UNIVERSIDAD DE SANTIAGO DE COMPOSTELA

ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

Sistema de tracking visual de objetos mediante técnicas de aprendizaje profundo

Autor:
Lorenzo Vaquero Otal

Directores:
Manuel Mucientes Molina
Víctor Manuel Brea Sánchez

Grado en Ingeniería Informática

Julio de 2018

Trabajo de Fin de Grado presentado en la Escola Técnica Superior de Enxeñaría de la Universidad de Santiago de Compostela para la obtención del Grado en Ingeniería Informática

D. Manuel Mucientes Molina, Profesor del Departamento de Electrónica y Computación de la Universidad de Santiago de Compostela, y **D. Víctor Manuel Brea Sánchez**, Profesor del Departamento de Electrónica y Computación de la Universidad de Santiago de Compostela,

INFORMAN:

Que la presente memoria, titulada *Sistema de tracking visual de objetos mediante técnicas de aprendizaje profundo*, presentada por **D. Lorenzo Vaquero Otal** para superar los créditos correspondentes al Trabajo de Fin de Grado de la titulación de Grado en Ingeniería Informática, fue realizada bajo nuestra dirección en el Departamento de Electrónica y Computación de la Universidad de Santiago de Compostela.

Y para que así conste a los efectos oportunos, expiden el presente informe en Santiago de Compostela, a 25 de junio de 2018:

El tutor,	El cotutor,	El alumno,
Manuel Mucientes Molina	Víctor Manuel Brea Sánchez	Lorenzo Vaquero Otal

Agradecimientos

- Gracias a mi familia, por el constante apoyo que siempre me han ofrecido, dándome ánimos para permitirme lograr todo aquello que me he propuesto.
- Gracias a mis amigos, por poder contar con ellos tanto para el ocio como para los estudios, estando ahí siempre que los necesitara.
- Gracias a mis directores de proyecto, Manuel y Víctor, por confiar en mí y darme la oportunidad de trabajar con ellos.
- Gracias en especial a Miri, por todo lo que me supone.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

La identificación de los elementos que conforman una imagen resulta una tarea sencilla para los seres humanos. A diario, somos capaces de detectar los objetos que se encuentran a nuestro alrededor y de percibirlos de manera física, atribuyéndoles un significado a los fotones que llegan a nuestras retinas [1]. Esta tarea que a las personas nos resulta inmediata, posee en realidad una enorme complejidad, especialmente en la última etapa de todas, la de la interpretación. Es por este motivo que este proceso tan complicado de emular por una máquina ha sido estudiado profundamente durante las últimas décadas.

Con la finalidad de dotar a un sistema informático con la capacidad de adquirir, procesar, analizar y comprender las imágenes del mundo real para producir información numérica o simbólica que pueda ser tratada, surge la disciplina científica de la visión por computador [2]. Existen numerosas aplicaciones dentro del ámbito de la visión por computador, de entre las cuales es posible destacar la del seguimiento de objetos, núcleo del Trabajo Final de Grado (TFG) realizado. Este *tracking* visual de objetos es aplicado sobre un vídeo, y hace posible mantener la identidad de diferentes elementos detectados a lo largo de los fotogramas del mismo [3].

Al mantener la identidad de todos los elementos detectados, mediante el *tracking*

se hace posible no sólo conocer qué regiones de una imagen se corresponden con un objeto (reconocimiento), sino incorporar una dimensión adicional, la del tiempo, para así detectar trayectorias, cambios de forma y comportamientos a lo largo de un vídeo.

Esta mayor comprensión de lo captado en un vídeo hace posible que los sistemas informáticos realicen un gran número de tareas que antaño se consideraban impensables para una máquina. Desde labores rutinarias como el conteo de personas hasta tareas que requieran de una rigurosa precisión, como el seguimiento de vehículos desde UAV. Todas ellas son susceptibles de ser realizadas por un sistema informático con una gran velocidad, eficiencia y exactitud.

De esta forma, este será el punto en torno al cual gire el presente Trabajo Final de Grado, buscando desarrollar un sistema de *tracking* visual que facilite la automatización de procesos, y el desarrollo de aplicaciones integradas en tiempo real.

Para lograr este objetivo, una de las aproximaciones más extendidas en la actualidad consiste en el aprendizaje automático de métricas de similaridad, mediante la utilización de redes neuronales convolucionales (CNNs) [4]. Esta elección surge de los resultados de iniciativas como el reto VOT (Visual Object Tracking) en el que, tras recopilar los resultados de muy diversas soluciones de tracking, se ha podido demostrar que este tipo de alternativas ofrecen muy buenas métricas con un coste computacional relativamente bajo [5].

Así, el seguimiento de los objetos a lo largo de vídeos se realizará mediante una red neuronal convolucional. El hecho de que sea una red neuronal significa que se trata de un modelo computacional basado en un gran conjunto de unidades neuronales simples (neuronas artificiales). Cada unidad neuronal estará conectada con muchas otras y los enlaces entre ellas podrán incrementar o inhibir el estado de activación de las neuronas adyacentes. La ventaja de estos sistemas radica en que aprenden a partir de datos, en lugar de ser programados de forma explícita [6].

Por otra parte, el hecho de que la red sea convolucional significa que esta consistirá en múltiples capas que realizarán operaciones sobre matrices bidimensionales, mediante filtros convolucionales.

Como punto de partida del TFG propuesto, se emplearán los fundamentos seguidos por la red SiamFC, creada por Luca Bertinetto y Jack Valmadre, la cual permite realizar el tracking de objetos de cualquier tipo a un elevado número de fotogramas por segundo [7]. La razón por la cual no resulta factible utilizar directamente la implementación original es debido a que esta se encuentra desarrollada en MatConvNet, e incorporar MATLAB a aplicaciones integradas no es posible. Existe una variante de SiamFC desarrollada en TensorFlow (CFNet [8]), pero dado que se encuentra incompleta y sólo es apta para pruebas, se hace inviable su utilización.

De esta forma, utilizando los principios de SiamFC, será posible obtener un sistema de *tracking* versátil y extensible, cuyos módulos puedan ser fácilmente manipulados para la aplicación de futuras mejoras o la integración total o parcial en otros sistemas de visión por computador.

1.2. Red SiamFC

Dado que el modelo desarrollado a lo largo del presente TFG se encuentra basado en la red **SiamFC** [7], resulta conveniente introducir sus principales características.

La red SiamFC obtiene la posición actual de un objeto mediante la comparación de una imagen de ejemplo del mismo con el fotograma presente. Es por esto que la primera acción necesaria para el *tracking* es crear dicha imagen ejemplar (*exemplar image*). Esta imagen es extraida 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. No obstante, para agilizar la inferencia, durante el seguimiento no se considera la totalidad del fotograma, sino que únicamente se busca al objeto dentro de una región reducida alrededor de la última posición conocida, denominada área de búsqueda (*search area*). La extracción de este área de búsqueda se realiza de forma muy similar a la extracción de la imagen ejemplar

en el primer fotograma, salvo que abarca una zona mayor y es escalada a 255×255 píxeles.

Así, partiendo de la imagen ejemplar y del área de búsqueda, se realiza una extracción de las características de las mismas mediante una red AlexNet [9] siamesa, para después comparar las salidas obtenidas mediante una operación de correlación cruzada. El resultado de este proceso es un mapa de valores (*score map*) de tamaño 17×17, que indica la probabilidad de que el objeto se encuentre en cada sección del área de búsqueda.

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

Una descripción gráfica del proceso llevado a cabo en cada fotograma se muestra en la Figura 1.1.

1.3. Objetivos generales

El objetivo último del proyecto es el desarrollo de una versión de la red siamesa convolucional SiamFC, propuesta en el artículo de Luca Bertinetto y Jack Valmadre [7], optimizada para el *tracking* de objetos pequeños y adaptada para el seguimiento de múltiples objetos simultáneos. Esta será bautizada como **SiamTF**, haciendo referencia a su precursora y al hecho de que se hallará implementada en TensorFlow.

Los objetivos específicos del TFG son los descritos a continuación en la tabla 1.1:

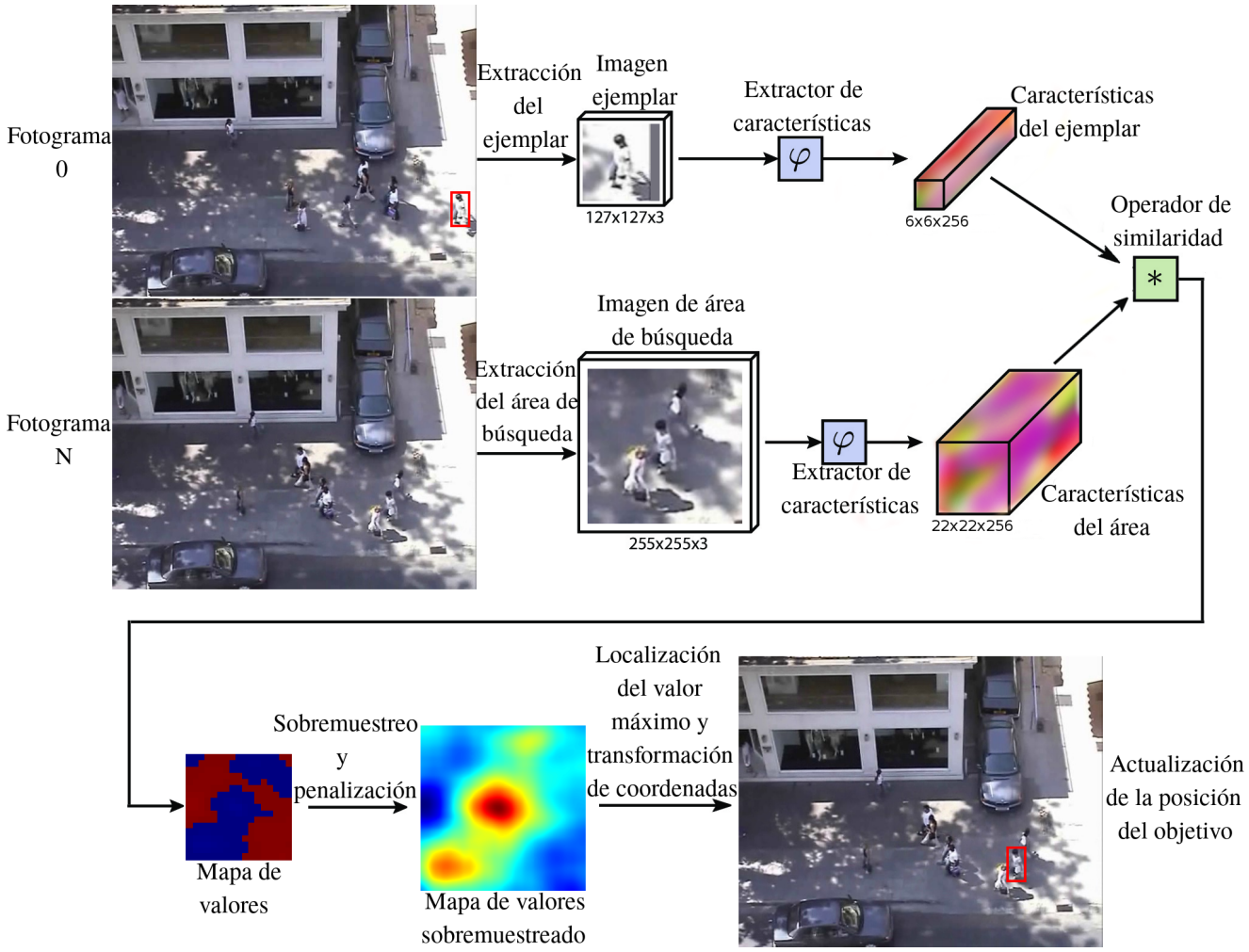


Figura 1.1: Funcionamiento de la red SiamFC

Objetivos del sistema	
ID	Descripción del objetivo
OS.01	El sistema desarrollado debe ser una solución extensible y eficiente de *tracking*, que mantenga la identidad de objetos arbitrarios detectados a lo largo de un vídeo y pueda ser integrado en una solución de *tracking* de alto nivel.
OS.02	Se busca que el sistema permita el *tracking* de múltiples elementos simultáneos.
OS.03	El sistema debe permitir el *tracking* de objetos pequeños, sin menoscabar el de elementos de otros tamaños.
OS.04	Resulta fundamental que el modelo pueda ser fácilmente entrenado a partir de bases de datos de vídeos etiquetados, proporcionadas por el usuario.
OS.05	El progreso de la inferencia y el entrenamiento del sistema deberá poder ser visualizado en tiempo real y de forma detallada.
OS.06	El sistema deberá poder ser evaluado siguiendo *benchmarks* estándar de *tracking*.

Tabla 1.1: Objetivos del sistema

1.4. Estructura de la memoria

La presente memoria se encuentra dividida en una serie de capítulos, cada uno de los cuales trata el proyecto desde un punto de vista diferente.

Dichos capítulos son los descritos a continuación:

Capítulo 1 - Introducción En este primer capítulo, se lleva a cabo una breve introducción del TFG realizado, poniendo de manifiesto su motivación y objetivos, así como los principios en los que se basa.

Capítulo 2 - Gestión del proyecto En este capítulo, se detallan todas las tareas relacionadas con la gestión de proyectos llevadas a cabo para la realización del TFG actual.

Capítulo 3 - Análisis de tecnologías y herramientas En este capítulo se exponen las tecnologías y herramientas empleadas para el desarrollo y la documentación del proyecto, así como las alternativas consideradas.

Capítulo 4 - Diseño e implementación En este capítulo se detalla el diseño llevado a cabo para la creación del sistema y se describen los detalles de implementación más interesantes del mismo.

Capítulo 5 - Pruebas y validación En este capítulo se detallan las pruebas realizadas para verificar el correcto funcionamiento del sistema y el cumplimiento de todos los requisitos establecidos.

Capítulo 6 - Resultados en los *benchmarks* propuestos Para facilitar la comparación del sistema desarrollado con otros *trackers*, en este capítulo se muestran los resultados obtenidos en los *benchmarks* Visual Object Tracking Challenge y Online Object Tracking Benchmark.

Capítulo 7 - Conclusiones y trabajo futuro En este capítulo se recogen las conclusiones obtenidas tras la finalización del desarrollo del proyecto y se lleva a cabo una reflexión sobre sus posibles mejoras y trabajo futuro.

Capítulo 2

Gestión del proyecto

En el presente capítulo se detallan todas las tareas relacionadas con la gestión de proyectos llevadas a cabo para la realización del TFG actual. Se incluyen la gestión del alcance, el establecimiento del catálogo de requisitos del sistema, la selección de la metodología de desarrollo, la gestión del tiempo, de riesgos y de la configuración y el análisis de los costes.

2.1. Enunciado del alcance del proyecto

En la presente sección se describe el resultado que se desea alcanzar tras la realización del proyecto, proporcionando un conocimiento común del alcance para todos los interesados.

2.1.1. Descripción del alcance del producto

La solución propuesta para alcanzar los objetivos presentados es un software que permita el *tracking* de múltiples objetos arbitrarios en vídeos, a partir de la delimitación de los mismos en el primer fotograma en el que aparecen.

Dicho software hará uso de una red neuronal convolucional, previamente entrenada mediante la base de datos ImageNet Large Scale Visual Recognition Challenge (ILSVRC), para realizar las inferencias. Estas inferencias consistirán en comparar

el aspecto inicial del objeto con el fotograma actual, obteniendo así un mapa de calor que indique la probabilidad de existencia del objeto en cada región de la imagen.

Además de poder ser entrenado y de realizar inferencias, el producto también deberá poder ser evaluado según las métricas establecidas por los *benchmarks* de *tracking* Visual Object Tracking Challenge (VOT) y Online Object Tracking Benchmark (OTB), por lo que cumplirá sus protocolos.

Para facilitar la interacción del usuario con el producto, todas las funciones del mismo contarán con una documentación que indique todas sus opciones y posibilidades, y los resultados de las inferencias serán mostrados visualmente, detallando el proceso seguido para llegar al resultado final. Además, se permitirá que el usuario procese sus propios vídeos en múltiples formatos y seleccione los objetos que desea seguir.

2.1.2. Diagrama de contexto

El diagrama de contexto mostrado en la Figura 2.1 representa al sistema en relación con su entorno, define los límites del mismo y muestra todos los productores y consumidores de información.

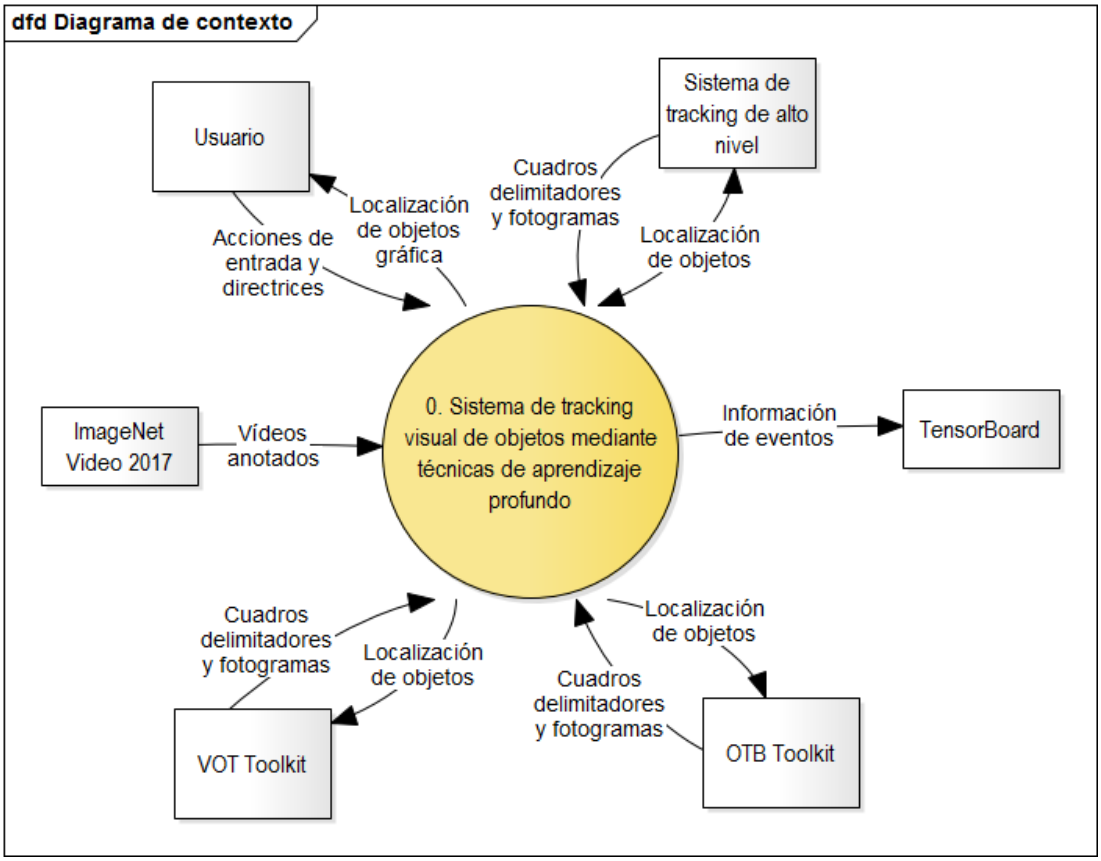


Figura 2.1: Diagrama de contexto del sistema

2.1.3. Criterios de aceptación del producto

Los criterios mínimos de aceptación del producto son los mostrados a continuación en la Tabla 2.1, los cuales deberán ser necesariamente aprobados por el cliente: los tutores del proyecto, D. Manuel Mucientes Molina y D. Víctor Manuel Brea Sánchez. Asimismo, se indica mediante un código identificador el objetivo de sistema resuelto por cada uno de los criterios propuestos:

Criterios de aceptación del producto	
ID	Descripción
CA.01	El sistema es capaz de inferir la posición de un objeto en más del 80 % de los fotogramas de un vídeo de ejemplo propuesto por los tutores del proyecto. (OS.01)
CA.02	El código fuente del sistema se encuentra organizado de forma modular y haciendo uso de interfaces. (OS.01)
CA.03	El sistema es capaz de procesar fotogramas en tiempo real (al menos 30 fotogramas por segundo). (OS.01)
CA.04	El sistema puede hacer *tracking* de más de un objeto de forma simultánea. (OS.02)
CA.05	El sistema es capaz de inferir la posición de un objeto pequeño en más del 75 % de los fotogramas de un vídeo de ejemplo propuesto por los tutores del proyecto. (OS.03)
CA.06	El sistema puede ser entrenado a partir de la base de datos ImageNet Large Scale Visual Recognition Challenge. (OS.04)
CA.07	Es posible visualizar de forma gráfica y en tiempo real el progreso de una inferencia. (OS.05)
CA.08	Es posible visualizar de forma gráfica y en tiempo real el progreso de un entrenamiento. (OS.05)
CA.09	El sistema puede ser evaluado mediante Visual Object Tracking Challenge. (OS.06)
CA.10	El sistema puede ser evaluado mediante Online Object Tracking Benchmark. (OS.06)

Tabla 2.1: Criterios de aceptación del producto

2.1.4. Entregables del proyecto

Al final de cada uno de los incrementos planteados, los tutores del proyecto dispondrán de un producto funcional que incluirá los requisitos satisfechos hasta la última iteración. De este modo, aunque el sistema esté inacabado, los tutores podrán, si así lo desean, empezar a trabajar con algunos de sus módulos.

Junto a cada incremento, se hará entrega también de los manuales técnicos y de usuario relacionados.

Así, el alumno se compromete a realizar la entrega de los siguientes items a los tutores del proyecto:

Entregables del proyecto de cara a los tutores	
ID	Entregable
EP.01	Incremento 1: Módulo de inferencia simple y utilidades de visualización
EP.02	Incremento 2: Módulo de entrenamiento
EP.03	Incremento 3: Módulo de evaluación
EP.04	Incremento 4: Utilidades de conversión, almacenamiento de eventos y dockerización
EP.05	Incremento 5: Optimización de hiperparámetros para objetos pequeños y adaptación a inferencia de múltiples objetos
EP.06	Manuales técnico y de usuario finales
EP.07	Memoria del proyecto

Tabla 2.2: Entregables del proyecto de cara a los tutores

De cara al depósito del Trabajo de Fin de Grado, dado que debe hacerse en una única vez, se planteará como una entrega final que contará con los siguientes elementos:

Entregables finales del proyecto	
ID	Entregable
EF.01	Software funcional ejecutable que cumple los criterios de aceptación
EF.02	Código fuente íntegro del producto y de otros elementos secundarios necesarios para su correcto funcionamiento
EF.03	La presente memoria junto con los manuales técnicos y de usuario

Tabla 2.3: Entregables finales del proyecto

2.1.5. Supuestos del proyecto

Se detallan a continuación las asunciones realizadas para este proyecto. Es decir, los hechos que se considerarán verdaderos durante el desarrollo del mismo, y que de no ser ciertos podrían poner en grave peligro su compleción:

Supuestos del proyecto	
ID	Descripción
SP.01	El servidor de computación GPGPU se encontrará disponible para el uso exclusivo del alumno durante un mínimo de 36 horas semanales.
SP.02	Los vídeos de ejemplo suministrados se encontrarán correctamente etiquetados y estarán fuertemente ligados al ámbito de la aplicación final.
SP.03	Los cuadros delimitadores de objetos proporcionados al sistema para la extracción de la imagen ejemplar serán correctos.

Tabla 2.4: Supuestos del proyecto

2.1.6. Restricciones del proyecto

Seguidamente, se detallan las limitaciones a las que se ve sometido el proyecto:

Restricciones del proyecto	
ID	Descripción
LP.01	La fecha de finalización límite del proyecto será el 2 de julio de 2018, para hacer posible su defensa en julio del mismo año.
LP.02	La cantidad de trabajo realizado por el alumno no podrá ser inferior a las 400 horas ni superior a las 425 horas.
LP.03	A excepción del servidor de computación GPGPU y el NVIDIA Jetson Developer Kit, el resto de elementos hardware y software utilizados durante el desarrollo del presente proyecto deberán ser adquiridos por el alumno.

Tabla 2.5: Restricciones del proyecto

2.1.7. Exclusiones del proyecto

En este apartado se recogen aquellos elementos que se consideran fuera del alcance del proyecto:

Exclusiones del proyecto	
ID	Descripción
XP.01	En ningún caso será el producto desarrollado el encargado de realizar las detecciones de objetos en vídeos.
XP.02	Se excluyen del proyecto todos aquellos requisitos que pongan en riesgo la finalización a tiempo del mismo, ya sea por su complejidad o impracticidad.
XP.03	Los requisitos funcionales del presente proyecto no serán ampliados una vez dé comienzo la fase ejecución del mismo.

Tabla 2.6: Exclusiones del proyecto

2.2. Catálogo de requisitos del sistema

En la presente sección se detallan los requisitos establecidos por el cliente, los cuales permiten describir con precisión el sistema de información y sirven de base para comprobar que es completa la especificación de los modelos obtenidos.

2.2.1. Requisitos de información

RI.01	**Objeto**
Descripción	Información relativa a un elemento, mostrado total o parcialmente en al menos un fotograma, del cual se desea mantener la identidad a lo largo de un vídeo.
Datos específicos	▪ ID ▪ Cuadros delimitadores ▪ Fotogramas en los que aparece ▪ Imagen ejemplar
Tiempo de vida	▪ Medio: 30 segundos ▪ Máximo: 5 minutos
Ocurrencias simultáneas	▪ Medio: 3 unidades ▪ Máximo: 10 unidades
Importancia	Vital
Estabilidad	Alta
Comentarios	La relación fotogramas–objetos se representa del presente modo (los objetos aparecen en fotogramas, en vez de que los fotogramas contengan objetos) para reforzar la conservación de la identidad que persigue el *tracking*.

RI.02	**Cuadro delimitador alineado con los ejes**
Descripción	Información relativa al conjunto de píxeles de una imagen comprendidos dentro de un cuadrilátero rectángulo, cuyos lados son paralelos y perpendiculares a los ejes de coordenadas.
Datos específicos	▪ Coordenada central ▪ Altura ▪ Anchura
Tiempo de vida	▪ Medio: 30 segundos ▪ Máximo: 5 minutos
Ocurrencias simultáneas	▪ Medio: 3 unidades ▪ Máximo: 10 unidades
Importancia	Vital
Estabilidad	Alta

RI.03	**Imagen ejemplar (*Exemplar image*)**
Descripción	Información relativa al conjunto de píxeles de un fotograma centrados en un objeto, que engloban su área más un margen definido. Mediante la comparación de este ejemplar y un área de búsqueda, se obtiene un mapa de valores.
Datos específicos	▪ Dimensiones (se trata de un cuadrado alineado con los ejes de coordenadas) ▪ Píxeles
Tiempo de vida	▪ Medio: 30 segundos ▪ Máximo: 5 minutos
Ocurrencias simultáneas	▪ Medio: 3 unidades ▪ Máximo: 10 unidades
Importancia	Vital
Estabilidad	Alta

RI.04	**Imagen de área de búsqueda (*Search area image*)**
Descripción	Información relativa al conjunto de píxeles de un fotograma centrados en la última posición conocida del objeto, que engloban el área en la cual se buscará la existencia de un objeto concreto. Mediante la comparación de esta área de búsqueda y de un ejemplar del objeto, se obtiene un mapa de valores.
Datos específicos	▪ Dimensiones (se trata de un cuadrado alineado con los ejes de coordenadas) ▪ Píxeles
Tiempo de vida	▪ Medio: 0.03 segundos ▪ Máximo: 1 segundo
Ocurrencias simultáneas	▪ Medio: 3 unidades ▪ Máximo: 10 unidades
Importancia	Vital
Estabilidad	Alta

RI.05	**Mapa de valores (*Score map*)**
Descripción	Información relativa a la similaridad entre una imagen ejemplar y las diversas subsecciones que componen un área de búsqueda. Se trata de un mapa de calor cuyos valores más altos indican la existencia de un ejemplar en un punto de un área de búsqueda.
Datos específicos	▪ Dimensiones (se trata de un cuadrado) ▪ Valores de los puntos
Tiempo de vida	▪ Medio: 0.03 segundos ▪ Máximo: 1 segundo
Ocurrencias simultáneas	▪ Medio: 1 unidad ▪ Máximo: 10 unidades
Importancia	Vital
Estabilidad	Alta

RI.06	**Vídeo**
Descripción	Información relativa a un conjunto de fotogramas ordenados, en los que pueden encontrarse objetos de los que se desea realizar *tracking*.
Datos específicos	▪ ID ▪ Fotogramas ▪ Objetos
Tiempo de vida	▪ Medio: 1 minuto ▪ Máximo: 20 minutos
Ocurrencias simultáneas	▪ Medio: 1 unidad ▪ Máximo: 1 unidad
Importancia	Vital
Estabilidad	Alta
Comentarios	La información de los objetos mostrados en un vídeo puede ser, a priori, conocida (para entrenamiento y evaluación) o desconocida (para llevar a cabo inferencia/test)

RI.07	**Base de datos de vídeos**
Descripción	Información relativa a un conjunto de vídeos organizados para facilitar los procesos de entrenamiento, evaluación y test del modelo desarrollado.
Datos específicos	▪ ID ▪ Vídeos para entrenamiento ▪ Vídeos para evaluación ▪ Vídeos para test ▪ Estadísticas de color
Tiempo de vida	▪ Medio: 1 año ▪ Máximo: 3 años
Ocurrencias simultáneas	▪ Medio: 1 unidad ▪ Máximo: 1 unidad
Importancia	Vital
Estabilidad	Alta

RI.08	**Estadísticas de color**
Descripción	Información estadística relativa al color de un conjunto de vídeos, empleada durante el entrenamiento para el proceso de aumento de datos (*data augmentation*).
Datos específicos	▪ Número de muestras ▪ Media de color ▪ Varianza de color ▪ Desviación típica de color
Tiempo de vida	▪ Medio: 1 año ▪ Máximo: 3 años
Ocurrencias simultáneas	▪ Medio: 1 unidad ▪ Máximo: 1 unidad
Importancia	Importante
Estabilidad	Media

RI.09	**Parámetros del modelo**
Descripción	Información relativa a los hiperparámetros de diseño considerados durante la construcción y ejecución del modelo.
Datos específicos	▪ Parámetros generales • ID de GPU • Carpeta de eventos (logs) • Tipo de rama • Tamaño del ejemplar • Tamaño del área de búsqueda • Cantidad de contexto • Factor de ajuste • Valor de epsilon • Pasos por evento generado ▪ Parámetros de *tracking* • Escalas consideradas • Factor de escala • Penalización por escalado • Suavizado de escalado • Tamaño mínimo de objeto • Tamaño máximo de objeto • Factor de sobremuestreo • Tipo de ventana de penalización • Influencia de la ventana de penalización • Ruta del punto de control del modelo

	▪ Parámetros de entrenamiento • Semilla (seed) utilizada • Tamaño del lote (batch) • Fracción de vídeos para entrenamiento • Radio positivo en las etiquetas • Radio neutro en las etiquetas • Número de pares por ciclo • Número de ciclos de aprendizaje • Máxima separación de fotogramas entre ejemplar y área • Método de inicializacion de pesos • Desviación estándar de la inicialización de pesos • Cantidad de momento • Tasa de aprendizaje inicial • Tasa de aprendizaje final • Política de templado de la tasa de aprendizaje • Factor de decaída de los pesos en las convoluciones • Factor de decaída de los pesos en la normalización por lotes • Estirado máximo en el aumento de datos • Traslación máxima en el aumento de datos • Factor de varianza del color en el aumento de datos • Probabilidad de reflejado en el aumento de datos
Tiempo de vida	▪ Medio: 6 meses ▪ Máximo: 2 años
Ocurrencias simultáneas	▪ Medio: 1 unidad ▪ Máximo: 1 unidad
Importancia	Vital
Estabilidad	Baja

RI.10	**Punto de control del modelo**
Descripción	Información relativa a los parámetros aprendidos por la red neuronal durante las fases de entrenamiento. Permiten tanto realizar evaluaciones e inferencias como retomar el entrenamiento.
Datos específicos	▪ Valores de los pesos de la red ▪ Grafo de ejecución
Tiempo de vida	▪ Medio: 6 meses ▪ Máximo: 2 años
Ocurrencias simultáneas	▪ Medio: 1 unidad ▪ Máximo: 10 unidades
Importancia	Vital
Estabilidad	Alta
Comentarios	El número de puntos de control empleados durante la inferencia será de 1. No obstante, a la hora de entrenar el sistema, se generarán múltiples que más tarde serán evaluados para escoger aquel que ofrezca mejores resultados.

RI.11	**Evento del modelo (logs)**
Descripción	Información relativa a los logs generados durante la ejecución del modelo, utilizados por TensorBoard para obtener un resumen del progreso y resultado del proceso.
Datos específicos	▪ Grafo de ejecución ▪ Fotogramas procesados ▪ Vídeos procesados ▪ Objetos detectados ▪ Error de entrenamiento ▪ Error de evaluación ▪ Ejemplares considerados ▪ Áreas de búsqueda consideradas
Tiempo de vida	▪ Medio: 2 días ▪ Máximo: 2 semanas
Ocurrencias simultáneas	▪ Medio: 10 unidades ▪ Máximo: 50 unidades
Importancia	Importante
Estabilidad	Alta
Comentarios	El número de datos específicos incluidos dentro de un evento del modelo podrá variar en función de la cantidad de información requerida por el usuario y del rendimiento deseado del sistema

RI.12	**Red siamesa**
Descripción	Información relativa a las capas, operaciones y tensores que conforman la red neuronal.
Datos específicos	▪ Parámetros ▪ Punto de control del modelo ▪ Tensor de ejemplar ▪ Tensor de área de búsqueda ▪ Tensor de mapa de valores
Tiempo de vida	▪ Medio: 1 día ▪ Máximo: 1 mes
Ocurrencias simultáneas	▪ Medio: 1 unidad ▪ Máximo: 1 unidad
Importancia	Vital
Estabilidad	Alta

RI.13	**Motor de inferencia (Tracker)**
Descripción	Información relativa al estado del sistema durante la realización de una inferencia o evaluación.
Datos específicos	▪ Parámetros ▪ Red siamesa ▪ Vídeo ▪ Objeto(s) ▪ Área(s) de búsqueda actual(es) ▪ Mapa(s) de valor(es) actual(es) ▪ Confidencia(s) actual(es)
Tiempo de vida	▪ Medio: 1 día ▪ Máximo: 1 mes
Ocurrencias simultáneas	▪ Medio: 1 unidad ▪ Máximo: 1 unidad
Importancia	Vital
Estabilidad	Media

RI.14	**Motor de entrenamiento (Trainer)**
Descripción	Información relativa al estado del sistema durante la realización de un entrenamiento.
Datos específicos	▪ Parámetros ▪ Red siamesa ▪ Base de datos de vídeos ▪ Coste (*loss*) actual
Tiempo de vida	▪ Medio: 16 horas ▪ Máximo: 7 días
Ocurrencias simultáneas	▪ Medio: 1 unidad ▪ Máximo: 1 unidad
Importancia	Vital
Estabilidad	Media

2.2.2. Requisitos funcionales

Actores

En este apartado se describen todas aquellas entidades externas al sistema que guardan una relación con este y que le demandan alguna funcionalidad.

Sistema de *tracking* de alto nivel: Sistema de análisis y asociación, capaz de utilizar información temporal, de movimiento y de detección para localizar nuevos objetivos y ejecutar el *tracker*, así como corregirlo en caso de error.

Usuario: Persona con acceso al sistema, la cual podrá realizar labores de entrenamiento, validación y uso como demostrador, así como las propias de análisis y asociación de un *tracker* de alto nivel.

VOT Toolkit: Conjunto de herramientas que permiten la validación de un *tracker* según las métricas propuestas por el Visual Object Tracking Challenge (VOT).

OTB Toolkit: Conjunto de herramientas que permiten la validación de un *tracker* según las métricas propuestas por el Online Object Tracking Benchmark (OTB).

Casos de uso del sistema

En el presente apartado se describirán los pasos y las actividades que podrán realizar los actores del sistema para poder llevar a cabo los procesos y tareas que este ofrece.

Se ha decidido llevar a cabo la captura de los requisitos funcionales mediante casos de uso, dada la flexibilidad y cantidad de información que estos ofrecen, además de que encajan realmente bien con paradigmas de programación orientados a objetos.

Cada caso de uso posee una serie de criterios de validación, los cuales pueden ser consultados en el Capítulo 5 de la presente memoria.

Diagrama de casos de uso

El diagrama de casos de uso del sistema es el que se muestra a continuación, en la Figura 2.2:

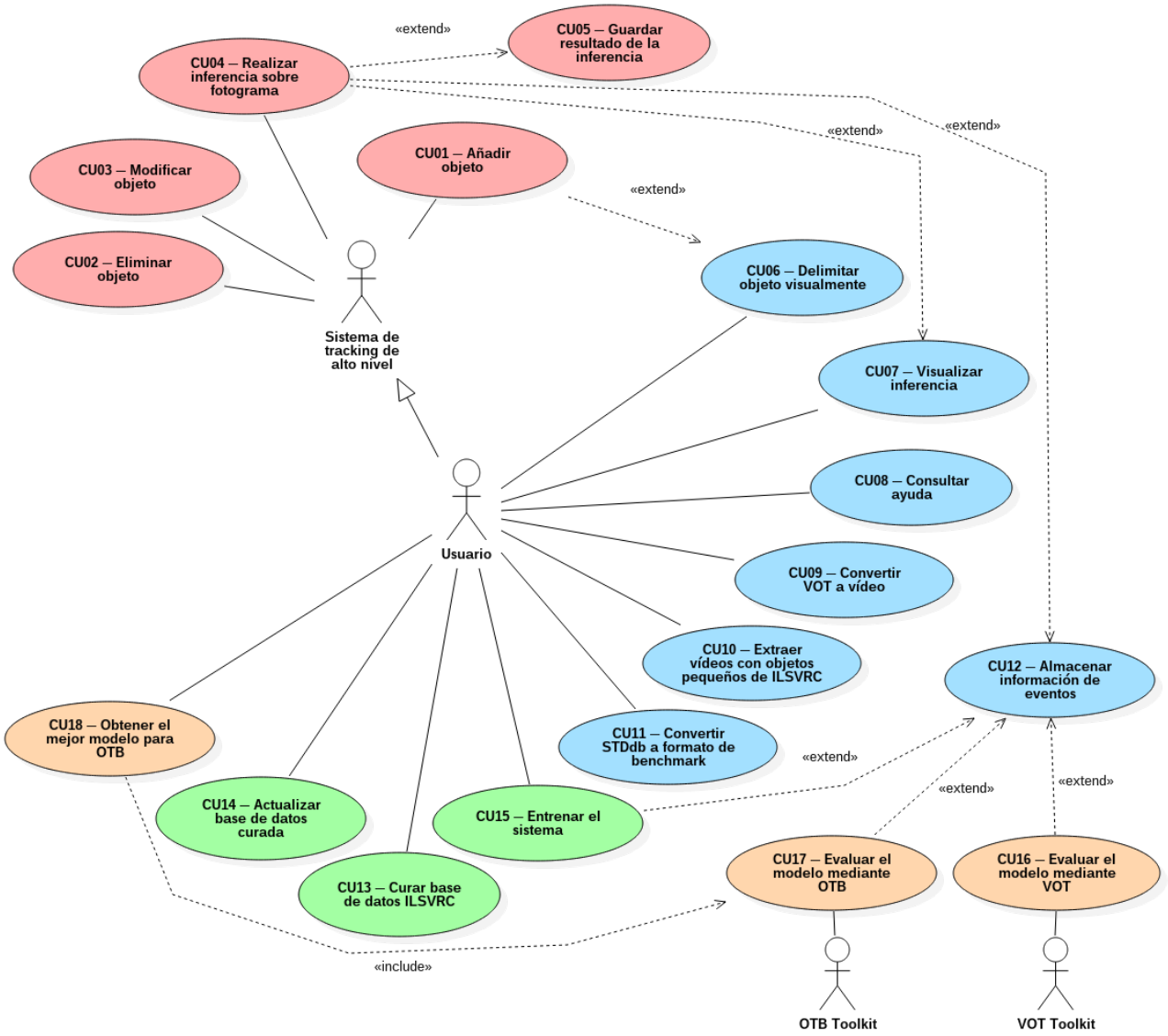


Figura 2.2: Diagrama de casos de uso del sistema

Subsistema de inferencia

La responsabilidad de este subsistema es la de ofrecer una serie de funciones que permitan realizar inferencias de uno o varios objetos a lo largo vídeos, así como almacenar toda la información relacionada del proceso.

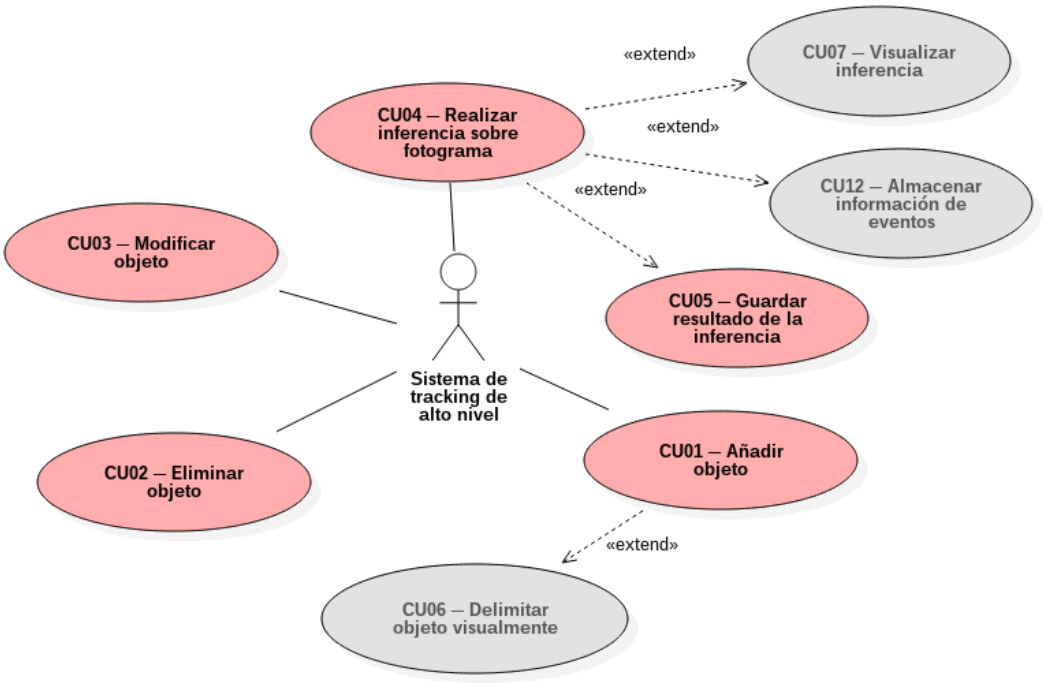


Figura 2.3: Casos de uso del subsistema de inferencia

CU.01	**Añadir objeto**
Actores	▪ Sistema de *tracking* de alto nivel
Descripción	Un Sistema de *tracking* de alto nivel, tras detectar un nuevo objeto del cual desea mantener su identidad, lo añade al sistema de *tracking*, indicando su posición y dimensiones. Esto permitirá su seguimiento en los sucesivos fotogramas.
Relaciones	▪ Extensión hacia **CU-06 - Delimitar objeto visualmente**.
Precondiciones	▪ No puede encontrarse actualmente en seguimiento un objeto con el mismo identificador que el nuevo.
Postcondiciones	▪ El sistema pasa a almacenar la información necesaria para realizar el *tracking* del nuevo objeto.
Escenario principal	1. El Sistema de *tracking* de alto nivel le comunica al sistema la posición y dimensiones (dentro de un fotograma) de un nuevo objeto del que hacer *tracking*, así como un identificador asociado. 2. El sistema almacena la posición y dimensiones del nuevo objeto, así como su imagen ejemplar. 3. Se notifica al Sistema de *tracking* de alto nivel el éxito de la operación.
Escenarios alternativos	1. El Sistema de *tracking* de alto nivel es del tipo Usuario, paso 0: *a*) Se ejecuta el **CU-06 - Delimitar objeto visualmente**. *b*) Se retorna al paso 1 del escenario principal con los datos obtenidos del paso anterior.

Tipos de datos	Identificador, cuadro delimitador alineado con los ejes y *tracker*
Frecuencia de ejecución	Variará en función del vídeo procesado. Se estima que entre 1 y 10 veces por minuto.
Importancia	Importante
Estabilidad	Alta
Criterios de validación	PV.01

CU.02	**Eliminar objeto**
Actores	▪ Sistema de *tracking* de alto nivel
Descripción	Un Sistema de *tracking* de alto nivel, tras detectar que un objeto ha abandonado la escena o que ha dejado de ser relevante, notifica al sistema de *tracking* que lo elimine de su lista de seguimiento. De esta forma, en los sucesivos fotogramas, la identidad de dicho objeto no será mantenida.
Relaciones	
Precondiciones	▪ El objeto que se desea eliminar debe encontrarse actualmente en seguimiento.
Postcondiciones	▪ El sistema deja de almacenar la información necesaria para realizar el *tracking* del objeto con el identificador coincidente.

Escenario principal	1. El Sistema de *tracking* de alto nivel le comunica al sistema el identificador del objeto que desea dejar de seguir. 2. El sistema elimina toda la información de *tracking* asociada al objeto. 3. Se notifica al Sistema de *tracking* de alto nivel el éxito de la operación.
Escenarios alternativos	
Tipos de datos	Objeto y *tracker*
Frecuencia de ejecución	Variará en función del vídeo procesado. Se estima que entre 1 y 10 veces por minuto.
Importancia	Importante
Estabilidad	Alta
Criterios de validación	PV.02

CU.03	**Modificar objeto**
Actores	▪ Sistema de *tracking* de alto nivel
Descripción	Un Sistema de *tracking* de alto nivel, tras detectar una falta de concordancia entre la información suministrada por el *tracker* y la posición real de un objeto, notifica al sistema de *tracking* que corrija esta información.
Relaciones	

Precondiciones	▪ El objeto que se desea modificar debe encontrarse actualmente en seguimiento.
Postcondiciones	▪ La información de *tracking* del objeto con el identificador coincidente pasa a encontrarse modificada.
Escenario principal	1. El Sistema de *tracking* de alto nivel le comunica al sistema el identificador del objeto que desea modificar, así como su nueva posición y dimensiones. 2. El sistema modifica la información de posición y dimensiones asociadas al objeto. 3. Se notifica al Sistema de *tracking* de alto nivel el éxito de la operación.
Escenarios alternativos	
Tipos de datos	Objeto y *tracker*
Frecuencia de ejecución	Variará en función del vídeo procesado. Se estima que entre 1 y 3 veces por minuto.
Importancia	Vital
Estabilidad	Alta
Criterios de validación	PV.03

CU.04	**Realizar inferencia sobre fotograma**
Actores	▪ Sistema de *tracking* de alto nivel
Descripción	Un Sistema de *tracking* de alto nivel le facilita al *tracker* un nuevo fotograma, a partir del cual se obtendrán unas nuevas posiciones y dimensiones para cada uno de los objetos seguidos.
Relaciones	▪ Extensión hacia **CU-05 - Guardar resultado de la inferencia**. ▪ Extensión hacia **CU-09 - Visualizar inferencia**. ▪ Extensión hacia **CU-14 - Almacenar información de eventos**.
Precondiciones	▪ Debe existir al menos un objeto a seguir en el sistema.
Postcondiciones	▪ La información de *tracking* de los objetos del sistema pasa a encontrarse actualizada.
Escenario principal	1. El Sistema de *tracking* de alto nivel le proporciona al sistema un nuevo fotograma. 2. El sistema modifica la información de posición y dimensiones asociadas a los objetos seguidos, de acuerdo a lo que aparece representado en la imagen. 3. La información actualizada de los objetos le es transmitida al Sistema de *tracking* de alto nivel.

Escenarios alternativos	1. Un parámetro del sistema indica que la inferencia debe ser guardada, paso 2: *a*) Se ejecuta el **CU-05 - Guardar resultado de la inferencia**, pasándole como entrada la información actualizada de los objetos. *b*) Se retorna al paso 2 del escenario principal. 2. El Sistema de *tracking* de alto nivel es del tipo Usuario, paso 3: *a*) Se ejecuta el **CU-09 - Visualizar inferencia**, pasándole como entrada la información actualizada de los objetos. *b*) Se retorna al paso 3 del escenario principal. 3. Un parámetro del sistema indica que la información de eventos debe ser guardada, paso 2: *a*) Se ejecuta el **CU-14 - Almacenar información de eventos**, pasándole como entrada el estado actual del sistema de *tracking*. *b*) Se retorna al paso 3 del escenario principal.
Tipos de datos	Fotograma, objetos y *tracker*
Frecuencia de ejecución	Entre 40 y 60 veces por segundo.
Importancia	Importante
Estabilidad	Alta
Criterios de validación	PV.04, PV.05

CU.05	**Guardar resultado de la inferencia**
Actores	▪ Sistema de *tracking* de alto nivel
Descripción	La información relativa a las inferencias del *tracker* es almacenada de forma estática, en el formato establecido por los parámetros del sistema.
Relaciones	▪ Extensión desde **CU-04 - Realizar inferencia sobre fotograma**.
Precondiciones	▪ Debe haberse realizado una inferencia sobre un fotograma.
Postcondiciones	▪ La información de *tracking* almacenada estáticamente pasa a encontrarse actualizada.
Escenario principal	1. De no existir, se crea un archivo para guardar el resultado de la inferencia, en el formato indicado por los parámetros del sistema. 2. Se añade al archivo estático la información de la inferencia actual, en el formato indicado por los parámetros del sistema. 3. Se notifica el éxito de la operación.
Escenarios alternativos	
Tipos de datos	Fotograma y objetos
Frecuencia de ejecución	Entre 40 y 60 veces por segundo.
Importancia	Estimulante

Estabilidad	Media
Criterios de validación	PV.06, PV.07
Comentarios	Este caso de uso ha sido modelado de forma independiente del **CU-04 - Realizar inferencia sobre fotograma**, para que así resulte más sencillo realizar una posible extensión de funcionalidad de los casos de uso **CU-15**, **CU-16** y **CU-17**, que incluya el almacenamiento estático de las inferencias realizadas.

Subsistema de utilidades

La responsabilidad de este subsistema es la de ofrecer una serie de utilidades que simplifiquen el uso de la aplicación por parte del usuario, así como proporcionar un valioso conjunto de herramientas de análisis y depuración de sistemas de *tracking*.

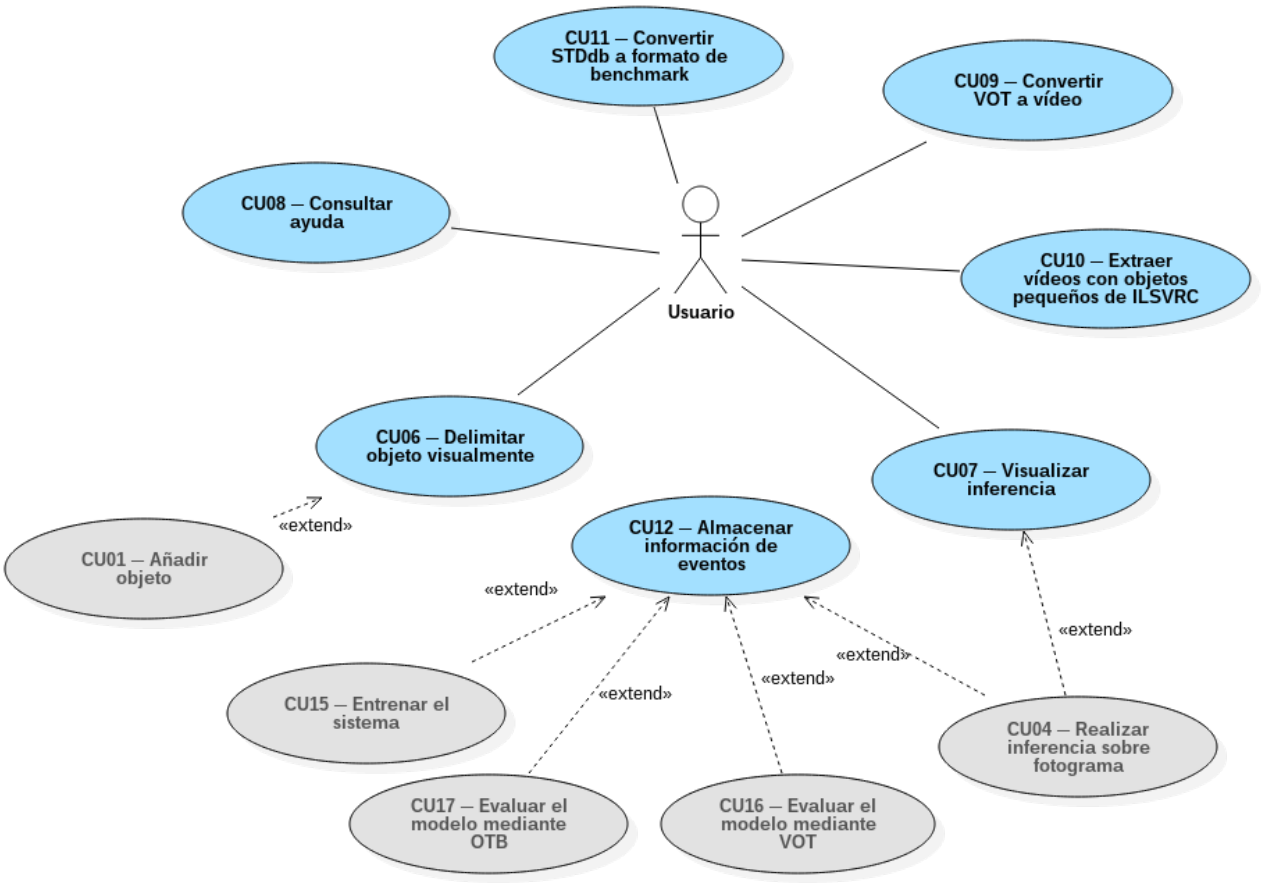


Figura 2.4: Casos de uso del subsistema de utilidades

CU.06	**Delimitar objeto visualmente**
Actores	▪ Usuario
Descripción	Un Usuario selecciona de forma visual el cuadro delimitador que contiene a un objeto que aparece en un vídeo, y esta información le es transmitida al sistema.
Relaciones	▪ Extensión desde **CU-01 - Añadir objeto**.
Precondiciones	
Postcondiciones	▪ El sistema recibe la información de posición y dimensiones de un nuevo objeto.
Escenario principal	1. El sistema le presenta al usuario una interfaz gráfica con la que interactuar, la cual contiene un fotograma del vídeo del cual se está realizando el seguimiento. 2. El usuario selecciona un cuadro delimitador en torno al objeto del cual desea realizar el *tracking*. 3. Se le transmite al sistema la información relativa al cuadro delimitador seleccionado.
Escenarios alternativos	1. El vídeo del cual se está realizando el tracking proviene de una fuente en *streaming*, paso 0: *a*) El sistema le presenta al usuario una interfaz gráfica con la que interactuar, la cual contiene un flujo contínuo de fotogramas obtenidos en directo de la fuente en *streaming*. *b*) Se retorna al paso 2 del escenario principal.
Tipos de datos	Cuadro delimitador alineado con los ejes y fotograma

Frecuencia de ejecución	Variará en función del vídeo procesado. Se estima que entre 1 y 10 veces por minuto.
Importancia	Importante
Estabilidad	Media
Criterios de validación	PV.08

CU.07	**Visualizar inferencia**
Actores	▪ Usuario
Descripción	El sistema le presenta a un Usuario información visual relativa al estado actual del *tracking*.
Relaciones	▪ Extensión desde **CU-04 - Realizar inferencia sobre fotograma**.
Precondiciones	▪ Debe haberse realizado previamente una inferencia sobre un fotograma.
Postcondiciones	▪ El Usuario es conocedor del estado actual del *tracking*.
Escenario principal	1. De no existir, se crea una interfaz gráfica que también permita obtener información relativa a la entrada del usuario. 2. La interfaz gráfica se completa con el último fotograma procesado y una serie de rectángulos indicando la posición de cada objeto.

Escenarios alternativos	1. Un parámetro del sistema indica que la información visualizada debe ser exhaustiva, paso 3: *a*) La interfaz gráfica se completa con información relativa a las imágenes de área de búsqueda y ejemplares de los objetos, así como una representación del mapa de valores.
Tipos de datos	Fotograma y objetos
Frecuencia de ejecución	Entre 40 y 60 veces por segundo.
Importancia	Importante
Estabilidad	Media
Criterios de validación	PV.09, PV.10, PV.11, PV.12
Comentarios	Este caso de uso ha sido modelado de forma independiente del **CU-04 - Realizar inferencia sobre fotograma**, para que así resulte más sencillo realizar una posible extensión de funcionalidad de los casos de uso **CU-16** y **CU-17**, que incluya la visualización en tiempo real de la evaluación.

CU.08	**Consultar ayuda**
Actores	▪ Usuario
Descripción	Un Usuario obtiene ayuda relativa a la utilización de cada una de las funcionalidades de las que dispone.
Relaciones	

Precondiciones	
Postcondiciones	▪ El Usuario es conocedor de la forma de interactuar con el sistema para obtener los resultados que desea.
Escenario principal	1. El Usuario recurre a la herramienta de la cual desea obtener asistencia. 2. El Usuario ejecuta la funcionalidad del sistema en modo ayuda. 3. El sistema le presenta al usuario todas las opciones de interacción con la herramienta.
Escenarios alternativos	1. El usuario ejecuta la funcionalidad del sistema de forma errónea, paso 2: *a*) Se le notifica al usuario que ha interactuado de forma incorrecta con el sistema. *b*) Se retorna al paso 3 del escenario principal.
Tipos de datos	
Frecuencia de ejecución	Variará en función de la experiencia del usuario. Se estima que entre 1 y 10 veces por cada 10 minutos, para usuarios noveles y entre 1 y 2 veces por cada 10 horas para usuarios experimentados.
Importancia	Vital
Estabilidad	Alta
Criterios de validación	PV.13

CU.09	**Convertir VOT a vídeo**
Actores	▪ Usuario
Descripción	Un vídeo en formato VOT facilitado por el Usuario es transformado a un formato de vídeo reconocido por la mayor parte de reproductores (`.mp4` o `.avi`).
Relaciones	
Precondiciones	
Postcondiciones	▪ Un archivo de vídeo con el contenido del VOT original pasa a encontrarse disponible en el almacenamiento estático de la máquina utilizada.
Escenario principal	1. El Usuario selecciona un vídeo en formato VOT y se lo facilita al sistema para su conversión, indicando el formato de salida. 2. El sistema carga los archivos de fotograma de VOT. 3. El sistema grupa los fotogramas en un vídeo con el formato deseado. 4. El sistema le notifica al Usuario el éxito de la operación.
Escenarios alternativos	1. Un parámetro del sistema indica que se desea almacenar el cuadro delimitador del objeto, paso 2: *a*) Basándose en la información del archivo de *groundtruth*, el sistema superpone uno o varios rectángulos sobre los fotograma. *b*) Se retorna al paso 3 del escenario principal.
Tipos de datos	Fotograma, objetos y vídeo

Frecuencia de ejecución	1 o 2 veces por cada dos semanas.
Importancia	Importante
Estabilidad	Alta
Criterios de validación	PV.14

CU.10	**Extraer vídeos con objetos pequeños de ILSVRC**
Actores	▪ Usuario
Descripción	Una base de datos ILSVRC facilitada por el Usuario es analizada para extraer, en formato ViTBAT, los vídeos que contengan objetos pequeños.
Relaciones	
Precondiciones	
Postcondiciones	▪ Uno o varios vídeos en formato ViTBAT pasan a encontrarse disponibles en el almacenamiento estático de la máquina utilizada.

Escenario principal	1. El Usuario selecciona una base de datos ILSVRC y se la facilita al sistema para la extracción de vídeos con objetos pequeños, indicando el tamaño máximo de los objetos y el número mínimo de fotogramas en los que deben aparecer. 2. El sistema analiza los vídeos de la base de datos, guardando en formato ViTBAT aquellos que cumplan con las especificaciones del Usuario. 3. El sistema le notifica al Usuario el éxito de la operación.
Escenarios alternativos	
Tipos de datos	Base de datos de vídeos y vídeos
Frecuencia de ejecución	1 o 2 veces por cada dos años.
Importancia	Estimulante
Estabilidad	Alta
Criterios de validación	PV.15

CU.11	**Convertir STDdb a formato de *benchmark***
Actores	▪ Usuario
Descripción	Una base de datos STDdb facilitada por el Usuario es transformada a formato OTB o VOT para poder ser utilizada para la evaluación de un modelo.

Relaciones	
Precondiciones	
Postcondiciones	▪ Una base de datos para *benchmark* en formato OTB o VOT pasa a encontrarse disponible en el almacenamiento estático de la máquina utilizada.
Escenario principal	1. El Usuario selecciona una base de datos STDdb y se la facilita al sistema para su conversión a formato de *benchmark*, indicando si desea la salida en OTB o VOT. 2. El sistema analiza los vídeos de la base de datos, guardándolos en formato OTB o VOT, en función de lo especificado por el Usuario. 3. El sistema le notifica al Usuario el éxito de la operación.
Escenarios alternativos	
Tipos de datos	Base de datos de vídeos y vídeos
Frecuencia de ejecución	1 o 2 veces por cada año.
Importancia	Estimulante
Estabilidad	Alta
Criterios de validación	PV.16, PV.17

CU.12	**Almacenar información de eventos**
Actores	▪ Sistema de *tracking* de alto nivel ▪ Usuario ▪ OTB Toolkit ▪ VOT Toolkit
Descripción	La información relativa a los logs generados durante la ejecución del modelo del sistema de *tracking* es almacenada de forma estática, para su posterior consulta.
Relaciones	▪ Extensión desde **CU-04 - Realizar inferencia sobre fotograma**. ▪ Extensión desde **CU-15 - Entrenar el sistema**. ▪ Extensión desde **CU-16 - Evaluar el modelo mediante VOT**. ▪ Extensión desde **CU-17 - Evaluar el modelo mediante OTB**.
Precondiciones	▪ Debe haberse realizado previamente alguna inferencia, paso de entrenamiento o paso de evaluación.
Postcondiciones	▪ La información de eventos del sistema pasa a encontrarse disponible en el almacenamiento estático de la máquina utilizada.
Escenario principal	1. De no existir, se crea un archivo para guardar los modelos del sistema. 2. Se añade al archivo estático la información del modelo de la última inferencia, paso de evaluación o paso de entrenamiento realizado. 3. El sistema notifica el éxito de la operación.

Escenarios alternativos	
Tipos de datos	Eventos del modelo
Frecuencia de ejecución	Entre 40 y 60 veces por segundo.
Importancia	Vital
Estabilidad	Alta
Criterios de validación	PV.18

Subsistema de entrenamiento

La responsabilidad de este subsistema es la de ofrecer una serie de funciones que permitan entrenar un modelo a partir de una base de datos de vídeos, así como almacenar toda la información relacionada del proceso.

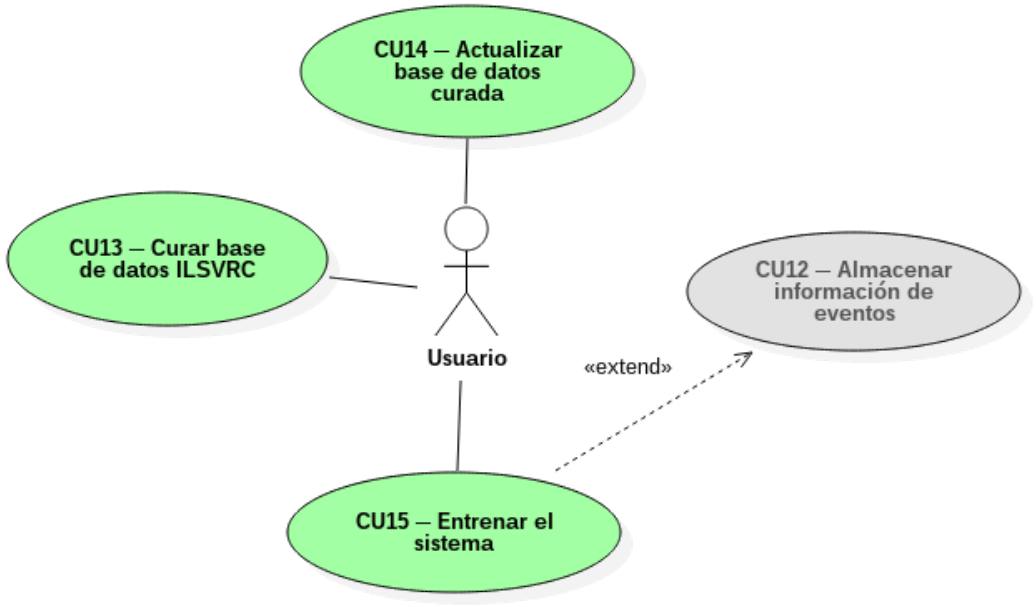


Figura 2.5: Casos de uso del subsistema de entrenamiento

CU.13	**Curar base de datos ILSVRC**
Actores	▪ Usuario
Descripción	Una base de datos ILSVRC facilitada por el Usuario es adaptada (curada) de forma que la herramienta de entrenamiento del sistema sea capaz de utilizarla sin realizar ningún tipo de transformación adicional sobre sus fotogramas.
Relaciones	
Precondiciones	
Postcondiciones	▪ La base de datos curada pasa a encontrarse disponible en el almacenamiento estático de la máquina utilizada.
Escenario principal	1. El Usuario selecciona una base de datos ILSVRC y se la facilita al sistema para su curación. 2. El sistema extrae una imagen ejemplar y de área de búsqueda de todos los objetos de la base de datos, por cada fotograma en el que aparecen. 3. El sistema agrupa los fotogramas extraidos en vídeos. 4. El sistema agrupa los vídeos extraidos en conjuntos de entrenamiento y evaluación. 5. El sistema extrae una serie de estadísticas de color de los vídeos. 6. El sistema le notifica al Usuario el éxito de la operación.
Escenarios alternativos	
Tipos de datos	Base de datos de vídeos, vídeos y estadísticas de color

Frecuencia de ejecución	1 o 2 veces por cada año.
Importancia	Vital
Estabilidad	Alta
Criterios de validación	PV.19

CU.14	**Actualizar base de datos curada**
Actores	▪ Usuario
Descripción	Los metadatos asociados a una base de datos curada, proporcionada por el Usuario, son actualizados para que se correspondan con la información que esta contiene.
Relaciones	
Precondiciones	
Postcondiciones	▪ La base de datos curada pasa a encontrarse en un estado coherente.
Escenario principal	1. El Usuario selecciona una base de datos curada que haya sido alterada manualmente y se la facilita al sistema para su actualización. 2. El sistema actualiza los metadatos de la base de datos curada, en función de la estructura de carpetas y archivos de la misma. 3. El sistema le notifica al Usuario el éxito de la operación.

Escenarios alternativos	
Tipos de datos	Base de datos de vídeos
Frecuencia de ejecución	1 o 2 veces por cada año.
Importancia	Importante
Estabilidad	Alta
Criterios de validación	PV.20

CU.15	**Entrenar el sistema**
Actores	▪ Usuario
Descripción	La red neuronal del sistema es entrenada sobre una base de datos curada, para minimizar el error de las inferencias realizadas. Como resultado se obtienen una serie de puntos de control del modelo.
Relaciones	▪ Extensión hacia **CU-14 - Almacenar información de eventos**.
Precondiciones	
Postcondiciones	▪ Una serie de puntos de control del modelo pasan a encontrarse disponibles en el almacenamiento estático de la máquina utilizada.

Escenario principal	1. El Usuario selecciona una base de datos curada y se la facilita al sistema para realizar un entrenamiento. 2. El sistema selecciona pares de áreas de búsqueda e imágenes ejemplares y realiza un aumento de las mismas basándose en las estadísticas de color de la base de datos y en los parámetros del sistema. 3. El sistema realiza la inferencia de los pares y utiliza el resultado para corregir el modelo 4. Los pasos 2 y 3 se repiten el número de veces indicado en los parámetros de entrenamiento. 5. Se almacena de forma estática un punto de control del sistema. 6. Los pasos 2 a 5 se repiten el número de veces indicado en los parámetros de entrenamiento. 7. El sistema le notifica al Usuario el éxito de la operación.
Escenarios alternativos	1. Un parámetro del sistema indica que la información de eventos debe ser guardada, paso 2: *a*) Se ejecuta el **CU-14 - Almacenar información de eventos**, pasándole como entrada el estado actual del sistema de *tracking*. *b*) Se retorna al paso 3 del escenario principal.
Tipos de datos	Base de datos de vídeos, puntos de control del modelo, estadísticas de color y *trainer*
Frecuencia de ejecución	1 o 2 veces por cada año.
Importancia	Vital
Estabilidad	Alta

Criterios de validación	PV.21

Subsistema de evaluación

La responsabilidad de este subsistema es la de ofrecer una serie de funciones que permitan evaluar un modelo según diversos *benchmarks*, así como obtener los puntos de control del modelo idóneos para las susodichas labores de *tracking*.

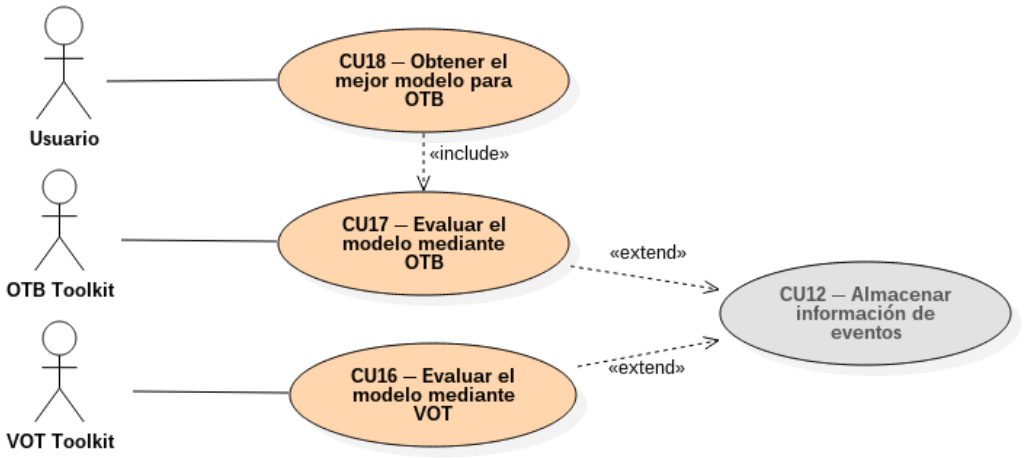


Figura 2.6: Casos de uso del subsistema de evaluación

CU.16	**Evaluar el modelo mediante VOT**
Actores	▪ VOT Toolkit
Descripción	Un punto de control del modelo es evaluado sobre un *benchmark* VOT, obteniendo la precisión (*accuracy*), robustez (*robustness*) y superposición esperada promedio (*average expected overlap*) del mismo.
Relaciones	▪ Extensión hacia **CU-14 - Almacenar información de eventos**.
Precondiciones	▪ El modelo debe haber sido entrenado y disponer de un punto de control válido a partir del cual realizar inferencias.
Postcondiciones	▪ La precisión, robustez y superposición esperada promedio del modelo pasan a encontrarse disponibles en el almacenamiento estático de la máquina utilizada.

Escenario principal	1. El VOT Toolkit selecciona un vídeo y le suministra al sistema su primer fotograma, así como la posición y dimensiones de un objeto. 2. El sistema almacena la posición y dimensiones del nuevo objeto, así como su imagen ejemplar. 3. El VOT Toolkit le suministra al sistema un nuevo fotograma. 4. El sistema modifica la información de posición y dimensiones asociadas al objeto seguido, de acuerdo a lo que aparece representado en la imagen. 5. La información actualizada del objeto le es transmitida al VOT Toolkit. 6. Los pasos 3 a 5 son repetidos hasta la finalización del vídeo. 7. Los pasos 1 a 6 son repetidos hasta que ya no existen más vídeos en el benchmark.
Escenarios alternativos	1. Un parámetro del sistema indica que la información de eventos debe ser guardada, paso 2: *a*) Se ejecuta el **CU-14 - Almacenar información de eventos**, pasándole como entrada el estado actual del sistema de *tracking*. *b*) Se retorna al paso 3 del escenario principal. 2. El VOT Toolkit determina que es necesario reiniciar el *tracker*, paso 5: *a*) El VOT Toolkit le suministra al sistema el fotograma actual y la posición y dimensiones de un objeto, como si se tratara de un nuevo vídeo. *b*) Se retorna al paso 2 del escenario principal.
Tipos de datos	Base de datos de vídeos, punto de control del modelo, objetos y *tracker*

Frecuencia de ejecución	1 o 2 veces por cada seis meses.
Importancia	Vital
Estabilidad	Alta
Criterios de validación	PV.22
Comentarios	Se ha decidido que este caso de uso sea independiente del **CU-17 - Evaluar el modelo mediante OTB** ya que, pese a que ambos persiguen el mismo propósito, los procesos para conseguirlos son muy diferentes.

CU.17	**Evaluar el modelo mediante OTB**
Actores	▪ OTB Toolkit
Descripción	Un punto de control del modelo es evaluado sobre un *benchmark* OTB, obteniendo el área bajo la curva (en función de la superposición límite) del mismo.
Relaciones	▪ Extensión hacia **CU-14 - Almacenar información de eventos**. ▪ Inclusión desde **CU-18 - Obtener el mejor modelo para OTB**.
Precondiciones	▪ El modelo debe haber sido entrenado y disponer de un punto de control válido a partir del cual realizar inferencias.

Postcondiciones	▪ El área bajo la curva en función de la superposición límite del modelo pasa a encontrarse disponible en el almacenamiento estático de la máquina utilizada.
Escenario principal	1. El OTB Toolkit selecciona un vídeo y le suministra al sistema todos sus fotogramas, así como la posición y dimensiones de un objeto en el primero de los mismos. 2. El sistema almacena la posición y dimensiones del nuevo objeto, así como su imagen ejemplar. 3. El sistema realiza la inferencia para todo el vídeo, retornando una lista de cuadros delimitadores. 4. Los pasos 1 a 3 son repetidos hasta que ya no existen más vídeos en el benchmark.
Escenarios alternativos	1. Un parámetro del sistema indica que la información de eventos debe ser guardada, paso 2: *a*) Se ejecuta el **CU-14 - Almacenar información de eventos**, pasándole como entrada el estado actual del sistema de *tracking*. *b*) Se retorna al paso 3 del escenario principal.
Tipos de datos	Base de datos de vídeos, punto de control del modelo, objetos y *tracker*
Frecuencia de ejecución	1 o 2 veces por cada seis meses, o 4 veces por hora bajo inclusión de **CU-18 - Obtener el mejor modelo para OTB**.
Importancia	Importante
Estabilidad	Alta
Criterios de validación	PV.23

Comentarios	Se ha decidido que este caso de uso sea independiente del **CU-16 - Evaluar el modelo mediante VOT** ya que, pese a que ambos persiguen el mismo propósito, los procesos para conseguirlos son muy diferentes.

CU.18	**Obtener el mejor modelo para OTB**
Actores	▪ Usuario
Descripción	Una serie de puntos de control del modelo son evaluados sobre un *benchmark* OTB, obteniendo el punto de control que ofrece una mayor área bajo la curva (en función de la superposición límite).
Relaciones	▪ Inclusión hacia **CU-17 - Evaluar el modelo mediante OTB**.
Precondiciones	▪ El modelo debe haber sido entrenado y disponer de más de un punto de control válido a partir del cual realizar inferencias.
Postcondiciones	▪ El área bajo la curva en función de la superposición límite de todos los modelo evaluados pasa a encontrarse disponible en el almacenamiento estático de la máquina utilizada, y el Usuario pasa a conocer el mejor punto de control.

Escenario principal	1. El Usuario selecciona un conjunto de puntos de control del modelo y se los suministra al sistema para obtener el mejor según OTB. 2. El sistema selecciona un punto de control y lo establece como el predeterminado del modelo. 3. Se ejecuta el **CU-17 - Evaluar el modelo mediante OTB**. 4. Los pasos 1 a 3 son repetidos hasta que ya no existen más puntos de control del modelo. 5. Se le notifica al Usuario el área bajo la curva obtenido por cada modelo.
Escenarios alternativos	
Tipos de datos	Puntos de control del modelo
Frecuencia de ejecución	1 o 2 veces por cada seis meses.
Importancia	Importante
Estabilidad	Media
Criterios de validación	PV.24

2.2.3. Requisitos no funcionales

En la presente sección se describirán todos aquellos requisitos no funcionales que deberá cumplir el sistema.

Cada requisito no funcional posee una serie de criterios de validación, los cuales pueden ser consultados en el Capítulo 5 de la presente memoria.

Requisitos de rendimiento

RN.01	**Velocidad de inferencia en tiempo real**
Descripción	El sistema deberá ser capaz de realizar inferencias de objetos simples en vídeos de alta resolución (1280 x 720) a una velocidad de al menos 30 imágenes por segundo. La velocidad será medida en el servidor de computación GPGPU facilitado por el CiTIUS.
Importancia	Vital
Estabilidad	Alta
Criterios de validación	PV.25

RN.02	**Espacio en memoria compatible con la NVIDIA Jetson TX2**
Descripción	El sistema deberá poder ser ejecutado en un módulo NVIDIA Jetson TX2 (con 8 GB de memoria de vídeo), teniendo en cuenta que un 50 % de la memoria del dispositivo podría encontrarse en uso por otras aplicaciones.
Importancia	Importante

Estabilidad	Media
Criterios de validación	PV.26

RN.03	**Rendimiento equiparable a SiamFC**
Descripción	El sistema deberá ofrecer un rendimiento equiparable (no menos de un 95 %) al obtenido por el tracker SiamFC. Concretamente: ▪ En el *benchmark* OTB-13, deberá alcanzar un área bajo la curva (en función de la superposición límite) de al menos 57,67 puntos. ▪ En el *benchmark* VOT-14, deberá alcanzar una precisión (*accuracy*) de al menos 59,30 puntos y una robustez (*robustness*) de al menos 82,18, así como una superposición esperada promedio (*average expected overlap*) de no menos de 23,90 puntos. ▪ En el *benchmark* VOT-15, deberá alcanzar una precisión (*accuracy*) de al menos 50.35 puntos y una robustez (*robustness*) de al menos 82.20, así como una superposición esperada promedio (*average expected overlap*) de no menos de 26,06 puntos.
Importancia	Vital
Estabilidad	Alta
Criterios de validación	PV.27

RN.04	**Inferencia eficiente de múltiples objetos**
Descripción	El sistema deberá ser capaz de realizar inferencias de múltiples objetos simultáneos sin replicar el modelo utilizado. Es decir, deberá adaptarse el modelo para que soporte múltiples objetos de forma nativa.
Importancia	Importante
Estabilidad	Alta
Criterios de validación	PV.28

Requisitos de datos

RN.05	**Compatibilidad con formatos de vídeo variados**
Descripción	El sistema deberá poder realizar inferencias a partir de vídeos en formato VOT, `.mp4` y dispositivos en *streaming*, como *webcams*.
Importancia	Importante
Estabilidad	Media
Criterios de validación	PV.29

RN.06	**Compatibilidad con objetos pequeños**
Descripción	El sistema deberá permitir el *tracking* de objetos pequeños (no más de 256 píxeles de área).
Importancia	Importante
Estabilidad	Alta
Criterios de validación	PV.30

Requisitos de usabilidad

RN.07	**Facilidad de despliegue**
Descripción	El sistema deberá ofrecer algún mecanismo de despliegue simplificado, para facilitar la realización de demostraciones en entornos compatibles. Esto podrá llevarse a cabo mediante la creación un paquete para Python o de una imagen para Docker.
Importancia	Estimulante
Estabilidad	Media
Criterios de validación	PV.31

RN.08	**Facilidad de uso**
Descripción	El sistema para inferencia deberá poder ser fácilmente usado por un usuario con conocimientos básicos de manejo de terminal, mediante la ejecución de un único comando con argumentos. No será necesario que el usuario realice ningún tipo de modificación del producto, ni que consulte ningún tipo de información más allá de la ayuda que ofrece el propio comando.
Importancia	Estimulante
Estabilidad	Media
Criterios de validación	PV.13

RN.09	**Comprensibilidad de la interfaz gráfica**
Descripción	La información mostrada de forma gráfica por el sistema durante el proceso de inferencia deberá ser intuitiva y comprensible para cualquier usuario que no se encuentre familiarizado con el *tracking* visual de objetos.
Importancia	Importante
Estabilidad	Alta
Criterios de validación	PV.09, PV.10, PV.11, PV.12

RN.10	**Manual en inglés**
Descripción	El manual del producto deberá encontrarse en inglés, uno de los 3 idiomas oficiales de la Universidad de Santiago de Compostela, y el más extendido para la realización de publicaciones científicas.
Importancia	Importante
Estabilidad	Alta
Criterios de validación	PV.13

Requisitos de mantenibilidad

RN.11	**Facilidad de adaptación**
Descripción	La modificación de los hiperparámetros del sistema deberá ser sencilla, editanto un único fichero. Asimismo, la incorporación de nuevos tipos de ramas para la red siamesa tendrá que poder hacerse mediante la modificación de no más de un archivo.
Importancia	Vital
Estabilidad	Alta
Criterios de validación	PV.32

Requisitos de implementación

RN.12	**Compatibilidad con TensorBoard**
Descripción	La informacón de eventos generada por el sistema debe ser compatible con TensorBoard, para permitir la visualización del grafo computacional de la red, así como del progreso de los entrenamientos, inferencias y validaciones.
Importancia	Vital
Estabilidad	Alta
Criterios de validación	PV.18

2.2.4. Matrices de trazabilidad

A continuación se recogen las matrices de trazabilidad consideradas más relevantes para la comprensión de los requisitos:

Matriz Casos de Uso – Objetivos del Sistema

	OS.01	OS.02	OS.03	OS.04	OS.05	OS.06
CU.01	↑	↑	↑			
CU.02	↑	↑	↑			
CU.03	↑	↑	↑			
CU.04	↑	↑	↑			
CU.05	↑				↑	
CU.06		↑			↑	
CU.07					↑	
CU.08	↑			↑	↑	
CU.09				↑		
CU.10			↑	↑		
CU.11				↑		↑
CU.12					↑	
CU.13				↑		
CU.14				↑		
CU.15				↑		
CU.16						↑
CU.17						↑
CU.18						↑

Tabla 2.52: Matriz de trazabilidad Casos de Uso – Objetivos del Sistema

Matriz Casos de Uso – Requisitos de Información

	RI.01	RI.02	RI.03	RI.04	RI.05	RI.06	RI.07	RI.08	RI.09	RI.10	RI.11	RI.12	RI.13	RI.14
CU.01	↱	↱	↱	↱	↱				↱			↱	↱	
CU.02	↱	↱							↱					
CU.03	↱	↱	↱	↱	↱				↱					
CU.04	↱	↱	↱	↱	↱	↱			↱	↱	↱	↱	↱	
CU.05	↱	↱	↱	↱	↱	↱								
CU.06	↱	↱				↱								
CU.07	↱	↱	↱	↱	↱	↱								
CU.08									↱					
CU.09	↱	↱				↱			↱					
CU.10	↱	↱				↱	↱	↱	↱					
CU.11	↱	↱				↱	↱	↱	↱					
CU.12			↱	↱	↱				↱		↱	↱	↱	↱
CU.13	↱	↱	↱	↱	↱	↱	↱	↱	↱					
CU.14			↱	↱	↱	↱	↱	↱	↱					
CU.15			↱	↱	↱		↱	↱	↱	↱	↱	↱		↱
CU.16	↱	↱	↱	↱	↱	↱	↱	↱	↱	↱	↱	↱	↱	
CU.17	↱	↱	↱	↱	↱	↱	↱	↱	↱	↱	↱	↱	↱	
CU.18	↱	↱	↱	↱	↱	↱	↱	↱	↱	↱	↱	↱	↱	

Tabla 2.53: Matriz de trazabilidad Casos de Uso – Requisitos de Información

2.3. Metodología de desarrollo

Dado que no existe ninguna clase de metodología ideal, válida para todo tipo de circunstancias, fue necesario analizar las características propias del proyecto y su marco de trabajo para poder escoger correctamente la metodología más apropiada.

Como punto de partida para la selección de la metodología de desarrollo, hubo que pesquisar cuál de las dos grandes corrientes del desarrollo software resultaría más apropiada para este caso: una metodología tradicional, centrada en el control del proceso, el seguimiento de las actividades y la disciplina en el trabajo, o una metodología ágil, enfocada en el factor humano y la adaptabilidad a los cambios, buscando modelos suficientemente buenos, no perfectos.

Una vez escogidos los principios en los que se basaría la metodología seguida, se hizo necesario decidir el ciclo de vida concreto a aplicar durante el proyecto. De entre todas las opciones ofrecidas por las metodologías pesadas, se decidió optar por un **ciclo de vida incremental**, dado que combina elementos del modelo

lineal en cascada con la filosofía interactiva de la construcción de prototipos. Esto permite partir de un plan sólido inicial, pero a la vez obtener una realimentación constante por parte de los tutores, definiendo un sistema que se actualizará con nuevas funcionalidades, lo cual resulta ideal dada la división tan clara de los módulos del producto.

Así, el proyecto contará primeramente con un análisis exhaustivo de los requisitos del sistema y del software, para luego llevar a cabo un diseño preliminar que permita orquestrar los diversos incrementos realizados, facilitando su integración. Tras esto, para cada incremento definido (que comprenderá el desarrollo de un conjunto de funcionalidades relacionadas), se llevará a cabo un diseño detallado del mismo, codificación y pruebas.

Algunas de las ventajas de esta elección son las descritas a continuación:

- El cliente obtiene resultados usables desde las primeras iteraciones, lo cual además mejora la validación del producto, detectando errores en fases tempranas y minimizando la no aceptación del resultado final.
- Resulta posible priorizar los requisitos definidos, implementando antes aquellos que el cliente considera más urgentes.
- Es posible gestionar las expectativas del cliente de forma regular, ya que este puede tomar decisiones en cada iteración.
- El esfuerzo dedicado a cada parte del sistema se encuentra claramente definido, permitiendo detectar variaciones importantes en la planificación, e impidiendo el gasto de más recursos de los dispuestos.
- Se facilita el desacople de los diversos módulos del sistema.
- Pese a que durante este proyecto no se dio la situación, permite getionar correctamente la incertidumbre, posibilitando que el cliente defina requisitos de alto nivel al comienzo del proyecto, que se irán especificando de forma más exhaustiva en las diversas iteraciones.

No obstante, siguen existiendo una serie de restricciones dada la elección de este ciclo de vida. Si bien su influencia no resulta excesivamente impactante en el presente proyecto, es necesario tenerlas presentes:

- La disponibilidad del cliente debe ser alta durante todo el proyecto, dado que deberá participar de forma continua.
- Existe una seria dificultad para detectar si los requisitos definidos son verdaderamente válidos.
- Cada iteración debe aportar valor para el cliente, dando como resultado una serie de requisitos terminados que no dejen tareas pendientes.
- La relación con el cliente debe estar basada en una colaboración simbiótica, más allá de una mera relación contractual en la que cada parte únicamente piensa en su beneficio a corto plazo.
- Si bien existe cierta tolerancia a los cambios y a la incertidumbre, un cambio radical en alguno de los requisitos base resultaría muy costoso de gestionar.
- Se depende de gran manera del correcto diseño preliminar del sistema. Si este resultara incorrecto, la integración de los diversos incrementos resultaría excesivamente compleja, siendo necesario realizar una gran cantidad de retrabajo.

2.4. Gestión del tiempo

En la presente sección se incluyen todos los procesos requeridos para administrar la finalización del proyecto dentro de los plazos previstos.

2.4.1. Estructura de Descomposición del Trabajo (EDT)

En la presente subsección se describe la Estructura de Descomposición del Trabajo seguida para el presente proyecto. Este EDT representa de forma jerárquica los principales paquetes de actividades que conforman el trabajo a ser ejecutado por el alumno. Su propósito es el de organizar y definir el alcance total aprobado, sirviendo como base para la planificación del proyecto.

El susodicho EDT es el mostrado a continuación, en la Figura 2.7:

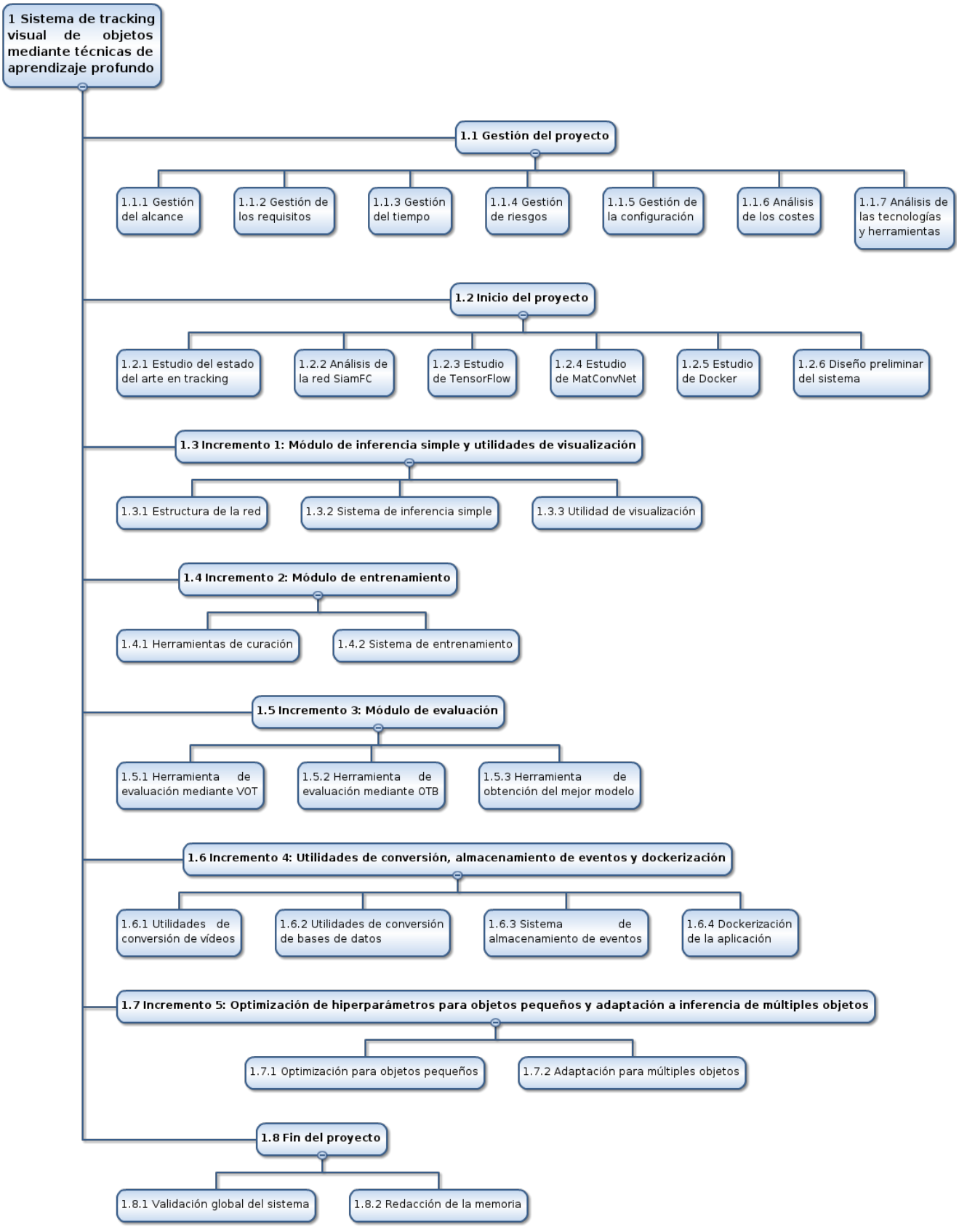


Figura 2.7: EDT del proyecto

2.4.2. Definición de las actividades

A continuación se describen las actividades que conforman los paquetes de trabajo indicados en el EDT del proyecto, las cuales se estiman con una duración de **413 horas** de ejecución (de las cuales, 48 horas son de colchón) y **24 horas** de planificación:

Gestión del proyecto

Con una duración estimada de **6 horas** en ejecución y **24 horas** en planificación, estas actividades involucran la planificación, la organización y el control de los recursos con el propósito de alcanzar los objetivos fijados por los requisitos del proyecto.

De forma más concreta, las actividades de gestión consideradas durante el presente proyecto son las siguientes:

Gestión del alcance: Comprende los procesos necesarios para garantizar que el proyecto incorpora todo el trabajo requerido para completarlo con éxito, definiendo y controlando qué se incluye y qué no se incluye.

Gestión de los requisitos: Comprende todos los procesos necesarios para identificar, documentar, mantener, comunicar y trazar los requisitos a lo largo del ciclo de vida del proyecto.

Gestión del tiempo: Comprende todas las actividades requeridas para garantizar la finalización del proyecto dentro del plazo fijado, estableciendo una planificación del cronograma y asegurando su cumplimiento.

Gestión de riesgos: Comprende aquellas actividades que permiten identificar, analizar y responder a factores de riesgo a lo largo del ciclo de vida del proyecto.

Gestión de la configuración: Comprende las actividades que permiten asegurar la calidad del producto a través del estricto control de los cambios realizados y de la disponibilidad constante de una versión estable de cada elemento.

Análisis de los costes: Comprende aquellas actividades que permiten estimar y asignar los costes del proyecto. Dado que no se percibe ningún ingreso por la realización de este proyecto ni se ha realizado ningún gasto monetario (pues todos los recursos requeridos ya habían sido previamente adquiridos), no ha sido necesario definir ninguna actividad de control de costes.

Análisis de las tecnologías y herramientas: Si bien no es una actividad de gestión de proyectos como tal, el análisis de las tecnologías y las herramientas supuso un aspecto crucial para definir el plan de proyecto y su organización y control.

Inicio del proyecto

Con una duración estimada de **54 horas** en ejecución (de las cuales, 6 horas son de colchón), las actividades de inicio del proyecto incluyen la formación del alumno en las tecnologías utilizadas y el diseño preliminar del sistema, propio de los ciclos de vida incrementales.

De forma más concreta, las actividades de inicio del proyecto consideradas son las siguientes:

Estudio del estado del arte en *tracking*: Se trata de una actividad de formación, consistente en la investigación documental del conocimiento acumulado dentro del área del *tracking* visual de objetos mediante CNNs.

Análisis de la red SiamFC: Se trata de una actividad de formación, consistente en el análisis del funcionamiento y características de la red neuronal convolucional y siamesa SiamFC [7].

Estudio de TensorFlow: Se trata de una actividad de formación, consistente en el estudio de la biblioteca TensorFlow, para hacer posible su utilización durante el proyecto.

Estudio de MatConvet: Se trata de una actividad de formación, consistente en el estudio de la herramienta MatConvet, para hacer más sencilla la comprensión del código fuente de SiamFC.

Estudio de Docker: Se trata de una actividad de formación, consistente en el

estudio de la tecnología Docker, para permitir que el producto desarrollado sea portable.

Diseño preliminar del sistema: Comprende las actividades propias de un ciclo de vida incremental que permiten obtener un diseño preliminar del sistema, a partir del cual se irán elaborando sus diversos módulos.

Incremento 1: Módulo de inferencia simple y utilidades de visualización

Con una duración estimada de **81 horas** en ejecución (de las cuales, 6 horas son de colchón), las actividades de este primer incremento se centran en desarrollar un módulo de inferencia simple (*tracking* de un único objeto) y una utilidad de visualización, que permita inducir si la inferencia es correcta.

De forma más concreta, las actividades consideradas durante el primer incremento son las siguientes:

Estructura de la red: Estas actividades comprenden el diseño detallado e implementación de la estructura de la red de inferencia para un único objeto (no varios a la vez). Dado que la red no se encuentra entrenada, no es posible realizar ninguna prueba más allá de comprobar que las dimensiones de las diversas capas son compatibles.

Sistema de inferencia simple: Estas actividades comprenden el diseño detallado, la implementación y las pruebas del sistema que permiten la entrada de fotogramas en la red y la interpretación de la salida de la misma, para un único objeto (no varios a la vez). También se crearán los manuales técnico y de usuario del módulo de inferencia.

Utilidad de visualización: Estas actividades comprenden el diseño detallado, la implementación y las pruebas de la utilidad de visualización, la cual permite inducir de forma sencilla si el sistema funciona de la forma correcta.

Incremento 2: Módulo de entrenamiento

Con una duración estimada de **60 horas** en ejecución (de las cuales, 6 horas son de colchón), las actividades de este segundo incremento se centran en desarrollar un módulo de entrenamiento que permita curar bases de datos de vídeos y entrenar la red a partir de esta información.

De forma más concreta, las actividades consideradas durante el segundo incremento son las siguientes:

Herramientas de curación: Estas actividades comprenden el diseño detallado, implementación y pruebas de las herramientas que permitirán pasar de una base de datos en un formato poco apropiado para el entrenamiento (como ILSVRC) a una base de datos de vídeos curada, que requiera de un preprocesamiento mínimo durante el aprendizaje. También se crearán los manuales técnico y de usuario de las herramientas de curación.

Sistema de entrenamiento: Estas actividades comprenden el diseño detallado, la implementación y las pruebas del sistema que permite que el modelo de red definido pueda aprender las características (*features*) necesarias para realizar inferencias correctas. La prueba unitaria de este sistema no resulta posible, pues únicamente puede validarse junto al de inferencia (y viceversa). También se crearán los manuales técnico y de usuario del módulo de entrenamiento.

Incremento 3: Módulo de evaluación

Con una duración estimada de **31 horas** en ejecución (de las cuales, 6 horas son de colchón), las actividades de este tercer incremento se centran en desarrollar un módulo que permita la evaluación del modelo mediante diversos *benchmarks* y la selección de la mejor época (*epoch*).

De forma más concreta, las actividades consideradas durante el tercer incremento son las siguientes:

Herramientas de evaluación mediante VOT: Partiendo de la funcionalidad de inferencia, estas actividades comprenden el diseño detallado, imple-

mentación y pruebas de un adaptador que haga al sistema compatible con el protocolo de *tracking* VOT.

Herramientas de evaluación mediante OTB: Partiendo de la funcionalidad de inferencia, estas actividades comprenden el diseño detallado, implementación y pruebas de un adaptador que haga al sistema compatible con el protocolo de *tracking* OTB.

Herramientas de obtención del mejor modelo: Haciendo uso de las evaluaciones objetivas que ofrecen VOT y OTB, estas actividades comprenden el diseño detallado, implementación y pruebas de un conjunto de herramientas que permita seleccionar la mejor época (*epoch*) obtenida durante el entrenamiento, de forma automática. También se crearán los manuales técnico y de usuario de las herramientas de obtención del mejor modelo.

Incremento 4: Utilidades de conversión, almacenamiento de eventos y dockerización

Con una duración estimada de **48 horas** en ejecución (de las cuales, 6 horas son de colchón), las actividades de este cuarto incremento se centran crear utilidades que hagan que el sistema resulte más polivalente y accesible, siendo capaz de trabajar con diversos formatos. En este incremento también se tratará la dockerización de la aplicación y el almacenamiento en tiempo real de los eventos del modelo.

De forma más concreta, las actividades consideradas durante el cuarto incremento son las siguientes:

Utilidades de conversión de vídeos: Estas actividades comprenden el diseño, implementación y pruebas de una serie de utilidades que permitirán al sistema trabajar con diversos formatos de vídeo y que estos formatos de vídeo resulten fácilmente visualizables para un humano. También se crearán los manuales técnico y de usuario de las utilidades de conversión de vídeos.

Utilidades de conversión de bases de datos: Estas actividades comprenden el diseño, implementación y pruebas de una serie de utilidades que permitirán la conversión de formatos de bases de datos de vídeos, para que el sistema pueda entrenar con información proveniente de más fuentes. También

se crearán los manuales técnico y de usuario de las utilidades de conversión de bases de datos.

Sistema de almacenamiento de eventos: Estas actividades comprenden el diseño, implementación y pruebas de un sistema que, durante la inferencia, entrenamiento y evaluación, recoja en tiempo real los eventos producidos por el modelo para su posterior análisis.

Dockerización de la aplicación: Estas actividades comprenden el diseño, implementación y pruebas de la inclusión de la aplicación en un contenedor Docker, para que sea portable y resulte transparente de utilizar. También se crearán los manuales técnico y de usuario de la aplicación dockerizada.

Incremento 5: Optimización de hiperparámetros para objetos pequeños y adaptación de inferencia de múltples objetos

Con una duración estimada de **67 horas** en ejecución (de las cuales, 6 horas son de colchón), las actividades de este quinto incremento se centran en adaptar el modelo desarrollado para que permita el *tracking* de múltiples objetos simultáneos y el seguimiento de objetos pequeños (hasta 256 píxeles de área).

De forma más concreta, las actividades consideradas durante el quinto incremento son las siguientes:

Optimización para objetos pequeños: Estas actividades comprenden el diseño, implementación y pruebas de una serie de optimizaciones tanto en hiperparámetros como en la lógica de procesado, que permitan que el sistema pueda realizar inferencias sobre objetos pequeños.

Adaptación para múltiples objetos: Estas actividades comprenden el diseño, implementación y pruebas de un nuevo modelo que permita el procesado de múltiples objetos simultáneos de forma eficiente.

Fin del proyecto

Con una duración estimada de **66 horas** en ejecución (de las cuales, 12 horas son de colchón), las actividades de fin del proyecto incluyen todas aquellas acciones necesarias para garantizar un correcto cierre de proyecto.

De forma más concreta, las actividades de inicio del proyecto consideradas son las siguientes:

Validación global del sistema: Estas actividades buscan asegurar que el producto desarrollado es correcto, comprobando que se cumplen todos los criterios de validación establecidos.

Redacción de la memoria: Si bien los diversos paquetes de trabajo anteriormente descritos incluyen la documentación de los mismos, las actividades aquí recogidas buscan completar las partes de la memoria faltantes, así como asegurar la corrección de la misma.

2.4.3. Plan de proyecto inicial

En este apartado se muestra el diagrama de Gantt de la planificación inicial del proyecto, el cual recoge una lista de las actividades en las que se este divide, su duración y dependencias.

Este diagrama de Gantt se encuentra estrechamente relacionado con el EDT descrito en el apartado anterior, y asume que la ejecución del proyecto comenzará el día 1 de marzo de 2018, con una dedicación semanal estimada de 30 horas (6 horas diarias, fines de semana excluidos).

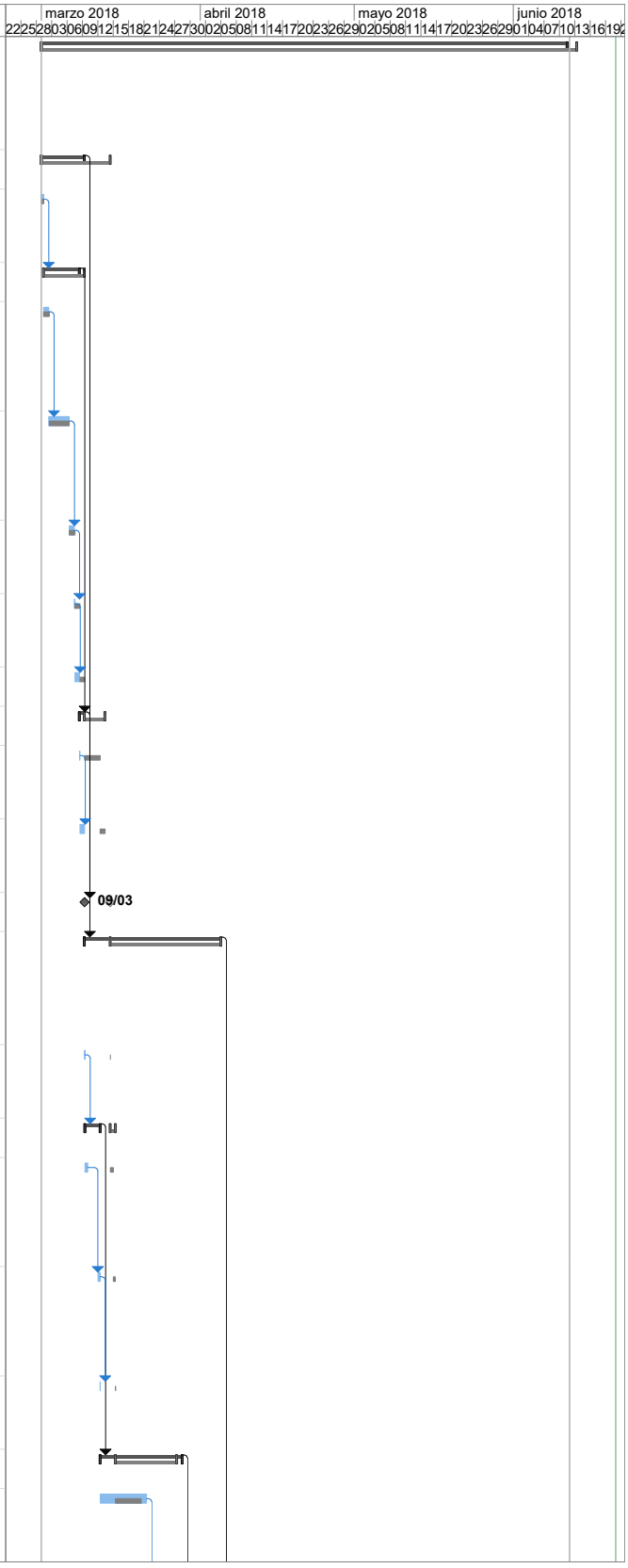
De este modo, y teniendo en cuenta los días festivos descritos en la tabla 2.54, el proyecto quedaría finalizado y listo para su entrega el día 13 de junio de 2018.

Días festivos durante el proyecto	
Fecha	Nombre
29/03/2018	Jueves Santo
30/03/2018	Viernes Santo
01/05/2018	Fiesta del Trabajo
10/05/2018	Día de la Ascensión
17/05/2018	Día de las Letras Gallegas

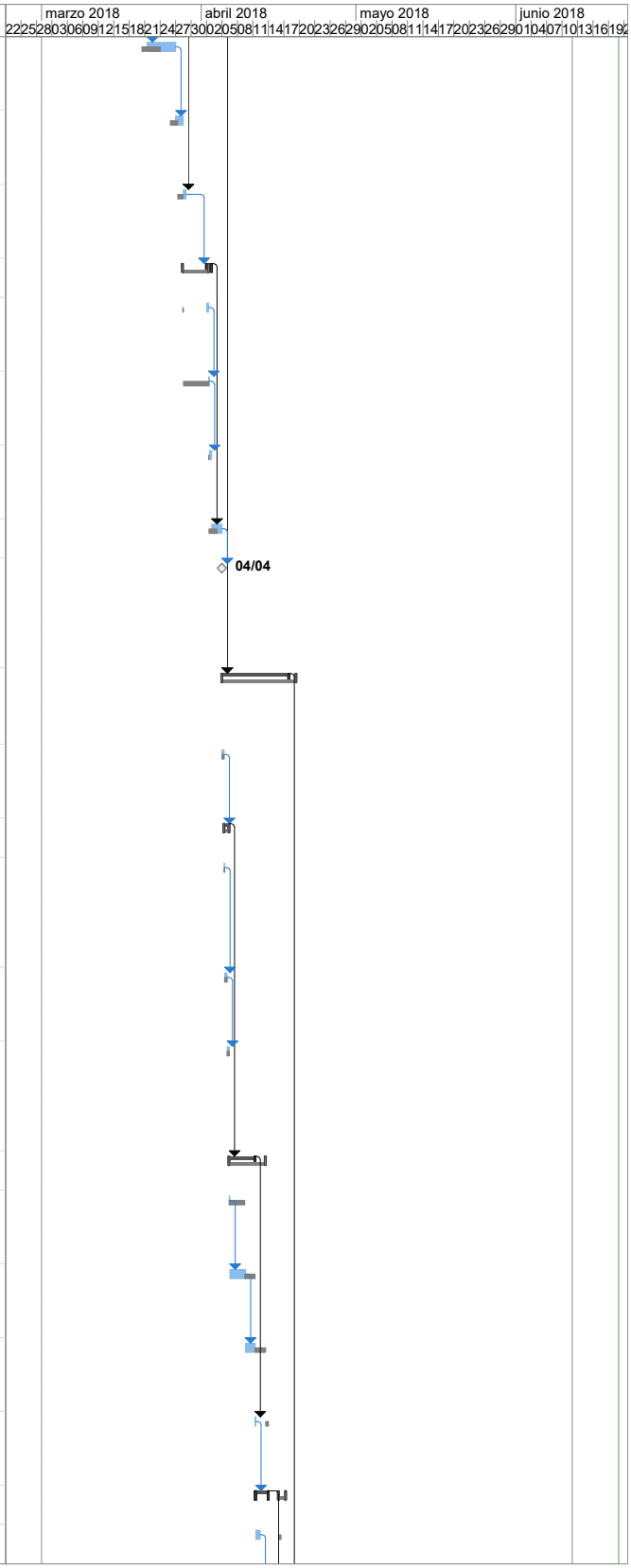
Tabla 2.54: Días festivos durante el proyecto

Nótese que en este diagrama de Gantt se muestran tanto la planificación inicial (en forma de línea base) como el desarrollo real. En la siguiente subsección se tratarán en detalle las causas que dieron lugar a las variaciones en el cronograma.

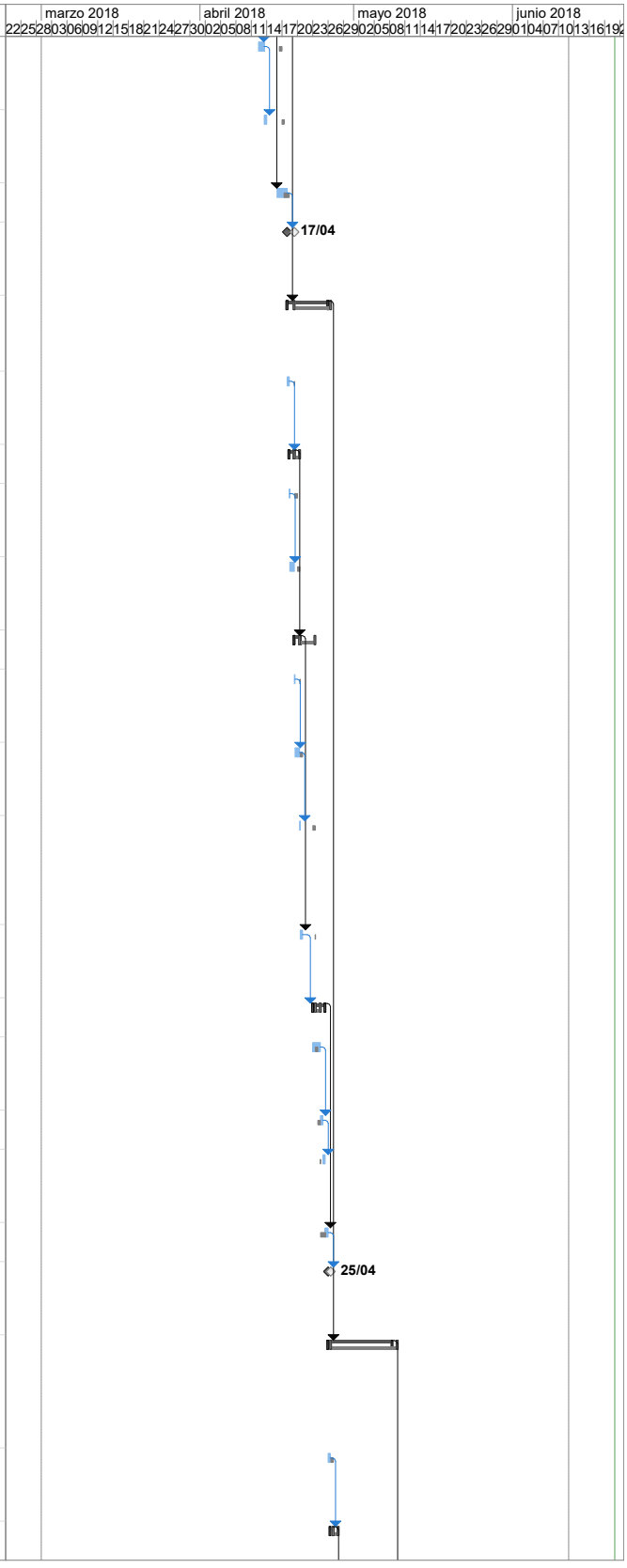
Id	Nombre de tarea	Duración	Comienzo
1	**Sistema de tracking visual de objetos mediante técnicas de aprendizaje profundo**	**67,75 días**	**jue 01/03/18**
2	**Inicio del proyecto**	**6,25 días**	**jue 01/03/18**
3	Seguimiento y control del proyecto	0,25 días	jue 01/03/18
4	**Formación**	**5 días**	**jue 01/03/18**
5	Estudio del estado del arte en tracking de objetos mediante CNNs	1 día	jue 01/03/18
6	Análisis del funcionamiento y características de la red SiamFC	2 días	vie 02/03/18
7	Formación y estudio de TensorFlow	1 día	mar 06/03/18
8	Formación y estudio de MatConvNet	0,5 días	mié 07/03/18
9	Formación y estudio de Docker	0,5 días	mié 07/03/18
10	**Diseño preliminar del sistema**	**1 día**	**jue 08/03/18**
11	Creación del diagrama de casos de uso	0,5 días	jue 08/03/18
12	Creación de los diagramas de contexto y sistema	0,5 días	jue 08/03/18
13	Marco de trabajo preparado	0 días	vie 09/03/18
14	**Incremento 1: Módulo de inferencia simple y utilidades de visualización**	**16,75 días**	**vie 09/03/18**
15	Seguimiento y control del proyecto	0,25 días	vie 09/03/18
16	**Diseño detallado**	**1 día**	**vie 09/03/18**
17	Creación del diagrama de clases del módulo de inferencia simple	0,5 días	vie 09/03/18
18	Diseño de la estructura de capas de la red de inferencia simple	0,25 días	lun 12/03/18
19	Diseño de la interfaz de usuario	0,25 días	lun 12/03/18
20	**Codificación**	**12 días**	**lun 12/03/18**
21	Creación de la estructura de la red	7 días	lun 12/03/18



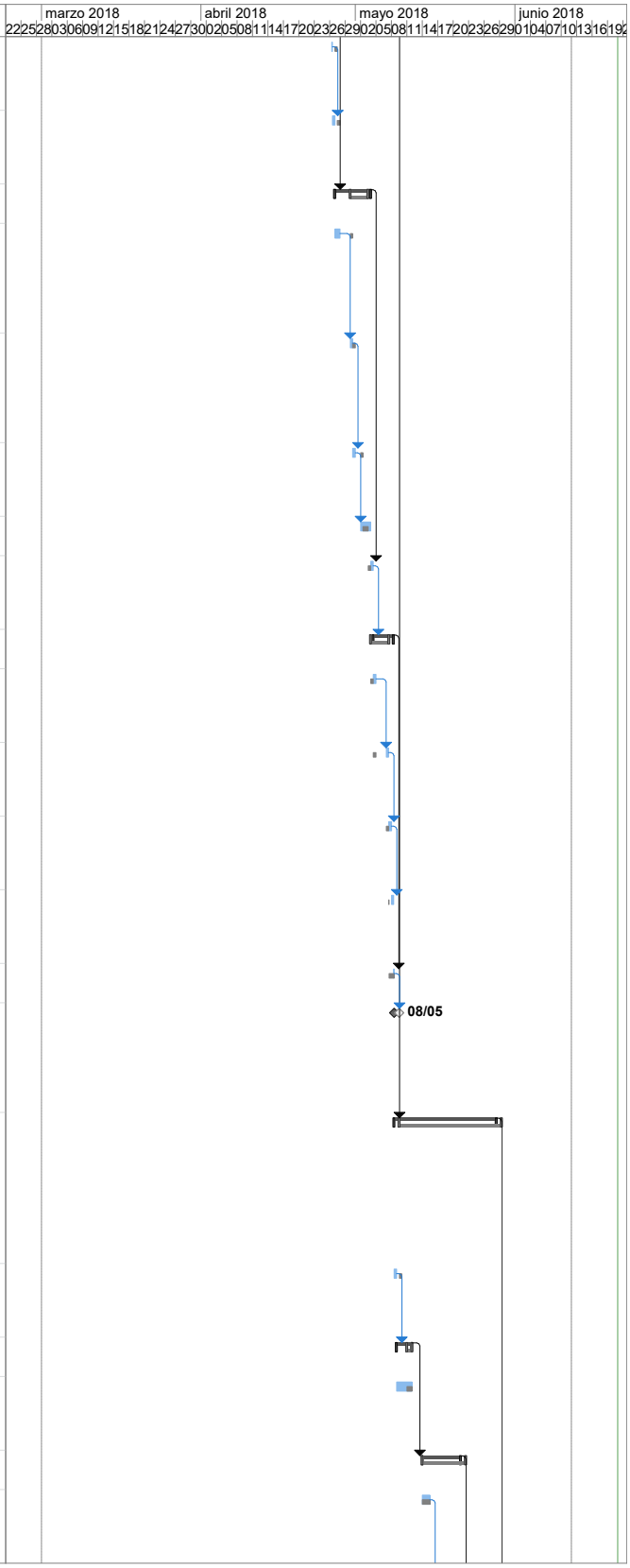
Id	Nombre de tarea	Duración	Comienzo
22	Implementación del módulo de inferencia simple	3,5 días	mié 21/03/18
23	Implementación de la utilidad de visualización	1,5 días	mar 27/03/18
24	Creación de los manuales técnico y de usuario	0,5 días	mié 28/03/18
25	**Pruebas**	**1 día**	**lun 02/04/18**
26	Prueba unitaria de la utilidad de visualización	0,25 días	lun 02/04/18
27	Prueba unitaria del actualizador de objetos	0,5 días	lun 02/04/18
28	Pruebas de aceptación del incremento	0,25 días	lun 02/04/18
29	Documentación del incremento	2 días	mar 03/04/18
30	Incremento 1: Módulo de inferencia simple y utilidades de visualización	0 días	mié 04/04/18
31	**Incremento 2: Módulo de entrenamiento**	**9 días**	**jue 05/04/18**
32	Seguimiento y control del proyecto	0,25 días	jue 05/04/18
33	**Diseño detallado**	**1 día**	**jue 05/04/18**
34	Diseño del modelo E-R de la base de datos de vídeos curada	0,5 días	jue 05/04/18
35	Diseño del diagrama de clases del módulo de entrenamiento	0,25 días	jue 05/04/18
36	Diseño de la estructura de capas de la red de inferencia simple	0,25 días	vie 06/04/18
37	**Codificación**	**3 días**	**vie 06/04/18**
38	Creación de la estructura de la red	0,5 días	vie 06/04/18
39	Implementación de las herramientas de curación	1 día	vie 06/04/18
40	Implementación del sistema de entrenamiento	1,5 días	lun 09/04/18
41	Creación de los manuales técnico y de usuario	0,5 días	mié 11/04/18
42	**Pruebas**	**2,25 días**	**mié 11/04/18**
43	Entrenamiento de la red	0,5 días	mié 11/04/18



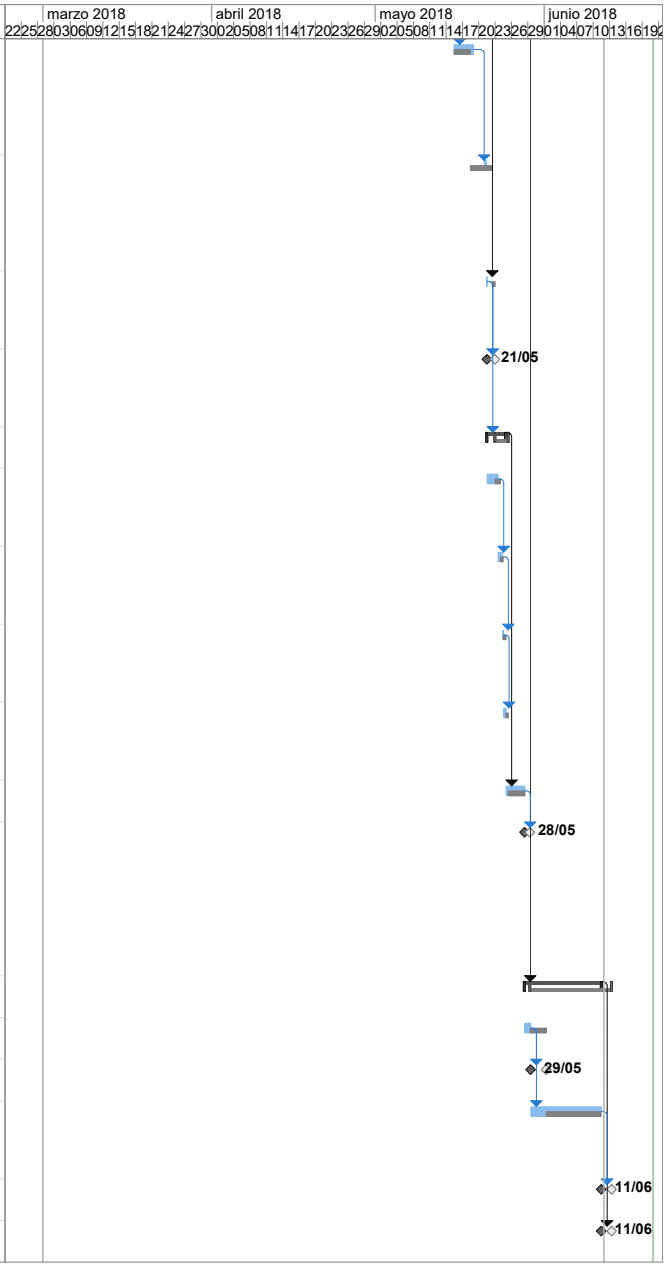
Id	Nombre de tarea	Duración	Comienzo
44	Prueba de inferencia con vídeos de ejemplo	1,5 días	jue 12/04/18
45	Pruebas de aceptación del incremento	0,25 días	vie 13/04/18
46	Documentación del incremento	2 días	lun 16/04/18
47	Incremento 2: Módulo de entrenamiento	0 días	mar 17/04/18
48	**Incremento 3: Módulo de evaluación**	**6 días**	**mié 18/04/18**
49	Seguimiento y control del proyecto	0,25 días	mié 18/04/18
50	**Diseño detallado**	**1 día**	**mié 18/04/18**
51	Diseño de la interacción con el protocolo de VOT	0,5 días	mié 18/04/18
52	Diseño de la interacción con el protocolo de OTB	0,5 días	mié 18/04/18
53	**Codificación**	**1,5 días**	**jue 19/04/18**
54	Implementación del adaptador a VOT	0,5 días	jue 19/04/18
55	Implementación del adaptador a OTB	0,5 días	jue 19/04/18
56	Implementación de la herramienta de obtención del mejor modelo	0,5 días	vie 20/04/18
57	Creación de los manuales técnico y de usuario	0,25 días	vie 20/04/18
58	**Pruebas**	**2,25 días**	**lun 23/04/18**
59	Prueba de compatibilidad de protocolos	1,5 días	lun 23/04/18
60	Evaluación del modelo	0,5 días	mar 24/04/18
61	Pruebas de aceptación del incremento	0,25 días	mié 25/04/18
62	Documentación del incremento	0,75 días	mié 25/04/18
63	Incremento 3: Módulo de evaluación	0 días	mié 25/04/18
64	**Incremento 4: Utilidades de conversión, almacenamiento de eventos y dockerización**	**7,75 días**	**jue 26/04/18**
65	Seguimiento y control del proyecto	0,25 días	jue 26/04/18
66	**Diseño detallado**	**0,75 días**	**jue 26/04/18**



Id	Nombre de tarea	Duración	Comienzo
67	Creación del diagrama de clases de las utilidades	0,5 días	jue 26/04/18
68	Establecimiento del contexto de la aplicación Docker	0,25 días	jue 26/04/18
69	**Codificación**	**4 días**	**vie 27/04/18**
70	Implementación de las utilidades de conversión de vídeos	1 día	vie 27/04/18
71	Implementación de las utilidades de conversión de bases de datos	0,5 días	lun 30/04/18
72	Implementación del almacenamiento de eventos	0,5 días	lun 30/04/18
73	Dockerización de la aplicación	2 días	mié 02/05/18
74	Creación de los manuales técnico y de usuario	0,5 días	vie 04/05/18
75	**Pruebas**	**1,75 días**	**vie 04/05/18**
76	Prueba de las utilidades de conversión	0,5 días	vie 04/05/18
77	Prueba del almacenamiento y visualización de eventos	0,5 días	lun 07/05/18
78	Prueba de sistema de la aplicación dockerizada	0,5 días	lun 07/05/18
79	Pruebas de aceptación del incremento	0,25 días	mar 08/05/18
80	Documentación del incremento	0,5 días	mar 08/05/18
81	Incremento 4: Utilidades de conversión, almacenamiento de eventos y dockerización	0 días	mar 08/05/18
82	**Incremento 5: Optimización de hiperparámetros para objetos pequeños y adaptación a inferencia de múltiples objetos**	**11,5 días**	**mar 08/05/18**
83	Seguimiento y control del proyecto	0,25 días	mar 08/05/18
84	**Diseño detallado**	**2 días**	**mié 09/05/18**
85	Diseño de la red para inferencia de múltiples objetos	2 días	mié 09/05/18
86	**Codificación**	**4,5 días**	**lun 14/05/18**
87	Creación de la estructura de la red	1,5 días	lun 14/05/18



Id	Nombre de tarea	Duración	Comienzo
88	Implementación del sistema de inferencia para múltiples objetos	2,5 días	mar 15/05/18
89	Establecimiento de los hiperparámetros para objetos pequeños	0,5 días	lun 21/05/18
90	Creación de los manuales técnicos y de usuario	0,25 días	lun 21/05/18
91	Manuales técnico y de usuario finales	0 días	lun 21/05/18
92	**Pruebas**	**3,25 días**	**lun 21/05/18**
93	Entrenamiento y evaluación final del modelo	2 días	lun 21/05/18
94	Prueba de la inferencia múltiple	0,5 días	mié 23/05/18
95	Evaluación de la inferencia de objetos pequeños	0,5 días	jue 24/05/18
96	Pruebas de aceptación del incremento	0,25 días	jue 24/05/18
97	Documentación del incremento	1,25 días	vie 25/05/18
98	Incremento 5: Optimización de hiperparámetros para objetos pequeños y adaptación a inferencia de múltiples objetos	0 días	lun 28/05/18
99	**Fin del proyecto**	**10,5 días**	**lun 28/05/18**
100	Validación global del sistema	1,5 días	lun 28/05/18
101	Producto finalizado	0 días	mar 29/05/18
102	Redacción de la memoria del proyecto	9 días	mar 29/05/18
103	Memoria finalizada	0 días	lun 11/06/18
104	Proyecto finalizado	0 días	lun 11/06/18



2.4.4. Plan de proyecto real

El cumplimiento exacto del plan inicial representa el caso ideal, pero altamente improbable, en el cual el desarrollo del proyecto se ha llevado a cabo tal cual se planificó. No obstante, como en todo proyecto software, se han producido una serie de desviaciones fruto de algunos riesgos materializados y la falta de experiencia en la planificación de proyectos informáticos.

De esta forma, pese a que la ejecución del proyecto sí comenzó el día 1 de marzo, tal y como estaba planeado, su finalización se produjo el día 11 de junio, 2 días antes de lo previsto, tras **406 horas y media** de trabajo, consumiendo casi todos los colchones.

La mayor parte de las actividades definidas fueron completadas dentro del tiempo previsto, con ligeras variaciones tanto a la alza como a la baja. No obstante, se dieron una serie de sucesos que provocaron variaciones más sustanciales, los cuales se describen a continuación:

- El tiempo requerido para la formación y del diseño preliminar fueron menores de lo estimado en un primer momento, finalizando 30 horas antes (6 días) de lo esperado. Esto se debió en gran medida a que el alumno ya era conocedor de algunas nociones básicas de aprendizaje automático y del uso general de sus librerías.

- La creación de la estructura de la red para inferencia no fue tan sencilla como se estimaba, ya que en el artículo publicado de SiamFC [7] se reflejaba una versión inconsistente de la red, distinta de lo que se muestra en el repositorio (**RSG.12**). Además, las normalizaciones por lotes aplicadas por defecto en TensorFlow resultaban distintas de las de MatConvNet, lo cual fue costoso de identificar (**RSG.30**). Esto supuso 18 horas adicionales de trabajo (3 días).

- La integración del modelo entrenado con el módulo de inferencia no fue directa, dado que existió una incompatibilidad entre tipos de datos con y sin signo (**RSG.07**). Esto supuso 6 horas (1 día) adicional de trabajo.

- La creación del módulo de evaluación mediante OTB no fue tan inmediata

como se creía en un principio, dada la prácticamente nula documentación de la herramienta (**RSG.30**). Esto supuso 6 horas adicionales de trabajo (1 día).

- La dockerización de la aplicación no resultó tan trivial como se había estimado desde un primer momento, dado que algunas de las librerías utilizadas carecían de todas sus funcionalidades al ser instaladas desde binarios (**RSG.11**). Esto supuso 6 horas adicionales de trabajo (1 día).
- La validación global del sistema y las pruebas de aceptación realizadas no fueron tan largas como se predijo en un primer momento, por lo que se requirieron en torno a 18 horas menos (3 días) de trabajo.
- Dado que de cara al final del proyecto se estaba en una situación favorable con respecto a la planificación, se decidió invertir el tiempo sobrante en ultimar la redacción final de la memoria, dedicándole 18 horas adicionales (3 días) de trabajo.

2.5. Gestión de riesgos

En este apartado de la memoria se procederá a detallar el plan de gestión de riesgos del proyecto. Dada la naturaleza específica del mismo, existen numerosos riesgos que deberán ser debidamente identificados y gestionados para evitar retrasos en la planificación, así como asegurar la calidad del producto y el cumplimiento de los objetivos.

2.5.1. Planificación de la gestión de riesgos

Metodología de identificación de riesgos

Para realizar la identificación de riesgos del proyecto, se utilizaron las siguientes técnicas:

Entrevistas: En primer lugar, se llevaron a cabo diversas reuniones con los tutores del proyecto, Don Manuel Mucientes Molina y Don Víctor Manuel

Brea Sánchez, durante la toma de requisitos, para la detección de aquellos riesgos que estuvieran directamente relacionados los usuarios finales del producto y la integración del mismo en un entorno en producción real.

Checklist: Seguidamente, se revisó un *checklist* elaborado por la Universidad de Castilla - La Mancha [10] que contenía una serie de riesgos para proyectos software. De esta lista, fueron elegidos aquellos los riesgos aplicables a este proyecto, los cuales fueron planteados de forma que resultaran específicos y bien definidos.

Análisis de proyectos pasados: A continuación, fue llevado a cabo un análisis de los riesgos considerados y experimentados a lo largo de otros proyectos realizados en el pasado. Si bien estos proyectos tenían una menor envergadura que el recogido en el presente documento, fueron una valiosa fuente de ideas y lecciones aprendidas.

Revisión de documentación: Durante el desarrollo de todo el proyecto, se realizaron revisiones periódicas de la documentación elaborada hasta el momento, para identificar nuevos riesgos o bien ajustar el análisis de los riesgos que han sido previamente identificados.

Metodología de análisis de riesgos

Durante el análisis de riesgos se llevó a cabo una **priorización cualitativa de riesgos**. Es decir, para cada uno de los riesgos detectados, se realizó el cálculo cualitativo de la **exposición** del mismo a partir de su **probabilidad de ocurrencia** real (Tabla 2.55) e **impacto sobre el proyecto** (Tabla 2.56), tal y como figura en la Tabla 2.57.

Ocurrencia del riesgo		Probabilidad
$\geq 80\,\%$	Debe asumirse que se va a producir en algún momento del desarrollo del proyecto, incluso en más de una ocasión.	Alta
Entre 30 % y 80 %	Deberá asumirse que alguno de los riesgos con esta probabilidad se manifieste durante el desarrollo del proyecto.	Media
$\leq 30\,\%$	No aparecerá durante el desarrollo del proyecto a no ser que se dé alguna circunstancia excepcional.	Baja

Tabla 2.55: Valoración de la probablidad

Recurso en Plazo / Esfuerzo / Coste		Impacto
$\geq 40\,\%$	Ya no se puede seguir con este TFG, hay que desecharlo completamente.	Catastrófico
Entre 20 % y 40 %	Hay que atrasar la entrega del TFG una convocatoria.	Alto
Entre 10 % y 20 %	Supone en torno a una semana adicional de trabajo.	Medio
$\leq 10\,\%$	Supone menos de media semana adicional de trabajo excepcional.	Bajo

Tabla 2.56: Valoración del impacto

Exposición		Probabilidad		
		Alta	**Media**	**Baja**
Impacto	**Catastrófico**	Alta	Alta	Alta
	Alto	Alta	Alta	Media
	Medio	Alta	Media	Baja
	Bajo	Media	Baja	Baja

Tabla 2.57: Cálculo de la exposición del riesgo a partir del producto de la probablidad y el impacto

Estrategias de planificación de riesgos

Una vez identificados y analizados los riesgos, para cada uno se determinó la mejor forma de planificación a partir de una de las siguientes estrategias:

Evitar: Toma de medidas que anulen la aparición o progresión de un riesgo. Este tipo de estrategias se encontrarán catalogadas dentro del apartado de "Acciones de prevención".

Transferir: Hacer que el riesgo sea gestionado por un tercero, como una empresa subcontratada. Este tipo de estrategias se encontrarán catalogadas dentro del apartado de "Acciones de prevención".

Mitigar: Se trata de planificar acciones que contengan el riesgo y estén orientadas a reducir el impacto y/o la probabilidad del mismo, antes de que éste ocurra. Este tipo de estrategias se encontrarán catalogadas dentro del apartado de "Acciones de prevención".

Contingencia: Realización de acciones de corrección que han sido previamente definidas y que son ejecutadas en respuesta a un riesgo que se ha dado. Este tipo de estrategias se encontrarán catalogadas dentro del apartado de "Acciones de corrección".

Aceptar: Asunción de que un riesgo ocurra. Por claridad se cubre en el apartado de "Acciones de prevención".

Categorías de riesgo

Debido a las distintas fuentes que riesgos de este proyecto, se define una estructura de desglose de riesgos (EDR) que permite ver las distintas categorías posibles para los riesgos del mismo.

Así, la estructura de desglose de riesgos para este proyecto se recoge en la Figura 2.8

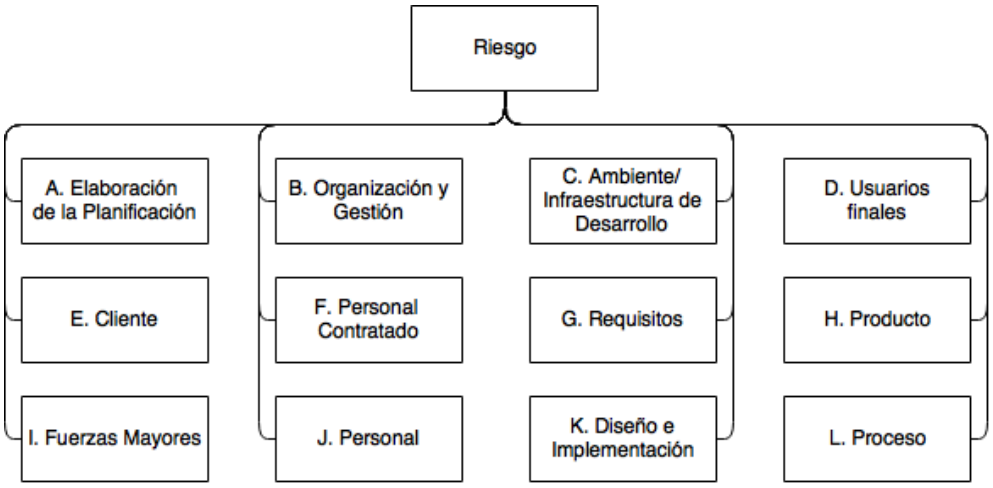


Figura 2.8: Estructura de desglose de riesgos

Número de riesgos a manejar en el proyecto

El establecimiento de un número máximo a riesgos a manejar se hace necesario, ya que así se realizará un análisis exigente de todos y cada uno de los riesgos considerados. Se descartarán aquellos riesgos con una menor probabilidad de ocurrencia o que tengan un impacto nimio en el proyecto, y se proporcionarán soluciones concretas para todos los riesgos que sean de relevancia.

Por ello, en este proceso se ha determinado que se tratarán, durante la ejecución del proyecto, **no más de 15 riesgos**, los cuales se corresponderán con aquellos de mayor exposición de entre los identificados. No obstante, para una correcta priorización, será necesario antes identificar todos los riesgos relevantes aplicables al proyecto.

Identificación y definiciones de riesgos

A continuación, se presentan unas fichas de descripción para los diferentes riesgos identificados.

RSG.01 - C	**La curva de aprendizaje para las tecnologías de desarrollo es más larga de lo esperado**	
Descripción	El tiempo necesario para formar al alumno en el uso de las herramientas seleccionadas para el desarrollo del proyecto es mayor al estimado en la planificación, debido a que su curva de aprendizaje es diferente a la esperada.	
Consecuencias	Este riesgo puede dar lugar a que el alumno no use correctamente las herramientas y/o a que se requiera la necesidad de alargar las tareas de formación.	
Probabilidad	**Impacto**	**Exposición**
Alta	Alto	Alta

RSG.02 - H	**El desarrollo de funciones software erróneas requiere volver a diseñarlas y a implementarlas**	
Descripción	Si algunas funcionalidades del producto son diseñadas o implementadas de forma errónea, será necesario llevar a cabo un retrabajo para que estas cumplan correctamente con los requisitos asociados. Si este problema no es detectado a tiempo y el software es diseñado sin el suficiente desacople y modularidad, muy probablemente habrá que realizar retrabajo en cascada que afecte a más módulos.	
Consecuencias	Se alargará el proyecto y aumentarán los costes del mismo, provocando una pérdida de motivación en el alumno. Además, muy probablemente, el rediseño y reimplementación serán llevado a cabo de forma apresurada, haciendo que la calidad del software resultante sea menor.	
Probabilidad	**Impacto**	**Exposición**
Media	Alto	Alta

RSG.03 - K	**Un mal diseño implica volver a diseñar e implementar**	
Descripción	El ciclo de vida y metodología escogidos en la planificación hacen que el éxito del proyecto se base, fundamentalmente, en el diseño inicial del producto. Si este diseño es erróneo, muy probablemente sea necesario reimplementar el proyecto casi al completo y rediseñar los diversos incrementos.	
Consecuencias	Se alargará la calendarización del proyecto y aumentarán sus costes, en mayor o menor medida dependiendo de la fase de diseño en la que se haya detectado el error. Muy probablemente, también se desmoralice al alumno debido a que tendrá que llevar a cabo retrabajo de forma apresurada, disminuyendo la calidad del software.	
Probabilidad	**Impacto**	**Exposición**
Media	Alto	Alta

RSG.04 - A	**Planificación optimista, "mejor caso" (en lugar de realista, "caso esperado")**	
Descripción	Al realizar la planificación del proyecto, esta es demasiado optimista, asumiendo que no existirá ningún problema de ejecución del plan o que no existirán vueltas a atrás en la planificación. Es decir, ausencia de colchones.	
Consecuencias	Existirán retrasos en las distintas entregas del proyecto, puesto que está garantizado que el plan del proyecto no se ajustará a la realidad.	
Probabilidad	**Impacto**	**Exposición**
Media	Alto	Alta

RSG.05 - B	**Pérdida de datos del proyecto**	
Descripción	Debido a una utilización ineficiente de la gestión de la configuración o a otros factores, se produce una pérdida de documentos esenciales del proyecto.	
Consecuencias	Se alargará la planifiación del proyecto, retrasando la entrega final cuanto sea necesario para recrear los datos del proyecto necesarios para su correcta consecución.	
Probabilidad	**Impacto**	**Exposición**
Media	Alto	Alta

RSG.06 - H	**El desarrollo de funciones software innecesarias alarga la planificación.**	
Descripción	Este riesgo es el caso de la implementación de requisitos estimulantes que se malinterpretan como requisitos vitales para el cliente. Puesto que el cliente no realizará esta distinción de requisitos, la sobreimplementación de funciones innecesarias llevará a alargar la planificación, realizando tareas que no serán valoradas de forma excesivamente positiva y, por tanto, retrasando el plan del proyecto.	
Consecuencias	El proyecto sufre retrasos en sus entregas debido a que se implementan requisitos que no son estrictamente necesarios para resolver los objetivos de negocio del cliente.	
Probabilidad	**Impacto**	**Exposición**
Media	Alto	Alta

RSG.07 - K	**Los componentes desarrollados por separado no se pueden integrar de forma sencilla, teniendo que volver a diseñar y repetir algunos trabajos.**	
Descripción	En alguna de las pruebas de integración que existen en la planificación, se pueden detectar errores que impiden la correcta integración del elemento desarrollado en el producto final.	
Consecuencias	Es necesario volver a la etapa de desarrollo del incremento concreto, haciendo también necesaria una reimplementación. El tiempo dedicado a esto aumenta el tiempo necesario para finalizar el proyecto.	
Probabilidad	**Impacto**	**Exposición**
Media	Alto	Alta

RSG.08 - H	**Los vídeos tipo escogidos para las pruebas no son verdaderamente representativos**	
Descripción	Uno de los criterios de validación consiste en que el producto funcione correctamente para los vídeos tipo. El vídeo tipo debe poseer características comunes a la mayoría de vídeos a los que se enfrentará el *tracker* en un entorno en producción.	
Consecuencias	Si un vídeo tipo no posee características comunes a la mayoría de vídeos en los que se va a utilizar el producto, es posible que el software se acepte pero que termine por no funcionar correctamente en la mayor parte de los escenarios.	
Probabilidad	**Impacto**	**Exposición**
Media	Alto	Alta

RSG.09 - H	**La aplicación no es capaz de funcionar a tiempo real**	
Descripción	A la hora de realizar inferencias o durante la evaluación mediante *benchmark*, la aplicación no es capaz de procesar fotogramas a tiempo real (al menos 30 fps).	
Consecuencias	Se obtendrá un producto aparentemente correcto, pero con un ámbito de aplicación reducido, por lo que probablemente no resulte de utilidad en una aplicación real.	
Probabilidad	**Impacto**	**Exposición**
Media	Alto	Alta

RSG.10 - G	**Se añaden requisitos extra**	
Descripción	En un primer momento del proyecto, se establecen los requisitos que se deben cumplir. Durante el transcurso del mismo, dichos requisitos se ven afectados de tal manera que se añaden más.	
Consecuencias	Debido a los cambios que sufre el listado de requisitos, es posible que sean necesarios cambios en el diseño y la implementación de varios componentes, para cumplir con las pruebas de verificación y validación. Debido a esto, puede darse un atraso en el proyecto. El cambio en los requisitos del producto puede resultar en el degradado del mismo, disminuyendo su calidad.	
Probabilidad	**Impacto**	**Exposición**
Alta	Medio	Alta

RSG.11 - C	**Ausencia de la versión correcta del software externo para la construcción del producto**	
Descripción	Al depender el producto de software de terceros para su correcta construcción y funcionamiento, resulta imperativo que la versión correcta (o versiones compatibles) de dichas librerías se encuentren disponibles para su uso durante todo el ciclo de vida del proyecto. En el caso de que alguna de esas librerías no se encuentre disponible, no será posible continuar con el desarrollo.	
Consecuencias	Al no poseer las librerías necesarias, tanto las versiones actuales del producto como las ya marcadas como tags podrían perder totalmente su funcionalidad o presentar un comportamiento inconsistente. Esta última consecuencia es especialmente peligrosa ya que, de no ser detectado el origen de la falla, podría pensarse que es un problema originado en el propio código desarrollado.	
Probabilidad	**Impacto**	**Exposición**
Alta	Medio	Alta

RSG.12 - C	**El artículo utilizado como referencia omite aspectos importantes**	
Descripción	Como suele ser común a la hora de presentar un artículo novedoso en un ámbito de alta competitividad, es muy probable que los autores Bertinetto y Valmadre hayan omitido valiosos detalles de implementación. Dichos detalles de implementación no resultan útiles para comprender los fundamentos de su modelo, pero sí resultan esenciales para poder replicar sus resultados.	
Consecuencias	Se obtendrá un producto aparentemente correcto, pero con un ámbito de aplicación reducido, por lo que probablemente no resulte de utilidad en una aplicación real.	
Probabilidad	**Impacto**	**Exposición**
Alta	Medio	Alta

RSG.13 - K	**El origen de la falta de rendimiento o los errores es desconocido**	
Descripción	A la hora de probar el sistema de inferencia o durante la validación, se detectan errores o una falta de rendimiento, de los cuales, dada la naturaleza del aprendizaje automático, no resulta posible encontrar la causa.	
Consecuencias	Se obtendrá un producto cuya funcionalidad se encuentre limitada, o que incluso no posea ningún tipo de utilidad. Si se dedica tiempo a hallar el origen de los errores, con un alto grado de probabilidad, no se llegará a ninguna conclusión de valor.	
Probabilidad	**Impacto**	**Exposición**
Alta	Medio	Alta

RSG.14 - I	**Acceso no autorizado al repositorio por parte de un tercero**	
Descripción	Durante el desarrollo del proyecto, un tercero no autorizado logra acceder al repositorio en el que se almacena la totalidad del TFG, pudiendo robar datos o incluso modificar información de forma arbitraria.	
Consecuencias	Un acceso no autorizado al repositorio podría dar lugar a una pérdida de los datos o a una alteración indebida de los mismos. También podría producirse un plagio del proyecto por parte de otra persona, ocasionando el suspenso inmediato del TFG tras llevar a cabo su entrega y presentación.	
Probabilidad	**Impacto**	**Exposición**
Baja	Catastrófico	Alta

RSG.15 - E	**El cliente no acepta el software entregado, incluso aunque cumpla todas sus especificaciones**	
Descripción	En el caso de que el cliente no tenga claros los requisitos que debería tener el producto que solitita, podría darse el caso de que, a la hora de hacerle la entrega del mismo, no se sienta satisfecho y no lo acepte. Si no existe un documento firmado en el que se hagan explícitas las funcionalidades a implementar, es posible que el cliente eche en falta requisitos vitales que en ningún momento mencionó.	
Consecuencias	Si el cliente no entra en razón, sería necesario iniciar un proceso recurriendo a las autoridades de la USC, con la finalidad de demostrar que el producto desarrollado es válido y que la no aceptación de los tutores es infundada. Esto muy probablemente haría que se perdiera cualquier posible relación futura con el cliente. En el peor de los casos, la resolución podría fallar a favor del cliente, dando lugar a que la entrega del TFG tuviera que ser pospuesta, o incluso invalidándolo.	
Probabilidad	**Impacto**	**Exposición**
Baja	Alto	Media

RSG.16 - A	**No se puede construir un producto de tal envergadura en el tiempo asignado**	
Descripción	Los requisitos de aceptación del producto no se pueden satisfacer en el tiempo asignado, debido a cualquier tipo de circunstancias no consideradas en la planificación.	
Consecuencias	Existirán retrasos en las distintas entregas del producto, puesto que el alumno no se podrá ceñir al tiempo asignado para las mismas. De esta forma, el cliente no estará satisfecho con el desarrollo del proyecto.	
Probabilidad	**Impacto**	**Exposición**
Baja	Alto	Media

RSG.17 - B	**Los planes del proyecto se abandonan por la presión, llevando al caos y a un desarrollo ineficiente**	
Descripción	Debido a la presión a la que somete al alumno el plan del proyecto, el proyecto cae en una espiral de caos en la que el producto posee una calidad menguada respecto a un desarrollo sin presión.	
Consecuencias	Los planes del proyecto son abandonados o el desarrollo de los mismos es caótico o lento.	
Probabilidad	**Impacto**	**Exposición**
Media	Medio	Media

RSG.18 - L	**La falta de un seguimiento exacto del progreso hace que se desconozca que el proyecto esté retrasado hasta que está muy avanzado**	
Descripción	Una mala gestión del proyecto, en la que no se van monitorizando de forma precisa las tareas realizadas y su progreso, conlleva al desconocimiento del estado actual del proyecto.	
Consecuencias	Desconocimiento del estado real del proyecto, descoordinación entre los miembros del equipo, y retrasos en la realización de las tareas.	
Probabilidad	**Impacto**	**Exposición**
Baja	Alto	Media

RSG.19 - I	**Indisponibilidad del servidor GPGPU**	
Descripción	A la hora de ir a utilizar el servidor GPGPU, este ya se encuentra ejecutando un proceso cuyo tiempo estimado hasta la finalización es considerable. Por lo tanto, no resulta posible ejecutar ningún nuevo proceso antes del fin de la presente jornada de trabajo.	
Consecuencias	Habrá que esperar una cantidad de tiempo importante e indeterminada, durante la cual no será posible entrenar el sistema, curar una base de datos ni llevar a cabo pruebas de rendimiento.	
Probabilidad	**Impacto**	**Exposición**
Alta	Bajo	Media

RSG.20 - G	**Las partes del proyecto que no se han especificado claramente consumen más tiempo del esperado**	
Descripción	Debido a la especificación ambigua de las funcionalidades de un módulo o componente del producto, es necesario llevar a cabo fases del proyecto ya realizadas o, por otra parte, debido a afirmaciones sobre el producto que no son consistentes, el desarrollo requiere de un tiempo mucho mayor que el necesario en circunstancias normales.	
Consecuencias	El proyecto sufre retrasos en sus entregas debido a que se requiere más tiempo del esperado.	
Probabilidad	**Impacto**	**Exposición**
Media	Medio	Media

RSG.21 - G	**Los requisitos no se han definido correctamente y su redefinición aumenta el ámbito del proyecto.**	
Descripción	Es bastante común que el cliente no exprese al completo lo que espera del producto a entregar. También es común que en el análisis se cometa algún error, malinterprete al cliente o que incluso se recojan mal los requisitos debido a la ambigüedad del lenguaje natural.	
Consecuencias	Es necesaria la redefinición de requisitos, suponiendo cambios en el diseño y la implementación del producto a entregar. El atraso en el proyecto se ve aumentado, la calidad del producto se ve negativamente afectada.	
Probabilidad	**Impacto**	**Exposición**
Media	Medio	Media

RSG.22 - L	**El exceso de rigor (aferramiento burocrático a las políticas y estándares de software) lleva a gastar más tiempo en gestión del necesario.**	
Descripción	Existen estándares software y políticas de rigor que, de ser seguidas en algunos casos, pueden aumentar la calidad del software y traer beneficios considerables al proyecto. De todas formas, si se aplican en el proyecto equivocado pueden tener consecuencias contrarias a las esperadas.	
Consecuencias	Se producen atrasos innecesarios al seguir estándares que implican un seguimiento excesivo de la burocracia.	
Probabilidad	**Impacto**	**Exposición**
Media	Medio	Media

RSG.23 - H	**Una calidad no aceptable requiere de un trabajo de comprobación, diseño e implementación superior al esperado**	
Descripción	Se realiza una construcción del producto con una calidad no acorde a la esperada, no siguiendo patrones de diseño ni buenas prácticas de programación.	
Consecuencias	Aumento del tiempo de desarrollo del producto, aumento en el tiempo requerido para la creación de las pruebas sobre el software, reducción de la productividad.	
Probabilidad	**Impacto**	**Exposición**
Media	Medio	Media

RSG.24 - B	**Establecimiento incorrecto de los elementos de configuración**
Descripción	Durante la creación del plan de gestión de la configuración, los elementos de configuración establecidos son incorrectos, en tanto en cuanto faltan componentes esenciales o han sido incluidos elementos superfluos.
Consecuencias	Un establecimiento incorrecto de los elementos de configuración podría dar lugar a que no se siguiera el proceso de control del cambio para todos los elementos necesarios, provocando inconsistencias, falta de seguimiento y permitiendo la modificación arbitraria de elementos críticos para la integridad del proyecto.

RSG.25 - B	**Establecimiento incorrecto de las líneas base**
Descripción	Durante la creación del plan de gestión de la configuración, las líneas base establecidas son incorrectas, en tanto en cuanto contienen elementos de configuración innecesarios o carecen de los esenciales. También podría darse el caso de que las fechas ligadas a dichas líneas base resultaran incorrectas.
Consecuencias	Un incorrecto establecimiento de las líneas base podría dar lugar a que alguna de las entregas del proyecto contuviera elementos de la configuración obsoletos o inconsistentes entre sí, o que las fechas escogidas fueran incorrectas o imposibles de cumplir. Estas entregas se considerarían incorrectas o fuera de plazo, pudiendo invalidar la presente versión del proyecto.

RSG.26 - L	**La gestión de riesgos del proyecto software consume más tiempo del esperado**	
Descripción	Durante la planificación inicial del proyecto, pueden no conocerse aún las implicaciones de muchos elementos que afectan al mismo, por lo que la gestión de riesgos puede llevar más tiempo de lo estimado en dicha planificación.	
Consecuencias	Todos los incrementos se verán atrasados respecto a lo planificado, debido a que se revisan los riesgos antes del comienzo de cada uno. La gestión de riesgos atrasa en general al desarrollo previsto del proyecto.	
Probabilidad	**Impacto**	**Exposición**
Media	Medio	Media

RSG.27 - E	**Indisponibilidad de los tutores de proyecto para proporcionar realimentación al alumno**	
Descripción	Dada la ausencia de los tutores de proyecto o la imposibilidad de acordar un calendario de reuniones, el alumno no puede plantear sus dudas e inquietudes ni recibir la opinión de sus tutores con respecto al avance del proyecto.	
Consecuencias	No resulta posible validar ningún requisito ni conocer la opinión de los tutores del proyecto relativa al progreso general del trabajo. También podría producirse una situación de bloqueo, en la que el alumno no puede continuar trabajando sin algún dato clave que únicamente sus tutores le pueden proporcionar	
Probabilidad	**Impacto**	**Exposición**
Media	Medio	Media

RSG.28 - J	**Manejo incorrecto de la herramienta de control de versiones**	
Descripción	A la hora de utilizar la herramienta de control de versiones (Git), el alumno no es capaz de realizar tareas cotidianas con fluidez y seguridad, poniendo en peligro el trabajo realizado.	
Consecuencias	Un manejo incorrecto de la herramienta de control de versiones podría dar lugar a que los nuevos cambios no fueran guardados, se sobrescribiera información, causando un estado inconsistente, o incluso se llegaran a perder todos los datos relativos a los elementos de configuración.	
Probabilidad	**Impacto**	**Exposición**
Baja	Medio	Baja

RSG.29 - L	**Mala definición del proceso de control del cambio**	
Descripción	El proceso de control del cambio se encuentra definido de forma incorrecta. Ya sea porque resulta demasiado pesado o superfluo para el presente proyecto, o porque no contempla algunas vicisitudes necesarias.	
Consecuencias	Una mala definición del proceso de control del cambio podría dar lugar a que resulte imposible de seguir, debido a la burocracia y redundancia necesarias al incluir demasiadas actividades, o a que las diversas decisiones tomadas conduzcan al proceso por un camino erróneo que proporcione un resultado contrario al esperado.	
Probabilidad	**Impacto**	**Exposición**
Baja	Medio	Baja

RSG.30- I	**La documentación de las herramientas utilizadas es escasa o errónea**	
Descripción	Las herramientas y librerías de terceros utilizadas carecen de una documentación actualizada y de calidad, en la cual aparezcan todas las posibilidades que ofrecen y su uso.	
Consecuencias	Algunas funcionalidades de las herramientas se hacen desconocidas para el alumno, el cual incluso puede llegar a creer que su producto posee un comportamiento erróneo al desconocer el funcionamiento exacto de algunas de las librerías utilizadas.	
Probabilidad	**Impacto**	**Exposición**
Baja	Medio	Baja

RSG.31 - J	**Desincronización con el repositorio**	
Descripción	De forma inconsciente, el alumno reanuda su sesión de trabajo, utilizando como base elementos obsoletos, los cuales ya no se encuentran vigentes en el presente proyecto.	
Consecuencias	Una desincronización entre el repositorio y el espacio de trabajo podría dar lugar a que los cambios y avances realizados no fueran correctamente guardados, impidiendo el trabajo desde otros ordenadores y entornos, y facilitando la pérdida de versiones e información relativa a los elementos de configuración más recientes. También podría darse el caso de que los cambios sean realizados sobre una versión antigua, de forma no intencionada.	
Probabilidad	**Impacto**	**Exposición**
Media	Bajo	Baja

2.5.2. Planificación de riesgos

Acciones planificadas

Para cada riesgo han sido planificadas una o varias acciones de corrección y/o prevención y se han definido una serie de indicadores que permitirán detectar de forma temprana la materialización de los mismos. Dicha información se encuentra registrada en las fichas que se muestran a continuación:

RSG.01 - C	**La curva de aprendizaje para las tecnologías de desarrollo es más larga de lo esperado**
Indicador	Se producen retrasos de más del 25 % del tiempo en la duración estimada de la formación.
Prevención	**Mitigar**: Ampliar el tiempo planificado de formación.
Corrección	**Mitigar**: Reducir el alcance del proyecto, utilizar alguna herramienta simplificada como Keras o dedicar horas extra.

RSG.02 - H	**El desarrollo de funciones software erróneas requiere volver a diseñarlas y a implementarlas**
Indicador	Un módulo en desarrollo no satisface los requisitos que debiera cumplir o no pasa las pruebas establecidas.
Prevención	**Mitigar**: Utilizar patrones de diseño que faciliten la reimplementación y rediseño.
Corrección	**Mitigar**: Dedicar horas extra al proyecto.

RSG.03 - K	**Un mal diseño implica volver a diseñar e implementar**
Indicador	Durante la implementación, se detecta la imposibilidad de replicar el diseño. O se detectan fallos en la integración de varios elementos del proyecto, debidos a un mal diseño de las interfaces de los mismos.
Prevención	**Mitigar**: Diseñar utilizando interfaces y patrones de diseño que faciliten el rediseño y la reimplementación.
Corrección	**Mitigar**: Dedicar horas extra al proyecto.

RSG.04 - A	**Planificación optimista, "mejor caso" (en lugar de realista, "caso esperado")**
Indicador	Retrasos en los hitos de más de un 15 % del tiempo estimado.
Prevención	**Evitar**: Utilizar colchones en la planificación suficientemente grandes.
Corrección	**Mitigar**: Reducir el alcance del proyecto.

RSG.05 - B	**Pérdida de datos del proyecto**
Indicador	Existe más de 1 hora de trabajo que deberá ser repetida exactamente igual que como ya fue realizada, por causas ajenas a un mal diseño o implementación.
Prevención	**Mitigar**: Realizar copias de seguridad frecuentes y en lugares separados del entorno del trabajo.
Corrección	**Mitigar**: Dedicar horas extra al proyecto o reducir el alcance del mismo.

RSG.06 - H	**El desarrollo de funciones software innecesarias alarga la planificación.**
Indicador	Se encuentra en desarrollo, a lo menos, un módulo relacionado con ningún requisito, o con requisitos únicamente estimulantes, sin aún haber finalizado los requisitos vitales e importantes del presente incremento.
Prevención	**Evitar**: hacer que las funciones a implementar durante las primeras fases del proyecto no provengan de requisitos estimulantes.
Corrección	**Mitigar**: Replanificar el proyecto.

RSG.07 - K	**Los componentes desarrollados por separado no se pueden integrar de forma sencilla, teniendo que volver a diseñar y repetir algunos trabajos.**
Indicador	La integración de una serie de elementos requiere de más de 2 horas de trabajo, o no se satisfacen las pruebas de integración definidas.
Prevención	**Mitigar**: diseñar el sistema haciendo uso de interfaces lo más desacopladas posibles.
Corrección	**Mitigar**: Dedicar horas extra al proyecto y revisar el diseño de los componentes afectados.

RSG.08 - H	**Los vídeos tipo escogidos para las pruebas no son verdaderamente representativos**
Indicador	Existe un empeoramiento de más de un 20 % al realizar las pruebas de rendimiento con vídeos propios.
Prevención	**Mitigar**: Consultar a un experto en el ámbito de la aplicación antes de seleccionar los vídeos tipo.

RSG.09 - H	**La aplicación no es capaz de funcionar a tiempo real**
Indicador	Los fotogramas por segundo procesados por el sistema son inferiores a 30.
Corrección	**Mitigar**: Consultar la guía de eficiencia y rendimiento de TensorFlow para aplicar las optimizaciones sugeridas.

RSG.10 - G	**Se añaden requisitos extra**
Indicador	El número de requisitos vitales e importantes es mayor que el que se definió al comienzo del proyecto.
Prevención	**Evitar**: Fijar los requisitos a implementar desde el principio del proyecto.

RSG.11 - C	**Ausencia de la versión correcta del software externo para la construcción del producto**
Indicador	Durante el desarrollo del producto, una actualización de alguna de las librerías del sistema da lugar a que se pierda funcionalidad.
Prevención	**Evitar**: Almcenar en un entorno virtual las versiones concretas de las librerías utilizadas durante el desarrollo, y establecerlas como elementos de configuración.
Corrección	**Mitigar**: Tratar de recurrir a la página oficial de la herramienta o a Internet Archive para buscar las versiones pasadas del software necesitado.

RSG.12 - C	**El artículo utilizado como referencia omite aspectos importantes**
Indicador	Los resultados de las pruebas de rendimiento realizadas son al menos un 10 % peores que los publicados en el artículo original, pese a replicar todo lo que en este se describe.
Prevención	**Mitigar**: Recurrir a otras fuentes de información, tales como congresos, repositorios de código fuente y otras publicaciones.

RSG.13 - K	**El origen de la falta de rendimiento o los errores es desconocido**
Indicador	Los resultados de las pruebas de rendimiento realizadas son al menos un 5 % peores que los publicados en el artículo original, y ha pasado más de una semana de desarrollo sin obtener mejora.
Corrección	**Mitigar**: Hacer uso del paquete pymatlab para establecer una interfaz entre MATLAB y Python, y así poder hacer llamadas a las funciones creadas por Luca Bertinetto y Jack Valmadre para comprobar que resultan equivalentes a los módulos desarrollados.

RSG.14 - I	**Acceso no autorizado al repositorio por parte de un tercero**
Indicador	Existe al menos un inicio de sesión en GitHub desde un dispositivo no reconocido.
Prevención	**Mitigar**: Establecer una autenticación en dos pasos para poder acceder a la cuenta y repositorios de GitHub.
Corrección	**Mitigar**: Notificar a la comisión de TFGs y a los tutores que alguien ha tenido acceso a datos privilegiados del proyecto.

RSG.15 - E	**El cliente no acepta el software entregado, incluso aunque cumpla todas sus especificaciones**
Indicador	Durante la validación, el cliente hace explícito su desacuerdo respecto al producto presentado. El producto presentado cumple con todas las especificaciones acordadas previamente.
Prevención	**Mitigar**: Realizar reuniones periódicas con el cliente para validar los requisitos.
Corrección	**Mitigar**: Recurrir al anteproyecto firmado por el cliente para validar el producto.

RSG.16 - A	**No se puede construir un producto de tal envergadura en el tiempo asignado**
Indicador	Existe un retraso de más de un 25 % con respecto a la planificación.
Corrección	**Mitigar**: Limitar el alcance del proyecto.

RSG.17 - B	**Los planes del proyecto se abandonan por la presión, llevando al caos y a un desarrollo ineficiente**
Indicador	Imposibilidad de determinar el estado y situación actual del proyecto y/o carencia o nulo seguimiento de procedimientos burocráticos a la hora de realizar actividades.
Prevención	**Mitigar**: Hacer planificaciones flexibles y con metodologías poco pesadas de ejecutar.

RSG.18 - L	**La falta de un seguimiento exacto del progreso hace que se desconozca que el proyecto esté retrasado hasta que está muy avanzado**
Indicador	Resulta imposible determinar los requisitos alcanzados en un punto determinado del desarrollo.
Prevención	**Mitigar**: Mantener reuniones periódicas con los tutores del proyecto en las que se muestren resultados tangibles, pese a que aún no se haya alcanzado el hito relacionado.

RSG.19 - I	**Indisponibilidad del servidor GPGPU**
Indicador	Al consultar los trabajos de SLURM con la herramienta smap, existe un trabajo en ejecución cuyo tiempo de cómputo es superior a 6 horas.
Corrección	**Mitigar**: Si el proceso en ejecución no utiliza todas las GPUs, lanzar el proceso propio sin utilizar el sistema de colas. En caso de que esto no sea posible, poner en cola el proceso propio y mientras dedicar el tiempo a otros módulos independientes o a la documentación.

RSG.20 - G	**Las partes del proyecto que no se han especificado claramente consumen más tiempo del esperado**
Indicador	Una tarea lleva más del 110 % del tiempo estimado para la misma. Al investigar las razones del atraso se detectan ambigüedades en la especificación.
Corrección	**Mitigar**: Mantener una nueva reunión con el cliente, para asegurar que se definen los requisitos faltantes, y que estos son validables y no ambiguos.

RSG.21 - G	**Los requisitos no se han definido correctamente y su redefinición aumenta el ámbito del proyecto.**
Indicador	Se detectan una o más contradicciones entre algunos de los requisitos. También se puede dar el caso en el cliente haga explícito su desacuerdo con los requisitos presentados en el acta del proyecto, o que los requisitos definidos no puedan ser validados formalmente.
Prevención	**Mitigar**: Dedicar tiempo adicional al análisis de requisitos.
Corrección	**Mitigar**: Reducir el alcance del proyecto o dedicar horas extra.

RSG.22 - L	**El exceso de rigor (aferramiento burocrático a las políticas y estándares de software) lleva a gastar más tiempo en gestión del necesario.**
Indicador	La gestión del proyecto se extiende más de un 15 % debido al seguimiento de políticas y estándares.
Prevención	**Mitigar**: Hacer que las políticas y estándares seguidos sean flexibles y poco pesados.
Corrección	**Mitigar**: Dedicar horas extra al proyecto.

RSG.23 - H	**Una calidad no aceptable requiere de un trabajo de comprobación, diseño e implementación superior al esperado**
Indicador	Existen funciones con más de 150 líneas de código.
Prevención	**Mitigar**: Seguir los principios de Clean Code y seguir los consejos proporcionados por el IDE.
Corrección	**Mitigar**: Reescribir las funciones problemáticas siguiendo los principios de Clean Code y usando el Code Cleanup del IDE como base.

RSG.24 - B	**Establecimiento incorrecto de los elementos de configuración**
Indicador	Uno de los elementos de configuración definidos puede ser modificado arbitrariamente sin afectar a la integridad del proyecto. O la eliminación de un componente no incluido entre los elementos de configuración provoca inestabilidad en el proyecto. O La normativa de la escuela incluye un elemento de configuración no establecido en el proyecto.
Prevención	**Mitigar**: Se definirán como elementos de configuración todos aquellos componentes que intervengan en la creación y gestión del proyecto o ayuden a la comprensión del TFG, por mínimos que sean. Se preferirá definir elementos de configuración en exceso que en defecto.

RSG.25 - B	**Establecimiento incorrecto de las líneas base**
Indicador	Detección de una fecha de entrega incorrecta. O alguna línea base contiene elementos de configuración que se encuentran obsoletos, sin justificación. O la entrega del TFG no cuenta con los elementos de configuración que la escuela considera necesarios. O la modificación de algún elemento de configuración no incluido en la línea base afecta al contenido de la memoria.
Prevención	**Mitigar**: Se identificarán los componentes obligatorios de un TFG revisando la normativa de la escuela, y se añadirán a las líneas base definidas junto a los elementos de la configuración que ayuden a su comprensión. Se consultarán TFGs de años pasados para tener una mejor perspectiva de qué es lo que se precisa y se prestará una especial atención a la calendarización y las fechas de entrega.

RSG.26 - L	**La gestión de riesgos del proyecto software consume más tiempo del esperado**
Indicador	La gestión de riesgos del proyecto software se excede en un 10 % más de lo planeado.
Prevención	**Mitigar**: Gestionar un número limitado pero suficiente de riesgos.
Corrección	**Mitigar**: Dedicar horas extra al proyecto o reducir el número de riesgos gestionados.

RSG.27 - E	**Indisponibilidad de los tutores de proyecto para proporcionar realimentación al alumno**
Indicador	Ha pasado más de una semana y media sin que el alumno pueda reunirse con al menos uno de sus tutores de proyecto.
Prevención	**Mitigar**: Establecer otros canales de comunicación no presencial, tales como correo electrónico o videoconferencia.

RSG.28 - J	**Manejo incorrecto de la herramienta de control de versiones**
Indicador	Necesidad de llevar a cabo una o más consultas en Internet por cada 3 interacciones con la herramienta. O sobrescritura de datos no deseada.
Prevención	**Mitigar**: Llevar a cabo algún tipo de curso formativo relativo a la herramienta de control de versiones empleada. Tras el curso, se procederá a realizar ejercicios de ejemplo relacionados con el control de versiones, con la finalidad de verificar que se posee la destreza necesaria para el desarrollo del proyecto.

RSG.29 - L	**Mala definición del proceso de control del cambio**
Indicador	La realización del proceso de control de un cambio requiere de más de 2 minutos de cobertura de plantillas. O se hace imposible continuar el proceso debido a que en la definición del mismo no se contempla una continuación posible al estado en el que se encuentra la ejecución actual.
Prevención	**Mitigar**: Minimizar las posibles ramificaciones del proceso y evitar los ciclos en el mismo. También será de especial interés que el proceso resultante sea breve y simple, encontrándose claramente descrito, haciendo uso de un lenguaje comprensible y sin lugar a ambigüedades.

RSG.30- I	**La documentación de las herramientas utilizadas es escasa o errónea**
Indicador	Existe al menos una función externa que no produce los resultados descritos por la documentación. O no resulta posible encontrar un documento que contenga todas las funciones disponibles de la librería, junto a sus argumentos y salidas.
Corrección	**Mitigar**: Consultar otras fuentes de información como foros, repositorios de código o blogs y videotutoriales.

RSG.31 - J	**Desincronización con el repositorio**
Indicador	La información local resultante tras una jornada no se encuentra incluida en el repositorio. O la actualización del entorno de trabajo local con información del repositorio provoca la pérdida de datos. O resulta imposible trabajar en la versión más actual del proyecto empleando otro ordenador.
Prevención	**Mitigar**: Llevar a cabo una actualización del repositorio con los nuevos cambios realizados tras cada sesión de trabajo, sin excepción. Además, al comienzo de cada sesión de trabajo, se descargarán los datos del repositorio relativos a la versión más actual con la que se desee trabajar.

2.5.3. Seguimiento y control de riesgos

Historial de seguimiento durante la planificación

Durante la planificación, se ejecutaron aquellas acciones de prevención que relativas a riesgos de exposición alta o media que sólo tuvieran sentido durante este período de tiempo y que no requirieran de más de 4 horas de trabajo adicional.

El resultado de aplicar dichas acciones es el mostrado a continuación, para cada riesgo afectado:

RSG.01 - C	**La curva de aprendizaje para las tecnologías de desarrollo es más larga de lo esperado**	Fecha: 14/02/2018
Acciones	**Mitigar**: Ampliar el tiempo planificado de formación.	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Media	Alto	Alta

RSG.02 - H	**El desarrollo de funciones software erróneas requiere volver a diseñarlas y a implementarlas**	Fecha: 26/02/2018
Acciones	**Mitigar**: Utilizar patrones de diseño que faciliten la reimplementación y rediseño	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Media	Medio	Media

RSG.03 - K	**Un mal diseño implica volver a diseñar e implementar**	Fecha: 26/02/2018
Acciones	**Mitigar**: Diseñar utilizando interfaces y patrones de diseño que faciliten el rediseño y la reimplementación	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Media	Medio	Media

RSG.04 - A	**Planificación optimista, "mejor caso" (en lugar de realista, "caso esperado")**	Fecha: 14/02/2018
Acciones	**Evitar**: Utilizar colchones en la planificación suficientemente grandes.	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Baja	Alto	Media

RSG.05 - B	**Pérdida de datos del proyecto**	Fecha: 28/02/2018
Acciones	**Mitigar**: Realizar copias de seguridad frecuentes y en lugares separados del entorno del trabajo	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Baja	Alto	Media

RSG.06 - H	**El desarrollo de funciones software innecesarias alarga la planificación.**	Fecha: 12/02/2018
Acciones	**Evitar**: Hacer que las funciones a implementar durante las primeras fases del proyecto no provengan de requisitos estimulantes	
Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Baja	Alto	Media

RSG.07 - K	**Los componentes desarrollados por separado no se pueden integrar de forma sencilla, teniendo que volver a diseñar y repetir algunos trabajos.**	Fecha: 22/02/2018
Acciones	**Mitigar**: Diseñar el sistema haciendo uso de interfaces lo más desacopladas posibles	
Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Baja	Medio	Baja

RSG.08 - H	**Los vídeos tipo escogidos para las pruebas no son verdaderamente representativos**	Fecha: 23/02/2018
Acciones	**Mitigar**: Consultar a un experto en el ámbito de la aplicación antes de seleccionar los vídeos tipo	
Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Baja	Alto	Media

RSG.10 - G	**Se añaden requisitos extra**	Fecha: 09/02/2018
Acciones	**Evitar**: Fijar los requisitos a implementar desde el principio del proyecto	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Nula	Medio	Nula

RSG.11 - C	**Ausencia de la versión correcta del software externo para la construcción del producto**	Fecha: 27/02/2018
Acciones	**Evitar**: Almcenar en un entorno virtual las versiones concretas de las librerías utilizadas durante el desarrollo, y establecerlas como elementos de configuración	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Media	Medio	Media

RSG.12 - C	**El artículo utilizado como referencia omite aspectos importantes**	Fecha: 23/02/2018
Acciones	**Mitigar**: Recurrir a otras fuentes de información, tales como congresos, repositorios de código fuente y otras publicaciones.	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Alta	Bajo	Media

RSG.14 - I	**Acceso no autorizado al repositorio por parte de un tercero**	Fecha: 12/02/2018
Acciones	**Mitigar**: Establecer una autenticación en dos pasos para poder acceder a la cuenta y repositorios de GitHub.	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Nula	Catastrófico	Nula

RSG.15 - E	**El cliente no acepta el software entregado, incluso aunque cumpla todas sus especificaciones**	Fecha: 09/02/2018
Acciones	**Mitigar**: Realizar reuniones periódicas con el cliente para validar los requisitos.	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Nula	Alto	Nula

RSG.17 - B	**Los planes del proyecto se abandonan por la presión, llevando al caos y a un desarrollo ineficiente**	Fecha: 15/02/2018
Acciones	**Mitigar**: Hacer planificaciones flexibles y con metodologías poco pesadas de ejecutar.	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Baja	Medio	Baja

RSG.18 - L	**La falta de un seguimiento exacto del progreso hace que se desconozca que el proyecto esté retrasado hasta que está muy avanzado**	Fecha: 09/02/2018
Acciones	**Mitigar**: Mantener reuniones periódicas con los tutores del proyecto en las que se muestren resultados tangibles, pese a que aún no se haya alcanzado el hito relacionado.	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Nula	Alto	Nula

RSG.21 - G	**Los requisitos no se han definido correctamente y su redefinición aumenta el ámbito del proyecto.**	Fecha: 14/02/2018
Acciones	**Mitigar**: Dedicar tiempo adicional al análisis de requisitos.	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Baja	Medio	Baja

RSG.22 - L	**El exceso de rigor (aferramiento burocrático a las políticas y estándares de software) lleva a gastar más tiempo en gestión del necesario.**	Fecha: 16/02/2018
Acciones	**Mitigar**: Hacer que las políticas y estándares seguidas sean flexibles y poco pesadas.	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Media	Bajo	Baja

RSG.23 - H	**Una calidad no aceptable requiere de un trabajo de comprobación, diseño e implementación superior al esperado**	Fecha: 28/02/2018
Acciones	**Mitigar**: Seguir los principios de Clean Code y seguir los consejos proporcionados por el IDE.	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Baja	Medio	Baja

RSG.24 - B	**Establecimiento incorrecto de los elementos de configuración**	Fecha: 16/02/2018
Acciones	**Mitigar**: Se definirán como elementos de configuración todos aquellos componentes que intervengan en la creación y gestión del proyecto o ayuden a la comprensión del TFG, por mínimos que sean. Se preferirá definir elementos de configuración en exceso que en defecto.	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Media	Bajo	Baja

RSG.25 - B	**Establecimiento incorrecto de las líneas base**	Fecha: 16/02/2018
Acciones	**Mitigar**: Se identificarán los componentes obligatorios de un TFG revisando la normativa de la escuela y se añadirán a las líneas base definidas junto a los elementos de la configuración que ayuden a su comprensión. Se consultarán TFGs de años pasados para tener una mejor perspectiva de qué es lo que se precisa y se prestará una especial atención a la calendarización y fechas de entregas.	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Media	Bajo	Baja

RSG.26 - L	**La gestión de riesgos del proyecto software consume más tiempo del esperado**	Fecha: 12/02/2018
Acciones	**Mitigar**: Gestionar un número limitado pero suficiente de riesgos.	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Baja	Medio	Baja

RSG.27 - E	**Indisponibilidad de los tutores de proyecto para proporcionar realimentación al alumno**	Fecha: 09/02/2018
Acciones	**Mitigar**: Establecer otros canales de comunicación no presencial, tales como correo electrónico o videoconferencia.	

Nueva Probabilidad	**Nueva Impacto**	**Nueva Exposición**
Baja	Medio	Baja

Historial de seguimiento durante la ejecución

Tras la realización de las medidas preventivas descritas en el apartado anterior, se limitó notablemente el número de riesgos peligrosos para el correcto desarrollo del proyecto. De esta forma, en el momento del comienzo de la ejecución, hubo que realizar el seguimiento y control de únicamente 3 riesgos de exposición alta y 11 riesgos de exposición media (un número que se encuentra dentro del límite de 15 riesgos definido con anterioridad).

Las incidencias acaecidas durante la ejecución del plan de proyecto, las acciones tomadas y el riesgo al que hacen referencia se encuentran recogidas a continuación:

RSG.12 - I	**El artículo utilizado como referencia omite aspectos importantes**	Fecha: 15/03/2018
Descripción	En el artículo original no se mencionaban algunos hiperparámetros relevantes. Además, existían algunas diferencias sustanciales entre éste y la versión del repositorio (el tamaño de los filtros de convolución era inconsistente).	
Acciones	Los hiperparámetros faltantes se extrajeron del código fuente del repositorio y, ante la aparición de inconsistencias, se primó la información contenida en el código fuente.	

RSG.30 - I	**La documentación de las herramientas utilizadas es escasa o errónea**	Fecha: 19/03/2018
Descripción	La normalización por lotes utilizada por defecto en TensorFlow es distinta de la de MatConvNet, por lo que las operaciones realizadas no resultaban equivalentes. Esto no aparecía reflejado en la documentación oficial.	
Acciones	Se recurrió a foros de Internet para averiguar la causa de la diferencia de funcionamiento y para obtener información acerca de cómo replicar una normalización por lotes de MatConvNet desde TensorFlow.	

RSG.07 - I	**Los componentes desarrollados por separado no de pueden integrar de forma sencilla, teniendo que volver a diseñar y repetir algunos trabajos**	Fecha: 11/04/2018
Descripción	Una incompatibilidad de tipos entre enteros con y sin signo dio lugar a que no fuera posible emplear el modelo entrenado para realizar inferencias.	
Acciones	Se ajustaron los sistemas de inferencia y entrenamiento para que hicieran uso de los mismos tipos de datos. Tras esto, se volvió a entrenar el modelo.	

RSG.30 - I	**La documentación de las herramientas utilizadas es escasa o errónea**	Fecha: 19/04/2018
Descripción	La documentación del OTB Toolkit era prácticamente nula y los resultados obtenidos por la red en dicho *benchmark* eran realmente adversos.	
Acciones	Se recurrió a foros de Internet para averiguar el modo de uso del *benchmark* y se analizó el código fuente del mismo para comprender los formatos de entrada que necesitaba.	

RSG.19 - I	**Indisponibilidad del servidor GPGPU**	Fecha: 10/04/2018 y 21/05/2018
Descripción	El servidor de computación se encontraba ejecutando una serie de trabajos con una duración estimada de 27 horas.	
Acciones	En las dos ocasiones en las que se materializó este riesgo, se optó por dejar en cola el proceso que se deseaba ejecutar, y mientras dedicar el tiempo a realizar parte de la documentación.	

RSG.11 - C	**Ausencia de la versión correcta del software externo para la construcción del producto**	Fecha: 02/05/2018
Descripción	A la hora de dockerizar el sistema, no era posible obtener los binarios necesarios de OpenCV directamente del repositorio.	
Acciones	Se realizó la instalación desde ficheros fuente de la versión necesaria de OpenCV, recurriendo al histórico de su repositorio.	

2.6. Gestión de la configuración

En la presente sección se detalla el plan de gestión de la configuración seguido durante el proyecto, establecido con la finalidad de garantizar la calidad e integridad del producto desarrollado.

2.6.1. Definición del sistema de gestión de la configuración

Para el almacenamiento de los elementos de configuración, se utilizó la plataforma online GitHub, la cual permite alojar proyectos utilizando el sistema de control de versiones Git. En ella se creó un repositorio para contener todos los elementos de configuración, incluyendo tanto la documentación y recursos del proyecto como

el propio código. Para el manejo de este repositorio, se usaron la interfaz web proporcionada y la herramienta Git por línea de comandos.

Al hacer uso de esta herramienta, no fue necesario incluir ningún tipo de indicativo de versión o registro de cambios en los propios documentos, ya que el propio Git se encarga de realizar un exhaustivo seguimiento de los mismos, y podrían darse inconsistencias.

2.6.2. Estructura del repositorio

El repositorio privado del alumno usado para el almacenamiento de todos los elementos relativos al proyecto sigue elesquema de directorios que se muestra a continuación en la Figura 2.9:

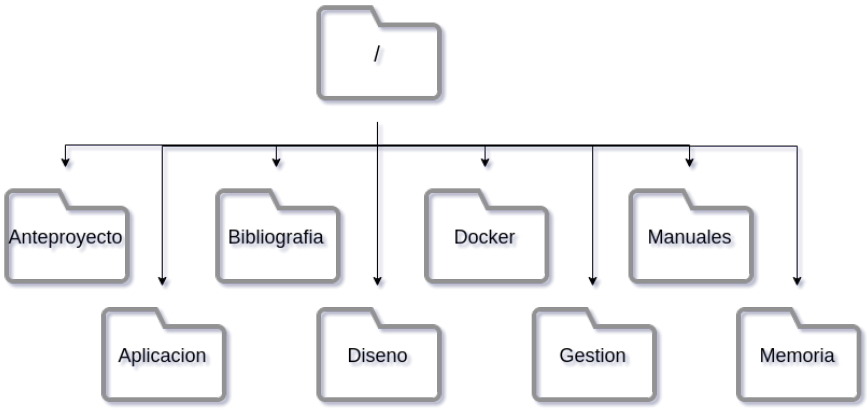


Figura 2.9: Estructura del repositorio

A continuación se pasará a describir cada uno de los elementos de la Figura 2.9:

Anteproyecto: Directorio en el cual se encuentra almacenado el anteproyecto de la presente memoria, así como los archivos necesarios para su creación.

Aplicacion: Directorio en el cual reside el código fuente del producto software desarrollado.

Bibliografia: Directorio en el que se encuentra almacenado el material bibliográfico consultado durante el desarrollo del presente Trabajo de Fin

de Grado.

Diseno: Directorio en el cual se encuentra recogido todo el material empleado para el diseño del software.

Docker: Directorio en el que se recoge el fichero `Dockerfile` y scripts de construcción necesarios para generar una imagen que contenga el producto dockerizado.

Gestion: Directorio en el que se recogen todos los documentos y diagramas de gestión desarrollados a lo largo del presente proyecto.

Manuales: Directorio en el que se almacenan los manuales de uso e instalación de la aplicación, escritos en Markdown para una mejor integración con el repositorio.

Memoria: Directorio en el que se encuentran todos los archivos necesarios para poder generar la memoria del proyecto.

2.6.3. Identificación de los elementos de configuración

Como mitigación al riesgo **RSG.24**, se han definido como elementos de configuración todos aquellos componentes que intervienen en la creación y gestión del proyecto o ayuden a la comprensión del TFG, por mínimos que sean. De esta forma, los elementos de los cuales se realizará gestión de la configuración son los siguientes:

- El archivo `.pdf` incluido en el directorio `Anteproyecto`, **sin** necesidad de gestionar los ficheros fuente empleados para su creación.
- Todos los ficheros incluidos en el directorio `Aplicacion`, **excepto** los archivos de compilación `.pyc` y los logs generados por la aplicación (almacenados en el subdirectorio `logs`). **Sí** serán incluidos como elementos de configuración los pesos entrenados de la red neuronal, almacenados en el subdirectorio `model`, así como el archivo `requirements.txt`, necesario para recrear un entorno virtual Python con las dependencias requeridas por la aplicación.

- Todos los archivos incluidos en el directorio `Bibliografia`.
- Todos los archivos incluidos en el directorio `Diseno`.
- Todos los archivos incluidos en el directorio `Docker`.
- Todos los archivos incluidos en el directorio `Gestion`.
- Todos los archivos incluidos en el directorio `Manuales`.
- Todos los archivos incluidos en el directorio `Memoria`, **excepto** los archivos de compilación LaTeX generados (`.aux`, `.bbl`, `.bcf`, `.blg`, `.lof`, `.log`, `.lol`, `.lot`, `.out`, `.run.xml`, `.synctex.gz` y `.toc`).
- Las plantillas en Markdown, para la generación de *issues* estandarizados.

De este modo, para facilitar la gestión del repositorio, todos los elementos que no estén en esta lista deberán ser incluidos en un `.gitignore`, puesto que no serán sometidos a GC.

2.6.4. Nomenclatura de los ficheros

Con el objetivo de estandarizar y facilitar la identificación de ficheros para su edición o lectura, se ha propuesto una estructura de nombre común para la mayor parte de los ficheros de proyecto. Los únicos archivos exentos de cumplir esta nomenclatura serán aquellos que se encuentren alojados en las carpetas `Aplicacion`, `Docker` y `Memoria` (a excepción del archivo `.pdf`).

De este modo, todos los archivos que sigan la nomenclatura definida deberán tener el siguiente nombre:

TFG_<Identificador>_<Nombre>.<Extensión>

Siendo cada campo lo siguiente:

TFG: Se utilizará para indicar que los elementos de configuración pertenecen al presente Trabajo de Fin de Grado.

Identificador: Código de tres letras mayúsculas que indicará la naturaleza del contenido del fichero, siguiendo la Tabla 2.58:

Nombre: Nombre identificativo compuesto por una cadena de caracteres alfanumérica que describa el contenido del fichero.

Extensión: Cadena de caracteres propia del formato del archivo, utilizada por el sistema operativo para decidir el procedimiento necesario para ejecutarlo o interpretarlo.

Identificadores de los elementos de configuración	
ID	Descripción
ANE	Hoja de cálculo automatizada para cálculo de costes y análisis económico
APY	Documento de solicitud de aprobación del anteproyecto
BIB	Elemento de la bibliografía
CDU	Diagrama de casos de uso del sistema
CRQ	Catálogo de requisitos que recoge todos los requisitos que dan soporte al sistema o software
DDC	Diagrama de contexto que define las interacciones del sistema con otras entidades externas
DFD	Diagrama de flujo de datos
EDG	Diagrama que describe una estructura de desglose
EDT	Mapa mental que refleja los distintos componentes de una estructura de descomposición del trabajo
MAN	Manual del sistema
MEM	Memoria del proyecto
MTT	Tabla que describe una matriz de trazabilidad
RIP	Registro de incidencias del proyecto
RSG	Registro de riesgos del proyecto
PLF	Diagrama de Gantt que describe la planificación del proyecto

Tabla 2.58: Identificadores de los elementos de configuración

2.6.5. Establecimiento de líneas base

Las líneas base se establecerán, al completar cada hito, mediante la creación de *releases* (también denominadas *tags*), las cuales recogen el estado del repositorio en un momento determinado del tiempo.

Estas *releases* incluirán todos los elementos de configuración definidos, y utilizarán un sistema de versionado simple incremental. No se podrá realizar una

release si existen *pull requests* pendientes (puesto que se trata trabajo casi terminado y pendiente de revisión que se debe incorporar lo antes posible). No obstante, es necesario aclarar que es posible realizar una entrega aún habiendo *issues* abiertos.

2.6.6. Definición del sistema de gestión de cambios

Para el registro de las peticiones de cambio, se utilizará la plataforma web GitHub. En ella, se almacenarán las peticiones de cambio como *issues*, las cuales serán debidamente clasificadas en función de su estado de aprobación.

Para acceder al registro de peticiones de cambio, únicamente se podrá utilizar la interfaz web de GitHub.

En cuanto al seguimiento de las peticiones de cambio, para solicitar una modificación o mejora, se deberá crear un *issue* (incidencia) en GitHub, de tal manera que aquellas modificaciones que requieran más de media hora de trabajo, o cuya fuente sean los tutores del proyecto, estarán ligados necesariamente a un *issue*.

Se distinguen tres estados posibles en los que puede estar un *issue* en función del progreso del cambio:

Aprobado: Corresponde a cambios que han sido aprobados y están ya implementados (*issue* cerrado) o en proceso de implementación (*issue* abierto).

Pendiente de aprobación: Corresponde a cambios que aún no han sido aprobados o rechazados por el grupo.

Rechazado: Corresponde a *issues* que han sido rechazados por el grupo. En este caso, los *issues* siempre tendrán que estar cerrados.

Por otra parte, y de manera opcional, si el cambio tuviera un *pull request* asociado, podrá añadirse una etiqueta denominada "**Cambio implementado**", cuyo objetivo es facilitar la gestión de los *issues*.

Todas aquellas proposiciones de cambio deberán ser registradas, para que quede constancia de las mismas.

Como base para la creación de propuestas de cambio, se utilizará una plantilla ligera que será integrada en los *issue*, la cual será presentada automáticamente para su cobertura al crear un *issue*. La plantilla de este proceso está escrita en Markdown para una presentación visual mejorada:

```
# ISSUE
- **Fuente**:
- **Descripción**:
- **Coste estimado**:
- **Observaciones**:
```

[1]

Los campos que también deberán encontrarse cubiertos obligatoriamente y que estarán presentes como una parte de la interfaz son los siguientes:

Label (para indicar el estado de un cambio): Es una etiqueta prefijada que se añade al *issue*. Dicha etiqueta puede tomar tres valores: "Pendiente de aprobación", "Aprobado" o "Rechazado". Además, aquellos cambios aprobados podrán contar con una etiqueta de "Cambio implementado".

Identificador: Es añadido automáticamente por GitHub. Es un número correlativo que empieza en 1.

Nombre del cambio: Debe añadirse en el título del *issue*.

A la hora de realizar un *commit* o proponer un *pull request*, será obligatorio referenciar aquellos *issues* que se solucionan (a no ser que el *commit* resuelva únicamente cambios menores).

2.6.7. Lecciones aprendidas

A continuación se describen las lecciones aprendidas durante el proyecto, que podrían dar lugar a un mejor proceso de gestión de la configuración:

- Una vez se acumulan *issues*, es complicado ver en qué parte del proyecto se ha realizado cada uno sin recurrir a su contenido (los títulos a veces no

[1] El campo de "Fuente", en caso de encontrarse vacío, indicará que se trata de un cambio propuesto por el alumno.

son muy descriptivos). Es necesario que los *issues* vayan asociados a hitos (*milestones*) (directamente relacionados con los incrementos).

- Resulta excesivamente tedioso definir *issues* para todos los cambios a realizar considerados mayores. Sería bueno replantear la definición de cambio mayor y establecer el límite que lo separa del cambio menor en una hora de trabajo, en vez de la media hora actualmente definida.
- En ocasiones, resulta difícil identificar a simple vista cual es la naturaleza o motivación de un cambio. Es por esto que sería bueno incorporar un conjunto mayor de *labels* que indiquen si un cambio es una corrección, una mejora o un requisito nuevo.

2.7. Análisis de los costes

En la presente sección se lleva a cabo un análisis de los costes del proyecto, en la cual se detallan los fondos requeridos y la justificación de cada importe.

Para la estimación de los costes, se ha realizado un desglose en las secciones que se muestran a continuación:

Gastos asociados al personal: Importe destinado a pagar a los miembros del equipo de proyecto.

Servicios contratados: Aquellos servicios, ya sean proporcionados por la empresa o por un tercero, necesarios para el desarrollo del proyecto.

Compras: Equipos, infraestructuras específicamente adquiridas para la realización del proyecto y licencias software de terceros necesarias para el desarrollo del proyecto.

Otros gastos directos del proyecto: Incluyen las bolsas de gastos y costes de formación.

Gastos indirectos: Estos costes son los que se relacionan de manera tangencial con el proyecto y las tareas previstas, resultando impracticable su estimación individual.

2.7.1. Gastos asociados al personal

Los gastos correspondientes al pago del equipo de proyecto ascienden a un total de **5.170,87€**, cubriendo un total de 406 horas y media de trabajo. Estos costes se deben íntegramente al trabajo realizado por el alumno, cuyo salario ascendería a 13.600,00€ brutos anuales. Esta cifra ha sido obtenida consultando la Guía Salarial del Sector TIC en Galicia 2016-2017 [11], publicada por Vitae Consultores.

Según esta guía, al rol de Analista/Programador Junior le corresponderían una media de 17.000,00€ brutos anuales a jornada completa (37,5 horas semanales, hasta un total de 1.664 horas cada año). Tomando esto como base y teniendo en cuenta que el alumno trabajará 30 horas semanales, se trataría de un salario bruto mensual de 971,43€ (en 14 pagas).

No obstante, el coste del trabajador para la empresa será mayor que su salario bruto. Utilizando la calculadora de contratos de la Universidad de Santiago de Compostela [12], considerando un contrato del tipo obra y servicio entre el 1 de marzo y el 11 de junio de 2018, realizado por un Ingeniero/a - Licenciado/a - Grado Investigador/a en formación durante 30 horas a la semana, es posible llegar a la conclusión de que habrá que asumir un gasto adicional de 1.224,79€ por la seguridad social del empleado, 549,44€ por pagas extras y 126,16€ de liquidación.

Relativo a los costes correspondientes al salario de los tutores del proyecto, estos no han sido considerados, puesto que en el presente proyecto asumen rol de clientes.

Las Tablas y la Figura mostrados a continuación resumen el coste de los recursos humanos del proyecto:

Salario bruto	3.270,48€
Pagas extras	549,44€
Liquidación	126,16€
Coste contrato	**3.946,08€**

Tabla 2.59: Coste del contrato del alumno

Coste contrato	3.946,08€
Seguridad Social	1.227,79€
Coste total	**5.170,87€**

Tabla 2.60: Coste total de los Recursos Humanos

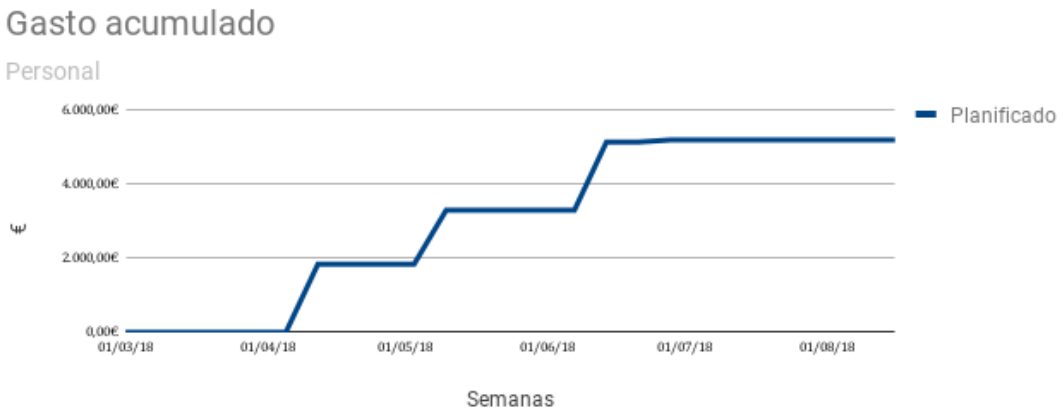


Figura 2.10: Gasto acumulado del proyecto asociado al personal

2.7.2. Gastos asociados a servicios

En el presente proyecto no existen gastos asociados a servicios, pues no se lleva a cabo ningún tipo de subcontratación de empresas que colaboren en el desarrollo del producto. En lo relativo a servicios contratados con un propósito ajeno al desarrollo del software, sus costes se tendrían en cuenta como gastos indirectos.

2.7.3. Gastos asociados a las compras realizadas

Dado que no existen planes para la puesta en producción del producto desarrollado, únicamente se considerarán las compras exclusivamente necesarias para el desarrollo del proyecto.

Nótese que los diversos costes materiales fueron calculados teniendo en cuenta la amortización, su valor residual y su precio base, de acuerdo a la siguiente fórmula:

$$\frac{\text{Precio del producto adquirido}}{12 \times \text{Vida útil del equipo}} \times \text{Meses de duración del proyecto} \qquad (2.1)$$

[2]

[2]Dada la especifidad de algunos de los elementos considerados, no ha sido posible aplicar el coeficiente de amortización de equipos informáticos propuesto por la Agencia Tributaria Española [13].

De esta forma, las diversas adquisiciones y sus respectivos costes para este proyecto de 3,4 meses se listan a continuación:

Lenovo Ideapad Y520-15IKBN: Con un coste en el momento de su adquisición de 699,99€ [14] y una vida útil de 5 años, supondrá **39,67€** de coste para el proyecto.

Dell PowerEdge R720: Con un coste en el momento de su adquisición de 1.640,13€ [15] y una vida útil de 4 años, supondrá **116,18€** de coste para el proyecto.

NVIDIA Jetson TX2 Developer Kit: Con un coste en el momento de su adquisición de 499,45€ [16] y una vida útil de 4 años, supondrá **35,38€** de coste para el proyecto.

NVIDIA Titan Xp: Con un coste en el momento de su adquisición de 1299,00€ [17] y una vida útil de 5 años, supondrá **73,61€** de coste para el proyecto.

Monitor LG 22M47VQ-P: Con un coste en el momento de su adquisición de 109,99€ [18] y una vida útil de 7 años, supondrá **4,45€** de coste para el proyecto.

Teclado Dell 580-ADGS: Con un coste en el momento de su adquisición de 5,50€ [19] y una vida útil de 7 años, supondrá **0,22€** de coste para el proyecto.

Ratón TeckNet Classic 2.4G: Con un coste en el momento de su adquisición de 7,49€ [20] y una vida útil de 7 años, supondrá **0,30€** de coste para el proyecto.

Microsoft Project 2016: Con un coste en el momento de su adquisición de 299,00€ [21] y una vida útil de 7 años, supondrá **12,10€** de coste para el proyecto.

Material de entrega: Se trata de un consumible no amortizable. Se estima un coste total de impresión y elaboración del CD en 20,00€. Dado que es preciso realizar 3 copias, el coste asciende a **60,00€**.

Así, los gastos asociados a las compras realizadas para el proyecto ascienden a **371,91€**, tal y como se puede ver en la siguiente Tabla y Figura resumen:

Lenovo Ideapad Y520-15IKBN	39,67€
Dell PowerEdge R720	116,18€
NVIDIA Jetson TX2 Developer Kit	35,38€
NVIDIA Titan Xp	73,61€
LG 22M47VQ-P	4,45€
Dell 580-ADGS	0,22€
TeckNet Classic 2.4G	0,30€
Microsoft Project 2016	12,10€
Material de entrega	60,00€
Coste total	**371,91€**

Tabla 2.61: Coste total de las compras realizadas

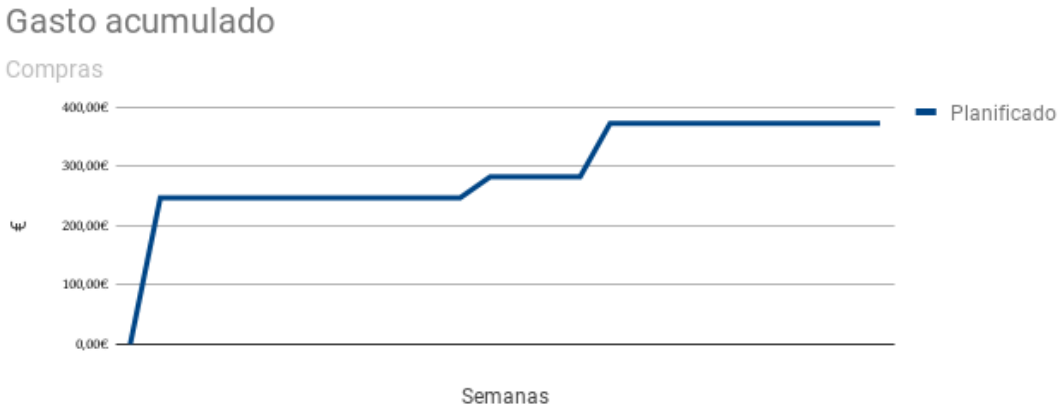


Figura 2.11: Gasto acumulado del proyecto asociado a las compras realizadas

2.7.4. Otros gastos directos

Otros gastos directos relativos al proyecto son los descritos a continuación:

Bolsa de riesgos y costes: Para hacer frente a los posibles riesgos materializados y costes imprevistos, se destinarán **831,42€** a la bolsa de riesgos y costes. Este valor ha sido calculado como el 15 % del capital destinado a compras y a recursos humanos.

El gasto acumulado relativo a otros gastos directos se encuentra descrito el siguiente gráfico resumen:

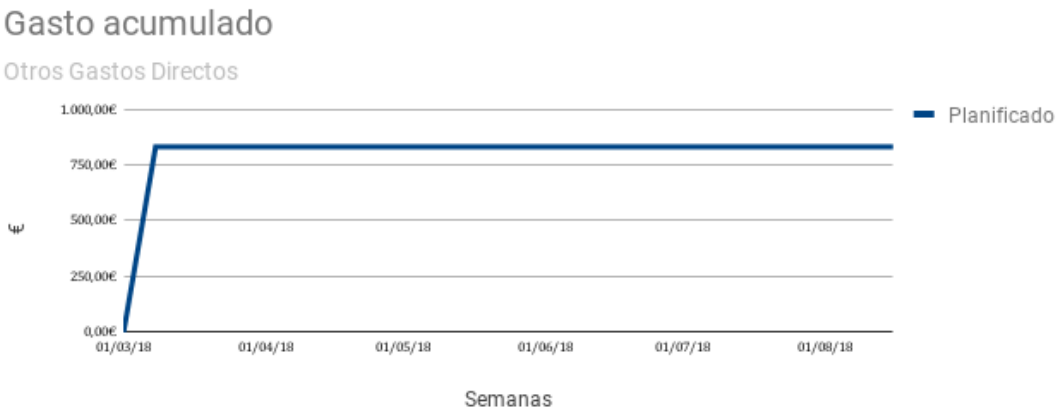


Figura 2.12: Gasto acumulado del proyecto asociado a otros gastos directos

2.7.5. Gastos indirectos

Además de los costes anteriormente desglosados, será necesario tener también en cuenta los costes tangenciales del desarrollo del proyecto, relativos al consumo eléctrico, servicio de Internet, instalaciones y demás agentes. Dado que estos elementos son difícilmente cuantificables, se estimará su coste como el 20 % de los gastos directos, tal y como sugiere la Secretaría General de la Universidad de Santiago de Compostela [22].

De este modo, los gastos indirectos del proyecto ascenderán a **1.274,84€**.

2.7.6. Total de gastos

Siguiendo los costes desglosados en este apartado, el coste total del proyecto asciende a **7.649,04€** tal y como se puede ver en la siguiente Tabla y Figuras resumen:

Gastos asociados al personal	5.170,87€
Gastos asociados a servicios	0,00€
Gastos asociados a las compras	371,91€
Otros gastos directos	831,42€
Gastos indirectos	1.274,84€
Coste total	**7.649,04€**

Tabla 2.62: Gastos totales del proyecto

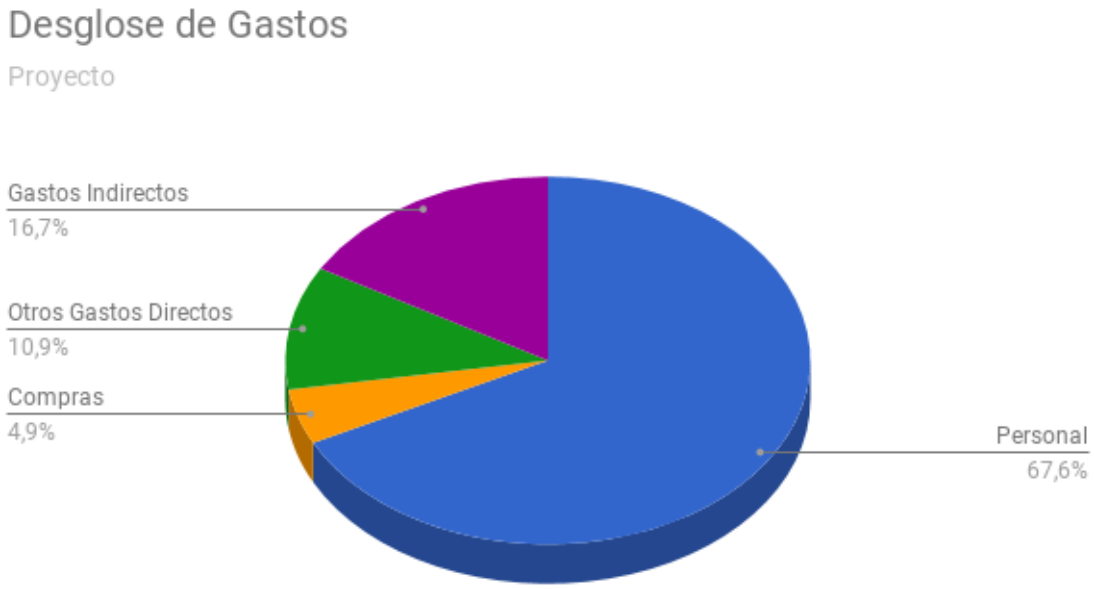


Figura 2.13: Desglose de los gastos del proyecto

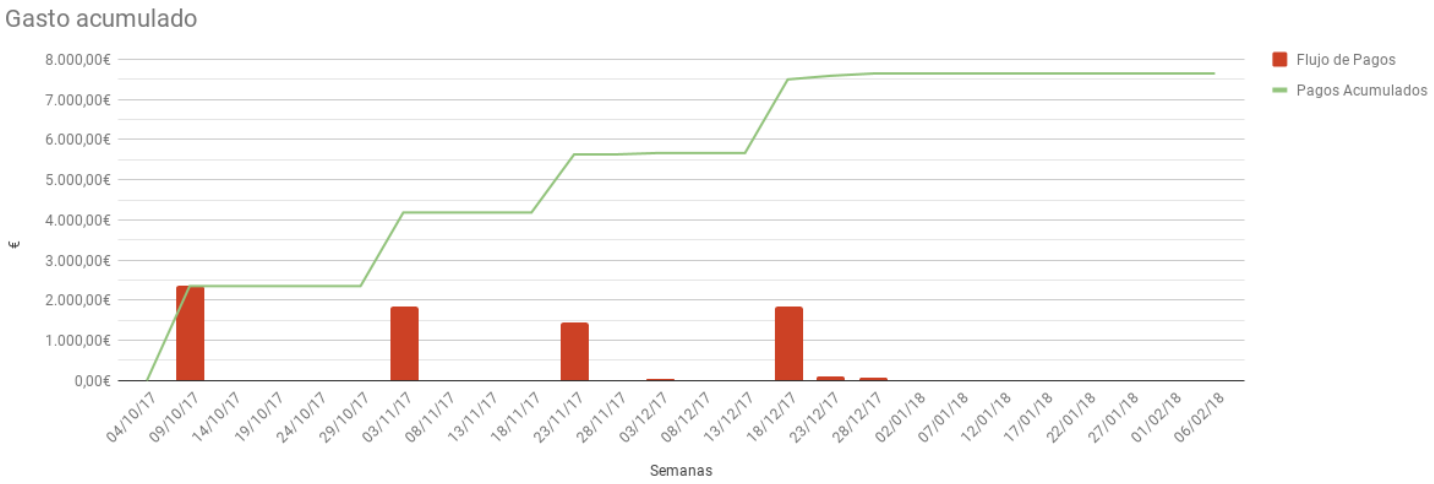


Figura 2.14: Gasto acumulado global del proyecto

Capítulo 3

Análisis de tecnologías y herramientas

En este capítulo se expondrán las tecnologías y herramientas empleadas para el desarrollo y la documentación del proyecto, así como las alternativas consideradas. Es necesario tener en cuenta que existen dependencias entre algunos de los citados elementos, por lo que en ocasiones la elección de una determinada tecnología condiciona el uso de otras.

3.1. Tecnologías y herramientas empleadas en el desarrollo

3.1.1. Plataforma

Dada la imposibilidad de realizar los entrenamientos del modelo en un tiempo razonable en una computadora doméstica, el desarrollo de este proyecto ha requerido de la utilización de dos plataformas para cómputo: un ordenador personal y un servidor de computación GPGPU.

Ordenador personal

El ordenador personal empleado para el desarrollo del proyecto es un portátil **Lenovo Ideapad Y520-15IKBN** [14] con GNU/Linux (**Ubuntu 16.04 LTS**) y **Windows 10** como sistemas operativos instalados.

La elección del sistema operativo GNU se debió a la gran cantidad de herramientas de desarrollo y librerías existentes para Linux, así como el control que ofrece sobre la máquina. Además, la plataforma objetivo del producto desarrollado es Unix, por lo que supone una gran ventaja desarrollar sobre una familia compatible.

Relativo a la utilización de Windows 10, esto se debe a que permite la utilización de herramientas exclusivas de Microsoft, como Microsoft Project [21].

En cuanto a la máquina empleada, se decidió utilizar aquella que ya se encontraba en disposición del alumno. No obstante, las características de la misma supusieron una gran ventaja, pues el hecho de que incorporara una tarjeta gráfica CUDA-Compatible (**NVIDIA GeForce GTX 1050**) permitió realizar numerosas pruebas de concepto y depuración, sin que fuera necesario lanzar la totalidad de los trabajos en el servidor GPGPU.

Servidor de computación GPGPU

El servidor de computación empleado fue un **Dell PowerEdge R720** [15] con sistema operativo **Ubuntu 14.04**, cuyo acceso fue facilitado por el CiTIUS (Centro Singular de Investigación en Tecnoloxías da Información).

El hecho de que el servidor de computación poseyera una tarjeta gráfica para computación de propósito general (una **NVIDIA Titan Xp** [17], en este caso) fue un factor determinante en su elección, pues las características GPGPU (bajo precio en relación a su potencia de cálculo, gran paralelismo, optimización para operaciones en punto flotante, ...) permiten acelerar enormemente los cálculos realizados en la red neuronal.

Dispositivo de computación integrado

Para las pruebas de funcionamiento del sistema en dispositivos integrados, se empleó un **NVIDIA Jetson TX2 Developer Kit** [16] con sistema operativo **Ubuntu 14.04** instalado, cuyo acceso fue facilitado por el CiTIUS. Esta elección se encontró motivada por la potente capacidad de computación GPGPU que ofrece (pese a ser un dispositivo integrado) y el hecho de que ya formaba parte del inventario del departamento.

3.1.2. Entorno de desarrollo

El Entorno de Desarrollo Integrado empleado fue **PyCharm** [23], un IDE específico para Python desarrollado por la compañía JetBrains. Esta elección se vio fuertemente condicionada por el lenguaje de programación utilizado y por el hecho de que pudo ser empleado a coste cero, debido al *JetBrains Educational Pack* ofrecido gratuitamente a estudiantes.

El uso de este IDE permitió facilitar el aprendizaje del lenguaje Python y la labor de desarrollo, así como aumentar la productividad.

3.1.3. Lenguaje de programación

El lenguaje de programación escogido para la implementación del sistema fue **Python** [24], un lenguaje interpretado multiparadigma que usa tipado dinámico. Su elección se basó fundamentalmente en la facilidad de desarrollo y aprendizaje que ofrece, así como la gran cantidad de librerías de aprendizaje automático y tratamiento de imágenes que se encuentran disponibles.

Otros lenguajes como C o C++ que podrían haber resultado más eficientes, fueron descartados tras la elección de la biblioteca de *machine learning* TensorFlow [25], pues la API que ofrece para estos lenguajes se encuentra más limitada. No obstante, en un entorno en producción, donde la definición de la red neuronal ya es estable y el rendimiento es crítico, resultarían una elección muy adecuada.

3.1.4. Biblioteca de aprendizaje automático

A la hora de seleccionar una biblioteca de aprendizaje automático para poder construir y entrenar redes neuronales, se optó en un primer momento por usar Caffe [26]. Esto se debió a que posee una suavizada curva de aprendizaje, es muy modular y destaca por su eficiencia. Además, es una librería frecuentemente utilizada en los proyectos de los tutores del presente trabajo, por lo que hubiera sido sencilla la resolución de posibles dudas. No obstante, la red a implementar posee una operación de convolución con un filtro dinámico, la cual no se encuentra soportada en Caffe de forma nativa. Así que, tras analizar la dificultad de implementación de un nuevo tipo de capa en Caffe, se optó por descartar dicha librería y buscar una alternativa.

Con la finalidad de aprovechar los avances realizados sobre Caffe, se buscó la posibilidad de realizar la convolución con filtro dinámico en Caffe2, una versión de Caffe con más características, pero no hubo éxito. Es por esto que se terminó escogiendo la biblioteca desarrollada por Google, **TensorFlow** [25]. Esta biblioteca, si bien posee una curva de aprendizaje más pronunciada, permite la realización de convoluciones con filtros dinámicos con relativa facilidad.

3.1.5. Visualizador del comportamiento de la red

Dado que se optó por utilizar TensorFlow como biblioteca de aprendizaje automático, resultó inmediata la elección de **TensorBoard** [27] para el análisis del comportamiento de la red. Esta suite de aplicaciones web se encuentra especialmente diseñada para desglosar e inspeccionar los grafos y ejecuciones de TensorFlow, y resultó una herramienta clave para depuración y la monitorización de los entrenamientos.

3.1.6. Base datos de vídeos etiquetados

Para el correcto entrenamiento de la red neuronal, se contó con el acceso a un gran número de vídeos con objetos etiquetados fotograma a fotograma. La base de datos de vídeos escogida fue la facilitada por el **ImageNet Large Scale**

Visual Recognition Challenge [28], puesto que cuenta con más de un millón de fotogramas etiquetados, los cuales contemplan una amplia cantidad de situaciones diferentes. Para poder descargar los vídeos etiquetados, fue necesario registrar una cuenta de estudiante o investigador y aceptar un acuerdo de no divulgación del material descargado.

3.1.7. Herramienta de software matemático

Debido a que numerosos kits de herramientas para la evaluación y comparación de modelos de *tracking* se encuentran desarrollados sobre programas matemáticos, fue necesaria la obtención de software de este tipo. En un principio se optó por la herramienta MATLAB [29], dada su mayor compatibilidad y características. Sin embargo, resultó imposible la obtención de una licencia con una versión compatible, por lo que se recurrió a su equivalente libre, **GNU Octave** [30].

3.1.8. *Benchmark* de *tracking*

Con la finalidad de evaluar los resultados obtenidos por el producto, se hizo necesaria su comparación con otras soluciones de *tracking*. Para poder realizar esto de forma cuantificable y objetiva, se recurrió a *benchmarks* de *tracking* públicos. En este caso, dada su popularidad y repercusión en el ámbito, se decidió utilizar las herramientas proporcionadas por el **Visual Object Tracking Challenge** [31] y el **Online Object Tracking Benchmark** [32] para obtener sus métricas.

3.1.9. Herramienta de visualizacion de imágenes

Para poder mostrarle al usuario los resultados del *tracking* realizado, se hizo necesario recurrir a herramientas de visualización de imágenes compatibles con Python. La herramienta escogida en este caso fue la biblioteca libre de visión artificial, **OpenCV** [33], puesto que proporciona una gran velocidad y multitud de características. No obstante, la compatibilidad entre versiones de OpenCV puede llegar a ser algo problemática, por lo que también se incorporó soporte para la librería **Matplotlib** [34], pese a ser menos veloz.

3.1.10. Soporte para computación de propósito general en GPU

Además de las tarjetas gráficas anteriormente mencionadas, para la ejecución del producto en arquitecturas GPU fueron necesarias las siguientes herramientas, impuestas por la librería TensorFlow:

CUDA [35]

Plataforma de computación en paralelo que permite emplear una variación del lenguaje de programación C para codificar algoritmos en GPU de NVIDIA.

cuDNN [36]

Librería de primitivas basadas en GPU para redes neuronales profundas.

3.1.11. Herramienta de despliegue de contenedores

Con la finalidad de facilitar la demostración del funcionamiento del producto y el testing, se decidió hacer uso de contenedores para agrupar las dependencias y simplificar el despliegue. Dada su popularidad y su excelente soporte para GPU, se optó por utilizar la aplicación **NVIDIA Docker** [37].

3.1.12. Software de control de versiones

Para gestionar la configuración del proyecto, se utilizó el software de control de versiones, **Git** [38]. Esta elección se debió a la facilidad de uso que ofrece y la familiaridad del alumno con el mismo.

3.2. Tecnologías y herramientas empleadas en la documentación

3.2.1. Editor de documentos

Para la edición de la memoria y de los manuales del producto, se decidió utilizar el sistema de composición de textos **LaTeX** [39], el cual ofrece una alta calidad tipográfica con un amplio abanico de posibilidades. Para poder hacer uso de LaTeX, fueron necesarias las siguientes herramientas:

TeX Live [40]

Distribución de software para la composición tipográfica TeX, que se encuentra incluido en un gran número de distribuciones Linux.

Texmaker [41]

Editor distribuido bajo la licencia GPL para escribir documentos LaTeX. Las razones por las cuales se escogió fueron su facilidad de uso e instalación y sus posibilidades de personalización.

3.2.2. Herramienta de administración de proyectos

Para administrar el proyecto desarrollando planes, asignando recursos y realizando el seguimiento de procesos, se empleó el software **Microsoft Project** [21]. Si bien esta herramienta es de pago, resulta mucho más estable que sus contrapartes libres, y ofrece muchas más posibilidades para la creación de calendarios y la búsqueda de caminos críticos y metodologías de eventos en cadena.

3.2.3. Herramientas de diagramación

La creación de las Estructuras de Descomposición de Tareas mostradas a lo largo de la presente memoria ha sido llevada a cabo empleando el editor en línea **WBSTool** [42]. Esta elección se debió a la familiaridad adquirida con la herramienta a lo largo de la carrera y a que permite de forma sencilla generar EDTs y exportarlos a diversos formatos una vez finalizados.

Relativo a los Diagramas de Contexto y de Flujo de Datos, estos fueron realizados con la herramienta online **Draw.io** [43], la cual fue elegida dada su versatilidad, facilidad de uso y resultados visualmente atractivos, amén de que se trata de código abierto.

Por último, los diagramas UML presentados en esta memoria fueron realizados haciendo uso del software **StarUML** [44], el cual ofrece un amplio abanico de posibilidades y ha sido muy utilizado durante la carrera, por lo que no fue necesaria ningún tipo de formación.

3.2.4. Hojas de cálculo

Para la obtención de diagramas y gráficos, así como la clasificación de los resultados obtenidos en las diversas pruebas, se decidió hacer uso de la herramienta **Google Spreadsheets** [45], dado que se encuentra alojada en la nube, resulta muy cómoda de utilizar y ofrece multitud de posibilidades.

Capítulo 4

Diseño e implementación

En el presente capítulo se detalla el diseño llevado a cabo para la creación del sistema y se describen los detalles de implementación más relevantes del mismo.

4.1. Arquitectura del sistema

En la presente sección, se encuentra descrita la arquitectura del sistema, es decir, una definición de alto nivel de sus componentes y de cómo se comunican entre ellos, sin entrar en detalles de implementación.

Partiendo del diagrama de contexto mostrado en la Figura 2.1, utilizado para definir el alcance del proyecto, es posible crear un diagrama de flujo de datos (DFD) de nivel 1 que representa las funciones que realiza el sistema, agrupadas en los módulos de los que se compone:

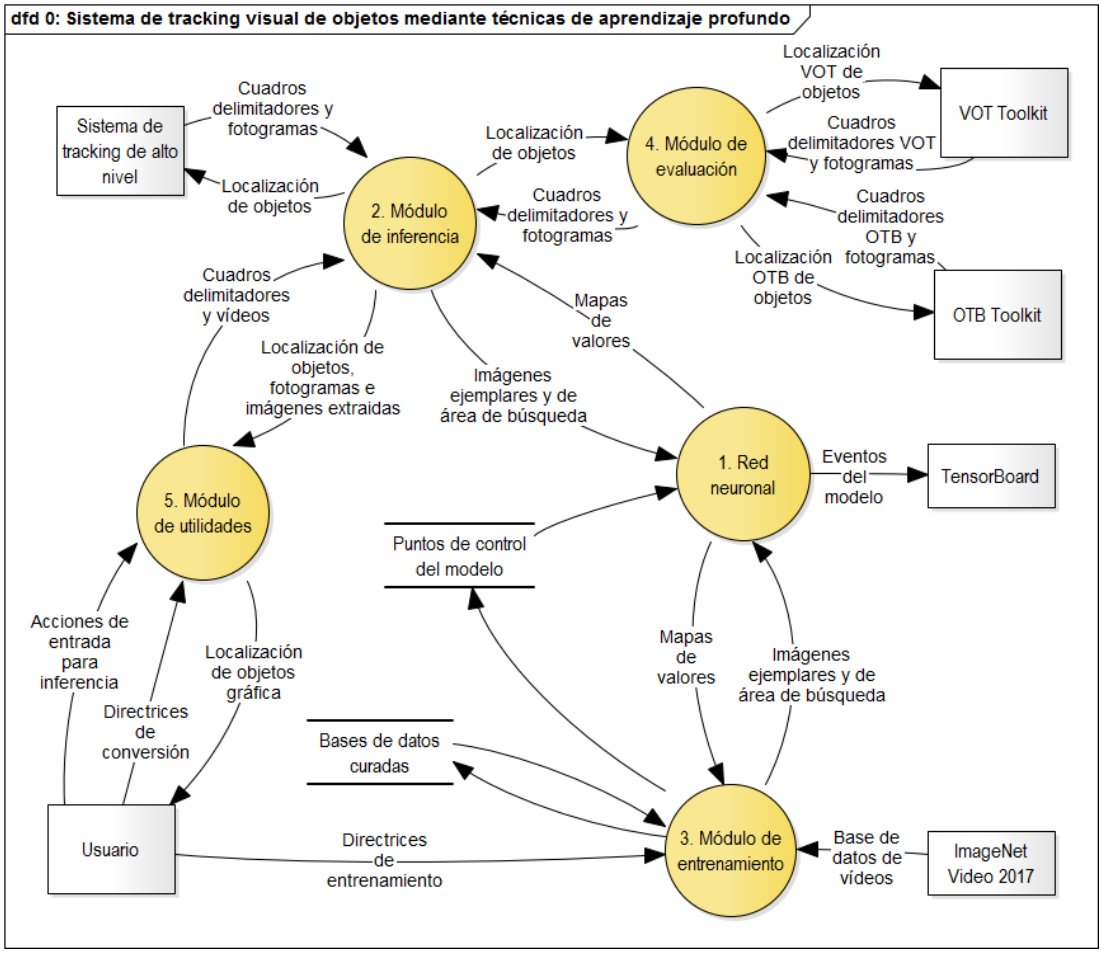


Figura 4.1: Diagrama de flujo de datos de nivel 1

Si bien existen interacciones entre los diversos componentes del sistema, las responsabilidades de cada uno se encuentran claramente definidas, pudiendo evolucionar y ser desarrollados por separado.

A continuación, se detallarán las características del diseño e implementación de cada uno de estos componentes:

4.2. Red neuronal

El subsistema de red neuronal es el núcleo del sistema de tracking, pues es el componente que permite hallar la posición de un objeto en un fotograma partiendo

de su aspecto inicial en el instante de la detección.

Esta red neuronal posee la arquitectura mostrada en la Figura 4.2, la cual puede dividirse en dos tipos de componentes: el extractor de características y el operador de similaridad.

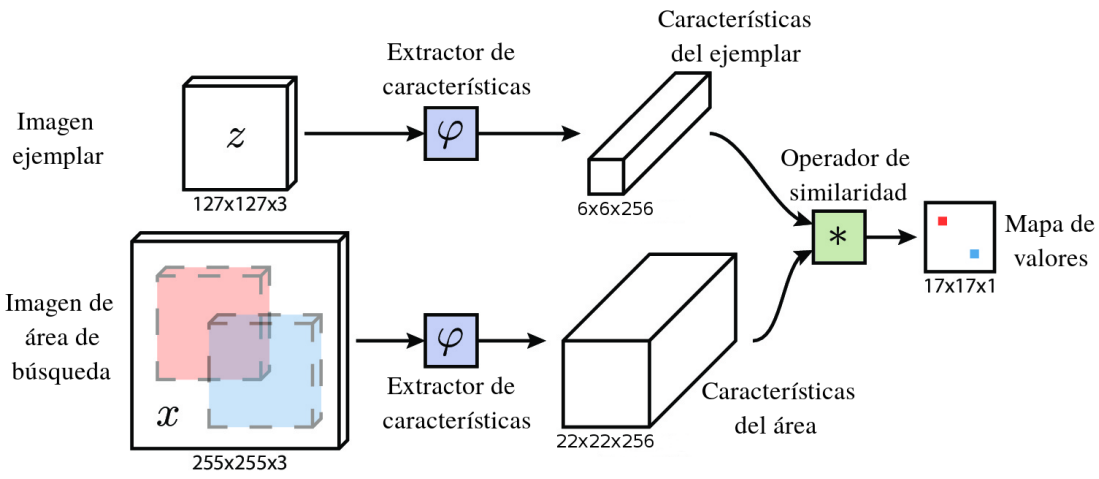


Figura 4.2: Arquitectura de la red neuronal

Para desacoplar la implementación del extractor de características de la del operador de similaridad, estos han sido diseñados de la forma mostrada en el siguiente diagrama de clases:

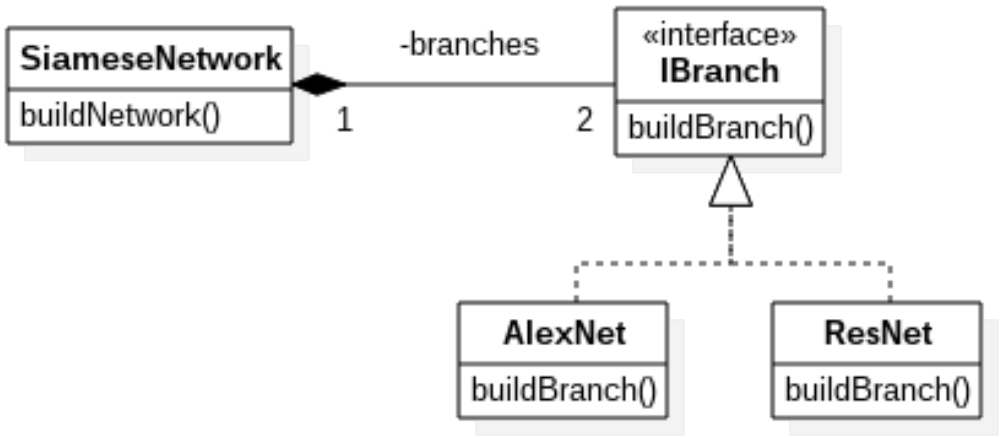


Figura 4.3: Diagrama de clases de la red neuronal

Las instancias del tipo `IBranch` representarán los extractores de características, mientras que las del tipo `SiameseNetwork` implementarán la operación de similaridad.

De esta forma, mediante la función `buildNetwork()` de la clase `SiameseNetwork`, se podrán crear redes siamesas como la mostrada en la Figura 4.2. Dado que TensorFlow realiza las operaciones de forma "perezosa", esta función únicamente creará la estructura de la red y devolverá unos tensores vacíos como resultado. Las operaciones no serán realizadas hasta que dichos tensores no sean evaluados.

4.2.1. Extractor de características

El extractor de características de la red procesa imágenes a color (con 3 canales) y las transforma en conjuntos de datos informativos no redundantes. Estas nuevas matrices de datos, de 256 canales de profundidad, contendrán información como las posiciones de bordes y esquinas, las acumulaciones de color o los cambios de brillo, los cuales ayudarán a la máquina a identificar lo representado en la imagen.

Para esta labor de extracción de características, se usa una versión de la red AlexNet [9], pero podría usarse cualquier otro tipo de extractor, basado en ResNet [46] o ResNeXt [47], por ejemplo. No obstante, AlexNet ofrece unos buenos resultados a una muy elevada tasa de imágenes por segundo, que es lo que esencialmente se persigue. Es por esto que se ha escogido como base del extractor.

Así, la red cuenta con dos ramas siamesas (las imágenes pasan por las mismas capas, que cuentan con la misma configuración con idénticos parámetros y pesos) basadas en AlexNet, las cuales admiten una imagen de ejemplo (el aspecto original del objeto a localizar) de 127×127 píxeles y una imagen del área de búsqueda (procedente del fotograma actual) de 255×255 píxeles, ambas a color (48.387 bytes y 195.075 bytes de datos, respectivamemente). La estructura de capas de estas ramas es la que se muestra a continuación, en la Tabla 4.1:

Capa	Tamaño del kernel	Número de filtros	Paso (*stride*)	Tamaño de activación de la imagen de ejemplo	Tamaño de activación del área de búsqueda	Canales
				127×127	255×255	$\times 3$
conv1	11×11	96	2	59×59	123×123	$\times 96$
pool1	3×3		2	29×29	61×61	$\times 96$
conv2	5×5	128 ($\times 2$)	1	25×25	57×57	$\times 256$
pool2	3×3		2	12×12	28×28	$\times 256$
conv3	3×3	384	1	10×10	26×26	$\times 384$
conv4	3×3	192 ($\times 2$)	1	8×8	24×24	$\times 384$
conv5	3×3	128 ($\times 2$)	1	6×6	22×22	$\times 256$

Tabla 4.1: Estructura de capas del extractor de características

Todas las capas convolucionales, excepto la última, cuentan con una función de activación ReLU [48], y no introducen ningún tipo de relleno (*padding*) en las operaciones, para mantener la naturaleza totalmente convolucional del modelo. Además, durante la fase de entrenamiento de la red neuronal, se lleva a cabo un proceso de normalización por lotes (*batch normalization*) [49] inmediatamente después de cada capa convolucional, para que así la varianza sea 1 y la media 0.

De este modo, tras la extracción de características, por cada nuevo fotograma se obtendrán dos matrices tridimensionales: un mapa de características $6 \times 6 \times 256$ de la imagen ejemplar y un mapa de características $22 \times 22 \times 256$ de la imagen de área de búsqueda. No obstante, para mejorar el entrenamiento y permitir cambios de escala durante la inferencia y evaluación, se añadirá una dimensión más a las matrices de entrada y de salida, que representará el tamaño del lote (*batch size*).

Dado que el modelo de *tracking* propuesto por Bertinetto y Valmadre no contempla la actualización de la imagen ejemplar a lo largo del tiempo (es decir, desde que se detecta un objeto, siempre se emplea la misma imagen ejemplar para este), es posible optimizar el cálculo de las características de las imágenes. De esta forma, las características del ejemplar sólo se computarán una vez, y en cada nuevo ciclo se calcularán únicamente las características del área de búsqueda, que será diferente cada vez.

Para la implementación del extractor de características siamés, se hizo uso de la

librería de TensorFlow, TF-Slim [50], pues ofrece las mismas operaciones de normalización por lotes que MatConvNet, el lenguaje original en el que se encuentra implementado SiamFC.

4.2.2. Operador de similaridad

El operador de similaridad de la red neuronal es el encargado de tomar como entrada dos mapas de características y generar un mapa de valores que indique la probabilidad para cada una de las secciones del área de búsqueda de que el objeto buscado se encuentre allí. En este caso, esta operación de similaridad es llevada a cabo mediante una capa de correlación cruzada entre las características del ejemplar y del área, seguida de una capa de ajuste que busca normalizar el resultado.

De este modo, suponiendo unas entradas de tamaño $6\times6\times256$ y $22\times22\times256$, mediante la correlación cruzada se obtendrá una salida de tamaño $17\times17\times1$, equivalente a un mapa de calor en el que los valores más altos indican que existe una mayor probabilidad de que el objeto se encuentre en esos puntos (extrapolados a puntos del área de búsqueda). Estas dimensiones resultado se obtienen dado que las características del ejemplar ($6\times6\times256$) se utilizan como un filtro sobre las carácterísticas del área de búsqueda ($22\times22\times256$), siendo desplazadas de 1 en 1 horizontal y verticalmente, como explica la Figura 4.4.

Nuevamente, el operador de similaridad también deberá admitir el procesado por lotes, por lo que en realidad lo que recibirá como entrada serán matrices de cuatro dimensiones ($n\times6\times6\times256$ y $n\times22\times22\times256$), y devolverá como salida una matriz de $n\times17\times17\times1$ elementos, en la que cada "rebanada" (*slice*) se corresponderá con el mapa de valores de un par de imágenes. El valor n utilizado será el tamaño del lote, es decir, el número de pares de imágenes procesados simultáneamente.

4.2.3. Pesos de la red

Como bien se ha explicado a lo largo de esta memoria, una red neuronal precisa de unos pesos correctamente entrenados en cada una de sus capas para poder extraer

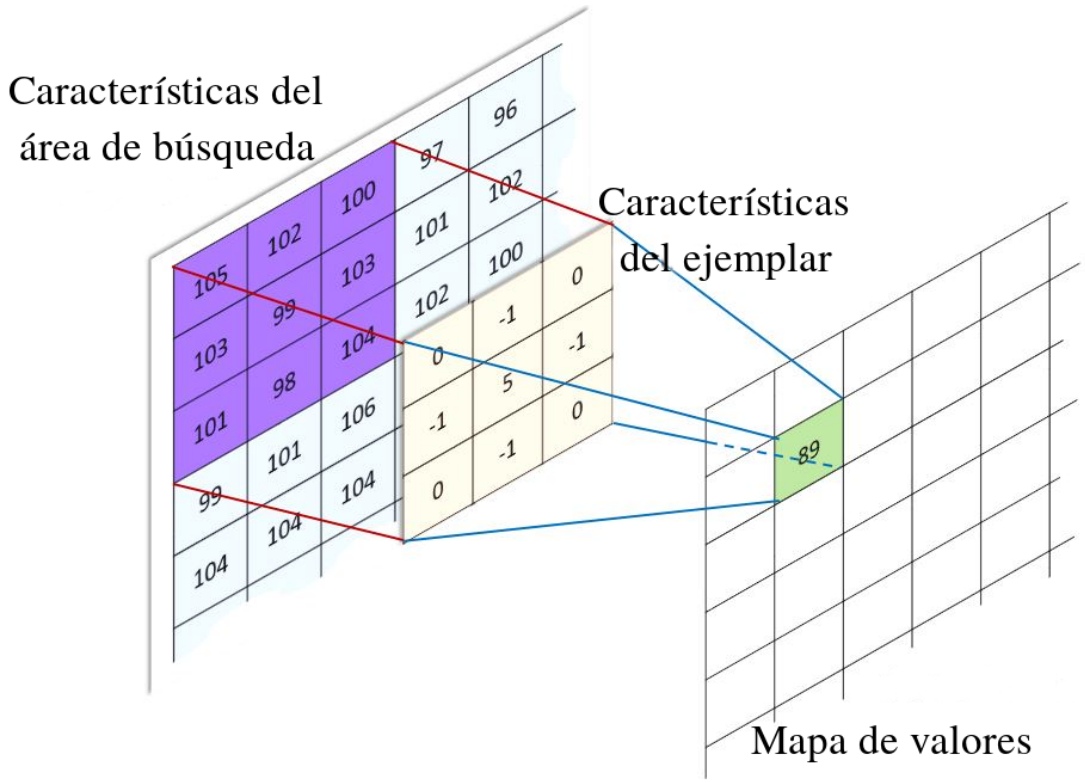


Figura 4.4: Funcionamiento de la correlación cruzada

resultados coherentes. Estos pesos entrenados, se encontrarán almacenados en los puntos de control del modelo y serán aplicados sobre la red neuronal siempre que se realicen inferencias o evaluaciones o se decida retomar un entrenamiento.

4.2.4. Eventos del modelo

A la hora de hacer uso de la red, tanto en modo de entrenamiento como en inferencia o validación, se ha configurado un sistema de recogida de eventos. Esta funcionalidad se encarga de almacenar en un formato compatible con TensorBoard tanto la información de estructura de la red como los datos que la atraviesan y la información de rendimiento. De este modo, es posible visualizar de forma sencilla y en tiempo real detalles de la red neuronal, como su grafo de ejecución, mostrado en la Figura 4.5:

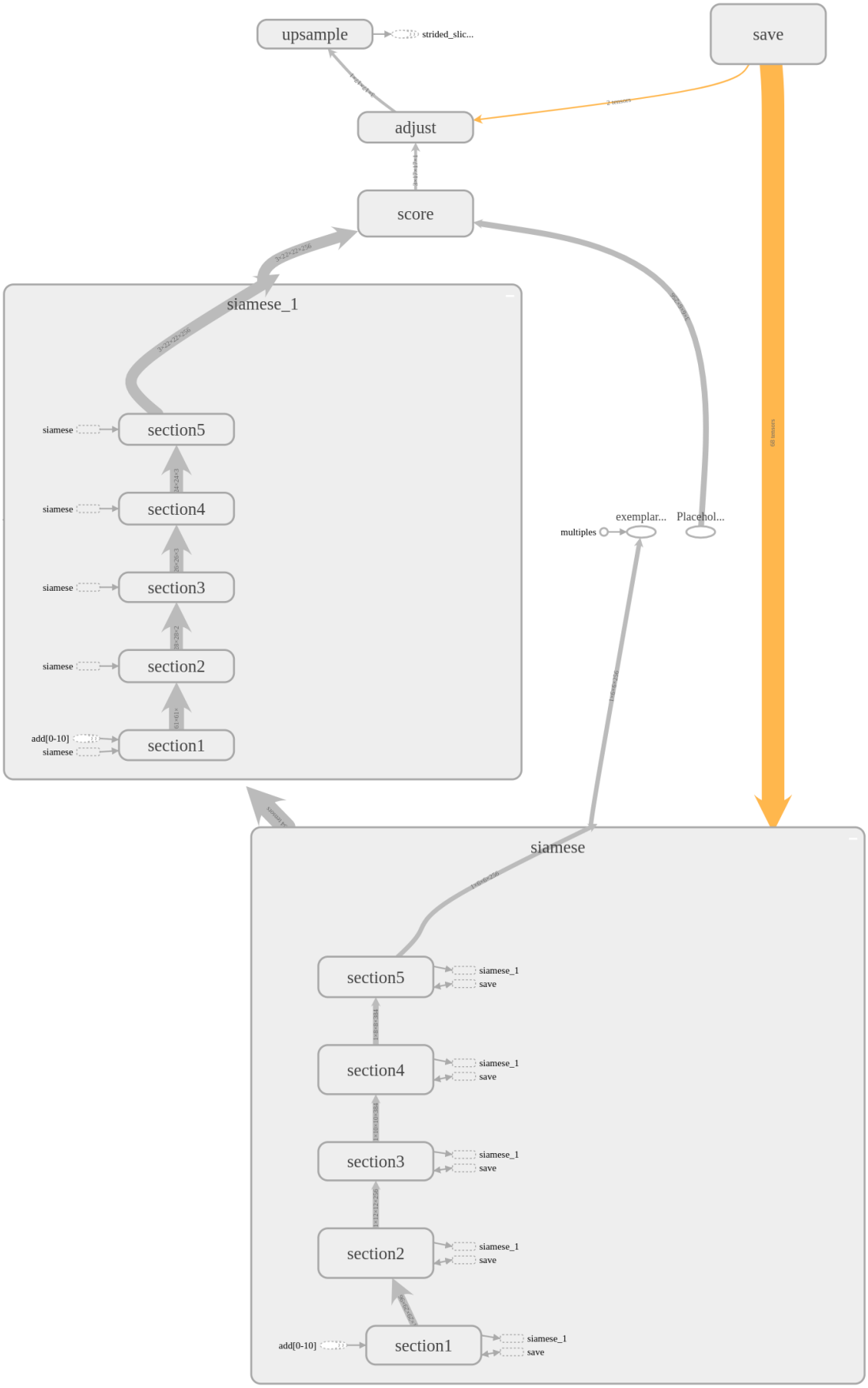


Figura 4.5: Grafo de ejecución de la red, visualizado mediante TensorBoard

4.3. Módulo de inferencia

El módulo de inferencia es el encargado de procesar los fotogramas recibidos para su transmisión a la red neuronal, así como de interpretar las salidas generadas por esta última y hacerlas accesibles desde fuera del sistema.

Este conjunto de responsabilidades ha sido dividido entre dos clases, tal y como se puede ver en la Figura 4.6:

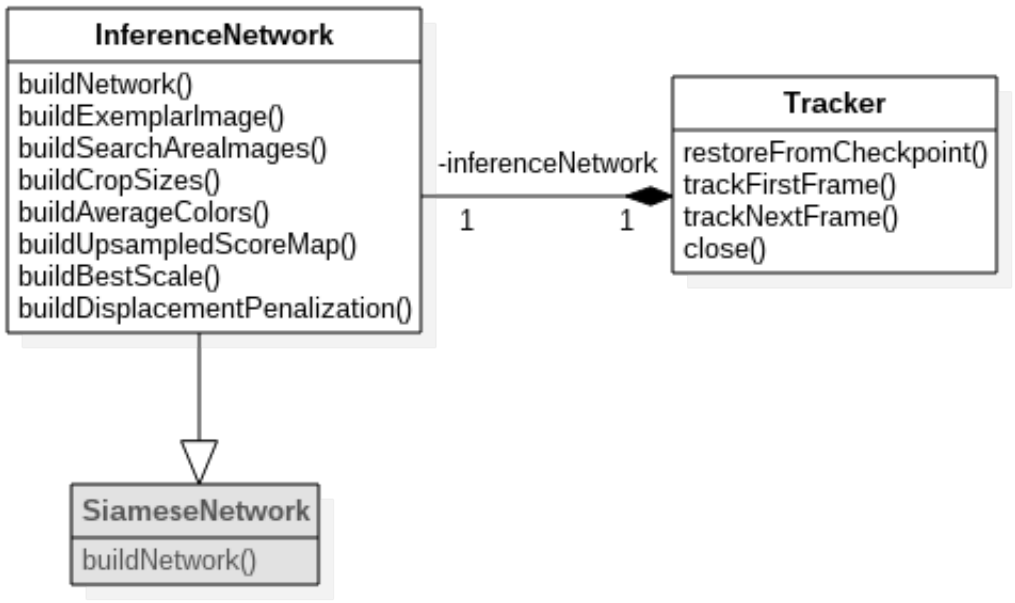


Figura 4.6: Diagrama de clases del módulo de inferencia

4.3.1. Red de inferencia (`InferenceNetwork`)

La clase `InferenceNetwork` es una especialización de la clase `SiameseNetwork`, la cual añade un gran número de funcionalidades para la transformación de fotogramas en imágenes ejemplares y de búsqueda y la conversión de mapas de valores en coordenadas. A continuación, se describirán con más detalle estas nuevas funciones:

`buildCropSizes()`

Partiendo de las dimensiones del cuadro delimitador del objeto, esta función realiza el cálculo del tamaño que deberán tener la imagen ejemplar y la imagen del área de búsqueda, recortadas sobre el fotograma. Es decir, pese a que finalmente serán escaladas a 127×127 píxeles y 255×255 píxeles, respectivamente, el recorte a realizar sobre el fotograma no tendrá por qué tener necesariamente este tamaño.

Tal y como se indica en la publicación de Bertinetto y Valmadre, para el recorte de la imagen ejemplar de un objeto de dimensiones $w \times h$ y un margen establecido de p, se trazará un cuadrado (centrado en el objeto) cuya área escalada por s sea igual a una constante A:

$$s(w + 2p) \times s(h + 2p) = A \tag{4.1}$$

En este caso, dado que las imágenes ejemplares serán de 127×127 píxeles, A será igual a 127^2.

En cuanto a la imagen del área de búsqueda, dado que serán de 255×255 píxeles, compartiendo escala con la imagen ejemplar, su tamaño de recorte en el fotograma será de aproximadamente cuatro veces el de la imagen ejemplar del objeto.

`buildAverageColors()`

Esta función crea un tensor que calculará la media de colores de un fotograma. Este valor será utilizado con fines estadísticos y a la hora de completar la información faltante cuando el objeto a seguir se encuentre en los bordes de la imagen.

`buildExemplarImage()`

Esta función parte de un fotograma y la localización de un objeto (descrita por un cuadro delimitador), y extrae su imagen ejemplar. Esta extracción se lleva a cabo en dos pasos:

1. Primeramente, haciendo uso de los valores calculados por `buildCropSizes()`, se recorta del fotograma un cuadrado centrado en el objeto. Dado que el recorte se realiza dejando unos píxeles de margen, si el objeto se encuentra en un lateral de la imagen, es posible que el área de recorte se salga fuera del fotograma. En estos casos, la información faltante se completará con los colores medios (`buildAverageColors()`) del fotograma.

2. Una vez recortada la imagen, esta es escalada mediante interpolación bilineal para que tenga el tamaño exacto de imagen ejemplar definido para la red (127x127 píxeles en esta versión).

El resultado de recortar un fotograma para obtener su imagen ejemplar se muestra a continuación:

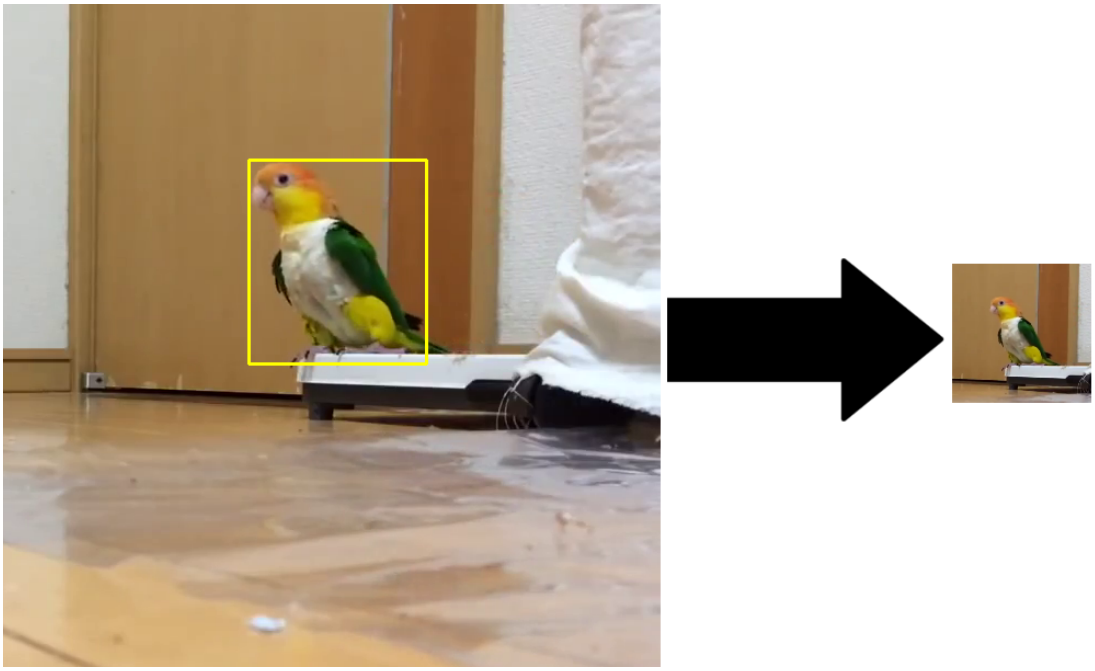


Figura 4.7: Extracción de la imagen ejemplar

`buildSearchAreaImages()`

Esta función parte de un fotograma y de la última localización conocida de un objeto, descrita por un cuadro delimitador, y extrae sus imágenes del área de búsqueda. La red no buscará al objeto en todo el fotograma, únicamente lo hará

en esta región.

Este proceso es muy similar al de la extracción de la imagen ejemplar, pero añadiendo un paso extra antes del escalado:

1.1 Dado que la red también deberá detectar cambios de tamaño en el objeto, se creará una pirámide de escalas del área de búsqueda. Esta variación se encuentra controlada por hiperparámetros, pero actualmente se manejan 3 escalas distintas distanciadas por un factor de 1,0375.

De esta forma, la red ya no trabajará con imágenes de entrada, sino con lotes (*batches*) de imágenes de entrada. Así, para imágenes a color (3 canales) y considerando, por ejemplo, 5 escalas, los tensores de entrada de la red neuronal serán un lote de imágenes ejemplares de $5\times127\times127\times3$ elementos y un lote de imágenes de área de búsqueda de $5\times255\times255\times3$ elementos.

El resultado de recortar un fotograma para obtener sus imágenes de área de búsqueda para 3 escalas se muestra a continuación:

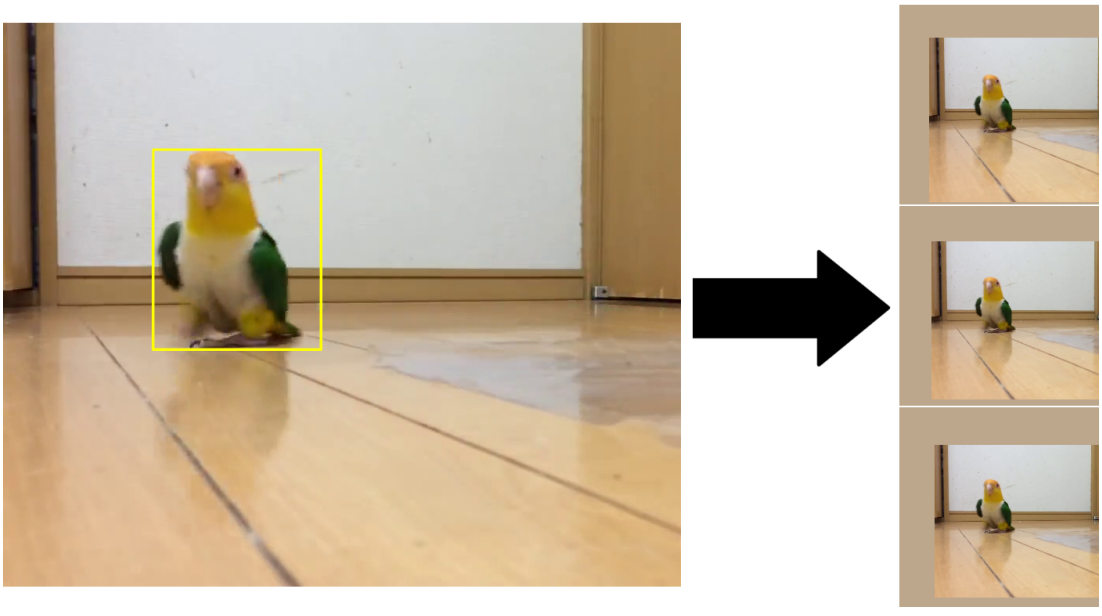


Figura 4.8: Extracción de la pirámide de imágenes de área de búsqueda para 3 escalas

`buildUpsamledScoreMap()`

Dado que el mapa de valores resultante de la red neuronal es relativamente abrupto (17×17 elementos), esta función aplicará un sobremuestreo mediante interpolación bicúbica para obtener una matriz de 272×272 elementos, que permitirá detectar cambios de posición de forma más suave. Esta transformación será realizada para todos los mapas de valores del lote obtenidos.

`buildBestScale()`

Esta función recibe como entrada un lote de mapas de valores sobremuestreados y devuelve el que tiene más probabilidades de representar la escala correcta. Este proceso es llevado a cabo en dos pasos:

1. Primeramente, se seleccionan los elementos con el valor más alto de cada uno de los mapas de valores.
2. Tras esto, se aplica una penalización a los elementos de los mapas de valores que representan escalas distintas de la actual. La escala del elemento con el valor más alto será la considerad correcta.

`buildDisplacementPenalization()`

Para poder distinguir al objeto seguido de otros elementos con un aspecto similar dentro del área de búsqueda, se hace necesario realizar una última transformación sobre el mapa de valores obtenido de los anteriores pasos. Dado que existe una gran probabilidad de que el objeto seguido no haya sufrido un gran desplazamiento, la función `buildDisplacementPenalization()` aplicará una ventana de coseno sobre el mapa de valores, la cual perjudicará a los elementos más distantes del centro.

`buildNetwork()`

Por último, la función `buildNetwork()` aplicará las operaciones anteriormente descritas en orden para que, a partir de un fotograma y el último cuadro delimi-

tador del objeto conocido, se pueda obtener un mapa de valores sobremuestreado y penalizado en la escala correcta, cuyo mayor elemento indique la posición más probable del objeto.

Esta función también se encarga de optimizar la inferencia de la red, permitiendo el almacenado en caché de las características de la imagen ejemplar, requiriendo su cálculo una única vez.

4.3.2. Motor de inferencia (`Tracker`)

En la clase `Tracker` se encuentran recogidas un conjunto de funciones para hacer simple la realización del *tracking*, sin necesidad de entrar en detalles del uso de redes neuronales. Las funciones que ofrece son las siguientes:

`restoreFromCheckpoint()`

Permite, de forma sencilla, aplicar a la red neuronal los pesos recogidos en un punto de control del modelo. Este es un requisito para el correcto funcionamiento de la inferencia, pues son estos pesos los que permiten obtener un mapa de valores coherente partiendo de una imagen ejemplar y una imagen de área de búsqueda.

`trackFirstFrame()`

Esta función, que recibe como entrada un fotograma y el cuadro delimitador de un objeto, extrae las características de la imagen ejemplar de dicho objeto y las almacena en caché, para que la inferencia pueda ser realizada con una mayor velocidad.

`trackNextFrame()`

Esta función recibe como entrada un fotograma a partir del cual (utilizando la última posición conocida del objeto y las características de la imagen ejemplar) extrae un mapa de valores asociado a una escala. Tras esto, haciendo uso de la posición del mayor elemento y del cambio de escala, es capaz de devolver un

nuevo cuadro delimitador que, en coordenadas del fotograma, exprese la nueva posición y tamaño del objeto a seguir.

Es necesario hacer un especial hincapié en la no trivialidad de esta conversión de coordenadas, pues deberá pasarse de un mapa de valores sobremuestreado a una imagen del área de búsqueda, cuyas esquinas no se corresponden con las del mapa de valores, dado que la correlación cruzada y convoluciones no aplican relleno para que el centro del filtro llegue a los bordes. Tras esto, se aplicará la escala necesaria y se desplazarán estos puntos al fotograma original.

`close()`

Por último, la función `close()` libera todos los recursos empleados.

4.3.3. Adaptación a múltiples objetos

Tal y como se indica entre los requisitos del sistema, se desea que la aplicación sea capaz de realizar inferencias de múltiples objetos simultáneos sin replicar el modelo utilizado, es decir, debe soportar el *tracking* de múltiples objetos de forma nativa. Para lograr esto, se han desarrollado otro par de clases de red de inferencia y motor de inferencia que soportan múltiples objetos de forma nativa.

El diagrama de clases que refleja esta incorporación se encuentra recogido en la Figura 4.9:

Si bien una gran parte de las funciones que llevan a cabo se ven muy similares, reciben y retornan unas estructuras de datos diferentes de las simples. A nivel lógico, llevan a cabo las mismas tareas, pero la forma de realizarlas es muy distinta.

Estos modelos para *tracking* de múltiples objetos resultan menos eficientes que el inicialmente mostrado si se quiere llevar a cabo el seguimiento de un único elemento. No obstante, mejoran en gran medida su rendimiento y consumo de memoria cuando existe más de un objeto a seguir (es por esto que ambas aproximaciones se conservan como parte de la aplicación).

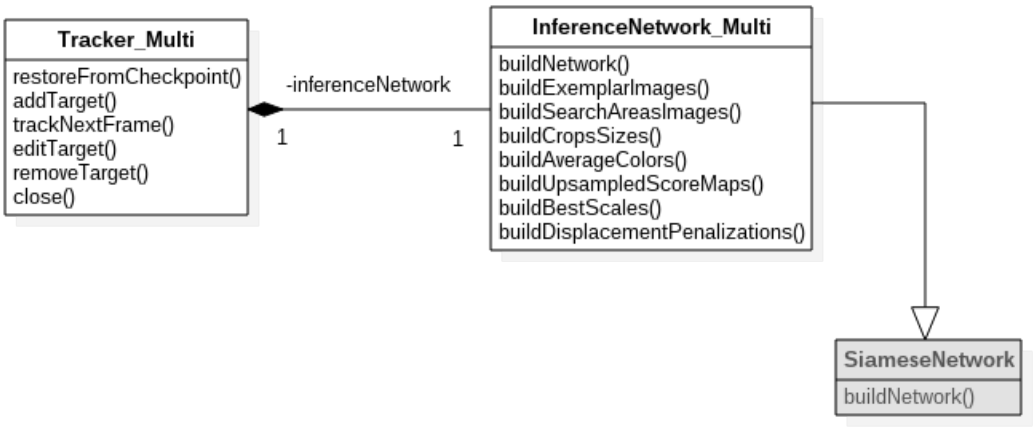


Figura 4.9: Diagrama de clases del módulo de inferencia para múltiples objetos

A continuación, se describen las particularidades del *tracking* de múltiples objetos, aplicadas a cada una de las clases presentadas.

Red de inferencia múltiple (`InferenceNetwork_Multi`)

Esta clase, en esencia, lleva a cabo lo mismo que la red neuronal para objetos simples, pues genera mapas de valores. Sin embargo, recibirá como entrada un fotograma y una lista de cuadros delimitadores (en vez de uno único), y devolverá una lista de mapas de valores, uno por cada uno de los objetos seguidos.

Para poder realizar esto, los diversos objetos son tratados de forma muy similar a las distintas escalas, siendo agrupados en un lote. No obstante, dado que las convoluciones bidimensionales utilizadas no admiten tensores de más de 4 dimensiones, hubo que "apilar" (*stack*) la información de los distintos objetos en la dimensión del lote. De esta forma, si por ejemplo se desea hacer *tracking* a 5 escalas de 7 objetos en un vídeo a color, se generarán unos tensores de entrada para la imagen ejemplar y el área de búsqueda de $35\times127\times127\times3$ y $35\times255\times255\times3$ elementos, respectivamente. Y se producirá como salida un conjunto de mapas de valores con la escala correcta y sobremuestreados, de dimensiones $7\times272\times272\times1$.

Así, al realizar las operaciones de la red neuronal de este modo, no es necesario modificar la red siamesa subyacente ni los pesos entrenados, y la GPU es capaz

de procesar en paralelo las distintas escalas y objetos de los que se lleva a cabo el *tracking*.

Motor de inferencia múltiple (`Tracker_Multi`)

En lo relativo al motor de inferencia múltiple, el mayor cambio sufrido de cara al usuario radica en que existen nuevas funciones a usar, las cuales permiten un control más preciso y granular de cada uno de los objetos seguidos. La función `trackFirstFrame()` ha sido "sustituida" por `addTarget()`, que en esencia realiza lo mismo (extraer las características de la imagen ejemplar de un objeto y almacenarlas en caché), pero ya no tiene por qué ejecutarse en el primer fotograma ni una única vez. En cuanto a `trackNextFrame()`, ahora devolverá una lista de objetos con sus cuadros delimitadores asociados.

4.4. Módulo de entrenamiento

El módulo de entrenamiento es el encargado de generar una serie de puntos de control del modelo que contengan un conjunto de pesos que permitan realizar inferencias correctas. Esto se lleva a cabo partiendo de la red siamesa desarrollada, y utilizando un conjunto de vídeos curado, compatible con el método de entrenamiento.

De esta forma, existirán dos componentes dentro de este módulo: el sistema de entrenamiento y las herramientas de curación.

4.4.1. Sistema de entrenamiento

De modo muy similar al módulo de inferencia, el sistema de entrenamiento se encuentra dividido en dos clases, como bien se describe en el siguiente diagrama:

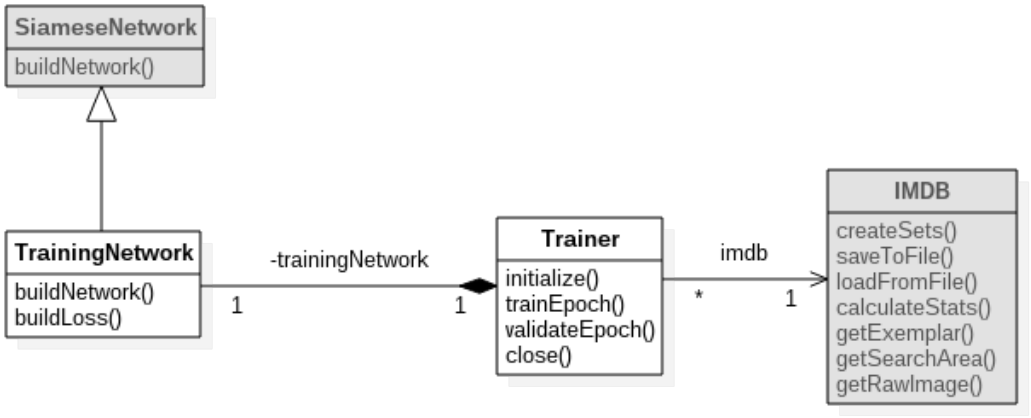


Figura 4.10: Diagrama de clases del sistema de entrenamiento

Red de entrenamiento (`TrainingNetwork`)

La clase `TrainingNetwork` es una especialización de la clase `SiameseNetwork`, la cual añade la funcionalidad de calcular el error o coste (*loss*) de cada intento de inferencia realizado. Estos intentos de inferencia se realizarán en lotes y su coste se calculará como la entropía cruzada sigmoide entre el mapa de valores obtenido y el esperado (*labels*). Así, considerando el mapa de valores creado como el logit (en términos de TensorFlow, valores que serán mapeados a probabilidades mediante Softmax [51]), el coste será:

$$\text{loss} = \text{logits} - \text{logits} * \text{labels} + \log(1 + \exp(-\text{logits})) \tag{4.2}$$

No obstante, para evitar desbordamientos cuando el mapa de valores obtenido tiene componentes menores que cero, TensorFlow realiza el cálculo de la entropía de la siguiente manera:

$$\text{loss} = \max(\text{logits}, 0) - \text{logits} * \text{labels} + \log(1 + \exp(-\text{abs}(\text{logits}))) \tag{4.3}$$

Motor de entrenamiento (`Trainer`)

En la clase `Trainer` se encuentran recogidas un conjunto de funciones para hacer simple la realización del entrenamiento de la red, sin necesidad de entrar en detalles del uso de redes neuronales. Las funciones con las que cuenta son las siguientes:

`initialize()`: Ya sea retomando el entrenamiento desde un punto de control del modelo como desde cero (inicializando aleatoriamente los pesos de la red mediante Kaiming [52]), esta función prepara el entorno para entrenar al sistema en sucesivas épocas, tomando como entrada una base de datos de vídeos curada (`IMDB`). En este punto, será cuando se calculen las etiquetas de inferencia ideales utilizadas durante el cálculo del error. Dado que en las imágenes procesadas, el objeto a localizar se encontrará siempre en el centro, las etiquetas calculadas serán siempre iguales, con el valor +1 en un radio de R en torno al centro y con el valor 0 en el resto de casos. Como la red es totalmente convolucional, no existe riesgo de que aprenda a tener una tendencia hacia el centro de la imagen.

`trainEpoch()`: Esta función lleva a cabo una época (*epoch*) de entrenamiento de la red neuronal, consistente en el procesado de 53200 pares de imágenes en lotes (*batches*) de 8 pares. Estos pares de imágenes, se obtienen a partir del conjunto de entrenamiento de una base de datos de vídeos curada, y consisten en una imagen ejemplar y una imagen de área de búsqueda del mismo objeto, separadas a lo sumo 100 fotogramas. Para añadir más variedad al conjunto de entrenamiento, se llevará a cabo un proceso de aumento de datos (*data augmentation*) con cada par de imágenes, en el cual cada imagen, en un pequeño grado, podrá ser escalada, volteada horizontalmente, estirada o alterado algún color. Así, para cada lote de imágenes, se generarán unos mapas de valores de salida y se compararán con las *labels* consideradas correctas, corrigiendo el error mediante propagación hacia atrás (*backpropagation*). Se usará la técnica de Descenso de Gradiente Estocástico (SGD) [53] y se modificará el ratio de aprendizaje geométricamente entre las épocas, desde 10^{-2} hasta 10^{-5}. Al final de cada época, se guarda el punto de control del modelo que describe los pesos actuales de la red.

`validateEpoch()`: De forma muy similar a `trainEpoch()`, pero con un conjunto distinto de vídeos y sin aplicar aumento de datos ni ajuste de los pesos, esta función lleva a cabo una época de validación. Esta época, que normalmente se realiza después de una de entrenamiento, permite obtener un indicador relativo al progreso del aprendizaje de la red, en un entorno considerado más realista y nunca antes visto por el sistema.

`close()`: Esta función libera todos los recursos empleados por el entrenamiento.

El progreso de un entrenamiento de 50 épocas, visualizado mediante TensorBoard gracias al almacenamiento de eventos del modelo, es el mostrado en la imagen 4.11.

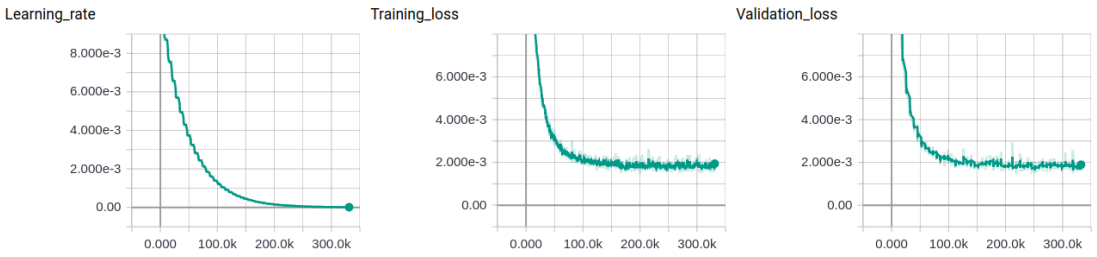


Figura 4.11: Progreso de un entrenamiento, visualizado mediante TensorBoard

4.4.2. Herramientas de curación

Este conjunto de utilidades permitirá la transformación de una base de datos de vídeos (en formato ILSVRC o STDdb) en otra base de datos de vídeos curada, cuyo contenido haga que se requiera el menor esfuerzo computacional posible durante el entrenamiento.

Dado que para el entrenamiento se necesitan imágenes ejemplares y de área de búsqueda centradas para cada objeto y cada fotograma, en un primer momento se barajó la idea de que la base de datos se encontrara conformada por estos pares de imágenes, tal y como lo hicieron originalmente Luca Bertinetto y Jack Valmadre para SiamFC. No obstante, esto supondría un gran gasto innecesario de espacio, pues la imagen ejemplar se encuentra contenida en la del área de búsqueda, y el aumento de datos mediante desplazamiento no se podría aplicar. Es por esto que se optó finalmente por almacenar una única imagen por cada

objeto en fotograma, la cual abarca algo más que el área de búsqueda (un 3 % más) para permitir el aumento de datos. Si bien esto requiere que el motor de entrenamiento realice computación extra para recortar las imágenes, se trata de un cálculo con un coste mínimo, que permite que la base de catos curada ocupe aproximadamente un 40 % menos de espacio en disco.

De esta forma, las herramientas de curación partirán de una base de datos de vídeos anotada y generarán un conjunto de imágenes estructurado, cuyos metadatos se almacenarán en formato pickle [54]. Este archivo pickle permitirá tratar la base de datos como si se tratara de una clase `IMDB` con la estructura mostrada en el diagrama 4.12:

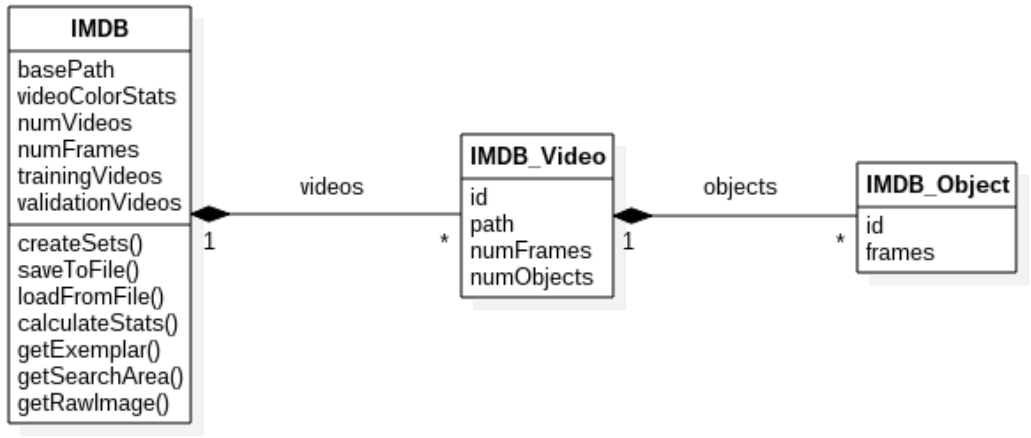


Figura 4.12: Diagrama de clases de la base de datos de vídeos curada

Así, el motor de entrenamiento podrá obtener fácilmente pares de imágenes ejemplares y de área de búsqueda, o imágenes parcialmente procesadas (*Raw*) para poder realizar aumento de datos.

4.4.3. Adaptación a objetos pequeños

Para adaptar el sistema al *tracking* de objetos pequeños, desde el punto de vista del entrenamiento del modelo, se realizó un ajuste de los pesos (*fine-tuning*) de la red. Esto se llevó a cabo a través de un entrenamiento con bajos valores de

aprendizaje, tomando como base los pesos aprendidos con ILSVRC y utilizando el conjunto de entrenamiento de STDdb de objetos pequeños.

De este modo, la red consiguió mejorar la extracción de características en objetos pequeños, logrando mejores resultados en los *benchmarks* realizados.

4.5. Módulo de evaluación

El módulo de evaluación es el encargado de permitir la evaluación objetiva de la calidad del modelo desarrollado mediante la obtención de métricas. De esta forma, su responsabilidad será la de adaptar el motor de inferencia para que pueda ser utilizado por *benchmarks* estándar de *tracking* visual de objetos.

Puesto que en la publicación original de SiamFC se mostraban los resultados obtenidos en VOT [31] y OTB [32], estos dos han sido los *benchmarks* a adaptar para la evaluación, para así facilitar las comparaciones.

Las clases adaptadoras creadas, ambas para *tracking* de objetos simple, son las mostradas en el siguiente diagrama:

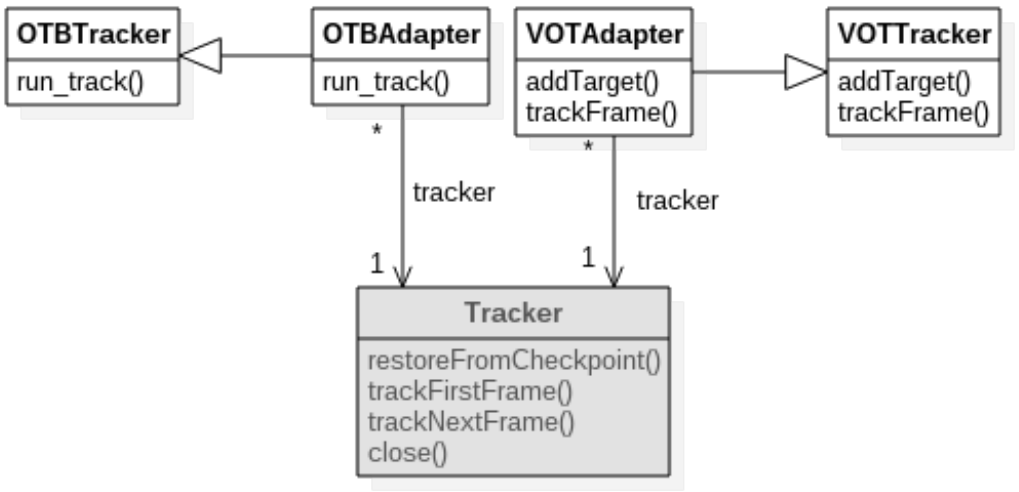


Figura 4.13: Diagrama de clases del módulo de evaluación

4.5.1. Evaluación mediante OTB

Pese a la falta de documentación existente relativa al *benchmark* OTB, su método de funcionamiento es reativamente simple: el `OTB Toolkit` ejecuta la función `run_track()` del `OTBTracker`, pasándole como entrada una lista de fotogramas y un cuadro delimitador inicial, y esperando como respuesta una lista de cuadros delimitadores. Para lograr este comportamiento, se ha creado la clase adaptadora `OTBAdapter`, que tomará parte en esta secuencia de acciones, tal y como se muestra en la Figura 4.14:

Tal y como se puede ver, no existe ningún tipo de supervisión por parte del `OTB Toolkit` durante el *tracking*, por lo que si el objeto es perdido durante los primeros fotogramas, el resultado del *benchmark* para dicha secuencia será adverso, al no ser posible el ajuste o recuperación.

4.5.2. Evaluación mediante VOT

Si bien los principios del `VOT Toolkit` son muy similares a los del *benchmark* OTB, existe la diferencia fundamental de que los fotogramas le son suministrados al *tracker* de 1 en 1. De esta forma, el *toolkit* irá analizando el progreso del *tracking* en tiempo real, y podrá hacer los ajustes necesarios para cambiar de vídeo fácilmente o resetear la secuencia.

Esta interacción entre `VOT Toolkit` y `Tracker` se logra gracias al uso del adaptador `VOTAdapter`, cuyo funcionamiento se encuentra recogido en la Figura 4.15:

Tal y como se puede ver, al existir supervisión por parte del `VOT Toolkit`, perder el objeto durante los primeros fotogramas del *tracking* (es decir, que la intersección entre la inferencia y el *groundtruth* sea nula) ya no condenará el resultado en dicha secuencia. Si bien esta pérdida será penalizada, se le ofrecerá al *tracker* la oportunidad de retomarlo donde lo dejó, partiendo de un objeto correctamente anotado.

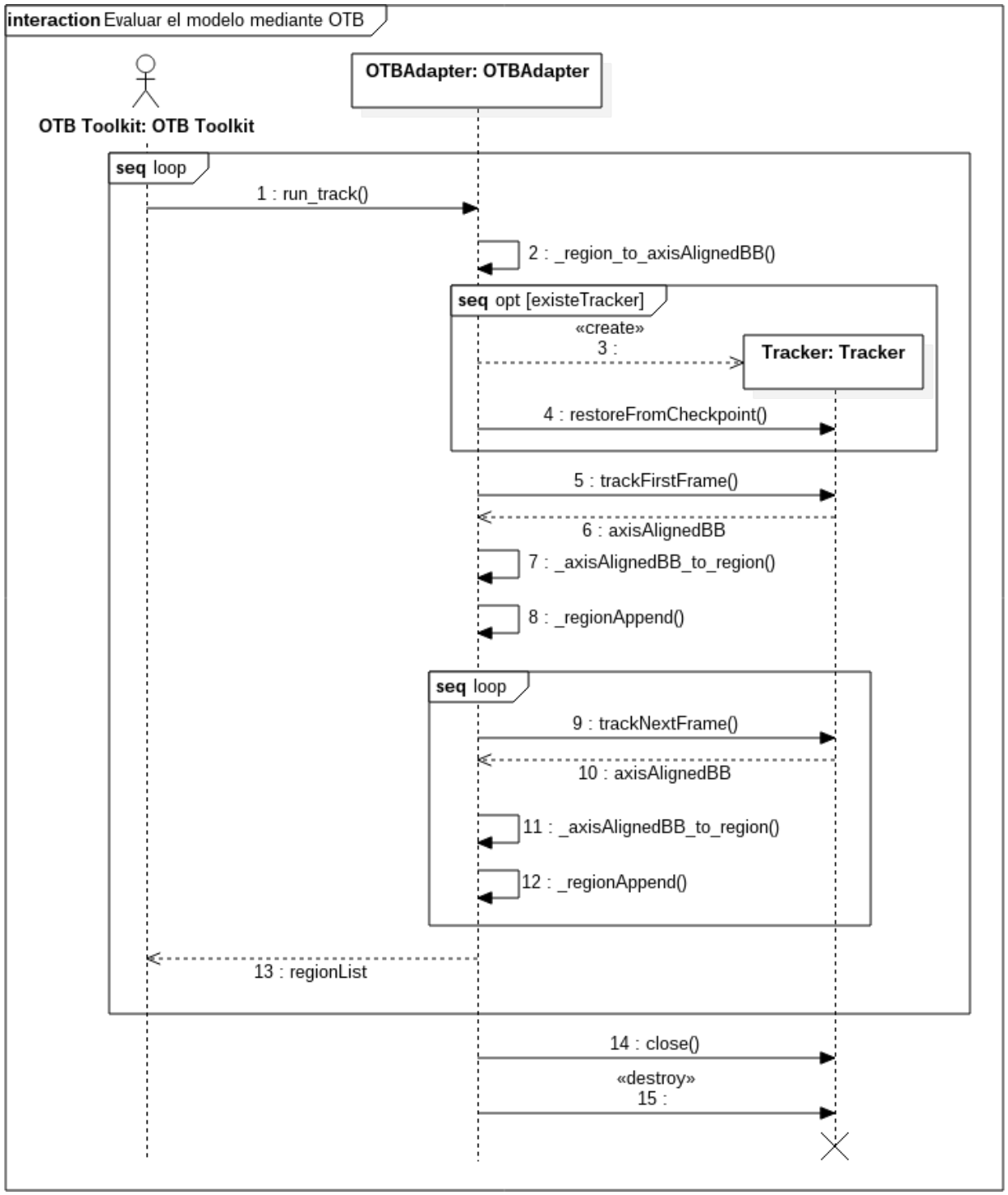


Figura 4.14: Diagrama de secuencia para la evaluación mediante OTB

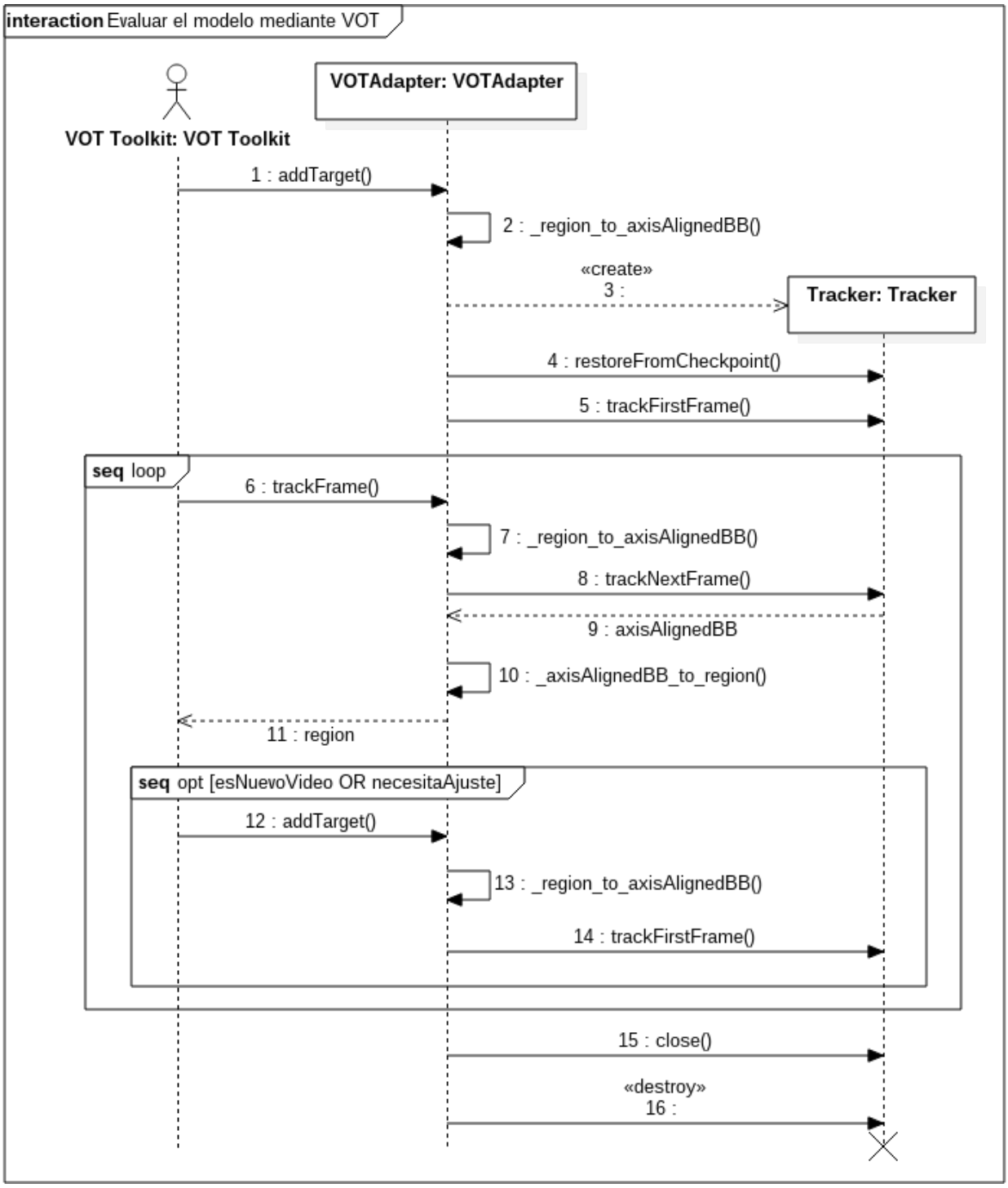


Figura 4.15: Diagrama de secuencia para la evaluación mediante VOT

4.5.3. Selección del mejor punto de control del modelo

Dado que con el uso de *benchmarks* es posible obtener resultados objetivos de la calidad del modelo, se hace sencillo el análisis de puntos de control del modelo para hallar aquel que ofrece los mejores resultados, ya que no siempre el último será el mejor. De esta forma, se ha desarrollado un script basado en OTB (puesto que es el *benchmark* con la menor duración) que recoge todos los puntos de control de un entrenamiento y los va inyectando en la red neuronal, para luego someterlos a prueba. Así, es posible obtener no sólo el mejor modelo del sistema para OTB, sino la evolución de la calidad del tracking para este *benchmark* durante el entrenamiento.

A continuación, en la Figura 4.16, se muestra el resultado de someter a prueba mediante OTB los 30 últimos puntos de control de un entrenamiento de 50 épocas:

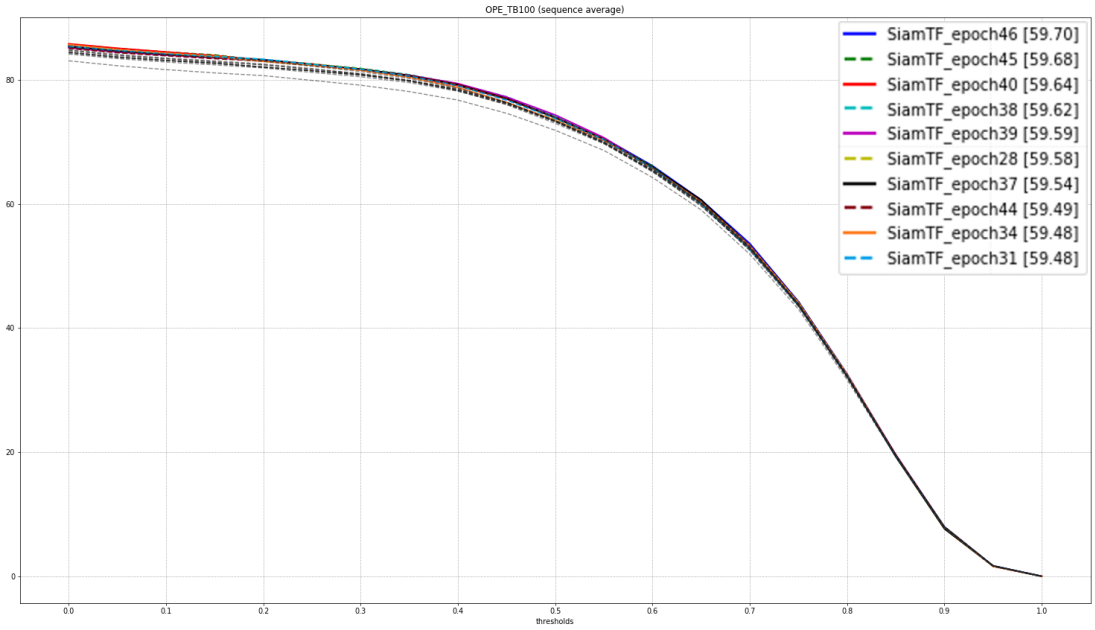


Figura 4.16: Análisis mediante OTB de los puntos de control generados durante un entrenamiento

4.6. Módulo de utilidades

El módulo de utilidades es el encargado de ofrecer una serie de herramientas y funciones que faciliten el uso de la aplicación por parte de un usuario, así como proporcionar un valioso conjunto de componentes de análisis y depuración de sistemas de *tracking*. Las utilidades desarrolladas, en función de su propósito, son las que se describen a continuación.

4.6.1. Visualización e interacción con el sistema

Las utilidades de visualización e interacción con el sistema permitirán a los usuarios obtener información de alto y bajo nivel relativa al proceso de *tracking*, así como definir de forma sencilla los objetos a seguir. Todo esto a través de una interfaz gráfica intuitiva y fácil de usar.

Con la finalidad de permitir una mayor compatibilidad entre dispositivos, estas utilidades pueden funcionar haciendo uso tanto de la librería Matplotlib [34] como de OpenCV [33]. Esta necesidad surgió al probar el sistema en múltiples plataformas, ya que algunas no contaban con el mismo software instalado.

Uno de los mayores retos de cara a la creación de interfaces elaboradas fue la limitación de OpenCV para la combinación de figuras, ya que toda unión de imágenes y/o gráficos debe realizarse manualmente, mediante la concatenación de matrices de píxeles. No obstante, a la hora de presentar resultados, resulta preferible sobre Matplotlib, ya que permite una mayor tasa de fotogramas por segundo.

Visualización de un único objeto

Cuando se está llevando a cabo el *tracking* de un único objeto, el usuario podrá indicarle al sistema si desea visualizar el proceso de forma simple o detallada.

La visualización simple del *tracking* consistirá únicamente en una ventana en la que se mostrará el vídeo, con un rectángulo indicando la posición actual del objetivo. Esta interfaz se encuentra representada en la Figura 4.17:

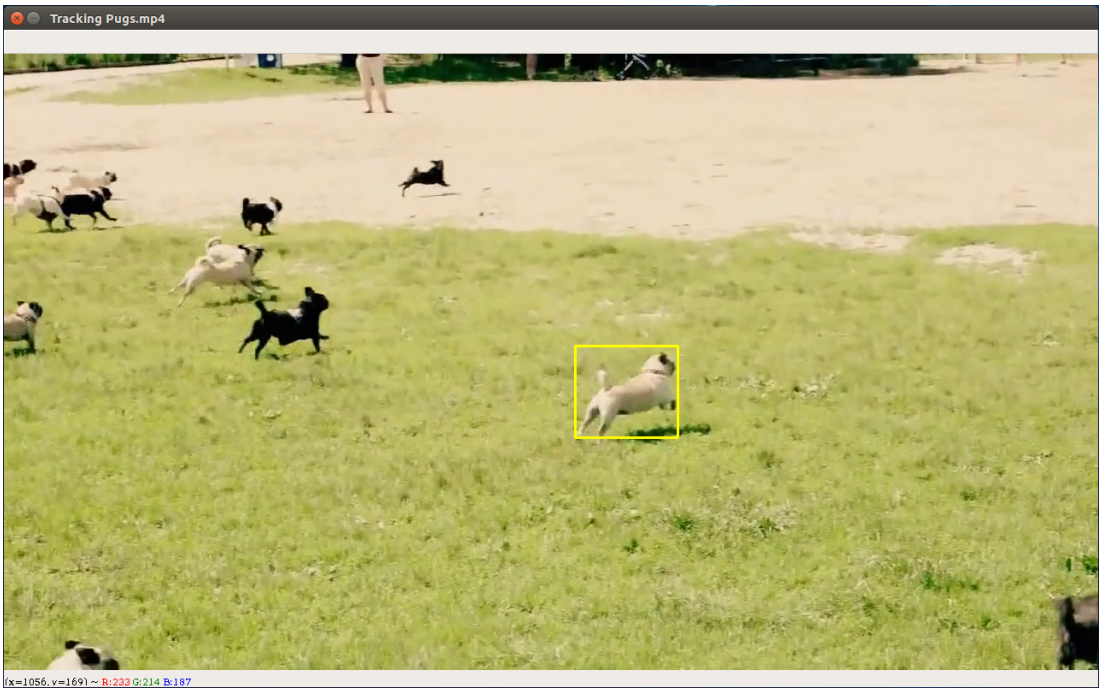


Figura 4.17: Interfaz de usuario simple para el *tracking* de un único objeto

En cuanto a la visualización detallada del *tracking*, esta fue diseñada teniendo en mente a usuarios experimentados en el mundo de la visión por computador, así como a desarrolladores familiarizados con la arquitectura de la red y el seguimiento de objetos.

Puesto que el objetivo de esta interfaz era albergar una mayor cantidad de información, se hizo necesaria la realización del *mockup* mostrado en la Figura 4.18 para organizar la distribución de los datos:

El aspecto final de la interfaz de usuario detallada, para *tracking* simple, es el mostrado en la Figura 4.19:

Durante la visualización del *tracking* de un único objeto, el usuario podrá finalizar el programa de forma controlada pulsando la tecla de `Escape` o mediante el icono de cerrar de la ventana. También podrá pausar y reanudar el proceso mediante la `Barra espaciadora` y variar la cantidad de *zoom* con la rueda del ratón.

No se permitirá avanzar o retroceder el vídeo dadas las características del modelo de *tracking*. Si el usuario quisiera inspeccionar con más detenimiento el seguimien-

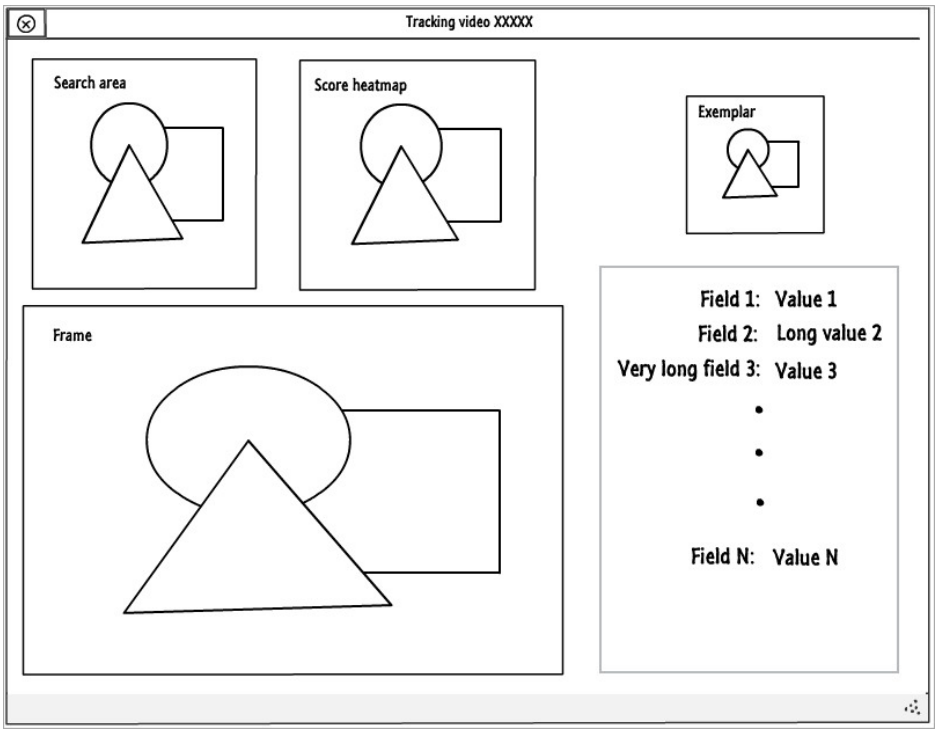


Figura 4.18: *Mockup* de la interfaz detallada para el *tracking* de un único objeto

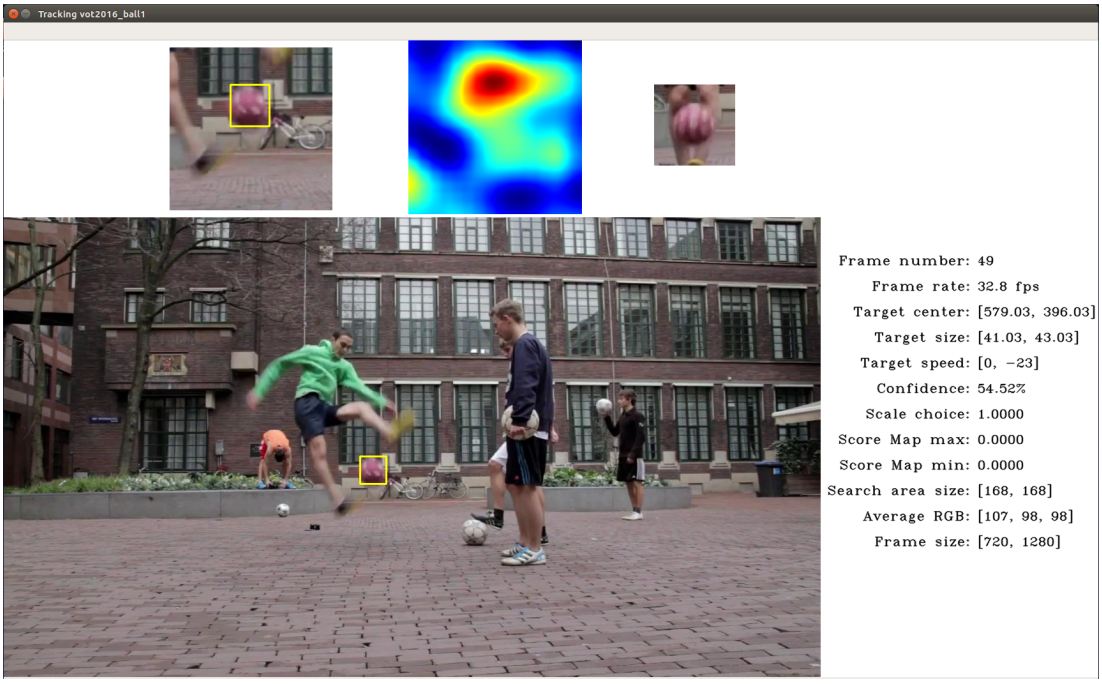


Figura 4.19: Interfaz de usuario detallada para el *tracking* de un único objeto

to fotograma a fotograma, tendrá que iniciar la aplicación indicando que desea que se guarde un resultado visual del tracking (en imágenes de cada fotograma o en un archivo de vídeo convencional).

Visualización de múltiples objetos

Al igual que en el caso de un único objeto, el usuario podrá escoger si desea una visualización simple o detallada del proceso. Dado que resulta inviable incluir toda la información de cada objeto en la versión detallada, se ha optado por mostrar lo esencial para el correcto análisis de la red, tal y como se puede ver en el *mockup* de la Figura 4.20:

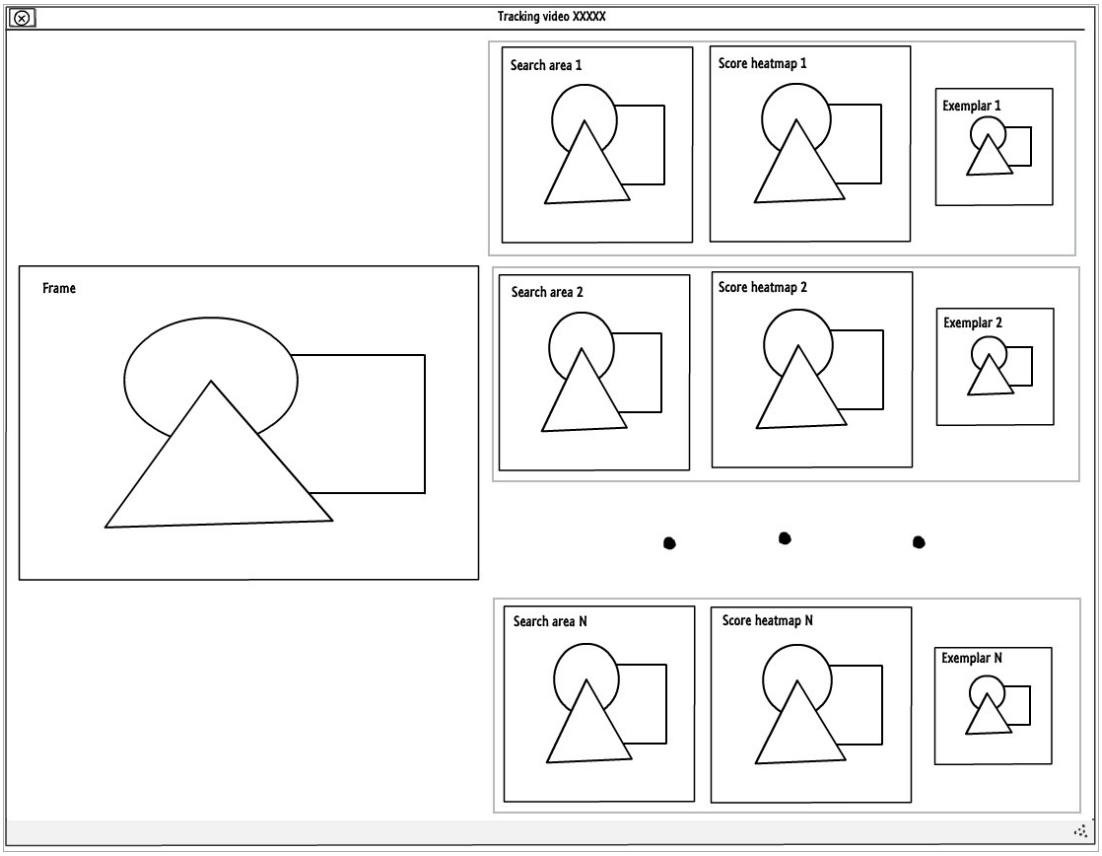


Figura 4.20: *Mockup* de la interfaz de usuario detallada para el *tracking* de múltiples objetos

El aspecto final de la interfaz es el mostrado a continuación, en la Figura 4.21:

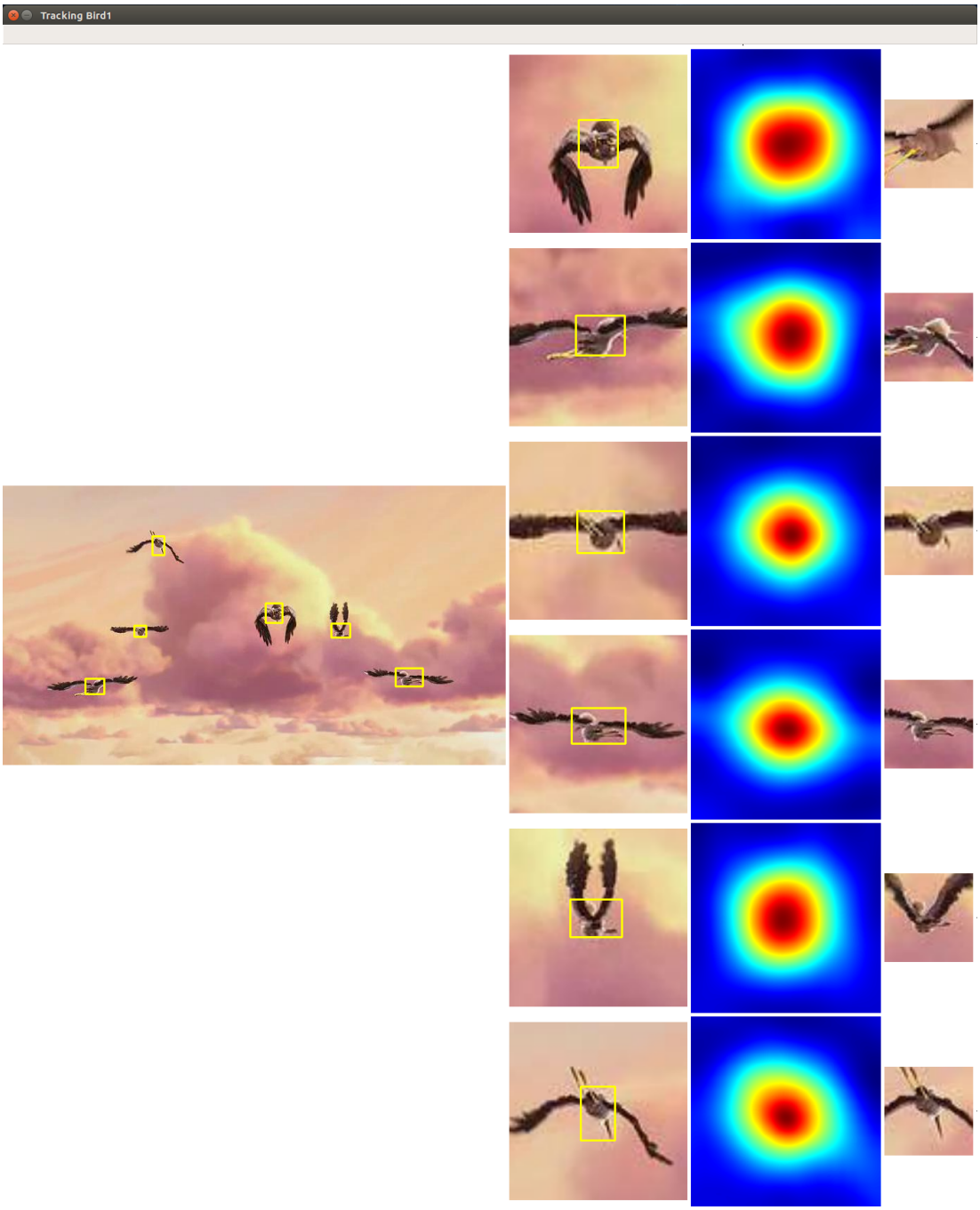


Figura 4.21: Interfaz de usuario detallada para el *tracking* de múltiples objetos

Esta interfaz posee los mismos controles y funcionalidades que la de objetos simples, pero incorpora la posibilidad de añadir o eliminar objetos del *tracker* mediante las teclas + y −.

Delimitación de objetos

Con la finalidad de simplificar el uso de la aplicación por parte de usuarios, así como facilitar las tareas de depuración, se ha creado una utilidad gráfica para el delimitado de objetos mediante cuadros delimitadores alineados con los ejes.

Al comienzo del *tracking* si no existe *groundtruth*, o al añadir un objeto nuevo durante el vídeo, el usuario de la aplicación interactuará con la herramienta mostrada en la Figura 4.22. Este útil le permitirá etiquetar el primer fotograma en el que aparece un objeto mediante un sistema intuitivo de pinchar y arrastrar, y la tecla `Enter` para confirmar.

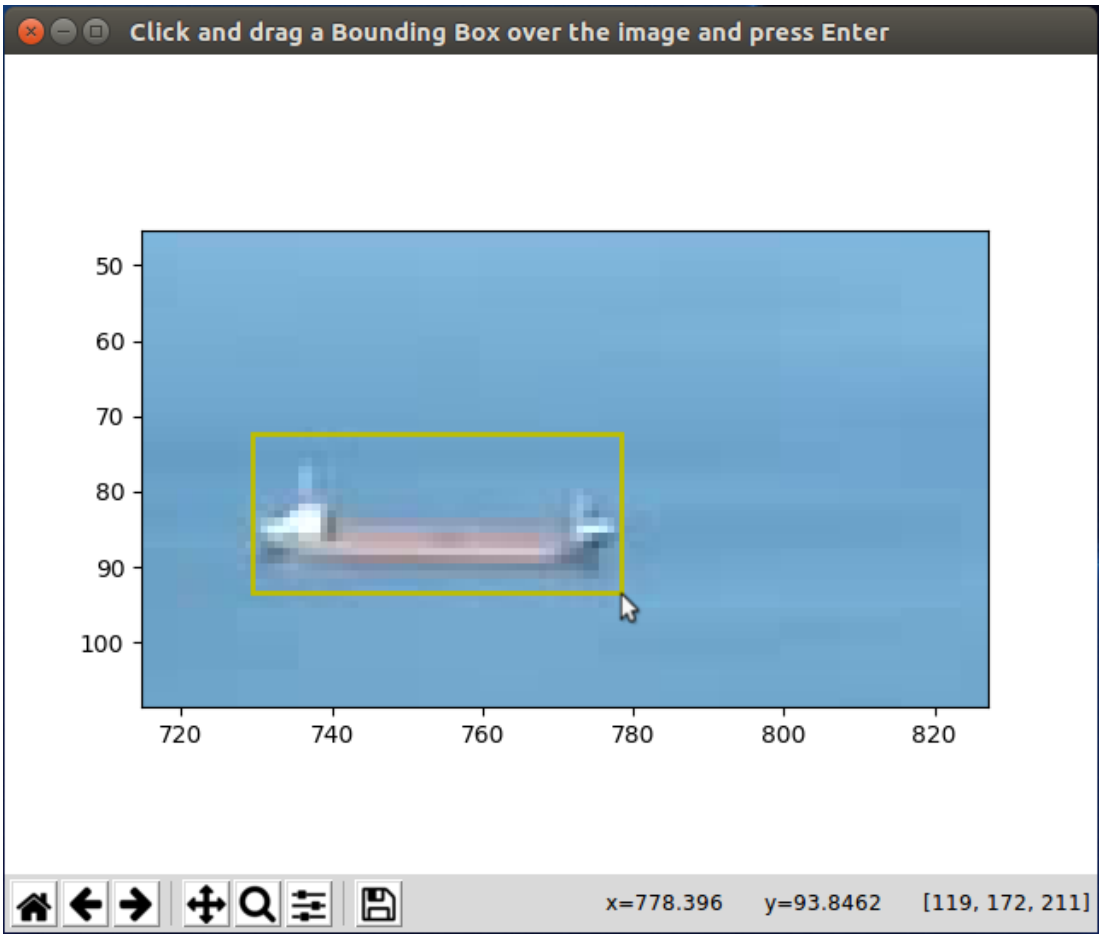


Figura 4.22: Interfaz de usuario para la delimitación de objetoss

Lo verdaderamente útil de esta interfaz es que permite no sólo la delimitación

de un objeto sobre un fotograma, sino también el etiquetado de objetos sobre un vídeo en movimiento, procedente de una fuente en *streaming* que no pueda ser pausada.

4.6.2. Conversión de vídeos y bases de datos

Para facilitar la visualización de los vídeos de *benchmarks* por parte de los usuarios y desarrolladores y para simplificar la incorporación de nuevos vídeos a estos últimos, se han desarrollado una serie de herramientas de conversión.

Conversión de vídeos

Relativo a la conversión de formatos de vídeo, se han desarrollado una serie de scripts para la transformación de vídeos en `.mp4` o `.avi` a formato OTB o VOT, para simplificar su inclusión en estos *benchmarks*.

También se hace posible la conversión inversa, para que los contenidos de estas bases de datos de vídeos puedan ser fácilmente interpretados por humanos. Actualmente, el script de conversión desde VOT y OTB permite no sólo obtener el vídeo original, sino también incorporar la representación visual del *groundtruth*.

Conversión de bases de datos de vídeos

En cuanto a las utilidades de conversión para bases de datos al completo, estas permiten transformar una base de datos anotada en formato STDdb a una base de datos OTB o VOT para *benchmarking*.

También se ha desarrollado una utilidad de conversión desde ILSVRC a ViTBAT (formato que es comúnmente utilizado como el paso previo a STDdb), que además permite extraer únicamente los vídeos que contienen objetos pequeños. Esto ha sido llevado a cabo como aportación a STDdb, pues el ámbito del *tracking* de objetos pequeños aún no cuenta con grandes bases de datos etiquetadas.

4.6.3. Almacenamiento de resultados de *tracking*

Con la finalidad de permitir un fácil análisis del proceso de *tracking* y para poder contar con un sólido punto de referencia para el etiquetado de vídeos, la aplicación puede guardar las anotaciones realizadas, fotograma a fotograma, en formato de vídeo o en formato *groundtruth* VOT. De esta forma, un usuario podrá analizar la información simple o detallada del visualizador, pero con las facilidades y herramientas que ofrece un reproductor de vídeos convencional.

4.6.4. DAO de vídeos

En contraste con las utilidades de conversión de vídeos anteriormente descritas, la finalidad de este objeto de acceso a datos de vídeos es la de proporcionar, de forma transparente para la aplicación, una interfaz común para el manejo de vídeos en distintos formatos. De esta forma, este DAO permite la obtención del siguiente fotograma y *groundtruth* (en caso de existir) de archivos de vídeo en formato `.mp4`, `.avi`, OTB o VOT y de fuentes de vídeo en streaming como dispositivos de grabación locales y cámaras IP remotas.

4.7. Dockerización de la aplicación

Con la finalidad de facilitar la portabilidad de la aplicación y minimizar su necesidad de dependencias para demostración y test en diversos dispositivos, se propuso la dockerización de la misma. Es decir, su inclusión en un contenedor Docker [55] para que su despliegue sea inmediato y su utilización tan sencilla como hacer uso de un ejecutable con las mismas opciones que el programa original.

Actualmente, el único subsistema que se encuentra integrado dentro de Docker es el de inferencia, dado que constituye la amplia mayoría de los casos de uso que se llevarán a cabo fuera del entorno de desarrollo.

Para poder aprovechar las ventajas que ofrece la computación GPU, deberá utilizarse el *plugin* NVIDIA Docker [37], el cual permite que los contenedores sean

conocedores de los detalles hardware relativos a las tarjetas gráficas NVIDIA instaladas en la máquina anfitrión.

4.7.1. Creación de la imagen

Haciendo uso del `Dockerfile` incluido en el respositorio del proyecto, el cual se basa en la imagen oficial de TensorFlow, la creación de la imagen Docker para el *tracker* es realmente sencilla. Únicamente se precisa el módulo de inferencia de la aplicación junto a sus dependencias y conexión a Internet, para la descarga de los paquetes necesarios.

La creación de esta imagen no es instantánea, ya que existen librerías como OpenCV que deben ser instaladas a partir de sus ficheros fuente, para poder contar con todas las funcionalidades que ofrecen, por lo que se ralentiza el proceso.

4.7.2. Uso de la imagen

Para simplificar el uso de la imagen por parte del usuario, se ha creado un script que permite la instanciación y utilización de un contenedor de forma transparente, como si se tratara de la aplicación original. Al finalizar el *tracking*, el contenedor creado es eliminado automáticamente.

Este script adaptador lleva a cabo un análisis gramatical de los argumentos introducidos, para así poder crear los volúmenes necesarios para el acceso a los ficheros del anfitrión por parte del invitado. También se establecen volúmenes para permitir la escritura de invitado a anfitrión, y son mapeados los dispositivos necesarios para el acceso a cámaras y al X Server, para la obtención de vídeo en *streaming* y la interacción con la interfaz gráfica, respectivamente.

Capítulo 5

Pruebas y validación

En este capítulo se detallan las pruebas realizadas para verificar el correcto funcionamiento del sistema y el cumplimiento de todos los requisitos determinados. Es esencial que los criterios de validación establecidos sean cuantizables, para que su cumplimiento pueda ser establecido de forma objetiva.

5.1. Pruebas de validación de requisitos

El objetivo de estas pruebas del software será el de verificar la completitud del producto entregado, comprobando que todos los requisitos planteados han sido cubiertos. No obstante, no tienen por qué garantizar el funcionamiento correcto de la aplicación en el 100 % de los casos, pues resulta impracticable la prueba exhaustiva.

Las pruebas planteadas han sido directamente extraidas de los requisitos establecidos y de los criterios de validación impuestos por el cliente. Su definición y resultados son los descritos a continuación:

PV.01	**Adición de objetos**
Requisitos involucrados	CU.01
Descripción	Un Sistema de *tracking* de alto nivel es capaz de añadir un objeto al *tracker*, de forma que en el siguiente fotograma este puede ser seguido.
Resultado esperado	El sistema pasa a almacenar la imagen ejemplar y características del objeto, así como su ID y cuadro delimitador.
Estado	Superada

PV.02	**Eliminación de objetos**
Requisitos involucrados	CU.02
Descripción	Un Sistema de *tracking* de alto nivel es capaz de eliminar un objeto del *tracker*, de forma que en el siguiente fotograma este no sea seguido.
Resultado esperado	El sistema deja de almacenar la imagen ejemplar y características del objeto, así como su ID y cuadro delimitador.
Estado	Superada

PV.03	**Modificación de objetos**
Requisitos involucrados	CU.03
Descripción	Un Sistema de *tracking* de alto nivel es capaz de modificar el estado de un objeto del *tracker*, de forma que en el siguiente fotograma, el *tracking* se vea afectado por esta modificación.
Resultado esperado	El sistema pasa a almacenar un nuevo cuadro delimitador para el objeto, y/o una nueva imagen ejemplar, junto a sus nuevas características.
Estado	Superada

PV.04	**Inferencia de objetos simple**
Requisitos involucrados	CU.04
Descripción	Al procesar un nuevo fotograma, el sistema devuelve un nuevo cuadro delimitador para un objeto seguido, de forma que el 80 % de los píxeles de dicho objeto se encuentran dentro del cuadro delimitador. El cuadro delimitador no podrá tener un tamaño un 50 % superior al mínimo necesario para contener al objeto completo.
Resultado esperado	El sistema pasa a almacenar un cuadro delimitador actualizado para el objeto, el cual comprende al menos el 80 % de los píxeles del mismo y no es un 50 % mayor que el tamaño mínimo necesario.
Estado	Superada

PV.05	**Inferencia de objetos múltiple**
Requisitos involucrados	CU.04, RN.03
Descripción	Al procesar un nuevo fotograma y existir más de un objeto a seguir en el sistema, se devuelven una serie de cuadros delimitadores que cumplen el **PV.04 - Inferencia de objetos simple** para cada objeto.
Resultado esperado	El sistema pasa a almacenar uno cuadros delimitadores actualizados para los objetos, los cuales comprenden al menos el 80 % de los píxeles del objeto y no son un 50 % mayores que los tamaños mínimos necesarios.
Estado	Superada

PV.06	**Almacenamiento visual de la inferencia**
Requisitos involucrados	CU.05
Descripción	Al realizar inferencia, el sistema es capaz de almacenar un vídeo en el que se incluyan los cuadros delimitadores calculados.
Resultado esperado	Tras una inferencia, se obtiene un vídeo que contiene el resultado de la misma
Estado	Superada

PV.07	**Almacenamiento VOT de la inferencia**
Requisitos involucrados	CU.05
Descripción	Al realizar inferencia, el sistema es capaz de almacenar un *groundtruth* VOT en el que se incluyen los cuadros delimitadores calculados, para un único objeto.
Resultado esperado	Tras una inferencia, se obtiene un *groundtruth* VOT que contiene el resultado de la misma.
Estado	Superada

PV.08	**Delimitación visual**
Requisitos involucrados	CU.06
Descripción	El sistema le permite al usuario el establecimiento de un cuadro delimitador mediante interfaz gráfica.
Resultado esperado	Tras la interacción del usuario, se obtienen una serie de coordenadas y dimensiones que se corresponden con el cuadro delimitador trazado en pantalla.
Estado	Superada

PV.09	**Visualización simple del *tracking* de un único objeto**
Requisitos involucrados	CU.07, RN.09
Descripción	Al inferenciar en un fotograma un único objeto, el sistema le presenta al usuario una interfaz gráfica con el resultado del *tracking*.
Resultado esperado	Tras una inferencia simple, se muestra por pantalla el fotograma actual con un rectángulo delimitando al objeto seguido.
Estado	Superada

PV.10	**Visualización simple del *tracking* de múltiples objetos**
Requisitos involucrados	CU.07, RN.09
Descripción	Al inferenciar en un fotograma múltiples objetos, el sistema le presenta al usuario una interfaz gráfica con el resultado del *tracking*.
Resultado esperado	Tras una inferencia múltiple, se muestra por pantalla el fotograma actual con un rectángulo delimitando a los objetos seguidos.
Estado	Superada

PV.11	**Visualización detallada del *tracking* de un único objeto**
Requisitos involucrados	CU.07, RN.09
Descripción	Al inferenciar en un fotograma un único objeto, el sistema le presenta al usuario una interfaz gráfica con el resultado detallado del *tracking*.
Resultado esperado	Tras una inferencia simple, se muestra por pantalla el fotograma actual con un rectángulo delimitando al objeto seguido y la imagen ejemplar, área de búsqueda y mapa de valores considerados.
Estado	Superada

PV.12	**Visualización detallada del *tracking* de múltiples objetos**
Requisitos involucrados	CU.07, RN.09
Descripción	Al inferenciar en un fotograma múltiples objetos, el sistema le presenta al usuario una interfaz gráfica con el resultado detallado del *tracking*.
Resultado esperado	Tras una inferencia múltiple, se muestra por pantalla el fotograma actual con un rectángulo delimitando a los objetos seguidos, y una imagen ejemplar, área de búsqueda y mapa de valores por cada objeto seguido.
Estado	Superada

PV.13	**Consulta de ayuda**
Requisitos involucrados	CU.08, RN.08, RN.10
Descripción	El usuario obtiene ayuda relativa a la utilización de una herramienta.
Resultado esperado	Tras utilizar incorrectamente una herramienta o al ejecutarla con el argumento `-h`, se muestra por pantalla una guía indicando todas las posibilidades de uso de la misma.
Estado	Superada

PV.14	**Conversión de VOT a `.mp4`**
Requisitos involucrados	CU.09
Descripción	El usuario solicita la conversión de un vídeo en formato VOT a `.mp4` y el sistema la realiza.
Resultado esperado	Tras la conversión, el usuario dispone de un vídeo en formato `.mp4` con los fotogramas mostrados en VOT, y los cuadros delimitadores descritos en su *groundtruth*.
Estado	Superada

PV.15	**Extracción de vídeos pequeños**
Requisitos involucrados	CU.10

Descripción	Partiendo de una base de datos ILSVRC, el sistema es capaz de extraer sus vídeos con objetos pequeños en formato ViTBAT.
Resultado esperado	Tras la extracción, se obtiene una base de datos de vídeos en formato ViTBAT que contiene exclusivamente objetos pequeños.
Estado	Superada

PV.16	**Conversión de STDdb a VOT**
Requisitos involucrados	CU.11
Descripción	El usuario solicita la conversión de una base de datos en formato STDdb a VOT y el sistema la realiza.
Resultado esperado	Tras la conversión, el usuario dispone de una base de datos en formato VOT, que contiene la misma información que la base de datos STDdb original.
Estado	Superada

PV.17	**Conversión de STDdb a OTB**
Requisitos involucrados	CU.11
Descripción	El usuario solicita la conversión de una base de datos en formato STDdb a OTB y el sistema la realiza.

Resultado esperado	Tras la conversión, el usuario dispone de una base de datos en formato OTB, que contiene la misma información que la base de datos STDdb original.
Estado	Superada

PV.18	**Almacenamiento de eventos**
Requisitos involucrados	CU.12, RN.12
Descripción	Durante la inferencia, evaluación o entrenamiento del sistema, se almacenan los eventos del modelo generados por la red neuronal.
Resultado esperado	Durante una inferencia, evaluación o entrenamiento, es posible visualizar la actividad de la red mediante TensorBoard.
Estado	Superada

PV.19	**Curación de ILSVRC**
Requisitos involucrados	CU.13
Descripción	El usuario solicita la curación de una base de datos ILSVRC, y el sistema la adapta para el entrenamiento.
Resultado esperado	Tras la curación, el sistema es capaz de entrenar con los mismos vídeos de la base de datos ILSVRC, pues la base de datos curada contiene, para cada objeto y fotograma, un archivo con su imagen ejemplar y área de búsqueda.

Estado	Superada

PV.20	**Actualización de bases de datos curadas**
Requisitos involucrados	CU.14
Descripción	El usuario solicita la actualización de una base de datos curada, y el sistema adapta sus metadatos.
Resultado esperado	Tras la modificación manual de una base de datos curada y su actualización, se dispone de una base de datos, apta para el entrenamiento, cuyos metadatos contemplan todas las imágenes almacenadas.
Estado	Superada

PV.21	**Entrenamiento del sistema**
Requisitos involucrados	CU.15
Descripción	El sistema aprende a llevar a cabo inferencias correctas, tomando como ejemplo las imágenes de una base de datos curada.
Resultado esperado	Tras una sesión de entrenamiento, el sistema es capaz de realizar inferencias con al menos un 50 % de corrección.
Estado	Superada

PV.22	**Evaluación mediante VOT**
Requisitos involucrados	CU.16
Descripción	La calidad de inferencia del sistema puede ser evaluada mediante el *benchmark* VOT.
Resultado esperado	Tras la evaluación, se obtienen las métricas reales del sistema, según VOT.
Estado	Superada

PV.23	**Evaluación mediante OTB**
Requisitos involucrados	CU.17
Descripción	La calidad de inferencia del sistema puede ser evaluada mediante el *benchmark* OTB.
Resultado esperado	Tras la evaluación, se obtienen las métricas reales del sistema, según OTB.
Estado	Superada

PV.24	**Obtención del mejor modelo OTB**
Requisitos involucrados	CU.18
Descripción	El usuario es capaz de obtener el mejor punto de control del modelo para OTB, ejecutando un único script.

Resultado esperado	Tras la ejecución del script, se le muestra por pantalla al usuario el mejor punto de control del modelo para OTB y su puntuación.
Estado	Superada

PV.25	**Inferencia en tiempo real**
Requisitos involucrados	RN.01
Descripción	El sistema deberá ser capaz de realizar inferencias de objetos únicos en vídeos de alta resolución a una velocidad de al menos 30 imágenes por segundo.
Resultado esperado	Realizando la inferencia en el servidor de computación GPGPU, el contador de imágenes por segundo indica que la velocidad media fue de al menos 30 fps.
Estado	Superada

PV.26	**Espacio en memoria compatible con la NVIDIA Jetson TX2**
Requisitos involucrados	RN.02
Descripción	El sistema deberá ser capaz de ser ejecutado en un módulo NVIDIA Jetson TX2 sin hacer uso de más del 50 % de la memoria.

Resultado esperado	Durante la ejecución del programa, el uso del comando `nvidia-smi` indica que el uso de GPU está por debajo del 50 %.
Estado	Superada

PV.27	**Rendimiento equiparable a SiamFC**
Requisitos involucrados	RN.03
Descripción	La calidad del sistema en los *benchmarks* establecidos es equiparable a la red SiamFC.
Resultado esperado	Tras la ejecución de los *benchmarks*, el área bajo la curva en OTB-13 es de al menos 57,67 puntos, la *accuracy*, *robustness* y *average expected overlap* en VOT-14 son de al menos 59,30, 82,18 y 26,06 puntos, respectivamente, y la *accuracy*, *robustness* y *average expected overlap* en VOT-15 son de al menos 50,35, 79,92 y 26,61 puntos, respectivamente.
Estado	Superada

PV.28	**Inferencia eficiente en múltiples objetos**
Requisitos involucrados	RN.04
Descripción	El sistema es capaz de realizar inferencias de múltiples objetos simultáneos sin replicar el modelo utilizado.
Resultado esperado	No existe más de una instancia de una red neuronal o motor de inferencia al realizar inferencia para múltiples objetos.

Estado	Superada

PV.29	**Compatibilidad con formatos de vídeo variados**
Requisitos involucrados	RN.05
Descripción	Es posible llevar a cabo inferencias sobre sobre vídeos en formato VOT, `.mp4` y *webcams*, sin necesidad de que el usuario lleve a cabo ninguna conversión.
Resultado esperado	Resulta aparentemente equivalente, de cara al usuario, realizar una inferencia sobre un vídeo en formato VOT, `.mp4` o una *webcam*.
Estado	Superada

PV.30	**Compatibilidad con objetos pequeños**
Requisitos involucrados	RN.06
Descripción	El sistema permite el tracking de objetos pequeños.
Resultado esperado	Se realiza una inferencia correcta del objeto pequeño en más del 75 % de los fotogramas del vídeo.
Estado	Superada

PV.31	**Facilidad de despliegue**
Requisitos involucrados	RN.07
Descripción	El sistema ofrece un mecanismo de despliegue simplificado, que minimice el número de dependencias necesarias.
Resultado esperado	Es posible distribuir el sistema mediante una imagen Docker.
Estado	Superada

PV.32	**Facilidad de adaptación**
Requisitos involucrados	RN.11
Descripción	La modificación de hiperparámetros y de la estructura de la red resulta sencilla.
Resultado esperado	Los parámetros del sistema residen en un único fichero, y las ramas de extracción de caracterísricas siguen interfaces.
Estado	Superada

5.2. Pruebas exploratorias

Este conjunto de pruebas, basadas en el conocimiento propio del *tester* sobre el software, buscarán la aparición de errores o comportamientos anómalos de la aplicación en situaciones poco comunes y que normalmente podrían pasar desapercibidas.

PE.01	**Objeto de área 0**
Descripción	Se le suministra a la aplicación un objeto cuyo cuadro delimitador tiene una área de 0.
Resultado esperado	La aplicación indica el error de froma controlada y no admite el objeto.
Estado	Superada

PE.02	**Objeto fuera del fotograma**
Descripción	Se fuerza una situación en la que el objeto a seguir se encuentra totalmente fuera del fotograma.
Resultado esperado	La imagen del área de búsqueda se encuentra conformada por un color sólido, que resulta ser la media de colores del fotograma.
Estado	Superada

PE.03	**Vídeo inexistente**
Descripción	Se intenta realizar tracking sobre un vídeo inexistente.
Resultado esperado	La aplicación indica el error y finaliza de forma controlada.
Estado	Superada

PE.04	**Interfaz gráfica en Docker desde SSH**
Descripción	Al ejecutar el contenedor de la aplicación en un servidor mediante SSH, es posible visualizar gráficamente el resultado del *tracking*.
Resultado esperado	Al ejecutar el contenedor mediante el script creado, la aplicación se visualiza de la misma manera que en local.
Estado	Superada

5.3. Criterios de aceptación del producto

Pese a que estos criterios de aceptación ya se encuentran cubiertos por las pruebas anteriormente mostradas, resulta conveniente hacer explícita su superación individual, para que no quepa duda de la aceptación del sistema.

CA.01	**Inferencia correcta**
Descripción	El sistema es capaz de inferir la posición de un objeto en más del 80 % de los fotogramas de un vídeo de ejemplo propuesto por los tutores del proyecto.
Estado	Superado

CA.02	**Modularidad**
Descripción	El código fuente del sistema se encuentra organizado de forma modular y haciendo uso de interfaces.
Estado	Superado

CA.03	**Tiempo real**
Descripción	El sistema es capaz de procesar fotogramas en tiempo real (al menos 30 fotogramas por segundo).
Estado	Superado

CA.04	**Múltiples objetos**
Descripción	El sistema puede hacer tracking de más de un objeto de forma simultánea.
Estado	Superado

CA.05	**Objetos pequeños**
Descripción	El sistema es capaz de inferir la posición de un objeto pequeño en más del 75 % de los fotogramas de un vídeo de ejemplo propuesto por los tutores del proyecto.
Estado	Superado

CA.06	**Compatibilidad con ILSVRC**
Descripción	El sistema puede ser entrenado a partir de una base de datos ImageNet Large Scale Visual Recognition Challenge.
Estado	Superado

CA.07	**Interfaz gráfica**
Descripción	Es posible visualizar de forma gráfica y en tiempo real el progreso de una inferencia.
Estado	Superado

CA.08	**Compatibilidad con TensorBoard**
Descripción	Es posible visualizar de forma gráfica y en tiempo real el progreso de un entrenamiento.
Estado	Superado

CA.09	**Compatibilidad con VOT**
Descripción	El sistema puede ser evaluado mediante Visual Object Tracking Challenge.
Estado	Superado

CA.10	**Compatibilidad con OTB**
Descripción	El sistema puede ser evaluado mediante Online Object Tracking Benchmark.
Estado	Superado

Capítulo 6

Resultados en los *benchmarks* propuestos

Para facilitar la comparación del sistema desarrollado con su referencia, SiamFC, y otros *trackers* de índole similar, en este capítulo se muestran los resultados obtenidos en los benchmarks Online Object Tracking Benchmark y Visual Object Tracking Challenge.

Dado que se encuentra basado en SiamFC y está implementado en TensorFlow, el identificador escogido para las gráficas comparativas ha sido `SiamTF`.

6.1. OTB 2013

El *benchmark* OTB-13 [32] se compone de 100 vídeos distintos de situaciones cotidianas. Para el cálculo de la calidad de un modelo, 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 inferencia y el *groundtruth* se encuentra sobre un determinado umbral. Los *trackers* son comparados en función del área bajo la curva de porcentajes de acierto para diferentes valores de este umbral.

Los resultados obtenidos para el OPE (*one pass evaluation*) son los mostrados en la gráfica 6.1, logrando un área bajo la curva de **59,70** puntos. Los resultados

de SiamFC para esta misma prueba son de 60,75 puntos, por lo que existe menos de un 2 % de diferencia, lo cual se considera aceptable. Además del modelo desarrollado, `SiamTF`, en esta gráfica también se muestran otros *trackers* presentados a OTB-13.

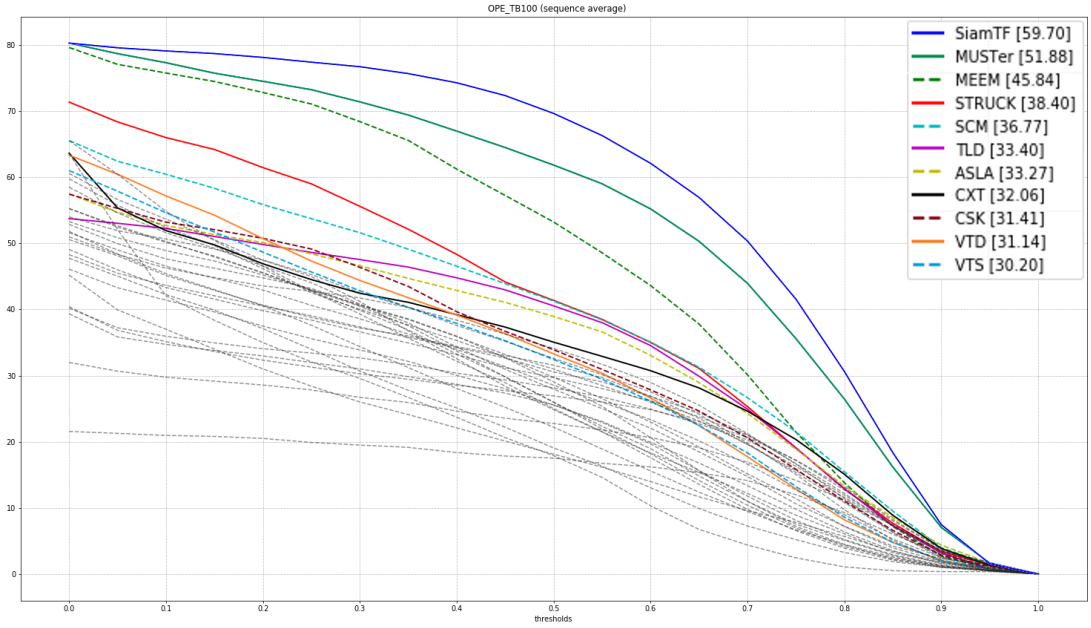


Figura 6.1: Gráfico OTB-13 de los resultados para OPE (*one pass evaluation*).

6.2. VOT 2014

El *benchmark* VOT-14 [31] evalúa los distintos *trackers* en una serie de secuencias escogidas a partir de un conjunto de 365 vídeos. 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 (IoU) entre la inferencia 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 gráfica 6.2, logrando unos valores de **64,57** y **87,02** puntos, respectivamente. Los resultados de SiamFC para esta misma prueba son de 62,42 y 86,51 puntos, por lo que existe en torno a un 1,5 % de mejora en promedio, lo cual se considera un éxito. Además del modelo desarrollado, `SiamTF`, en esta gráfica también se muestran otros *trackers* presentados a VOT-14.

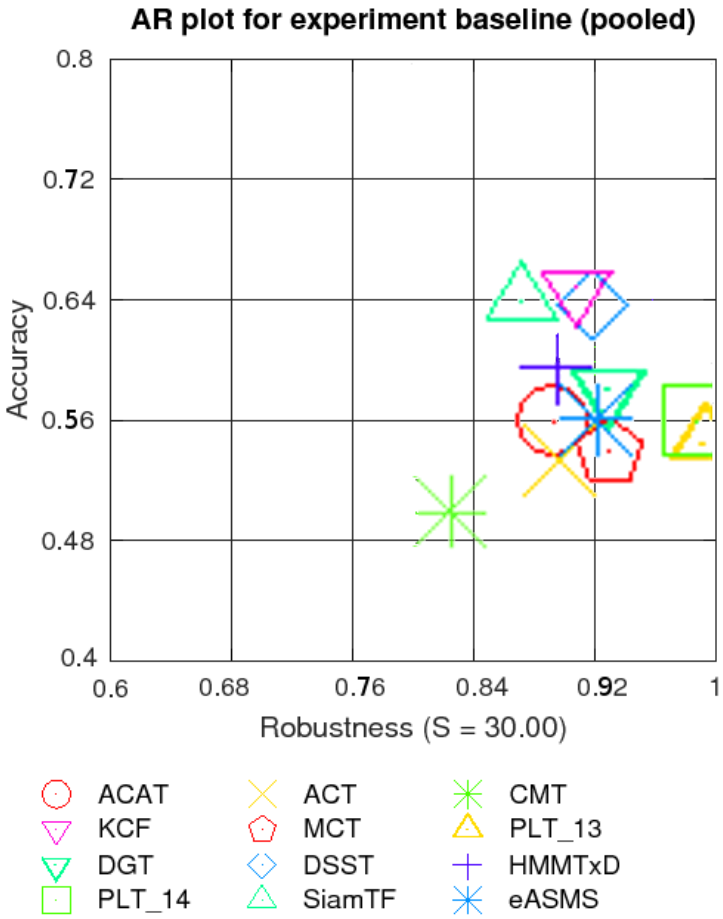


Figura 6.2: Gráfico VOT-14 de los resultados para *accuracy-robustness*.

En relación con la superposición esperada promedio, el *tracker* desarrollado obtuvo una puntuación de **25,36**, la cual resulta ser ligeramente mejor que la lograda por SiamFC, de 25,15 puntos.

6.3. VOT 2015

Con las mismas métricas que el VOT-14 pero con un conjunto diferente de vídeos, los resultados obtenidos de precisión-robustez en el VOT-15 son los mostrados en la gráfica 6.3, logrando unos valores de **58,23** y **82,27** puntos, respectivamente. Los resultados de SiamFC para esta misma prueba son de 53,35 y 86,53 puntos, por lo que existe en torno a un 9 % de mejora en la robustez y casi un 5 % de empeoramiento en la precisión, lo cual se considera aceptable. Además del modelo desarrollado, `SiamTF`, en esta gráfica también se muestran otros *trackers* presentados a VOT-15.

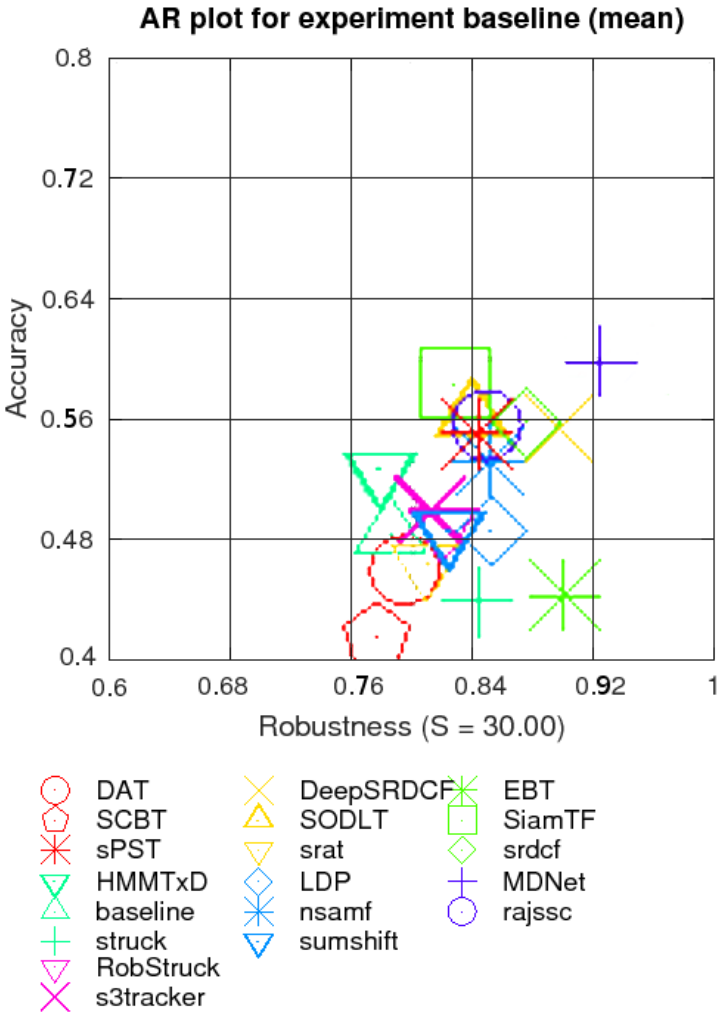


Figura 6.3: Gráfico VOT-15 de los resultados para *accuracy-robustness*.

En relación con la superposición esperada promedio, el *tracker* desarrollado obtuvo una puntuación de **26,31** puntos, algo inferior a la lograda por SiamFC, de 27,43 puntos, pero dentro del margen del 5 % definido por el RN.03.

6.4. Objetos pequeños

Para evaluar la calidad del modelo en el *tracking* de objetos pequeños, se hizo uso de las métricas del OTB con un conjunto de vídeos de *testing* extraidos de la base de datos STDdb. Se escogió usar OTB en vez de VOT dado que permite visualizar los resultados obtenidos por vídeo, posibilitando analizar la razón por la que algunos objetos son seguidos de forma incorrecta.

Los resultados obtenidos son de **58,05** puntos, tal y como se pueve ver en la Figura 6.4. Al ser tan similares a los obtenidos en OTB-13, es posible afirmar que el modelo desarrollado realiza el *tracking* de objetos pequeños con una excelente calidad.

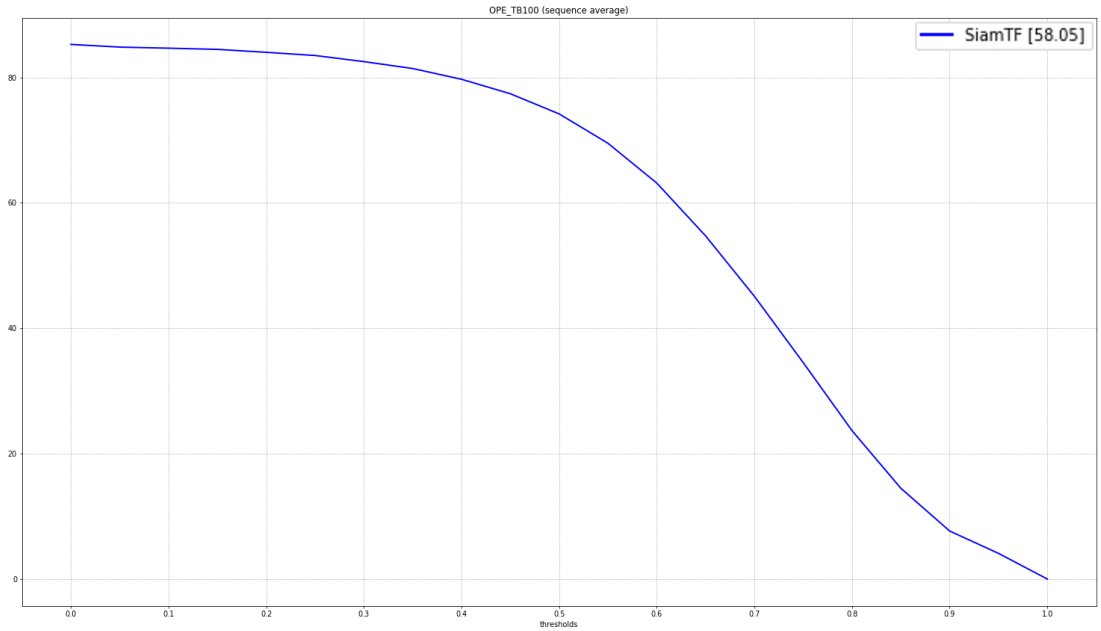


Figura 6.4: Gráfico OTB de los resultados para OPE en objetos pequeños.

6.5. Comparativa entre redes

Durante el desarrollo, se descubrió que las capas de red descritas en el artículo de SiamFC no se correspondían con el código fuente disponible en MatConvNet, ni eran coherentes con algunas de las gráficas mostradas. Es por esto que se llevó a cabo una comparativa de diversas configuraciones (`Artículo`, `Repositorio` y `Mezcla`), para dar con la que ofrecía los mejores resultados.

Tras realizar la comparativa mostrada en la Tabla 6.1, se determinó que la red basada en la configuración del código fuente del repositorio, `Repo`, era la mejor.

	Artículo	**Repositorio**	**Mezcla**
conv1 filter	96	96	96
conv2 filter	128 (×2)	128 (×2)	256 (×2)
conv3 filter	192	384	192
conv4 filter	96 (×2)	192 (×2)	192 (×2)
conv5 filter	64 (×2)	128 (×2)	128 (×2)
branch output	128	256	256
PC webcam fps **AVG | MAX**	∼27 | ∼34	∼23 | ∼29	∼20 | ∼24
GPGPU fps **AVG | MAX**	∼71 | ∼90	∼68 | ∼84	∼67 | ∼78
GPGPU HD fps **AVG | MAX**	∼54 | ∼65	∼52 | ∼64	∼52 | ∼64
OTB **AUC**	58,78	59,70	59,81
VOT-14 **OVE | ACC | ROB**	24 | 63 | 86	25 | 64 | 89	24 | 62 | 85
VOT-15 **OVE | ACC | ROB**	27 | 57 | 84	26 | 58 | 83	25 | 56 | 83

Tabla 6.1: Comparativa de redes

Tal y como se puede ver, en cuanto a la velocidad, el tensor de salida menos profundo de la configuración de `Artículo` permite una tasa de imágenes por segundo mayor, siendo las opciones `Repositorio` y `Mezcla` algo más lentas conforme se aumenta la profundidad de sus capas. No obstante, esta diferencia se atenúa cuando los fotogramas tratados tienen una gran resolución.

En lo relativo a los resultados OTB, es posible percibir un ligero empeoramiento usando la configuración del `Artículo`, mientras que las otras dos alternativas presentan unos resultados muy similares.

Por último, las métricas para las diversas configuraciones en VOT-14 y VOT-15 resultan muy similares. La opción `Mezcla` presenta un resultado ligeramente inferior, mientras que `Artículo` y `Repositorio` se encuentran muy igualadas.

6.6. Análisis de tiempos para múltiples objetos

Haciendo uso del servidor de computación GPGPU cedido por el CiTIUS para procesar un vídeo HD de 1.800 fotogramas, se ha realizado un análisis de la evolución de la velocidad de procesado en función del número de objetos seguidos. En la Figura 6.5 se encuentran recogidos los resultados obtenidos:

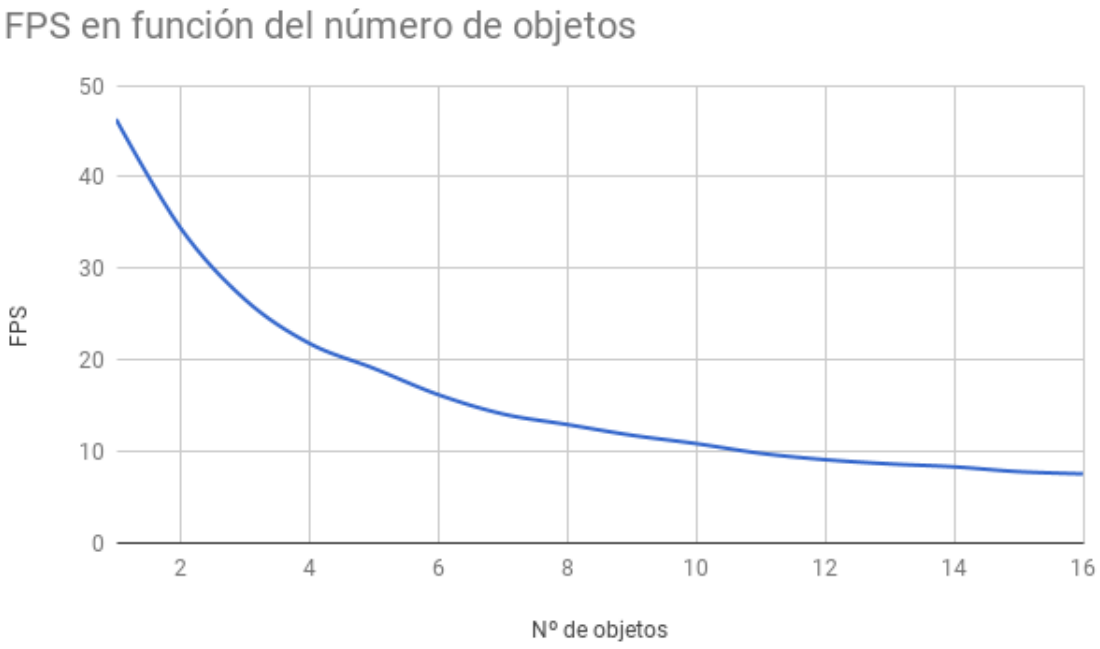


Figura 6.5: Fotogramas por segundo en función del número de objetos.

Tal y como se puede ver en la Figura 6.5, la velocidad de inferencia disminuye de forma logarítmica conforme se van incorporando nuevos objetos. Esto supone una mejora de velocidad con respecto a la replicación de *trackers* individuales, pero

sigue sin ser apropiado para aplicaciones reales con un gran número de objetos simultáneos.

Capítulo 7

Conclusiones y trabajo futuro

En el presente capítulo se recogen las conclusiones obtenidas tras la finalización del desarrollo del proyecto y se lleva a cabo una reflexión sobre sus posibles mejoras y trabajo futuro.

7.1. Conclusiones

El proyecto llevado a cabo ha supuesto para el alumno un primer punto de contacto con el mundo del aprendizaje automático, y en especial con la visión por computador y todas las posibilidades que ofrece. Resulta sorprendente la precisión con la que un modelo correctamente entrenado pueda realizar inferencias, pese a que su desarrollador no posea el conocimiento específico acerca de los criterios seguidos por la máquina para la asociación.

Haciendo uso de esta técnica de inteligencia artificial, ha sido posible desarrollar en TensorFlow un *tracker* en tiempo real basado en SiamFC, el cual presenta una calidad de inferencia equiparable, llegando incluso a superarlo en algunos *benchmarks*. Seguir la arquitectura de red creada por Luca Bertinetto y Jack Valmadre no fue una tarea trivial, ya que en su publicación omitían numerosos detalles importantes y existían algunas contradicciones con respecto a su implementación.

Con respecto a la red SiamFC, se han aplicado una serie de mejoras interesantes como la optimización para objetos pequeños, la inclusión de *tracking* para múlti-

ples objetos o la mejora del entrenamiento y curación de bases de datos. Esta última mejora no sólo ha logrado potenciar los resultados obtenidos en *benchmarks*, sino también reducir en un 40 % el espacio necesitado para almacenar las bases de datos curadas.

Además de la implementación de la red como tal, también se han desarrollado una serie de herramientas para el análisis de sistemas de *tracking* visual en general. Otras de las utilidades más destacadas son las de conversión de formatos entre bases de datos, las cuales permiten la utilización de repositorios de vídeos propios para *benchmarking*.

Para facilitar el despliegue del sistema en diversos dispositivos para *testing* y demostración, se llevó a cabo la dockerización de la aplicación de inferencia. Esto permite desplegar en un contenedor y con las mínimas depencendias una instancia totalmente funcional del *tracker*.

Si bien surgieron algunos inconvenientes durante la realización del proyecto, es posible concluir que este fue finalizado de forma muy satisfactoria en contenido y plazo, ya que todos los objetivos fueron cumplidos y las expectativas de completitud del proyecto fueron superadas.

7.2. Trabajo futuro

En el término más inmediato, la siguiente tarea a realizar sería la integración de la aplicación en un sistema de *tracking* de alto nivel. De esta forma, se contaría con un detector para que los objetos a seguir no tuvieran que ser manualmente delimitados por un humano. En este sistema de alto nivel también existiría una capa adicional de supervisión, la cual permitiría realizar asociaciones de datos espacio-temporales para asegurar que las posiciones transmitidas de los objetos son las correctas.

Otra acción a realizar realmente interesente sería la mejora de la calidad del modelo, para que las inferencias realizadas contaran con una mayor precisión. No obstante, dada la naturaleza del aprendizaje automático, esto no resultaría trivial. Esta mejora podría llevarse a cabo mediante la afinación de hiperparámetros, la

ampliación del conjunto de entrenamiento o la modificación de la estructura de la red neuronal.

Muy ligada a la mejora anteriormente descrita, se encontraría la de adaptar el sistema para que haga uso de cualquier tipo de delimitador, no únicamente rectángulos y no necesariamente alineados con los ejes. Sin embargo, se desconoce actualmente cómo afrontar este reto en lo relativo a rotaciones, y se podría degradar la velocidad de seguimiento actualmente alcanzada.

En lo relativo al rendimiento, sería muy interesante el desarrollo de la presente arquitectura en C, aunque existe la limitación de que la API ofrecida por TensorFlow para este lenguaje no se encuentra completa.

También sería muy recomendable la mejora del *tracking* de múltiples objetos. Si bien el modelo actual resulta más eficiente que crear una instancia de red para cada elemento, no es totalmente escalable. Es por esto que se debería desarrollar algún tipo de arquitectura que permitiera seguir un número arbitrario de objetos con un coste casi constante.

Por último, para permitir una óptima incorporación del producto desarrollado en un sistema integrado, habría que realizar una serie de pruebas que permitieran descubrir la razón por la que en el Jetson Developer Kit se aprecia una falta de rendimiento con respecto a GPUs de sobremesa. En relación con esto, sería interesante comprobar si supondría una mejora la realización de algunas operaciones de CPU en GPU, o viceversa.

Apéndice A
Manual técnico

Dado que es el idioma más extendido para la realización de publicaciones científicas, el manual técnico se encuentra escrito en inglés, tal y como dicta uno de los requisitos no funcionales del proyecto. Además, este manual ha sido escrito en Markdown [56], para facilitar su inclusión en repositorio y lograr una visualización mejorada desde línea de comandos.

SiamTF tracker

Arbitrary real-time object tracking with Fully Convolutional Siamese networks, in TensorFlow.

Technical manual

In order to use this program, you can deploy it in a **Docker** container or run it **directly on a computer** or server. If you choose the Docker option, keep in mind that only the inference module is available.

Docker installation

In order to use this tracker within Docker, you shoud have **NVIDIA Docker** installed. The installation is straightforward if you follow their quickstart guide.

Once you have installed NVIDIA Docker, you only need to download this repository, `cd` into the `Docker` directory and run the `build_SiamTF.sh` script, as follows:

```
git clone https://github.com/lorenzovaquero/siamfc-tensorflow
.git
cd siamfc-tensorflow/Docker
./build_SiamTF.sh
```

Now you have a Docker image called `siamtf` ready to use.

Direct installation

In order to use this program directly on a **Unix-based** computer or server, it's necessary to have installed the following **Python 2.7** requirements (they can reside on a virtual evironment):

- numpy 1.14.0 or higher - Fundamental package for scientific computing with Python.
- tensorflow-gpu 1.5.0 or higher - Machine Learning software library for high performance numerical computation, using GPUs.
- scipy 0.17.0 or higher - Python-based ecosystem of open-source software for mathematics, science, and engineering.
- opencv_python 3.4.0.12 or higher - Library of programming functions mainly aimed at real-time computer vision.
- matplotlib 2.1.2 or higher - Plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
- Pillow 3.1.2 or higher - Friendly Python Imaging Library fork.
- nvidia-ml-py 7.352.0 or higher - Python Bindings for the NVIDIA Management Library. (Only needed if you want to automatically-choose a GPY based on it's free memory)

You can easily install these packages with `pip` and the included `requirements.txt` file as follows:

```
git clone https://github.com/lorenzovaquero/siamfc-tensorflow
.git
cd siamfc-tensorflow/Aplicacion
```

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

However, keep in mind that there are some packages that require additional libraries and drivers. One of these packages is `tensorflow-gpu` and, as stated in its installation guide, it needs the **CUDA Toolkit**, the **cuDNN SDK**, the **GPU drivers** that support your version of the CUDA Toolkit and the **libcupti-dev** library.

Apéndice B
Manual de usuario

Dado que es el idioma más extendido para la realización de publicaciones científicas, el manual de usuario también se encuentra escrito en inglés, tal y como dicta uno de los requisitos no funcionales del proyecto. Este manual también ha sido escrito en Markdown [56], para facilitar su inclusión en repositorio y lograr una visualización mejorada desde línea de comandos.

SiamTF tracker

Arbitrary real-time object tracking with Fully Convolutional Siamese networks, in TensorFlow.

User manual

Due to its multiple functionalities, you can use this program in multiple ways:

Inference

In order to use the program in inference mode, you need to run the `track.py` Python file from the `Aplicacion/scripts` folder. If you are using the program in Docker mode, you have to run the `run_SiamTF.sh` script from the `Docker` folder. They both take the same arguments and produce the same results.

The arguments admitted are as follows:

```
usage: track.py [-h] [-p file] [-o file] [-f folder] [-v file
] [--detailed] [--matplotlib] [--verbose] [--nodisplay] [--no
logs] [--scheduled] [--multi] [-w deviceNumber] [video]

Tracks a target inside a video and displays the results

positional arguments:
```

```
  video            video to analyze (in mp4, avi, VOT's forma
t, OTB's format or an IP camera stream)

optional arguments:
  -h, --help       show this help message and exit
  -p file          JSON file with tracking parameters
  -o file          creates a groundtruth text file with the t
racking results
  -f folder        saves the video frames containing the boun
ding box to the folder
  -v file          creates a mp4 video containing the boundin
g box of the tracking result
  --detailed       shows detailed video information during tr
acking
  --matplotlib     displays the tracking progress with matplo
tlib instead of opencv
  --verbose        prints detailed information about the prog
ram execution
  --nodisplay      doesn't show the video during tracking
  --nologs         doesn't store execution logs
  --scheduled      loads in memory the entire video, before s
tarting the tracking
  --multi          allows tracking multiple objects at the sa
me time
  -w deviceNumber  loads the video in real time from the devi
ceNumber camera
```

Below are some usage examples:

```
./track.py -w 0 --detailed --multi

./run_SiamTF.sh Pugs.mp4 --detailed

./track.py -w "nvcamerasrc ! video/x-raw(memory:NVMM), width=
(int)1280, height=(int)720,format=(string)I420, framerate=(fr
action)30/1 ! nvvidconv flip-method=0 ! video/x-raw, format=(
string)BGRx ! videoconvert ! video/x-raw, format=(string)BGR
! appsink" --detailed --multi
```

During inference, if you chose to use the display, you can play and pause the video with the `Space bar`. If you are in `--multi` mode, you can add more objects with the `+` key or remove them with `-`.

Training

If you want to manually train the model, you need a curated dataset. In order to do this, you have to download an ILSVRC or STDdb database and run the `curateILSVRC.py` or `curateSTDdb.py` Python scripts. They both take the same arguments:

```
usage: curateXXXXX.py [-h] [-r sampleRatio] [-b batchSize] [-
p file] [--raw] input output

Creates a curated dataset for training the program.
```

```
positional arguments:
  input           original dataset unzipped folder
  output          destination of the curated dataset

optional arguments:
  -h, --help      show this help message and exit
  -r sampleRatio  dataset percentage used for extracting stat
s
  -b batchSize    number of images loaded at the same time fo
r extracting stats
  -p file         JSON file with parameters
  --raw           saves raw images for augmentation, without
splitting between exemplar and search areas
```

Below are some usage examples:

```
./curateSTDdb.py ~/STDdb-100/std_dataset ~/STDdb-100/std_data
set-Curated-Raw --raw

./curateILSVRC.py ~/ILSVRC2017/ILSVRC ~/ILSVRC2017/ILSVRC-Cur
ation-Raw
```

After the curation, you can train your model with the `train.py` Python script. It takes the following arguments:

```
usage: train.py [-h] [-p file] [-r folder] [--verbose] [--nol
```

```
ogs] dataset modelFolder

Trains the model using a curated dataset

positional arguments:
  dataset      curated ILSVRC dataset for training
  modelFolder  output folder for the learned models

optional arguments:
  -h, --help   show this help message and exit
  -p file      JSON file with training parameters
  -r folder    resume a training session from the checkpoints
 stored in the folder
  --verbose    prints detailed information about the program
execution
  --nologs     doesn't store execution logs
```

Below is an usage example:

```
./train.py ~/ILSVRC2017/ILSVRC-Curation-Raw ~/MyModels
```

Evaluation

In order to evaluate the model quality, you can use the VOT Toolkit or the OTB Toolkit. Since the official sites already contain tutorials, this manual only indicates the path of the files used by the toolkits: `Aplicacion/evaluation/VOT` and `Aplicacion/evaluation/OTB`.

File conversion

This application also contains a set of tools for converting videos and databases.

Video to VOT

In order to convert a `.mp4` or `.avi` video into VOT's format, you need to use the `video_to_VOT.py` Python file from the `Aplicacion/scripts` folder. It takes the following arguments:

```
usage: video_to_VOT.py [-h] [--low] inputVideo outputFile

Converts a mp4 video to VOT's format.

positional arguments:
  inputVideo  input mp4 video file
  outputFile  output VOT's format video

optional arguments:
  -h, --help  show this help message and exit
  --low       outputs a low-quality video (faster tracking)
```

Below is an usage example:

```
./video_to_VOT.py Pugs.mp4 pugs --low
```

VOT to video

In order to convert a VOT video into `.mp4` or `.avi`, you need to use the `VOT_to_video.py` Python file from the `Aplicacion/scripts` folder. It takes the following arguments:

```
usage: VOT_to_video.py [-h] [--tracking] inputFile outputVide
o

Converts a VOT video to mp4 format.

positional arguments:
  inputFile    input VOT folder
  outputVideo  output mp4 video

optional arguments:
  -h, --help   show this help message and exit
  --tracking   includes groundtruth's bounding box
```

Below is an usage example:

```
./VOT_to_video.py tc_SuperMario_ce mario.mp4 --track
```

STDdb to OTB or VOT

In order to convert a STDdb video database into VOT's or OTB's format for benchmarking, you need to use the `STDdb_to_OTB.py` or the `STDdb_to_VOT.py` Python files from the `Aplicacion/scripts` folder.

They both take the same arguments:

```
usage: STDdb_to_OTB.py [-h] input output

Transforms a STDdb dataset into OTB's format.

positional arguments:
  input       STDdb dataset unzipped folder
  output      destination of the VOT dataset

optional arguments:
  -h, --help  show this help message and exit
```

Below is an usage example:

```
./STDdb_to_OTB.py ~/STDdb-100/std_dataset ~/STDdb-100/std_dat
aset-OTB
```

ILSVRC small videos to ViTBAT

In order to convert the small object videos within a ILSVRC database into ViTBAT's format, you need to use the `ILSVRCSmallVideoExtractor_ViTBAT.py` Python file from the `Aplicacion/scripts` folder. It takes the following arguments:

```
usage: ILSVRCSmallVideoExtractor_ViTBAT.py [-h] [-s area] [-t
 frames] input output
```

```
Extracts ImageNet VID videos of small objects in ViTBAT forma
t

positional arguments:
  input       ImageNet VID dataset unzipped folder
  output      destination folder of the ViTBAT videos

optional arguments:
  -h, --help  show this help message and exit
  -s area     maximum area size of the objects
  -t frames   minimum time in frames that the object must be
in scene
```

Below is an usage example:

```
./ILSVRCSmallVideoExtractor_ViTBAT.py -s 256 -t 150 ~/ILSVRC2
017/ILSVRC ~/ILSVRC2017/ILSVRC-ViTBAT
```

Parameters

The different tools and functionalities of this application can take a series of parameters in JSON format. Next, the admitted parameters are described along with their default value, type and description:

GENERAL PARAMETERS

- gpuId = 0 # integer [-1, +inf) | id of the GPU used for computations

("-1" means auto-select)

- logsFolder = os.path.abspath(os.path.join(os.path.dirname(**file**), '...', 'logs')) # path | folder for storing tracking and training logs
- branchType = 'AlexNet' # text ['AlexNet' OR 'AlexNet'] | type of neural network that will make up the branches
- totalStride = 8 # integer [1, +inf) | total stride of the network, obtained by multiplying the individual strides of each layer
- exemplarSize = 127 # integer [1, +inf) | side size (in pixels) of the square exemplar image that will enter the network
- searchAreaSize = 255 # integer [1, +inf) | side size (in pixels) of the square search area images that will enter the network
- contextAmount = 0.5 # float [0, +inf) | amount of context left around a bounding box for obtaining its exemplar image
- adjustFactor = 0.001 # float (0, +inf) | value of the adjust convolutional layer, in order to tune the loss calculation
- epsilon = 1e-6 # float (0, +inf) | value used in order to avoid zeroes in the variance calculation

TRACKER PARAMETERS

- numScales = 3 # integer [1, +inf) | number of scales that will be considered during tracking (only odd numbers will make a change) | a "5" would mean that there are considered 2 bigger scales, 2 smaller scales and the same scale

- scaleStep = 1.0375 # float (1, +inf) | step of the change in size when searching over scales | it will be the base of an

exponentiation, so it souldn’t be more than “1.5”

- scalePenalty = 0.9745 # float [0, 1] | penalization factor applied to the score maps of a different scale

- scaleDamping = 0.59 # float [0, 1] | factor for updating the scale by linear interpolation, to reduce damping. Reduces the effect of the ‘scaleStep’ when updating the bounding box

- sizeMinFactor = 0.2 # float [0, 1] | target’s lower size limit. It’s a factor that multiplies the target’s initial size, and prevents shrinking the object if it gets smaller than that

- sizeMaxFactor = 5.0 # float [1, +inf) | target’s upper size limit. It’s a factor that multiplies the target’s initial size, and prevents enlarging the object if it gets bigger than that

- upsampleFactor = 16 # integer [1, +inf) | factor for upsampling the score map with bicubic interpolation, in order to increase the accuracy, making it less coarse

- windowing = ‘cosine’ # text [‘cosine’ OR ‘uniform’] | stablishes the distribution of the weights at the displacement penalization window | a “uniform” value will cancel any displacement penalization

- windowInfluence = 0.176 # float [0, 1] | influence of the displacement window (in convex sum)

- modelPath = os.path.abspath(os.path.join(os.path.dirname(**file**), '...', 'model')) # path | folder where the trained model is stored

- modelName = 'model_tf.ckpt' # text | name of the trained model file

- trackingLogsFolderName = 'tracking' # text | name of the subfolder where the tracking logs will be stored

- trackingLogsStep = 100 # integer [1, +inf) | number of frames run between log writes at tracking

TRAINING PARAMETERS

- randomSeed = 1 # integer [0, 4294967295] | seed used for generating the random numbers during training (it allows result replication)

- trainBatchSize = 8 # integer [1, +inf) | number of image pairs that will be processed at each step (at the same time) during training

- imdbTrainingRatio = 1 # float [0, 1] | percentage of imdb videos that wil be used for training (the rest will be used for evaluation)

- searchAreaAugmentationMargin = 8 # integer [0, 'searchAreaSize'/2) | margin (in pixels) left in the search area images used for training in order to allow the augmentation

- positiveRadius = 16 # integer [1, ‘totalStride’ * ‘scoreSize’ * sqrt(2) / 2] | radius of pixels, centered at the target which will be given a positive label during training

- neutralRadius’] = 80 # integer [0, ‘totalStride’ * ‘scoreSize’ * sqrt(2) / 2 - 'positiveRadius | radius of pixels, centered at the target which will be given a neutral label during training (the positive radius won’t be altered)

- numPairs = 53200 # integer [1, +inf) | number of image pairs that will be processed at each epoch during training

- trainNumEpochs = 50 # integer [1, +inf) | number of epochs that will be done during training (each epoch will process ‘numPairs’ image pairs)

- frameRange = 100 # integer [1, +inf) | maximum separation (in frames) between an exemplar image and its search area during training

- initMethod = ‘kaiming’ # text [‘kaiming’ OR ‘xavier’] | method used for initializing the weights during training

- momentum = 0.9 # float (0, +inf) | amount of momentum applied during training when computing gradients

- stddev = 0.01 # float [0, +inf) | Standard deviation of the normal

distribution when initializing net weights

- trainLearningRateStart = 1e-2 # float (0, +inf) | value of the learning rate at the start of the training

- trainLearningRateStop = 1e-5 # float (0, +inf) | value of the learning rate at the end of the training

- trainLearningRatePolicy’] = ‘exponential’ # text [‘exponential’ OR ‘cosine’ OR 'linear | annealing policy used for decaying the learning rate during training

- convolutionWeightDecay = 5e-4 # float (0, 1] | weight decay applied at convolutions during training, where after each update, the weights are multiplied by this factor in order to prevent them from growing too large

- batchNormalizationWeightDecay = 0.95 # float (0, 1] | weight decay applied at batch normalizations during training, where after each update, the weights are multiplied by this factor in order to prevent them from growing too large

- maxStretch = 0.05 # float [0, 1] | maximum amount of stretch applied when altering images for training

- maxTranslate = 4 # integer [0, searchAreaAugmentationMargin] | maximum amount of translation applied when altering images for training

- colorVarianceFactor = 0.05 # float [0, +inf) | influence of the dataset color standard deviation when altering images for training

- mirroringProbability = 0 # float [0, 1] | probability of mirroring a frame when altering images for training

- trainingLogsFolderName = ‘training’ # text | name of the subfolder where the training logs will be stored

- trainingLogsStep = 100 # integer [1, +inf) | number of batches run between log writes at training

VALIDATION PARAMETERS

- validationLogsStep = 100 # integer [1, +inf) | number of batches run between log writes at validation

CALCULATED PARAMETERS

- scoreSize = (‘searchAreaSize’ - ‘exemplarSize’) / ‘totalStride’ + 1 # integer | Size of the score map that results from cross-correlate the searh area and exemplar features

- trackingLogsFolder = os.path.join(‘logsFolder’, ‘trackingLogsFolderName’) # text | path of the folder where the tracking logs will be stored

- modelFile = os.path.join(‘modelPath’, ‘modelName’) # text | path of the trained model file used for tracking

- trainingLogsFolder = os.path.join(logsFolder, 'trainingLogsFolderName) # text | path of the folder where the training logs will be stored

- imdbEvaluationRatio = 1 - imdbTrainingRatio # float | percentage of imdb videos that wil be used for evaluation (the rest will be used for training)

- trainingSearchAreaSize = ‘searchAreaSize’ - 2 * ‘searchAreaAugmentationMargin’ # integer | size of the search area images used for training (if the raw image is not directly used), after the augmentation is applied

- trainingRawSize = ‘searchAreaSize’ + 2 * ‘searchAreaAugmentationMargin’ # integer | size of the raw images saved in order to have ‘searchAreaSize’ size images during training after data augmentation

Apéndice C
Glosario

API: Una API o Interfaz de programación de aplicaciones es un conjunto de subrutinas, funciones y procedimientos ofrecidos por una biblioteca para ser utilizados por otro software como una capa de abstracción.

Backpropagation: La propagación hacia atrás de errores es un método utilizado para entrenar redes neuronales artificiales. Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas siguientes de la red, hasta generar una salida. La señal de salida se compara con la salida deseada y se calcula una señal de error que permite corregir los pesos de las neuronas.

Benchmark: También llamado prueba de rendimiento o comparativa, es una técnica utilizada en informática para medir la calidad o el rendimiento de un sistema o uno de sus componentes.

CNN: Una red neuronal convolucional es un tipo de red neuronal artificial donde las neuronas corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual primaria (V1) de un cerebro biológico.

CUDA: La Arquitectura Unificada de Dispositivos de Cómputo es una plataforma de computación en paralelo, incluyendo un compilador y un conjunto de herramientas de desarrollo creadas por NVIDIA, que permiten a los programadores usar una variación del lenguaje de programación C para codificar algoritmos en GPU.

Curación: Un proceso de curación, en el ámbito del entrenamiento para apren-

dizaje por computador, consiste en la adaptación de una base de datos de entrenamiento genérica, para que se ajuste al modelo desarrollado y se requiera del mínimo esfuerzo computacional durante el entrenamiento.

Detección: Técnica de visión por computador consistente en identificar elementos de interés en una imagen y establecer áreas delimitadoras que los contengan, reconociendo el tipo de los objetos.

Entrenamiento: En aprendizaje por computador, un entrenamiento consiste en la exposición del modelo ante conjuntos de datos etiquetados con *groundtruths*, de forma que este pueda aprender una serie de patrones que le permitirán realizar futuras inferencias correctas ante datos nuevos. Este aprendizaje se lleva a cabo a través del cálculo de resultados mediante inferencia y la corrección del error mediante *backpropagation*, ajustando los pesos de las neuronas.

Evaluación: La evaluación de un modelo consiste en la exposición del mismo ante conjuntos de datos con *groundtruths* conocidos, para que se realicen inferencias. Estas inferencias realizadas son después comparadas con los valores considerados reales, permitiendo obtener una valoración de la calidad del modelo.

Fuente en streaming: En el ámbito del aprendizaje por computador, una fuente en streaming es un dispositivo digital (como una cámara o una red de computadores) que distribuye contenido multimedia en directo, de manera que el sistema cliente utiliza esta información a la vez que la descarga. Al ser esta información distribuida en tiempo real, no es posible acceder a datos del futuro ni del pasado (a no ser que estos últimos se almacenen en algún tipo de búfer).

GPGPU: *General-Purpose Computing on Graphics Processing Units* es un concepto reciente dentro de informática que trata de estudiar y aprovechar las capacidades de cómputo de una GPU, dado su bajo precio en relación a su potencia de cálculo, gran paralelismo y optimización para operaciones en punto flotante, entre otros.

GPU: Una unidad de procesamiento gráfico es un coprocesador dedicado al pro-

cesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de trabajo del procesador central.

Groundtruth: En aprendizaje por computador, el *groundtruth* es el valor medido considerado correcto para una inferencia, utilizado durante el entrenamiento y la evaluación.

ILSVRC: La ImageNet Large Scale Visual Recognition Competition es una iniciativa surgida en el departamento de Ingeniería Informática de la Universidad de Princeton que busca crear una gran base de datos para la investigación de software de visión por computador.

Inferencia: Se trata de una evaluación que, partiendo de una serie de datos y hechos, es capaz de extraer una consecuencia o predicción. En este caso, partiendo de datos nunca antes vistos, es posible extraer conclusiones basadas en los conjuntos de información explorados durante el entrenamiento.

Objeto pequeño: Dentro del ámbito de la visión por computador, se consideran objetos pequeños aquellos elementos captados en fotogramas cuyo cuadro delimitador mínimo tiene un área de menos de 256 píxeles.

OTB: El Online Object Tracking Benchmark es una técnica comparativa desarrollada por Y. Wu, J. Lim y M.H. Yang para medir la calidad de diversos *trackers* visuales.

STDdb: La Small Target Detection database es una base de datos de vídeos anotados, en los que figuran objetos pequeños para detección y *tracking*. Se encuentra en desarrollo por el CiTIUS dentro del proyecto CNNs for Small Target Detection, financiado por Gradiant.

Tensor: En matemáticas y en física, un tensor es cierta clase de entidad algebraica de varios componentes, que generaliza los conceptos de escalar, vector y matriz de una manera que sea independiente de cualquier sistema de coordenadas elegido.

Tracking: Técnica de visión por computador consistente en mantener la identidad de diferentes elementos detectados a lo largo de los fotogramas de un vídeo, de forma que en todo momento es posible establecer una región

delimitadora que indique la posición para cada objeto.

VOT: La iniciativa Visual Object Tracking Challenge busca proporcionar a la comunidad de *tracking* visual una forma definida y repetible de comparar *trackers* de corta duración, así como establecer una plataforma común en la que compartir los diversos avances y resultados.

Bibliografía

[1] *¿Quieres saber cómo se produce la visión? – Blog de Clínica Baviera*. URL: https://www.clinicabaviera.com/blog/quieres-saber-como-se-produce-la-vision/ (visitado 02-06-2018).

[2] *Computer vision – Wikipedia*. URL: https://en.wikipedia.org/wiki/Computer_vision (visitado 02-06-2018).

[3] *Video tracking – Wikipedia*. URL: https://en.wikipedia.org/wiki/Video_tracking (visitado 02-06-2018).

[4] Dongxuan Li y Wenjie Chen. "Object tracking with convolutional neural networks and kernelized correlation filters". En: *Chinese Control And Decision Conference* n.º 29 (mayo de 2017), págs. 1039-1044.

[5] M. Kristan y col. "The Visual Object Tracking VOT2017 Challenge Results". En: *The IEEE International Conference on Computer Vision (ICCV)* (oct. de 2017).

[6] *Artificial neural network – Wikipedia*. URL: https://en.wikipedia.org/wiki/Artificial_neural_network (visitado 02-06-2018).

[7] Luca Bertinetto y Jack Valmadre. "Fully-Convolutional Siamese Networks for Object Tracking". En: *arXiv preprint* 1606.09549 (oct. de 2016).

[8] Jack Valmadre y col. "End-To-End Representation Learning for Correlation Filter Based Tracking". En: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (jul. de 2017).

[9] Alex Krizhevsky, Ilya Sutskever y Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". En: *Proceedings of the*

25th International Conference on Neural Information Processing Systems 1 (2012), págs. 1097-1105.

[10] Escuela Superior de Informática – Universidad de Castilla - La Mancha. "Gestión de Riesgos en Proyectos Software". En: *Planificación y Gestión de Sistemas de Información* 7 (feb. de 2017).

[11] Vitae Consultores. "Guía Salarial del Sector TIC en Galicia 2016-2017". En: *Estudio Salarial 2016-2017* (jun. de 2017).

[12] *Xestión de Actividade de I+D – Calculadora de Contratos*. URL: http://imaisd.usc.es/ferramentas/calculadora/calculadoracontratos.asp (visitado 02-06-2018).

[13] *Agencia Tributaria – Tabla de coeficientes de amortización lineal*. URL: https://www.agenciatributaria.es/AEAT.internet/Inicio/_Segmentos_/Empresas_y_profesionales/Empresas/Impuesto_sobre_Sociedades/Periodos_impositivos_a_partir_de_1_1_2015/Base_imponible/Amortizacion/Tabla_de_coeficientes_de_amortizacion_lineal_.shtml (visitado 02-06-2018).

[14] *Lenovo Legion Y520 – Lenovo España*. URL: https://www3.lenovo.com/es/es/laptops/ideapad/legion-y-series/Legion-Y520/p/88GMY500808 (visitado 02-06-2018).

[15] *Dell PowerEdge R720 – Dell España*. URL: http://www.dell.com/es/empresas/p/poweredge-r720/pd (visitado 02-06-2018).

[16] *Kits de desarrollo y Módulos Jetson para sistemas embebidos – NVIDIA Jetson*. URL: https://www.nvidia.es/autonomous-machines/embedded-systems-dev-kits-modules/ (visitado 02-06-2018).

[17] *Tarjeta gráfica NVIDIA Titan Xp – NVIDIA*. URL: http://www.nvidia.es/graphics-cards/geforce/pascal/titan-xp/ (visitado 02-06-2018).

[18] *22M47VQ-P – LG España*. URL: http://www.lg.com/es/monitores/lg-22M47VQ-P (visitado 02-06-2018).

[19] *Teclado multimedia Dell - KB216 – Dell España*. URL: http://www.dell.com/es-es/shop/teclado-multimedia-dell-kb216/apd/580-adgs/accesorios-para-pc (visitado 02-06-2018).

[20] *TeckNet Classic TrueWave 2.4G Wireless*. URL: http://www.tecknet.co.uk/m002-grey.html (visitado 02-06-2018).

[21] *Microsoft Project – Software de administración de proyectos*. URL: https://products.office.com/es-ES/project/ (visitado 02-06-2018).

[22] *Acordos Consello de Goberno: Custos Indirectos*. URL: http://imaisd.usc.es/ftp/oit/documentos/591_gl.pdf (visitado 02-06-2018).

[23] *PyCharm – Python IDE for Proffesional Developers*. URL: https://www.jetbrains.com/pycharm/ (visitado 02-06-2018).

[24] *Python*. URL: https://www.python.org/ (visitado 02-06-2018).

[25] *TensorFlow*. URL: https://www.tensorflow.org/ (visitado 02-06-2018).

[26] *Caffe – Deep Learning Framework*. URL: http://caffe.berkeleyvision.org/ (visitado 02-06-2018).

[27] *TensorBoard – Visualizing Learning*. URL: https://www.tensorflow.org/programmers_guide/summaries_and_tensorboard (visitado 02-06-2018).

[28] O. Russakovsky y col. "ImageNet Large Scale Visual Recognition Challenge". En: *IJCV* (2015).

[29] *MATLAB – MathWorks*. URL: https://www.mathworks.com/products/matlab.html (visitado 02-06-2018).

[30] *GNU Octave*. URL: https://www.gnu.org/software/octave/ (visitado 02-06-2018).

[31] M. Kristan y col. "The Visual Object Tracking VOT2015 Challenge results". En: *The IEEE International Conference on Computer Vision (ICCV)* (2015).

[32] Yi Wu, Jongwoo Lim y Ming-Hsuan Yang. "Online Object Tracking: A Benchmark". En: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013).

[33] *OpenCV library*. URL: `https://opencv.org/` (visitado 02-06-2018).

[34] *Matplotlib: Python plotting*. URL: `https://matplotlib.org/index.html` (visitado 02-06-2018).

[35] *Procesamiento paralelo CUDA – NVIDIA*. URL: `http://www.nvidia.es/object/cuda-parallel-computing-es.html` (visitado 02-06-2018).

[36] *NVIDIA cuDNN – NVIDIA Developer*. URL: `https://developer.nvidia.com/cudnn` (visitado 02-06-2018).

[37] *NVIDIA Docker – GitHub*. URL: `https://github.com/NVIDIA/nvidia-docker` (visitado 02-06-2018).

[38] *Git*. URL: `https://git-scm.com/` (visitado 02-06-2018).

[39] *LaTeX – A document preparation system*. URL: `https://www.latex-project.org/` (visitado 02-06-2018).

[40] *TeX Live – TeX Users Group*. URL: `https://www.tug.org/texlive/` (visitado 02-06-2018).

[41] *Texmaker – Free cross-platform latex editor*. URL: `http://www.xm1math.net/texmaker/` (visitado 02-06-2018).

[42] *WBS Tool – A free web software for WBS Charts*. URL: `http://www.wbstool.com/` (visitado 02-06-2018).

[43] *draw.io*. URL: `https://www.draw.io/` (visitado 02-06-2018).

[44] *StarUML*. URL: `http://staruml.io/` (visitado 02-06-2018).

[45] *Spreadsheet – Google*. URL: `https://spreadsheets.google.com/` (visitado 02-06-2018).

[46] K. He y col. "Deep residual learning for image recognition". En: *arXiv preprint* 1512.03385 (2015).

[47] S. Xie y col. "Aggregated Residual Transformations for Deep Neural Networks". En: *arXiv preprint* 1611.05431v1 (2016).

[48] R. Hahnloser y col. "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit". En: *Nature* 405 (2000), págs. 947-951.

[49] Sergey Ioffe y Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". En: *arXiv preprint* 1502.03167 (2015).

[50] *TensorFlow-Slim – GitHub*. URL: `https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/slim` (visitado 02-06-2018).

[51] *Softmax function – Wikipedia*. URL: `https://en.wikipedia.org/wiki/Softmax_function` (visitado 02-06-2018).

[52] Kaiming He y col. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". En: *arXiv preprint* arXiv:1502.01852 (2015).

[53] *Stochastic gradient descent – Wikipedia*. URL: `https://en.wikipedia.org/wiki/Stochastic_gradient_descent` (visitado 02-06-2018).

[54] *Pickle – Python object serialization*. URL: `https://docs.python.org/2/library/pickle.html` (visitado 02-06-2018).

[55] *Docker*. URL: `https://www.docker.com/` (visitado 02-06-2018).

[56] *Markdown – Wikipedia*. URL: `https://en.wikipedia.org/wiki/Markdown` (visitado 02-06-2018).