo menos de 25 fps para 5 elementos. Entretanto, la escalabilidad de la arquitectura de SiamMT se pone de manifiesto, permitiendo que SiamMT-RPN siga operando cerca de los 60 fps, mientras que SiamMT-FC únicamente reduce su velocidad hasta 39 fotogramas por segundo al alcanzar los 5 objetos.

A partir de este punto, SiamTF persiste en su decremento en velocidad, reportando 4 fps para 30 objetos, hasta su colapso antes de llegar a los 40 elementos. Este colapso se debe a la incapacidad del hardware utilizado de extraer simultáneamente las características de 120 áreas de búsqueda. No obstante, SiamMT-RPN y SiamMT-FC sí hacen posible el seguimiento de estas cantidades de objetos, permitiendo el *tracking* de 40 elementos a 25 y 17 fotogramas por segundo, respectivamente.

Así, SiamMT-FC es capaz de seguir en tiempo real (25 fps) hasta 19 elementos, mientras que SiamMT-RPN incrementa esta cifra hasta los 40. La diferencia en velocidad entre estos dos *trackers* se ve disminuida paulatinamente conforme se aumenta el número de objetos, pero con SiamMT-RPN siempre por encima. Al final del *benchmark*, con 100 objetos a seguir, SiamMT-RPN y SiamMT-FC obtienen una velocidad de 13 y 9,2 fotogramas por segundo, respectivamente.

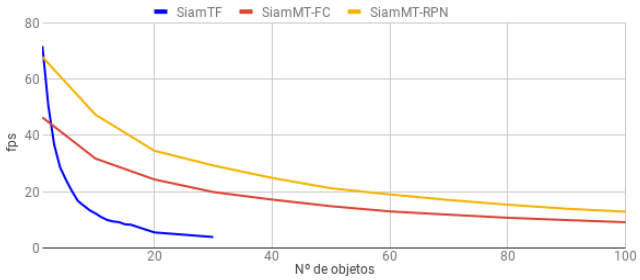


Figura 6: Velocidad de inferencia para el *benchmark* MT-Low.

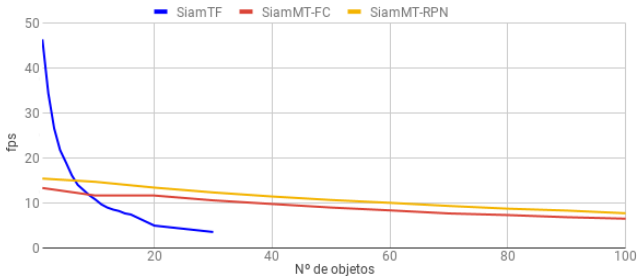


Figura 7: Velocidad de inferencia para el *benchmark* MT-FHD.

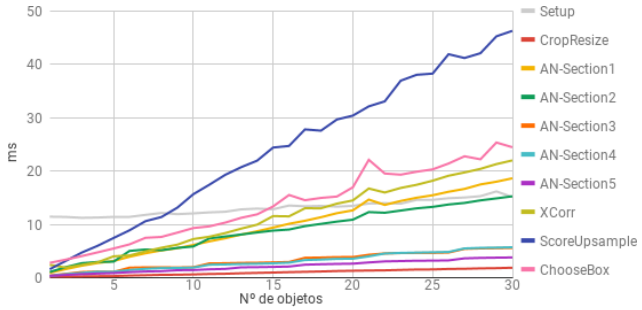
***Benchmark* MT-FHD.** La comparativa de resultados para el *benchmark* MT-FHD se encuentra recogida en la Figura 7. De forma similar a como ocurría en el *benchmark* MT-Low, SiamTF decae su velocidad exponencialmente, obteniendo los mejores resultados cuando el número de elementos a seguir es bajo. En este caso, permite una velocidad de inferencia en tiempo real para hasta 3 objetos. Poco después de este punto, al sobrepasar los 10 elementos simultáneos, los fotogramas por segundo de SiamTF caen por debajo de los de SiamMT, hasta su colapso antes de alcanzar los 40 objetos. Como se puede ver al comparar las Figuras 6 y 7, SiamTF obtiene unos tiempos de inferencia menores cuando procesa vídeos FHD, pese a que no realiza extracción global de características. Esto se debe al coste temporal en la transferencia de los fotogramas desde memoria RAM hasta memoria de GPU, el cual se ve incrementado cuando las imágenes a mover cuentan con una resolución mayor.

En lo relativo a SiamMT-RPN y SiamMT-FC, estas poseen una velocidad de inferencia similar entre sí, pero con SiamMT-RPN por encima en todo momento. Dado que deben extraer las características de la totalidad del fotograma, presentan un notorio coste computacional general, resultando en 15,5 y 14,4 fps para un único objeto, respectivamente. No obstante, si se aplicara un recorte de la región mínima en la que se encuentran los elementos, como se describió anteriormente, estas velocidades se verían incrementadas enormemente hasta los 40 fotogramas por segundo.

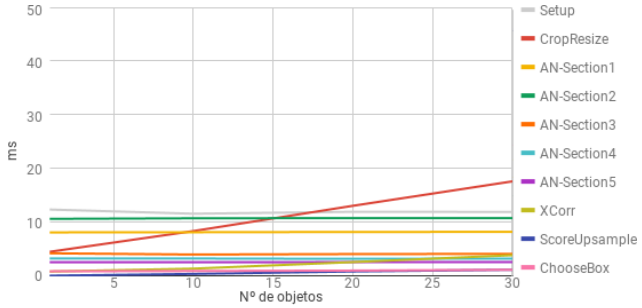
Puesto que la extracción global de características del fotograma es el proceso más costoso de la arquitectura de SiamMT, su velocidad en vídeos FHD no se ve afectada prácticamente al añadir nuevos objetos. Así, SiamMT-RPN y SiamMT-FC permiten seguir 60 objetos a 10,1 y 8,4 fps, respectivamente, y alcanzan el final del *benchmark* (100 objetos) manteniéndose a 7,8 y 6,7 fotogramas por segundo.

Análisis del coste por operación. La Figura 8 recoge una comparativa del coste de las distintas capas de SiamTF y SiamMT-FC para el *benchmark* MT-FHD en función del número de objetos seguidos. Tal y como se puede ver, la mayor parte de las capas de SiamTF presentan un tiempo de ejecución variable en función de $\mathcal{N}$, al contrario que las de SiamMT-FC:

- **Carga de los tensores en GPU (`Setup`).** En ambas redes se lleva a cabo la misma carga de datos, con la diferencia de que SiamMT-FC la implementa de un modo más eficiente.
- **Recorte y reescalado de las áreas de búsqueda (`CropResize`).** Puesto que SiamMT-FC tiene que hacer un recorte y reescalado sobre características, con una lógica adicional y más canales involucrados que en el caso de $\boxplus$, el coste de $\boxplus^{\Psi}$ es más sensible frente al número de objetos seguidos.
- ***Backbone* (`AN`).** SiamTF y SiamMT-FC utilizan el mismo *backbone* φ basado en AlexNet. No obstante, mientras que SiamTF debe ejecutarlo para extraer las características de $3\mathcal{N}$ áreas de búsqueda, SiamMT-FC lo emplea una única vez para todo el fotograma, lo cual resulta más rápido cuando $\mathcal{N}$ es elevado.
- **Operación de similaridad (`XCorr`).** Como se menciona en la Sección 3.2, SiamMT-FC utiliza una operación de similaridad especialmente diseñada para la comparación a pares entre tensores ($\star^{\Psi}$), lo que hace que tenga un coste casi constante, muy distinto del ofrecido en SiamTF con $\star$. Esto se debe a que $\star$ requiere de una operación adicional por objeto seguido, mientras que $\star^{\Psi}$ hace la totalidad de la comparación dentro de la misma correlación cruzada en profundidad, como se puede ver en la comparativa de la Figura 9.
- **Sobremuestreo de los mapas de valores (`ScoreUpsample`).** El sobremuestreo de los mapas de valores en SiamMT-FC se lleva a cabo en GPU, mientras que SiamTF lo realiza en CPU, lo cual resulta menos paralelizable.
- **Selección de los nuevos cuadros delimitadores (`ChooseBox`).** De forma muy similar a como ocurre con `XCorr`, SiamMT-FC implementa cálculos tensoriales en paralelo, mientras que SiamTF lo hace de forma secuencial.

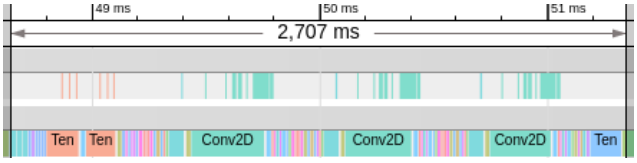


(a) Tiempo por capa de SiamTF en función del número de objetos.

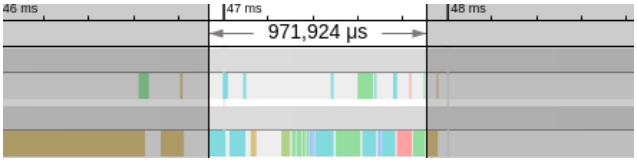


(b) Tiempo por capa de SiamMT-FC en función del número de objetos.

Figura 8: Comparación de los tiempos medios por capa en el *benchmark* MT-FHD para (a) SiamTF y para (b) SiamMT-FC, en función del número de objetos seguidos. En SiamTF, el coste de la mayor parte de las capas aumenta a medida que crece el número de objetos. Mientras, en SiamMT-FC, pocas son las capas cuyo tiempo depende de $\mathcal{N}$, siendo la operación de recorte y reescalado ⧉^{Ψ} la única que acaba superando el umbral de los 15 ms.



(a) Correlación cruzada replicada para 3 objetos.



(b) Correlación cruzada por pares para 3 objetos.

Figura 9: Desglose de operaciones para la fusión de características mediante (a) correlaciones replicadas y mediante (b) correlación cruzada por pares. Las correlaciones replicadas $\star$ introducen cálculos adicionales al incrementarse el número de objetos seguidos. No obstante, el número de operaciones empleadas por la correlación cruzada por pares $\star^{\Psi}$ se mantiene constante.

5. Conclusiones

En este artículo se ha propuesto un sistema completo de *tracking* visual de múltiples objetos, denominado SiamMT. Este aplica técnicas de seguimiento individual a múltiples objetos de forma eficiente y escalable, algo que no había sido desarrollado hasta la fecha. Como habilitadores, se introducen el concepto de extracción global de características y dos nuevas operaciones: ⧉^{Ψ}, capaz de recortar y reescalar características independientemente del *backbone* utilizado, y $\star^{\Psi}$, que efectúa una correlación cruzada por pares de forma eficiente en GPU.

El sistema puede emplear tanto una correlación cruzada como una subred RPN para la fusión de características, los cuales han sido probados en distintos *benchmarks*, demostrando una excelente robustez a la vez que son capaces de seguir varias decenas de objetos en tiempo real. Asimismo, la calidad de seguimiento del *tracker* puede ser incrementada con la inclusión de nuevas bases de datos de ejemplos en sus conjuntos de entrenamiento.

Referencias

[1] A. Andriyenko, K. Schindler, and S. Roth. Discrete-continuous optimization for multi-target tracking. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, jun 2012.

[2] Shuai Bai, Zhiqun He, Ting-Bing Xu, et al. Multi-hierarchical independent correlation filters for visual tracking. *CoRR*, abs/1811.10302, 2018.

[3] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1806–1819, sep 2011.

[4] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The CLEAR MOT metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

[5] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking. In *Computer Vision – ECCV 2016 Workshops*, pages 850–865, 2016.

[6] Dav Bolme, J. Ross Beveridge, Bruce A. Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, jun 2010.

[7] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *CoRR*, abs/1605.07678, 2016.

[8] Young chul Yoon, Abhijeet Boragule, Young min Song, Kwangjin Yoon, and Moongu Jeon. Online multi-object tracking with historical appearance matching and scene adaptive detection filtering. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, nov 2018.

[9] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conferen-*

ce on Computer Vision and Pattern Recognition (CVPR'05), 2005.

[10] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Discriminative scale space tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8):1561–1575, aug 2017.

[11] Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg, and Joost van de Weijer. Adaptive color attributes for real-time visual tracking. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, jun 2014.

[12] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *CoRR*, abs/1603.07285, 2016.

[13] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, sep 2010.

[14] T. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering*, 8(3):173–184, jul 1983.

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

[16] Qing Guo, Wei Feng, Ce Zhou, et al. Learning dynamic siamese network for visual object tracking. In *2017 IEEE International Conference on Computer Vision (ICCV)*, oct 2017.

[17] Yunhui Guo, Yandong Li, Rogério Schmidt Feris, Liqiang Wang, and Tajana Rosing. Depthwise convolution is all you need for learning multiple visual domains. *CoRR*, abs/1902.00927, 2019.

[18] Richard H. R. Hahnloser, Rahul Sarpeshkar, Misha A. Mahowald, Rodney J. Douglas, and H. Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951, jun 2000.

[19] Sam Hare, Amir Saffari, and Philip H. S. Torr. Struck: Structured output tracking with kernels. In *IEEE International Conference on Computer Vision (ICCV)*, pages 263–270, 2011.

[20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

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

[22] Joao F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, mar 2015.

[23] Andrew G. Howard, Menglong Zhu, Bo Chen, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[24] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37, pages 448–456, 2015.

[25] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Neural Information Processing Systems (NIPS)*, pages 2017–2025, 2015.

[26] Du Yong Kim and Moongu Jeon. Spatio-temporal auxiliary particle filtering with ℓ_1-norm-based appearance model learning for robust visual tracking. *IEEE Transactions on Image Processing*, 22(2):511–522, feb 2013.

[27] Matej Kristan, Aleš Leonardis, Jiří Matas, et al. The sixth visual object tracking VOT2018 challenge results. In *Lecture Notes in Computer Science*, pages 3–53. Springer International Publishing, 2019.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105, 2012.

[29] Ratnesh Kumar, Guillaume Charpiat, and Monique Thonnat. Multiple object tracking by efficient graph partitioning. In *Asian Conference on Computer Vision (ACCV)*, pages 445–460, 2015.

[30] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. MOTChallenge 2015: Towards a benchmark for multi-target tracking. *arXiv:1504.01942 [cs]*, Apr. 2015. arXiv: 1504.01942.

[31] Laura Leal-Taixé, Anton Milan, Konrad Schindler, et al. Tracking the trackers: An analysis of the state of the art in multiple object tracking. *CoRR*, abs/1704.02781, 2017.

[32] Kuan-Hui Lee and Jenq-Neng Hwang. On-road pedestrian tracking across multiple driving recorders. *IEEE Transactions on Multimedia*, 17(9):1429–1438, sep 2015.

[33] Bo Li, Wei Wu, Qiang Wang, et al. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[34] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, jun 2018.

[35] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, et al. Microsoft COCO: common objects in context. In *13th European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

[36] L. Liu, J. Xing, H. Ai, and X. Ruan. Hand posture recognition using finger geometric feature. In *21st International Conference on Pattern Recognition (ICPR2012)*, pages 565–568, Nov 2012.

[37] Wei Liu, Dragomir Anguelov, Dumitru Erhan, et al. SSD: Single shot MultiBox detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37, 2016.

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

[39] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*, Mar. 2016. arXiv: 1603.00831.

[40] Anton Milan, Seyed Hamid Rezatofighi, Anthony R. Dick, Konrad Schindler, and Ian D. Reid. Online multi-target tracking using recurrent neural networks. *CoRR*, abs/1604.03635, 2016.

[41] Anton Milan, Konrad Schindler, and Stefan Roth. Detection- and trajectory-level exclusion in multiple object tracking. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, jun 2013.

[42] Young min Song and Moongu Jeon. Online multiple object tracking with the hierarchically adopted GM-PHD filter using motion and appearance. In *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, oct 2016.

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

[44] Hamed Pirsiavash, Deva Ramanan, and Charless C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2011.

[45] Esteban Real, Jonathon Shlens, Stefano Mazzocchi, Xin Pan, and Vincent Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. *CoRR*, abs/1702.00824, 2017.

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

[47] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, pages 91–99, 2015.

[48] Seyed Hamid Rezatofighi, Anton Milani, Zhen Zhang, et al. Joint probabilistic matching using m-best solutions. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2016.

[49] Ergys Ristani and Carlo Tomasi. Tracking multiple people online and in real time. In *Asian Conference on Computer Vision (ACCV)*, pages 444–459, 2014.

[50] Olga Russakovsky, Jia Deng, Hao Su, et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, apr 2015.

[51] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations (ICLR)*, 2015.

[52] Arnold W. M. Smeulders, Dung Manh Chu, Rita Cucchiara, et al. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1442–1468, jul 2014.

[53] Siyu Tang, Mykhaylo Andriluka, Bjoern Andres, and Bernt Schiele. Multiple people tracking by lifted multicut and person re-identification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jul 2017.

[54] Jack Valmadre, Luca Bertinetto, Joao Henriques, Andrea Vedaldi, and Philip H. S. Torr. End-to-end representation learning for correlation filter based tracking. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jul 2017.

[55] Lorenzo Vaquero. Sistema de tracking visual de objetos mediante técnicas de aprendizaje profundo. Bachelor's thesis, Universidad de Santiago de Compostela, 2018.

[56] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H. S. Torr. Fast online object tracking and segmentation: A unifying approach. In *2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[57] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, sep 2017.

[58] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, jun 2013.

[59] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, sep 2015.

[60] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jul 2017.

[61] Bo Yang and R. Nevatia. An online learned CRF model for multi-target tracking. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, jun 2012.

[62] Ehwa Yang, Jeonghwan Gwak, and Moongu Jeon. Multi-human tracking using part-based appearance modelling and grouping-based tracklet association for visual surveillance applications. *Multimedia Tools and Applications*, 76(5):6731–6754, feb 2016.

[63] Kwangjin Yoon, Du Kim, Young-Chul Yoon, and Moongu Jeon. Data association for multi-object tracking via deep neural networks. *Sensors*, 19(3):559, jan 2019.

[64] Kwangjin Yoon, Young min Song, and Moongu Jeon. Multiple hypothesis tracking algorithm for multi-target multi-camera tracking with disjoint views. *IET Image Processing*, 12(7):1175–1184, jul 2018.

[65] Amir Roshan Zamir, Afshin Dehghan, and Mubarak Shah. GMCP-tracker: Global multi-object tracking using generalized minimum clique graphs. In *European Conference on Computer Vision (ECCV)*, pages 343–356, 2012.

[66] Guangcong Zhang and Patricio A. Vela. Good features to track for visual SLAM. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2015.

[67] Zheng Zhu, Qiang Wang, Bo Li, et al. Distractor-aware siamese networks for visual object tracking. In *European Conference on Computer Vision (ECCV)*, pages 103–119, 2018.