Desarrollo de Entornos Virtuales para el Entrenamiento de Clasificadores

Laura Sellart Roldán

Resumen — Detectar objetos en imágenes es una de las funcionalidades clave de múltiples aplicaciones. Los detectores más prometedores se entrenan a partir de muestras etiquetadas, basándose en su apariencia. Este proceso de etiquetaje es una tarea subjetiva realizada por humanos, que requiere de un gran esfuerzo y resulta en anotaciones poco precisas. El desarrollo de entornos virtuales controlables permite la generación automática de imágenes etiquetadas con gran precisión, donde los datos sintéticos obtenidos sirven para entrenar clasificadores, son posteriormente utilizados para detectar objetos en imágenes reales. En este proyecto, se implementa una herramienta para la obtención automática de imágenes sintéticas etiquetadas para entrenar clasificadores, a partir de un entorno virtual urbano controlable. Como prueba de concepto abordamos la detección de peatones a partir de las imágenes obtenidas del entorno creado. Los resultados se comparan con conjuntos de entrenamiento existentes, tanto virtuales como reales, demostrando que entrenamientos sintéticos y reales dan resultados parecidos en las detecciones, y que las mejoras en la calidad gráfica y la flexibilidad de los entornos virtuales permiten realizar grandes avances en la clasificación de objetos.

Palabras clave — Animación 3D realista, Detección de peatones, Entorno virtual controlable, Entrenamiento con datos sintéticos, Visión por computador.

Abstract — Detecting objects in images is a key functionality of multiple applications. The most promising detectors are trained from labeled samples, based on their appearance. This annotation process is a subjective task performed by humans, it requires a lot of effort and results in inaccurate annotations. The development of controllable virtual environments allows the automatic generation of labeled images with great precision, where synthetic data is used for training classifiers, which are later used to make detections on real images. In this project, it is implemented a tool for automatic production of synthetic labeled images for training, from a controllable virtual urban environment. Proof of concept is based on pedestrian detection with the images obtained in the created environment. The results are compared with different datasets, virtual and real, showing that synthetic and real based training give similar results in detections, and that improvements in graphic quality and flexibility of the virtual environment allows important advances in object classification.

Index Terms— Computer Vision, Pedestrian detection, Realistic 3D animation, Syntethic training data, Virtual Controllable Environment.

1 Introducción

La detección automática de objetos en imágenes, es la base de múltiples aplicaciones en video vigilancia, sistemas multimedia, generación automática de estadísticas y, nuestra área de interés, la conducción asistida o incluso automática. El propósito de los *sistemas avanzados de asistencia al conductor* (ADAS), y más específicamente los *sistemas de protección de peatones (*PPS), es aumentar la seguridad al volante detectando situaciones peligrosas y reaccionando ante ellas para evitar accidentes de tráfico, siendo un importante campo de investigación [1, 2].

Para ello, se utilizan sistemas que detectan peatones en tiempo real, los cuales utilizan clasificadores que aprenden la apariencia de los peatones, a través de ejemplos previamente categorizados: *peatón* y *fondo* (Figura 1).

Estos sistemas se deben adaptar a la variabilidad de

Figura 1. Ejemplos de imágenes categorizadas, a la izquierda positivas (*peatón*) y a la derecha negativas (*fondo*). Las capturas de la primera fila pertenecen al conjunto de datos de Daimler AG y la segunda fila al conjunto de datos INRIA.

- *E-mail de contacto: laura.sellart@cvc.uab.es*
- *Mención realizada: Ingeniería del Software*
- *Trabajo tutorizado por: Dr. Antonio M. López (Ciencias de la computación)*
- *Curso 2014/15*

poses de las personas, iluminaciones, fondos, tamaños y deformaciones propias de las imágenes como el movimiento o la saturación. Su captura y anotación requiere de un gran esfuerzo humano, sujeto a errores, y los sistemas existentes de recolección de datos para la etiquetación masiva no proporcionan la suficiente precisión, llevando a errores durante el entrenamiento (Figura 2). Los factores mencionados, la cantidad de ejemplos con variabilidad alta [3] y la calidad de los metadatos que acompañan a las imágenes, son las claves para conseguir buenos resultados.

Figura 2. A la izquierda, ejemplo de una imagen anotada manualmente con la aplicación LabelMe, mediante la creación de polígonos con poca precisión. A la derecha, ejemplo de las imágenes generadas con nuestra aplicación de manera automática, con las anotaciones por colores a nivel de píxel, de gran precisión.

Paralelamente, el mundo de la Animación por Computador avanza cada vez más hacia el realismo, mejorando la calidad de los entornos, efectos y modelos 3D. En el área de ADAS, al diferir cada vez menos los entornos reales de los modelados por ordenador, se ha realizado una apuesta por la generación de entornos virtuales para el entrenamiento de detectores con imágenes sintéticas. Estos entornos permiten una extracción automática de imágenes con anotaciones precisas. Existen varias pruebas de concepto [4] que demuestran que imágenes sintéticas son igual de válidas que las capturadas en el mundo real.

Por ello, este proyecto se propone la creación de una herramienta de generación automática de imágenes categorizadas a partir de un entorno urbano variable, con el objetivo de entrenar detectores de objetos. Esta herramienta, permite un control total de las características de la escena y los objetos en ella, consiguiendo una mayor calidad y variabilidad de ejemplos anotados que los que conseguiríamos realizando capturas manuales. Además, permite la obtención de información que seria prácticamente imposible de obtener en el mundo real, como los angulos de orientación de un objeto respecto a la cámara, o la trayectoria de destino de un peatón o vehículo.

La facilidad del motor gráfico Unity 3D para creación de videojuegos y la cantidad de contenidos 3D gratuitos disponibles en internet para modelar la escena, hacen que este proyecto sea una cómoda y escalable solución.

La prueba de concepto final demuestra la potencia de nuestra herramienta, generando un conjunto de imágenes para el entrenamiento sintético en un clasificador de peatones del grupo ADAS del *Centro de Visión por Computador* (CVC) y probando los resultados en imágenes de conjuntos de datos reales muy reconocidas como son INRIA [5] y Daimler [6]. El resultado se compara con otro conjunto de datos sintético existente, la CVC-04 Virtual-World Pedestrian Dataset 2 (CVC4) [7] generada a partir de la modificación del videojuego Half Life 2. Para refinar este resultado, se realiza una *domain adaptation* (DA), que consiste en mezclar unos pocos ejemplos del mundo real, con una gran cantidad del mundo virtual, para entrenar un detector adaptado, consiguiendo un mayor acercamiento a los valores objetivo.

La herramienta desarrollada permite una automatización total de la generación, ofreciendo controles para modificar diferentes aspectos añadiendo variabilidad, además de una gran precisión y escalabilidad en los objetos anotados. Los resultados confirman que los valores obtenidos en los entrenamientos con imágenes reales y sintéticas son muy parecidos, a pesar de no haber hecho una selección concreta de las imágenes generadas. La herramienta está diseñada para tener total flexibilidad para capturar diferentes entornos y a ser modificada adaptándose a posibles cambios o mejoras en los sistemas, teniendo por ello muchas perspectivas de futuro.

1.1 Objetivos

Los objetivos del proyecto son:

a) Creación entorno virtual.
b) Desarrollo herramienta de generación automática de imágenes anotadas.
c) Prueba de concepto basada en detectar peatones.

a. *Creación del entorno virtual.* La base de este proyecto es la creación de un entorno virtual realista de ciudad, utilizando contenidos gratuitos de internet, para la extracción de imágenes. La ciudad ha de contener varias calles con sus respectivas aceras, señales, edificios, vallas, coches, personas, bicicletas y vegetación. Para enriquecer el detalle de la escena, se han de incluir objetos como toldos, manchas de suciedad y tapas de alcantarilla, entre muchos otros.

b. *Desarrollo herramienta de generación automática de imágenes anotadas.* El siguiente paso es generar una herramienta capaz de extraer imágenes del entorno virtual. Estas imágenes han de ser de tres tipos diferentes por captura realizada (Figura 3): la captura sintética del mundo virtual, el *ground truth*, donde tenemos las anotaciones por colores a nivel de pixel de la categoría del objeto y el *depth map*, donde el color de cada píxel se corresponde con la distancia de la cámara hasta el objeto que contiene, dentro de una escala de grises. A su vez, se ha de generar un *fichero de anotaciones* con la información de los peatones que aparecen en la imagen. Para ello, se ha de permitir realizar ciertas configuraciones previas en los ajustes del entorno, para poder definir diferentes momentos del día e iluminaciones, además de las características de salida de

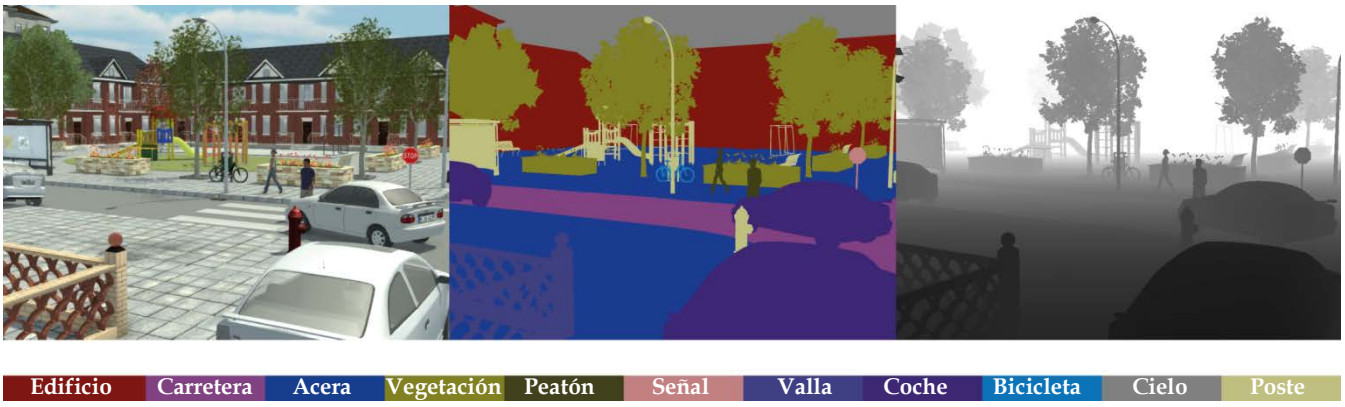


Figura 3. Ejemplo de las tres imágenes generadas en cada captura. La primera, la captura real del mundo virtual, la segunda, el *ground truth*, con anotaciones a nivel de pixel de la categoría, y la tercera, el *depth map*, un mapa de profundidad de hasta 50m de distancia, con anotaciones de la distancia a nivel de pixel.

las imágenes. El sistema debe ser capaz de crear imágenes del entorno virtual de manera automática, sin supervisión de un humano. Para ello, se ha de generar una funcionalidad que permita realizar recorridos, donde la cámara automáticamente se posiciona a lo largo de un camino, y realice en cada punto la captura del entorno.

c. *Prueba de concepto basada en detectar peatones.* El último paso es validar los resultados realizando una demostración centrada en la detección de un único objeto, los peatones. Para ello, se ha de generar un conjunto de imágenes sintéticas anotadas, con peatones y fondos, para comparar el aprendizaje obtenido con ellas contra el obtenido por imágenes del mundo real. A su vez, se debe comparar el resultado con los proporcionados por otro conjunto de datos sintético. Para ello, se ha de utilizar un clasificador de peatones proporcionado por el grupo ADAS del CVC, y realizar diferentes experimentos hasta conseguir los valores deseados. Finalmente, se han de refinar los resultados realizando *domain adaptation*.

1.2 Estructura del artículo

En la Sección 2 se realiza un análisis de los avances hasta la fecha en detección de objetos, anotación masiva de ejemplos, y realismo por computador, permitiendo generaciones automáticas de imágenes para entrenar clasificadores con datos sintéticos. En la Sección 3, se detalla la metodología utilizada, el diseño, y las funcionalidades implementadas. En la Sección 4, se muestran las pruebas de validación realizadas, y una discusión sobre los resultados, para finalmente, en la Sección 5, concluir con unas reflexiones sobre el proyecto.

2 Estado del arte

La búsqueda del sistema perfecto de detección de objetos en imágenes es un campo de investigación con más de 15 años de recorrido. Durante la última década, en el campo de la detección de peatones, se han desarrollado muchas soluciones que han conseguido adaptarse en escenarios cambiantes del mundo real. Se han realizado estudios sobre la detección en imágenes del espectro visible, y capturas térmicas, pero la gran mayoría se centran en el espectro visible, debido a dos factores: al hecho de que los accidentes suelen ocurrir durante el día, y al retraso en la tecnología térmica, ofreciendo cámaras caras y con baja resolución, tema que actualmente se está resolviendo.

Las técnicas existentes analizan las texturas, colores, relación entre *frames* consecutivos y distinciones de *peatón-fondo*, realizando una división de la imagen en partes según las anotaciones, para finalmente generar un modelo basado en la apariencia. En ellos el peatón se puede entender como un todo [8], o por partes [9]. Se encuentran diferentes artículos que resumen los avances hasta la fecha [1].

La precisión en las clasificaciones, cuando se utilizan datos de dos entornos diferentes capturados con sensores distintos, como son las imágenes del entorno virtual y real, suele generar peores resultados que si se utilizan datos de un mismo dominio. Este problema, llamado *dataset shift*, es un problema común que se encuentra en otras áreas como el reconocimiento de voz. En nuestro sistema, está presente la diferencia entre los dominios, al realizar las diferentes combinaciones de los conjuntos de datos a entrenar, combinando tanto reales como virtuales. Para ello, se utiliza una estrategia denominada adaptación de dominio o *domain adaptation*, en la que el conjunto de entrenamiento se adapta, mezclando sus ejemplos con unos pocos del conjunto de prueba, consiguiendo una mejora en las detecciones.

La captura de imágenes del mundo real se realiza a través de la grabación de recorridos mediante la utilización de coches con una o varias cámaras, colocadas en un compartimento discreto del retrovisor del parabrisas. Mediante vehículos como los de la iniciativa ecoDrivers [10] y los Mercedes-Benz S-, E- y C-Class de Daimler AG [11], se han capturado conjuntos de imágenes que, mediante un proceso de anotación manual, han generado conjuntos de datos hoy en día muy conocidos como CVC02, INRIA, Daimler.

La cantidad de ejemplos y variabilidad entre ellos es indispensable para una buena clasificación, por lo que se han desarrollado técnicas para multiplicar el conjunto de ejemplos, mediante la manipulación de las imágenes de peatones, en las que se transforma la forma, textura, y fondo mediante ligeras modificaciones en la posición de la región de interés, llamado *jittering*, y mediante la extracción de muestras a partir de falsos positivos, llamado *bootstrapping*.

El proceso de anotación de las imágenes capturadas es totalmente supervisado; personas deben categorizar objetos en las imágenes, y delinear su silueta. Por ello esta tarea genera un gran esfuerzo y está sujeta a errores humanos que disminuyen la calidad de los resultados (Figura 2). Es indispensable disponer de una gran cantidad de ejemplos correctamente anotados para el buen aprendizaje, generando una gran cantidad de trabajo. Existen iniciativas para recolectar anotaciones de manera masiva a través de internet, como *Amazon's Mechanical Turk*, MTurk o LabelMe, en las cuales personas etiquetan objetos en imágenes de manera voluntaria. A pesar de ello, se continúa sin poder afirmar que sea un método perfecto que no acabe generando errores durante el entrenamiento del clasificador, lo que ha generado un nuevo campo de investigación donde se discuten las necesidades de estos datos, y cómo recolectarlos [3].

Debido a este gran esfuerzo humano para la generación de conjuntos de datos, en el campo de ADAS se ha realizado una apuesta por el uso de entornos virtuales, aprovechando el gran avance en el campo de la animación por computador hacia el realismo. Los avances en tecnología y fotorealismo han permitido que películas, simuladores y videojuegos consigan recrear cada vez mejor el mundo real, no solo en apariencia, sino también en las mecánicas del mismo como animaciones o comportamientos, permitiendo un control total del entorno y su visualización.

Se han generado aplicaciones mediante la adaptación de videojuegos realistas. Existen ejemplos como los proyectos creados con VDrift [12] o Half Life 2 [13] para generar capturas automáticas con anotaciones precisas. En ellos, se ha adaptado el entorno de acuerdo a las necesidades, para conseguir la extracción de imágenes imitando las características de las obtenidas en el mundo real, como la altura y velocidad del coche, además de un sistema de extracción de anotaciones.

De esta manera, se elimina totalmente el esfuerzo humano en el proceso de captura y anotación, además de tener un gran control del entorno, y por tanto, se dispone de una gran variabilidad de ejemplos con total precisión. Diversas pruebas de concepto demuestran que el realismo en la animación por computador es totalmente válido para aprender la apariencia de los objetos y detectarlos en imágenes del mundo real [4] [13].

En este proyecto se realiza un avance sobre la creación y entrenamiento de datos sintéticos para el entrenamiento de clasificadores, generando un entorno urbano con mayor realismo que proyectos parecidos anteriores, para comprobar el rendimiento en un caso de detección sobre imágenes del mundo real, gracias a varios conjuntos de datos conocidos. Realizando un desarrollo desde zero, se obtiene una herramienta totalmente manejable y adaptada a nuestros objetivos. El contenido de la escena generada se puede obtener gratuitamente en internet, gracias a diversos portales donde usuarios publican modelos 3D.

Dentro de los motores de desarrollo de videojuegos del mercado, uno de los que destaca es Unity3D. Contando con una amplia comunidad de usuarios y siendo prácticamente multiplataforma, se adapta totalmente a nuestras necesidades permitiendo generar entornos y herramientas programadas en C#. Cuenta con toda una suite de utilidades que permiten automatizar aspectos como el acabado de las imágenes o la inteligencia artificial, además de su propia tienda virtual donde se pueden obtener extensiones de funcionalidades, efectos, y todo tipo de modelos.

Con todo lo mencionado, se demuestra que nuestra propuesta es una aportación necesaria para el avance en la investigación sobre la detección de objetos, que va a permitir un control total sobre la creación de imágenes y anotaciones, asegurando resultados sintéticos igual de válidos que datos reales, y todo ello aprovechando los avances en animación por computador y desarrollo de videojuegos.

3 METODOLOGÍA

3.1 Planificación

En la planificación, se decidió emplear una metodología de desarrollo ágil, centrada en las funcionalidades a implementar, en la cual se han realizado cinco iteraciones. En cada una, se han implementado diversas utilidades y generado versiones entregables de la aplicación, para poder demostrar de manera visual los avances conseguidos y discutir con el tutor mejoras a realizar.

El proyecto se ha llevado a cabo concurrentemente con unas prácticas de empresa en el CVC, lo que ha permitido dedicar más horas de desarrollo. La duración total del proyecto ha sido de 6 meses, en la Figura 4 se muestra la planficación seguida con un Diagrama de Gantt.

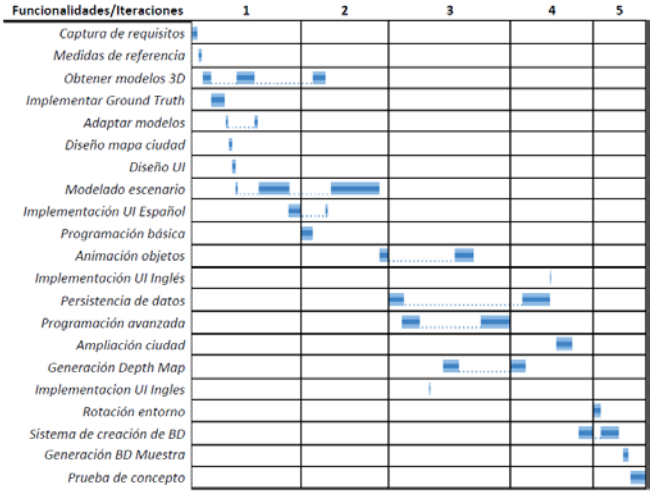


Figura 4. Diagrama de Gantt, con el despliege de las cinco iteraciones de funcionalidades realizadas, durante 1000 horas de desarrollo.

A partir de la captura de requisitos, se han definido las tareas a realizar, seguido del diseño de los componentes, para continuar con la implementación mediante un control de versiones. Durante el proceso, se han realizado pruebas de rendimiento, para controlar la carga de la aplicación en el sistema debido a la complejidad de la escena determinada por el número y la calidad de los objetos y efectos, y de esta manera optimizar el flujo de ejecución. En la validación de cada una de las funcionali-

dades, se han realizado pruebas guiadas por casos de uso, generando todos los escenarios posibles mediante la definición de un estado inicial, unos pasos a realizar, y un estado final. Para ello, se han tenido que realizar ejecuciones completas de la aplicación, generando imágenes y mejorando la aplicación si el resultado no era el deseado. Finalmente, se han generado imágenes del mundo virtual centradas en peatones, generando un conjunto de entrenamiento para probar los resultados obtenidos mediante el software clasificador de peatones proporcionado.

Ha sido necesaria cierta formación para el desarrollo de utilidades concretas, pero ya se disponían de los conocimientos básicos de uso del motor gráfico Unity3D.

Como recursos utilizados, respecto al software, destacamos Unity3D como motor gráfico, con códigos programados en C#. El clasificador HOG-SVM del grupo ADAS del CVC para la validación de los resultados, y creación de pequeñas herramientas en Java con Eclipse, y Matlab. IrfanView para la visualización de las imágenes y TortoiseSVN como control de versiones. Respecto a los modelados 3D, se han consultado portales como Archive3d.net o TurboSquid.com, entre otros. Los conjuntos de datos públicos utilizados para la detección de peatones son CVC4 [7], INRIA [5] y Daimler [6].

3.2 Diseño

En el diseño de clases, se han analizado las necesidades de código, respecto a la configuración de las entidades de nuestra aplicación, entendiendo entidad como objeto representativo que engloba multiples funcionalidades, como por ejemplo el punto de luz global que simula el sol del mundo virtual. En Unity3D, cada objeto de la escena puede ser controlado mediante código, enlazando éste como un componente suyo. Por ello, se han definido varias clases dependiendo del objeto al que van anclado. En la Figura 5 se puede observar un ejemplo con el detalle de los componentes asignados a un peatón de la escena. El resto de nuestras clases desarrolladas, como Interfaz, Control de Escena, Control de Cámara, Generar Información de Peatones, y Capturador de Imágenes, entre otras, se enlazan de manera similar con distintos objetos. Se han utilizado dos patrones de diseño, *Observer* para la colocación y articulación aleatoria de personajes animados, y *Bridge* para los idiomas.

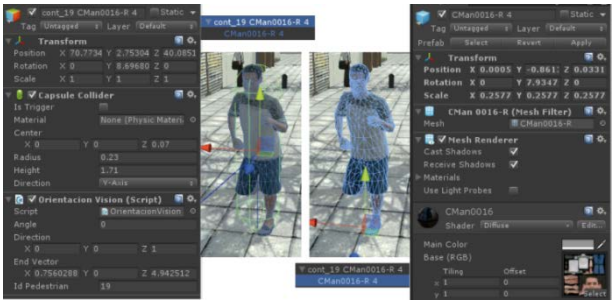


Figura 5. Ejemplo de los componentes enlazados a un peatón en el editor de Unity3D. A la izquierda se muestra un objeto contenedor del peatón, con tres componentes: *Transform*, *Capsule Collider* y el código C# Orientación Visión, equivalentes a la posición y tamaño, eje central y código que define el ángulo de visión del peatón para las anotaciones. A la derecha, se muestra el contenido del peatón con los componentes *Mesh Filter* y *Mesh Renderer* para definir la malla, y el material que la cubre.

En el diseño de la escena generada, se han analizado los modelos disponibles, tanto su variabilidad como posibilidades de reutilización, para distribuirlos de manera que se aprovecharan lo máximo posible. Para ello, se han duplicado objetos variando su apariencia, y se ha jugado con el sentido de circulación de las calles para rellenar las áreas que forzosamente se verían por la cámara, consiguiendo finalmente un total de 14 calles.

Respecto a la experiencia de usuario, se ha realizado el diseño de una interfaz por niveles controlada por ratón, facilitando la navegación por los menús disponibles donde se permiten ajustar los parámetros de la escena y activar los modos de captura, escoger el tipo de calidad gráfica entre alta-baja, además de una sección de ayuda con explicación de los comandos básicos y la opción de cambiar entre español-inglés.

3.3 Implementación del sistema

3.3.1 Funcionalidades básicas

Para la obtención de imágenes sintéticas, es necesario el desarrollo de ciertas funcionalidades básicas como el escenario, el sistema de captura, y el control del entorno.

Primero se ha construido el escenario de ciudad mediante el editor de Unity3D, a partir de los modelos 3D obtenidos. Estos han sido adaptados, modificando su escala y apariencia, para ser lo más realistas posibles. En una primera fase se ha desarrollado el centro de la ciudad, para más tarde ampliarla y añadir variabilidad de objetos y detalles. En la Figura 6 se pueden apreciar los resultados. Entre los modelos, debemos destacar los peatones, ya que son la clave en nuestra prueba de concepto. Se disponen de 59 distintos, todos ellos rígidos excepto cuatro articulados, que permiten animaciones de movimiento. En el Apéndice 1 se muestran en detalle.

Para el ajuste de las características por parte del usuario, se ha implementado una interfaz por niveles, el detalle se muestra en el Apéndice 2. Con ella podemos acceder a los sistemas de captura automática explicados más adelante, y a los ajustes del entorno, en los menús de *Configurar Escena* y *Configurar Cámara*. Se puede ascender por los diferentes niveles del menú hasta hacer desaparecer la interfaz con la tecla Escape. Además de botones para esconder la interfaz y salir, se dispone de un menú de ayuda, y cambio de idioma.

Para el control del entorno, se dispone de los ajustes de ángulo de la luz global o sol de nuestra escena, tonalidades de luz ambiental y global, texturas de cielo, intensidades de luces y sombras, y la activación de farolas y animaciones. Todo ello se puede almacenar en lo que llamamos *Slot Escena*, disponiendo de seis diferentes que se guardan en fichero al salir de la aplicación. Con la opción rotación se puede activar la transición automática entre los diferentes slots cada cierto intervalo de tiempo. Todo ello configurable en el menú de *Configurar Escena*. En la Figura 8 se observan posibles configuraciones de la escena.

Figura 8. Diferentes momentos del día en el mundo virtual.

En el sistema de captura se utiliza la cámara de la escena, controlada mediante el teclado. Permite movernos por la ciudad y generar las imágenes de lo que está procesando en ese momento. Se disponen de ajustes como el *field of view* (FOV), velocidad, ángulo de giro, altura, inclinación y distancia de visión. Más enfocados al movimiento, se ofrecen las opciones de activar o desactivar la colisión con los objetos, además de la oscilación, para simular las deformaciones que sufrirían las imágenes capturadas por un coche real, como los baches, donde se fija que cada cierta distancia, se modifiquen ligeramente la altura e inclinación de la cámara. Todo esto configurable en el menú de *Configurar Cámara*, y también persistente al almacenarse en lo que llamamos *Slot Camara*.

Se han implementado las animaciones de las personas, coches y bicicletas de la escena, donde se ha utilizado un sistema de inteligencia artificial proporcionado por Unity3D. Permite definir la superficie transitable del terreno de la escena y generar los denominados *agentes*. Estos son entidades que se dirigen de manera automática hacia un punto definido como destino. Aprovechando esta herramienta, se han definido para cada una de las entidades a animar, un camino de nodos a seguir siendo estos los destinos del agente. De esta manera, tenemos un sistema de navegación automático que esquiva objetos evitando colisiones y recalculando el camino óptimo de manera constante. Gracias a eso se dispone de una ciudad viva con personas animadas, moviéndose de manera no articulada, excepto cuatro peatones que disponen de esqueleto animado. Estos, cuando activan su animación, aplican una serie de movimientos para cada una de las articulaciones de su esqueleto, realizando las acciones de andar y esperar quieto.

3.3.2 Funcionalidades específicas

Para la generación automática de datos sintéticos, es necesario el desarrollo de ciertas características concretas, como mecanismos de recorrido y captura automáticos, la extracción de capturas específicas con metadatos asociados, y técnicas de tratamiento de las imágenes generadas.

En la generación de datos sintéticos, por cada frame capturado se generan cuatro ficheros, tres imágenes y un *fichero de anotaciones*. En la Figura 3 se muestra un ejemplo. La primera imagen es la captura del mundo sintético, equivalente a la captura del mundo real. En este modo la cámara de nuestra aplicación tiene fijados filtros que favorecen un acabado realista: antialiasing, blur, HDR y SSAO. La segunda imagen, el *ground truth*, se genera modificando el material de todos los objetos de la escena y asignando un cierto color según la categoría de este, por lo que se rellena de manera exacta y precisa la figura del objeto con el nuevo material. En este modo la cámara no lleva ningún filtro. Deshabilitando los efectos de iluminación se obtiene una imagen de dos dimensiones en perspectiva, con las anotaciones de cada objeto a nivel de pixel. La tercera imagen es el mapa de profundidad o *depth map*, en el cual la cámara utiliza un shader especial que accede al buffer de profundidad de la tarjeta gráfica. Este asigna un color a cada pixel dentro de una escala de grises según la distancia hasta la cámara, por lo que se consiguen las anotaciones de distancia a nivel de pixel. Se puede configurar la resolución de salida al valor deseado, para este proyecto se ha utilizado una resolución de 1024x768 píxeles. El último fichero contiene datos relacionados con los peatones dentro de la captura. En la Figura 9 se muestra un ejemplo de su utilización. Más adelante, en esta misma sección, se detalla su creación y funcionamiento.

Para la captura automática de imágenes, se han diseñado diferentes sistemas dependiendo del punto de vista de la cámara. En el primer sistema, la captura se realiza a lo largo de la carretera de las calles de la ciudad, funcionalidad nombrada como A*utoPath*, en la que se han definido nodos a lo largo del camino y el sistema interpola la posición y rotación de la cámara entre ellos para adaptarse a la superficie del suelo, simulando el comportamiento de un coche. En el siguiente sistema, llamado G*rabar Recorrido*, es el usuario quien puede grabar, en diferentes ranuras, los movimientos que realiza por la escena durante un periodo de tiempo, almacenando modificaciones de altura e inclinación. Y por último el sistema centrado en peatones, es nombrado como C*apturar sobre Peatones*, donde se definen diferentes ubicaciones donde colocar personas en la ciudad, en lo que llamamos *cajas de peatones*. Los modelos de personas son asignados de manera aleatoria en las cajas y automáticamente la cámara rota sobre cada uno, capturando varios ángulos y poses de este. Una vez capturados todos, el sistema rota los peatones entre las cajas y repite el proceso, realizando de esta manera todas las permutaciones posibles, permitiendo

Figura 6. Vista aérea y diferentes planos de la ciudad.

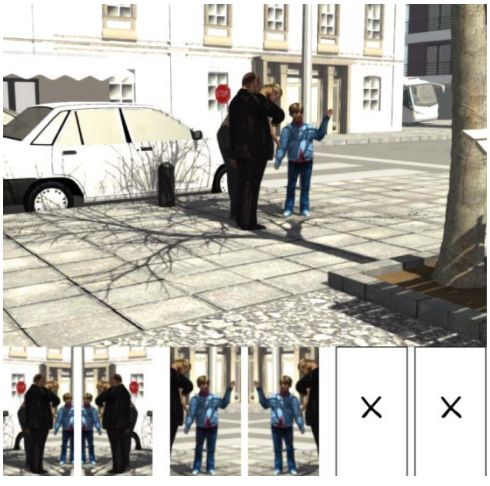
obtener un mismo peatón con diferentes fondos.

Se dispone la opción de mantener activados durante la captura automática las animaciones de los personajes y la rotación de las características de la escena, para conseguir mayor variabilidad en las muestras generadas.

De cara a la adaptación de las imágenes generadas para el entrenamiento del clasificador de peatones, se ha implementado un sistema de *Recorte de Peatones*. En cada captura, el sistema detecta y almacena la información del tamaño, posición del centro, porcentaje de visibilidad, orientación, distancia a la cámara e identificador de cada uno de los peatones que aparezca, generando un fichero de texto, que llamamos *fichero de anotaciones*. Con esta información, podemos crear imágenes válidas mediante nuestro R*ecortador de Peatones* en Matlab, totalmente adaptadas para los clasificadores. Mediante la información del *fichero de anotaciones*, se recorta cada uno de los peatones que aparezcan en la imagen (Figura 9), generando una nueva de una resolución concreta. Además, se genera la versión rotada del peatón, en base a un espejo vertical. En este proceso, se descartan peatones cortados por los bordes de la imagen, menores a una altura de 72 píxeles, y con un porcentaje de visibilidad inferior al 90%. A su vez, se organizan los ficheros originales además de los recortes generados en una jerarquía organizada y definida de carpetas.

Para la organización de las capturas, se ha implementado un programa en Java, llamado *BDOrganizer*, para obtener un subconjunto de imágenes, indicando las permutaciones de salida, peatones, y ángulos de imagen deseados.

Finalmente, una vez implementados y probados todos los sistemas, se ha realizado la creación de los datos sintéticos de peatones, mediante la funcionalidad de *Capturar sobre peatones,* en diferentes puntos de la ciudad y con varias configuraciones de la escena. La captura de imágenes de fondo, que no contienen peatones, se ha realizado manualmente para tener mayor control de la cercanía y ángulo de los objetos capturados. Para ello, se han deshabilitado los peatones de la escena y se ha capturado realizando un recorrido aleatorio. La generación de los mapas de profundidad se ha desactivado ya que no son necesarios en las pruebas que se realizan.



523.69,471.74,118.00,356.00,1,100,125.63,6.09,16
626.91,448.08,124.00,256.00,1,99,19.44,6.74,48
516.32,471.10,138.00,310.00,1,33,125.60,6.69,33
centroX,centroY,anchura,altura,separador,visibilidad,angulo,distancia,ID

Figura 9. Ejemplo de *fichero de anotaciones*. En la parte superior su captura correspondiente, con tres peatones, el más alejado solo un 33% visible. Al ejecutar el *Recortador de Peatones*, se generan las imágenes derecha y el correspondiente reflejo vertical, como se aprecia en la fila inferior, donde se ha descartado el tercer peatón al no alcanzar el 90% de visibilidad.

4 Detección de Peatones

4.1 El clasificador

En este apartado, comentamos brevemente los principales ingredientes del funcionamiento del software que se nos proporciona para clasificar peatones, llamado *DetectionSys*.

En particular utilizamos un modelo holístico de los peatones, basado en los *Histogramas de Gradientes Orientados (*HOG*)* [8] como descriptor visual, y las utiliza *Maquinas de Vectores de Soporte* (SVM) como método de aprendizaje para obtener un clasificador lineal.

Una vez se tiene el clasificador, dada una imagen, este se usa para clasificar multitud de ventanas de la misma. Es decir, para decidir si una ventana se ajusta o no a un peatón. Las ventanas se generan de forma deslizante en diferentes niveles de una pirámide (para tener en cuenta disitintas distancias de detección).

Durante el aprendizaje del clasificador, se realiza la extraccion de diferentes recortes de cada uno de los fondos proporcionados. Seguidamente, se realiza una fase de *bootstrapping* para mejorar los resultados obtenidos, donde se realiza una clasificación sobre las imágenes negativas y se obtienen más fondos a partir de los falsos positivos detectados.

Para su ejecución, es necesario configurar ciertos parámetros que definirán su comportamiento sobre los datos. En concreto, debemos definir el tamaño de los peatones a detectar, el espacio de colores de las imágenes, y los dominios con los que realizar la *domain adaptation*.

4.2 Los conjuntos de datos

En la demostración de nuestra propuesta, se ha trabajado con dos conjuntos de datos de imágenes reales, y dos de imágenes virtuales: la generada por nuestra aplicación, y otra para comparar resultados.

Como conjunto de datos genérico de peatones, se ha utilizado INRIA [5]. Muy reconocido y utilizado en múltiples trabajos, consta de ejemplos a color recolectados de escenas urbanas, de campo y de interior. Como conjunto de datos para la conducción asistida, se ha utilizado el también reconocido Daimler [6], con imágenes en escala de grises capturadas desde un coche durante un recorrido urbano. Ambos proporcionan conjuntos preparados de entrenamiento y validación, que aseguran buenos resultados en los clasificadores corrientes. De ambos conjuntos de datos se han utilizado grupos reducidos igualmente representativos, para reducir el tiempo de ejecución por experimento.

Como conjunto de datos virtual, se ha utilizado CVC4 [7]. Creado y probado recientemente [4, 13], consta de ejemplos a color capturados en un entorno urbano virtual mediante recorridos aleatorios. Se ha generado a partir de los contenidos del videojuego Half Life 2, permitiendo la generación de capturas y anotaciones de manera precisa y automática. A diferencia del resto, CVC4 no proporciona un conjunto de test ni *ficheros de anotaciones*.

El conjunto de datos sintético generado por nuestra aplicación se ha llamado *Multiple Annotated Virtual Images* (MAVI), también generado a partir de un entorno urbano. Para su construcción, se han creado cinco conjuntos de ejemplos de peatones. Se distinguen por la localización en la ciudad, la superficie de colocación, las condiciones de la escena y la densidad entre ellos. Por otra parte, se han generado cuatro conjuntos de datos de fondos, con diferentes condiciones de cámara y escena. Para generar el conjunto final, se ha realizado una selección concreta de los positivos, que se explica en la Sección 4.3.

En la Tabla 1 se muestra la comparación de los diferentes conjuntos de datos con sus características finales.

4.3 Evolución de MAVI

Durante el proceso de validación, se han detectado dos mejoras a realizar en las anotaciones de MAVI. En la primera, debido a la importancia del correcto centrado del peatón en el recorte, se ha redefinido el centro de los modelos en la escena, ya que se había fijado como su valor absoluto, de manera que si el peaton, por ejemplo, tiene un brazo extendido, su centro se entiende como el punto medio entre la distancia del extremo de la mano hasta el hombro contrario. Con este sistema, el peaton no quedaba bien recortado, pero se posicionaba en ciertos casos en un lateral. Para solucionarlo, se ha definido el centro del peaton como un eje vertical que atraviesa su centro absoluto, de manera que no afecta la rotación del peaton, ni la pose de este. En la segunda, se ha añadido el metadato del identificador del peaton a las anotaciones, para tener un mayor control de los modelos contenidos en MAVI, y poder generar imágenes de ciertos peatones selectivamente.

Con ello, hemos mejorado la precisión de las anotaciones, y conseguido generar MAVI mediante nuestro sistema de C*apturar sobre peatones*, realizando rotaciones de 60º sobre el modelo generando 6 capturas por peatón. El conjunto final suma un total de 7302 frames positivos. Para reducir la redundancia de imágenes en el conjunto, y de cara a generar un conjunto de imágenes con cierto sentido lógico, se ha generado mediante nuestro programa *BDOrganizer* un subconjunto de imágenes, donde el 50% son peatones de frente o de espaldas, y el 50% restante son peatones en el resto de poses posibles. Con ello, disponemos de ejemplos positivos con poses concretas y con menor redundancia. Una vez hecho esto, se ha utilizado el *Recortador de Peatones*, generando dos salidas distintas: peatones recortados a 64x128 para adaptarse al tamaño de INRIA, y con peatones recortados a 48x96 para adaptarse a Daimler.

Respecto a los negativos, al no disponer de un sistema específico de generación de fondos automática, se han obtenido realizando capturas puntuales por la ciudad, permitiendo obtener ejemplos con total control de la configuración de la cámara. A su vez, se han obtenido algunos a partir de las ejecuciones realizadas de los sistemas de generación automática, donde se han deshabilitado los modelos de peatones. Como se puede observar, nuestro conjunto de negativos es de menor tamaño que el resto, al haber descartado ejemplos que añadian redundancia sobre los datos.

Finalmente, disponemos de la jerarquía de carpetas necesaria para el clasificador, generada por nuestro R*ecortador de Peatones*. Con esto, obtenemos nuestro conjunto sintético MAVI, preparado para diferentes conjuntos de datos y organizado de cara al detector. El conjunto final consta de 5532 recortes extraidos de 1956 imágenes positivas, y de 347 imágenes negativas.

4.4 Valores de referencia

Para la obtención de los valores de referencia en el experimento, se ha realizado un análisis de los resultados obtenidos a partir del entrenamiento y validación sobre un mismo conjunto de datos, además de generar los resultados del conjunto virtual CVC4 respecto los conjuntos de datos reales que disponemos.

	Entrenamiento			Validación			Resolución		Propiedades		
	Peatones	Positivos	Negativos	Peatones	Positivos	Negativos	Peatones	Frames	Anotaciones	Imagen a color	Peatones Ocluidos
INRIA	2422	616	1225	1126	290	458	96x160-70x134	Múltiples	X	X	X
Daimler	7844	-	2387	6745	976	793	48x96	640x480	X		X
CVC4	2422	-	6829	-	-	-	64x128	640x480		X	
MAVI	5532	1956	347	-	-	-	64x128	1024x768	X	X	X

Tabla 1. Comparación de los conjuntos de datos de peatones finalmente utilizados. Las primeras seis columnas indican las cantidades de ejemplos de entrenamiento y validación de los distintos conjuntos. Nos referimos a Peatones como a imágenes recortadas donde aparece un peaton. Se incluyen sus versiones duplicadas realizando un espejo vertical. Como positivos nos referimos a imágenes donde aparece al menos un peatón, y negativos a imágenes donde no aparece ninguno. Las siguientes cinco columnas indican propiedades respecto a la resolución de las imágenes, y aspectos concretos como la existencia de *ficheros de anotaciones*, color en las imágenes y oclusión en los peatones.

En la obtención de los resultados, se utiliza el método de evaluación *per-Image*. El sistema evalua la clasificación con la curva *Detection Error Trade off* (DET), mostrando los *miss rate* contra los *False Positives Per Image* (FPPI), donde usaremos el valor de *miss rate* promedio en porcentaje, obtenido del área inferior de la curva, entre los valores 10^0 y 10^{-1}.

Para comprobar la eficacia del clasificador con nuestros ejemplos, se han realizado para todos los conjuntos reales, un entrenamiento y validación sobre el mismo conjunto de datos. De esta manera, se consigue la referencia del conjunto de datos de ejemplos reales que mejor funciona en nuestro sistema de detección, y su porcentaje de fallos.

4.5 Obtención de resultados

Debemos conseguir superar con MAVI los resultados obtenidos con CVC4 detectando en imágenes reales, y todo ello acercándonos lo máximo posible al resultado obtenido en el entrenamiento y validación con ejemplos reales.

En la clasificación, el espacio de color es un parámetro importante a configurar. En las pruebas que se realizan, todos los conjuntos son a color excepto Daimler. En los experimentos en que se dispone de uno de los conjuntos a color, se ha utilizado el espacio RGB, como se realizó con INRIA en otro experimento [8], por lo que únicamente se ha usado escala de grises en la clasificación de Daimler contra Daimler.

Se han realizado clasificaciones con todos los conjuntos de datos reales y virtuales, utilizando los ejemplos proporcionados de entrenamiento y validación de las mismas, a excepción de las comparaciones entre reales y virtuales, como entrenar con INRIA y detectar en Daimler o CVC4 detectando en MAVI, y viceversa. En INRIA y Daimler, se disponen de conjuntos de entrenamiento y test preparados para dar buenos resultados en la mayoría de clasificadores, pero MAVI y CVC4, al diseñarse especialmente para el entrenamiento, no disponen de conjunto de validación, por lo que ha sido necesario generarlo mediante una selección de 60% de ejemplos para entrenamiento y 40% para test.

Gracias a ello, se ha comprobado el comportamiento del detector con los diferentes conjuntos de datos, y se han extraido las medidas de referencia a las que acercarnos, como se muestran en la Tabla 2. Respecto a los reales, con INRIA, debemos conseguir con entrenamientos sintéticos aproximarnos a un 17.23% de *miss rate*, y con Daimler a un 29.09%.

	Entrenamiento			
Test	INRIA	Daimler	CVC4	MAVI
INRIA	17,23%		34,15%	29,90%
Daimler		29,09%	30,18%	34,18%

Tabla 2. Resultados de las clasificaciones. Se muestran los valores de *miss rate* obtenidos entre los cuatro conjuntos de datos de peatones. Por columnas, los conjuntos de entrenamiento, y por filas, los de test. Las casillas en gris oscuro marcan pruebas no realizadas, en concreto, conjuntos reales contra reales.

DA	CVC4	MAVI
Con DA de INRIA	26.09%	24.71%
Sin DA de INRIA	34.15%	29.90%

Tabla 3. Resultados de la clasificación con *domain adaptation* en INRIA. En la primera fila, resultados con la adaptación realizada, y en la inferior, utilizando únicamente el conjunto de entrenamiento virtual.

Para maximizar el rendimiento de MAVI en el entrenamiento contra los conjuntos de datos reales INRIA y Daimler, se ha modificado el valor de dos parámetros del clasificador. El primero, el número de recortes negativos que se extraen de un fondo, se ha aumentado de dos a tres recortes. El segundo, el numero de fases de *bootstraping*, se ha incrementado, realizando dos iteraciones, extrayendo negativos a partir de detecciones de falsos positivos. Con nuestro conjunto virtual optimizado, hemos conseguido los *miss rates* de 29.90% con INRIA, y 34.18% con Daimler.

En el siguiente paso realizado, se ha clasificado el conjunto virtual CVC4, detectando los conjuntos de datos reales. Los resultados obtenidos han sido con INRIA, un *miss rate* de 34.15% y con Daimler, un *miss rate* 30.18%.

El último paso ha sido realizar *domain adaptation* de los conjuntos de datos virtuales con el único conjunto real que viene preparado para ello, INRIA. Dispone de ficheros que permiten la selección de un porcentaje de las muestras, para que estas sean mezcladas con las virtuales de MAVI durante el entrenamiento. En concreto, se ha mezclado un 10% aleatorio de los ejemplos reales con los virtuales, obteniendo los resultados que muestran la Tabla 3. En CVC4, un 24.71% de *miss rate* respecto al 34.15% sin realizar *domain adaptation*, y en MAVI, un 26.09% de *miss rate* respecto al 29.90% sin adaptar.

4.6 Discusión

Las clasificaciones realizadas muestran que, con INRIA, nuestro conjunto virtual MAVI da mejores resultados respecto al otro conjunto virtual CVC4, mejorando un 4.25% y un 1.38% en *domain adaptation,* consiguiendo un *miss rate* final de 24.71%, muy cercano al 17.23% obtenido con ejemplos reales de INRIA en el entrenamiento.

Con Daimler, en cambio, MAVI da peores resultados, con un 4% más de *miss rate* respecto a CVC4, consiguiendo esta un 30.18%. Cabe decir que ambos conjuntos virtuales están muy cercanos al valor ideal del entrenamiento y validación en Daimler, con un 29.09% de *miss rate*.

Esta diferencia entre los resultados obtenidos de los conjuntos de datos reales, se debe a que tienen características muy distintas. INRIA, como el conjunto generado con nuestra aplicación, está formado por peatones en espera o en movimiento, con poses de frente, de espaldas, y rotados en todo tipo de angulos. La cantidad de cada uno de ellos es aproximadamente de un 25%, 25% y 50% respectivamente, siguiendo cierta proporcionalidad. En cambio, Daimler se ha generado a partir de una captura desde un coche, teniendo por tanto, una gran cantidad de peatones con poses rotadas y en movimiento, y en cambio pocos con posiciones de frente o de espaldas y en espera.

Los peatones recortados que proporciona disponen de menor resolución que el conjunto INRIA.

Otra diferencia se atribuye a que en INRIA se incluyen personas con ropas de verano, en cambio en Daimler, al tratarse de una secuencia grabada en un mismo dia, todos ellos aparecen con un mismo tipo de ropa, en concreto abrigos muy gruesos. CVC4 supo adaptarse mejor a ella, al contener más personas con ese tipo de vestido, en cambio MAVI dispone de una gran parte con ropa de verano.

Todos estos factores generan una gran diferencia en los modelos generados por el clasificador, y por tanto, entre los resultados obtenidos. Queda pendiente, como trabajo inmediato, añadir la variabilidad necesaria a nuestro conjunto virtual para lograr mejorar la adaptación con los dos conjuntos reales.

No obstante, cabe destacar que se ha realizado un avance respecto al entrenamiento sintético, a partir de un entorno virtual suficientemente realista como para poder compararse con el mundo real, y aprender su apariencia visual para acercarse a los valores del reconocido conjunto de datos INRIA. Las ventajas de nuestro sistema respecto a los videojuegos adaptados existentes son que nuestra aplicación está especialmente generada para la extracción de imágenes en un entorno realista y urbano de bastante calidad gráfica, ofrece gran flexibilidad para manejar sus atributos, y almacenar los datos en conjuntos ordenados de información que cualquier clasificador puede utilizar. Al tratarse de información virtual, los valores de las anotaciones que obtenemos tienen la mayor precisión posible, lo cual sería imposible de conseguir manipulando manualmente las imágenes obtenidas en un entorno real.

5 Conclusiones

En este proyecto hemos desarrollado una herramienta adaptada a los detectores de objetos, capaz de generar de manera automática información ajustada a sus necesidades. Se ha diseñado un entorno urbano controlable de estética realista, para la extracción de una gran cantidad de imágenes con variabilidad entre ellas, gracias a diferentes modos de captura, y todo ello jugando con los modelos 3D de la escena y la modificación del entorno. Además, hemos podido demostrar un avance sobre resultados existentes, generando un conjunto de imágenes a partir de 59 peatones, y realizando una clasificación con HOG-SVM. Con ello hemos comprobado que nuestro conjunto sintético ha mejorado respecto al existente CVC4, dando resultados cercanos a los del conjunto de datos reales INRIA, donde tenemos todavía margen de mejora. MAVI nos permite obtener la información del *ground truth* y *depth map* de todos los objetos, a diferencia de CVC4 que solo dispone del *ground truth* de peatones. Con nuestra herramienta obtenemos datos sintéticos más precisos y manejables, demostrando que dan buenos resultados para la creación de detectores de peatones.

Con todo lo visto, podemos decir que el desarrollo de entornos virtuales realistas es el futuro para la obtención de imágenes con menor esfuerzo y mayor calidad, permitiendo un entrenamiento con datos sintéticos eficaz, donde la clave es la variabilidad de ejemplos.

La prueba de concepto realizada se ha centrado en la captura de peatones a partir de un entorno urbano, pero la flexibilidad del sistema es muy extensa, permitiendo adaptarse a cualquier otro objetivo de captura, como pueden ser coches o ciclistas, e incluso implantarse en otra escena distinta como una autopista u otra ciudad, permitiendo extraer imágenes con sus respectivas anotaciones. Además, la automatización de los métodos de recorrido y captura por el entorno permiten la creación de todo tipo de secuencias. Todo esto, permite continuar el desarrollo para generar información destinada, incluso, a otras áreas de investigación, como el *tracking* de peatones.

Agradecimientos

A mi tutor de proyecto, Antonio M. López, y a David Vázquez. Gracias a los dos por confiar en mí, y por toda vuestra ayuda en el desarrollo de este proyecto.

Referencias

[1] D. Gerónimo, A. M. López, A. Sappa, T. Graf, «Survey of pedestrian detection for advanced driver assistance systems», de *IEEE T-PAMI*, 2010.

[2] D. Gavrila, M. Enzweiler, «Monocular pedestrian detection: Survey and experiments», de *IEEE T-PAMI*, 2009.

[3] T.L Berg, A. Sorokin, G. Wang, D.A. Forsyth, D. Hoeiem, A. Farhadi, «It's all about the data», de *Proceedings of the IEEE*, 2010.

[4] D. Vázquez, A. M. López, D. Ponsa, D. Gerónimo, «Virtual and Real World Adaptation for Pedestrian Detection», de *IEEE T-PAMI*, 2014.

[5] «INRIA Dataset», lear.inrialpes.fr/data.

[6] «Daimler Dataset», www.gavrila.net/Datasets/Daimler_Pedestrian_Benchmark_D/daimler_pedestrian_benchmark_d.html.

[7] «CVC-04 Virtual-World Pedestrian Dataset 2», www.cvc.uab.es/adas/site/.

[8] N. Dalal, B. Triggs, «Histograms of oriented gradients for human detection», de *IEEE CVPR*, 2005.

[9] P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, «Object Detection with Discriminatively Trained Part Based Models», de *IEEE T-PAMI*. 2010.

[10] «eCo-Drivers», www.cvc.uab.es/adas/projects/eco-drivers/.

[11] «Mercedes-Benz Intelligent Drive», www.gavrila.net/Media_Coverage/Intelligent_Drive/intelligent_drive.html.

[12] «Project Synthetic Dataset», campar.in.tum.de/Chair/ProjectSyntheticDataset.

[13] J. Marin, D. Vazquez, D. Gerónimo, A. M. López, «Learning Appearance in Virtual Scenarios for Pedestrian Detection», de *In 23rd IEEE CVPR* , 2010.

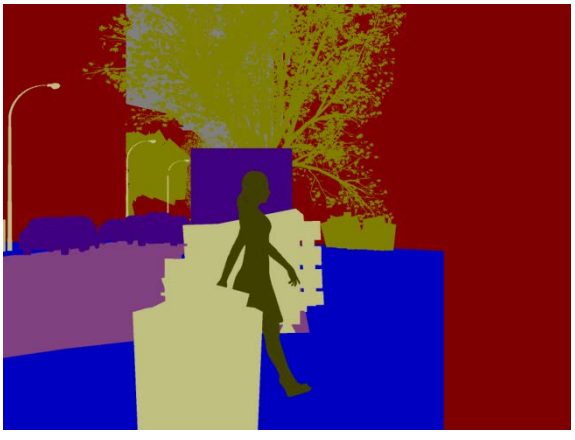
APÉNDICE

A1. Proceso de anotación del peatón

En la imagen se muestran los 59 peatones utilizados en la escena urbana. A continuación, se especifican los pasos de creación de la captura y anotación simultánea de un peatón cualquiera, con una resolución de 1024x768:

1) Captura real del mundo sintético sobre un peatón ocluido, realizando una **captura de pantalla** almacenándose en disco.

2) *Ground truth*, realizando la **sustitución de los materiales** por colores de todos los objetos de la escena según su categoría, **mostrados por la cámara**, realizando una **captura de pantall**a.



3) *Depth Map* de **50m de distancia**, obteniendo los valores del **buffer de distancia** de la gráfica, y convertiendolos en una **escala de grises**, para después realizar la **captura de pantalla**.

4) **Conversión del peatón ocluido en blanco**, y el resto de **objetos de la escena en negro**, para calcular y almacenar el **número de píxeles blancos**, obteniendo la **anchura** y **altura del peatón**.

5) **Deshabilitación de los objetos de la escena**, quedando **peatón en blanco sin ocluir**, para calcular y almacenar el **número de píxeles sin ocluir** del peatón.

6) Consultar **centro del peaton**, a partir del **eje vertical** definido como azul, y la **colisión que lo envuelve definiendo su centro** de color verde, para calcular su **posición en pantalla**.

7) Calcular **diferencia entre píxeles del peatón ocluido y sin ocluir**, para obtener el **porcentaje de visibilidad**.

8) Calcular **ángulo de visión del peatón respecto a cámara** a partir del **rayo definido manualmente**, que indica su orientación, y la **orientación de la cámara** en ese momento.

9) Calcular la **distancia de la cámara al peatón**.

10) Obtener **identificador del peatón**.

11) **Almacenar** ***fichero de anotaciones*** **en disco**, nombrado como Annotation.txt.

Anotación resultante generada:
455.02,263.17,187.00,408.00,1,74,87.01,5.33,35

Desglose:
- 455.02: Posicion X en la imagen respecto el punto inferior izquierdo de la imagen.
- 263.17: Posicion Y en la imagen respecto el punto inferior izquierdo de la imagen.
- 187.00: Anchura.
- 408.00: Altura.
- 1: Separador.
- 74: Porcentaje de visibilidad del peatón
- 87.01: Angulo de visión respecto a la cámara.
- 5.33: Distancia hasta la cámara.
- 35: Identificador del peatón.

A2. Ventanas de la interfaz de usuario

A continuación se muestran las diferentes ventanas de la interfaz de usuario en castellano, y un ejemplo de esta en ingés:

Menú principal

Versión en Inglés

Grabar Recorrido

Configurar Cámara

Configurar Escena

Configurar Captura Auto.

Ayuda