UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

INGENIERÍA

EN INFORMÁTICA

PROYECTO FIN DE CARRERA

Atravel: Aprendizaje automático de trayectorias y velocidades en entornos monitorizados para la detección de situaciones anómalas.

Roberto García Bermejo

Septiembre, 2012

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

INGENIERÍA

EN INFORMÁTICA

PROYECTO FIN DE CARRERA

Atravel: Aprendizaje automático de trayectorias y velocidades en entornos monitorizados para la detección de situaciones anómalas

Autor: Roberto García Bermejo

Director: Javier Alonso Albusac Jiménez

Tutor: David Vallejo Fernández

Septiembre, 2012

Roberto García Bermejo

E-mail: robertogb9@gmail.com
© 2012 Roberto García Bermejo

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Se permite la copia, distribución y/o modificación de este documento bajo los términos de la Licencia de Documentación Libre GNU, versión 1.3 o cualquier versión posterior publicada por la *Free Software Foundation*; sin secciones invariantes. Una copia de esta licencia esta incluida en el apéndice titulado «GNU Free Documentation License».

Muchos de los nombres usados por las compañías para diferenciar sus productos y servicios son reclamados como marcas registradas. Allí donde estos nombres aparezcan en este documento, y cuando el autor haya sido informado de esas marcas registradas, los nombres estarán escritos en mayúsculas o como nombres propios.

TRIBUNAL:

Presidente:

Vocal 1:

Vocal 2:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE	VOCAL 1	VOCAL 2	SECRETARIO
Fdo.:	Fdo.:	Fdo.:	Fdo.:

Resumen

Los sitemas de vigilancia inteligente (SVI) desempeñan funciones de detección, clasificación, seguimiento y análisis de comportamiento de objetos de forma automática. Para ello, utilizan técnicas, métodos y algoritmos de Inteligencia Artificial (IA). La necesidad de disminuir o evitar el factor humano en el desarrollo de estos sistemas, ha provocado un crecimiento en la investigación de las distintas etapas que componen un SVI.

En la actualidad existen diferentes enfoques sobre el análisis de comportamientos de objetos, cada uno con sus ventajas e inconvenientes. La mayoría de éstos realizan estudios sobre el comportamiento de objetos en 2D. Los resultados obtenidos están enfocados a entornos de monitorización específicos, resultando más favorables en determinados entornos. De forma, que para la monitorización de un entorno determinado es necesario estudiar los diferentes métodos para elegir el que mejor se ajuste.

En este proyecto fin de carrera, propone un sistema de vigilancia inteligente que permite aprender, clasificar y monitorizar trayectorias y velocidades de objetos móviles. Permite la monitorización de entornos multicámara con el objetivo de eliminar situaciones afectadas por el ruido o la oclusión de objetos en la escena. La representación de trayectorias se realiza en 3D para dotar de mayor precisión a la detección de situaciones anómalas en el entorno de monitorización. El análisis de la velocidad con la que se desplazan los vehículos a lo largo de la escena es calculada en cada uno de los tramos en los que se divide una trayectoria, proporcionando mayor flexibilidad y precisión en el estudio del comportamiento de objetos.

Atravel se caracteriza por la realización de un aprendizaje automático de trayectorias. Los motivos que permiten dotar al sistema de esta capacidad de aprendizaje, son la implementación de una buena clasificación y fusión de los distintos comportamientos de objetos reconocidos. En este PFC se incorporan técnicas de clasificación de comportamientos en función de la trayectoria, el recorrido y la velocidad con la que un objeto se desplaza a lo largo del entorno de monitorización. Además, incorpora un método de fusión para conseguir un refinamiento u optimización de las diferentes situaciones que pueden suceder en el entorno.

Índice general

Resumen IX

Índice general XI

Índice de figuras XV

1. Introducción **1**

1.1. ¿Qué es un sistema de vigilancia inteligente(SVI)? 4

1.1.1. Evolución de los sistemas de vigilancia 6

1.1.2. Factores que conllevan a utilizar los SVI 8

1.1.3. Sectores de aplicación 9

1.1.4. Análisis de trayectorias 14

1.1.5. Análisis de velocidad 15

1.2. Problemática 15

1.3. Estructura del documento 17

2. Objetivos **19**

3. Antecedentes. **23**

3.1. Introducción general. 23

3.1.1. Python. 23

3.1.2. OpenCV 24

3.1.3. OpenGL. 25

3.1.4. Pyside. 26

3.2. Fundamentos físicos y matemáticos. 27

3.2.1. Velocidad 27

3.2.2. Punto. 27

3.2.3. Vector. 28

3.2.4. Equivalencia de vectores. 30
3.2.5. Suma de vectores. 30
3.2.6. Diferencia de vectores. 31
3.2.7. Multiplicación de un vector 32
3.2.8. Producto escalar. 33
3.2.9. Producto vectorial. 34
3.2.10. Bounding box. 35
3.2.11. Método un punto está contenido dentro de un poliedro. 36
3.3. Estado de las tecnologías de análisis de vídeo. 39
3.3.1. Descripción de las técnicas de análisis de vídeo. 39
3.3.2. Análisis en grupo. 44
3.3.3. Sistemas de videovigilancia con sistemas multicámara. 46
3.3.4. Análisis de sistemas comerciales 47
3.4. Necesidades y retos del análisis de vídeo. 50
3.5. Tendencias tecnológicas para la videovigilancia. 53
3.6. Análisis de trayectorias y patrones de aprendizaje. 54
3.7. Ejemplos Sistemas de vídeo vigilancia. 56
3.7.1. Aprendizaje de patrones a partir del movimiento de objetos para la detección de anomalías. 56
3.7.2. Aprendizaje a partir de mezcla de modelos. 58
3.7.3. Recuperación de las trayectorias de los objetos para la estimación de la geometría de carriles utilizando cámaras de vigilancia de tráfico no estacionarias. 59
3.7.4. Detección de rutas en vídeo vigilancia. 62
3.7.5. Sistema de visión cognitiva. 65
3.7.6. Interpretación semántica de actividades de objetos en sistemas de vídeo vigilancia. 68
3.7.7. Análisis de trayectorias para sistemas deportivos o de videovigilancia. 72
3.7.8. Aprendizaje de patrones de movimiento realizando una clasificación basada en los Modelos Ocultos de Markov(HMM). 73
3.7.9. Análisis de trayectoria utilizando una clasificación espectral y extracción de patrones. 76
3.7.10. Análisis de la velocidad de objetos utilizando imágenes 2D 79

4. Método de trabajo **83**
4.1. Metodología de desarrollo 83
4.2. Tecnologías y herramientas utilizadas. 86

4.2.1. Lenguaje de programación. 86
4.2.2. Hardware. 86
4.2.3. Software. 87

5. Arquitectura de Atravel **91**
5.1. Módulo de entrada 93
5.1.1. Submódulo de archivo resultante del Tracking 3D 94
5.1.2. Submódulo que contiene los archivos de flujo de vídeo de la escena observada. 96
5.2. Módulo de procesamiento 97
5.2.1. Explicación de estructuras básicas. 100
5.2.2. Estructura Features_travel_object 116
5.2.3. Base de conocimiento de patrones aprendidos 124
5.2.4. Submódulo de representación de trayectoria 3D. 126
5.3. Submódulo de similitud por trayectoria. 150
5.4. Submódulo de similitud por recorrido 157
5.5. Submódulo de similitud por velocidad 158
5.6. Submódulo de fusión de patrones 161
5.6.1. Módulo de Visualización. 166
5.7. Patrones y diagramas de clases de diseño. 169

6. Evolución y Resultados **173**
6.1. Evolución 173
6.1.1. Iteraciones 173
6.1.2. Recursos y costes 182
6.2. Resultados 182
6.2.1. Resultados con entorno virtual 184
6.2.2. Resultados con entorno real 189

7. Conclusiones y propuestas **197**
7.1. Objetivos alcanzados 199
7.2. Propuestas de trabajo futuro 203
7.3. Conclusiones personales 206

A. Manual de referencia **209**
A.1. Manual de usuario 209

B. Imágenes a color **223**

Bibliografía **229**

Índice de figuras

3.1. Punto en el espacio 28
3.2. Vectores en el espacio 29
3.3. Vector unitario (ver en Anexo B imagen a color). 29
3.4. Vectores en el espacio 30
3.5. Vectores en el espacio 31
3.6. Vectores en el espacio 31
3.7. Diferencia de vectores 32
3.8. Multiplicación de vectores 32
3.9. Producto escalar de vectores. 33
3.10. Producto vectorial. 34
3.11. Sentido del producto vectorial. 35
3.12. Ejemplo de bounding box alineado con los ejes. 35
3.13. Ejemplo de bounding box rotado con respecto a los ejes. 36
3.14. Un punto P está contenido en una cara de un bounding box. 37
3.15. Representación de trayectorias realizadas arbitrariamente. 55
3.16. Trayectoria del objeto. 57
3.17. Resumen de los componentes de PTMS. 60
3.18. Representación de trayectorias. 61
3.19. Representación de los carriles centrales. 61
3.20. Representación de los carriles centrales. 62
3.21. Representación de la trayectoria siguiendo el modelo de Dimitrios Makris y Tim Ellis. 63
3.22. Representación de la topología de trayectorias en una escena. 63
3.23. Representación de la topología de trayectorias en una escena. 64
3.24. Esquema de módulos del sistema KUES. 66
3.25. Reconstrucción de la actividad de la escena. 67
3.26. Clasificación de las trayectorias en árbol. 69

3.27. Evolución de la detección de trayectorias y su modelado utilizando HMM. . 75
3.28. Imagen de intersección del vídeo donde aparecen los HMMs y los diferentes estados normales de cada trayectoria. Las trayectorias anómalas están dibujadas de color rojo. 75
3.29. División de trayectorias. 77
3.30. Clasificación de trayectorias y etiquetas. 79

4.1. Modelo del proceso software utilizado. 84
4.2. Diagrama de casos de uso. 85
4.3. Diagrama de clases, diseño preliminar. 86
4.4. Cámara Logitech Sphere AF. 87

5.1. Arquitectura general de *Atravel*. 92
5.2. Representación del contorno de un objeto, utilizando una *caja 3D*. 95
5.3. Diagrama de flujo correspondiente al aprendizaje de patrones. 100
5.4. Representación de colisión de dos bounding box (ver en Anexo B imagen a color). 108
5.5. No existe colisión si $a_x{}^m ax$ es menor que $b_x{}^m in$.. 109
5.6. No existe colisión si $a_x{}^m in$ es mayor que $b_x{}^m ax$. 109
5.7. Representación de los cuatro vértices con mayor valor en el eje Z. 110
5.8. Representación del vector normal de una cara de un BB (ver imagen a color en Anexo B). 111
5.9. Representación del BB resultante de la fusión de dos BB (ver imagen a color en el Anexo B). 114
5.10. Representación de un bounding box utilizando sólo dos vértices. 126
5.11. Representación de un bounding box intermedio. 128
5.12. Representación de los vértices de un bounding box. 128
5.13. Representación de los vértices de un bounding box intermedio. 129
5.14. Representación de las caras que componen un bounding box. 130
5.15. Representación de dos bounding box solapándose. 133
5.16. Representación del ángulo de desviación de dos bounding box. 134
5.17. Representación del punto central de un bounding box. 135
5.18. Representación del vector dirección de dos bounding box clave. 136
5.19. Representación del recorrido de un objeto (ver en Anexo B imagen a color). 148
5.20. Representación del recorrido completo de un objeto sin realizar cambios importantes en su desplazamiento (ver imagen a color en el Anexo B). . . . 149

5.21. Representación del recorrido completo de un objeto realizando cambios importantes en su desplazamiento (ver imagen a color en el Anexo B). . . . 150
5.22. Diagrama de flujo correspondiente al submódulo de similitud por trayectoria. 156
5.23. Diagrama de flujo correspondiente al submódulo de similitud por recorrido. 160
5.24. Diagrama de flujo correspondiente al submódulo de similitud por velocidad. 162
5.25. Diagrama de clases de la capa de presentación. 170
5.26. Diagrama de clases de la capa de dominio. 170
5.27. Diagrama de clases de la capa de dominio. 171

6.1. Tabla de duración de cada iteración. 183
6.2. Recursos y costes utilizados en la elaboración de *Atravel*. 183
6.3. Representación del número de archivos y líneas de código, comentadas y en blanco, utilizadas en el desarrollo de *Atravel*. 184
6.4. Representación del entorno virtual 3D. 185
6.5. Resultados utilizando el entorno virtual 3D. 186
6.6. Resultados utilizando el entorno virtual 3D. 186
6.7. Resultados utilizando el entorno virtual 3D. 188
6.8. Resultados utilizando el entorno virtual 3D. 188
6.9. Resultados de la aplicación en el entorno virtual. 189
6.10. Gráfico de curva ROC en el entorno virtual (ver imagen a color en Anexo B). 189
6.11. Entorno real monitorizado (ver imagen a color en Anexo B). 190
6.12. Comportamiento de vehículos en la escena monitorizada (ver imagen a color en el Anexo B). 191
6.13. Comportamiento de peatones en la escena monitorizada (ver imagen a color en Anexo B). 191
6.14. Resultados obtenidos utilizando un entorno real. 193
6.15. Resultados obtenidos utilizando un entorno real. 193
6.16. Resultados obtenidos utilizando un entorno real. 194
6.17. Resultados obtenidos utilizando un entorno real. 195
6.18. Resultados obtenidos utilizando un entorno real. 195
6.19. Curva ROC resultante utilizando entorno real. 196

A.1. Ventana inicial de *Atravel*. 210
A.2. Selección del número de cámaras. 211
A.3. Componentes de flujo de vídeo. 212
A.4. Componentes representación 3D. 213
A.5. Inicio de la monitorización. 214

A.6. Visualización de monitorización. 214
A.7. Aspecto inicial de la interfaz principal. 215
A.8. Componente de visualización 3D de la interfaz principal. 215
A.9. Componente que contiene propiedades de los objetos reconocidos. 216
A.10. Visualización de una trayectoria del patrón. 216
A.11. Visualización de recorrido de un patrón. 217
A.12. Visualización de la trayectoria de todos los patrones. 217
A.13. Visualización del recorrido de todos los patrones. 218
A.14. Eliminación trayectoria de un patrón. 218
A.15. Eliminación recorrido de un patrón. 219
A.16. Eliminación de todo lo representado en el componente de visualización. . . 219
A.17. Representación de característica de un patrón. 220
A.18. Representación trayectoria de un objeto. 220
A.19. Representación recorrido de un objeto. 221
A.20. Representación similitud de un objeto. 221
A.21. Representación del log. 222

B.1. Vector unitario 223
B.2. Representación de colisión de dos bounding box. 224
B.3. Representación del recorrido de un objeto. 224
B.4. Representación del vector normal de una cara de un BB. 224
B.5. Representación del BB resultante de la fusión de dos BB. 225
B.6. Representación del recorrido completo de un objeto sin realizar cambios importantes en su desplazamiento. 225
B.7. Representación del recorrido completo de un objeto realizando cambios importantes en su desplazamiento. 226
B.8. Gráfico de curva ROC en el entorno virtual. 226
B.9. Entorno real monitorizado. 227
B.10. Comportamiento de vehículos en la escena monitorizada 227
B.11. Comportamiento de vehículos en la escena monitorizada (ver imagen a color en el Anexo B). 227
B.12. Comportamiento de peatones en la escena monitorizada. 228

Lista de algoritmos

1. Calcular ángulo existente entre dos vectores. 101
2. Calcular similitud entre dos vectores. 102
3. Calcular la distancia entre dos puntos. 102
4. Calcular la existencia de colisión entre dos BB 108
5. Calcular si un punto está en el interior de un BB. 112
6. Cálculo de la fusión de dos BB. 113
7. Intersección entre dos bounding box. 134
8. Calcular punto central de un bounding box. 136
9. Detección de un bounding box clave. 137
10. Generación optimización del recorrido 138
11. Generación del recorrido de un objeto. 143
12. Continuación del algoritmo *Generación del recorrido de un objeto*. 144
13. Obtención parcial del recorrido de un objeto. 147
14. Similitud atendiendo a los bounding box clave. 152
15. Similitud entre trayectorias atendiendo a todos sus BB. 154
16. Continuación del algoritmo *Similitud entre trayectorias atendiendo a todos sus BB*. 155
17. Continuación 2 del algoritmo *Similitud entre trayectorias atendiendo a todos sus BB*. 157
18. Similitud atendiendo al recorrido de un objeto. 159
19. Similitud atendiendo a la velocidad. 161
20. Fusión de patrones. 164

Agradecimientos

Me gustaría agradecer a mis padres Jesús Vicente y María del Carmen, por su apoyo incondicional durante todos estos años de estudio de la titulación. Ellos siempre han estado conmigo tanto en momentos buenos o malos, para aportarme los valores y conocimientos necesarios para superar las diferentes situaciones encontradas, gran parte de mí se la debo a ellos.
También me gustaría agradecer a amigos y compañeros, pues ellos también han puesto su granito de arena en la consecución de este proyecto.
A Javier Alonso Albusac Jiménez, David Vallejo Fernández y Carlos González Morcillo por brindarme la oportunidad de realizar un proyecto de este calibre a vuestro lado.
Por último, me gustaría hacer mención especial a mi hermano David, él es una de las personas más importantes en mi vida y la persona que más me ha apoyado en los momentos difíciles ante los que me he encontrado, es difícil de explicar, pero sin él creo que no hubiera sido posible la realización de este proyecto.
De corazón, muchas gracias a todos.

1

Introducción

En los últimos años los sistemas de vigilancia han ido cobrando cada vez más protagonismo en el área de la investigación de multitud de organizaciones e instituciones. Uno de los principales motivos que han llevado a organizaciones e instituciones a la utilización de sistemas de vigilancia, es preservar la seguridad dentro de su organismo. La seguridad es uno de los pilares básicos que debe tener toda orgnización para el correcto desarrollo y crecimiento de la misma.

La seguridad se puede dividir en cinco funciones según [VG08]: protección civil, prevención, intervención, investigación y recuperación. Desde los acontecimientos del 11 de Septiembre de 2001 en los Estados Unidos, se ha producido un cambio en el paradigma de la seguridad. La prevención de sucesos catatróficos se ha convertido en una función más importante que la investigación post-incidencia. Se buscan métodos o tecnologías eficaces que ayuden a prevenir eventos sospechosos y prevenir posibles daños a personas, lugares y datos.

En Canadá a pesar de que la sensación de inseguridad no es tan grave como en otro lugares de Estados Unidos, la lucha contra la delicuencia y el refuerzo de la seguridad nacional siguen siendo una prioridad para su gobierno. En Febrero de 2008 el gobierno destinó 630 millones de dólares para reforzar durante dos años la ley y el orden. También aportó 145 millones de dólares para reforzar la seguridad en las zonas fronterizas. Implantó una serie de medidas de seguridad, tales como el pasaporte electrónico, uso de la biometría para la expedición de visados, etc. Además el gobierno de Canadá anunció que el presupuesto del departamento de Defensa Nacional se incrementará un mínimo de 1.5 % anual para el año 2010-11.

Cada vez son más los organismos que basan gran parte de su seguridad en la utilización de sistemas de vigilancia, hasta llegar a la actualidad en la que prácticamente todo organismo de gran envergadura, necesita de estos sistemas para mantener el control total en la organización ante cualquier situación comprometida que pueda suceder en ésta.

Pero, ¿qué es un sistema de vigilancia?, se puede definir como el proceso organizado, selectivo y permanente de captar la información del exterior para después seleccionarla, analizarla, difundirla y transformarla en conocimiento válido para la toma de decisiones.

Podemos diferenciar tres tipos de sistemas de vigilancia según [VG08]:

1. **Activo**: Se realiza vigilancia sobre un área para reforzar la seguridad en el trabajo, en lugar de utilizar agentes de seguridad.

2. **Pasivo**: Un empleado visualiza un determinado número de monitores, mientras éste realiza otras tareas.

3. **Grabación**: Permite la recopilación de información para la investigación y como una prueba de evidencia. Las grabaciones se conservarán durante un tiempo dependiendo de las necesidades de espacio y el mantenimiento de registros.

Sin embargo, la utilización de sistemas de vigilancia conlleva la aparición de factores negativos, tales como la necesidad de contratar personal adecuado para lleve a cabo la tarea, analice y sintetice constantemente toda la información obtenida por los sistemas de vigilancia. Esta tarea es muy compleja y factores como la fatiga o la distracción de las personas encargadas de la supervisión, puede provocar la ignorancia de sucesos anómalos en el entorno observado.[VG08]

El ser humano tiene mayor facilidad para detectar objetos en movimiento con ciertas anomalías, y tiene mayores problemas a la hora de detectar objetos estáticos de tamaño pequeño situados en el escenario. Además un operario humano tiene dificultades para detectar anomalías en escenarios en los que existe multitud de objetos en movimiento. Un estudio [Smi04] afirma que el rendimiento de una persona visualizando imágenes a través de un monitor disminuye en gran medida tras 20 minutos de supervisión.

Además, en la actualidad los sistemas de vigilancia [VG08] incorporan cada vez un mayor número de sensores de captación de imagen, creando una gran infraestructura de videovigilancia formada por multitud de cámaras, gran tráfico de datos a través de la red, etc. Llevar a cabo esta tarea por personas es casi imposible, dado la alta concentración de datos que llegan a la central de control y la complejidad de análisis que esto supone. Normalmente la tarea del ser humano en las centrales de control, están relegadas al

manejo de situaciones de emergencia, seguimiento de alarmas de fuego, control de radio comunicaciones, etc.

Para paliar este problema de fiabilidad en la supervisión de la información captada por los sensores de vigilancia, aparece la incorporación a los sistemas de vigilancia de las Tecnologías de la información y comunicaciones (TIC). La utilización de las TIC en los sistemas de vigilancia han supuesto una revolución en la seguridad y vigilancia dando lugar a la aparición de los sistemas de vigilancia inteligente (SVI).

Sin embargo, estos sistemas tienen infinidad de variantes y configuraciones que tienen sus ventajas e inconvenientes. Una gran parte de los SVI utilizados son muy dependientes del entorno observado detectando situaciones anómalas muy concretas. Lo que provoca que tengan una configuración muy específica, poco flexible y escalable. Además no utilizan un sistema multicámara para la detección de objetos en la escena lo que provoca mayor riesgo en la detección de objetos debido a oclusiones parciales o totales. La mayoría de los SVI realizan un análisis de trayectorias en dos dimensiones lo que provoca una pérdida de información de la profundidad de los objetos dando lugar a detección de objetos deformes dependiendo de la perspectiva de la cámara. En SVI que realizan la detección de objetos en 2D aparecen importantes problemas en la exactitud de reconocimiento de patrones, ya que sería muy dependientes de la perspectiva de la cámara y suelen tener problemas para detectar la velocidad de los objetos.

Para ayudar a reducir estos problemas se ha realizado el Proyecto Fin de Carrera. Atravel, es un sistema que se encarga de realizar un aprendizaje automático de trayectorias y velocidades de objetos en tres dimensiones, en un entorno monitorizado para la detección de situaciones anómalas. El sistema se encarga de aprender la trayectoria y velocidad con la que los objetos que son detectados en un escenario monitorizado. A partir de ahí, se almacenan en una base de conocimiento y al mismo tiempo se analiza su comportamiento con respecto a los patrones anteriormente aprendidos. Los patrones almacenados representan las diferentes acciones detectadas en la escena monitorizada. La creación de los patrones se realiza tras la finalización del desplazamiento de un objeto a lo largo del entorno monitorizado. Su comportamiento es comparado con los patrones existentes y atendiendo a su comportamiento es clasificado. El proceso de aprendizaje de patrones es automatizado gracias a la realización de las tareas de clasificación y fusión de patrones. De esta forna, un comportamiento normal es representado mediante un patrón de forma más precisa a medida que se van sucediendo la clasificación de comportamientos similares y fusionando sus propiedades. Así pues, y tras un determinado período de aprendizaje de la aplicación, el usuario podrá visualizar el comportamiento de todo objeto que se desplace por el entorno, atendiendo a su clasificación con los patrones existentes.

1.1. ¿Qué es un sistema de vigilancia inteligente(SVI)?

Se puede definir como el proceso que emplea técnicas de inteligencia artificial (IA) para la observación, detección y análisis del comportamiento de objetos en un entorno observado. El principal objetivo de los sistemas de vigilancia inteligente es evitar al máximo posible el contacto de un operario humano con el sistema. De esta forma se reducen riesgos a la hora de detectar anomalías en el entorno observado, debido a que ahora, la responsabilidad de la supervisión recae sobre el sistema. Además permite la reducción de costes, ya que se reduce el número de personal contratado para la supervisión.

La utilización de sistemas de vigilancia inteligente tiene muchas ventajas [VG08]:

- Permite la activación de una alarma que estará gestionada por un operador humano, pudiendo realizar operaciones de movimiento o zoom sobre alguna de las cámaras para tener mayor seguridad de la situación. Proporciona una intervención al instante, en tiempo real en lugar de una intervención post-suceso.
- Se reduce el ancho de banda y el espacio de almacenamiento necesario en la transmisión o grabación, permitiendo grabar sólo los datos de acciones relevantes en la escena.
- Minimiza la labor del personal de seguridad en la continua monitorización.
- Proporciona velocidad a la hora de buscar un suceso relevante en los datos almacenados.

Los sistemas de vigilancia inteligente permiten detectar objetos en la escena observada y realizar un seguimiento sobre su comportamiento. Sin embargo, los SVI siguen siendo una tecnología nueva con bastantes limitaciones técnicas. Por ejemplo, a la hora de distinguir entre un comportamiento habitual o sospechoso es complicado para un sistema artificial, ¿cómo puede distinguir el sistema si una persona está corriendo porque llega tarde o porque acaba de cometer un robo y ha echado a correr. En la inteligencia artificial existe un compromiso entre el número de sucesos reconocidos y el número de falsas alarmas que ocurren. En los SVI cuanto más permisivo sea el umbral de detección de situaciones sospechosas, mayor es la probabilidad de que se produzca una amenaza real, pero mayor será el número de falsas alarmas disparadas innecesarias. Puesto que cada alarma necesita ser evaluada por personal de monitorización, es fundamental que el número de falsas alarmas se reduzca al máximo posible para evitar perder tiempo y productividad al personal

de monitorización. Mejorar el rendimiento de los SVI es uno de los retos que tienen los SVI a la hora de utilizar nuevas tecnologías de análisis de comportamientos.

La mayoría de los SVI están formados por cinco fases o capas en los cuales se desarrolla una tarea bien diferenciada. En las primeros fases se trata de obtener, procesar y clasificar la información captada por los sensores utilizados, y en las fases finales se realiza el reconocimineto de patrones y la toma de decisiones. Existen cinco niveles bien diferenciados que se encargar de realizar las siguientes tareas:

1. **Modelado del entorno.**

   En este nivel se trata de colocar los sensores de captación en el lugar adecuado para que la correcta vigilancia del escenario indicado. También se realiza el modelado de las normas sobre las acciones que pueden aparecer en el entorno, tales como el tipo de objeto, acciones, etc.

2. **Detección y clasificación.**

   El sistema obtiene información captada por los sensores y la clasifica siguiendo métodos y modelos que han sido desarrollado en el SVI.

3. **Tracking o seguimiento.**

   Una vez que el sistema ha obtenido los objetos que intervienen en el entorno, se realiza un seguimiento a éstos almacenando la información relevante para después analizar su comportamiento.

4. **Análisis de comportamientos.**

   En este nivel se realiza una comparación de comportamientos entre los nuevos objetos que aparecen en la escena observada y patrones de comportamiento, previamente adquiridos por el sistema, para detectar qué tipo de comportamiento realiza el objeto.

5. **Toma de decisiones.**

   El sistema debe detectar si los comportamientos de los objetos captados por el sistema son anómalos o si por el contrario su comportamiento es adecuado y demasiado frecuente, puede aportar nuevas características a los patrones ya definidos modificando parcialmente sus propiedades.

1.1.1. Evolución de los sistemas de vigilancia

Según [NC08] existen cinco etapas que marcan la evolución de los sistemas de vigilancia:

- **Sistemas analógicos CCTV basados en VCR** Es el tradicional sistema analógico de CCTV en el que todos los componentes son analógicos. El VCR se utiliza como almacenamiento de la información captada. Cada una de las cámaras esta conectada mediante un cable coaxial al VCR. La industria de la videovigilancia realizó especificaciones de 15fps (frames por segundo), 7.5fps, 3.75fps y 1.875fps. Para sistemas de vigilancia con múltiples cámaras era necesaria la utilización de un multiplexor que se encargada de combinar las señales procedentes de las diferentes cámaras, dando lugar a un única señal de vídeo multiplexada.

  Aunque los sistemas de vigilancia analógicos funcionaban bastante bien, éstos incluían algunas desventajas importantes, como el problema de la escalabilidad y la necesidad de mantener un operario que cambiase la cinta de VCR. También hay que tener en cuenta que la calidad de los sistemas análogicos no es la que existe en la actualidad y en muchos casos las cámaras captaban imágenes en blanco y negro.

- **Sistemas analógicos basados en DVR** Apareció a mediados de la década de 1990, la industria de la videovigilancia calificó la introducción del DVR como la primera revolución digital. Ahora el vídeo captado por cámaras de vídeo analógicas era digitalizado y comprimido en "discos duros" que tenían mucho más espacio que las VCR. La mayoría de los DVR estaban compuestos por varias entradas de vídeo, por lo general 4, 16 o 32 entradas, lo que implica la eliminación del multiplexor como componente del sistema de vigilancia.

  La introducción de los DVR en los sistemas de vigilancia proporcionó las siguientes ventajas:

  1. Eliminación de las cintas de vídeo y de los cambios constantes de éstas en el sistema.
  2. Mayor calidad de imagen.
  3. Mayor velocidad de búsqueda para posicionarse en un lugar determinado del vídeo almacenado.

- **Sistemas analógicos basados en red de DVR** Estos sistemas aparecieron debido a la incorporación de un puerto Ethernet en los DVR, permitiendo la monitorización de los vídeos de manera remota utilizando computadoras.

La introducción de estos sistemas en el mercado tuvo como principales ventajas:

1. Monitorización remota de los vídeos a través de computadoras.
2. Ejecución de operaciones en el sistema a distancia.

Sin embargo este tipo de sistemas presentaba también algunos inconvenientes, como la excesiva carga de tareas que desempeñaba el DVR como la digitalización de los vídeos de todas las cámaras, la compresión de éstos, la grabación, etc.

- **Sistemas de vídeo basados en un codificador de vídeo utilizando la red.** Un codificador de vídeo se conecta con las cámaras analógicas y es el encargado de digitalizar y comprimir el vídeo. Los datos son enviados por una red IP mediante un switch de red a una computadora, que ejecuta un software de gestión de vídeo para la monitorización y el almacenamiento de los datos. Este es un sistema de vigilancia utilizando la red, porque es envíado a través de una red IP. Las tareas que antes realizaba únicamente el DVR, ahora son compartidas por el codificador de vídeo y la computadora, el codificador se encarga de la digitalización y la compresión y la computadora se encarga de la grabación de los datos, proporcionando una mejor escalabilidad.

  Los sistemas de vigilancia basados en un codificador de vídeo presentan las siguientes ventajas:

  1. Escalabilidad del sistema, pudiéndolo expandir utilizando el mayor número de cámaras posible.
  2. Posibilidad de grabar la información en un computador remoto.
  3. Utilización de un hardware estándar, en cuanto a la red y al PC servidor para el grabado y la gestión de los vídeos.

- **Sistemas de vigilancia basados en cámaras IP.** Una cámara IP es simplemente una cámara de vídeo conectada a una red IP. En sistema de vigilancia basados en cámaras IP, todo el tráfico de datos circula sobre una red IP a través de switches y llegan a la computadora en la que se encuentra instalada el software de gestión de vídeo encargado del almacenamiento de la información. Se trata de un sistema totalmente digital ya que ninguno de los componenes que lo forman son analógicos.

  Uno de los beneficios más importantes que ofrece la utilización de cámaras IP es la digitalización de los datos captados dentro de la cámara, y a partir de aquí en todo el sistema se trabaja con datos digitales, esto proporciona al sistema alta calidad y uniformidad en los datos. Esto es importante ya que en la transformación de datos

analogicos a digitales (y viceversa) se pierde calidad en los vídeos. Además las señales analógicas pierden calidad al ser transportadas por cables de gran longitud y estar almacenados en cintas durante bastante tiempo también se produce un deterioro de los vídeos. Con la utilización de cámaras IP, pueden estar todas las cámaras conectadas sobre el mismo cable IP, y además de utilizar la red para enviar datos al PC servidor, también pueden recibir órdenes desde el servidor. Este detalle es muy importante ya que permite modificar los parámetros de configuración de las cámaras de forma remota.

Las principales ventajas que ofrece el sistema de vigilancia basado en cámara IP son:

1. Capacidad de utilizar cámaras de alta resolución (megapíxeles).
2. Calidad de imagen uniforme independientemente de la distancia.
3. Capacidad para utilizar funcionalidad inalámbrica.
4. Capacidad de realizar ajuste en la configuración de las cámaras de forma remota.
5. Total flexibilidad y escalabilidad.

A pesar de que una cámara IP puede ser comparada con una cámara analógica utilizando un codificador de vídeo, existen muchas funcionalidades que la cámara IP puede ofrecer frente a la cámara analógica. Algunas de estas funcionalidades pueden ser la incorporación de potencia de cálculo, la posibilidad de incluir inteligencia artificial en el interior de la cámara. Este aspecto es una nueva tendencia a estudiar dentro de la vídeo vigilancia debido a la gran necesidad de analizar y gestionar los vídeo con gran eficacia. Por último, añadir que las cámaras IP superan en muchos aspectos a las cámaras analógicas, tales como la calidad de imagen, resolución y la inteligencia integrada.

1.1.2. Factores que conllevan a utilizar los SVI

- Elevada tasa de acierto en el reconocimiento de objetos, frente al número de falsas alarmas producidas.
- Los vídeos captados por las cámaras son analizados por un ordenador.
- Por la naturaleza de las escenas a monitorizar, la utilización de SVI es más apropiada ya que si la escena presenta pocos cambios el personal de vigilancia tiende a agotarse, apareciendo factores negativos como la distracción y si la escena presenta muchos cambios resulta demasiado compleja para el análisis humano de las situaciones.

- Escalabilidad: Capacidad para añadir monitores de visualización aumentando el número de cámaras existentes en la red.

1.1.3. Sectores de aplicación

Los sistemas de vigilancia son utilizados originalmente por los servicios públicos [VG08], policía, medios de transporte y administración. Cuando algunas empresas comenzaron a utilizarlo en zonas estratégicas, tales como en refinerías, centrales nucleares, presas, y complejos farmaceúticos. Los casinos también aparecen como pioneros en el despliegue de sistemas de vídeo vigilancia. Hoy en día podemos encontrar cámaras de vigilancia tanto en lugares públicos como en privados, tales como edificios, tiendas, aparcamientos, estaciones de autobús, metro o tren, aeropuertos, carreteras, bancos, etc. La vídeo vigilancia ha sido desarrollada a diferentes escalas. Para la monitorización de pequeños delitos(por ejemplo, vandalismo, asalto, carteristas,etc) la vigilancia utilizada está enfocada más a la investigación post-incidencia. Este tipo de vigilancia requiere de recursos tecnológicos simples como tecnologías analógicas que no implican la aparición de vigilancia inteligente. Este tipo de sistemas se pueden encontrar en tiendas pequeñas o en algunos hogares.

La vigilancia en urbanizaciones, aparcamientos o en tiendas más grandes, necesitan sistemas de vigilancia con mayor número de cámaras. El propósito de esta vigilancia es la investigación a posteriori, es decir, en caso de que se produzca un suceso sospechoso obtener la información capturada y analizarla. Sin embargo, lo ideal sería que se pudiera contar con un sistema de vigilancia que avise en tiempo real mediante algún tipo de alerta de la aparición de un suceso anómalo, para una intervención inmediata. La aparición de sistemas que analicen las imágenes captadas en tiempo real es un gran reto para estos sistemas.

A veces los sistemas de vigilancia tienen lugar en grandes ciudades, medios de transporte, campus universitarios, y demás eventos (festivales, cumbres económicas, juegos olímpicos, etc) convirtiéndose en sistemas de vigilancia a gran escala. Para ello se requieren cientos o miles de cámaras de vídeo que deben dar respuesta a peticiones de diferentes organismos como el departamento de policía, servicios de emergencia, etc. Teniendo el gran número de cámaras utilizadas y el gran número de intervenciones que se realizan, surge la necesidad de incorporar en el análisis de comportamientos un procesamiento automático del flujo de vídeo, generando alarmas cuando se produzca algún evento sospechoso.

Debido a que cada vez los SVI cuentan con más presupuesto invertido, resulta más sencillo añadir software de análisis de vídeo a los sistemas de vigilancia. Recientemente se están empezando a utilizar sistemas de vigilancia en unidades móviles, ambulancias,

patrullas, autobuses, etc. Sin embargo existen problemas en la transmisión de los flujos de vídeo desde la unidad a la central de control a través de red inalámbrica.

Gobierno y seguridad pública.

Las instituciones gubernamentales deben garantizar la seguridad de la población y de las infraestructuras públicas. La utilización de los SVI tiene lugar en:

- Infraestructuras sensibles.
- Edificios y entornos de instituciones gubernamentales.
- Laboratorios.
- Bases militares.
- Cárceles.

Muchos sectores del gobierno requieren de SVI para: la identificación de personas y vehículos, conteo de personas, reconocimineto de conducta sospechosa o violenta (peleas, delitos, etc), detección de intrusos o monitorización de calles. Los análisis más sofisticados prevén que puedan determinar las emociones de personas. [VG08]

Educación.

Los sistemas de vigilancia van aumentando en las instituciones académicas. Se utilizan para supervisar la seguridad de profesores y estudiantes y protegerlos de posibles delitos y actos de vandalismo. Los trágicos asesinatos en el colegio Columbine en Estados Unidos y en Quebec en la universidad politécnica de Dawson, han puesto en relieve la importancia de la seguridad en las instituciones académicas. La monitorización de estos recintos puede ser de gran tamaño, sobretodo en el caso de las universidades, que están compuestos por varios edificios, diferentes zonas de acceso y monitorización de parkings. En estos entornos los SVI se utilizan para:

- Controlar el perímetro de los edificios de la institución.
- Controlar las vías de accesos a la institución.
- Detectar y hacer un seguimiento de actos vandálicos(delitos, comportamientos violentos, etc).

- Reconocimiento de las matrículas de vehículos.
- Como evidencia en las investigaciones penales.

En la mayoría de instituciones académicas hoy en día mantienen una infrestructura IP, que puede ser aprovechada para la instalación de sistemas de vigilancia digital. [VG08]

Establecimientos comerciales.

Los SVI en los comercios están en continuo crecimiento [VG08], se utilizan tanto para monitorizar el interior del comercio como el exterior, o el aparcamiento. Incluso los pequeños comercios incluyen cámaras en su interior para mantener evidencias en caso de ocurrir un delito o incidente. Los sistemas de vigilancia en este sector están dirigidos a:

- Monitorización de registros y transacciones (robo de empleados y el fraude).
- Protección de infraestructuras o de los bienes materiales.
- Seguimiento de mercancias.
- Proteger a los clientes y trabajadores.
- Control de acceso a las zonas privadas o almacenes.
- Control de situaciones de emergencia (incendio, alarmas, etc.).
- Monitorización de aparcamientos, vehículos, entradas o salidas.

En este sector el análisis de vídeo se realiza para contar el número de clientes que entran y salen del comercio, analizar su comportamiento y elaborar estadísticas sobre los hábitos de consumo.

Medios de transporte.

La seguridad en aeropuertos, estaciones de tren o autobús, puertos marítimos y demás sistemas de transporte es una de las labores más crítica para la economía de un país [VG08]. Un incidente en estos lugares puede tener resultados catastróficos y unas pérdidas considerables. Teniendo en cuenta la gran cantidad de personas que utilizan a diario los medios de transporte hace que la seguridad en éstos tenga que ser extraordinaria. Los sucesos terroristas ocurridos en estos lugares han exaltado la importancia de la seguridad

en los medios de transporte, considerándola como uno de los principales retos para los gobiernos.

El software de análisis debe realizar diferentes funciones como: detectar la aparición de intrusos en zonas de paso restringido, detección de personas que entran por zonas exclusivamente de salida, detección de equipaje abandonado, reconocimiento de caras de personas, detección de matrículas de vehículos en la entrada de aparcamientos, detección de comportamientos sospechosos (instalación de vagabundos en las infrestructuras, actos de vandalismo, realización de graffitis en las paredes de las infraestructuras, deteccion de personas que invaden los carriles de medios de transporte.

Este sector plantea importantes retos a los SVI tales como analizar exhaustivamente el comportamiento de personas antes grandes aglomeraciones y factores de limpieza del espacio observado, como aparición de polvo en las cámaras, etc. En los espacios exteriores también plantea retos en los que las cámaras deben funcionar correctamente ante condiciones climáticas cambiantes, tales como la aparición de lluvia, viento, nieve, niebla, etc.

Aeropuertos.

Tras los sucesos del 11 de Septiembre de 2001 [VG08], las medidas de seguridad de los aeropuertos se hicieron más estrictas sobretodo en los grandes aeropuertos y se abrieron a la utilización de nuevas tecnologías. Para los aeropuertos la seguridad de centrarse en el control de las áreas de acceso, sobre todo en las áreas de acceso de los aviones y la seguridad de los pasajeros y trabajadores, también se utiliza para aportar seguridad en las zonas de aparcamiento, vías de acceso a las pistas, etc. En la actualidad los sistemas de vigilancia en los aeropuertos carecen la mayoría de análisis inteligente, las imágenes son archivadas y almacendas para ser revisadas en caso de que ocurra alguna incidencia. Sin embargo en los grandes aeropuertos se están utilizando cada vez más, para funciones como, conteo de personas, reconocimiento de matrículas de los vehículos situados en el aparcamiento para monitorizar posibles delitos, y reconcimiento de caras de personas con el objetivo de reconocer y evitar el acceso a las infraestructuras de posibles terroristas.

Estaciones y transporte público.

El objetivo de los sistemas de videovigilancia en estaciones de tren o autobús y demás medios de transporte es controlar las vías de acceso, las plataformas o andenes, vías, túneles, etc. Los sistemas de vigilancia en estos lugares están comenzando a utilizar la inteligencia

artifical anticipándose al reconocimiento de equipaje sospechoso o comportamientos inadecuados, debido a la comprobación de la eficacia del análisis de vídeo en estos sistemas. En la utilización de los SVI más avanzados se encuentran Reino Unido, Almenania, Francia y Estados Unidos.

Puertos Marítimos.

Al igual que en el resto de medios de transporte los sistemas de vigilancia inteligente son utilizados en las principales áreas de acceso y reconocer a vehículos en el perímetro de zonas del puerto para garantizar la seguridad de los pasajeros y trabajadores, así como el cuidado de las infraestructuras. Sin embargo, la utilización de SVI en estos lugares suponen un importante desafío para los SVI porque la monitorización se realiza muy cerca del agua. Los reflejos que el agua despide, o el movimiento brusco de las aguas, son variaciones medioambientales que deben perturbar los algoritmos de análisis, especialmente para la detección de movimiento [VG08].

Bancos de crédito.

La videovigilancia es ampliamente utlizada por la banca en general. La existencia de cámaras de vídeo suponen una primera barrera para el robo a mano armada. En caso de producirse un robo, el material de vídeo archivado sirve para identificar o investigar sobre el autor del delito. También permite controlar el comportamiento de los empleados en caso de que no tengan una conducta adecuada. La introducción de los SVI pueden aumentar la eficacia en la monitorización para analizar el comportamiento de toda persona que aparezca en el entorno observado, e incluso pueden realizar un reconocimiento facial para reconocer a sus trabajadores o clientes [VG08].

Industria del juego o casinos.

En los centros de ocio como los casinos la videovigilancia ha estado siempre muy presente para proteger a los clientes y las infraestructuras de éstos [VG08]. También han sido utilizados para evitar fraudes realizados por los clientes, robos en las cajas registradoras, etc. Dado que la seguridad en un casino viene dada por observar el comportamiento de personas en un entorno lleno de gente, implica la necesidad de utilizar SVI para hacer la tarea más fácil a los miembros de seguridad. El reconocimiento de las caras, los gestos y comportamientos adecuados son algunas de las funciones útiles que se deben llevar a cabo en el análisis de este entorno.

Otros.

Los sistemas de videovigilancia están siendo utilizados en otros entornos como almacenes, bloques de pisos, urbanizaciones, etc. El objetivo de estos sistemas es garantizar la seguridad de los ciudadanos o de los trabajadores [VG08]. Sin embargo, también pueden ser utilizados como ayuda algunas intervenciones quirúrgicas. Estos sistemas en combinación con herramientas de medición, pueden ser utilizados para controlar a un paciente de forma remota. La empresa estadounidense Indigo Vision, ha desarrollado un sistema de análisis inteligente para detectar el movimiento de aterrizaje de helicópteros en un lugar específico de un hospital americano. Cada vez son más frecuentes los sistemas de vídeo vigilancia en los hogares privados y pequeños comercios.

1.1.4. Análisis de trayectorias

Normalmente en los entornos monitorizados, los objetos que aparecen son dinámicos, lo que dificulta la tarea de análisis de su comportamiento. Para reconocer la aparición de situaciones anómalas en la escena, existe la necesidad de analizar la trayectoria de los objetos. La mayoría de sistemas de vigilancia realizan un seguimiento utilizando una única cámara y almacenando la trayectoria en 2D, lo que desemboca en una importante imprecisión a la hora de detectar situaciones anómalas. La pérdida de una dimensión y la oclusión total o parcial a la hora de analizar una trayectoria puede eliminar información importante sobre el comportamiento del objeto.

El análisis de trayectorias de cada uno de los objetos que aparecen en el entorno de monitorización permite obtener información muy valiosa acerca de su comportamiento. Con la obtención de la trayectoria de un objeto se puede obtener su localización en cada instante de tiempo durante su desplazamiento en la escena. La representación de esta información en 3D permite realizar un análisis más exhaustivo sobre el comportamiento de un objeto.

En este proyecto se opta por la utilización de un sistema multicámara, para combatir al máximo posible la oclusión de objetos, y la representación de trayectorias en 3D para dotar al sistema de la precisión necesaria en la detección de situaciones anómalas. Para la detección de las situaciones anómalas, el sistema se encargará de realizar una clasificación del comportamiento de los objetos detectados en el entorno de monitorización. La clasificación de los objetos en las diferentes categorías es necesario dotar al sistema de mecanismos de comparación de similitud para poder realizar una correcta clasificación. El reconocimiento de las diferentes situaciones serán almacenadas como patrones y serán

clasificadas según su comportamiento. Además, para desarrollar un sistema óptimo se ha de dotar de la capacidad de fusión de los diferentes patrones reconocidos con anterioridad.

1.1.5. Análisis de velocidad

. Para clasificar el comportamiento de un objeto que se desplaza por un entorno de monitorización, además de realizar un análisis de la trayectoria del objeto, es interesante realizar un estudio de la velocidad a la que se desplaza.

El estudio de la velocidad a la que se desplaza un objeto permite obtener información complementaria muy importante para la clasificación de objetos. De manera, que dependiendo de su velocidad un objeto puede ser clasificado de distinto tipo o clase. También puede resultar determinante en la detección de situaciones anómalas, como por ejemplo, un ladrón que quiere escapar tras cometer un delito. Para el estudio de la velocidad con la que un objeto se mueve a lo largo de la escena, únicamente es necesario conocer la variación espacial con respecto al tiempo.

Algunos proyectos realizan un análisis de la velocidad media con la que se desplazan los objetos, mientras que otros realizan un análisis de la velocidad del objeto por tramos o fragmentos. Este último análisis proporciona mayor flexibilidad y robustez, ya que permite distinguir diferentes zonas por las que los objetos suelen llevar mayor o menor velocidad, proporcionando información más completa acerca del comportamiento del objeto.

En este proyecto se realiza un análisis de la velocidad a la que se desplazan los objetos a lo largo del entorno de monitorización atendiendo a la variación espacial con respecto al tiempo, por tramos o fragmentos. Es decir, el sistema analizará la velocidad a la que se desplazan los objetos en las diferentes zonas que componen su movimiento sobre la escena. La realización de este análisis proporciona al sistema de mayor robusted, flexibilidad y precisión en el estudio del comportamiento de los objetos detectados sobre el entorno de monitorización.

1.2. Problemática

En este proyecto se desarrollan principalmente las capas de análisis de comportamientos y toma de decisiones, pertencientes a un sistema de vigilancia inteligente. Existen diferentes problemas que aparecen en el desarrollo de estas funciones. En la capa de análisis de comportamientos el principal problema es la necesidad de entrenar al sistema para que el correcto funcionamiento de éste a la hora de detectar objetos en el entorno observado

y la posterior clasificación en los patrones correctos. Esto conllevaría la presencia durante un determinado período de tiempo (que vendrá determinado por la complejiad del entorno observado) de personal cualificado para llevar a cabo el entrenamiento del sistema. Además la definición de los patrones debería realizarse de forma manual, lo que puede derivar en errores humanos a la hora de crear los patrones, debido a la poca flexibilidad de la configuración. Los principales inconvenientes de un entrenamiento del sistema de forma manual son:

- **Flexibilidad del sistema**: Para la elaboración de un sistema de vigilancia inteligente es importante, eliminarl al máximo la intervención de personal humano, es decir, es necesario que el sistema sea lo más autónomo e independiente posible.
- **Escalabilidad del sistema.** Es una propiedad que deben tener los SVI para que puedan ser establecidos en un entorno monitorizado, obteniendo unos resultados óptimos, ya tenga una complejidad mayor o menor en la escena. Además deben proporcionar la fiabilidad necesaria para sobreponerse a posibles cambios en el entorno monitorizado.

1.3. Estructura del documento

Este documento ha sido elaborado siguiendo la normativa de Proyecto Fin de Carrera (PFC) en la Escuela Superior de Informática (ESI) de la Universidad de Castilla-La Mancha (UCLM). Está compuesto de las siguientes capítulos:

Capítulo 1: Introducción

Se realiza una breve introdución del ámbito en el que se encuentra *Atravel*

Capítulo 2: Objetivos

En este capítulo se especifican las necesidades que este proyecto debe cubrir.

Capítulo 3: Antecedentes

Se realiza un estudio sobre la evolución de los SVI centrando la atención en las capas de análisis de comportamientos y toma de decisiones.

Capítulo 4: Metodología de Desarrollo y Herramientas

En este capítulo se especifican los métodos utilizados para el desarrollo del proyecto, así como las herramientas necesarias para su implementación.

Capítulo 5: Arquitectura de Atravel

Se especifica la metodología utilizada en el desarrollo del proyecto. Se desarrolla un enfoque Top-Down compuesto por diferentes módulos.

Capítulo 6: Evolución y Resultados

Se explican las distintas iteraciones que se han llevado a cabo, la evolución y los resultados obtenidos en el desarrollo del proyecto.

Capítulo 7: Conclusiones y Propuestas

En esta sección se especifican las conclusiones establecidas y posibles propuestas futuras para la continuación en el desarrollo de futuros proyectos.

Bibliografía

En este capítulo se especifican las fuentes que se han utlizado en la elaboración del proyecto.

Anexo

En esta sección se explica el manual de usuario correspondiente a la utilización del proyecto desarrollado.

2

Objetivos

El principal objetivo de **Atravel** es desarrollar un sistema de aprendizaje y análisis de trayectorias en 3D y velocidades en entornos reales. Atravel podrá ser utilizado en cualquier sistema de vigilancia inteligente aportando flexiblidad, escalabilidad y autonomía al producto final.

A continuación, se detallan los diferentes subobjetivos funcionales que se han marcado para el desarrollo de **Atravel** :

1. **Representación multicámara del entorno monitorizado mediante flujos de vídeo.** Atravel soportará la reproducción de flujo de vídeo que represente la monitorización de un entorno observado real, para el análisis posterior.

2. **Representación del entorno monitorizado en 3D.** El sistema muestra el entorno monitorizado en 3D con capacidad de rotación, traslación y zoom para que el usuario pueda visualizar los objetos que se encuentran en la escena en todo momento.

3. **Almacenamiento de la información.** El sistema recogerá la información 3D correspondiente a cada objeto y será almacenada, para el posterior análisis de su comportamiento. El sistema contará con una base de conocimiento para el almacenamiento de los patrones reconocidos y otra base de conocimiento en la que se guardará la información de los objetos activos en el entorno monitorizado. Además este objetivo será clave para la posterior representación de los objetos en 3D en el componente gráfico.

4. **Definición de patrones.** El sistema realizará la definición de patrones, que servirán como modelo para la detección de situaciones anómalas. Los patrones reconocidos

serán almacenados en una base de conocimiento, para el posterior análisis de comportamientos.

5. **Proceso de clasificación.** El sistema estudiará el comportamiento de los objetos detectados en el entorno de monitorización y los clasificará en diferentes grupos atendiendo a su comportamiento dentro de la escena. Para la posterior detección de situaciones anómalas en la escena.

6. **Aprendizaje supervisado.** Atravel proporciona la opción al usuario para poder clasificar manualmente el tipo de patrón, en caso de que éste sea de tipo "No Catalogado". De esta manera el usuario podrá adelantarse al aprendizaje automático de patrones facilitando el desarrollo de esta tarea.

7. **Similitud con patrones** Todo objeto detectado debe ser comparado con respecto a los patrones existentes en la base de conocimiento con el objetivo de realizar una posterior clasificación adecuada de su comportamiento. La similitud es realizada atendiendo a su *trayectoria*: Atravel obtiene la trayectoria realizada por los objetos activos en el entorno observado y realiza una comparación con los patrones existentes en la base de c onocimiento extrayendo el porcentaje de similitud con cada uno de ellos. Atendiendo a su *recorrido*:Atravel obtiene el recorrido seguido por los objetos activos en la escena y realiza una comparación con los patrones existentes en la base de conocimiento extrayendo el porcentaje de similitud por recorrido con cada uno de ellos. El recorrido descrito por el desplazamiento de un objeto viene determinado por la zona en la que realiza su movimiento, es decir, el recorrido representa el espacio en la escena de forma menos restrictiva que la trayectoria. De forma que el recorrido representa el espacio por el cuál ha tenido lugar el desplazamiento del objeto sin representar específicamente la trayectoria. Atendiendo a su *velocidad*: El sistema obtiene la velocidad de los objetos activos que hay en la escena y realiza una comparación con la velocidad de los patrones reconocidos en el sistema. El calculo de la velocidad con la que se desplaza un objeto a lo largo de la escena, es mediante el cociente de la variación de espacio con respecto al tiempo.

8. **Fusión de patrones.** El sistema realizará una fusión de patrones automática con el objetivo de liberar memoria e ir aprendiendo ante posibles modificaciones en el entorno. Este objetivo es imprescindible para la consecución del aprendizaje automático. La fusión de patrones proporciona una optimización de la base de conocimiento, reduciendo el número de éstos y enriqueciendo los diferentes comportamientos de objetos, mediante el uso de patrones.

9. **Detección de situaciones anómalas.** El sistema analizará el comportamiento de los objetos que aparezcan en el entorno monitorizado e informará de las situaciones anómalas que sucedan en base al análisis de trayectorias y velocidades realizado.

10. **Interfaz gráfica parcialmente desacoplada.** La aplicación contará con una interfaz que permitirá al usuario visualizar el análisis de comportamientos de los objetos. El usuario podrá interactuar con el sistema cambiando la funcionalidad que desee visualizar. El módulo de análisis de comportamiento de los objetos está desacoplado de la interfaz pudiendo ser utilizado en proyectos futuros.

11. **Escalabilidad.** Atravel debe garantizar su buen comportamiento en cualquier entorno monitorizado, ya se más complejo o sencillo, así como su buen funcionamiento con la utilización de un numero determinado de sensores de captación.

12. **Estándares libres.**

    La aplicación será desarrollada con estándares y software libre, con el objetivo de ser reutilizada por distintos usuarios empleando alguna licencia de libre uso. Además la comunidad de usuarios podrán elaborar nuevas versiones de la aplicación con el objetivo de completar la aplicación con nueva funcionalidad.

En conclusión, **Atravel** es un sistema de aprendizaje y análisis de trayectorias y velocidades flexible y escalable, capaz de detectar y avisar de comportamientos anómalos en entornos reales monitorizados. Debido a que los SVI se encuentran en continuo crecimiento, **Atravel** cuenta con un diseño adaptable con la capacidad de incorporar nuevos módulos en el futuro, como por ejemplo, reconocimiento de caras de humanos, o detección de matrículas de vehículos, etc.

3

Antecedentes.

3.1. Introducción general.

En esta sección se pretende poner en contexto al lector sobre las tecnologías, lenguajes de programación, base matemática y aspectos relevantes, necesarios para el correcto entendimiento de Atravel.

3.1.1. Python.

En la actualidad existen multitud de lenguajes de programación, cada uno con sus ventajas e inconvenientes. Para el desarrollo de este proyecto se ha decidido utilizar el lenguaje de programación de **Python** por los siguientes motivos [AMV09]:

- Es un lenguaje muy expresivo y además muy compacto. El número de líneas de código de un programa suele ser considerablemente más reducido utilizando Python que utilizando otro lenguaje de programación como C. Por este motivo, muchos autores catalogan a Python como un lenguaje de muy alto nivel.
- Se trata de un lenguaje muy legible, gracias a su compactación.
- Ofrece un entorno interactivo que permite comprobar la funcionalidad de unas determinadas líneas de código de manera instantánea.
- Es un lenguaje que puede ser utilizado como lenguaje imperativo procedimental o como lenguaje orientado a objetos.

- Posee estructuras de datos definidas muy ricas que aligeran la complejidad del código a la hora de desarrollar un algoritmo. Tales como por ejemplo, listas, diccionarios, etc. Además, la manipulación de estas estructuras es muy sencilla e intuitiva.
- El entorno de ejecución de Python detecta errores de programación proporcionando con exactitud el origen del error.
- El intérprete de Python es totalmente gratuito y proporciona versiones para cualquier plataforma ya sea Linux, Windows o Macintosh. El intérprete se puede descargar desde la página web[1].

En este apartado merece una mención especial, dos de las estructuras más importantes que ofrece e incorpora *Python* como son las **listas** y **diccionarios**, ya que han sido las dos estructuras de datos por motivos de flexibilidad, organización y facilidad de uso.

3.1.2. OpenCV

OpenCV (Open Source Computer Vision Library) [BK08] es una biblioteca de libre uso para aplicaciones en tiempo real de visión por computador. Fue creado por un grupo de investigación de *Intel* en desarrollo de un proyecto para la creación de aplicaciones de uso intensivo de la CPU. *OpenCV* está diseñado para la creación de aplicaciones de visión por computador de forma rápida y sencilla. Abarca un gran ámbito de áreas de aplicación como: calibraciónde cámaras, robótica, seguridad, interfaces de usuario, etc.

Está compuesto por cuatro componentes:

- **CV**: contiene los algoritmos para el procesamiento de imágenes de alto nivel.
- **MLL**: se trata de una biblioteca de aprendizaje, cuenta tanto con herramientas de clúster como con una gran variedad de clasificadores estadísticos.
- **HighGUI**: este componente se encarga de las operaciones de almacenamiento, reproducción (imágenes y vídeo) y entrada y salida.
- **CXCore**: es el núcleo que da soporte a los componentes anteriores. Contiene las estructuras de datos básicas, soporte de XML y funciones de visualización.

[1] http://www.python.org

3.1.3. OpenGL.

Es la abreviatura de Open Graphics Library (Biblioteca de gráficos abierta).Es un conjunto de bibliotecas que son utilizadas mediante lenguajes de programación para conseguir un interfaz entre las aplicaciones y el hardware gráfico. Se utiliza para la generación gráficos en 2D y 3D con un coste computacional muy reducido. Siendo muy adecuada su utilización cuando se tiene la necesidad de representar constantemente la actualización de cambios producidos en la aplicación. La biblioteca de OpenGL [Shr09] está compuesta por unas 150 instrucciones que aportan la funcionalidad necesaria para el desarrollo de aplicaciones. OpenGL se caracteriza por:

- OpenGL ofrece alta portabilidad ya que presenta independencia con la plataforma hardware y el sistema operativo utilizado.
- Se trata de un sistema procedural y no descriptivo.
- La actualización de los objetos creados o modificados se realiza con una rapidez inmediata, esto tiene gran importante para la creación de aplicaciones que necesitan ser actualizadas con gran frecuencia.
- Incluye primitivas gráficas, como puntos, líneas o polígonos.
- Funcionalidades gráficas de iluminación y sombreado, aplicación de texturas, animaciones, etc.
- Incluye contextos de visualización o rendering. Además OpenGL de incorporar funciones para la visualización o rendering, incluye una biblioteca llamada GLU que contiene aquellas funciones que no son estrictamente de visualización.
- No incluye ninguna función para la gestión de ventanas, la interacción continua del usuario con la aplicación y el manejo de ficheros.

Debido a que OpenGL es independiente de la gestión de ventana y del sistema operativo utilizados, es necesaria la utilización de bibliotecas que pongan en contacto el sistema operativo y la gestión de ventanas con OpenGL. Estas bibliotecas son independientes a OpenGL y diferentes para cada sistema operativo. Existe unas herramientas que facilitan esta unión entre OpenGL, el sistema operativo y gestión de ventanas, se denomina GLUT. Otras de las bibliotecas más utilizadas son WGL y GLX. OpenGL ofrece diferentes funciones gráficas:

- Doble buffer.

- Z-buffer.
- Transformación de matrices.
- Efectos de luz.
- Propiedades del material.
- Smooth shading.
- Transparencias.
- Mapeo de texturas.
- NURBS.
- Niebla.
- Extensiones OpenGL.

3.1.4. Pyside.

Pyside[2] es una biblioteca de python para el desarrollo de interfaces gráficas de usuario multiplataforma. Fue desarrollada por Nokia y utiliza el lenguaje de programación Python para la manipulación del framework de Qt. Qt es un potente framework de interfaz de usuario (IU) multiplataforma. Permite el desarrollo e implementación de aplicaciones en multitud de sistemas operativos diferentes, sin necesidad de modificar el código fuente. Además utiliza un lenguaje moderno y dinámico como es Python. Combinando la potencia de Qt con Python, Pyside proporciona la utilización del framework de Qt para los desarrolladores de software utilizando Python. Pyside presenta una plataforma de primer nivel para el desarrollo rápido de aplicaciones para los principales sistemas operativos actuales. Pyside ha sido publicado debido a inexistencia de bindings de Qt utilizando Python de carácter libre. Pyside está elaborado bajo la licencia LGPL 2.1 permitiendo el software libre o abierto y el desarrollo de software propietario. Pyside está basado en PyQt que es una tecnología existente que al igual que Pyside permite la manipulación del framework de Qt utilizando el lenguaje Python, pero PyQt tiene licencia GPL. La necesidad de tener una plataforma con licencia LGPL desembocó en la creación de Pyside.

[2] www.pyside.org

3.2. Fundamentos físicos y matemáticos.

En esta sección se van a explicar los conceptos físicos y matemáticos que se han sido utilizados para el desarrollo de determinadas áreas en el proyecto. Debido a que el desarrollo del proyecto tiene lugar en un espacio euclídeo $\Re^3$, se especificará la definición de los elementos físicos que nos podemos encontrar en él.

3.2.1. Velocidad

La velocidad[3] es una magnitud física que expresa la distancia recorrida por un objeto por unidad de tiempo. Se representa como $\vec{v}$ o v. Su unidad en el sistema internacional es $\frac{m}{s}$. Para definir la velocidad es necesario considerar la dirección de desplazamiento y el módulo, el cual se denomina rapidez o celeridad. La velocidad es un vector, tiene magnitud y dirección [WIL03]. La velocidad puede tomar valores positivos o negativos, lo que determina la dirección. La velocidad puede definirse de la siguiente manera:

$$\vec{v} = \frac{\Delta x}{\Delta t} = \frac{(x_2 - X_1)}{(t_2 - t_1)} = \frac{(x - x_0)}{(t - t_0)} = \frac{(x - x_0)}{t} \tag{3.1}$$

y de forma simplificada es definida como:

$$\vec{v} = x_0 + \vec{v}t \tag{3.2}$$

Donde x_0 es la posición inicial, x es la posición final y $\Delta t = t$ es $t_0 = 0$.

3.2.2. Punto.

El punto [JEM04] es el ente geométrico más simple que nos podemos encontrar en un espacio euclidiano. No tiene longitud, área o volumen. Está compuesto por una terna ordenada de valores que representa una posición en el espacio con respecto a un sistema de coordenadas preestablecido. Suele ser representado como una terna de valores (a_x, a_y, a_z) y representa la posición del punto con respecto a los tres ejes, siendo a_x la posición con respecto al eje X, a_y representa la posición del punto con respecto al eje Y y a_z representa la posición del punto con respecto al eje Z. En la figura 3.1 podemos ver un ejemplo de punto en el espacio:

[3] http://es.wikipedia.org/wiki/Velocidad

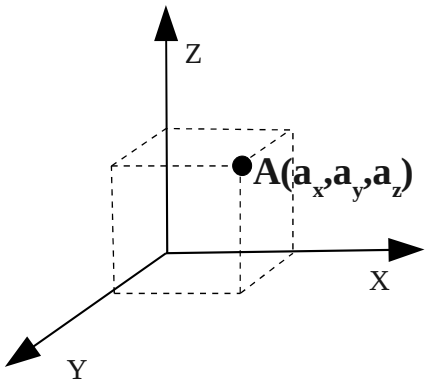

FIGURA 3.1: Punto en un espacio euclídeo $\Re^3$.

3.2.3. Vector.

Un vector [JEM04] es un ente geométrico que se caracteriza por tener longitud, dirección y sentido. Normalmente los vectores vienen determinados por el segmento existente entre dos puntos por ejemplo, dados los puntos A = (a_x, a_y, a_z) y B = (b_x, b_y, b_z) el vector resultante sería el segmento orientado situado entre ambos puntos $\vec{v} = \vec{AB}$, este vector es diferente del vector resultante $\vec{v} = \vec{BA}$, esto se debe a que aunque la longitud y dirección de ambos es idéntica, el sentido es totalmente opuesto. El vector $\vec{v} = \vec{AB}$ tiene como origen, el punto A y como destino el punto B, mientras que el vector $\vec{v} = \vec{BA}$ tiene como origen el punto B y como destino el punto A. Para obtener las componentes de un vector conociendo los dos puntos del espacio que lo comprenden, se realiza la diferencia entre el punto de destino con respecto al punto de origen. De manera que dados los puntos A y B, el vector $\vec{v} = \vec{AB}$ sería: $\vec{v} = (b_x, b_y, b_z) - (a_x, a_y, a_z)$. Las componentes del vector $\vec{v}$ serían:

- $\vec{v}_x = (b_x - a_x)$
- $\vec{v}_y = (b_y - a_y)$
- $\vec{v}_z = (b_z - a_z)$

En la figura 3.2 se puede ver la representación de un vector.

Dado un vector $\vec{v} = (v_x, v_y, v_z)$, podemos obtener la longitud o módulo de un vector resolviendo la siguiente ecuación:

$$\|\vec{v}\| = \sqrt{v_x^2 + v_y^2 + v_z^2} \tag{3.3}$$

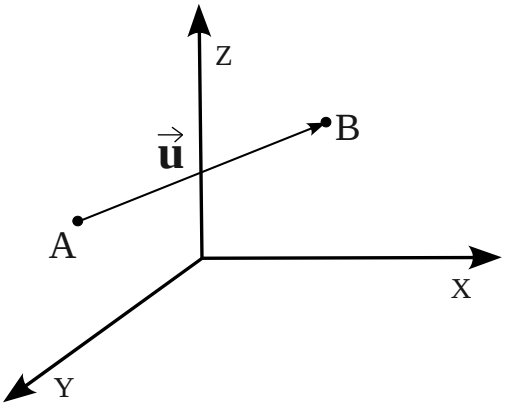

FIGURA 3.2: Vector en un espacio euclídeo $\Re^3$.

Un vector es **unitario** cuando su módulo vale 1, es decir, su módulo es la unidad. Dado un vector $\vec{v}$ se puede obtener su vector unitario (ver figura B.1) resolviendo la siguiente expresión matemática:

$$\vec{u} = \frac{\vec{v}}{\|\vec{v}\|} = (\frac{\vec{v}_x}{\|\vec{v}\|}, \frac{\vec{v}_y}{\|\vec{v}\|}, \frac{\vec{v}_z}{\|\vec{v}\|}) \tag{3.4}$$

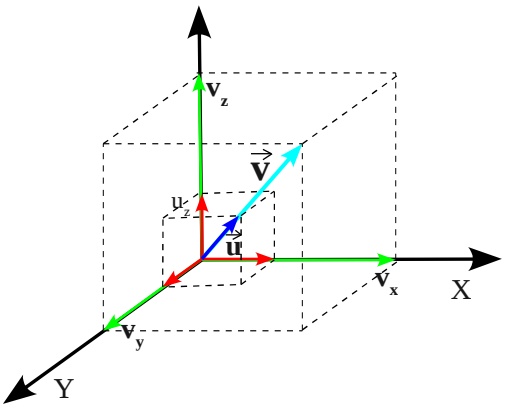

FIGURA 3.3: Vector unitario (ver en Anexo B imagen a color).

Se dice que un vector está **normalizado** cuando su módulo es la unidad, es decir, cuando se trata de un vector unitario.

3.2.4. Equivalencia de vectores.

Una de las características más importantes de los vectores es la definición de equivalencia [PJS]. Se dice que dos vectores son equivalentes cuando su longitud y su dirección son idénticas. Como se puede observar en la figura 3.4 aparecen representados diferentes vectores. Los vectores $\vec{u}$ y $\vec{v}$ son equivalentes ya que tienen la misma longitud y dirección aunque se encuentren desplazados en el espacio.

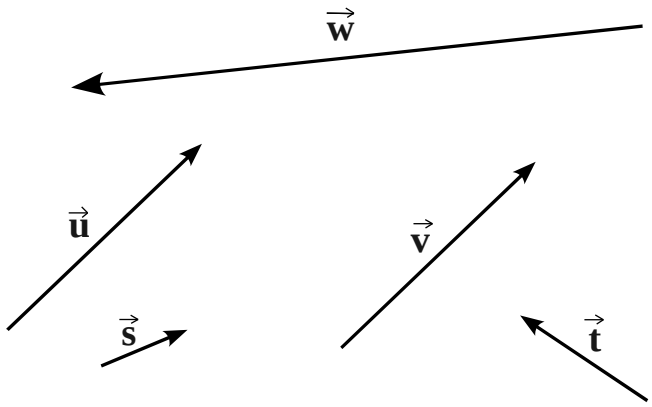

FIGURA 3.4: Equivalencia de vectores.

3.2.5. Suma de vectores.

Es una operación característica de los vectores [PJS], por la cuál dos o más vectores se pueden sumar dando como resultado un nuevo vector cuyas componentes resultan de la suma de cada componente de los correspondientes vectores. Por ejemplo, dados dos vectores, $\vec{u} = (u_x, u_y, u_z)$ y $\vec{v} = (v_x, v_y, v_z)$ como vemos en la figura 3.5, la suma de ambos da como resultado al vector $\vec{w}$.

$$\vec{w} = (u_x + v_x, u_y + v_y, u_z + v_z) \tag{3.5}$$

La suma de vectores también se puede aplicar a más de dos vectores, por ejemplo, dados cuatro vectores $\vec{u}$, $\vec{v}$, $\vec{s}$ y $\vec{t}$ (ver figura 3.6), el vector resultante es la suma de las componentes de cada uno de los vectores, en este caso es: $\vec{w} = \vec{u} + \vec{v} + \vec{s} + \vec{t}$.

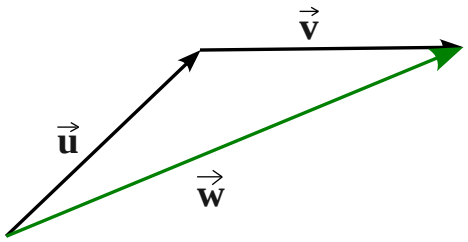

FIGURA 3.5: Suma de dos vectores.

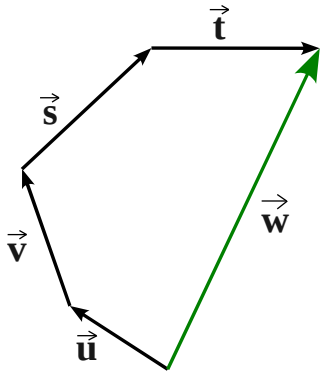

FIGURA 3.6: Suma de cuatro vectores.

3.2.6. Diferencia de vectores.

La diferencia de vectores se puede entender con una suma de vectores en la que uno de los vectores modifica su sentido. Para modificar el sentido de un vector sólo es necesario multiplicar las componentes del vector por -1. De manera que dados dos vectores $\vec{u}$ y $\vec{v}$ (ver figura 3.7) la diferencia entre ambos da como resultado un vector cuyas componentes es la suma de las componentes, teniendo en cuenta que uno de los dos vectores ha sido modificado por su opuesto.

$$\vec{w} = \vec{u} + (-\vec{v}) \Rightarrow \vec{w} = (u_x, u_y, u_z) + (-v_x, -v_y, -v_z) \tag{3.6}$$

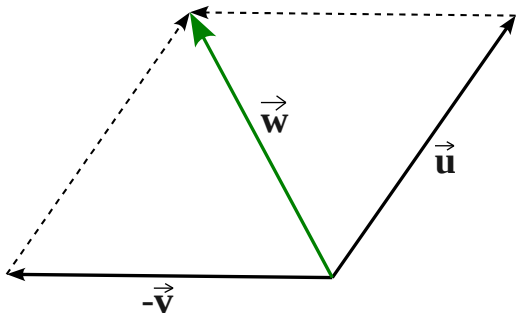

FIGURA 3.7: Diferencia de vectores.

3.2.7. Multiplicación de un vector

Es una operación fundamental de los vectores. Esta operación permite cambiar la longitud del vector, sin modificar su dirección. Como podemos observar en la figura 3.8 el vector $\vec{u}$ tiene exactamente la misma dirección que $\vec{v}$ pero es el doble de largo. Matemáticamente el vector $\vec{u}$ es representado como $\vec{u} = 2 \cdot \vec{v}$. En general la multiplicación de un vector es representada como: $\vec{u} = \alpha \cdot \vec{v}$ donde alfa es un número real, de manera que también puede ser un número negativo.

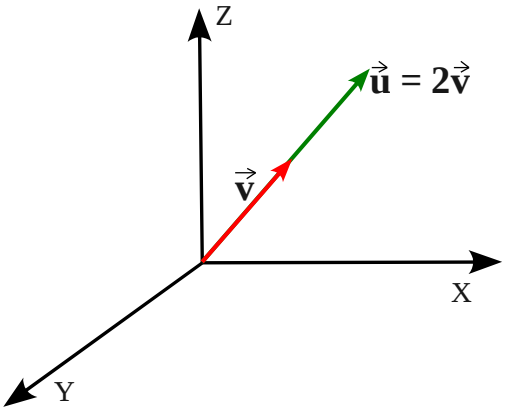

FIGURA 3.8: Multiplicación de vectores.

3.2.8. Producto escalar.

El producto escalar es un número real que viene determinado por suma de la multiplicación de las componentes de los dos vectores. Dados dos vectores $\vec{u} = (\vec{u}_x, \vec{u}_y, \vec{u}_z$ y $\vec{v} = (\vec{v}_x, \vec{v}_y, \vec{v}_z)$ su producto escalar es:

$$\vec{u} \cdot \vec{v} = \vec{u}_x \cdot \vec{v}_x + \vec{u}_y \cdot \vec{v}_y + \vec{u}_z \cdot \vec{v}_z = \|\vec{u}\| \cdot \|\vec{v}\| \cdot cos(\theta) \tag{3.7}$$

También podemos definir el producto escalar de $\vec{u}$ y $\vec{v}$ como la proyección de $\vec{u}$ sobre $\vec{v}$ multiplicado por el módulo de $\vec{v}$ y viceversa, ver figura 3.9.

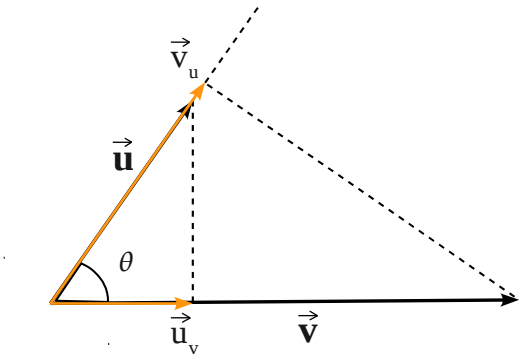

FIGURA 3.9: Producto escalar de vectores.

Siendo la representada la proyección de $\vec{u}$ sobre $\vec{v}$ como $\vec{u}_v = \|\vec{u}\| \cdot cos(\theta)$ y la proyección de $\vec{v}$ sobre $\vec{u}$, es $\vec{v}_u = \|\vec{v}\| \cdot cos(\theta)$ el producto escalar viene dado por:

$$\vec{u} \cdot \vec{v} = \vec{u}_v \cdot \|\vec{v}\| = \vec{v}_u \cdot \|\vec{u}\| \tag{3.8}$$

Se dice que dos vectores son perpendiculares si su producto escalar es igual a 0.

$$\vec{u} \cdot \vec{v} = 0 \tag{3.9}$$

A través del producto escalar podemos obtener el ángulo que forman los dos vectores con la siguiente expresión matemática:

$$cos(\theta) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \cdot \|\vec{v}\|} \tag{3.10}$$

El producto escalar tiene como propiedades:

- Distributiva: $\vec{u}\cdot(\vec{v}+\vec{s}) = (\vec{u}\cdot\vec{v})+(\vec{u}\cdot\vec{s})$
- Conmutativa: $\vec{u}\cdot\vec{v} = \vec{v}\cdot\vec{u}$

3.2.9. Producto vectorial.

Se puede definir como el vector ortogonal al plano que forman los vectores los cuales se desea calcular su producto vectorial. Dados dos vectores $\vec{u} = (u_x, u_y, u_z)$ y $\vec{v} = (v_x, v_y, v_z)$ (ver imagen 3.10) el producto escalar viene determinado por la expresión matemática:

$$\vec{u}\times\vec{v} = (u_y v_z - u_z v_y,\ \ u_z v_x - u_x v_z,\ \ u_x v_y - u_y v_x) \tag{3.11}$$

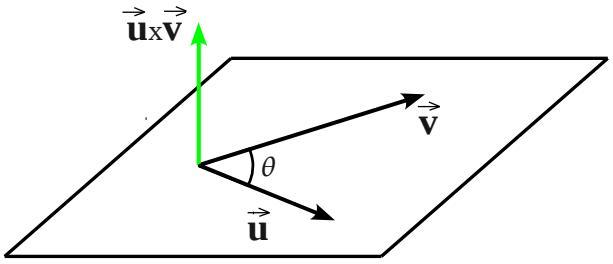

FIGURA 3.10: Producto vectorial.

El sentido del vector resultante tras aplicar el producto vectorial se asigna siguiendo la regla del sacacorchos dependiendo desde donde avance. Ver figura 3.11. Si el producto vectorial es $\vec{u}\times\vec{v}$ entonces el sentido del vector resultante es hacia arriba, mientras que si el producto vectorial es $\vec{v}\times\vec{u}$ el sentido del vector resultante es al contrario, hacia abajo. Las propiedades del producto vectorial son:

- No goza de la propiedad conmutativa: $\vec{u}\times\vec{v} = -\vec{v}\times\vec{u}$
- Propiedad distributiva con respecto a la suma: $\vec{u}\times(\vec{v}+\vec{w}) = \vec{u}\times\vec{v}+\vec{u}\times\vec{w}$
- Asociativa: $m(\vec{u}\times\vec{v}) = (m\vec{u})\times\vec{v} = \vec{u}\times(m\vec{v}) = (\vec{u}\times\vec{v})m$

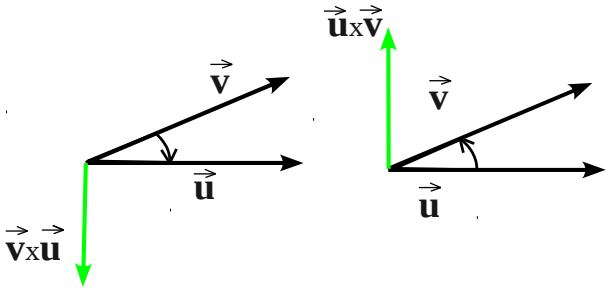

FIGURA 3.11: Sentido del producto vectorial.

3.2.10. Bounding box.

Se define como volumen con forma de caja 3D que engloba a uno o más objetos. Viene definido por ocho puntos. Podemos encontrar dos tipos de bounding box.

1. Bounding box alineado con respecto a los ejes o AABB (Axis Aligned Bounding Box): también llamado como caja rectangular, es una caja en la que las caras están normalizadas coincidiendo con los ejes de coordenadas. Está definido por dos puntos totalmente opuestos, un punto inferior a la izquierda $\mathbf{a}^{min}$ y el otro el punto superior derecha $\mathbf{a}^{max}$(ver figura 3.12), donde $\mathbf{a}^{min} \leq \mathbf{a}^{max}, \forall i \in x,y,z$.

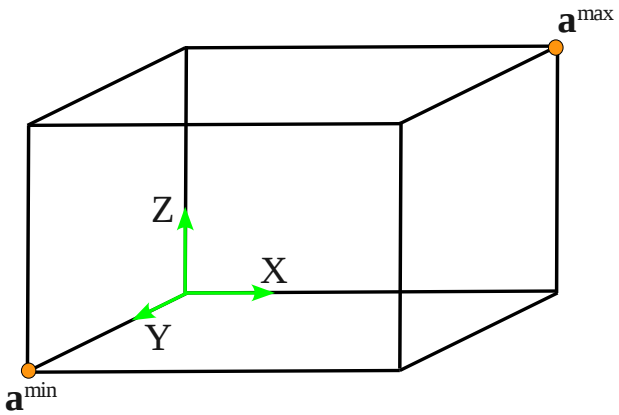

FIGURA 3.12: Ejemplo de bounding box alineado con los ejes.

2. Bounding box rotado con respecto a los ejes o OBB(Oriented Bounding Box): Se caracteriza porque las caras que componen el bounding box son ortogonales dos a

dos. Se puede ver como un AABB rotado con respecto a los ejes de coordenadas. Un OBB puede ser definido por el punto central b^{center} del bounding box y tres vectores $\vec{b}_x, \vec{b}_y, \vec{b}_z$ (ver figura 3.13) que describen la dirección de los lados de la caja.

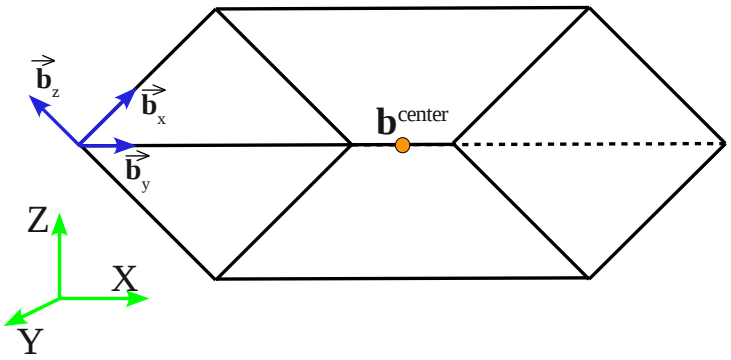

FIGURA 3.13: Ejemplo de bounding box rotado con respecto a los ejes.

3.2.11. Método un punto está contenido dentro de un poliedro.

Es una funcionalidad necesaria para la elaboración de este proyecto. Se trata de conocer si un punto en el espacio 3D se encuentra dentro de un bounding box. Según [PJS], dado un bounding box compuesto por seis caras es necesario calcular para cada una de las caras si el punto pertenece a alguna de las caras o se encuentra dentro del bounding box. Se dice que un punto en el espacio 3D P forma parte de alguna de las caras (en este caso la cara i), si para alguna de las caras el producto escalar del vector normal de dicha cara $\vec{n}_i$ con respecto al vector formado por el punto y uno de los vértices de la cara v_i es igual a 0, esto se debe a que el ángulo que forman el vector normal $\vec{n}_i$ con el vector que une v_i con el punto P es de 90 grados, y por definición el producto escalar de dos vectores es: $\vec{u} \cdot \vec{v} = \|\vec{u}\| \cdot \|\vec{v}\| \cdot cos(\theta)$, como el ángulo es 90 grados y el coseno de 90 es cero, su producto escalar es cero (véase la figura 3.14). La expresión matemática es:

$$\vec{n}_i \cdot (P - V_i) = 0 \tag{3.12}$$

Para conocer si un punto P está contenido en el interior de un bounding box, se realiza el producto escalar para todas las caras del bounding box, el producto escalar entre el vector

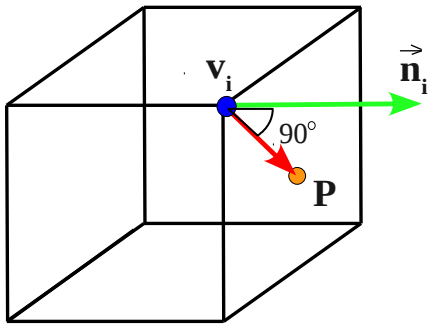

FIGURA 3.14: Un punto P está contenido en una cara de un bounding box.

normal de cada cara y el vector formado por uno de los vértices de dicha cara y el punto P es menor que cero.

$$\vec{n}_i \cdot (P - V_i) < 0 \tag{3.13}$$

También se puede comprobar si un punto está fuera de un bounding box. Se dice que un punto está fuera de un bounding box si el producto escalar de alguna de las caras (i) del bounding box de su vector normal ($\vec{n}_i$) con el vector que une un vértice de dicha cara con el punto P es mayor que cero.

$$\vec{n}_i \cdot (P - V_i) > 0 \tag{3.14}$$

Por tanto, se dice que un punto está en el interior de un bounding box cuando para todas las caras del bounding box, el producto escalar del vector normal de cada cara con respecto al vector formado por un vértice y el punto a analizar es menor o igual que cero.

$$\vec{n}_i \cdot (P - V_i) \leq 0 \tag{3.15}$$

Mediante este algoritmo se puede distinguir la posición de un punto con respecto a un bounding box:

- El punto P se encuentra en el interior del bounding box si no existe ninguna cara en la que el producto escalar de 3.15 es igual a cero.
- Si existe alguna cara en que el producto escalar 3.15 es igual a cero, se puede comprobar que el punto pertenece a alguna de las caras.

- Si existen dos caras en las que el producto escalar 3.15 es igual a cero, entonces se especifica que el punto pertenece a una arista del bounding box.
- Si existen tres caras en las que el producto escalar 3.15 es igual a cero, entonces se puede establecer que el punto pertenece a un vértice del bounding box.

El algoritmo tiene una complejidad de **O(*n*)** dónde *n* es el número de vértices que tiene el bounding box.

El pseudocódigo del algoritmo es:

Entrada: Un punto 3D $P(P_x, P_y, P_z)$ y un bounding box $B(V_0, V_1, V_2, V_3, V_4, V_5, V_6, V_7)$ definido por ocho vértices. Donde cada vértice es un punto en el espacio 3D de la forma $V = V_x, V_y, V_z$.
Salida: **cierto** Si el punto *P* está en el interior del bounding box y **falso** en caso de que esté fuera del bounding box.

```
para i = 0 hasta 6 hacer
  si Producto escalar(B.caras(i).normal(), (P - B.caras(i).vertice)) >0 entonces
    devolver falso
  fin si
fin para
devolver cierto
```

3.3. Estado de las tecnologías de análisis de vídeo.

El análisis de vídeo junto con la videovigilancia inteligente son campos muy activos en la investigación. En los últimos diez años multitud de ediciones de revistas y artículos se han dedicado a la explicación e investigación sobre el análisis de vídeo en los sistemas de vigilancia. También se han desarrollado productos a raíz de la importante labor de investigación llevada a cabo en este sector. El desarrollo de proyectos como VSAM en Estados Unidos y PRISMATICA en Europa han animado el desarrollo de la detección, seguimiento y análisis de basados en secuencias de vídeo [VG08]. Sin embargo, estas tecnologías están emergiendo actualmente y su uso en aplicaciones de gran escala está todavía falto de madurez. El costo de estas tecnologías y sus limitaciones técnicas actuales para la monitorización de entornos complejos suponen un obstáculo para el crecimiento de estas tecnologías.

3.3.1. Descripción de las técnicas de análisis de vídeo.

El análisis de vídeo consiste en una reproducción asistida por computador que realiza un análisis como si una persona estuviera monitorizando los vídeos captados por las cámaras de vigilancia [VG08]. El software de análisis procesa el flujo de vídeo y automáticamente detecta los objetos (personas, vehículos, etc.) que hay en el entorno. Una vez que los objetos han sido detectados, son almacenados. Sus acciones son clasificadas y analizadas por un operador humano. Las aplicaciones de software de análisis pueden ser utilizadas de dos formas:

- **En tiempo real**: en este caso se detectan las situaciones en el flujo de vídeo que representan una posible amenaza en el entorno y se activa una alarma.
- **Vigilancia forense**: permiten encontrar rápidamente en el flujo de vídeo almacenado el incidente sucedido.

Alarmas en tiempo real: La mayoría de las alarmas de los SVI son definidas por los usuarios del sistema. Éstas pueden ser genéricas como la detección de objetos abandonados o que un objeto circula a una velocidad superior a la establecida. Estas alertas pueden ser definidas en la fase de detección de movimiento de un objeto en la escena. Además se podrán definir alarmas más específicas una vez se hayan clasificado los objetos y se hayan analizado. Las alarmas relacionadas con la semejanza de un comportamiento a un modelo de comportamiento definido en el sistema se considera un tipo de alarma predefinida. Algunas

alarmas pueden generarse automáticamente durante el análisis del flujo de vídeo en tiempo real. Durante un período de tiempo el sistema aprende un nuevo modelo de comportamiento y después cuando se produce un comportamiento diferente al modelo aprendido genera una alarma. Por ejemplo un software de análisis de vídeo aprende un modelo en el que los vehículos circulan por la calzada y otro modelo en que los peatones caminan por la acera, si se produce que un vehículo circula por la acera, el sistema activa una alarma [VG08].

Mediante el análisis del flujo de vídeo que se incorpora en los SVI podemos distinguir diferentes niveles de análisis:

- Detección de cambios en el entorno monitorizado.
- Segmentación de objetos en movimiento.
- Seguimiento de objeto.
- Clasificación e identificación de objetos.
- Clasificación de comportamientos.

Detección de cambios.

En los SVI la detección de cambios en el entorno monitorizado es la base para el análisis inteligente de vídeo. Se puede detectar la actividad de objetos en el entorno, también pueden captar la aparición o desaparición de objetos (objetos abandonados, objetos robados, etc.). Además puede servir para informar automáticamente al sistema de la aparición de alteraciones intencionadas en una cámara (aparición de polvo, telarañas, pintura, adhesivos, etc.). Muchas de las técnicas empleadas para la detección de movimiento, se basan en la detección de cambios en el entorno monitorizado. Sin embargo, la detección de cambios no siempre se atribuye al movimiento de un objeto en la escena, sino que puede estar originada por alteraciones medioambientales en el entorno. En el análisis de vídeo se deben diferenciar entre posibles fluctuaciones a nivel de píxel que pueden venir originadas por alteraciones del clima y el movimiento de objetos dentro de un espacio observado [VG08].

Este es uno de los principales problemas existentes en el análisis de vídeo, ya que para la monitorización de entornos complejos pueden aparecer alteraciones como: cambios de iluminación (sombras, movimiento del sol, aparición de nubes, nieve, lluvia, reflejos de luz procedente de ventanas o del agua, deslumbramiento por las fuentes de iluminación del entorno monitorizado, etc.) o movimientos no relevantes para el sistemas(movimiento de alguna bandera a causa del viento, el movimiento de árboles o arbustos, o el movimiento del agua del mar por la aparición de olas.

Existen muchos algoritmos que no tienen en cuenta estos problemas medioambientales ya que trabajan en zonas de interior, en entornos con poco movimiento. Sin embargo, otros algoritmos más robustos permiten su utilización en entornos más complejos en áreas externas donde aparecen las alteraciones del medio. Para solucionar estos problemas se han propuesto dos técnicas de segmentación:

- **Sustracción del fondo.** Un conjunto de técnicas van comparando cada fotograma de una secuencia de imágenes y se extraen aquellas zonas en las que no se producen perturbaciones. Las áreas de movimiento son determinadas píxel a píxel por la diferencia en intensidad en caso de superar un umbral determinado. La diferencia entre píxeles es muy sensible y alteraciones medioambientales en la escena puede llevar a confusión en la detección de movimiento, para evitar esto se dispone de un umbral configurable para evitar confusiones, de manera que si la diferencia de píxeles supera este umbral se considera que hay un objeto en movimiento. Además el método de la sustracción de fondo es un método que es mayormente utilizado en sistemas de vigilancia interiores donde los cambios medioambientales están más controlados [VG08].

- **Diferencias en el tiempo.** Esta técnica se basa en la detección de cambios en una secuencia de fotogramas consecutivos en un período de tiempo. Con la utilización de estas técnicas a menudo se pueden detectar pequeños espacios de tiempo en los que no se detecta el movimiento del objeto. Estas técnicas requieren de la utilización de filtros y operaciones morfológicas para detectar el movimiento de objetos. Con el objetivo de almacenar sólo las zonas en las que se producen a menudo movimiento de objetos y eliminar los movimientos ocasionales, algunas técnicas utilizan un mapa de regiones en las que se produce un alto nivel de actividad basado en patrones de movimiento [VG08].

- **Flujo óptico.** Los métodos de detección de movimiento utilizando flujo óptico consisten en detectar los cambios de dirección de los píxeles asociados al movimiento de los objetos en la escena. Sin embargo, éstos requieren la realización de cálculos complejos que son difíciles de hacer en tiempo real. Además esta técnica se muestra muy sensible al ruido de la imagen [VG08].

Seguimiento de objetos.

Una vez realizada la tarea de detección de objetos, los SVI tienen que realizar una asignación de las nuevas posiciones de los objetos en cada frame, de forma que se realiza un

seguimiento del objeto. El seguimiento de un objeto comienza cuando se detecta un objeto en el entorno monitorizado y se termina cuando el objeto abandona el entorno monitorizado. El seguimiento de los objetos puede realizarse en 2D mediante la monitorización de una sola cámara o en 3D combinando dos vistas con relaciones geométricas conocidas. Muchas técnicas de seguimiento de objetos están basadas en métodos matemáticos que permiten predecir la posición de un objeto en un fotograma determinado basándose en las posiciones del objeto en fotogramas anteriores.

El seguimiento de varios objetos al mismo tiempo plantea muchos retos. Cada objeto detectado en un fotograma debe ser archivado en su correspondiente secuencia de fotogramas anteriormente reconocidos. Esta correspondencia se realiza por las características del objeto o por su apariencia (por ejemplo, vértices, volumen, área, etc.) y en gran medida por el análisis de su posición en el fotograma actual con respecto a su posición en fotogramas anteriores.

Las oclusiones presentan uno de los problemas más difíciles de abordar en el seguimiento de objetos. Un sistema de videovigilancia puede perder la pista de un objetos que se encuentra en el entorno observado, pero no es detectado por la oclusión total o parcial de éste. Las oclusiones se pueden dar bien por la propia definición del entorno monitorizado (detrás de un árbol, farola, arbusto, etc.) o bien por la existencia de otros objetos en la escena que se encuentren delante del objeto. Otro de los grandes problemas para el seguimiento de objetos es la separación de objetos, cuando varios objetos se encuentran muy juntos [VG08].

Clasificación e identificación de objetos

Los objetos detectados por los sistemas de vigilancia son habitualmente clasificados en diferentes categorías como: peatones, vehículos, animales, equipaje, etc. Esta clasificación puede realizarse antes de la etapa de seguimiento de objetos, para centrar la atención sólo en los objetos relevantes para el sistema de vigilancia. En general, la clasificación de los objetos que aparecen en el entorno suele realizarse atendiendo a las propiedades de apariencia o forma del objeto y de los atributos de movimiento. Por ejemplo, un peatón generalmente es más alto que ancho, mientras que un vehículo suele ser más ancho que largo. Además el movimiento de peatones tiene unas características específicas y presenta cierta periodicidad en su movimiento [VG08].

La identificación de objetos conlleva a un mayor reconocimiento. Después de reconocer la categoría a la que pertenece un objeto, éste debe ser identificado. En los últimos años, se han realizado importantes esfuerzos en la investigación y el desarrollo del reconocimiento de la cara de los humanos y del número de matrícula de vehículos, con el fin de identificar a

posibles delincuentes, terroristas, bandas organizadas, etc. La identificación de la cara y la forma de caminar son dos de las herramientas principales para la identificación de personas por parte de los sistemas de videovigilancia. El análisis de la forma de caminar de un individuo, proporciona información previa cuando la detección del objeto está lejos aún para ser identificado por la cara, en entornos de monitorización amplios. La identificación facial proporciona una información más precisa pero para su correcto comportamiento necesita una imagen con buena resolución y una posición adecuada (por ejemplo si el individuo está mirando hacia abajo o lleva una gorra impide una identificación correcta). La mayoría de los sistemas de reconocimiento facial toleran una rotación de la cara en unos 45 grados. Muchos identificadores faciales y técnicas de comparación han sido propuestas desde la década de 1990 y se han basado principalmente en la extracción de descriptores matemáticos o puntos topológicos de la cara. Inicialmente los sistemas de identificación de caras se basaban en fotografías para identificar a los individuos. Sin embargo, modificaciones en el aspecto físico de la cara como la utilización de gafas, la aparición de barba proporcionaban son factores que alteran el buen comportamiento para la identificación facial. Aparecen nuevas técnicas de identificación facial en 3D que permiten mejorar el rendimiento y dar mayor solidez en la identificación de caras que las técnicas en 2D. Sin embargo, el reconocimiento facial en entornos donde los individuos suelen ir agrupados todavía sigue siendo un problema sin resolver en los sistemas de análisis de vídeo.

La identificación de matrículas en los sistemas de vigilancia es una tarea difícil. Para esto se requieren imágenes de alta resolución. Existen muchas alteraciones medioambientales en el análisis de imagen: la meteorología ambiental como la lluvia, nieve, o cambios en la iluminación del sol, sombra de los edificios o árboles, el brillo de los faros de los vehículos, y las oclusiones totales o parciales. Para realizar la lectura de una matrícula en primer lugar se debe detectar un rectángulo de la placa entre todos los detalles de la imagen.

A continuación, se procede con la identificación óptica de caracteres. Una matrícula reconocida en un ángulo determinado distorsiona la lectura de la misma y dificulta el proceso de identificación. Con el objetivo de maximizar la eficiencia, el proceso de identificación de matrícula utilizan sistemas especializados que se centran en el posicionamiento de la cámara y en la calidad de la iluminación.

Clasificación de comportamientos.

Uno de los objetivos de los sistemas de videovigilancia es la interpretación de los comportamientos individuales de los objetos detectados en un entorno monitorizado. En términos generales el comportamiento se define como las acciones observadas por los SVI,

ya sean humanos, vehículos, animales, etc. El reconocimiento del comportamiento de un objeto, requiere de análisis semántico, a veces más complejo de lo que aparecen en las imágenes captadas. Este es uno de los retos más difíciles para los sistemas de análisis de vídeo [VG08].

El objetivo es detectar el comportamiento simple de los objetos, como por ejemplo, un individuo que deja una bolsa en el suelo de una sala. Sin embargo, el comportamiento de los objetos se puede complicar si se encadenan varios comportamientos, por ejemplo, el individuo A deja una bolsa en una sala, después llega un individuo B coge la bolsa y sale de la sala, después aparecen el individuo A y el individuo B juntos y se dan la mano; conclusión la bolsa no ha sido robada.

Analizar el comportamiento e interpretarlo supone el reconocimiento de patrones de movimiento y a partir de ellos en un nivel superior, extraer una descripción de la acción. El conjunto de atributos obtenidos del comportamiento del objeto debe ser asociado a un modelo que su vez representa un comportamiento determinado. Por lo tanto el problema recae en definir los modelos de comportamientos, bien por aprendizaje automático o bien por definición de los mismos, y encontrar un método de comparación entre comportamientos de objetos detectados en el entorno con los modelos de comportamientos definidos y sea flexible tolerando algunas pequeñas variaciones con respecto a los modelos de comportamiento definidos.

Los modelos ocultos de Markov, las redes neuronales y las redes Bayesianas son algunas de las técnicas más utilizadas para la creación de modelos de comportamiento normal y detectacción de comportamientos anómalos. Éstos están basados en un sistema de reglas, y activan una alarma en caso de incumplimiento. Por ejemplo, un objeto de un tamaño superior a un umbral definido se queda estacionado en mitad de la calzada obstaculizando la circulación.

3.3.2. Análisis en grupo.

Uno de los dominios de investigación futura de los sistemas de análisis de vídeo es el análisis en grupo y la monitorización. Debido al crecimiento de la población mundial, y la densificación de los principales centros urbanos se realiza la cuestión de garantizar la seguridad de los ciudadanos en grandes concentraciones. Los sistemas de videovigilancia se ha convertido en una vía interesante, para garantizar la seguridad en entornos con una elevada concentración de objetos [VG08].

Actualmente, muchas de las cámaras utilizadas en eventos importantes (por ejemplo, eventos deportivos, reuniones políticas, etc.) pero la eficiencia de los sistemas de análisis de vídeo sigue siendo un debate. En [Zha08] se realiza una investigación sobre el análisis de vídeo. En el análisis de grupo también se realiza un análisis en las etapas anteriormente detalladas. Sin embargo, el análisis en grupo presenta un conjunto de problemas específicos que se derivan de la composición de un grupo por un determinado número de personas.

La estimación de la densidad de objetos que aparecen en el entorno monitorizado y su gestión es un paso fundamental para el análisis en grupo, por lo que requiere una evaluación aproximada del número de individuos. Las numerosas oclusiones y el agrupamiento de objetos dificultan la tarea de conteo de objetos. Algunas técnicas realizan una relación proporcional entre el número de píxeles correspondientes con las zonas de detección de objetos (después de la segmentación) y el número de personas. Otros métodos se basan en la textura de la imagen con el fin de obtener la densidad de objetos en el entorno. Algunos algoritmos detectan el contorno de la cabeza, mientras que otros utilizan un histograma de varias características para deducir el número aproximado de personas. Aunque el seguimiento de objetos es uno de los temas de investigación más estudiados en los sistemas de videovigilancia, la mayoría de los algoritmos desarrollados se aplican sobre un pequeño número de personas (menos de diez personas por unidad de tiempo). El seguimiento de peatones en grupo presenta muchas dificultades, especialmente para saber la cantidad de personas que componen el grupo y realizar un seguimiento y las numerosas oclusiones que pueden aparecer.

Algunos métodos realizan un seguimiento de las principales características de los objetos, las cuales corren menos riesgo de ser alteradas que si el seguimiento se basa en el reconocimiento de contornos. Otros métodos se basan en el modelo del cuerpo humano o de sus partes. Los métodos de correspondencia probabilística o de filtros específicos son utilizados durante el análisis del vídeo. Existen algunos artículos que realizan un seguimiento utilizando varias cámaras: M2Traker trabaja con cámaras sincronizadas.

3.3.3. Sistemas de videovigilancia con sistemas multicámara.

Los sistemas de videovigilancia modernos, sobre todo los que utilizan red IP, pueden estar compuestos de cientos de cámaras. A veces puede que varias cámaras visualicen el mismo espacio del entorno monitorizado. Algunas cámaras pueden ser de accionado eléctrico y puede ser controlado para capturar más detalles sobre determinadas situaciones en entornos muy amplios. Por ejemplo, la cámara puede realizar zoom sobre una persona que desea entrar en un área con el fin de identificarlo. En determinadas redes, las cámaras son inteligentes, lo que significa que tienen su propia unidad de proceso. Las cámaras pueden intercambiar información con un sistema central o directamente entre las propias cámaras [VG08].

Las redes de cámaras de vigilancia permiten seguir los objetos en áreas extensas. Además la utilización de varias cámaras para supervisar un mismo entorno elimina muchos de los problemas de oclusión, al contar con diferentes puntos de vista. En una arquitectura distribuida, el proceso de análisis puede realizarse en paralelo, los que proporciona mayor rapidez de análisis y menos utilización del ancho de banda ya que sólo se envían metadatos.

Sin embargo, estas redes también presentan algunos problemas que los científicos están intentando resolver.

- **Calibración de la cámara.** Es una operación que consiste en establecer correspondencia entre una marca con coordenadas globales situada en el entorno observado y el sistema de coordenadas de la cámara así como el ajuste de parámetros intrínsecos a la cámara, tales como por ejemplo la distorsión. Esta labor que requiere de una gran precisión puede resultar muy tediosa para una red de videovigilancia compuesta por muchas cámaras. Por estos motivos es preferible desarrolla funciones de análisis que no requieran la calibración de las cámaras.

- **Detección de movimiento.** Para una cámara de accionado eléctrico que provoca el movimiento de la cámara se debe distinguir entre el movimiento de la cámara por orden eléctrica o la detección de movimiento en el entorno, ya que el movimiento de la cámara debido a una orden produce una elongación de la apariencia del objeto que puede llevar a errores.

- **Seguimiento de objetos desde diversas cámaras.** Con el objetivo de ser capaz de seguir un objeto desde una cámara a la siguiente, se debe establecer la correspondencia entre los distintos puntos de vista de las diferentes cámaras teniendo entre ellas un punto en común. El proceso de reiniciar una cámara consiste en calcular

los parámetros para transformar una imagen de una cámara a la siguiente basándose en el cambio del punto de referencia y en el modelo de movimiento. Esta operación necesita de un conocimiento previo de la topología de la escena. Esto permite la representación de objetos en 3D. Los cambios producidos en la apariencia del objeto, iluminación, etc. en los cambios de vista de cada cámara pueden complicar el proceso de reposición.

- **Detección de alteraciones en la cámara.** Un sistema de videovigilancia en red tiene más de una cámara, por lo que es más dificil controlar su funcionamiento. El sistema debe tener herramientas integradas que detecten la aparición de alguna avería o alteración en las cámaras del sistema. Los sistemas de videovigilancia multicámara son utilizados sobretodo, para monitorizar áreas extensas o distribuidas, tales como en medios de transporte, infraestructuras bancarias, instituciones de gobierno (bases militares, prisiones, lugares estratégicos, radares, hospitales, etc), edificios públicos, centros comerciales y grandes aparcamientos.

3.3.4. Análisis de sistemas comerciales

Muchos de los productos de análisis de vídeo han incorporado al mercado sistemas de videovigilancia inteligente desde la década 2000. Normalmente la mayoría de estos sistemas realizan los mismos tipos de análisis utilizando diferentes técnicas. Sin embargo, hay algunos que han incorporado funcionalidades más específicas (por ejemplo, reconocimiento facial o de matrícula) [VG08].

Las principales funciones del análisis de vídeo que hay en el mercado se pueden resumir de la siguiente manera:

- Seguimiento por cámara PTZ cuando es activada por alarma.
- Detección por avería en las cámaras.
- Detección de objetos desaparecidos o abandonados.
- Detección de intrusos (personas o vehículos) en áreas controladas: límite virtuales, zonas de seguridad, control de acceso a personas o vehículos identificándolos, control de las salidas de emergencia, etc.
- Detección y seguimiento de personas y vehículos: análisis de velocidad y dirección de la trayectoria y detección de estacionamiento de personas o vehículos en zonas sensibles de las infraestructuras.

- Conteo de personas o vehículos, detección de grupos.
- Detección de robo o fraude en lugares de venta(caja registradora, objetos valiosos, etc.).
- Reconocimiento de comportamientos: detección de individuos sospechosos, peleas, carreras, caídas, movimientos que impiden la fluidez en el tráfico, realización de graffitis y vandalismo.
- Monitorización del tráfico: control de la velocidad, densidad y dirección, control de objetos que circulan por la acera, concentración de objetos, desprecio de los límites de velocidad, etc.
- Detección y reconocimiento de caras.
- Reconocimiento de matrículas de vehículos.
- Detección de incendios por análisis de imagen.

Las tecnologías de análisis de vídeo son jóvenes y necesitan desarrollarse. Muchas de las aplicaciones comerciales trabajan bien en entornos comerciales, pero la mayoría de ellos no funcionan tan bien a la hora de ser utilizados por los clientes. Estos sistemas requieren de grandes ajustes y configuración de parámetros para lograr un comportamiento eficaz y adecuado.

Las funciones básicas como la detección de movimiento y de situaciones sospechosas son lo suficientemente importantes como para ser utilizadas en una amplia gama de aplicaciones.

En general, la investigación sobre técnicas de detección y seguimiento de objetos han sido muy desarrolladas y algunas de ellas se incorporan a los sistemas comerciales implementados para casos demasiado sencillos. Sin embargo, se necesita un gran esfuerzo en el desarrollo de estas técnicas con el objetivo de lograr el rendimiento y la eficacia suficiente para operar en entornos complejos. La detección y seguimiento de grupos de objetos en entornos monitorizados con una red multicámara es uno de los problemas más difíciles.

Existen muchos productos especializados en la detección y seguimiento basado en imágenes. Su eficacia depende en gran medida de la aplicación y de la complejidad del entorno en el que se utiliza. Además están condicionadas por las condiciones de monitorización del entorno (la iluminación, el posicionamiento de las cámaras, etc.), por

tanto, realizar un control de estas propiedades en el entorno mejora en gran medida el comportamiento de estos sistemas, obteniendo resultados muy satisfactorios.

Determinados comportamientos son detectados con el software de análisis existente en el mercado. De la monitorización de un entorno se pueden extraer características simples (velocidad, dirección, etc) y se pueden deducir si una persona va corriendo o si una persona se cae. Sin embargo, es difícil determinar el grado de gravedad de las diversas acciones reconocidas. Por ejemplo, ¿cómo podemos diferenciar que un individuo está corriendo porque se trata de un delincuente que está tratando de huir o si por los contrario se trata de un individuo que llega tarde para coger un avión?. El reconocimiento de comportamientos sigue siendo uno de los principales campos a desarrollar en el análisis de vídeo.

[Hea08] sugiere una lista de características deseables en sistemas comerciales de análisis de vídeo.

1. **Reducir al máximo el número de configuraciones necesarias.** Un sistema comercial no debe necesitar un gran período de tiempo en su configuración, parametrización y tiempo de entrenamiento. Además el usuario debe introducir la mínima cantidad de información para la instalación del sistema de vídeo.

2. **Ofrecer un rendimiento adecuado.** El número de falsas alarmas debe ser lo más reducido posible, para evitar la actuación del personal de seguridad innecesariamente. Sin embargo es más crítico que el sistema no avise ante una amenaza real. Se debe buscar un equilibrio entre el número de falsas alarmas y el número de veces que se omite la alarma. Este equilibrio depende en gran medida de aplicación específica.

3. **Eliminar la calibración de la cámara o automatizar este proceso.** La correcta calibración de la cámara permite conocer el tamaño real y la velocidad de los objetos en la escena. Se establece una correspondencia entre la geometría de la imagen y de la escena.

4. **Soporte para una amplia gama de cámaras.** Se debe de posibilitar el análisis de vídeo tanto para cámaras a color, en blanco y negro, infrarojo, térmico o omnidireccional.

5. **Ser los más flexible posible.** Lo ideal sería que la detección de eventos relevantes sea posible en los diferentes tipos de entornos y casos de aplicación. Sin embargo, algunos ajustes tienen problemas muy específicos, como la detección de personas cerca del mar. El movimiento del agua y los reflejos del sol con ésta provocan problemas en los algoritmos basados en la detección y seguimiento de objetos.

6. **Ofrecer mecanismos de protección de la privacidad.** Los sistemas de análisis deben guardar la información detallada de los objetos visualizados y sólo mostrar las situaciones sospechosas sin detallar información comprometida.

3.4. Necesidades y retos del análisis de vídeo.

Dada la gran cantidad de información capturada por la seguridad, urge la necesidad de desarrollar métodos automáticos para la detección de situaciones sospechosas. De manera que sólo las situaciones sospechosas son analizadas por personal humano. Existen algunas técnicas muy satisfactorias pero pocas de ellas están comercializadas para usarlas en un entorno real.

También existen muchos algoritmos aplicables a los sistemas de videovigilancia pero éstos son desarrollados para resolver un problema muy específico de la monitorización y no aborda todos los problemas que la videovigilancia abarca. Existen sistemas muy completos que operan sobre entornos con condiciones casi reales. Actualmente el análisis de vídeo trabaja bien en:

- Detección de movimiento.
- Detección de objetos específicos.
- Reconocimiento de matrículas en sistemas específicos.
- Detección de comportamientos específicos, tales como: caminar, correr, llevar un objetos, etc.

La siguiente lista contiene los aspectos que se deben mejorar o desarrollar:

- Detección y seguimiento y reconocimiento en tiempos cercanos al tiempo real, es decir, que el reconocimiento de incidentes sea lo más rápido posible a su suceso. Esto es importante en infraestructuras sensibles en las que la rápida intervención del personal de seguridad puede evitar resultados catastróficos.
- Reconocimiento de caras en tiempo real (25 o 30 frames/sec).
- Reconocimiento de caras en medio de una multitud de personas. Esto requiere de una calidad de imagen suficiente para ser accesible con cámara de megapíxeles, ahora en el mercado.

- Conteo de personas en entornos complejos con gran concentración de personas, bajo diferentes puntos de vista.
- Seguimiento de objetos en entornos de gran concentración (seguimiento de una persona dentro de un grupo o de un vehículo en concreto dentro de todo el flujo de tráfico.
- Seguimiento de las propiedades del cuerpo humano para comprender su comportamiento.
- Diferenciación de áreas con determinados colores. Esta función permitiría conocer por ejemplo el color de un coche y podría ayudar al algoritmo de reconocimiento de matrículas, o para reconocer a una persona por el color de la ropa que lleva.
- Fusión de diversos datos de vídeo (por ejemplo, fusión del reconocimiento de la cara con su forma de caminar), o la mezcla de datos obtenidos por diversos sensores (sensores de temperatura, sistemas biométricos, etc) son algunos de los métodos construidos.

Muchos algoritmos de análisis de vídeo funcionan sólo con cámaras fijas, con buena resolución de imagen e iluminación adecuada y constante, lo cuál no es muy útil para la mayoría de las aplicaciones. Algunos de los problemas más frecuentes que nunca se tienen en cuenta tales como los cambios bruscos de la cámara (tela de araña, polvo, movimientos de cámara). La robustez y precisión de los algoritmos de detección, seguimiento y reconocimiento son vitales cuando se trata de un entorno con condiciones ambientales cambiantes. Con el objetivo de aumentar su uso, los sistemas de videovigilancia inteligentes deben ser lo suficientemente fuertes como para hacer frente a las diferentes condiciones climáticas, adaptarse a los cambios de iluminación de la escena (luz natural o artificial) y adaptarse a los fallos de hardware y software producidos.

La evaluación del rendimiento de los SVI es uno de los retos más importantes. Existen tareas en vigilancia como la detección de objetos abandonados, cuya monitorización humana obtiene unos resultados horribles. Sin embargo, es difícil comparar los sistemas entre sí, ya que la mayoría de las pruebas de rendimiento se llevan a cabo con diferentes conjuntos de datos.

Por lo tanto, para poder comparar el rendimiento de diferentes SVI es necesario el desarrollo de métodos objetivos y sistemáticos. La evaluación del sistema depende en gran medida de los resultados en las tareas de análisis de vídeo.

A continuación, se van a detallar algunas iniciativas de evaluación de análisis de vídeo en SVI:

- **PETS (Performance Evaluation of Tracking and Surveillance).** Propone una métrica para la evaluación de algoritmos online.
- **CREDS (Challenge for Real-time Event Detection Solutions).** Competencia para la detección eventos predefinidos para la vigilancia en el metro de París (detección de personas que caminan sobre las plataformas).
- **I-LIDS (Imagery Library for intelligent Detection Systems).** Fue lanzado para la oficina de la sección de desarrollo cientíco de Gran Bretaña. Permite realizar test de rendimiento sobre datos de vídeo, para detectar vehículos aparcados, equipaje abandonado, control de intrusión en determinadas áreas, control de puertas, etc.
- **ETISEO (Évaluation du traitement de l'interprétation de séquences video – performance evaluation project for video surveillance).** Proyecto llevado a cabo por el gobierno francés, permite el reconocimiento del cuerpo de personas bajo petición, este proyecto puede ser utilizado con fines de investigación, no para comercializarlo.
- **FRVT (Face Recognition Vendor Tests).** Permite realizar pruebas de evaluación de los diferentes sistemas de reconocimiento facial y conducta utilizando una base de datos para proporcionar una comparativa objetiva entre los diferentes proveedores.

Por último es importante que los sistemas de análisis se adapten a las nuevas arquitecturas de los sistemas de videovigilancia: sistemas heterogéneos multicámara, inteligencia distribuida y vídeos de alta resolución. Es necesario seguir investigando y desarrollando para conseguir los siguientes retos:

- Introducción de inteligencia en las cámaras.
- Creación de algoritmos flexibles.
- Habilitar el seguimiento utilizando cámaras de accionamiento eléctrico con un enfoque multiescala: detección del ancho del campo, extraer detalles importantes de los objetos cuando se le aplica una zoom a un determinado objeto.
- Desarrollo de algoritmos de adaptación y calibración automáticos que ajusten los parámetros de las cámaras (lente, zoom, apertura de visión, perspectiva, etc.) ante cambios en la escena que están monitorizando.

3.5. Tendencias tecnológicas para la videovigilancia.

Muchos sistemas de videovigilancia actuales todavía se basan en las tecnologías tradicionales: cámaras analógicas, cables coaxiales, grabadoras digitales. Sin embargo, el continuo avance en este aspecto ha dado lugar a la aparición de nuevas arquitecturas de los sistemas de videovigilancia, componentes y capacidades.

A continuación, se detallan algunas de las principales tendencias tecnológicas que los expertos prevén en los próximos años:

- La utilización de redes IP en sistemas de videovigilancia. En la actualidad es utilizada para la monitorización de extensas áreas. Los sistemas de videovigilancia con IP podrían ser utilizados por usuarios de todo tipo, si el coste de la cámara IP se redujera considerablemente.

- La utilización de megapíxeles en las cámaras suponen un gran paso en los sistemas de videovigilancia, ya que ofrecen alta resolución y una amplia cobertura. Además también mejoran el rendimiento del proceso de análisis. Por tanto, puede convertirse en una de las cámaras más utilizadas.

- El estándar de compresión H.264 (MPEG-4 Parte 10/AVC de Advanced Video Coding) debe establecerse en los sistemas de videovigilancia. Este estándar permite ahorrar ancho de banda y espacio de almacenamiento. La utilización de este estándar ayudará a la utilización de cámaras utilizando megapíxeles.

- Los sistemas de vigilancia contarán con una red de múltiples cámaras operando en tiempo real. Estos sistemas tenderán procesar la información y a distribuir las grabaciones en nodos distribuidos (ya sea en las propias cámaras o codificadores) con el objetivo de disminuir la transmisión de datos.

- Mejorar la integridad y la interoperabilidad con los diversos sistemas de vigilancia, centralizando su control. Por tanto, los sistemas de control de acceso, soluciones biométricas, RFID, sistemas de detección de intrusos, etc. deben estar conectados a la central de control.

- Los SVI y los productos de análisis de vídeo deben progresar para que sean más robustos y eficiente en aplicaciones de seguridad. La detección, seguimiento y análisis de objetos (personas,vehículos, caras, matrículas, etc) en escenas complejas con alta concentración de objetos, debe mejorar así como el análisis de comportamiento serán los campos más activos en la investigación de sistemas de vigilancia.

- La utilización de sistemas de videovigilancia se extenderán más allá del área de la seguridad. Se solicitarán aplicaciones para el análisis del marketing, estudio del comportamiento de los consumidores, además de las funciones de vigilancia dentro de la organización.

3.6. Análisis de trayectorias y patrones de aprendizaje.

En general, la primera etapa del camino de reconocimiento requiere realizar un seguimiento de los objetos individuales, estableciendo la correspondencia fotograma a fotograma. Esta tarea ha sido ampliamente estudiada por numerosos investigadores [ME].

En la extracción de trayectorias de peatones, el método debe ser capaz de hacer frente a la aparición de una amplia gama de movimientos incoherentes. Por ejemplo, en determinadas vías las personas realizan una trayectoria irregular al tratar de esquivar a otras personas u obtáculos que aparecen en la vía, o bien por el propio caminar serpeante de ciertas personas cuando caminan individualmente. Las variaciones de las trayectorias dependen en cierta medida de la amplitud del camino que siguen. Esto supone una dificultad añadida a la hora de realizar un reconocimiento de la trayectoria de los objetos. Además se dificulta aún más la representación de las trayectorias cuando las personas realizan una trayectoria saliendo de la vía por la que caminan, por ejemplo si un peatón siguiese un camino de un jardín y para evitar chocar con un obstáculo estático o con otras personas tuviera que pisar el césped abandonando el camino marcado (ver figura 3.15).

Fernyhough [Fer96] construyó una base de datos en la que almacenaba las trayectorias reconocidas en el entorno observado, de objetos acumuladas durante un período de tiempo. Fernyhough obtenía las regiones del entorno en las que se producían las diferentes trayectorias, para ello utilizaba una clasificación propuesta por Howarth y Buxton [HB92], en la que dividian el espacio en regiones que representaban las áreas de comportamientos similares.

Jonhson [JH95] utilizó un modelo basado en la cuantificación vectorial para la distribución de trayectorias utilizando redes neuronales. Una trayectoria era representada por un conjunto de vectores que contemplaban la posición y la velocidad. Se almacenaban muchos vectores para formar una distribución en el plano de la imagen monitorizada, formada por un conjunto de vectores. La cuatificación de los vectores se realizó siguiendo dos redes de aprendizaje muy competitivas, la primera fue, modelar la distribución de los vectores de flujo, y la segunda fue la modelación de las distribuciones de las trayectorias. Una capa de neuronas conecta las dos redes e introduce un elemento de almacenamiento

FIGURA 3.15: Representación de trayectorias realizadas arbitrariamente.

dentro de la arquitectura de red. El modelo permite la detección de movimientos atípicos de manera instantánea lo que facilita la detección de trayectorias atípicas. Este modelo se amplió para satisfacer las necesidades de predicción de trayectorias en [N.98], pero no proporciona un mecanismo de identificación de las trayectorias.

Sumpter y Bulpitt [SB98] adoptaron la metodología de Johnson [JH95] pero aumentando la realimentación del sistema con neutonas desde la salida hasta la entrada, para aprender patrones de activación. A continuación, experimentó el sistema para predecir el comportamiento de un rebaño de patos frente a las amenazas de depredadores.

Boyd [BV99] clasifica el flujo de movimiento (manualmente) en diferentes regiones utilizando un modelo gráfico, con enlaces que representan el movimiento entre regiones adyacentes, realizando un análisis basado en una red tomográfica para modelar estadísticamente el flujo de datos en una red de comunicaciones informática. El método evita la extracción de las trayectorias, registrando únicamente los cambios estimados acerca de la densidad de objetos entre regiones adyacentes para modelar el flujo de patrones. Este método tampoco cumple con la identificación y predicción necesaria.

3.7. Ejemplos Sistemas de vídeo vigilancia.

Existen diferentes formas de realizar un análisis y representación de trayectorias, a continuación se van a detallar algunas de ellas:

3.7.1. Aprendizaje de patrones a partir del movimiento de objetos para la detección de anomalías.

Según [ABS], para realizar el análisis de las trayectorias es necesario utilizar los datos obtenidos en la fase de seguimiento (tracking). Estos datos servirán como datos de entrada para realizar el análisis de trayectorias. El módulo de seguimiento de objetos produce un conjunto de m trayectorias $\{T_1, T_2, T_3, ..., T_i, ...T_m\}$ donde cada una de las trayectorias está formada por una serie de observaciones del mismo objeto en diferentes instantes de tiempo. Por ejemplo, $T_i = \{O_1, O_2, ..., O_j, ..., O_n\}$ donde cada $O_j = (t, x, y, w, h)$. cada O_j contiene las principales características del objeto en cada reconocido en la escena en cada instante de tiempo, de forma que:

- t contiene el tiempo en el que se ha detectado el objeto.
- (x, y) almacenan la posición en la que se ha detectado el objeto.
- w contiene la anchura del objeto detectado.
- h almacena la altura del objeto.

Se utilizan estas caracterísiticas comentadas anteriormente, para generar un conjunto de vectores (ver figura3.16) de transición que serán utilizados para entrenar los modelos estadísticos para que éstos proporcionen información acerca de su movimiento y su tamaño. Cada vez que un objeto es detectado en una nueva posición se realiza una computación del conjunto total de vectores de transición con el objetivo de predecir futuras posiciones del objeto a lo largo de su trayectoria. Además se obtiene la velocidad de un objeto, comparando la posición del objeto en un determinado O_j con la posición del objeto en las sucesivas posiciones que vaya recorriendo $\{O_{j+1}, ..., O_{j+\tau}\}$. También se calculan los vectores de transición $\{\gamma_j^{j+1}, ..., \gamma_j^{j+\tau}\}$ donde $\gamma_j^{j+\tau} = (x_{j+\tau} + y_{j+\tau}, \tau, w_j, h_j)$. La posición final del objeto que es $(x_{j+\tau} + y_{j+\tau})$ se obtiene de la posición del objeto $O_{j+\tau}$. (w_j, h_j) representan el tamaño del objeto detectado en el entorno. τ es el número de veces que se ha detectado el objeto en la escena monitorizada.

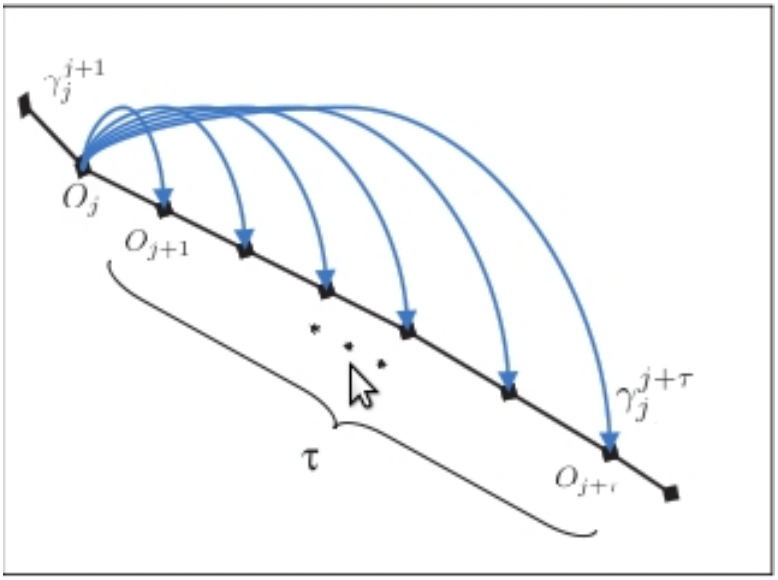

FIGURA 3.16: Trayectoria del objeto formada por un conjunto de vectores (color azul). O_j y O_k representan dos detecciones del objeto diferentes a lo largo de su trayectoria y γ_j^k representa el vector transición entre O_j y O_k.

Para construir patrones modelos se utilizan las propiedades de movimiento y tamaño de lo objetos que se han especificado anteriormente. En el análisis se utiliza una variable aleatoria Γ_l compuesta por cinco dimensiones para cada uno de los píxeles l donde $\gamma = (x', y', \delta t, w_l, h_l)$ representa un resultado particular de Γ_l. Cada vector de transición generado por la detección de un objeto está compuesto de cinco variables.

La función de densidad probabilística (pdf) sobre las propiedadades del espacio está modelada por un modelo Gaussiano mixto (GMM). En el pdf se crea la probabilidad para cada píxel de las escena y modela la probabilidad de que esa posición sea el origen de la transición. En el pdf se estima la probabilidad de detectar el movimiento de un objeto de un tamaño determinado a una posición determinada y en un tiempo determinado. Además el pdf puede obtener, en áreas de transicón de múltiples objetos en diferente dirección, puede obtener, la dirección, la velocidad y el tamaño de los distintos objetos.

El aprendizaje de modelos se realiza después de un determinado período de acumulación de datos procedentes de la etapa de seguimiento. La duración depende del grado de concentración de la escena y de la precisión que necesiten los modelos. Para cualquier posición l en la escena, todas las posiciones detectadas de todas las trayectorias de los objetos contribuyen a la localización pdf de dicha posición.

El pdf para la variable aleatoria Γ_l es generado utilizando las instancias entrenadas de los vectores γ's con respecto a la posición l.

El método de entrenamiento se desribe a continuación y es empleado para todos los píxeles. Un modelo GMM se utiliza para modelar el pdf de una variable aleatoria Γ_l. La probabilidad de que un γ pertenezca a GMM viene dada por la siguiente ecuación:

$$P(\Gamma_l = \gamma|\theta_l) = \sum_{i=1}^{n} \alpha_l^i p(\gamma|\theta_l^i) \tag{3.16}$$

donde n es el número de componentes detectados en el entorno, θ_l^i es el conjunto de parámetros definiendo el componente ith con anchura α_l^i, y $\theta \equiv \{\theta_l^1, ..., \theta_l^n, \alpha_l^1 ..., \alpha_l^n\}$ definen el conjunto de parámetros completos necesarios para especificar el modelo de mezcla. Cada componente es representado como una distribución Gaussiana de la forma:

$$p(\gamma|\theta_l^i) = \frac{1}{(2\pi)^{d/2}|\sum_l^i|^{1/2}} e^{-1/2(\gamma-\mu_l^i)^\tau \sum_l^{i\,-1}(\gamma-\mu_l^i)} \tag{3.17}$$

donde d es la dimensionalidad del modelo y $\theta_l^i = \{\mu_l^i, \sum_l^i\}$ son los parámetros del modelo.

3.7.2. Aprendizaje a partir de mezcla de modelos.

El cálculo de los parámetros de la GMM se realizan mediante un algoritmo basado en la maximización de la expectativa (Expectative Maximization, EM) el cuál fue propuesto por Figuereiredo y Jain [FJ02]. El enfoque utilizado en este sistema, presenta tres limitaciones con respecto al algoritmo básico EM. La primera, es que el número de componentes puede ser variable. Este algoritmo calcula en número de componentes eliminando aquellos que no son compatibles con los datos. En segundo lugar, no necesita ser inicializado ni estar formado inicialmente por un número elevado de componentes que se extienden a lo largo de los datos. Tercero, este algoritmo también evita la convergencia hacia una estimación simple cercana a un conjunto de parámetros en el espacio. Los detalles de este algoritmo se pueden ver en [FJ02].

Después de terminar el proceso de aprendizaje sobre la escena, los parámetros de GMM han almacenado la posición de cada uno de los píxeles como modelo de la escena. Dada una nueva detección de un objeto en la escena, si sólo tenemos en cuenta el punto central del contorno que lo rodea, entonces los modelos creados comienzan a dispersarse. Para lograr un mejor resultado espacial, se actualizan todos los píxeles vecinos del contorno que envuelve al objeto detectado. Hay que tener en cuenta que éste algoritmo no realiza una fusión para obtener las principales trayectorias en la escena. Esto reduce las probabilidades de realizar una estimación incorrecta entre dos o más trayectorias. Otra

característica importante del algoritmo es la capacidad de mejorar el aprendizaje de patrones de movimiento, y adaptarse a los cambios de comportamiento producidos en la escena.

3.7.3. Recuperación de las trayectorias de los objetos para la estimación de la geometría de carriles utilizando cámaras de vigilancia de tráfico no estacionarias.

En [JMSV04] se realiza un algoritmo para la generación de trayectorias de vehículos, a través de la segmentación y seguimiento de los objetos en movimiento teniendo en cuenta la posible oclusión parcial o total del objeto a lo largo de su trayectoria, sin necesidad de calibrar las cámaras. Se utilizan cámaras que permiten la ampliación de la panorámica (Pan Tilt Zoom (PTZ)). En [JMSV04] se plantea un algoritmo de separación y fusión en la predicción de trayectorias (Predictive Trajectory Merge and Split (PTSM)). El sistema propuesto es un enfoque multietapa que almacena las trayectorias de los objetos en movimiento y a continuación realiza una estimación sobre la geometría de los carriles de circulación en diferentes vías. Un resumen del sistema propuesto se puede ver en la figura 3.17.

Según [JMSV04] algoritmos anteriores necesitaban conocer previamente el número de carriles de los que está compuesto una vía o la geometría de ésta. Sin embargo, con este sistema no es necesario este conocimiento previo.

Las trayectorias reconocidas son preprocesadas para eliminar posibles inconsistencias o ruido que han podido aparecer en la captación de las imágenes. A través del análisis de todas las trayectorias generadas se puede obtener la geometría de los carriles que forman la carretera, gracias a aquellas coordenadas que son más repetitivas. Además se puede conocer la dirección que llevan los vehículos añadiendo una señal de comienzo y de final de la trayectoria reconocida.

Después las trayectorias son representadas con polinomios de grado M utilizando después el algoritmo de los mínimos cuadrados. Con esto se puede conocer la dirección de la trayectoria. Se comienza con $M = 1$, además se realiza un promedio del error de ajuste para determinar con la mayor exactitud el valor de M. Si el valor del error es mayor que 1 entonces se incrementa M en una unidad y se reajusta la trayectoria $(M \leq 3)$. Se ha detectado en numerosas pruebas en carreteras con elevado número de carriles que las trayectoias de los vehículos son representadas con polinomios de orden 3 es suficiente para describir la trayectorias de los vehículos con su curvatura adecuada. Finalmente se utiliza

el algoritmo RANSAC [MAF81] que utiliza los mínimos cuadrados para eliminar los casos extremos o que no hayan sido clasificados por pérdida de puntos en el tracking.

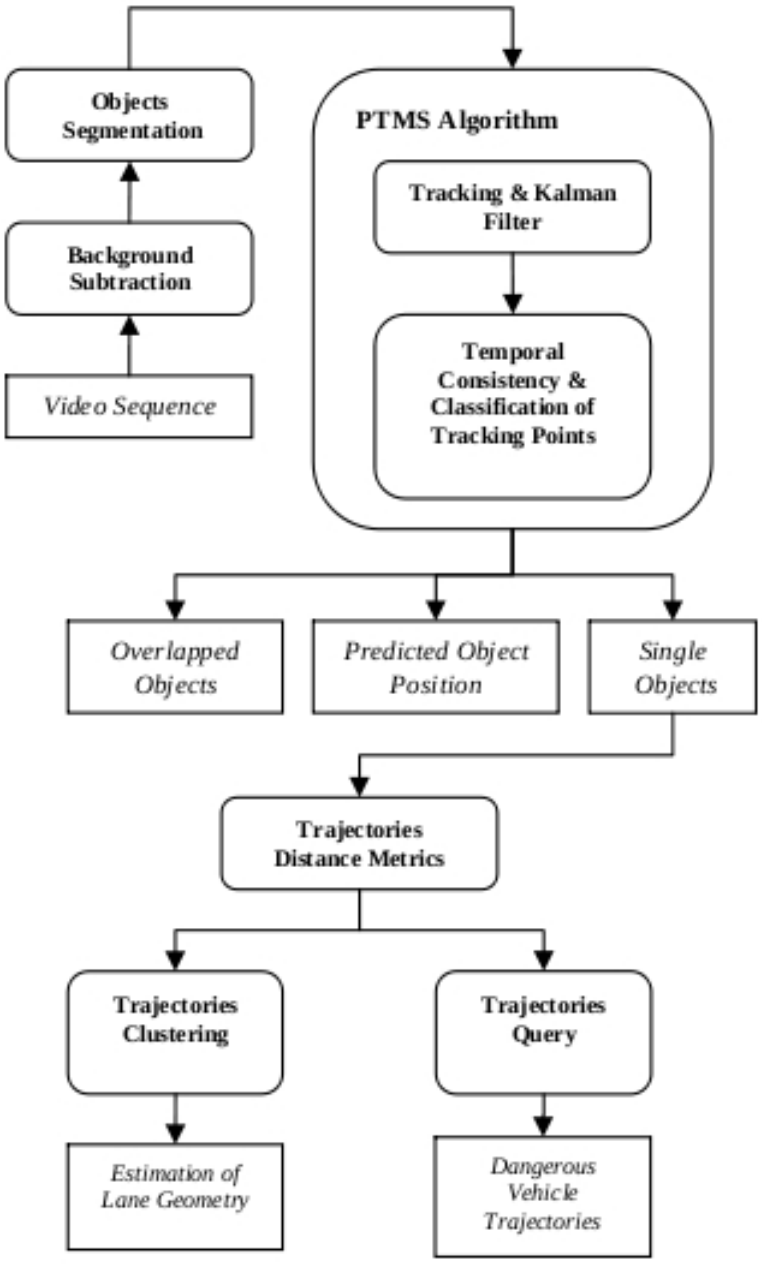

FIGURA 3.17: Resumen de los componentes de PTMS

En este sistema se trabaja con puntos arbitrarios. El sistema se muestra muy sensible a los puntos extremos y las operaciones a realizar son muy costosas ya que su complejidad es $O(MN)$ donde M y N con conjuntos de puntos que representan el tamaño de la trayectoria. El tiempo de computación puede ser reducido considerablemente si se utilizan coeficientes de los polinomios para evaluar trayectoria. Se utiliza una medida de distancia máxima para

comparar las trayectorias y comprobar si se encuentran dentro de los límites o si superan esta distancia significa que pertenecen a otra trayectoria. Si la distancia entre las trayectorias es mínima entonces se puede decir que tiene direcciones similares. Existe una distancia media que es utilizada para clasificar aquellas trayectorias que tienen una forma peculiar. También se utiliza una distancia mínima que sirve para comparar trayectorias que son muy parecidas o existen puntos de intersección con la trayectoria de referencia.

En la figura 3.18 podemos ver la representación de los diferentes puntos correspondientes a los posicionesde vehículos generadas por el algoritmo PTMS.

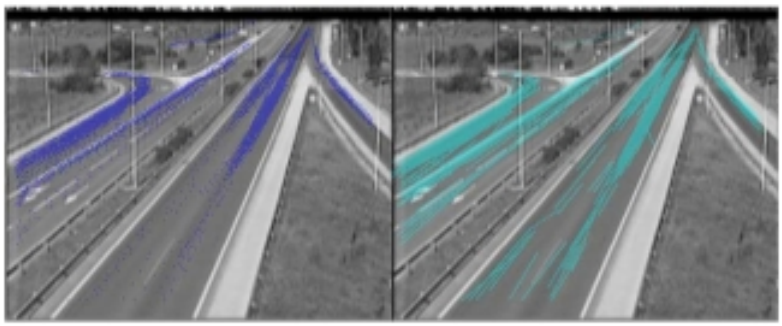

FIGURA 3.18: A la izquierda se puede observar los puntos que corresponden a trayectorias de vehículos y a la derecha se puede ver la interpolación de estos puntos utilizando polinomios de bajo orden.

Una vez obtenidos las posiciones de vehículos en el entorno, se realiza una clasificación de éstos dejando únicamente los patrones que representan la dirección de cada carril. La represetacion de las trayectorias mediante puntos en la escena representa las trayectorias de vehículos individuales (ver figura 3.18) para después realizar una estimación de los carriles centrales que componen la carretera (ver imagen 3.19).

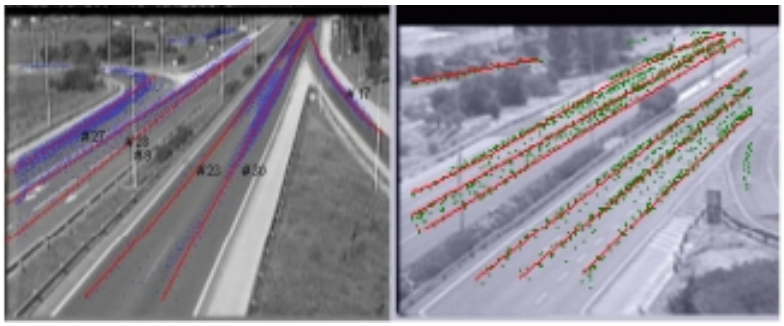

FIGURA 3.19: Representación de los carriles centrales a traveś de la previa representación de puntos en la escena.

Antes de realizar la estimación de los carriles centrales de la carretera se le aplica el método de RANSAC para limpiar aquellos puntos extremos provocados por alguna

situación extraña o aparición de ruido en la escena. Una vez aplicado este método se obtiene la estimación de los carriles (ver imagen 3.20).

FIGURA 3.20: Representación de los carriles centrales, depués de aplicar el método RANSAC.

Estos resultados pueden ser utilizados en vías con alta concentración de vehículos para analizar el comportamiento del flujo de tráfico en tiempo real. El sistema realiza una estimación de la velocidad de los vehículos, frecuencia de cambios de carril, detección de accidentes, y clasificación de comportamientos anómalos.

3.7.4. Detección de rutas en vídeo vigilancia.

En [ME] se presenta un modelo de extracción automática de las trayectoria de peatones a partir de secuencia de vídeo real. Los patrones de trayectorias mediante la acumulación de trayectoias capturadas en un largo período de tiempo, que posteriormente servirán para clasificar las nuevas trayectorias reconocidas. Los patrones de trayectorias reconocidos suponen una compresión eficiente de los datos necesarios de las trayectorias para ser almacenados. Además el modelo puede ser utilizado para realizar una predicción probabilística de la posición de un peatón en las futuras posiciones y también puede servir de ayuda en la detección de comportamientos anormales debido a su atípico movimiento.

En el modelo realizado por Dimitrios Makris y Tim Ellis en [ME] las trayectorias son representadas por un eje central y compuesto por un conjunto de nodos, los cuáles representan una media de la trayectoria. La trayectoria está delimitada por una envoltura que delimita el espacio por el que pasan todas las trayectorias individuales. Los nodos que forman el eje central de la trayectoria se encuentran todos a la misma distancia. Cada trayectoria tiene dos nodos marcados, uno que indica el comienzo de la trayectoria y otro que indica el final de la trayectoria, y que generalmente corresponden con zonas denominadas como zonas de entrada o salida en la escena (ver figura 3.21).

Además, en el modelo presenta un nivel semántico que se obtiene a través de los componentes del sistema. Las trayectorias tienen las siguientes características:

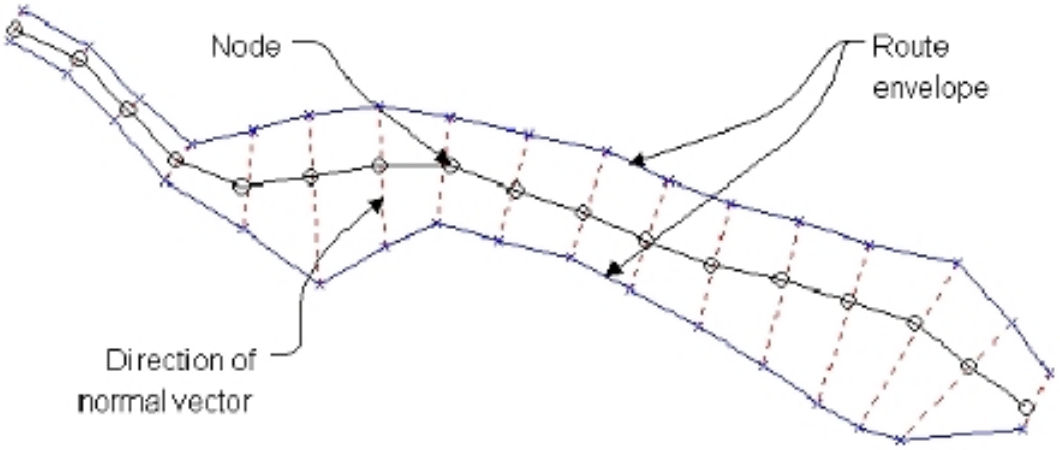

FIGURA 3.21: Representación de la trayectoria siguiendo el modelo de Dimitrios Makris y Tim Ellis.

1. Zonas de entrada y salida: estas zonas son aquellas en las que los peatones entran y salen en la escena.

2. Zonas de intersección: estas zonas son aquellas en las que las trayectorias se cruzan entre sí superponiéndose.

Los peatones entran y salen de la escena por zonas específicas y siguen un trayectoria determinada. Una trayectoria de un objeto viene determinada desde que el objeto es reconocido que entra hasta que éste sale de la escena, las uniones son aquellas zonas en las que las trayectoria de un objeto se bifurcan o se fusionan. Para representar todas estas zonas se utiliza un grafo que representa la topología de la red de nodos (ver figura 3.22).

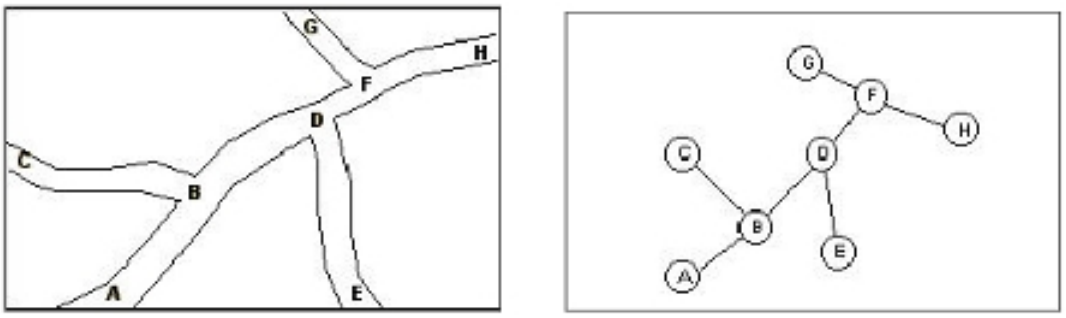

FIGURA 3.22: La imagen de la izquierda es una representación espacial de los caminos, mientras que la imagen de la derecha representa la topología de puntos de entrada y salida combinados con puntos de bifurcación o unión de trayectorias.

Los modelos son aprendidos a partir de una secuencia de imágenes en las que se extraen las trayectorias de los objetos. Las trayectorias son agrupadas mediante un análisis

geométrico en el que se compara la distancia de una trayectoria reconocida con respecto a la envoltura de las trayectorias modelo.

Las trayectorias son aprendidas mediante un conjunto de trayectorias. La descripción de las trayectorias es almacenada en una base de datos. El algoritmo de aprendizaje de trayectorias obtiene la trayectoria reconocida recientemente y realiza una comparación con cada una de las trayectorias que existen en la base de datos. Si se detecta que coincide con alguna de ellas, entonces la trayectoria almacenada es actualizada, si no existe coincidencia con ninguna, pues se inicializa una nueva trayectoria. Después de que una trayectoria coincida con alguna de las trayectorias existentes en la base de datos, se realiza una fusión con las trayectorias de la base de datos. En general, la base de datos contendrá un pequeño número de trayectorias almacenadas, que representarán las principales zonas por las que se producen movimientos de peatones. Las trayectorias que apenas han sido actualizadas son finalmente descartadas.

Cada trayectoria es comparada con las trayectorias existentes en la base de datos. En la comparación se utiliza una métrica que mide la distancia entre la trayectoria modelo y la nueva trayectoria reconocida. La medida de distancia utiliza es la máxima separación entre la trayectoria modelo y la recientemente reconocida(ver figura 3.23)

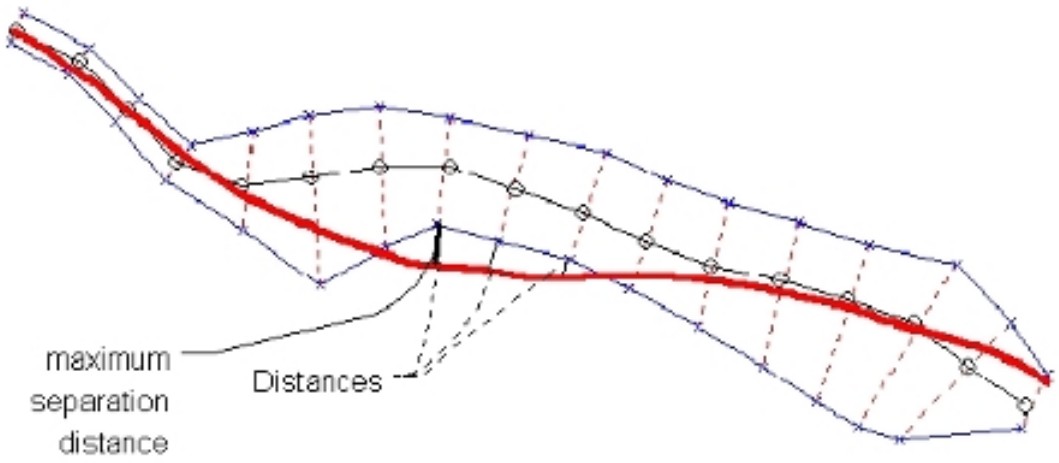

FIGURA 3.23: Comparación entre una trayectoria modelo (almacenada en la base de conocimiento) y una trayectoria reconocida recientemente.

Para la obtención de la máxima distancia de separación entre la trayectoria modelo y la actualmente reconocida es necesario realizar una comparación de cada nodo que define a la trayectoria recientemente reconocida y los nodos correspondientes del contorno (envoltorio) definido por la trayectoria modelo. Si el nodo de la nueva trayectoria está dentro de la trayectoria modelo, entonces el valor de la distancia es negativo, de lo contrario el valor de la distancia será un valor positivo y contendrá la distancia euclídea entre los nodos.

De todas las comparaciones de cada uno de los nodos, se obtiene aquella distancia cuyo valor es mayor. Si la distancia máxima supera un umbral determinado, entonces no existe coincidencia entre la nueva trayectoria reconocida y el modelo de trayectoria almacenada. Si la trayectoria no coincide con ninguna de las trayectorias modelo almacenadas en la base de conocimiento, entonces es inicializada como una nueva trayectoria.

En caso de que la máxima distancia de separación entre una trayectoria recientemente reconocida y una trayectoria modelo de la base de conocimiento, sea inferior al umbral establecido, se realiza un proceso de fusión entre ambas trayectorias. En este proceso de fusión, se la trayectoria que contenga la máxima anchura, es la trayectoria utilizada como trayectoria principal y la otra queda como trayectoria secundari. Para realizar la fusión de las trayectorias se van actualizando nodo a nodo con respecto a la trayectoria principal. Aquellos nodos que no coincidan con la trayectoria principal se realiza una estimación del nodo de la trayectoria secundaria con el nodo de la trayectoria principal.

3.7.5. Sistema de visión cognitiva.

La universidad de Kingston presenta un sistema de videovigilancia experimental(KUES) [DMB08] presentan un sistema de videovigilancia llamado KUES centrándose en los aspectos cognitivos. Se trata de un modelo de aprendizaje automático de una escena estática. Después una gran cantidad de observaciones, el sistema realiza una identificación espacial de situaciones estableciendo consistencia entre los datos, utilizando un mecanismo de atención de movimiento. También se realiza cómo con la utilización de patrones que representan situaciones típicas pueden ser utilizadas para la detección de situaciones sospechosas. El esquema del sistema KUES se puede ver en la figura 3.24.

Cada cámara de vigilancia genera un flujo de vídeo independiente. El módulo de detección de movimiento (Motion Detection Module) [DM02] establece un modelo de píxeles y el fondo del entorno observado por la vista de cada cámara e identifica los píxeles en cada frame donde aparece movimiento (foreground). asumiendo que la variación de valores en los píxeles es a causa del movimiento de objetos y no por la aparición de ruido en la escena. Despúes los píxeles de foreground son clasificados en Blobs (Binary Large Objects).

El módulo de seguimiento (Motion Tracking Module), proporciona una trayectoria para cada objeto individual detectado, que es representada a través de una secuencia en el tiempo de las posiciones del objeto en el entorno observado. Las posiciones que se obtienen son las del centro del objeto desde una vista de una cámara. Por este motivo, el seguimiento de los objetos realiza una correspondencia de los blobs en la secuencia de frames reconocidos

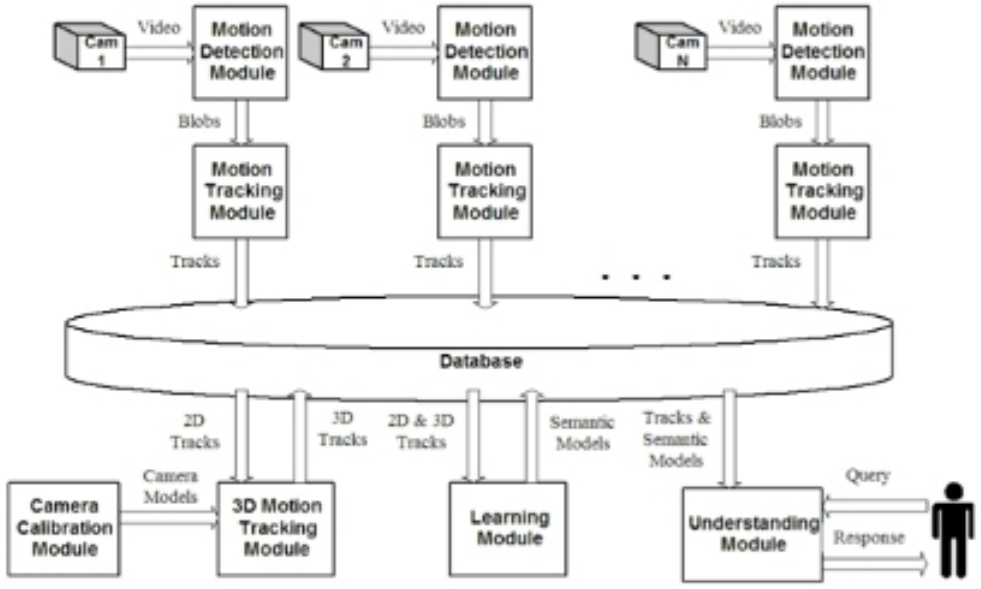

FIGURA 3.24: Esquema de módulos del sistema KUES.

y resuelve posibles ambigüedades causadas por las oclusiones. Los Blobs contienen un conjunto de características (posición, velocidad, tamaño y color) que son utilizadas como criterios de camparación en los algoritmos.

El módulo de seguimiento 3D (3D Motion Tracking Module) permite codificar el movimiento completo e individual de un objeto dentro del entorno observado. La imagen proporcionada por las diferentes cámaras son combinadas de acuerdo con modelos geométricos obtenidos tras un proceso de calibración. Las trayectorias 3D son representadas en términos de un plano utilizando un sistemas de coordenadas común. La ventaja de realizar una representación de las trayectorias en 3D es que permite la representación de objetos en términos de una escena real.

El módulo de aprendizaje (Learning Module) realiza un seguimiento de 2D a 3D para generar modelos semánticos y de actividad. Los modelos semánticos reconocidos de la escena ayudan a entender el movimiento y responder a consultas realizadas por el personal de seguridad. Mientras que los modelos de actividad son utilizados para detectar situaciones sospechosas no recogidas en estos modelos.

El sistema de vigilancia KUES consiste en la utilización de varios módulos que operan en diferentes niveles. Además, éste requiere la atención de personal de seguridad para operar en diferentes entornos durante un período de tiempo para entender las situaciones normales en la escena y emitir alarmas ante situaciones sospechosas.

Dimitrios Makris y Tim Ellis han propuesto un esquema general para describir la actividad observada por un sistema de videovigilancia. El modelo proporciona capacidad

cognitiva a los sistemas de videovigilancia, para proporcionar un conocimiento de alto nivel a las actividades que ocurren en la escena. Las situaciones que ocurren en la escena se pueden clasificar en tres categorías semánticas(objetos, acciones y eventos).

Específicamente las actividades observadas pueden estar basados en un diccionario semántico que contiene tres categorías principales:

1. Detección de objetos (peatones o vehículos).
2. Acciones([And89], [DM04]).
3. Característica estáticas de la escena.

De acuerdo con este esquema los objetos pueden realizar movimientos dentro de la escena y pueden interactuar con otros objetos o con las características de la escena. Más formalmente, las expresiones semánticas son creadas utilizando el movimiento de objetos como sujeto de las expresiones semánticas, las acciones como verbos y las características estáticas de la escena como objetos o complementos de la expresión.

El modelo se basa en un sistema de videovigilancia con la habilidad de aprender automáticamente utilizando los datos obtenidos por el sistema durante un extenso período de tiempo.El aprendizaje automático se puede utilizar para crear modelos diferentes de cada objetos (es decir, peatones, vehículos,camiones, etc) e identificar automáticamente el tipo de situación que se produce en un lugar particular de la escena.

La geometría de la escena condiciona los movimientos de los objetos. Además los tipos de eventos que se producen se pueden asociar con regiones específicas. Por ejemplo, restringir la circulación de vehículos que circulan por un carril específico con una dirección específica. El método de aprendizaje semántico que atiende a los mecanismos de detección de movimientos de los objetos empleados en el modelo KUES es actualmente un enfoque de ingeniería inversa para la identificación de estas regiones (ver figura 3.25). Este modelo hace una libre correspondencia con el método [Joh98], que puede aprender

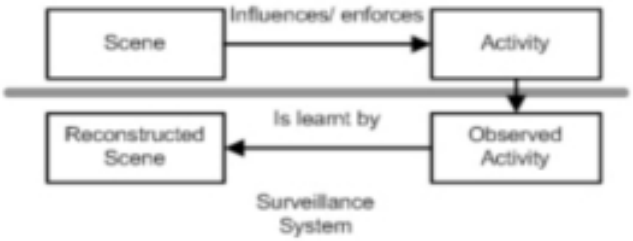

FIGURA 3.25: Reconstrucción de la actividad de la escena.

automáticamente el posicionamiento de cámaras que componen una red, determinando si

las vistas de cada cámara se solapan, están muy distantes, etc. Puede integrar un modelo que detecte la actividad que aparece en la escena observada. El método está basado en una correspondencia en la detección de los objetos captados por las distintas vistas de las cámaras. El método permite comunicarse con las cámaras adyacentes, aunque las vistas de éstas no se solapen. También proporciona posibles movimientos de objetos en aquellos espacios o huecos entre las distintas vistas de la cámara.

La detección automática de actividades sospechosas es el mayor requisito de los sistemas de videovigilancia, debido a la incapacidad del se humano de mantener su atención ante un monitor y detectar las situaciones sospechosas durante un largo período de tiempo.

Se realiza un modelado de las trayectorias normales y aprende situaciones normales utilizando un modelo probabilístico. Posteriormente se relaciona el comportamiento atípico con situaciones anormales en la escena. El modelado de la escena se construye dependiendo de las acciones que se produzcan en la escena después de un período de observación. Posteriormente se asignan patrones de comportamientos típicos o normales. Se incorpora un método de detección automática de situaciones atípicas que pueden ser sospechosa basándose en trayectorias descritas por los modelos ocultos de Markov (RBHMM).

Los estados de un RBHMM son definidos como los nodos de todos los modelos de trayectorias aceptados, añadiendo dos estados extra:

- **"Estado fuera de cualquier nodo"**: Este estado indica que la situación reconocida no se encuentra entre las situaciones modeladas anteriormente.
- **"Estado final"**: Estado que indica la conclusión de la monitorización de la escena.

Esto permite derivar los nodos en trayectorias unidireccionales, de manera que permite incorporar la información de direccionalidad a cada nodo.

3.7.6. Interpretación semántica de actividades de objetos en sistemas de vídeo vigilancia.

Jiangung Lou, Qifeng Liu, Tieniu Tan and Weiming Hu [JLH02], propusieron un framework de interpretación semántica del comportamiento de peatones y vehículos. Las trayectorias tanto de peatones como de vehículos son almacenadas tras un proceso de seguimiento (tracking) y se realiza un análisis utilizando una clasificación dinámica que está basada en una interpretación semántica de alto nivel.

En un sistema de videovigilancia, las trayectorias son adquiridas utilizando algoritmos de seguimiento de bajo nivel. En este modelo se centra únicamente en la interpretación

semántica de las situaciones. Las trayectorias se generan almacenando la posición y velocidad de los objetos en cada frame.

El análisis de patrones de trayectorias, junto con la clasificación automática de las trayectorias en patrones, es una tarea muy importante para la correcta interpretación de las situaciones detectadas. A menudo se puede analizar la actividad de una trayectoria reconocida, comparandola con los patrones de trayectorias reconocidos, analizando la velocidad y otra información dinámica contenida en la trayectoria del objeto. El sistema realiza una clasificación en árbol, tal y como muestra la figura 3.26.

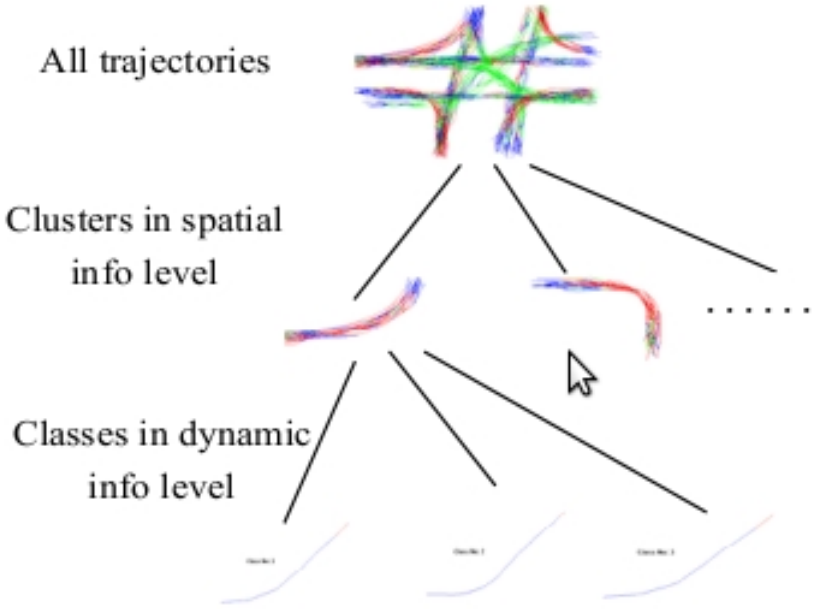

FIGURA 3.26: Clasificación de las trayectorias en árbol.

Se obtiene la información espacial de cada trayectoria para clasificarlas de forma dinámica en diferentes patrones que representan situaciones de diferentes tipos. Para esta clasificación se utilizan algoritmos de comparación.

Uno de los principales problemas que se presenta es cómo obtener el grado de similitud entre dos trayectorias. Se puede realizar un análisis espacial y dinámico de las trayectorias. En [NS00] y [JH96] utilizan vectores cuantificados para realizar una correspondencia en el espacio euclídeo, el cual viene determinado por las caracterísiticas del objeto y la velocidad de éste. EL algoritmo no utiliza la información global de las trayectorias, sino que realiza una correspondencia de las características en el espacio. En [J.F00] los autores utilizan un porcentaje de solapamiento de píxeles como medida de similitud entre dos trayectorias; y si existe un solapamiento del 80 % entonces es clasificada como idéntica a la trayectoria a la que se parece. Esta técnica puede ser denominada como una clasificación global de las

trayectorias, pero depende de un valor umbral, el cuál puede ser erróneo si aparece ruido en la etapa de seguimiento. Alguno de los problemas considerados como ruido que puede aparecer en el seguimiento de los objetos, es el problema de la oclusión.

Jiangung Lou, Qifeng Liu, Tieniu Tan and Weiming Hu definen una nueva fórmula de la distancia para obtener el grado de similitud entre trayectorias. Esta fórmula es muy similar a la distancia de Hausdorff. Esta distancia puede considerarse información global de las trayectorias. Dadas dos trayectorias A y B donde A tiene t puntos y B tiene T puntos, la distancia espacial entre las dos trayectorias se define:

$$D_c = min_{A,B}\{D_{A,B}, D_{B,A}\} D_{A,B} = max_{i=0..t}\{min_{j=0..T}(d_{i,j})\} D_{B,A} = max_{i=0..T}\{min_{j=0..t}(d_{i,j})\} \tag{3.18}$$

donde $D_{i,j}$ es la distancia Euclídea desde el punto i situado en una trayectoria hasta el punto j situado en la otra trayectoria. También se define una métrica que describe el grado de similitud entre dos trayectorias de forma dinámica:

$$D_t = min_{A,B}\{DV_{A,B}, DV_{B,A}\} \tag{3.19}$$

donde $DV_{A,B} = \frac{\sum_{alliinA} dv_{i,j}}{t+1}, j = arg_{0 \leq j \leq T} min(d_{i,j})$

$DV_{B,A} = \frac{\sum_{alliinB} dv_{i,j}}{T+1}, j = arg_{0 \leq j \leq t} min(d_{i,j})$

y $dv_{i,j}$ es la diferencia de velocidad entre el punto i situado en una trayectoria y el punto j situado en la otra trayectoria.

El objetivo en el análisis de una trayectoria es conocer las diferentes acciones de una trayectoria. Para realizar el análisis de trayectorias, Jiangung Lou, Qifeng Liu, Tieniu Tan and Weiming Hu presentan un método basado en los Modelos Ocultos de Markov (HMM)[FM98]. Cada trayectoria es divida en pequeñas porciones, y cada porción contiene 20 puntos. Se asignan acciones dentro de estas cuatro acciones posibles: avanzar hacia adelante, girar a la derecha, girar a la izquierda y detenerse. El método presenta dos características principales que lo diferencian del método que se encuentra en [FM98]:

- Cada trayectoria es fragmentada en partes, las cuáles se divide en algunos pequeños segmentos, posteriormente se modela cada fragmento utilizando HMM (se ha detectado que en la representación de un patrón aparecen diferentes acciones en una misma trayectoria, al fragmentarlo, en cada fragmento sólo se realiza una acción de las cuatro posibles anteriormente comentadas.).
- Las curvas que tienen una trayectoria se obtienen por comparación de la velocidad de traslación y de la velocidad angular.

Para realizar la clasificación de las trayectorias se realiza en árbol. Las trayectorias de los objetos se clasifican de manera dinámica realizando una comparación con los patrones definidos y se va realizando al mismo tiempo que los objetos van siendo detectados y seguidos con la trayectoria de los objetos incompleta. Esta clasificación se realiza mediante un clasificador Bayesiano. La distancia desde un punto A de una trayectoria con el punto correspondiente de una trayectoria patrón es $d_i = min_{<1 j<T}(d_{i,j})$. Además se asume que este d_i, $i = 1..t$ satisface una distribución Gaussiana. Esta suposición es razonable cuando dos trayectorias son clasificadas por el mismo patrón. Además la densidad de probabilidad conjunta es:

$$p(x|k) = \prod_{i=1}^{t} p(x_i|k) \tag{3.20}$$

donde $p_i(x|k)$ es la distribución de densidad Gaussiana del punto x de una trayectoria que pertenece al patrón k. Los parámetros de esta distribución Gaussiana pueden ser estimados por el cálculo de la matriz de dispersión en cada grupo. La densidad de probabilidad posterior puede ser:

$$p(k|x) = \frac{P(k) \cdot p(x|k)}{\sum_{allk} P(k) \cdot p(x|k)} \tag{3.21}$$

donde $P(k)$ es la probabilidad previa de la clase k y puede ser sustituida por ω_k.

Una vez que las trayectorias son completadas y clasificadas, la clasificación en árbol debe ser actualizada. Se actualiza, la anchura, los parámetros de una distribución Gaussiana y la representación de los patrones de trayectorias almacenados, teniendo en cuenta la trayectoria recientemente reconocida. A veces, puede ocurrir que una trayectoria no pueda ser clasificada en ningún grupo de trayectorias. En este caso, se afirma que se está produciendo una situación anómala y se puede emitir un aviso de la producción de éste. Al mismo tiempo, se añade al árbol una nueva rama que contiene la representación de esta trayectoria. El árbol estará cambiando constantemente, y a veces puede ocurrir que dos grupos de trayectorias similares representen a un mismo grupo de objetos. Cada cierto tiempo se debe examinar los patrones existentes y realizar una fusión de aquéllos patrones que tengan una alto grado de similitud.

En cuanto a las reglas semánticas que incorpara el método propuesto por Jiangung Lou, Qifeng Liu, Tieniu Tan and Weiming Hu, se introduce una gramática simple para generar descripciones en lenguaje natural. Debido a que en la mayoría de los sistemas de videovigilancia se realizan preguntas del tipo ¿Quién hace qué y en dónde? ¿y cómo?. Para responder a estas preguntas, el sistema ha implementado una única regla gramatical. La regla es: **Regla semántica : (El objeto)(la acción) en (el lugar) [a una velocidad**

despacio/normal/rápido] El contenido situado entre corchetes, es opcional mientras que el contenido entre paréntesis debe sustituirse por la información proporcionada por diferentes módulos del sistema.

El sistema no realiza una descripción semántica para cada frame, sólo se realizará una descripción semántica cuando suceda una de estas tres situaciones:

1. Esté ocurriendo una nueva acción.
2. El objeto esté entrando en una nueva región.
3. Se produzca una situación atípica.

3.7.7. Análisis de trayectorias para sistemas deportivos o de videovigilancia.

Y. L. de Meneses , P. Roduit,F. Luisier y J. Jacot [YLdM05] presentan un modelo para el análisis de trayectorias combinando las dimensiones espaciales y temporales. Ofrece la posibilidad de extraer información muy rica sobre el comportamiento de los objetos en movimiento. La metodología está basada en una extensión de los Modelos de Distribución de Puntos (PDM). Se introducen PDM, para analizar los movimientos de los objetos en el espacio, en el tiempo y en el dominio espacio-temporal. Las trayectorias son modeladas como una trayectoria media y un conjunto de modalidades de deformación, en los dominios del espcio, tiempo y espacio-temporal.

La herramienta ha sido probada para extraer los datos de cada trayectoria de objetos, mediante un sistema de visión, que realiza un seguimiento de vehículos circulando por un circuito cerrado. Se realiza una extensión de los PDM porque es necesario establecer la información dentro de un dominio temporal, el cual tiene un rango dinámico y una varianza diferente que la información de un dominio espacial. La proyección de las modalidades espacio-temporales sobre la dimensión espacial y temporal muestra que en realidad son diferentes a las modalidades exclusivamente espaciales y exclusivamente temporales. Esto indica que contienen información adicional con respecto a los PDMs. Sin embargo, en comparación con los PDMs espaciales, la interprestación de los resultados es mucho más complicada, porque los datos son más ricos. Se puede encontrar una relación con la trayectoria más lenta del conjunto de trayectorias lo que implica que temporalmente es la más larga. Parte del problema reside en el hecho de existir comportamiento desconocido. Por consiguiente sería deseable aplicar la metodología para el comportamiento de

trayectorias conocidas. Esto puede venir dado por una ecuación diferencial estocástica o por un robot, etc.

Este modelo pone de manifiesto la importancia de pre-procesamiento de los datos, y proporciona una solución para el problema de análisis de trayectorias en circuitos cerrados. Esto implica que el comienzo y el final de cada trayectoria estaba totalmente definido (1 vuelta) pero esto no podría resultar tan fácil, especialmente en espacios abiertos. Lo mismo se aplica para la información del dominio temporal porque los tiempos de las vueltas eran similares vuelta tras vuelta, pero contenían información relevante, para analizar por ejemplo, el estilo de conducción de cada piloto.

3.7.8. Aprendizaje de patrones de movimiento realizando una clasificación basada en los Modelos Ocultos de Markov(HMM).

Cuando una escena es supervisada durante todo el tiempo, la detección de patrones de movimiento puede ser aprendida y utilizada para detectar actividad atípica. El primer desafío conlleva a la fragmentación de las trayectorias, la dispersión de datos para el entrenamiento, y la localización en el tiempo de una desviación. Eran Swears, Anthony Hoogs and A.G. Amitha Perera en [ESP] presentan un enfoque de aprendizaje de patrones de movimiento y deteccion de situaciones anómalas utilizando la jerarquía de clasificación de los modelos ocultos de Markov(HMMs). En contínuo flujo de vídeo obtenido es utilizado para la estructura de aprendizaje y entrenamiento de los modelos HMMs, donde las trayectorias pueden ser de longitud muy variable, y las escenas pueden ser muy complejas compuestas por un número desconocido de patrones de movimiento. Los HMM son utilizados para clasificar las trayectorias reconocidas tanto para la clasificación de trayectorias que representan comportamientos normales, como para aquellas que representan comportamientos extraños. El algoritmo de clasificación de trayectorias utiliza una jerarquía de agrupación que utiliza la técnica de HMM teniendo en cuenta todos los parámetros de HMM(incluído el número de estados) utilizando un algoritmo de Máxima expectactiva(EM) y los criterio de Akaike[KB04].

Se han realizado múltiples propuestas utilizando HMM para clasificar las secuencias de datos. Smyth[Smy97] y Li y Biswas[CL99] proponen técnicas de clasificación de trayectorias estáticas en las que es necesario realizar un exhaustivo número de iteraciones de las distintas clases y estados, lo cuál resulta inviable para cuando es necesario tratar un gran número de datos. Otros han utilizado técnicas de clasificación para el aprendizaje de patrones de movimiento en vídeo [DM02] y [JLH02]. Para modelar un conjunto de trayectorias similares, ellos hacen una media de todas las trayectorias similares y modelan

el patrón definiendo una trayectoria central y su envoltura que está definida por los límites más extensos de todas las trayectorias clasificadas. Sin embargo, este método no puede detectar anomalías en la velocidad y dirección de los objetos detectados.

El enfoque presentado por [ESP] trata de aprender los movimientos normales de las trayectorias y velocidades observados utilizando HMMs[ES08]. Se utilizan HMM para aprender comportamientos normales en la escena, sin definir modelos, etiquetas, fuertes suposiciones, y con pocas trayectorias. Las trayectorias reconocidas son modeladas al mismo tiempo en el que se van detectando. Para ello, se utiliza un algoritmo de clasificación de HMM que combina las trayectorias que son similares con los modelos HMM, creando nuevos modelos o actualizando modelos anteriormente definidos. Más concretamente, el algoritmo compara cada posición de la trayectoria observada y su velocidad con los modelos HMM existentes(sólo con aquéllos que tengan unas características similares, en cuanto a posición y velocidad) que tengan mayor proximidad espacial con la trayectoria que han sido recibidas. Si la nueva trayectoria observada no coincide con ningún modelo HMM de los existentes, entonces es considerada como una situación anómala, y posiblemente puede ser agrupada como un nuevo patrón de movimiento HMM o se clasifica con aqueĺ modelo HMM al que más se asemeje. El modelo global de la escena se modifica, cuando los patrones HMMs cambian, bien por la detección de una nueva trayectoria no recogida anteriormente o bien por la modificación de algún patrón HMM debido a nueva información reconcida. Las trayectorias fragmentadas pueden iniciar su análisis en alguno de los estados ocultos de HMMs realizandose por una extensión del algoritmo. Existen una serie de desafíos que deben abordarse en la utilización de HMM para el modelado de los movimientos normales.

1. Determinar la utilización de los HMMs para modelar los diferentes grupos o patrones de trayectorias.

2. Calcular el número de estados óptimos en cada HMM.

3. Estimas los parámetros de cada HMM.

Estas etapas deben ser realizadas de manera conjunta para lograr un rendimiento razonable sobre flujos de datos complejos.

En la figura 3.27 se puede observar las diferentes etapas del aprendizaje de patrones de movimiento en una compleja escena.

La estructura de la escena y los HMMs son aprendidos automáticamente por la evolución de las trayectorias, y cada HMM corresponde a una única ruta. En la figura 3.27

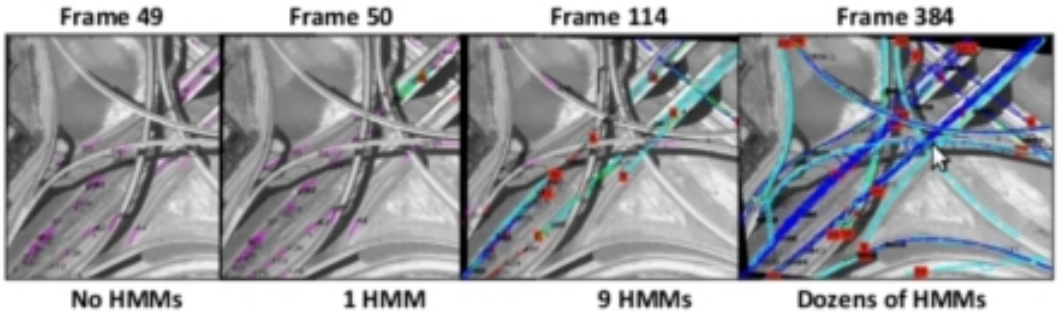

FIGURA 3.27: Evolución de la detección de trayectorias y su modelado utilizando HMM.

también se pueden apreciar situaciones anómalas dibujadas de color rojo, en el frame 114. Algunas se producen a causa de que la trayectoria que realiza no está aún reconocida porque no está los suficientemente próxima como para ser clasificada en algún patrón. Otras se producen por modificaciones temporales de la escena, otras debido al mal comportamiento de vehículos y otras debido al exceso de velocidad que describen.

En la figura 3.28 se puede observar cada HMM y cada trayectoria asignada, ya que ambas están representadas con el mismo color. Las trayectorias anómalas son representadas de color rojo a lo largo de su recorrido, aunque el algoritmo detecte fragmentos que representan acciones correctas.

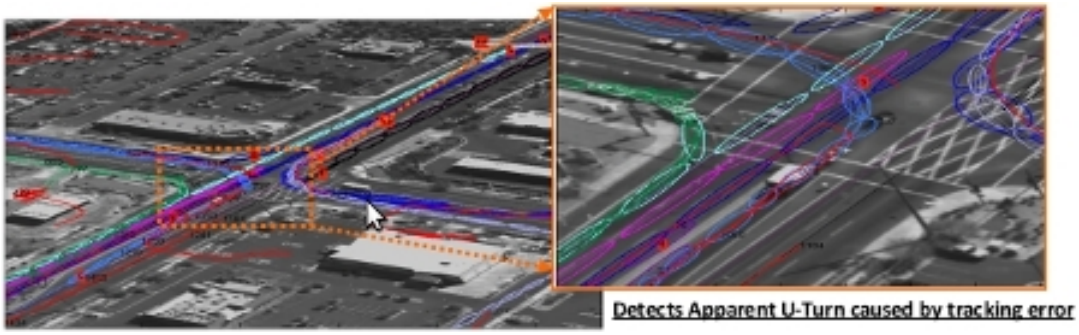

FIGURA 3.28: Imagen de intersección del vídeo donde aparecen los HMMs y los diferentes estados normales de cada trayectoria. Las trayectorias anómalas están dibujadas de color rojo.

Existen varios aspectos que del método propuesto que se pueden mejorar. Durante la fase de entrenamiento algunos HMMs tienden a solaparse variando los puntos de origen y destino de los vehículos. Esto puede mejorarse mediante la creación de un algoritmo de fusión que permita fusionar las zonas comunes de los HMM, aunque estos tenga un coste computacional elevado. También, los HMM por defecto utilizan distribuciones Gaussianas en la observación de cada una de las dimensiones, lo que provoca la aparición de errores en la transición de estados ocultos. Esto puede ser mejorado modelando las trayectorias usando

una distribución uniforme en la dirección del movimiento y calculando una distribución Gausiana perpendicular a la dirección del movimiento.

3.7.9. Análisis de trayectoria utilizando una clasificación espectral y extracción de patrones.

Actualmente los métodos más comunes de análisis de trayectorias se centran en estudiar sus características geométricas de toda trayectoria. Sin embargo, los fragmentos que constituyen las trayectorias que se repiten en un elevado número de trayectorias son a veces la parte más interesante en el análisis y en la detección de situaciones atípicas. Los autores Yuanfeng YANG, Zhiming CUI, Jian WU, Guangming ZHANG y Xuefeng XIAN proponen en [YY12] un método en el que primero las trayectorias son fragmentadas en subtrayectorias incorporando intervalos de apariencia en común (CAIS) en cada una de ellas. Proponen un método de similitud de trayectorias que incorpora la localización, el espacio y la dirección de las trayectorias. Finalmente, presentan un algoritmo de extracción de patrones, que trata de identificar las acciones frecuentes.

Planteamiento del problema.

Un vídeo está compuesto por una secuencia de frames(FS). El movimiento de los objetos a lo largo del entorno observado viene determinada por una secuencia de fotogramas con información espacio-temporal y se denomina secuencia de una trayectoria(TS). Cada FS contiene una o más trayectorias[CX09]. Las trayectorias pueden ser clasificadas o agrupadas según su posición, tiempo, velocidad y dirección. Cada trayectoria pertenece a una clase o grupo. Cuando los objetos entran o salen de la escena, los grupos o clases están cambiando, y una misma trayectoria podría pertenecer a varias clases o grupos[Kal05]. El objetivo del método de análisis de trayectorias es identificar los patrones de trayectorias realizados por el movimiento de los objetos. Cada patrón es un conjunto de subtrayectorias. Debido a que la clasificación no se realiza sobre trayectorias completas sino que se realiza de subtrayectorias, provoca que la trayectoria de un objeto perteneza a varios grupos o patrones(esto es posible siempre que la trayectoria de un objeto esté fragmentada en varias subtrayectorias).

División de la trayectoria.

Las trayectorias son dividas en el método propuesto, por dos motivos:

- Los objetos no presentan un movimiento homgéneo durante toda su trayectoria, sino que cambia de posición, tiempo, velocidad y y dirección a lo largo de su trayectoria. A la hora de realizar la similitud por trayectoria, si realizamos una comparación de una trayectoria completa el grado de similitud es más bajo que si se realiza una comparacion por fragmentos.
- Es imposible mantener el mismo movimiento del objeto; algunos objetos podrían entrar o salir, lo que es similar al concepto de clasificación de movimiento [Kal05].

Cada trayectoria es dividida por el concepto de intervalo de apariencia común (CAI). La figura 3.29 muestra un esquema del método de divisón de trayectorias utilizado.

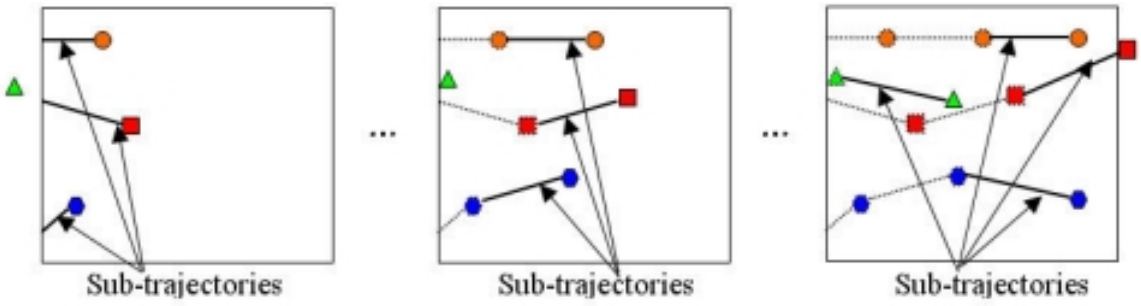

FIGURA 3.29: División de trayectorias.

En general, se genera un nuevo fragmento de trayectoria, cada vez que un nuevo objeto sale o entra en la escena observada por las cámaras.

Medida de similitud por trayectoria.

En [ZZT06] se presentan seis medidas diferentes para la clasificación en escenas observadas, y también realiza un ranking de éstas, según su correcto comportamiento y evaluación del rendimiento y coste. La distancia PCA junto con la distancia Euclídea son la medida con mejor rendimiento. Pero la información de velocidad es eliminada al reajustar el seguimiento y se pierde la variación de velocidad. La distancia LCSS puede distinguir la diferencia entre el movimiento de trayectorias que están siguiendo el mismo camino y misma orientación pero diferente velocidad. Sin embargo, el LCSS incialmente se propone para reconocer lenguaje de signos, donde lo principal no es conocer la similitud de la posición, pero sí la similitud en cuanto al contorno del objeto. La distancia Hausdorff tiene un costo más elevado y sin considerar el impacto de la dirección de la trayectoria. El método de entrenamiento de la trayectoria requiere un gran número de puntos de la trayectoria o un ajuste excesivo. En el modelo que presentan Yuanfeng YANG, Zhiming

CUI, Jian WU, Guangming ZHANG, Xuefeng XIAN se utiliza la distancia Euclídea. Las subtrayectorias clasificadas contienen una longitud variable y los resultados de la clasificación de subtrayectorias dan lugar a regiones. La medida de similitud propuesta se utiliza la distania Euclídea modificada, para obtener la distancia y la dirección entre dos posiciones.

Clasificación espectral de subtrayectorias.

Se ha utilizado el algoritmo de clasificación espectral [YY12] para mejorar el algoritmo de K-medias. Supongamos que hay un conjunto de subtrayectorias $S = \{S_1, S_2, ..., S_n\}$, las principales etapas del algoritmo de clasificación espectral para agrupar las subtrayectorias en K grupos.

- **Etapa 1:** Cálculo de la matriz de similitud $A \in R^{NxN}$ para los pares de subtrayectorias del conjunto de datos S.
- **Etapa 2:** Construcción de la matriz Laplaciana $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ donde D es la matriz diagonal cuyo elemento (i,1) es la suma de los i-ésimos números que forman A.
- **Etapa 3:** Aplicar una descomposición de los valores de L para encontrar el principal K de los vectore $x_1, x_2, ...x_k$.
- **Etapa 4:** Formar una nueva matriz $X = [x_1, x_2, ..., x_k]$ agrupando los K vectores en columnas y renormalizando cada fila de X.
- **Etapa 5:** Clasificación de filas de X dentro de los K grupos utiliando el algoritmo de $K - medias$.
- **Etapa 6:** Finalmente asignar la subtrayectoria S_i al grupo j si y solo si la fila i de la matriz se ha asignado al grupo j.

Extracción de patrones mediante el análisis de trayectorias.

El algoritmo de clasificación espectral agrupa las subtrayectorias en K grupos $\{C_1, C_2, ..., C_k\}$, a cada subtrayectoria se le asigna una etiqueta dependiendo del grupo al que pertenezca. De esta forma cada movimiento de trayectorias es transformado en una secuencia de etiquetas correspondiente a cada subtrayectoria que describe el movimiento realizado por el objeto en cada etapa. Un ejemplo se puede observar en la figura 3.30 donde TR_{01} puede ser representado por una secuencia de etiquetas $C_1C_2C_5$.

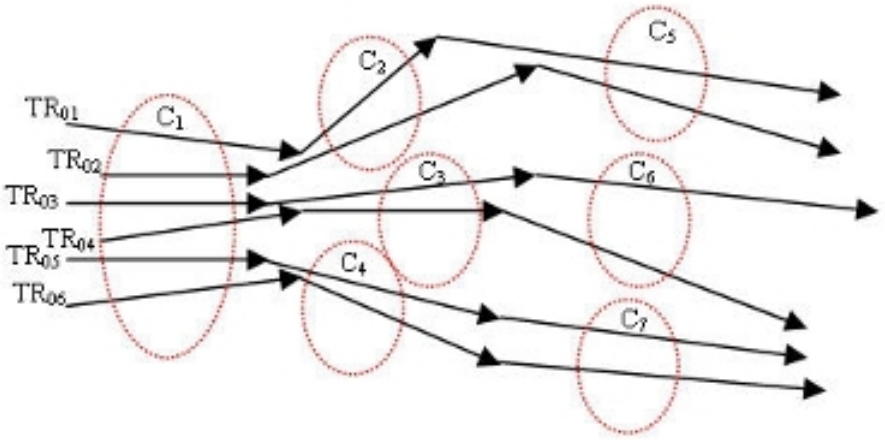

FIGURA 3.30: Clasificación de trayectorias y etiquetas.

A cada trayectoria se le asigna una secuencia de etiquetas $C_1, C_2, ..., C_{nc}$ que representan el comportamiento del movimiento del objeto. En el modelo propuesto, se identifican una serie de reglas que representan el comportamiento del movimiento de los objetos, utilizando un algoritmo de extracción de una serie de patrones.

Para la extración de una patrones se utiliza *PrefixSpan* [JP04]. Las subtrayectorias iniciales pueden estar definidas por la parte inicial de las trayectorias o grupos de trayectorias. En la etapa de adquisión de subtrayectorias iniciales, muchas de ellas se obtienen de partes comunes de los movimientos de los objetos. La parte inicial de una subtrayectoria es mucho más determinante que la parte final de la misma, porque la obtención de la parte inicial de una susbtrayectoria permite realizar una predicción del posible comportamiento del objeo, anticipándose a su movimiento.

Veamos α puede ser una secuencia de patrones que se encuentra en una secuencia de la base de datos S. La proyección de α sobre la base de datos se expresa $S|\alpha$ es la colección de las partes iniciales de las subtrayectorias de la secuencia S con respecto al prefijo α. El pseudocódigo del algoritmo propuesto se puede obtener en [YY12].

3.7.10. Análisis de la velocidad de objetos utilizando imágenes 2D

En el árticulo [JA11] se presenta un componente independiente que puede ser integrado dentro de un sistema de vídeo vigilancia denominado *OCULUS*. El objetivo de este componente es clasificar la velocidad la velocidad con la que se desplazan los objetos como normal o anómala, con el fin de detectar aquellas situaciones anómalas, teniendo en cuenta la clase de objeto y la información espacio-temporal como la localización y movimientos. El componente analiza la velocidad de los objetos detectados en tiempo real

sin necesidad de: utilizar diversas cámaras, representación 3D del entorno o una precisa estimación de valores. Este método requiere conocer previamente los parámetros de la cámara (altura, ángulo, nivel de zoom, etc.). El conocimiento que utiliza este componente es automáticamente adquirido por un algoritmo de aprendizaje que genera reglas difusas altamente interpretables. El diseño del componente requiere de la elección de un conjunto de variables y restricciones. Los valores de las variables pueden ser proporcionados directamente por los sensores distribuidos (por ejemplo, cámaras) o por los preprocesadores localizados en capas inferiores. Para realizar una correcta clasificación de la velocidad de un objeto como normal o anómala es necesaria la siguiente información, proporcionada por los preprocesadores situados en capas inferiores.

- Los objetos que aparecen en la escena monitorizada.
- Localización de cada objeto detectado.
- Clase a la que pertenece un objeto. La clase de objeto es un factor importante a la hora de determinar si una velocidad es anómala o no.
- Desplazamientos horizontales, verticales o globales, del objeto en la imagen 2D.
- Desplazamientos normales, permitidos en cada región de la imagen 2D para clase de objeto.

Se han creado un conjunto de variables para definir el concepto de velocidad organizándolas en cuatro grupos: (a) variables que determinan la localización vertical y horizontal de un objeto, (b) variables que permiten establecer relación con un objeto de cada clase, (c) variables que proporcionan información sobre el desplazamiento en la imagen 2D, y finalmente (d) variables exclusivamente utilizadas para definir las restricciones espaciales.

Además, se describen restricciones que representan la velocidad normal en el entorno monitorizado. En caso de que un objeto cumpla al menos una de las restricciones definidas inicialmente, entonces se considera que la velocidad a la que se desplaza el objeto es normal. Cada restricción contiene una lista con la clase de objetos cuyo desplazamiento a una velocidad determinada es adecuado.

En resumen, la particularización de este componente es la utilización de un algoritmo de aprendizaje automático, que genera un conjunto de reglas difusas genéricas e interpretables. Precisamente, a partir de estas reglas se realiza la clasificación del movimiento de un objeto como normal o anormal.

Este método ofrece las siguientes ventajas:

- Alto grado de interpretabilidad de los conocimientos adquiridos.
- Utilización para diferentes entornos de monitorización con diferentes tipos de objetos.
- Número reducido de restricciones para la generación del entrenamiento del sistema (sólo una restricción por cada situación posible).
- Alta escalabilidad para incluir un nuevo tipo de restricción para la ampliación del componente desarrollado.
- Capacidad de seguimiento en tiempo real.
- Bajo coste.

4

Método de trabajo

En este capítulo se determina la metodología utilizada en el desarrollo del proyecto, así como una descripción del ciclo de vida del mismo. También se especifican las tecnologías y herramientas empleadas en la implementación del sistema.

4.1. Metodología de desarrollo

El modelo del proceso software utilizado en el desarrollo de *Atravel* es un **modelo en cascada iterativo** [SOM05]. Se ha optado por este modelo ya que, considera las actividades fundamentales del ciclo de vida del software como fases separadas del proceso. Además se ha considerado realizar una modificación en el modelo del proceso software en cascada iterativo para dotar de mayor flexibilidad al modelo de casacada. Esto es que el modelo en cascada no permite comenzar con la siguiente etapa del proceso software sin antes haber finalizado la etapa o fase anterior. Para evitar esta rigidez del modelo en cascada se ha incorporado una modificación en el modelo, dotándolo de caracter iterativo. Esta característica hace más flexible y fiable el proceso de desarrollo software ya que permite realizar iteraciones en los prototipos generados para refinar el producto final. De manera que, permite realizar modificaciones en todas las etapas anteriores atendiendo a los resultados obtenidos en comparación con los resultados esperados. Para la filosofía de trabajo en la consecución del proyecto *Atravel* es importante tener en la etapa de diseño una propiedad de realimentación, ya que es necesario la creación de diferentes aproximaciones de diseño antes de conseguir un diseño definitivo. Al tratarse de un modelo iterativo,

aumenta la probabilidad de que la funcionalidad del prooducto final sea la esperada. Se puede observar un esquema del modelo desarrollado en la figura 4.1

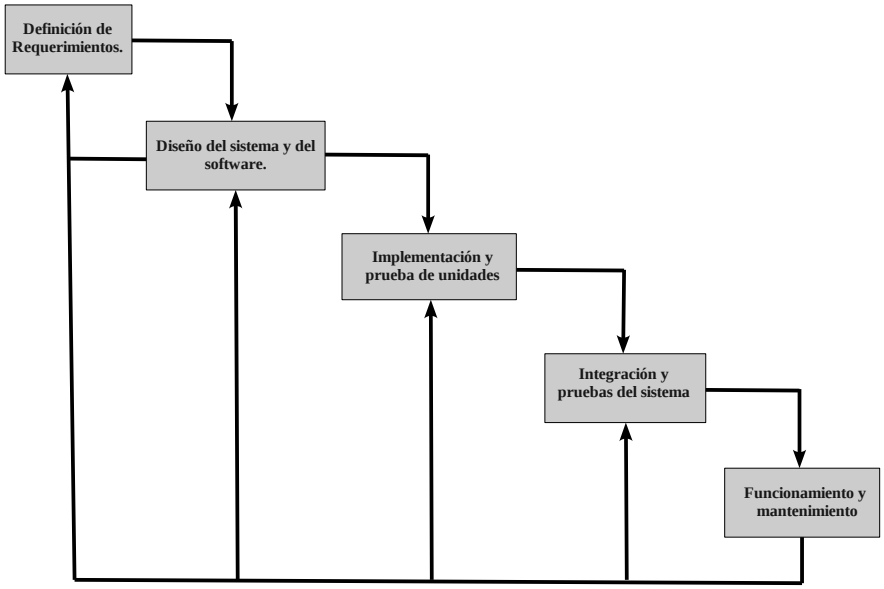

FIGURA 4.1: Modelo del proceso software utilizado.

Las etapas que componen el modelo son:

1. *Definición de requerimientos.*
2. *Diseño del sistema y del software.*
3. *Implementación y prueba de unidades.*
4. *Integración y Pruebas del sistema.*
5. *Funcionamiento y Mantenimiento.*

En la primera etapa de *Definición de requerimientos* se ha realizado un diagrama de casos de uso para especificar los requisitos funcionales necesarios para la elaboración de *Atravel* (véase la figura 4.2).

Dada la importancia de la fase de *Diseño del sistema y del software*, en la que la existencia de un error se arrastra al resto de etapas del sistema se ha realizado un diseño preliminar con la distintas funcionalidades del sistema (ver la figura 4.3).

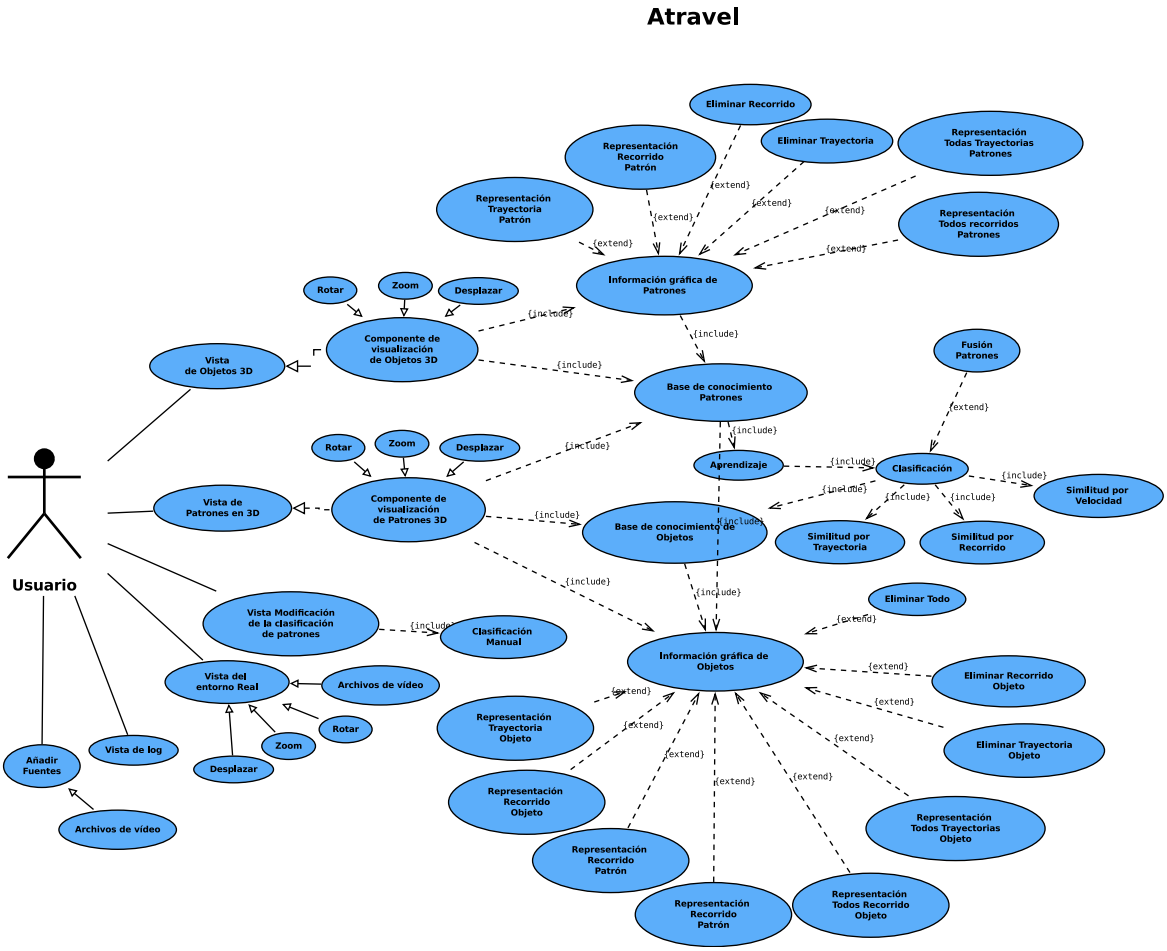

FIGURA 4.2: Diagrama de casos de uso.

Para organizar las diferentes funcionalidades del sistema y descargar responsabilidades en los diferentes componentes se ha utilizado un patrón multicapa. Ésta se caracteriza por tener 3 capas diferentes, cada una con una responsbilidad muy diferente.

Esta arquitectura nos permite de cierto modo *encapsular* la funcionalidad de cada capa, de manera que para acceder de una capa a otra sólo es posible acceder desde las capas adyacentes. Las capas que componen la arquitectura son:

- **Capa de Presentación:** Es la capa más externa de la arquitectura. Por esta capa se realiza toda la comunicación entre el usuario y la aplicación. Está compuesta fundamientalmente por las interfaces gráficas de usuario que tiene la aplicación. Se puede accener a esta capa únicamentes desde la capa de Dominio.

- **Capa de Dominio:** Es la capa que contiene toda la funcinalidad de la aplicación, es decir, toda la lógica de la aplicación, se encuentra almacenada en esta capa.

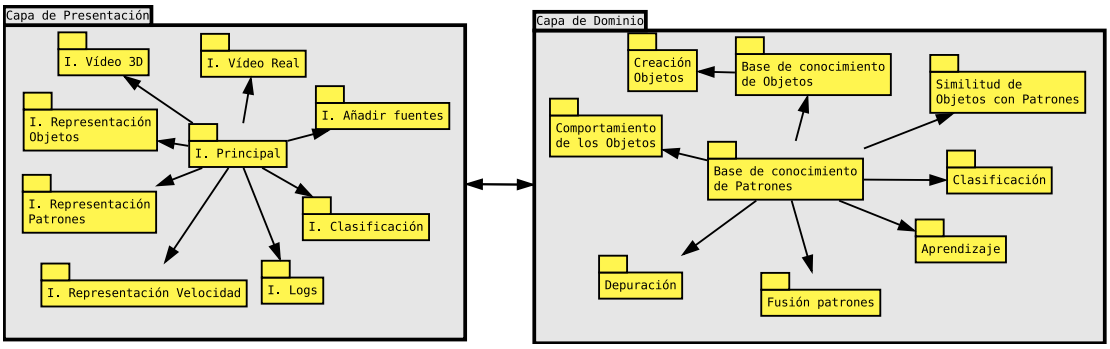

FIGURA 4.3: Diagrama de clases, diseño preliminar.

- **Capa de Persistencia:** Es la capa que almacena que proporciona volatibilidad a la aplicación, es decir, son los ficheros que almacenan el estado de la aplicación en caso de que ocurra un fallo hardware en el sistema donde se ejecuta la aplicación.

4.2. Tecnologías y herramientas utilizadas.

4.2.1. Lenguaje de programación.

El lenguaje de programación utilizado en el desarrollo de la lógica de *Atravel* ha sido **Python**[1]. La versión utilizada ha sido *python 2.6.5*. Mientras que para la implementación de las interfaces gráficas se ha utilizado **PySide** [2]. Se ha utilizado *Python* [AMV09] para el desarrollo de la lógica, ya que se trata de un lenguaje orientado a objetos y además es un lenguaje muy sencillo de utilizar y resulta muy legible. Al tratarse de un lenguaje orientado a objetos, proporciona a la aplicación encapsulación, abstracción y polimorfismo. Para el desarrollo de interfaces se ha utilizado *Pyside* ya que es un binding de Python creado para la manipulación del framework de Qt. *Pyside* permite el desarrollo rapido y ágil de interfaces de usuario.

4.2.2. Hardware.

La realización del proyecto *Atravel* ha sido llevado a cabo en una arquitectura hardware: *i386* de 64 bits y con 4GB de memoria RAM. En la consecución de las pruebas en tiempo real de *Atravel* se han utilizado dos cámaras dos cámaras webcam modelo *Logitech Sphere AF*[3] (véase la figura 4.4).

[1] www.python.org
[2] www.pyside.org
[3] http://www.logitech.com

FIGURA 4.4: Cámara Logitech Sphere AF.

4.2.3. Software.

En este apartado se especifican las bibliotecas, aplicaciones, herramientas y tecnologías utilizadas en el desarrollo de *Atravel*.

Herramientas de diseño y modelado 3D.

- **Blender 3D**: se ha utilizado esta herramienta para diseñar y modelar entornos virtuales 3D con el objetivo de realizar pruebas constantemente. Dada la complejidad del sistema es importante conocer el comportamiento del sistema al mismo tiempo que se implementan las distintas funciones, algoritmos e interfaces que los componen. La versión utilizada es la *2.49b* y está disponible en [4].

Software de desarrollo.

- **Python**: Debido a que se trata de un lenguaje interpretado, no es necesario compilar ni enlazar. *Python* incorpora un entorno de ejecución que detecta con exactitud errores en el código. De manera, que la depuración del código se realiza en el mismo entorno de programación. Además cuenta con entorno interactivo que permite comprobar en cualquier instante la funcionalidad de estructuras o pequeñas funciones programadas. La versión de Python utilizada es *Python 2.6.5*. Se puede obtener en `www.python.org`.

- **Kile**: Esta aplicación se ha utilizado para llevar a acabo el desarrollo de la documentación de *Atravel*. La versión que se ha utilizado es *2.0.85* [5]. *Kile* hace uso de la plataforma de desarrollo *KDE* versión *KDE 4.4.5*.

[4] http://download.blender.org/release/
[5] http://packages.ubuntu.com/lucid/i386/kile/download

- **Pyside**: Se ha utilizado como herramienta de diseño de interfaces gráficas de *Atravel*. Se puede instalar en `http://qt-project.org/wiki/PySide_Binaries_Linux/`.

Sistema Operativo.

- **Ubuntu**: Se ha utilizado este sistema operativo porque es un sistema muy completo y de libre uso. Concretamente se ha utilizado la versión *10.04 LTS - Lucyd Linx*.

Control de Versiones.

- **Google Proyect Hosting**: Es un servicio de alojamiento web de código abierto, rápido, sencillo y fiable creado por *Google Code*. Para la gestión y el control de versiones de *Atravel* se ha utilizado *RabbitVCS SVN*.

Documentación

- **Doxygen**: Se ha utilizado para generar la documentación del código fuente de *TraceMon*. La versión que se ha utilizado es *1.7.1* [6].
- **LaTeX**: Herramienta[7] que se ha utilizado para la elaboración de la memoria de este proyecto.
- **LibreOffice Draw**: Aplicación que se ha utilizado para generar las imágenes vectoriales de este documento. La versión que se ha utilizado es *3.3.2*[8].
- **Gimp**: Herramienta[9] que se ha utilizado para generar las imágenes no vectoriales de este documento.
- **Dia**: Herramienta[10] que se ha utilizado para la creación de diagramas UML. La versión que se ha utilizado es *0.97.1*.

Bibliotecas

- **OpenCV**: Esta biblioteca se ha utilizado para el tratamiento de las imágenes y gestión de vídeos en la interfaz gráfica. La versión que se ha utilizado es *2.0.1*[11].

[6] http://www.icewalkers.com/Linux/Software/57870/Doxygen.html
[7] https://help.ubuntu.com/community/LaTeX
[8] http://www.ubuntu-es.org/node/143190
[9] http://packages.ubuntu.com/maverick/gimp
[10] http://packages.ubuntu.com/maverick/dia
[11] http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.1/OpenCV-2.1.0.tar.bz2/download

Esta biblioteca ha sido instalada sobre *Ubuntu 10.04* siguiendo el tutorial de `http://www.samontab.com/web/2010/04/installing-opencv-2-1-in-ubuntu/`.

- **PyOpenGL**: Se ha utilizado para la representación en 3D de los objetos y las trayectorias detectadas en la interfaz de usuario. Se ha utilizado la versión *PyOpengl 3.0.0* y se puede obtener `http://pypi.python.org/pypi/PyOpenGL`. Esta biblioteca incorpora las bibliotecas GLU y GLUT.

- **python-qt4**: Biblioteca utilizada para establecer la unión entre python y Qt.

- **FTGL**: Biblioteca utilizada para la importación de fuentes al componente OpenGl utilizado.

- **Pyside**: Se ha utilizado esta biblioteca para la realización de las interfaces gráficas de usuario de *Atravel*. La versión utilizada es *Pyside 1.1.1* y se puede obtener en `www.pyside.org`.

5

Arquitectura de Atravel

En este capítulo se realiza una descripción de la arquitectura de *Atravel*, para ello, se ha utilizado un *diseño descendente* o *Top-Down*. Una vez que se ha especificado la funcionalidad general de la aplicación, se despliega dicha funcionalidad en distintos módulos o componentes, de manera que cada componente desarrolla una funcionalidad específica de la aplicación.

En la figura 4.2 se puede observar la funcionalidad de *Atravel* representada en un diagrama de casos de uso. Teniendo en cuenta la funcionalidad de la aplicación y la utilización de un patrón multicapa, la arquitectura de *atravel* se puede ver en la figura 5.1.

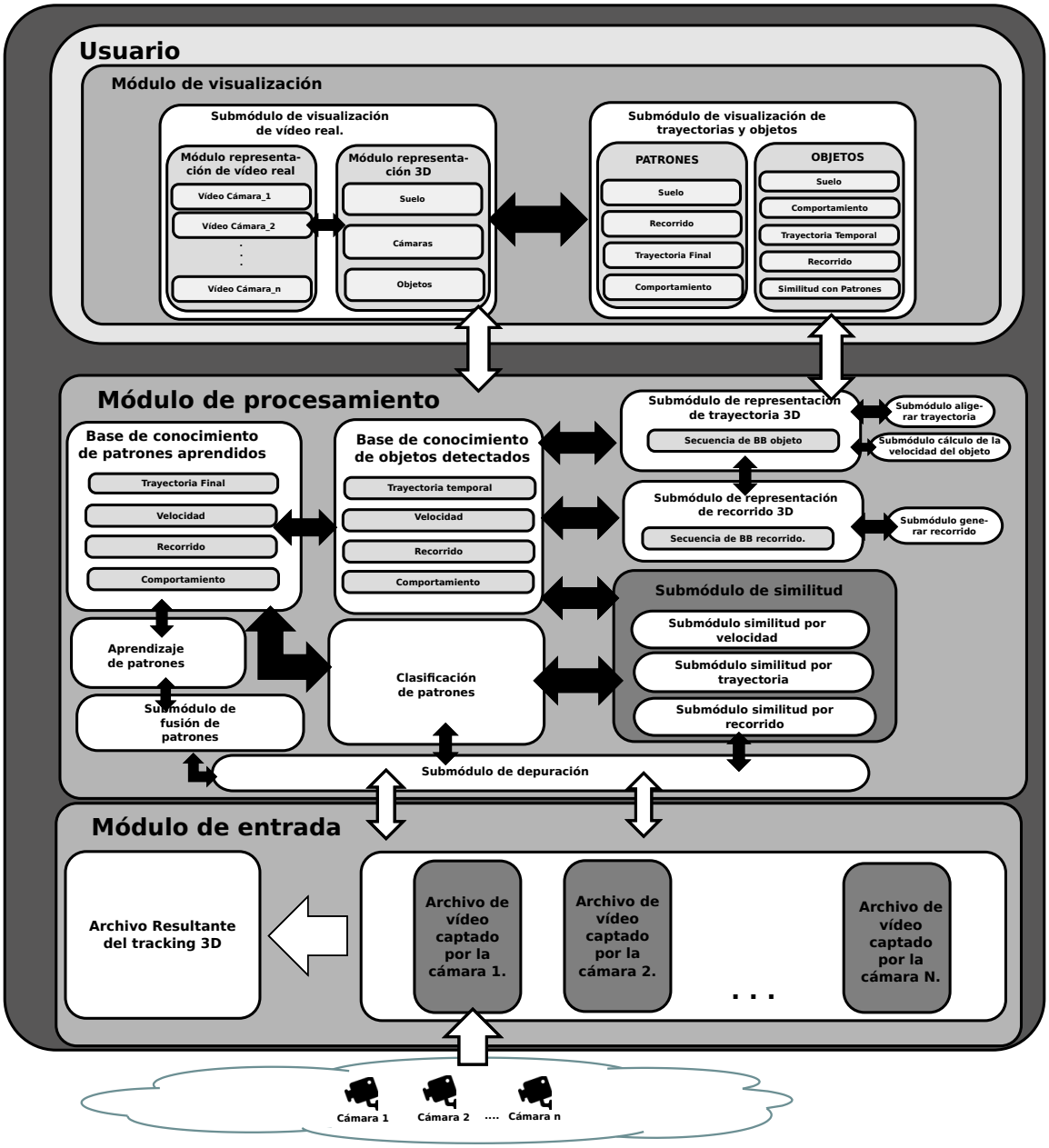

FIGURA 5.1: Arquitectura general de *Atravel*.

La arquitectura está compuesta de cuatro módulos:

- **Módulo de entrada**: está formado por toda la información externa necesaria para el correcto funcionamiento de la aplicación. El usuario debe incorporar un archivo con los resultados obtenidos tras la etapa de tracking en 3D. También debe especificar el número de dispositivos de vídeo utilizados junto con los archivos de flujo de vídeo obtenidos en la monitorización. Este componente únicamente está en contacto con

el módulo de procesamiento, de manera que todos los datos de entrada al sistema necesitan ser procesados por la aplicación.

- **Módulo de procesamiento**: está compuesto por aquéllos submódulos encargados de llevar a cabo una funcionalidad concreta de la aplicación. Es el módulo donde tiene lugar toda actividad encargada del procesamiento de información necesaria para el funcionamiento de *Atravel*. En este módulo se realizan funcionalidades entre las que destacan, la representación de trayectorias y recorridos de los objetos reconocidos en 3D, creación de patrones, similitud de trayectorias y recorridos, análisis de la velocidad de los objetos, fusión de patrones, aprendizaje de patrones y clasificación de comportamientos de objetos en normal o anómalo. Este módulo está en contacto módulo de entrada que es el módulo que recibe los datos y con el módulo de visualización, con el que se envía la información solicitada por el usuario.

- **Módulo de visualización**: Es el módulo que permite interactuar con el usuario y la funcionalidad de la aplicación. A través de éste, un usuario realiza una serie de peticiones a la aplicación y es este módulo el encargado de comunicarse con el módulo de procesamiento, para dar las repuestas a las peticiones y que los usuarios puedan visualizarlas. Entre las funciones que se pueden realizar desde este módulo se encuentran: mostrar flujo de vídeo real de la escena monitorizada, representación en 3D de trayectoria y recorrido de patrones, representación en 3D de objetos en tiempo real, representación en 3D de trayectoria y recorrido de objetos detectados, visualización de porcentaje de similitud de un objeto con los diferentes patrones reconocidos (trayectoria, recorrido y velocidad) y visualización del comportamiento de patrones y objetos.

5.1. Módulo de entrada

Es el encargado de recibir por parte del usuario toda la información necesaria para el correcto funcionamiento de la aplicación. En este módulo podemos distinguir tres submódulos diferentes:

- Archivo resultante del *Tracking 3D*.
- Archivo de importación de una base de conocimiento de patrones.
- Archivos de flujo de vídeo de la escena monitorizada, captada por los distintas cámaras que monitorizan la escena.

El archivo resultante de la etapa de *Tracking 3D* debe corresponder con el *tracking* realizado sobre los diferentes objetos que aparecen en los distintos archivos de flujo de vídeo, captados por cada una de las cámaras colocadas en la escena a monitorizar. El módulo de entrada es el encargado de proporcionar los datos necesarios al módulo de procesamiento para su correcta ejecución.

5.1.1. Submódulo de archivo resultante del Tracking 3D

En este submódulo el usuario necesita incorporar un archivo que contenga los resultados obtenidos tras el procesamiento de los flujos de de vídeo proporcionados por los dispositivos de vídeo. Este archivo debe ser un simple archivo de texto con una estructura específica previamente definida. Para entender la estructura del archivo es necesario realizar una breve descripción del proceso de monitorización anterior al análisis del comportamiento de una escena.

El procedimiento es el siguiente, una vez que se han colocado las cámaras en el entorno a monitorizar, se procede a detectar los objetos en movimiento que tienen lugar en la escena observada. Una vez que se ha detectado la presencia de objetos en la escena, se procede a realizar un seguimiento, el cuál consiste en almacenar la información relevante del objeto en cada instante de tiempo. Una vez que ha realizado el *Tracking* se pasa a realizar un análisis del comportamiento del movimiento de los objetos detectados en el entorno monitorizado. *Atravel* a grandes rasgos se encarga de analizar el comportamiento de los objetos detectados en el entorno. Por tanto, en la etapa de *Tracking* se debe almacenar en un archivo de texto las propiedades relevantes de los objetos en cada instante de tiempo para su posterior análisis en la siguiente etapa. Es importante especificar que en el proceso de detección de objetos, cada objeto detectado es representado mediante una *caja* 3D que se ajusta al contorno del objeto. Como se muestra en la figura 5.2.

La caja está compuesta por ocho vértices que engloban al objeto detectado. Por tanto, el archivo de texto que contiene los resultados del *Tracking* debe contener toda la información relevante del objeto, para su posterior análisis. De manera que el archivo de texto viene determinado por los siguientes campos: *Tipo de objeto (cadena o string), Identificador (cadena o string),* $vertice0_x$ *(float),* $vertice0_y$ *(float),* $vertice0_z$ *(float),* $vertice1_x$ *(float),* $vertice1_y$ *(float),* $vertice1_z$ *(float),* $vertice2_x$ *(float),* $vertice2_y$ *(float),* $vertice2_z$ *(float),* $vertice3_x$ *(float),* $vertice3_y$ *(float),* $vertice3_z$ *(float),* $vertice4_x$ *(float),* $vertice4_y$ *(float),* $vertice4_z$ *(float),* $vertice5_x$ *(float),* $vertice5_y$ *(float),* $vertice5_z$ *(float),* $vertice6_x$ *(float),* $vertice6_y$ *(float),* $vertice6_z$ *(float),* $vertice7_x$ *(float),* $vertice7_y$ *(float),* $vertice7_z$ *(float), Frame (entero), Hora (cadena o string), Fecha (cadena o string).*

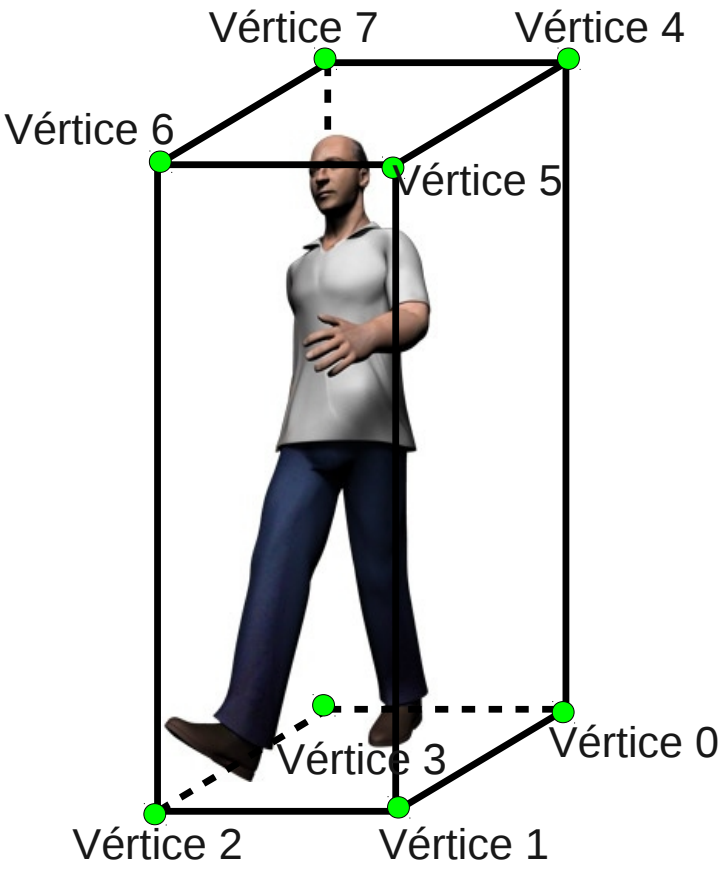

FIGURA 5.2: Representación del contorno de un objeto, utilizando una caja ”3D“.

Donde los campos tienen el siguiente significado:

- *Tipo de objeto*: Hace referencia a la tipología del objeto detectado, en este caso puede ser peatón o vehículo.
- *Identificador*: Permite identificar el objeto a lo largo de la escena, se trata de una cadena de caracteres única.
- *Coordenadas de los vértices*: Contienen el valor de tipo *float* de las coordenadas en cada uno de los ejes.
- *Frame*: Contiene el número de fotograma o imagen en la cual se ha detectado la presencia del objeto correspondiente. Viene determinado por un número entero.

- *Hora*: Hace referencia a la hora, minutos y segundos exactos en los que se detectado el objeto en la escena. El formato viene dado por tres pares de dígitos enteros separados por el carácter *dos puntos* (:). Por ejemplo 12:40:55 la primera dupla de dígitos hace referencia a las horas(en este caso 12), la segunda dupla de dígitos hace referencia a los minutos(en este caso 40) y la tercera dupla de dígitos hace referencia a los minutos (en este caso 55).

- *Fecha*: Contiene el día, el mes y el año en el que se ha detectado el objeto en la escena. El formato de la fecha viene determinado por dos parejas de números y un cuarteto de números enteros separados por el carácter *barra* (/). Un ejemplo de fecha sería 25/07/2012 , los dos dígitos primeros hacen referencia al día(en este caso el día 25), la segunda pareja de dígitos hace referencia al mes (en este caso el séptimo mes que es Julio) y el cuarteto de números especifica el año (en este caso tuvo lugar en el año 2012).

El archivo de texto resultante en la etapa de tracking contendrá todos los datos de cada objeto en el instante detectado haciendo referencia a los campos anteriormente definidos. De manera que cada línea del archivo tendrá este formato:

```
1  <tipo>,<identificador>,<vertice_0_x>,<vertice_0_y>,<vertice_0_z>,<vertice_1_x>,<
       vertice_1_y>,<vertice_1_z>,<vertice_2_x>,<vertice_2_y>,<vertice_2_z>,<vertice_3_x>,<
       vertice_3_y>,<vertice_3_z>,<vertice_4_x>,<vertice_4_y>,<vertice_4_z>,<vertice_5_x>,<
       vertice_5_y>,<vertice_5_z>,<vertice_6_x>,<vertice_6_y>,<vertice_6_z>,<vertice_7_x>,<
       vertice_7_y>,<vertice_7_z>,<frame>,<hora>,<fecha>
```

Por tanto, el archivo está formado por una serie de líneas con el formato descrito anteriormente, donde cada línea contiene las propiedades de un objeto en el fotograma correspondiente. De manera que los objetos detectados están ordenados por fotograma o frame, así los objetos detectados en el primer frame, se encuentran almacenados en las primeras líneas del archivo de texto resultante de la etapa de *tracking*.

Una vez recibido el archivo resultante de la etapa de *Tracking*, el módulo de entrada lo envía al módulo de procesamiento para que éste se encargue de procesarlo.

5.1.2. Submódulo que contiene los archivos de flujo de vídeo de la escena observada.

Este submódulo permite la reproducción de los vídeos reales correspondientes al archivo resultante del *Tracking* realizado. Cada archivo de vídeo se corresponde con la visualización del entorno monitorizado desde diferente perspectiva, lo que proporciona la visualización de los objetos en la escena desde diferentes perspectivas.

El número de archivos de flujo de vídeo debe ser especificado por el usuario previamente. Aunque la aplicación está diseñada para utilizar un número indefinido de archivos de vídeo, ya que utiliza estructuras dinámicas, se ha restringido el número de cámaras a utilizar a tres. Es decir, el usuario sólo puede realizar la monitorización de una escena utilizando, una, dos o tres cámaras diferentes. El principal problema que surge al diseñar una aplicación que permita la visualización de un número ilimitado de cámaras es que depende de la resolución de pantalla del monitor en que se desee realizar la visualización de las imágenes captadas por todas las cámaras con un mismo golpe de vista.

En la tarea de cargar los archivos de vídeo y su visualización se ha utilizado *OpenCV*. Para cargar los archivos de vídeo en la aplicación se ha hecho uso del objeto *CvCapture*. Este objeto puede ser inicializado de diversos modos, dependiendo del tipo de fuente utilizada. En este caso, la inicialización del objeto *CvCapture* se ha llevado a cabo de la siguiente manera:

```
1   self.capture = cvCreateFileCapture("ruta del archivo de video")
```

Los archivos de vídeo, que el usuario puede cargar en la aplicación pueden estar en cualquier formato, *OpenCV* se encarga de su reproducción.

Además, la aplicación está preparada para recibir información de flujo de vídeo Online si fuese necesario.

5.2. Módulo de procesamiento

En este módulo tiene lugar toda la actividad de procesamiento de datos llevada a cabo por la aplicación. Se trata del módulo más importante de la arquitectura de *Atravel*, ya que es en éste donde se llevan a cabo la gran mayoría de las funcionalidades que la aplicación proporciona.

Una vez que el usuario ha proporcionado la información correspondiente al módulo de entrada, cada módulo realiza el procesamiento de una tarea específica. A grandes rasgos el funcionamiento de *Atravel* es el siguiente: el sistema obtiene los objetos detectados en la escena monitorizada en cada instante de tiempo, esto se obtiene a través del archivo de tracking 3D que forma parte del módulo de entrada (sección 5.1), se almacenan los datos más relevantes del objeto en la base de conocimiento de los objetos detectados, a continuación se llama al submódulo de representación de trayectorias 3D. Éste se encarga de crear la trayectoria del objeto, utilizando los datos anteriores (si existen). Este submódulo hace utilización de otro submódulo denominado submódulo de optimización

de la trayectoria que se encarga de detectar si la posición del objeto es lo suficientemente discriminatoria como para formar parte de la trayectoria del objeto, eliminando aquellas detecciones de los objetos cuya posición no supone una influencia lo suficientemente importante en la creación de la trayectoria, como para ser representada. El submódulo de optimización de las trayectorias tiene como función principal generar trayectorias de la forma más eficiente y eficaz posible, utilizando el menor número de recursos. Después de generar la trayectoria de cada objeto se procede a determinar el recorrido realizado por el objeto detectado. Éste tiene lugar en el submódulo de representación del recorrido 3D, el cuál hace uso del submódulo generar recorrido, el cuál realiza un recorrido parcial de la trayectoria total del objeto. Posteriormente el submódulo de representación de recorrido 3D genera el recorrido realizado por una trayectoria como una secuencia de éstos.

Una vez que se ha detectado el movimiento de un objeto, éste es almacenado en la *base de conocimiento de objetos detectados*. En ésta se almacena:

- **Trayectoria temporal**: representa el movimiento del objeto a lo largo de la escena observada hasta un instante de tiempo determinado, puede que sea su trayectoria parcial o total.
- **Velocidad**: permite conocer la velocidad con que el objeto se desplaza a lo largo el entorno observado.
- **Recorrido**: representa la zona del espacio de la escena, donde ha tenido lugar el movimiento del objeto.
- **Comportamiento**: etiqueta que permite clasificar el movimiento del objeto a lo largo del entorno observado.
- **Similitud con patrones**: el movimiento del objeto detectado realiza una comparación con cada uno de los patrones almacenados en la base de conocimiento y almacena el porcentaje de similitud.

Al mismo tiempo en que se realiza la generación de la trayectoria y el recorrido del movimiento del objeto a lo largo de la escena observada, se realiza una comparación con los patrones reconocidos y almacenados en la base de conocimiento de patrones. La comparación se realiza en tres niveles diferentes:

- **A nivel de trayectoria**: tiene lugar en el *submódulo de similitud por trayectoria* y como su propio nombre indica realiza una comparativa con la trayectoria definida en cada patrón de la base de conocimiento.

- **A nivel de recorrido**: tiene lugar en el *submódulo de similitud por recorrido*, en éste se realiza una comparación del recorrido del objeto detectado con el recorrido de cada patrón perteneciente a la base de conocimiento.
- **A nivel de velocidad**: se realiza en el *submódulo de similitud por velocidad*, y se reliza un análisis de la velocidad del movimiento del nuevo objeto detectado en la escena con respecto a la velocidad correspondiente con cada uno de los patrones existentes en la base de conocimiento.

Una vez que ha sido reconocido el movimiento de un objeto a lo largo de la escena se pasa a clasificar el comportamiento del movimiento, realizado, de manera que éste puede quedar catalogado en una de las tres clases posibles: *No catalogada*, *Normal*, *Anormal*. A continuación, el comportamiento del objeto detectado pasa a formar parte de la base de conocimiento, con la generación de un nuevo patrón.

La *base de conocimiento de patrones aprendidos* contiene la siguiente información:

- **Trayectoria final**: Almacena la información correspondiente al movimiento del objeto, una vez que ha salido de la escena observada.
- **Velocidad**: permite conocer la velocidad con que el objeto se desplaza a lo largo el entorno observado.
- **Recorrido**: representa la franja espacial donde ha tenido lugar el movimiento del objeto en el entorno observado.
- **Comportamiento**: etiqueta que permite clasificar el movimiento del objeto a lo largo del entorno observado
- **Similitud**: el movimiento del objeto detectado realiza una comparación con cada uno de los patrones almacenados en la base de conocimiento y almacena el porcentaje de similitud.

Posteriormente, se activa el *submódulo de fusión de patrones* para reducir al máximo el número de patrones y que es sistema permita realizar un aprendizaje, perfeccionando los patrones obtenidos al cabo de un largo período. Este submódulo es muy importante para la correcta clasificación de movimientos de objetos.

Para introducir al lector en el proceso de aprendizaje de patrones, se ha realizado el siguiente diagrama de flujo que lo representa (véase la figura 5.3):

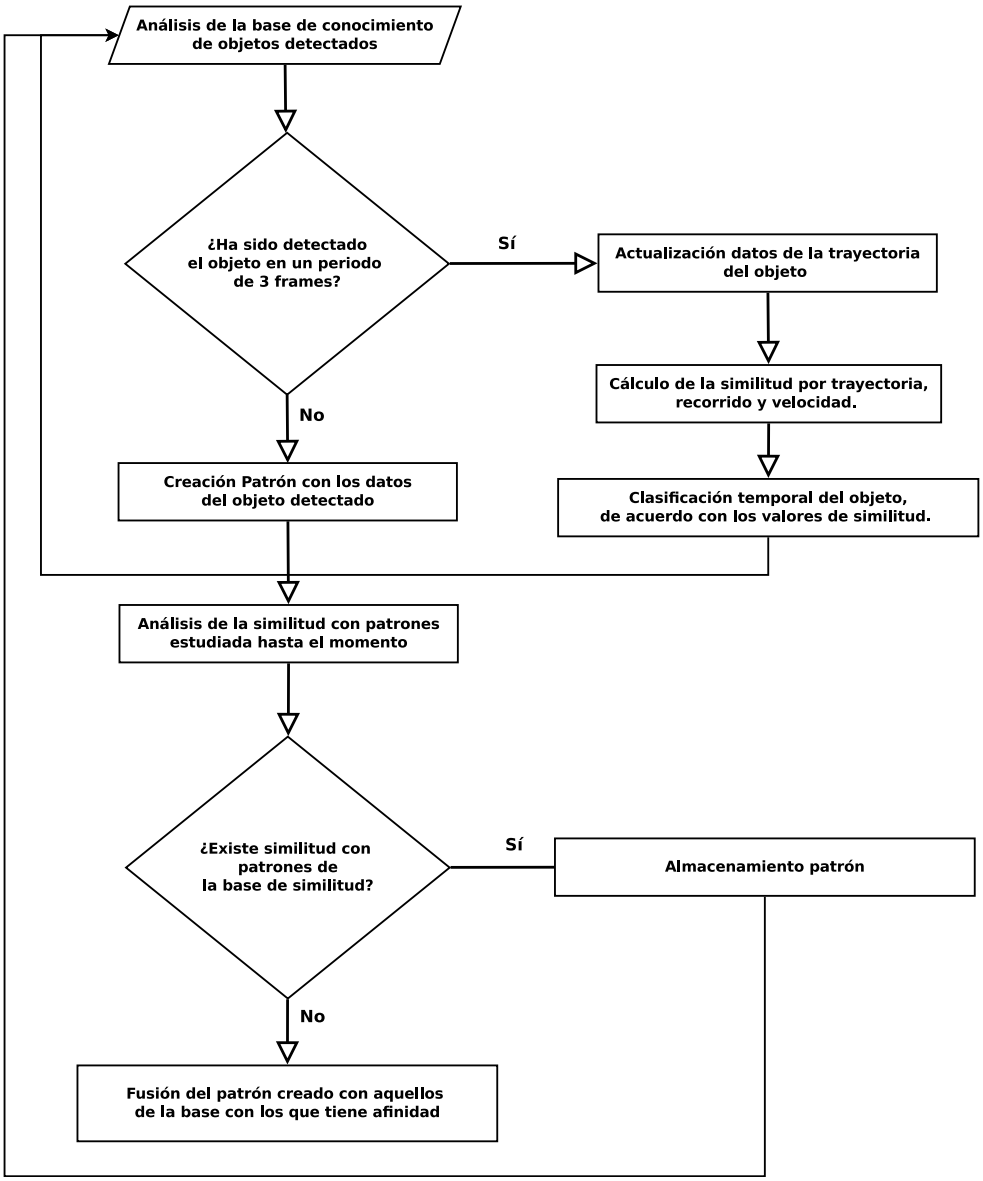

FIGURA 5.3: Diagrama de flujo correspondiente al aprendizaje de patrones.

5.2.1. Explicación de estructuras básicas.

Estructura Coordinate

El objeto *Coordinate* está compuesto por tres atributos que contienen el valor correspondiente a los tres ejes de coordenadas por separado. Sus atributos son:

- **X**: Contiene el valor correspondiente al eje de coordenas X. Es de tipo *float* debido a la importancia de precisión en los valores.

- **Y**: Contiene el valor correspondiente al eje de coordenas Y. Es de tipo *float* ya que es importante obtener suficiente precisión en los cálculos.
- **Z**: Contiene el valor correspondiente al eje de coordenas Z. Es de tipo *float* al igual y por el mismo motivo que los valores en los ejes anteriores.

Algunas de las funciones destacables de esta estructura son:

1. **Calcular el ángulo entre dos vectores**: Permite calcular el ángulo que forman dos vectores en el espacio 3D. El pseudocódigo es el siguiente:

Algoritmo 1 Calcular ángulo existente entre dos vectores.

Entrada: Dados dos vectores en el espacio 3D de la forma $V = (V_x, V_y, V_z)$ y $U = (U_x, U_y, U_z)$.

Salida: Devuelve un *Float* que expresa el número de grados que existe entre los dos vectores.

$Value = ((V_x \cdot U_x) + (V_y \cdot U_y) + (V_z \cdot U_z))/(\sqrt{{V_x}^2 + {V_y}^2 + {V_z}^2} \cdot \sqrt{{U_x}^2 + {U_y}^2 + {U_z}^2})$

$nguloenradianes = \arccos Value$

$nguloengrados = math.degrees(nguloenradianes)$.

devolver *nguloengrados*.

En esta función se realiza el producto de los dos vectores, componente a componente y se calcula la diferencia entre el producto del módulo de ambos vectores. Posteriormente se calcula el ángulo medido en radianes realizando la operación de *Arcoseno* de un ángulo. Por último, se pasa el ángulo medido en radianes a grados y es devuelto el valor. El cálculo de este valor, está basado en los conceptos explicados en la sección 3.2.3.

2. **Calcular si dos vectores unitarios se encuentran en la misma dirección**: En esta función se determinada si dos vectores unitarios se encuentran en una dirección similar o no. Para ello existe una constante o **umbral** cuyo valor permite establecer el límite para establecer si dos vectores diferentes se encuentran en la misma dirección.

   Para calcular la dirección del vector se calcula la diferencia en valor absoluto entre los valores de los tres ejes de coordenadas. Si la suma de todos los valores es superior al umbral previamente establecido, entonces se dice que los dos vectores no se encuentran en una dirección similar, por lo que se encuentran en distinta dirección.

3. **Calcular la distancia euclídea entre dos puntos**: Esta función permite calcular la distancia entre dos puntos situados en el espacio euclídeo. El pseudocódigo es el siguiente:

Algoritmo 2 Calcular similitud entre dos vectores.

Entrada: Dados dos vectores unitarios en el espacio 3D de la forma $V = (V_x, V_y, V_z)$ y $U = (U_x, U_y, U_z)$.

Salida: Devuelve un **cierto** si los dos vectores tienen una dirección similar y **falso** si éstos tienen una dirección muy diferente.

Se establece un umbral de diferencia entre vectores.

si $(\|U_x - V_x\| + \|U_y - V_y\| + \|U_z - V_z\|) > umbral$ **entonces**
  devolver falso
si no
  devolver cierto
fin si

Algoritmo 3 Calcular la distancia entre dos puntos.

Entrada: Dados dos puntos espacio 3D de la forma $P = (P_x, P_y, P_z)$ y $Q = (Q_x, Q_y, Q_z)$.

Salida: Devuelve un float con el valor de la distancia.

$Distancia = \sqrt{(P_x - Q_x)^2 + (P_y - Q_y)^2 + (P_z - Q_z)^2}$

devolver *Distancia*

Base de conocimiento de objetos detectados.

En esta sección se va a explicar las principales características que tiene este submódulo. Se trata de una de las partes más importantes para el correcto funcionamiento de *Atravel*. La principal función de este submódulo es la de **almacenamiento de información**. Su diseño es fundamental para proporcionar al sistema la agilidad, eficiencia, rapidez y eficacia necesarias para un comportamiento óptimo de la aplicación.

A grandes rasgos puede resumirse, como el lugar por el cual se almacena toda la información necesaria de todos los objetos detectados en la escena monitorizada, en cualquier instante de tiempo. Por tanto, estamos ante uno de los submódulos más importantes.

Inicialmente se detectan la aparición de los diferentes objetos que tienen lugar en el entorno monitorizado. Éstos son detectados y leídos del archivo de texto resultante de la etapa de *Tracking*. A continuación, la aplicación identifica el objeto detectado, en caso de que el objeto no haya sido reconocido con anterioridad, es creado y almacenado en la base de conocimiento de objetos detectados. Si el objeto ha sido creado anteriormente se realiza una búsqueda de éste, para actualizar su información, incluyendo la información recién detectada. Dada la necesidad de almacenar los datos recogidos en la escena observada, su almacenamiento debe ser lo más eficiente, manejable y sencillo posible. Para cumplir estos requisitos se ha desarrollado una base de conocimiento de objetos detectados, que almacena

toda la información de los objetos detectados para su posterior análisis y generación de comparaciones necesarias para el desarrollo de la aplicación.

Para el almacenamiento de información utilizando el lenguaje de programación de *Python*, se ha utilizado la estructura de datos de ***diccionario*** [Lut09]. Esta estructura se caracteriza por almacenar datos de la forma <*Clave*><*Valor*>. Las razones por las que se ha decidido utilizar esta estructura de datos son:

- **Flexibilidad**: Los diccionario son junto con las listas las estructuras de datos más flexibles que Python integradas en *Python*.
- **Facilidad de búsqueda**: Al ser un tipo de estructura integrada en el lenguaje, lo que facilita la búsqueda de items del diccionario. Esta característica evita la construcción de algoritmos de búsqueda y estructuras de datos implementadas manualmente en lenguajes de bajo nivel.
- **Utilización de Claves**: Permite la recuperación de elementos del diccionario a través de su clave. Lo cual resulta muy beneficioso para la recuperación de un elemento del diccionario, ya que no es necesario recorrer todos los elementos que lo componen.
- **Objetos desordenados**: *Python* almacena aleatoriamente los elementos de izquierda a derecha, para proporcionar mecanismos de búsqueda rápida. Las claves proporcionan una posición simbólica (no física) de los elementos del diccionario.
- **Longitud variable, heterogéneos y arbitrariamente anidables**: Los diccionarios pueden crecer y ser comprimidos sin necesidad de realizar copias nuevas. Pueden contener objetos de cualquier tipo y pueden ser anidables hasta cualquier tipo de profundidad. Además se pueden añadir cualquier tipo de objetos como listas, diccionarios y otros objetos.

Las operaciones y consultas con diccionarios que se han realizado se pueden ver en [Lut09].

De forma que el almacenamiento de los objetos detectados en la escena monitorizada se realiza en un diccionario. Cada entrada o elemento del diccionario tiene una *clave* que debe ser única y un *valor*. La *clave* del diccionario corresponde al identificador del objeto y en este caso se trata de una cadena de caracteres (*string*) de tamaño variable. El *valor* está formado por un objeto de tipo *Trajectory*.

El tipo de objeto *Trajectory* contiene información acerca de los objetos detectados en la escena monitorizada. La información que contienen es la siguiente:

- **name_object**: Es de tipo *string* o cadena. Contiene el identificador del objeto, éste debe ser único. Su tamaño es variable.

- **type_object**: Este campo alberga el tipo de objeto que ha sido detectado en la escena. Es de tipo *string*. Sólo puede tomar dos valores, **VEHICLE** o **PEDESTRIAN**. No puede tomar otro valor en este campo. EL motivo es que un objeto detectado sólo puede ser de tipo vehículo, en tal caso en este campo se almacenaría la cadena de caracteres *VEHICLE* , o bien de tipo peatón, en cual caso se almacena la cadena de caracteres reservada *PEDESTRIAN*. Esto es así, ya que por definición *Atravel* sólo está diseñado para estudiar el comportamiento de vehículos y peatones.

- **trajectory_vector**: Este es un atributo muy importante, ya que contiene la información espacial de la trayectoria realizada por el movimiento del objeto a lo largo de la escena monitorizada. Es de tipo **lista** sección 3.1.1. Esta lista está compuesta de una serie de objetos cuyo orden es importante para la representación espacial en tres dimensiones de la trayectoria. Debido a la estructura de representación de trayectorias en 3D ha sido el *bounding box o (BB)*. El campo trajectory_vector es una lista de objetos de tipo **Features_object**. El motivo por el cual se ha utilizado un objeto de tipo *Features_object* es porque además de la información que nos ofrece el conjunto de BB es importante almacenar otras características como el número de frame, el tiempo en el que ha tenido lugar y el nombre del objeto el cual ha sido detectado y la velocidad con la que se desplaza en cada momento a lo largo de la escena. El objeto *Features_object*, resuelve perfectamente el problema de almacenar en cada instante de tiempo, el espacio recorrido por el objeto, la velocidad y el resto de características que van a permitir el correcto funcionamiento de la aplicación en posteriores análisis.

- **current_unitary_vector**: Este atributo ha sido incorporado, dada la necesidad de calcular la dirección del movimiento del objeto que se desplaza sobre la escena, en un instante determinado. Es importante recalcar que se trata de un vector normalizado, cuyo módulo es igual a uno (véase la sección 3.2.3). Este atributo es de tipo **Coordinate** 5.2.1, este objeto ha sido creado, pare representar magnitudes en 3D.

- **travel_list**: Se trata de una lista de objetos de tipo **Features_travel_object** 5.2.2. En este campo se almacena la información referente a la representación del valor espacial del recorrido que realiza el objeto sobre la escena. Es decir, la zona por la que el objeto se desplaza en el entorno observado (ver sección 5.2.4). La representación del recorrido viene fundamentalmente definida por una serie de bounding box que representan el desplazamiento espacial del objeto a lo largo de la escena. La forma de calcular de obtener esta lista se encuentra detallada en 5.2.4.

- **similarity_structure**: Este atributo ha sido creado debido a la necesidad de almacenar los datos referentes a la similitud de patrones donde hay que diferenciar entre diversos tipos de similitud:

  - *Similitud por trayectoria*: permite conocer el grado de similitud entre la trayectoria recién reconocida y los patrones existentes en la base de conocimiento de patrones detectados. Para ello se utilizan diversos métodos que se detallan posteriormente.
  - *Similitud por recorrido*: esta similitud permite reconocer el grado de similitud existente entre el recorrido de un objeto recientemente detectado en el entorno y el resto de recorridos que pertenecen al conjunto de patrones existentes en la *base de conocimiento de patrones aprendidos.*
  - *Similitud por velocidad*: esta estructura permite conocer el grado de similtud existente entre un objeto recién detectado con respecto el resto de los objetos que están en la base de conocimiento aprendidos.

  El tipo de objeto utilizado para este atributo es un objeto de tipo *Similarity_structure*. Este objeto ha sido creado para albergar toda la información correspondiente a la similitud del objeto con respecto al resto de objetos de la base de conocimiento que se han enumerado anteriormente. Este objeto se encarga de almacenar todos los datos referentes al grado de similitud sobre el comportamiento de objetos en el espacio observado. Para almacenar la similitud por trayectoria, el objeto contiene una lista (estructura predefinida de *Python*) en la cual se almacenan los identificadores de los patrones, con los que tiene mayor afinidad el objeto recién detectado. El principal motivo de tener esta lista con los identificadores es la de ahorrar en la búsqueda de elemento en el diccionario de similitud, de manera que en esta lista sólo se encontrarán los patrones que tienen al menos un 50 % de similitud con el movimiento del objeto recién detectado. De esta forma el acceso al patrón es directo, ya que se accede al diccionario de similitud con la clave (identificador del patrón) a la información del patrón. De forma complementaria también existe una lista de patrones con los que no existe suficiente grado de similitud en cuanto a la trayectoria. Este complementa a la lista anterior, ya que de todos los patrones que forman la *base de conocimiento de patrones aprendidos* tienen que estar presentes bien en la lista de patrones similares o en la lista de patrones descartados. La principal funcionalidad de esta estructura es conocer el estado en el que se encuentran los algoritmos de similitud, ya que al comienzo del análisis de estos algoritmos no se conoce con exactitud el grado de similitud existente entre el movimiento del objeto recientemente

detectado con respecto a los patrones existentes en la base de datos de patrones aprendidos. La incorporación de esta estructura permite conocer con cuáles de los patrones de la base de conocimiento de patrones, no existe la similitud suficiente como para seguir analizando su comportamiento.

También contiene un diccionario en el cual se almacena la similitud (tanto similitud por trayectoria, como por recorrido, como por velocidad) con cada uno de los patrones que componen la *base de conocimiento de patrones aprendidos*. Este diccionario tiene como clave, una cadena de caracteres (*string*) y como valor un objeto de tipo **Similarity_data** cuya funcionalidad se puede observar en la sección 5.2.2. Además el objeto *similarity_structure* cuenta con una lista que contiene los identificadores de los patrones que tienen un alto grado de similitud con la representación del recorrido realizado por el objeto en la escena. Al igual que con la trayectoria los elementos de esta lista son de tipo *string* o cadena de caracteres que pertenece a los identificadores de los distintos patrones que forman la *base de conocimiento de patrones aprendidos*. El principal objetivo de esta lista es agilizar el proceso de búsqueda de elementos del diccionario de similitud por trayectoria. Complementariamente a esta lista existe también otra lista que contiene los identificadores de los patrones con los que el objeto recién detectado no tiene un alto grado de similitud. Esta lista es de gran utilidad al principio del análisis de similitud, ya que permite conocer el número de patrones descartados. Lo cual se traduce en una ventaja para el algoritmo de similitud de recorrido, ya que se aborta el análisis de similitud a aquellos patrones con los que no existe una gran similitud en las primeras detecciones.

- **number_missing_frames**: Ante la dificultad de conocer si un objeto ha dejado de ser detectado en la escena, para poder utilizarlo, clasificarlo y analizarlo con respecto a los patrones, es necesario determinar un mecanismo de detección de objetos que dejan de ser detectados en la escena. Este mecanismo tiene una importancia muy elevada a la hora de crear y clasificar las nuevas trayectorias de objetos en patrones para hacer posible el aprendizaje automático de las trayectorias y velocidades que llevan los objetos en el entorno. Se trata de un atributo de tipo **entero o integer**. Su función es llevar el contador del número de frames que han pasado sin recibir información a cerca de dicho objeto. Cuando este contador supera el valor de una constante predefinida (denominada *NUMBER_MISSING_FRAMES*), entonces se considera que se han sucedido demasiados frames o fotogramas en los que no se han recibido información del objeto detectado y se determina que el objeto ha abandonado la escena. Entonces es cuando el objeto deja de formar parte de la *base de conocimiento*

de objetos detectados, para pasar a formar parte de la *base de conocimiento de patrones*.

- **color**: A la hora de representar la trayectoria o recorrido del objeto y teniendo en cuenta el diseño de la representación. Debido a que se ha diferenciado la representación de las trayectoria y recorridos de objetos a diferentes colores, dependiendo del tipo de trayectoria que realice el objeto. Es necesario la inclusión de este atributo que defina al objeto. Los colores que se han utilizado corresponden al tipo de trayectoria que realice el objeto, así pues, las trayectorias pueden ser clasificadas de tres formas diferentes: *NOT_CLASSIFIED, NORMAL o ABNORMAL.*
  - *NOT_CLASSIFIED*: Este tipo de trayectorias son representadas con el color ambar.
  - *NORMAL*: Las trayectorias de este tipo son representadas con el color verde.
  - *ABNORMAL*: Las trayectorias que son anómalas o atípicas son representadas con el color rojo.

Estructura Bounding_box

Uno de las funciones más importantes es la que permite conocer si existe colisión entre dos BB diferentes. Esta función ha sido implementada debido a la necesidad de conocer si un BB determinado es parecido espacialmente a otro BB. Se ha establecido por definición que dos BB son similares si entre ellos existe al menos un punto en el espacio común a ellos. Una forma de conocer si los dos tienen un punto en común es conociendo si entre ellos existe colisión (ver figura B.2). Esta función sirve de gran utilidad a la hora de establecer similitud entre dos trayectorias diferentes, ya que si existe colisión entre dos BB se puede decir que realizan la misma trayectoria aunque no realicen exactamente la misma, se dice que el comportamiento es muy similar dada su similitud.

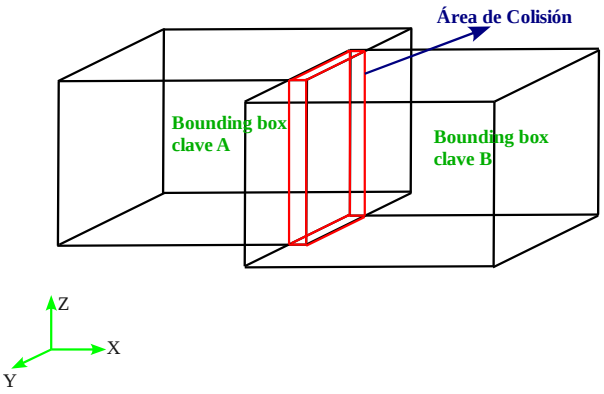

FIGURA 5.4: Representación de colisión de dos bounding box (ver en Anexo B imagen a color).

El pseudocódigo de la función de colisión es:

Algoritmo 4 Calcular la existencia de colisión entre dos BB

Entrada: Dado dos bounding box $B1(V0,V1,V2,V3,V4,V5,V6,V7)$ y $B2(V0,V1,V2,V3,V4,V5,V6,V7)$. Cada vértice es un punto en el espacio 3D de la forma $V = (V_x, V_y, V_z)$.

Salida: **cierto** si existe colisión y **falso** si no existe colisión.

Obtener el valor mínimo del eje de coordenadas X de $B1$, min_x.
Obtener el valor máximo del eje de coordenadas X de $B1$, max_x.
Obtener el valor mínimo del eje de coordenadas Y de $B1$, min_y.
Obtener el valor máximo del eje de coordenadas Y de $B1$, max_y.
Obtener el valor mínimo del eje de coordenadas Z de $B1$, min_z.
Obtener el valor máximo del eje de coordenadas Z de $B1$, max_z.
Obtener el valor mínimo del eje de coordenadas X de $B2$, min_x_2.
Obtener el valor máximo del eje de coordenadas X de $B2$, max_x_2.
Obtener el valor mínimo del eje de coordenadas Y de $B2$, min_y_2.
Obtener el valor máximo del eje de coordenadas Y de $B2$, max_y_2.
Obtener el valor mínimo del eje de coordenadas Z de $B2$, min_z_2.
Obtener el valor máximo del eje de coordenadas Z de $B2$, max_z_2.
si $(max_x < min_x_2)$ ó $(min_x > max_x_2)$ ó $(max_y < min_y_2)$ ó $(min_y > max_y_2)$ ó $(max_z < min_z_2)$ ó $(min_z > max_z_2)$ **entonces**
  devolver falso
si no
  devolver cierto
fin si

.

La funcionalidad de este algoritmo consiste en calcular el máximo y mínimo de cada BB en cada eje de coordenadas. Una vez que se ha obtenido el valor máximo y mínimo de cada coordenada se realizan las siguientes comprobaciones para conocer si dos BB colisionan.

¿Cómo se se conoce si existe colisión?, se realiza una comparación de los valores máximos y mínimos de cada BB de la siguiente manera. Para cada uno de los valores máximos y mínimos de los ejes se puede establecer que: Si el máximo valor del BB_1 en el eje de coordenadas X es menor que el mínimo valor del BB_2, entonces es imposible que exista colisión entre dos BB (ver figura 5.5).

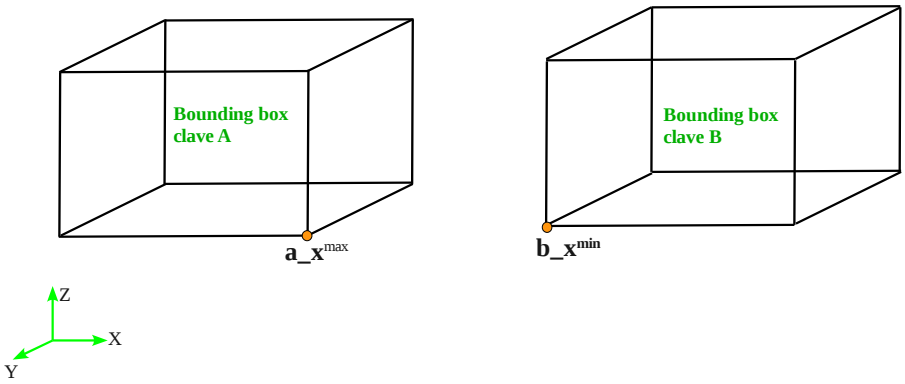

FIGURA 5.5: No existe colisión si $a_x{}^max$ es menor que $b_x{}^min$..

Al igual que tampoco puede existir colisión entre dos BB si el valor mínimo de X del BB_1 es mayor que el máximo valor de X del BB_2. Se puede ver un ejemplo en la figura 5.6.

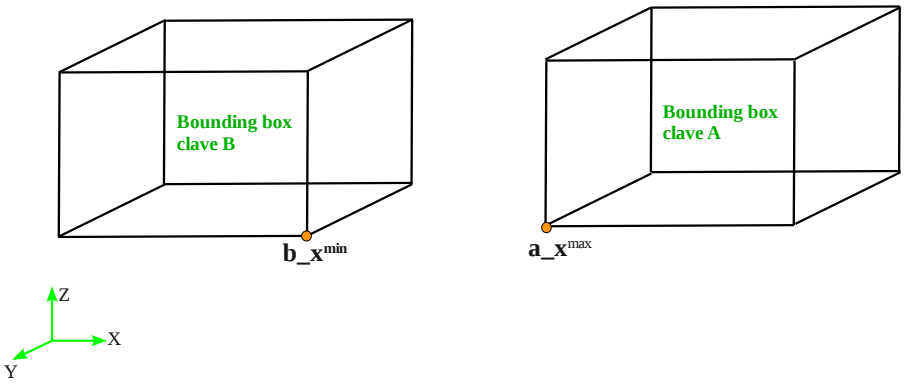

FIGURA 5.6: No existe colisión si $a_x{}^min$ es mayor que $b_x{}^max$.

De la misma forma ocurre con los dos ejes restantes tanto con Y como con Z. Ya que si se el valor máximo de Y perteneciente al BB_1 es menor que el valor mínimo en Y del BB_2 es imposible que exista colisión. Al igual que ocurre si el valor mínimo de Y del

BB_1 es mayor que el valor máximo de Y del BB_2. Esto también ocurre con los valores de los ejes de coordenadas de Z, ya que si el valor máximo de Z del BB_1 es menor que el valor mínimo de Z del BB_2 o si el el valor mínimo de Z del BB_1 es mayor que el valor máximo de Z del BB_2, es imposible que exista colisión entre dos bounding box.

La función anteriormente utilizada permite conocer si dos BB colisionan entre sí, pero es importante remarcar que sólo es válida, cuando dos BB se encuentran alineados con los ejes de coordenadas, es decir, la dirección de las aristas del BB corresponden con los ejes X, Y y Z. Sin embargo, las trayectorias de los objetos tanto las reconocidas como las almacenadas, están compuestas tanto de BB alineados con los ejes, como por BB que se no se encuentran alineados con los ejes de coordenadas. Esto dificulta la tarea de obtener la colisión entre dos BB. Cuando los BB se encuentran alineados con los ejes de coordenadas es muy sencillo, conocer las dimensiones que delimitan a un BB, ya que calculando el valor máximo en cada uno de los ejes de coordenadas se puede conocer cuales son los vértices cuyos valores son los mayores para el eje de coordenadas X, Y y Z. Por ejemplo, si queremos calcular cuales son los vértices que tienen mayor valor en Z nos encontraríamos cuatro vértices que tienen el mismo valor en Z que serían el vértice 4, el 5 el 6 y el 7 de un BB box alineado con los ejes véase la figura 5.12. Sin embargo, para obtener el valor máximo en Z puede no existe el mismo valor para cuatro de esos vértices y los cuatro vértices mayores puede que no sean identificados (ver figura 5.7).

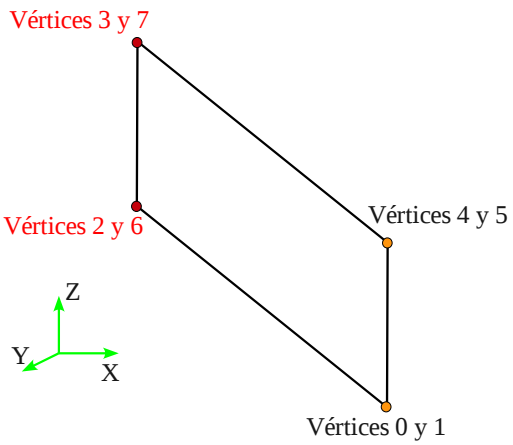

FIGURA 5.7: Representación de los cuatro vértices con mayor valor en el eje Z.

Para evitar, estos problemas de colisión entre dos BB diferentes se ha creado un nuevo método que permite calcular si existe colisión en un BB tanto si se encuentran alineados con los ejes como si no se encuentran alineados con los ejes. El método utilizado es aquel que permite calcular si un punto en el espacio está en el interior de un bounding box. De

manera que para conocer si dos bounding box colisionan basta con calcular si existe alguno de los ocho vértices que forman el bounding box está en el interior del bounding box. El método de conocer si un vértice está en el interior de un BB es que se encuentra detallado en la sección 3.2.11. Sin embargo, para obtener unos resultados perfectos se han realizado algunas modificaciones del algoritmo anterior. Existía un problema a la hora calcular el vector normal a una cara del BB y que la normal de una cara con respecto al punto puede que se calcule de forma positiva o negativa. Para que los resultados sean óptimos la normal de una cara debe calcularse hacia fuera del BB (véase figura B.4).

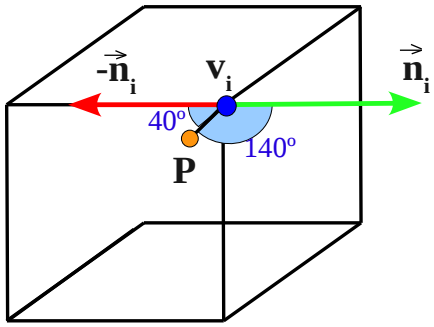

FIGURA 5.8: Representación del vector normal de una cara de un BB (ver imagen a color en Anexo B).

El principal problema de calcular el vector normal hacia fuera del BB es que posteriormente al calcular el coseno de ángulo que forman el vector normal con el vector que une a un vértice de la cara con el punto *P* que es analizado, es totalmente opuesto si se ha calculado $\vec{n}$ o si por el contrario se ha calcular el -$\vec{n}$. Si el vector que ha sido calculado es el $\vec{n}$ el ángulo que forma con el vector que une un vértice con el punto de la cara es mayor que 90 grados y menor que 180 grados cuyo coseno es negativo mientras que si el vector normal calculado es el -$\vec{n}$ el ángulo que forma es menor de 90 grados y su coseno es positivo, por tanto al calcular el producto escalar el resultado es muy dispar. Por lo que, es necesario incluir una modificación en el algoritmo para evitar esta confusión y el funcionamiento sea el correcto. El resto del algoritmo se mantiene invariable con respecto al algoritmo detallado en la sección 3.2.11. El pseudocódigo del algoritmo es:

Otro algoritmo destacable en esta estructura es la función de **Fusión de Bounding box**. se trata de la principal función existente para llevar a cabo la fusión de trayectorias. A la hora de realizar la fusión de dos trayectorias, se ha de tener en cuenta cómo se puede realizar la fusión de dos BB, para posteriormente obtener la lista de BB que forman la

Algoritmo 5 Calcular si un punto está en el interior de un BB.

Entrada: Un punto 3D $P(P_x, P_y, P_z)$ y un bounding box $B(V_0, V_1, V_2, V_3, V_4, V_5, V_6, V_7)$ definido por ocho vértices. Donde cada vértice es un punto en el espacio 3D de la forma $V = V_x, V_y, V_z$.

Salida: **cierto** Si el punto P está en el interior del bounding box y **falso** en caso de que esté fuera del bounding box.

```
para i = 0 hasta 6 hacer
    Calcular el vector normal de la cara.
    Calcular vector entre un vértice de la cara y el punto central de ésta.
    Calcular el coseno del ángulo que forma estos dos vectores.
    si El coseno del ángulo es mayor que cero entonces
        Multiplicar por -1 todos valores del vector normal calculado.
    fin si
    si Producto escalar(B.caras(i).normal(), (P - B.caras(i).vertice)) >0 entonces
        devolver falso
    fin si
fin para
devolver cierto
```

fusión de dos trayectorias completas. Es decir, la fusión de una trayectoria puede definirse como la consecuencia de una serie de fusiones de pares de BB. Un ejemplo de fusión de dos BB puede observarse en la figura B.5.

En la figura B.5 se puede observar como se realiza la fusión de dos BB vistos desde una perspectiva desde el eje z (vista de pájaro). La principal función realizada en este algoritmo es englobar a los dos BB que tienen similitud y juntarlos en un BB mayor que englobe a los dos. De esta forma, lo que se consigue es que cuanto mayor sea el número de trayectorias que se fusionen mayor es la fiabilidad de clasificación del comportamiento de un objeto sobre la escena. Para la generación del bounding box resultante de la fusión de los dos BB inicialmente proporcionados, es necesario obtener los valores máximos y mínimos en cada uno de los ejes, para cada uno de los dos BB iniciales. Posteriormente se calcula cuál de los dos BB tiene mayor y menor valor en el eje de coordenadas X , Y y Z. Una vez obtenidos los valores máximo y mínimos pertenecientes al BB resultante. Se calculan los vértices del BB resultante siguiendo la normativa de vértices explicada en la sección 5.2.4. Se devuelve el BB resultante.

El pseudocódigo del algoritmo es el siguiente:

Algoritmo 6 Cálculo de la fusión de dos BB.

Entrada: Dos bounding box $B1(V_0,V_1,V_2,V_3,V_4,V_5,V_6,V_7)$ y $B2(V_0,V_1,V_2,V_3,V_4,V_5,V_6,V_7)$ definidos por ocho vértices cada uno. Donde cada vértice es un punto en el espacio 3D de la forma $V = V_x,V_y,V_z$.

Salida: Un bounding box de la forma $B(V_0,V_1,V_2,V_3,V_4,V_5,V_6,V_7)$ que engloba a los dos BB de entrada.

Obtener el máximo valor de en el eje X de $B1$.
Obtener el mínimo valor de en el eje X de $B1$.
Obtener el máximo valor de en el eje Y de $B1$.
Obtener el máximo valor de en el eje Y de $B1$.
Obtener el máximo valor de en el eje Z de $B1$.
Obtener el máximo valor de en el eje Z de $B1$.
Obtener el máximo valor de en el eje X de $B2$.
Obtener el máximo valor de en el eje X de $B2$.
Obtener el máximo valor de en el eje Y de $B2$.
Obtener el máximo valor de en el eje Y de $B2$.
Obtener el máximo valor de en el eje Z de $B2$.
Obtener el máximo valor de en el eje Z de $B2$.
Comparar los valores máximos de X y almacenar en max_x el máximo valor de X de los dos BB.
Comparar los valores mínimos de X y almacenar en min_x el mínimo valor de X de los dos BB.
Comparar los valores máximos de Y y almacenar en max_y el máximo valor de Y de los dos BB.
Comparar los valores mínimos de Y y almacenar en min_y el mínimo valor de Y de los dos BB.
Comparar los valores máximos de Z y almacenar en max_z el máximo valor de Z de los dos BB.
Comparar los valores mínimos de Z y almacenar en min_z el mínimo valor de Z de los dos BB.
Calcular el vértice 0.
Calcular el vértice 1.
Calcular el vértice 2.
Calcular el vértice 3.
Calcular el vértice 4.
Calcular el vértice 5.
Calcular el vértice 6.
Calcular el vértice 7.
Construir BB resultante utilizando V0,V1,V2,V3,V4,V5,V6,V7.
devolver BB_resultante.

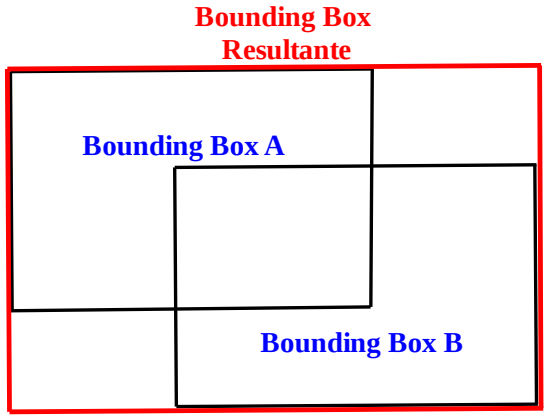

FIGURA 5.9: Representación del BB resultante de la fusión de dos BB (ver imagen a color en el Anexo B).

Estructura Features_object

Cuando un objeto es detectado en la escena observada, es necesario captar una ser de características que definen al objeto con el objetivo posterior de analizar su comportamiento. Un objeto es detectado en diferentes fotogramas o frames de imagen grabados. Por tanto, existe la necesidad de almacenar toda la información del objeto en cada frame. Para conseguir en etapas posteriores un análisis exhaustivo, ágil y eficaz es necesario organizar la información del objeto, de manera que su manipulación sea lo más ágil y rápida posible. Para ello, se ha creado la estructura **Features_object** que permite captar la información necesaria del objeto en cada fotograma que es detectado.

Esta estructura es la más cercana al archivo de texto resultante del tracking 3D. Se encarga de almacenar la información resultante en este archivo. La estructura Features_-object está formada por:

- **Bounding_Box**: Se trata de una estructura compuesta por ocho vértices o puntos tridimensionales que definen el espacio ocupado en la escena por un objeto. Proporciona las dimensiones con las que un objeto es detectado y se le aplica como una especie de caja que envuelve al objeto. El principal objetivo es la representación del objeto lo más aproximado posible a las dimensiones reales del objeto. De forma que permite la representación del objeto de forma muy sencilla(utilizando ocho vértices) y proporciona información fundamental acerca del tamaño del objeto.

Además se trata de una propiedad muy importante para determinar las dimensiones reales del objeto.

- **Frame**:Es una atributo que permite generar información necesaria para diversos casos. Es de tipo *entero o integer* y permite obtener datos como por ejemplo, el número de objetos que han sido detectados en cada fotograma. A la hora de determinar si un objeto que ha sido anteriormente detectado en la escena monitorizada y ya ha abandonado la escena, la función de este atributo resulta de vital importancia para la transformación del movimiento de un objeto en patrones.

- **name**: Es de tipo *cadena o string*. Está compuesto por una serie de caracteres. Habitualmente es conocido como el identificador del objeto, ya que éste atributo es único, es decir, no pueden existir dos objetos con el mismo identificador. Dada la necesidad de identificar todo objeto desemboca en la necesidad de dotar a cada objeto de un campo que lo distinga de los demás y sea único. Esta funcionalidad es la que caracteriza a es atributo, la de identificar a cada objeto.

- **time**: Por motivos de cálculo de la velocidad del objeto y debido a la necesidad de conocer el instante de tiempo en que un objeto es reconocido en la escena observada, se ha utilizado un atributo que almacene el instante de tiempo en el cuál se ha producido la detección del objeto. De forma que dados dos detecciones del objeto en diferente instante de tiempo se puede obtener la diferencia de tiempo entre dos detecciones del objeto y por consiguiente es posible obtener la velocidad con la que se desplaza. Es de tipo *string o cadena* y contiene sólo la información referente a la hora, minutos y segundos en los cuales se ha detectado el objeto en la escena. El formato de la cadena de caracteres viene determinado por tres parejas de dígitos separados por el carácter ":". De forma que la primera pareja de dígitos refleja el valor correspondiente a las horas, las segunda pareja de dígitos hace referencia a los minutos y la tercera pareja de dígitos contiene los segundos en los que se ha detectado al objeto. Un ejemplo puede ser, 13:05:23 que refleja que la detección del objeto ha tenido lugar a las 13 horas, 5 minutos, 23 segundos.

- **date**: Es un atributo creado que es complementario al anterior ya que complementa el instante de tiempo en el cuál ha tenido lugar la detección del objeto. Es de tipo *cadena o string* y el formato de la cadena viene determinado por ocho dígitos, de los cuáles los cuatro primeros vienen agrupados de dos en dos y separados por el carácter "/" y éstos se separan de los últimos cuatro dígitos que se encuentran unidos. Los dos primeros dígitos hacen referencia al día en que se produjo la detección del

objeto, el segundo par refleja la información acerca del mes y los últimos cuatro dígitos determinan el año en el que se produjo la detección de los objetos en la escena observada. Un ejemplo de fecha puede ser 12/06/2012, el cuál debe ser interpretado como el día 12 del mes sexto(Junio) del año de 2012.

- **velocity**: Para conocer la velocidad con la que un objeto se desplaza a lo largo de la escena monitorizada, es necesario calcularla y almacenarla en algún lado, para posteriormente realizar análisis de velocidad con respecto a los patrones que forman *la base de conocimiento de patrones aprendidos*. Para calcular la velocidad de un objeto se debe de tener una posición inicial del objeto y una posición final y conocer el tiempo que ha transcurrido para que el objeto se desplace desde la posición inicial a la posición final. De manera que el cálculo de la velocidad se puede ver en la sección 5.2.4. EL valor de velocidad que se almacena en este atributo contiene el valor de la velocidad existente entre la última detección del objeto y la detección anteriormente almacenada. De forma que se almacena la velocidad por fragmentos de la definición de trayectoria. Así pues se calcula la velocidad con la que se desplaza el objeto sobre la escena en los diferentes fragmentos que componen la trayectoria. El tipo de atributo que se almacena es *float*, permitiendo almacenar la velocidad del objeto con mayor precisión que si se utilizase un *entero o integer*.

5.2.2. Estructura Features_travel_object

La trayectoria que describen los objetos que se desplazan a lo largo de la escena, realizan un recorrido que se obtiene de su movimiento (véase sección 5.2.4). Sin embargo, para conseguir realizar un análisis entre el recorrido seguido por un objeto en concreto con respecto a los patrones existentes es necesario almacenar los datos resultantes de calcular el recorrido realizado por un objeto. Para ello se ha utilizado una estructura denominada **Feaatures_travel_object** que almacena todos los datos necesarios para el cálculo del recorrido de un objeto en un instante de tiempo determinado. Como se ha detallado en la sección 5.2.4 el recorrido sobre el movimiento de un objeto a lo largo de la escena viene determinado por una lista de objetos que lo representan. Estos objetos son de tipo *Features_travel_object* los cuáles contienen la información sobre el recorrido del objeto en cada uno de sus fragmentos definidos. Cada objeto *Features_travel_object* almacena los siguientes datos:

- **Bounding_box**: En este atributo o campo se almacena el espacio que abarca el recorrido del objeto en la última detección del objeto sobre la escena. Es de

tipo *Bounding_box* y está formado por ocho vértices tridimensionales, los cuáles determinan el espacio tridimensional que corresponde a un fragmento determinado del recorrido completo del objeto a lo largo de la escena.

- **name**: Cada vez que se desea realizar la comparación de objeto recientemente utilizado es necesario obtener el identificador de dicho objeto. EL identificador del objeto se puede conocer accediendo a la información almacenada en la trayectoria. Pero para evitar este acceso, se ha utilizado un nuevo atributo dentro del objeto, por el cuál cuando se desee comparar el recorrido de dos objetos todos los datos necesarios para la comparación se encuentran en el mismo objeto sin necesidad de realizar consultas a otras estructuras de la *base de conocimiento de objetos detectados*. Es de tipo *cadena o string* y su tamaño es variable, dependiendo del identificador asignado.

- **last_BB**: A la hora de crear el bounding box que determine el recorrido del objeto en una determinada zona del espacio 3D, es necesario conocer el bounding box correspondiente a la trayectoria realizada por el objeto y almacenada en la *base de conocimiento de objetos detectados*. Esto es, la generación del fragmento espacial correspondiente al recorrido realizado por el objeto puede que abarque varios bounding box clave correspondientes a la trayectoria, debido a que no existe suficiente variación en la dirección del objeto (véase la sección 5.2.4). Por lo que es necesario establecer aquellos bounding box desde el cuál debe calcularse el fragmento del recorrido.

  De manera que un ejemplo, imaginemos que la trayectoria está definida por 8 bounding box y en el atributo *last_BB* se encuentra almacenado el número 5. A la hora de calcular el recorrido del objeto tras la nueva detección de éste se debe tener en cuenta desde el BB número 5 hasta el BB número 8 que es el actualmente reconocido. Este atributo es de tipo *entero o integer* ya que los datos que debe almacenar corresponden a este tipo de datos.

Estructura Similarity_data

Atravel es una aplicación que se encarga de establecer correspondencia entre los patrones definidos y el movimiento de los objetos detectados en el entorno observado. Para poder obtener una similitud entre el nuevo desplazamiento del objeto en la escena y los patrones anteriormente definidos, es necesario realizar un gran número de comparaciones entre patrones para establecer el grado de similitud entre el nuevo movimiento del objeto y el conjunto de patrones que forman la base de conocimiento. Estas comparaciones se

realizan accediendo a los datos almacenados en las estructuras del objeto y de lo patrones, sin embargo, se ha tenido que generar una estructura para almacenar todos estos datos resultantes de la comparación entre cada patrón y los objetos que se están detectando en la escena monitorizada. La estructura encargada de almacenar todos estos datos es la *Similarity_data*. En la aplicación se ha diseñado un diccionario particular para cada objeto recientemente detectado. De manera que en este diccionario se almacenan todos los datos resultantes de la comparación de cada objeto en concreto con cada uno de los patrones que constituyen la *base de conocimiento de patrones aprendidos*. Este diccionario tiene como clave el identificador de cada patrón de la base y el valor es un objeto de tipo *Similarity_data*. Los principales datos que se almacenan en esta estructura son:

- **BB_checked_vector**: Se trata de un vector de tipo lista que contiene el número de bounding box(BB) que componen la trayectoria de los objetos a lo largo de su desplazamiento sobre la escena. Es de tipo lista, cuya estructura está construida en el interior del núcleo de *Python* (sección 3.1.1). Esta lista contiene una serie de valores de tipo *boolean* que corresponden con el número de bounding box clave que forman la trayectoria del objeto. El número de elementos que se pueden encontrar en esta lista corresponde con el número de bounding box clave que definen la trayectoria del objeto. Los valores que se adquieren en esta lista hacen referencia a la correspondencia entre un los bounding box clave de la nueva trayectoria realizada por el objeto con respecto a los bounding box clave que forman la trayectoria de los patrones almacenados.

  Este atributo contiene una lista de elementos con el valor de *True* o *False* dependiendo de si el BB clave del objeto es similar al BB clave del patrón analizado. Un BB clave de un objeto es similar a un BB clave del patrón en concreto, cuando estos dos BB tienen al menos algún punto en el espacio en común. Por tanto, se almacena *True* cuando exista colisión entre los dos BB, y se almacena *False* cuando no exista colisión entre los dos bounding box. El seudocódigo de esta función se detalla en el algoritmo 11. De forma que, en esta lista se encuentra la lista de respuestas que se han obtenido tras la comprobación de similitud de los BB del objeto detectado recientemente con el patrón. La creación de este atributo tiene como principal motivo, almacenar la similitud entre BB clave para posteriormente analizar el grado de similitud con el patrón. Conociendo el número de BB clave a los que se parece se puede obtener de forma muy sencilla el grado de similitud entre las dos trayectorias. Por ejemplo si un patrón está formado por 5 BB claves, y en esta lista hay almacenados cuatro elementos con valor *True* y uno con valor a *False*, significa que el grado de similitud

es del 80 % ya que no existe una similitud total con el patrón. Por tanto, se trata de un campo o atributo de gran importancia en el cálculo de la similitud entre trayectorias.

- **similarity_porcentage**: A la hora de conocer el grado de similitud se ha de tomar alguna medida estándar, que pueda ser interpretada de forma homogénea por la mayoría de los usuarios. En la representación del grado de similitud se ha utilizado el **tanto por ciento**. Se podría haber establecido alguna otra medida del grado de similitud entre dos bounding box, como por ejemplo establecer una escala desde cero hasta cinco, siendo el máximo grado de similitud 5 y el menor 0. Con la utilización de *tanto por ciento*, permite conocer con exactitud el grado de similitud de forma estándar y normalizada. Además es muy usable y entendible por la mayoría de las personas. Dadas dos trayectorias $t1$ y $t2$ se dice que son idénticas si el grado de similitud es del 100 %, sin embargo, se dice que son totalmente distintas si el grado de similitud es de 0 %. De forma que cuanto más similitud exista entre las dos trayectorias mayor porcentaje es almacenado. Este atributo es de tipo *float*.

- **BB_travel_vector**: Al igual que es necesario obtener la similitud por trayectoria también es necesario calcular la similitud por recorrido. Debido a que se trata conceptos diferentes, cuya definición ha sido detallada en la sección 5.2.4. Se trata de un atributo de tipo lista, y los elementos que forman la lista son de tipo *Boolean*. Los elementos de la lista toman el valor *True* en caso de que exista similitud entre el objeto recientemente detectado y el patrón que es analizado en cada momento. En caso de que no exista similitud entre el objeto recién detectado y el patrón se almacena en la lista el valor de *False*. Para incorporar a la lista si existe similitud entre un nuevo BB de un objeto detectado es necesario realizar diferentes comprobaciones, no sólo del BB correspondiente sino que además realiza la comprobación con los dos BB anteriores y los dos BB posteriores al correspondiente, para obtener mayor precisión en la similitud de dos trayectorias. En esta comprobación no sólo se realizan las comprobaciones de BB claves sino que también se realiza comprobación de similitud entre BB intermedios. Debido a la comprobación de similitud entre BB intermedios, es necesario la utilización de un método distinto para conocer si dos BB colisionan entre sí. La función utilizada para la colisión de dos BB se puede ver en el algoritmo 11.

- **travel_porcentage**: Debido a la generación de similitud por recorrido es necesario conocer una medida que aporte al usuario un grado de similitud existente entre dos trayectorias. Para ello se ha utilizado el **tanto por ciento o porcentaje**. Esta medida aporta al usuario un valor acerca del grado de similitud existente entre la trayectoria

del objeto recién detectada y los patrones existentes en la base de conocimiento. Es de tipo *float* para dotar de mayor precisión al porcentaje calculado. El grado de similitud es mayor cuanto más cercano es el porcentaje a 100 % y más bajo es el nivel de similitud cuanto más cercano es el porcentaje a 0 %.

- **travel_dicarded**: Cuando se desea obtener el grado de similitud de la trayectoria de un objeto con respecto a los patrones almacenados en la *base de conocimiento de patrones aprendidos* es necesario tener un mecanismo de parada, que evite seguir realizando la comparación con el patrón si en las primeras comprobaciones no existe similitud.

  Para evitar, comprobar la similitud con cada uno de los patrones de la base de conocimiento de patrones cada vez que se detecta una nueva posición del objeto se ha utilizado un atributo que permite evitar la comprobación con aquellos patrones con los que anteriormente no existía similitud suficiente. De esta forma, proporciona mayor eficacia y agilidad en la comprobación de similitud entre trayectorias evitando comprobaciones innecesarias, ya que si una trayectoria no es similar al principio, ya se considera una trayectoria distinta.

  Además este atributo permite evitar las comparaciones cada "2" fallos, es decir, aunque una trayectoria no tenga similitud con el BB correspondiente de un patrón, la aplicación sigue realizando comprobación de similitud con dicho patrón hasta llegar a dos veces como máximo. Es decir, si un objeto realiza la comprobación con un patrón y no existe similitud no se desecha dicho patrón sino que es chequeado en la siguiente detección nueva del objeto. En caso de que se realice una nueva comprobación con el patrón y no exista similitud, entonces el patrón no se vuelve a comprobar y se descarta en cuanto a posible similitud.

  De manera que actúa como contador, en caso de que el valor de este atributo tenga valor 2, entonces ya no se vuelve a chequear la similitud con dicho patrón, ya que se considera diferente a la representación de su trayectoria. El tipo de este atributo es *entero o integer*.

- **travel_last_tocuhing**: Para conocer el grado de similitud existente utilizando el método de similitud por recorrido, detallado en la sección 5.3, ha sido necesario incorporar este atributo para conocer cuál es el último bounding con el cuál ha habido coincidencia, es decir, el BB perteneciente al patrón con el que ha habido colisión con el nuevo BB del objeto detectado en la escena. La creación de este atributo es necesaria, ya que el algoritmo de similitud realiza comprobaciones no sólo con el BB del patrón correspondiente, sino también con los dos BB anteriores y con los dos

BB posteriores, de manera que en posteriores comprobaciones es necesario conocer cuál fue el último BB con el que hubo colisión para chequear a partir de ahí y evitar falta de precisión en la comprobación de similitud. El tipo de este atributo es *entero o integer*.

- **velocity_list**: Este atributo es de tipo *lista*. Los elementos que componen la lista son de tipo *boolean*. Este atributo ha sido utilizado para almacenar la similitud por tramos o fragmentos entre la trayectoria de un objeto detectado a lo largo de la imagen y la trayectoria de cada patrón almacenado en la base de conocimiento. Una vez que se obtiene la velocidad a la que se desplaza un objeto sobre la escena, se chequea con respecto a la velocidad a la que se desplaza un patrón en el lugar correspondiente. Los elementos de la lista tomarán el valor *True* cuando la velocidad con la que se desplaza un objeto es similar a la velocidad del patrón en dicho tramo. Se dice que dos objetos se desplazan a la misma velocidad si coincide su valor o si no supera un valor por encima o por debajo que actúa de margen con respecto a la velocidad. En caso de que la velocidad supere el margen de velocidad, bien en positivo o bien en negativo el valor de los elementos de la lista tomará un valor a *False*.

- **traveler_vector**: Ante la necesidad de calcular el grado de similitud referente al recorrido realizado por el movimiento de un objeto a lo largo de la escena monitorizada. Es necesario almacenar el número de BB en los que se ha producido colisión entre BB que componen el recorrido de dos trayectorias de dos objetos distintos. El atributo *traveler_vector* es de tipo *lista* y los elementos que va a contener son de tipo *boolean*. Un elemento de la lista tomará valor *True* cuando exista colisión entre el BB clave que forma el recorrido de un objeto recién detectado y el BB clave correspondiente al recorrido de cada patrón perteneciente a la *base de conocimiento de patrones aprendidos* y tomará valor de *False* cuando no existe colisión entre los BB. Este atributo ha sido necesaria su creación, ya que para calcular el porcentaje de similitud por recorrido es fundamental conocer el número de colisiones existentes para obtener el porcentaje real de similitud del recorrido de dos trayectorias.

- **traveler_porcentage**: Es de tipo *float* y contiene el valor de similitud de recorrido entre dos trayectorias del objeto. Para representar el grado de similitud por recorrido se ha utilizado el *tanto por ciento* ya que permite representar manera formal el grado de similitud existente entre dos recorridos distintos.

Estructura Similarity_structure

Para conseguir el objetivo de aprendizaje automático, la aplicación debe adquirir determinada informacion durante un período de tiempo que le permita clasificar posteriormente el comportamiento de determinados objetos que aparecen en el entorno monitorizado. Para realizar esta clasificación de comportamiento, es necesario realizar comparaciones entre el comportamiento de los objetos recien detectados con respecto a los patrones ya aprendidos y almacenados en la base de conocimiento. Cada objeto detectado en la escena observada debe ser comparado con cada uno de los patrones que forman la *base de conocimiento de patrones aprendidos*, para conocer a qué comportamiento tiene más afinidad. Una vez realizada la comparación de un objeto determinado en un instante determinado es necesario almacenar toda esta información en alguna estructura para poder ser utilizada en posteriores detecciones del mismo objeto en diferentes instantes de tiempo. Además la estructura de almacenamiento debe ser lo más flexible y fácil de manipular posible, dado el variable número de patrones que pueden existir. Para albergar toda esta información se ha utilizado una estructura de **diccionario**, en la cuál la clave viene determinada por el identificador del patrón correspondiente y su valor es un objeto de tipo *Similarity_data* (véase sección 5.2.2). De esta forma en este diccionario se almacena para cada objeto la similitud de éste con cada uno de los patrones existentes en la *base de conocimiento de patrones*, con todos los datos necesarios para conocer el grado de similitud del objeto con cada uno de los patrones. Además para agilizar el acceso a este diccionario se han utilizado cuatro listas, las cuáles permiten acceder exclusivamente a las entradas del diccionario (correspondientes a los patrones) que tienen suficiente afinidad como para terminar clasificando el comportamiento de un objeto con dicho patrón o patrones. A continuación, se realiza una explicación más detallada de la funcionalidad que cubren los elementos de esta estructura:

- **list_similar_path**: Es de tipo *lista*.Los elementos que forman la *lista* son de tipo *cadena o string*. Este atributo se ha creado con el objetivo de agilizar el proceso de similitud por trayectoria. Los elementos de la lista contienen los identificadores de los patrones que forman la *base de conocimiento de patrones aprendidos* con los que existe suficiente afinidad con respecto a la trayectoria del objeto detectado. Por tanto, en esta lista, sólo aparecerán los identificadores de los patrones cuya trayectoria es similar a la trayectoria del objeto detectado en la escena monitorizada. De forma que cuando vuelva a detectarse en la escena el objeto en una posición diferente, sólo se comprobará la similitud por trayectoria con los patrones que aparecen en esta lista.

- **list_descarted_path**: Es de tipo *lista*. Los elementos que componen la *lista* son de tipo *cadena o string*. Esta *lista* es complementaria a la lista anterior. Ya que

contiene los identificadores de los patrones que no tienen la suficiente afinidad con la trayectoria del objeto detectado en la escena. Tiene como objetivo complementar a la lista *list_similar_path* en determinadas condiciones del algoritmo de similitud por trayectoria.

- **similarity_store**:El lugar en el cuál se almacena la información de similitud del objeto con respecto a todos los patrones existentes en la base de conocimiento. Es de tipo de datos de *diccionario*. Tiene como clave, el identificador de cada patrón de la base y como valor un objeto de tipo *Similarity_data*, en cual se almacena toda la información de similitud, tal y como se explicado en la sección 5.2.2. La razón por la que se ha utilizado este atributo, es debido a la necesidad de almacenar en alguna estructura toda la información tanto de similitud por recorrido, como por trayectoria como por velocidad, del objeto con cada uno de los patrones, con el fin de obtener al final el patrón con el que existe mayor afinidad. El motivo por el que se ha decidido utilizar una estructura de diccionario son principalmente la flexibilidad que aporta para almacenar la información, y el fácil y rápido acceso a cada uno de los elementos que lo componen.

- **list_similar_travel**: Es de tipo *lista*, los elementos que forman la lista son de tipo *string o cadena de caracteres*. Esta lista tiene la misma funcionalidad que la lista *list_similar_path*, es decir, agilizar las comparaciones de similitud, permitiendo sólo la comparación de similitud con aquellos patrones con los que existe un mínimo de afinidad. Esto es cuando un patrón no tiene la similitud necesaria, es eliminado de esta lista para evitar seguir comparando su similitud con el objeto concreto, ya que por definición se considera que si dos recorridos no son similares al comienzo de su recorrido, se consideran recorrido diferentes y no es necesario seguir realizando comparaciones de similitud, porque ya se trata de recorridos diferentes.

- **list_descarted_travel**: Este atributo es complementario al atributo de *list_similar_travel*. Su función es almacenar los identificadores de los patrones de la *base de conocimiento de patrones aprendidos* para tener una noción de aquellos patrones que no tienen afinidad en el recorrido con el objeto detectado. Es de tipo *lista* y los elementos que forman la lista son de tipo *string o cadena de caracteres*, ya que almacenan el identificador de los distintos patrones descartados en la tarea de obtener el grado de similitud con respecto a los patrones.

5.2.3. Base de conocimiento de patrones aprendidos

En este submódulo se realiza el almacenamiento de toda la información correspondiente a los patrones aprendidos durante un determinado período de monitorización de la escena monitorizada. Los patrones almacenados en esta estructura deben mantener toda la información de su trayectoria, recorrido, velocidad, etc. Un patrón es en realidad, un objeto que ha sido desplazado por la escena monitorizada y ha dejado completado su movimiento por la escena. Un patrón hereda todas las características procedentes de un objeto almacenadas en la *base de conocimiento de objetos detectados*. Además incorpora nuevas características como un identificador nuevo, el tipo de comportamiento realizado y la fecha en la que se ha producido su creación.

Un patrón puede ser determinado por el movimiento de un objeto por la escena. Conforme van apareciendo nuevos comportamientos de los objetos a lo largo de la escena se va modificando el número de patrones, ya que si existen varios comportamientos similares de objetos los patrones se van fusionando con el objetivo de ir asemejándose lo máximo posible a los patrones existentes en la escena. Así pues, tras un determinado período de monitorización de la escena el sistema ha adquirido un número determinado de patrones y tras determinadas pasadas de un objeto por una trayectoria y una velocidad en concreto dicho comportamiento es declarado como normal, ya existen suficientes objetos que han realizado el mismo comportamiento. En caso de que un comportamiento no haya sido detectado nunca en la escena, el sistema lo clasifica como comportamiento anormal ya que no existen patrones que hayan tenido el mismo comportamiento. Inicialmente cuando la base está vacía, los patrones reconocidos son clasificados con el tipo **NOT_CLASSIFIED** conforme se van detectando comportamientos similares el sistema va fusionando patrones y actualizando el número de veces que un comportamiento es detectado en la escena observada. Cuando se supere un número determinado de patrones con el mismo comportamiento entonces es clasificado como un comportamiento normal. En caso de que haya pasado demasiado tiempo y una trayectoria de tipo *NOT_CLASSIFIED* no tenga similitud con ningún otros comportamiento, entonces es clasificada como *ABNORMAL*.

Dada la importancia de esta base de conocimiento, su estructura debe ser lo más flexible posible. Para implementar esta base de datos se utilizado una estructura de *diccionario*. Esta estructura se caracteriza por su agilidad en la inserción de objetos. En cada entrada del diccionario se almacenará cada uno de los patrones reconocidos y aprendidos. Cada elemento del diccionario tiene como clave el identificador del patrón reconocido y como valor un objeto de tipo *Trajectory_pattern* que contiene toda la información acerca del patrón.

Estructura Trajectory_pattern

Esta estructura ha sido implementada para almacenar toda la información acerca de un patrón. Los principales elementos que se almacenan en esta estructura son toda la información acerca de la trayectoria, velocidad y recorrido que se encuentra almacenado en un objeto de tipo *Trajectory*. Con toda la información explicada en la sección 5.2.4. También se almacena en esta estructura, la categoría en la que es clasificado el comportamiento del objeto. El comportamiento del objeto puede ser clasificado en tres categorías *NORMAL*, *ABNORMAL* y *NOT_CLASSIFIED*, la fecha en la que se ha creado el patrón y el tipo de patrón que es. Estas son las principales características que almacenan en esta estructura:

- **Object_current**: Contiene toda la información del patrón acerca de la trayectoria, el recorrido y la velocidad que definen el comportamiento espacial del patrón. Este atributo es de tipo *Trajectory*.
- **Category**: En este atributo se almacena la información referente al comportamiento del patrón. Puede tomar tres valores distintos dependiendo de su comportamiento. Si el comportamiento del patrón es detectado un número importante de veces es clasificado como *NORMAL*, si el comportamiento del patrón está recientemente detectado y sólo ha sido detectado una vez es clasificado como *NOT_CLASSIFIED*. Sin embargo, si nos e detecta ningún otro patrón en un tiempo determinado su categoría es modificada y pasa a ser clasificado como *ABNORMAL*.
- **Type_object**: Este campo ha sido implementado, dada la necesidad de distinguir entre un movimiento realizado por un vehículo y otro realizado por un peatón, de forma que si el patrón reconocido es realizado por un vehículo este atributo tomará como valor la cadena de caracteres *VEHICLE* y si es realizado por un peatón tomará el valor *PEDESTRIAN*.
- **Date**: Este atributo contiene la fecha en la que ha sido detectado el patrón. Es de tipo *string o cadena* y viene determinado por seis dígitos separados por el carácter ":" dos a dos. De forma que el primer par de dígitos hace referencia a la hora en la que ha sido creado, la segunda pareja de de dígitos hace referencia a los minutos en los que ha tenido lugar la creación del patrón y la tercera pareja de dígitos hace referencia a los segundos.

5.2.4. Submódulo de representación de trayectoria 3D.

Este submódulo es el encargado de obtener la información necesaria de los objetos detectados en el entorno observado a través del archivo de tracking proporcionado por el módulo de entrada de la aplicación.

El archivo obtenido de la etapa de *tracking 3D*, especificado en la sección 5.1.1, contiene las características fundamentales de los objetos en cada frame (fotograma). Por tanto, se puede conocer el número de objetos detectados en cada frame de la escena monitorizada. De manera que para representar la trayectoria realizada por un objeto a lo largo de la escena, es necesaria la detección de las diferentes posiciones del objeto detectadas durante todo su movimiento sobre el entorno observado. Un objeto detectado en la escena es representado mediante un bounding box (caja tridimensional que envuelve al objeto detectado), como se puede observar en la figura 5.2.

En una primera aproximación, un boundng box venía definido por sólo dos vértices, el vértice inferior y más a la izquierda y el vértice superior más a la derecha tal como muestra la figura 5.10

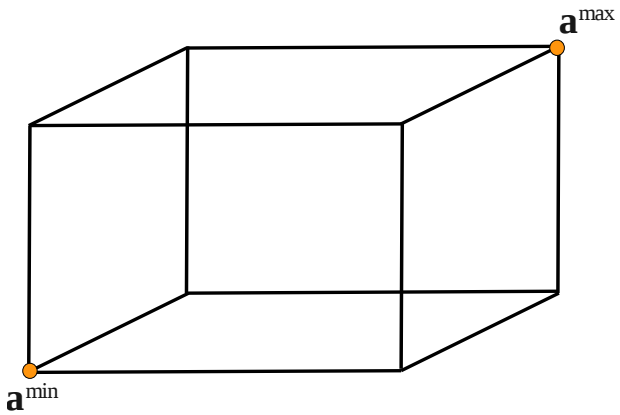

FIGURA 5.10: Representación de un bounding box utilizando sólo dos vértices.

Debido a que el bounding box viene determinado por el vértice inferior más a la izquierda y el vértice superior más a la derecha, el resto de vértices que forman el bounding box. Esto es un procedimiento muy sencillo en el cual sólo hay que realizar una resta sobre los valores correspondientes a cada eje de coordenadas. De manera que si el bounding box A viene determinado por $a^{min} = (min_x, min_y, min_z)$ y $a^{max} = (max_x, max_y, max_z)$ se puede obtener la anchura, altura y profundidad del bounding box de la siguiente manera:

$$anchura = \mid max_x - min_x \mid \; altura = \mid max_z - min_z \mid \; profundidad = \mid max_y - min_y \mid \quad (5.1)$$

Una vez que se obtiene la anchura, altura y profundidad del bounding box que envuelve al objeto detectado, se pueden calcular las coordenadas 3D de todos los vértices que componen el bounding box:

$$\begin{gathered} vertice_0 = (max_x, max_y, max_z - altura) \\ vertice_1 = (max_x, max_y - profundidad, max_z - altura) \\ vertice_2 = (min_x, min_y, min_z) \\ vertice_3 = (min_x, min_y + profundidad, min_z) \\ vertice_4 = (max_x, max_y, max_z) \\ vertice_5 = (max_x, max_y - profundidad, max_z) \\ vertice_6 = (min_x, min_y, min_z + altura) \\ vertice_7 = (min_x, min_y + profundidad, min_z + altura) \end{gathered} \tag{5.2}$$

Por tanto, la definición de un bounding box(BB) utilizando sólo dos vértices permite la representación de éste sin necesidad de almacenar más información. La representación de una trayectoria en tres dimensiones viene determinada por la unión de cada uno de los BB detectados junto con los BB que los unen. El motivo que ha llevado a esta decisión, es la necesidad de almacenar la trayectoria que describe el movimiento de un objeto con el fin, de realizar una posterior clasificación más precisa sobre la similitud de comportamientos de objetos detectados en la escena de monitorización. Ya que los objetos detectados son representados con un objeto de tipo BB, el desplazamiento intermedio entre cada detección del objeto también ha sido implementado de tipo BB, para facilitar su representación, aprovechando la representación.

Sin embargo, a la hora de definir una trayectoria como una serie de bounding box enlazados entre sí, aparecen problemas importantes al realizar la representación de una trayectoria. El principal problema que existe, tiene lugar al enlazar dos bounding box clave utilizando un bounding box denominado intermedio, ya que permite representar el movimiento del objeto desde un bounding box clave hasta otro bounding box clave,como se puede ver en la figura 5.11. Por tanto, las aristas de un bounding box clave se encuentran alineados con los ejes de coordenadas (X,Y y Z), mientras que los bounding box intermedios, no están alineados con los ejes. El bounding box intermedio es un bounding box deformado, es decir, sus aristas no se encuentran alineadas con los ejes de coordenadas, y tienen un tamaño variable, en función de la distancia entre cada vértice de unión (ver figura 5.11).

Dada la necesidad de representación de bounding box cuyas aristas no se encuentran orientadas conforme a los ejes de coordenadas, es necesario modificar la primera

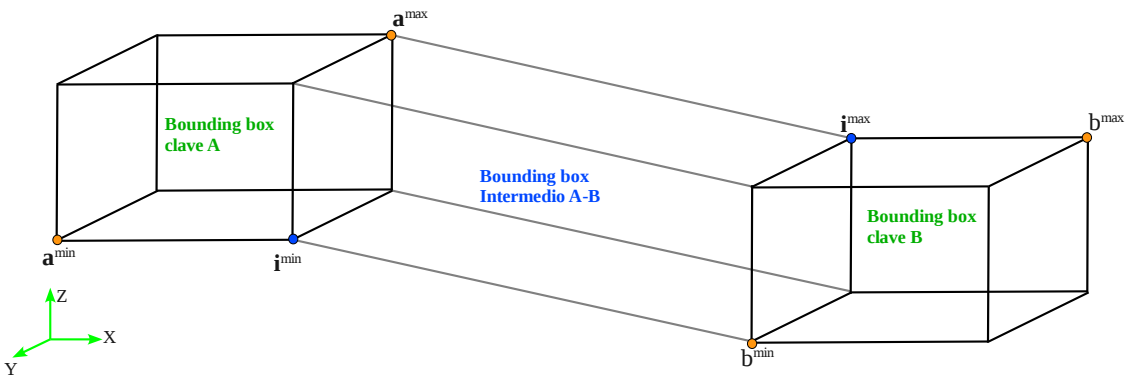

FIGURA 5.11: Representación de un bounding box intermedio.

aproximación de definición de bounding box. De manera que almacenando la información correspondiente de todos los vértices que constituyen un bounding box, no habría ningún problema a la hora de representar aquellos bounding box que se encuentren deformados.

Por tanto, la definición de un bounding box viene determinada por las coordenadas 3D de los ocho vértices que lo componen:

$$Bounding_box = (vertice0, vertice1, vertice2, vertice3, vertice4, vertice5, vertice6, vertice7) \tag{5.3}$$

Además para facilitar su manipulación y representación se ha determinado un orden por definición. El orden que siguen se puede observar en la figura 5.12.

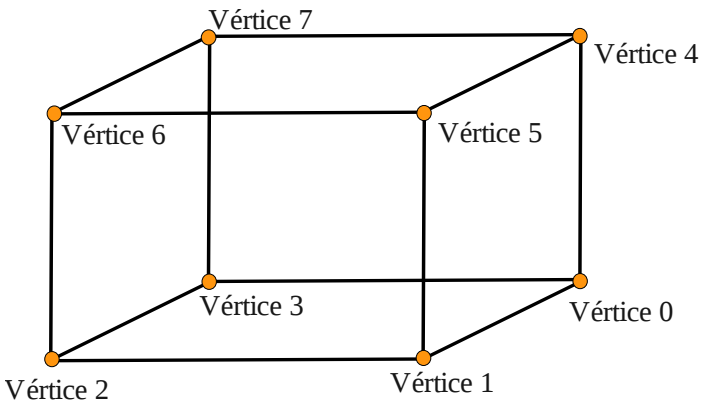

FIGURA 5.12: Representación de los vértices de un bounding box.

Para la correcta representación de las trayectorias de los objetos, es necesario realizar una buena definición de los bounding box intermedios, ya que gran parte del éxito de

similitud de trayectorias y aprendizaje de patrones se basa en una buena representación del movimiento realizado por los objetos a lo largo de la escena. Para la representación del bounding box intermedio, es necesario definir los ocho vértices que lo componen. Para ello, se ha decidido que el bounding box intermedio venga definido por cuatro vértices pertenecientes al bounding box clave situado a la izquierda con respecto al bounding box intermedio y los otros cuatro vértices pertenecientes al bounding box clave situado más a la derecha tal y como se puede observar en la figura 5.13.

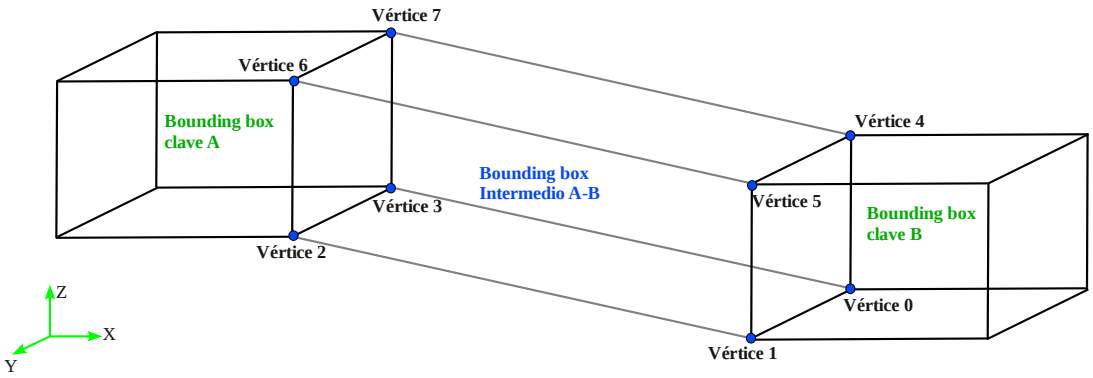

FIGURA 5.13: Representación de los vértices de un bounding box intermedio.

A la hora de obtener los vértices que van a formar el bounding box intermedio, es importante determinar exactamente cuales de éstos, son los más adecuados. Para ello, se ha de determinar los cuatro vértices de cada BB son los que definen de forma más adecuada el BB intermedio. Además, es necesario tener en cuenta que los cuatro vérties elegidos de cada BB deben pertenecer a una misma cara de este.

Un bounding box viene determinado por seis caras, donde una cara está definida por cuatro vértices coplanares (a modo de rectángulo). Las caras se encuentran ordenadas para facilitar su manipulación en algoritmos y cálculo de operaciones. El orden de las caras que forman el bounding box es el siguiente:

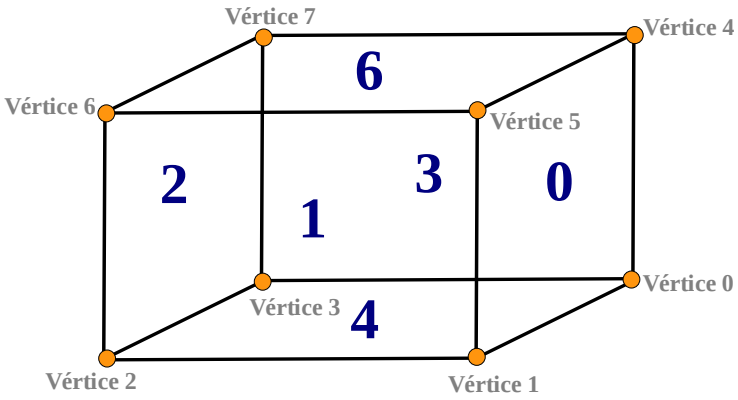

FIGURA 5.14: Representación de las caras que componen un bounding box.

Una cara viene determinada por una lista de cuatro vértices con un orden específico (ver imagen 5.14):

$$\begin{aligned}
&Cara0 : [< vertice0 >, < vertice3 >, < vertice7 >, < vertice4 >] \\
&Cara1 : [< vertice0 >, < vertice1 >, < vertice5 >, < vertice4 >] \\
&Cara2 : [< vertice1 >, < vertice2 >, < vertice6 >, < vertice5 >] \\
&Cara3 : [< vertice2 >, < vertice3 >, < vertice7 >, < vertice6 >] \\
&Cara4 : [< vertice0 >, < vertice1 >, < vertice2 >, < vertice3 >] \\
&Cara5 : [< vertice4 >, < vertice5 >, < vertice6 >, < vertice7 >]
\end{aligned} \tag{5.4}$$

De manera que un bounding box puede ser definido por una lista de seis elementos y en la que cada cara es una lista de cuatro elementos:

$$BB : [[< Cara0 >], [< Cara1 >], [< Cara2 >], [< Cara3 >], [< Cara4 >], [< Cara5 >]] \tag{5.5}$$

Donde cada cara está compuesta por una lista de cuatro vértices en un orden específico tal y como se muestra en la ecuación 5.4.

Por tanto, para obtener los cuatro vértices de cada bounding box clave los cuatro deben formar una cara del bounding box clave de la izquierda y los otros cuatro deben pertenecer a otra cara del bounding box clave de la derecha. De manera que al final se puede resumir el proceso como la elección de una cara de cada bounding box para realizar la representación del bounding box intermedio.

La cuestión principal es conocer cuáles son las caras idóneas pertenecientes a los dos bounding box clave para realizar la representación del bounding box intermedio.

Para resolver este problema se han realizado un algoritmo basado en el mecanismo de intersección de un rectángulo con un segmento (o recta). De forma, que las caras resultantes, para realizar la representación de bounding box intermedio se obtienen de la intersección del segmento, que tiene como origen el punto central de uno de los dos bounding box clave y como punto de destino el punto central del otro bounding box clave, con las caras correspondientes de los dos bounding box.

Algoritmo elección de caras.

Una vez que se han obtenido las caras que van a formar el bounding box intermedio, es necesario realizar una ordenación de éstos con el orden especificado en la figura 5.12. De esta forma, se puede realizar posteriormente una representación de forma directa. El procedimiento de ordenación de vértices consiste en calcular la distancia de un vértice perteneciente a un bounding box con el resto de vértices que pertenecen al otro bounding box. Dada la distancia existente entre pares de vértices pertenecientes a cada bounding box se realiza la ordenación de éstos para que después tenga lugar una correcta representación. Para ello, se obtienen las cuatro relaciones en los que la distancia es menor con el objetivo de que el bounding box represente lo mejor posible el movimiento realizado por el objeto.

El procedimiento seguido para la ordenación de vértices que constituyen el bounding box intermedio es el siguiente:

1. Dadas las dos caras iniciales se obtiene cuál de ellas se encuentra a la derecha o a la izquierda con respecto del eje X.

2. Se obtienen los vértices que forman la cara que se encuentra a la derecha y se calculan aquellos vértices que tienen el valor de la coordenada Z más bajo. Después obtenidos estos dos puntos, se calcula cuál de ellos tiene el valor de la coordenada Y más pequeño, si resulta que los dos contienen el mismo valor en el Y, se pasa a analizar el valor contenido en X, el vértice que contenga mayor valor en el eje X, ese es considerado el vértice 0 del bounding box intermedio resultante. Dados los dos vértices cuyos valores en las coordenadas Y sean diferentes, el vértice 0 será aquel que tenga menor valor Y. En ambos casos, de los vértices seleccionados inicialmente con un valor en el eje de coordenadas Z menor, y una vez definido el vértice 0, el otro vértice es definido como vértice 4 de la cara analizada. Una vez que se han seleccionado el vértice 0 y el vértice 4 de la cara, se pasa a calcular cuál de los otros pertenece a los vértices 4 y 7. Para ello, se analiza cuál de estos dos tiene mayor valor en la coordenada Y, el que tenga menor valor en este eje de coordenadas será

definido como vértice 7. En caso de que los dos vértices tengan el mismo valor con respecto al eje de coordenadas Y, se pasa a analizar su valor en el eje de coordenadas X. Ahora el vértice 7 será aquel que tenga mayor valor en el eje de coordenadas X. En ambos casos y una vez determinado el vértice 7 el vértice restante sería definido como vértice 4. Es importante, recalcar, que dada una cara de un bounding box, es imposible que dos vértices tengan el mismo valor en el eje de coordenadas en el eje Y y en el eje X, ya que entonces dejaría de ser una cara y sería una recta, por lo que esta opción no ha sido contemplada.

3. Ahora se vuelve a realizar la ordenación de vértices para la cara de la izquierda, se van a calcular la ordenación de los vértices 1, 2, 5 y 6. El procedimiento sería el mismo que el explicado en el punto dos, con la única diferencia de que se calculan los vértices anteriormente comentados. Es decir, primero se obtienen los vértices con menor valor en el eje Z, después y una vez seleccionados éstos se pasa a especificar cuál de los dos vértices tiene menor valor en el eje Y, el que tenga menor valor en Y será determinado como vértice 1. En caso de que el valor de las coordenadas de ambos sean idénticas, se analizará el valor de éstas en el eje X. De los dos vértices con menor valor en Z y que contenga mayor valor en el eje X, será designado como vértice 1. En ambos casos y una vez determinado el vértice 1 se pasa el vértice restante de los dos es designado como vértice 2. Una ve que están determinados los vértices 1 y 2 se pasa a determinar cuál de los dos vértices que tienen mayor valor en el eje de coordenadas Z es designado como vértice 5 y vértice 6. Para ello, se analiza el valor de ambos en el eje Y, el vértice que tenga mayor valor en el eje de coordenadas Y será designado como el vértice 6, en caso de que ambos tengan el mismo valor en Y, se analizará el valor de éstos en el eje de coordenadas X. Aquel que tenga mayor valor en el eje X será determinado como vértice 6. En ambos casos y una vez designado el vértice 6 el otro vértice pendiente será designado como vértice 5.

4. Después de tener ordenados los vértices que constituyen el bounding box intermedio, éste es creado y almacenado en su lugar correspondiente.

Una trayectoria en 3D viene determinada por una serie de bounding box consecutivos que representan el movimiento del objeto a lo largo del entorno observado.

Submódulo de optimización de trayectorias.

Cuando un objeto es detectado en la escena, éste es reconocido por el sistema y almacenado en el archivo de texto resultante tras la etapa de *Tracking 3D*. El objeto es

envuelto en un bounding box ajustado a las dimensiones del objeto detectado. En el archivo de texto se reflejan las características del bounding box que envuelve al objeto. Sin embargo, almacenar toda la información de todas las detecciones del objeto en la escena monitorizada puede resultar ineficiente e ineficaz en la etapa de análisis de comportamiento. Ya que al comprobar y analizar información irrelevante o que no aporte datos lo suficientemente importantes puede ocasionar mayor dificultad de cálculo y la necesidad de un mayor número de recursos para la ejecución de la aplicación.

Para evitar la representación de una trayectoria 3D utilizando un número excesivo de bounding box, se ha creado un submódulo de optimización las trayectorias 3D. Este módulo permite proporciona a *Atravel* mayor eficiencia y eficacia en la representación de las trayectorias 3D.

Las funciones básicas que realiza este módulo son:

- Eliminar aquellos bounding box del objeto detectado que se solapen con un bounding box anteriormente detectado aportando información redundante (Ver figura 5.15).

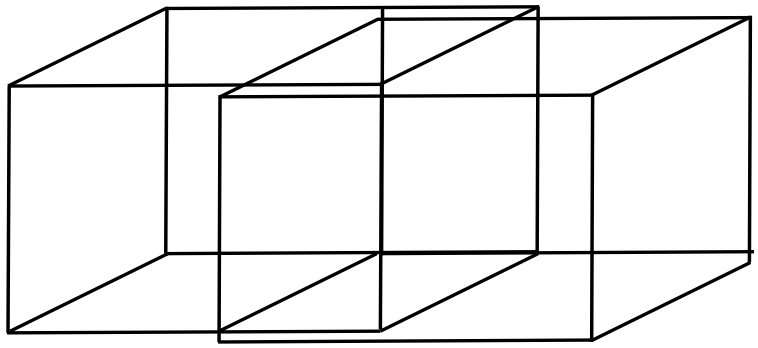

FIGURA 5.15: Representación de dos bounding box solapándose.

Para la eliminación de aquellos bounding box que se encuentran en solapamiento es condición necesaria conocer si un bounding box detectado recientemente solapa a la representación de otro bounding box. Para conocer si un bounding box interseca a otro se ha creado una función cuyo pseudocódigo es:

En definitiva un nuevo bounding box detectado recientemente no será almacenado nunca si solapa a otro bounding box generado por el mismo objeto en un instante de tiempo anterior.

Algoritmo 7 Intersección entre dos bounding box.

Entrada: Dos bounding box formados por ocho vértices cada uno $B1(V0,V1,V2,V3,V4,V5,V6,V7)$ y $B2(V0,V1,V2,V3,V4,V5,V6,V7)$. Cada vértice es un punto en el espacio 3D de la forma $V = (V_x, V_y, V_z)$.

Salida: **cierto** Si existe intersección entre el bounding box $B1$ y el bounding box $B2'$ y **falso** en caso de que no exista interesección entre B_1 y B_2.

para $B1$ y $B2$ **hacer**
  Obtener el valor mínimo del eje de coordenadas X.
  Obtener el valor máximo del eje de coordenadas X.
  Obtener el valor mínimo del eje de coordenadas Y.
  Obtener el valor máximo del eje de coordenadas Y.
  Obtener el valor mínimo del eje de coordenadas Z.
  Obtener el valor máximo del eje de coordenadas Z.
  si $((B1_{maxX} < B2_{minX})$ o $(B1_{minX} > B2_{maxX}))$ o $((B1_{maxY} < B2_{min_Y})$ o $(B1_{minX} > B2_{maxY}))$ o $((B1_{maxZ} < B2_{min_Z})$ o$(B1_{minZ} > B2_{maxZ}))$ **entonces**
    devolver falso
  fin si
fin para
devolver cierto

- Eliminar los bounding box que se encuentran en la misma dirección, con el objetivo de eliminar bounding box innecesarios, permitiendo que a la hora de representar el movimiento de un objeto, éste sea representado con el menor número de bounding box posibles (Ver figura 5.16).

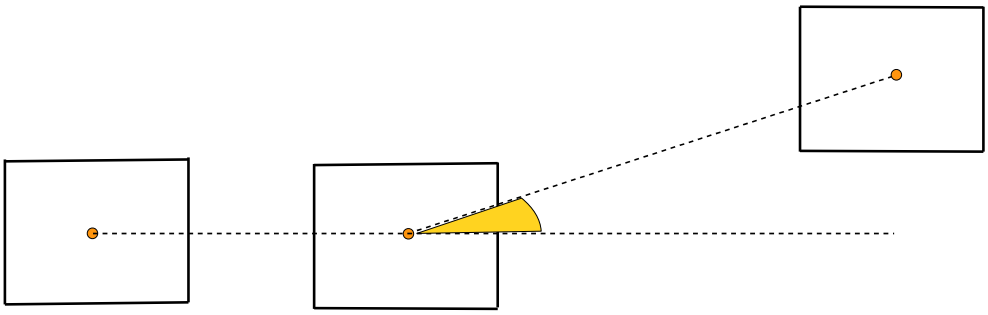

FIGURA 5.16: Representación del ángulo de desviación de dos bounding box.

Para resolver esto se ha utilizado un mecanismo que hace uso de las propiedades de los vectores situados en un espacio euclidiano 3D. Para ello, para una primera detección del objeto se almacenan todas sus características, incluida la posición del objeto en ese instante de tiempo. Posteriormente es detectado la segunda vez, en caso de que el bounding box que envuelve al objeto detectado se esté solapando con el

primer bounding box reconocido y almacenado, éste es desechado. Sin embargo, en caso de que el bounding box detectado no se solape con el bounding box inicialmente almacenado, entonces se guarda también en la base de conocimiento de objetos detectados. Una vez que en la base de conocimiento de objetos detectados se han guardado los datos correspondientes a dos posiciones diferentes del movimiento del objeto a lo largo de la escena, se calcula el punto central de los dos bounding box almacenado y se genera el vector dirección entre ambos puntos. Con esto se obtiene la dirección del movimiento que realiza el objeto.

Para calcular el punto central de un bounding box se realiza de la siguiente manera (ver figura 5.17).

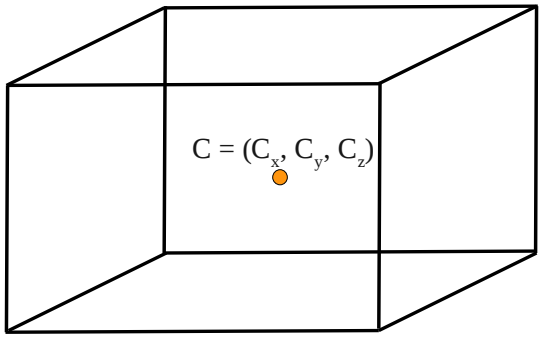

FIGURA 5.17: Representación del punto central de un bounding box.

El algoritmo en pseudocódigo del cálculo del punto central de un bounding box es:

Después de calcular el punto central de los dos bounding box se calcula el vector dirección entre estos dos puntos (ver figura 5.18).

Para calcular el vector dirección de dos puntos determinados se ha utilizado el procedimiento explicado en el apartado 3.2.3. Posteriormente se calcula el vector unitario resultante del vector anterior tal y como se especifica en la sección 3.2.3, con el objetivo de normalizar los resultados obtenidos en el cálculo del vector.

Una vez que el movimiento que se han almacenado dos posiciones del objeto de su trayectoria a lo largo de la escena, las siguientes detecciones del objeto en la escena observada son sometidas al mismo procedimiento, es decir, en primer lugar se calcula si el bounding box recientemente detectado no se encuentra solapando al bounding box clave anteriormente almacenado, se calcula la nueva dirección que lleva el objeto.

Algoritmo 8 Calcular punto central de un bounding box.

Entrada: Dado un bounding box $B1(V0,V1,V2,V3,V4,V5,V6,V7)$. Cada vértice es un punto en el espacio 3D de la forma $V = (V_x, V_y, V_z)$.

Salida: Un punto en el espacio euclídeo de la forma $C = (C_x, C_y, C_z)$.

Obtener el valor mínimo del eje de coordenadas X, min_x.
Obtener el valor máximo del eje de coordenadas X. max_x
Obtener el valor mínimo del eje de coordenadas Y. min_y
Obtener el valor máximo del eje de coordenadas Y. max_y
Obtener el valor mínimo del eje de coordenadas Z. min_z
Obtener el valor máximo del eje de coordenadas Z. max_z
Se calcula la coordenada X del punto central de $B1$ $C_x = min_x + |max_x - min_x|/2$
Se calcula la coordenada Y del punto central de $B1$ $C_y = min_y + |max_y - min_y|/2$
Se calcula la coordenada Z del punto central de $B1$ $C_z = min_z + |max_z - min_z|/2$
devolver Punto central del bounding box de la forma $C = (C_x, C_y, C_z)$.

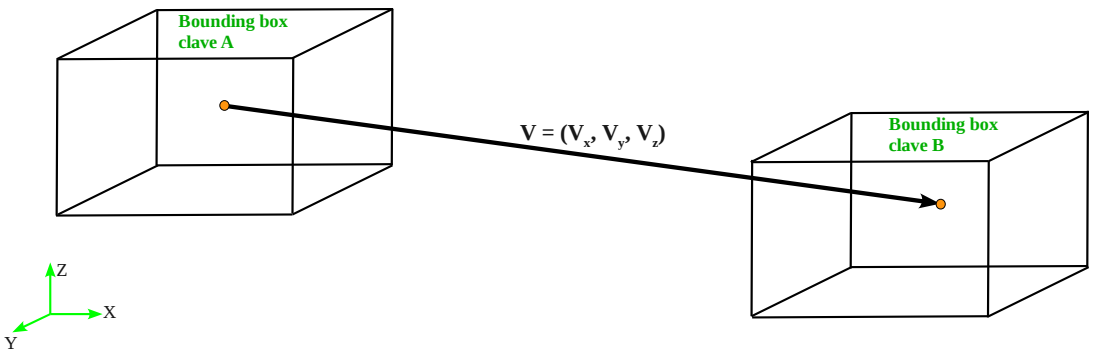

FIGURA 5.18: Representación del vector dirección de dos bounding box clave.

Esto se realiza calculando el vector dirección desde el punto 3D situado en el centro del último bounding box clave almacenado hasta el punto 3D central al bounding box recientemente reconocido. Posteriormente, se obtiene el vector unitario y se realiza una comparación con el vector unitario anteriormente calculado, con el objetivo de calcular la variación existente del nuevo vector, con respecto al vector antiguo. Para calcular la variación de distancia se realiza la diferencia de los valores con respecto a los tres ejes de coordenadas en valor absoluto. Después se realiza la suma de la diferencia en cada eje y si el valor obtenido es superior a un valor umbral predefinido entonces es considerado un **bounding box básico o clave** ya que tiene información importante para la representación del movimiento del objeto.

El pseudocódigo de esta función es el siguiente:

Algoritmo 9 Detección de un bounding box clave.

Entrada: Dados dos vectores $V = (V_x, V_y, V_z)$ y $U = (U_x, U_y, U_z)$.

Salida: **falso** si la diferencia es superior al umbral predefinido, **cierto** si la diferencia no es suficiente.

$|V_x - U_x|$
$|V_y - U_y|$
$|V_z - U_z|$

Se suman todos los valores absolutos de las tres componentes anterioremente calculados.

devolver **falso** si la suma de todos los valores supera un umbral predefinido y **cierto** si no lo supera.

Por tanto, la forma de operar del **submódulo de optimización de trayectorias** para el reconocimiento de una trayectoria es el siguiente:

1. El objeto es detectado por primera vez en el entorno monitorizado. Éste es representado por un bounding box que envuelve la silueta del objeto. Inicialmente, la primera detección del objeto es almacenada sin ninguna restricción. Este bounding se convierte en el primer bounding box clave para la representación final de la trayectoria.

2. El objeto es detectado de nuevo, en este caso es importante comprobar si el bounding box envolvente al objeto presenta algún punto de intersección con el último bounding box clave almacenado. En caso afirmativo, el bounding box es eliminado ya que no representa información lo suficientemente importante como para ser almacenado. En caso negativo, es necesario comprobar el número de bounding box clave almacenados. Si sólo hay un bounding box clave almacenado de la trayectoria del objeto, significa que no se conoce aún la dirección que lleva el movimiento del objeto, entonces el bounding box detectado es almacenado, para definir la dirección del movimiento del objeto.

3. Una vez que la trayectoria del objeto cuenta con dos bounding box clave y se conoce la dirección que lleva, las siguientes detecciones del objeto se realiza una comprobación de la dirección del objeto. En esta comprobación se calcula la nueva dirección que toma el objeto con respecto a la dirección anteriormente obtenida. En caso de que exista un cambio brusco en la dirección del movimiento del objeto, se considera de gran importancia almacenar el bounding box detectado con toda la información. En caso de que la dirección del objeto no presente un cambio muy brusco el bounding box recién detectado será eliminado. Es importante remarcar que la eliminación del bounding box no se realiza en el momento en que se comprueba su

condición, sino que es eliminado a posteriori. Es decir, sólo se eliminaría el bounding box recién detectado en caso de que se detecte el objeto posteriormente. De esta forma, en caso de que no se detectasen ninguna ocurrencia más del objeto en la escena observada, se puede obtener la última detección del objeto, aportando mayor precisión en la representación de la trayectoria 3D.

Algoritmo 10 Generación optimización del recorrido

Entrada: Dado un conjunto de bounding box correspondiente al movimiento de diferentes objetos en la escena de la forma $B1(V0,V1,V2,V3,V4,V5,V6,V7)$. Cada vértice es un punto en el espacio 3D de la forma $V = (V_x, V_y, V_z)$.

Salida: Una lista de bounding box que representan el recorrido del objeto de la forma $recorrido = B0,B1,B2,..,BN$.

para bounding box detectado B **hacer**
  Reconocer el objeto detectados.
  si El objeto existe **entonces**
    si Si es el segundo BB detectado **entonces**
      si Si No se produce intersección con el BB anterior **entonces**
        Almacenar el BB y el resto de propiedades
        Calcular la dirección del objeto.
      si no
        Eliminar el BB.
      fin si
    si no
      si Si no se produce intersección con el BB anterior. **entonces**
        Cálculo de la nueva dirección del BB detectado.
        si Si la nueva dirección varía de forma superior al umbral. **entonces**
          Almacenar el BB y el resto de propiedades.
          Actualizar la dirección del objeto.
        si no
          Eliminar el BB.
        fin si
      si no
        Eliminar el BB.
      fin si
    fin si
  si no
    Crear el objeto de nuevo, almacenar BB y resto de propiedades.
  fin si
fin para
devolver Devolver la lista de BB que forman el recorrido.

De esta forma, sólo se almacenan las características necesarias del movimiento del objeto, aportando mayor eficiencia y agilidad en la representación de trayectorias en 3D.

Por tanto, el submódulo de optimización las trayectorias de los objetos recibe por parte del submódulo de representación de trayectoria 3D, el bounding box recién detectado y el resto de bounding box que componen la trayectoria hasta el momento, este módulo se encarga de comprobar las características de éstos como se ha explicado anteriormente y devuelve la trayectoria 3D del objeto compuesta por el mínimo número de bounding box clave posibles para su representación.

Submódulo de cálculo de velocidad.

Según la definición de velocidad especificada en la sección 5.2.4, podemos obtener la velocidad del movimiento llevada a cabo por un objeto determinado en cualquier punto. Para ello, únicamente es necesario obtener la posición 3D del objeto y el tiempo en el que es detectado el objeto en dos puntos diferentes.

De manera que la velocidad existente entre dos bounding box viene determinada por la siguiente ecuación 5.6:

$$\vec{v} = \frac{\Delta x}{\Delta t} = \frac{(x_2 - X_1)}{(t_2 - t_1)} \tag{5.6}$$

Donde la posición de los dos puntos pertenecen al punto central de cada uno de los dos bounding box, para tener una distancia real. Por ejemplo, dados dos bounding box como se pueden ver en la figura 5.18, se calculan los puntos centrales de ambos bounding box los cuales son $C1 = (C1_x, C1_y, C1_z)$ y $C2 = (C2_x, C2_y, C2_z)$ respectivamente. La primera posición del objeto se detectó en instante t_1 (Bounding box A) y la segunda posición del objeto tuvo lugar en el instante t_2 (Bounding box B) se puede calcular la velocidad que lleva el objeto en ese trayecto de la siguiente forma:

$$\vec{v} = \frac{(C2_x - C1_x) + (C2_y - C1_y) + (C2_z - C1_z)}{t_2 - t_1} \tag{5.7}$$

De esta forma se puede conocer la velocidad con la que se desplaza el objeto en la escena monitorizada. Conociendo la velocidad del objeto en los diferentes segmentos de tiempo, se pueden obtener posteriormente nuevos valores como la media de éste en determinadas zonas y calcular controlar la velocidad de los objetos en las diferentes zonas determinadas con el objetivo de dotar de flexibilidad al sistema a la hora de realizar un control más exhaustivo.

Este submódulo es utilizado cada vez que un bounding box es detectado, de forma que se calcula la velocidad existente entre ambos. Este cálculo es realizado en este submódulo y una vez calculada la velocidad entre dos bounding box se almacena en la trayectoria del objeto para su posterior utilización y representación.

Submódulo de representación de recorrido 3D

En este submódulo se realiza la representación por recorrido en 3D. En primer lugar, necesario realizar una definición formal de recorrido. **Recorrido** es definido como el espacio que se recorre o que ha de recorrer una persona o cosa[1]. Sin embargo, se va proceder a explicar la creación de un recorrido específicamente, ya que presenta alguna peculiaridades. La representación de un recorrido en *Atravel* puede definirse como la zona espacial ocupada por una trayectoria del objeto a lo largo del entorno monitorizado. De manera que representa de forma menos restrictiva de representación del movimiento de un objeto. El recorrido realizado por un objeto representa el espacio correspondiente a la representación de la trayectoria sin tener en cuenta pequeñas variaciones con respecto a una línea recta trazada desde la posición inicial del objeto hasta su posición final. Así pues, el recorrido de un objeto cuya trayectoria apenas varíe de acuerdo a una línea recta trazada desde su posición incial en la escena hasta su posición final, será representado como una trayectoria totalmente recta con respecto a la línea imaginaria trazada. Sólo aparecerán variaciones en la definición del recorrido de un objeto, cuándo su trayectoria presente un cambio brusco en su desplazamiento a lo largo del entorno de monitorización. Por *cambio brusco* se entiende toda aquella variación importante de la trayectoria del objeto con respecto a la línea imaginaria calculada desde la posición inicial del objeto detectado hasta su posición final.

A continuación se pueden comprobar las diferencias existentes entre trayectoria y recorrido:

- El recorrido de un objeto, recoge la información más general de la trayectoria realizada por un objeto. No es sensible a los cambios de dirección pequeños, aunque sí a los cambios de dirección bruscos.
- Al tener una información más general del movimiento del objeto, permite obtener mayor porcentaje de similitud entre trayectorias similares.
- Mientras que la trayectoria determina de manera más precisa la posición exacta de un objeto en un instante determinado, con una longitud determinada, el recorrido permite abstraerse de esta característica y obtener información más genérica acerca de la ruta seguida por el objeto en la escena observada.
- El recorrido de un objeto permite tener una aproximación de la ruta sin necesidad de un gran tiempo de entrenamiento.

[1] http://es.thefreedictionary.com/recorrido

El submódulo de representación de recorrido es utilizado desde la aparición de una nueva detección de un bounding box en el entorno observado. Es entonces cuando se actualiza el recorrido seguido por un objeto. Para la creación del recorrido de una trayectoria de un objeto recién obtenida y su representación es necesario realizar los siguientes pasos:

- Un objeto es detectado en la escena por primera vez, éste es almacenado y guardado en la base de conocimiento de objetos detectados. Posteriormente es detectado por segunda vez, aquí ahora se aplican las operaciones explicadas en la sección 5.2.4. Es decir se aligera la trayectoria del objeto para cada nueva detección del objeto en el entorno monitorizado. Cuando un objeto es detectado por segunda vez en la escena, se calcula la dirección del movimiento del objeto.

- Cuando se han detectado los dos primeros bounding box clave y se conoce la dirección del objeto, se calcula el recorrido definido por el objeto hasta el momento. Como sólo está formado por dos bounding box se genera un recorrido para esos dos bounding box.

- Posteriormente pueden llegar nuevas detecciones del objeto en cuestión, entonces se calcula si existe un cambio muy brusco en la dirección realizada por el objeto. La detección de un cambio brusco viene determinada por el cálculo de un umbral, el cuál se explicará posteriormente.

- Por cada nueva detección del objeto se calcula la nueva dirección a la que se desplaza en la escena monitorizada. En caso de que ésta no presente ningún cambio brusco en la dirección a la que se desplazaba en frames anteriores, se elimina el antiguo recorrido establecido y se calcula un nuevo recorrido para que éste sea lo más limpio y preciso posible. El nuevo recorrido es realizado desde el último cambio brusco obtenido en el movimiento del objeto en la escena.

- En caso de que tras la detección de un objeto existe un cambio importante en la representación de la dirección del objeto, no se realiza el cálculo de un nuevo recorrido entero sino que se calcula sólo a partir del último cambio brusco obtenido en la dirección del objeto.

- Así el recorrido de un objeto que se desplaza por la escena, viene determinado por una serie de bounding box. El número de bounding box clave viene determinado por el número de cambios bruscos en el movimiento del objeto. Así pues, la trayectoria de un objeto que presente tres cambios bruscos a lo largo de su recorrido el número de bounding box final será de cinco bounding box. Ya que además de los bounding box

clave hay que tener el cuenta los bounding box intermedios a los bounding box(BB) clave.

El pseudocódigo del algoritmo utilizado para la creación del recorrido realizado por cada objeto es el siguiente:

La creación del recorrido de un objeto a lo largo del entorno observado viene determinado por el algoritmo anterior. *Atravel* tras recibir información precedente del archivo resultante del tracking, en el cuál se obtiene la información procedente de la detección de cada objeto que aparece en el entorno monitorizado (ver sección 5.1). Cada objeto detectado es envuelto con un bounding box que representa el tamaño del objeto y su posición en un determinado instante de tiempo. Así pues en cada instante es posible que aparezcan en la escena diferentes objetos en distinta posición. Para la generación del recorrido sobre el movimiento realizado por un objeto se ha utilizado el algoritmo 5. Este algoritmo puede detallarse de la siguiente forma:

- Inicialmente se obtienen los objetos que se están desplazando sobre la escena, cada uno de ellos es detectado e identificado. Una vez obtenidas las características del objeto en cada instante de tiempo diferente, el sistema se encarga de identificar la existencia anterior del objeto, es decir, puede ocurrir que el objeto detectado sea la primera vez que es detectado en la escena monitorizada. O por el contrario que ya haya sido detectado con anterioridad. En caso de que no haya sido detectado nunca en el entorno monitorizado, es añadido a la lista de objetos con las características detectadas de éste.

- En caso de que el objeto reconocido ya hubiese sido reconocido anteriormente, es necesario diferenciar entre la segunda detección del objeto y el resto de detecciones del objeto a lo largo del entorno monitorizado.

- Si es la segunda vez que es detectado, se comprueba si el bounding box que envuelve al objeto interseca directamente con el primer bounding box de la primera detección del objeto (véase la figura 5.15), para realizar esta comprobación se ha utilizado el algoritmo 1. En caso de que exista solapamiento entre ambos, la detección del objeto es eliminada, por lo que la lista de objetos detectados sólo está formada por un objeto, hasta una próxima detección del objeto. En caso de que no se produzca solapamiento entre los dos bounding box que envuelven la detección de los dos objetos, entonces el objeto es almacenado en la lista de objetos detectados con todas sus características. Además al tener en la lista de objetos detectados dos objetos es posible calcular la dirección que lleva el movimiento del objeto a lo largo de la escena observada, de

Algoritmo 11 Generación del recorrido de un objeto.

Entrada: Dado un conjunto de bounding box correspondiente al movimiento de diferentes objetos en la escena de la forma $B1(V0,V1,V2,V3,V4,V5,V6,V7)$. Cada vértice es un punto en el espacio 3D de la forma $V = (V_x, V_y, V_z)$.

Salida: Una lista de bounding box que representan el recorrido del objeto de la forma $recorrido = B0,B1,B2,..,BN$.

para bounding box detectado B **hacer**
  Reconocer los objetos detectados.
  si El objeto existe **entonces**
    sdfg
    si Si el número de detecciones del objeto es mayor que uno **entonces**
      sdfg
      si Si No se produce intersección con el BB anterior en la lista. **entonces**
        Almacenar el objeto recién detectado en la lista de trayectoria.
        Calcular la dirección de la trayectoria del objeto.
        Calcular dirección del recorrido.
        Calcular objeto intermedio entre ambos para la trayectoria del objeto.
        si Existe un cambio superior al umbral de recorrido **entonces**
          Almacenar como objeto importante.
          Calcular objeto intermedio entre el primer objeto y el recién almacenado.
        si no
          Recalcular el recorrido de la trayectoria.
        fin si
      si no
        Eliminar el BB.
      fin si
    si no
      Ver algoritmo 6
    fin si
  si no
    Crear el objeto de nuevo, almacenar en la lista de objetos detectados para trayectoria.

    Almacenar el objeto detectado en la lista del recorrido del objeto.
  fin si
fin para
devolver Devolver la lista de BB que forman el recorrido.

Algoritmo 12 Continuación del algoritmo *Generación del recorrido de un objeto*.

si Si el número de detecciones del objeto es mayor que dos. **entonces**
  si Si no se produce intersección con el BB anterior. **entonces**
    Cálculo de la nueva dirección del BB detectado.
    si Si la nueva dirección varía de forma superior al umbral. **entonces**
      Almacenar el objeto detectado en la lista de trayectoria.
      Actualizar la dirección del objeto.
    si no, **si** Si el número de objetos de la lista de recorrido es superior a uno **entonces**
      si La diferencia en la dirección del nuevo objeto detectado es superior a un umbral **entonces**
        Almacenar el nuevo objeto detectado en la lista de recorrido.
        Generar el objeto intermedio del recorrido.
        Almacenar el objeto intermedio.
      si no
        Recalcular el último objeto hasta el último objeto imprescindible situado en la lista de objetos de recorrido.
      fin si
    fin si
  si no
    Eliminar el BB.
  fin si
si no
  Eliminar el BB.
fin si

manera que es calculada (véase la figura 5.18). La función utilizada para calcular la dirección es algoritmo 3.

- En caso de que el objeto sea como mínimo la tercera vez que es reconocido, es decir, que la longitud de la lista de objetos detectados tenga una longitud de dos o más elementos. En este caso se comprueba si existe solapamiento con los bounding box anteriores y además se calcula la nueva dirección a la que se desplaza el objeto en la nueva detección realizada. Para ello, se calcula el vector dirección desde el objeto recién detectado y el último objeto almacenado en la lista del número de veces en que el objeto es detectado. Si el nuevo objeto detectado no se solapa con el objeto detectado anterior y además no lleva la misma dirección, sino que realiza un cambio brusco en la dirección de su trayectoria, es almacenado en la lista de objetos detectados, ya que se trata de una detección que refleja información determinante para la representación de la trayectoria y la velocidad del objeto en su movimiento a lo largo de la escena. En caso de que exista solapamiento entre el nuevo objeto recién detectado con el objeto anteriormente detectado o bien en el cálculo de la dirección del nuevo objeto con respecto al anterior se comprueba que la dirección de objeto apenas varía, entonces en este caso el objeto es eliminado, debido a que se considera que no aporta información relevante para la representación de su trayectoria.

- En caso de que la nueva detección pase a formar parte de la lista de objetos detectados, se comienza a comprobar condiciones para realizar el cálculo del recorrido del objeto a lo largo de la escena. Para ello, se ha utilizado una lista de objetos. Cada objeto contiene las características necesarias para la creación del recorrido del objeto en cada instante de tiempo. Cuando se detecta el objeto por primera vez éste es almacenado tanto en la lista de objetos (en realidad se detecta el mismo objeto, pero en diferentes posiciones de la escena) de la trayectoria como en la lista de objetos del recorrido. Esto se debe a que cómo se trata de la primera detección es necesaria su almacenamiento para comenzar a realizar cálculos utilizando los datos proporcionados por éste.

- Posteriormente cuando se produce una segunda detección del objeto en diferente posición, se comprueba que el bounding box que envuelve al objeto detectado interseca con el bounding box que envuelve al objeto almacenado anteriormente (es decir el primer objeto de la lista). En caso que exista colisión entre ambos, el objeto recién detectado es eliminado, debido a que está demasiado pegado al objeto almacenado anteriormente y no presenta información lo suficientemente relevante. En caso no existir colisión (ver algoritmo 1) con el objeto almacenado en la lista de

objetos que forman el recorrido, se comprueba el vector dirección existente entre el nuevo objeto recién detectado y el objeto anteriormente almacenado y si la diferencia existente es superior a un umbral determinado, entonces éste nuevo objeto detectado es considerado de gran relevancia en la representación del recorrido del objeto. Para calcular la dirección del objeto se calcula siguiendo el algoritmo 3. Si el nuevo objeto es almacenado, entonces es necesario calcular el objeto intermedio existente entre el antiguo bounding box detectado y el nuevo objeto recién detectado.

- En caso de que el objeto recién detectado no presente un cambio de dirección los suficientemente determinante como para ser almacenado en la lista de objetos que forman el recorrido, será eliminado y se procederá a recalcular el recorrido hasta el último cambio brusco almacenado con anterioridad. Para recalcular el recorrido se hará uso del algoritmo 7.

De esta forma, se obtiene el recorrido del movimiento de un objeto sobre la escena monitorizada. Con la obtención del recorrido obtenemos una trayectoria menos estricta o más permisiva para la representación de la trayectoria de un objeto que permite alcanzar posteriormente un mayor grado de similitud entre las diferentes trayectorias sin tener en cuenta pequeñas rectificaciones en la representación de las trayectorias de los objetos y sólo calculando los cambios bruscos en la representación de la trayectoria de los objetos.

El objetivo de realizar una representación del recorrido de la trayectoria es poner en relieve la trayectoria realizada por el desplazamiento del objeto a lo largo de la escena representando el mínimo número de pequeños cambios de dirección para obtener un aprendizaje acelerado de las rutas o patrones finales obtenido tras un período de monitorización de la escena. Es decir, con la utilización del recorrido de una trayectoria, se puede obtener una visión más general acerca de las rutas o patrones finales que se pueden obtener tras una gran cantidad de horas de monitorización del entorno. Por ejemplo, en un parque existe un paso que está rodeado de setos, donde aparecen movimiento de niños jugando, o paseando en bici, con la introducción de recorrido se obtiene que las acciones realizadas en dicho paso son normales que que la descripción del recorrido de la escena abarca todo el paso del parque, mientras que con la utilización de trayectorias únicamente, sería más dificil clasificar cada acción como normales y necesitaría mayor tiempo de aprendizaje para lograr una clasificación parecida a la de recorrido.

A continuación se van a mostrar algunos ejemplos, que resultan de la utilización del algoritmo de recorrido y su representación tras el reconocimiento de la trayectoria de un objeto que se desplaza sobre la escena.

Algoritmo 13 Obtención parcial del recorrido de un objeto.

Entrada: Dado una lista de bounding box correspondientes al movimiento sin ningún cambio brusco de un objeto en la escena de la forma $B1(V0,V1,V2,V3,V4,V5,V6,V7)$. Cada vértice es un punto en el espacio 3D de la forma $V = (V_x, V_y, V_z)$.

Salida: Bounding box que representan el recorrido del objeto en este trayecto $recorrido = V0,V1,V2,V3,V4,V5,V6,V7$.

Obtener el valor máximo en el eje de coordenadas X de todos los bounding box que forman la lista de bounding box.

Obtener el valor máximo en el eje de coordenadas Y de todos los bounding box que forman la lista de bounding box.

Obtener el valor máximo en el eje de coordenadas Z de todos los bounding box que forman la lista de bounding box.

Obtener el valor mínimo en el eje de coordenadas X de todos los bounding box que forman la lista de bounding box.

Obtener el valor mínimo en el eje de coordenadas Y de todos los bounding box que forman la lista de bounding box.

Obtener el valor mínimo en el eje de coordenadas Z de todos los bounding box que forman la lista de bounding box.

Crear $V0$ del bounding box resultante, para crear este vértice se obtiene el máximo X, el mínimo en Y y el mínimo en Z.

Crear $V1$ del bounding box resultante, para crear este vértice se obtiene el máximo X, el máximo en Y y el mínimo en Z.

Crear $V2$ del bounding box resultante, para crear este vértice se obtiene el mínimo X, el máximo en Y y el mínimo en Z.

Crear $V3$ del bounding box resultante, para crear este vértice se obtiene el mínimo X, el mínimo en Y y el mínimo en Z.

Crear $V4$ del bounding box resultante, para crear este vértice se obtiene el máximo X, el mínimo en Y y el máximo en Z.

Crear $V5$ del bounding box resultante, para crear este vértice se obtiene el máximo X, el máximo en Y y el máximo en Z.

Crear $V6$ del bounding box resultante, para crear este vértice se obtiene el mínimo X, el máximo en Y y el máximo en Z.

Crear $V7$ del bounding box resultante, para crear este vértice se obtiene el mínimo X, el mínimo en Y y el máximo en Z.

Crear bounding box resultante $recorrido = V0,V1,V2,V3,V4,V5,V6,V7$.

La construcción del recorrido de un objeto lleva consigo diferentes fases que se han podido observar en algoritmo 5. En primer lugar tras la detección del objeto por primera vez, no existe diferencia entre la lista de objetos de recorrido y la lista de objetos de trayectoria, ya que aún no se han obtenido suficientes datos sobre el objeto como para diferenciar entre su trayectoria y su recorrido.

Tras la detección y almacenamiento del segundo objeto detectado perteneciente al objeto, se produce una modificación en la representación de la trayectoria y el recorrido ya que siguiendo el algoritmo 7 se obtiene el recorrido formado por dos detecciones del objeto (ver figura B.3).

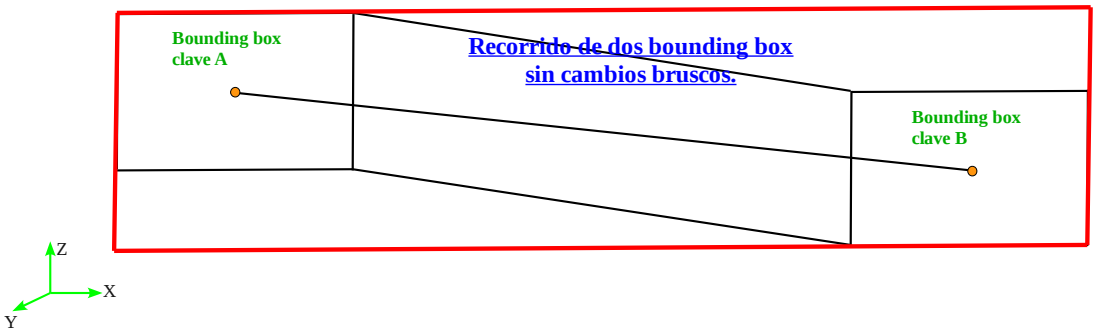

FIGURA 5.19: Representación del recorrido de un objeto (ver en Anexo B imagen a color).

Como se puede apreciar en la figura B.3 el rectángulo que engloba a los BB representa el recorrido obtenido utilizando el algoritmo 7 con las dos detecciones de los objeto actuales. De color negro podemos observar los dos bounding box que representan la posición del objeto en diferentes instante de tiempo y el bounding box que une a ambos para representar espacialmente el hueco entre las dos detecciones y obtener una trayectoria continua. En esta imagen se puede observar la parte inicial de una trayectoria y el modo de obtener el recorrido de un objeto cuando sólo se ha detectado dos veces.

También puede ocurrir que un objeto se desplace a lo largo del entorno observado y no existe un cambio brusco en su desplazamiento. En este caso la representación y generación del recorrido del objeto viene definida por un sólo bounding box que representa todo el recorrido del objeto. En la figura B.6 aparece representada la trayectoria del objeto que representa el desplazamiento del objeto a lo largo del entorno monitorizado de color negro, y en color rojo se puede observar el bounding box que representa el recorrido del objeto tras su trayectoria. En este caso al no existir un cambio lo suficientemente importante con

respecto a la primera detección del objeto en la escena es representado como un único objeto con un bounding box que abarca el largo y ancho de la trayectoria.

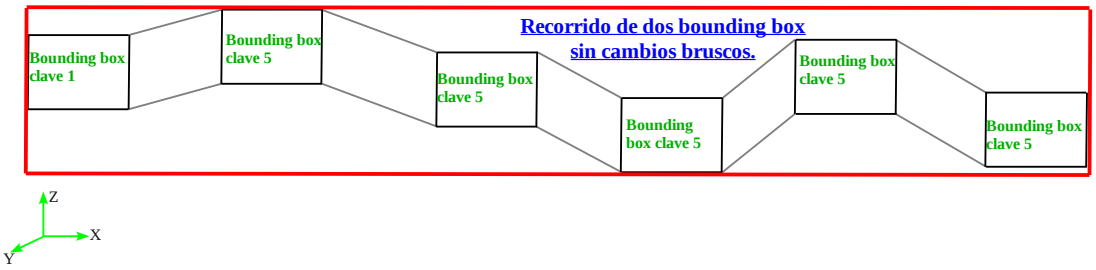

FIGURA 5.20: Representación del recorrido completo de un objeto sin realizar cambios importantes en su desplazamiento (ver imagen a color en el Anexo B).

Otro caso que puede ocurrir es que el objeto se desplace en la escena generando cambios importantes en la representación de su trayectoria. En este caso el recorrido del objeto no puede ser representado por un único bounding box que envuelva y represente toda la trayectoria del objeto. En la figura B.7 se puede ver un ejemplo de este caso. En el cual un objeto puede realizar un movimiento muy variante en cuanto a la dirección que lleva a lo largo de la escena. En este caso es necesario representar los cambios en la representación del recorrido del objeto.

En este ejemplo, la trayectoria del objeto viene representada de color negro, y se pueden diferenciar los distintos bounding box clave que representan el movimiento el objeto a lo largo de la escena. Mientras que el recorrido del objeto viene representado por el conjunto de bounding box dibujados de color rojo, de forma que cuando existe un cambio de dirección importante (la diferencia de valores en sus coordenadas, supera un umbral predeterminado) se considera que existe un cambio importante en la representación del recorrido del objeto.

Como se ha podido observar en los ejemplos anteriores, se puede concluir que con la generación del recorrido del movimiento de los objetos a lo largo de la escena observada, se pierde precisión en la representación de la trayectoria del objeto, sin embargo, permite tener una noción del espacio que ocupando con su movimiento, lo cual es importante para tener una aproximación más veloz de los patrones aprendidos. Esto permite tener una aprendizaje acelerado de éstos sin necesidad de un gran número de horas de monitorización de la escena. La generación del recorrido permite tener una abstracción de la escena e identificar los patrones existentes de una forma rápida.

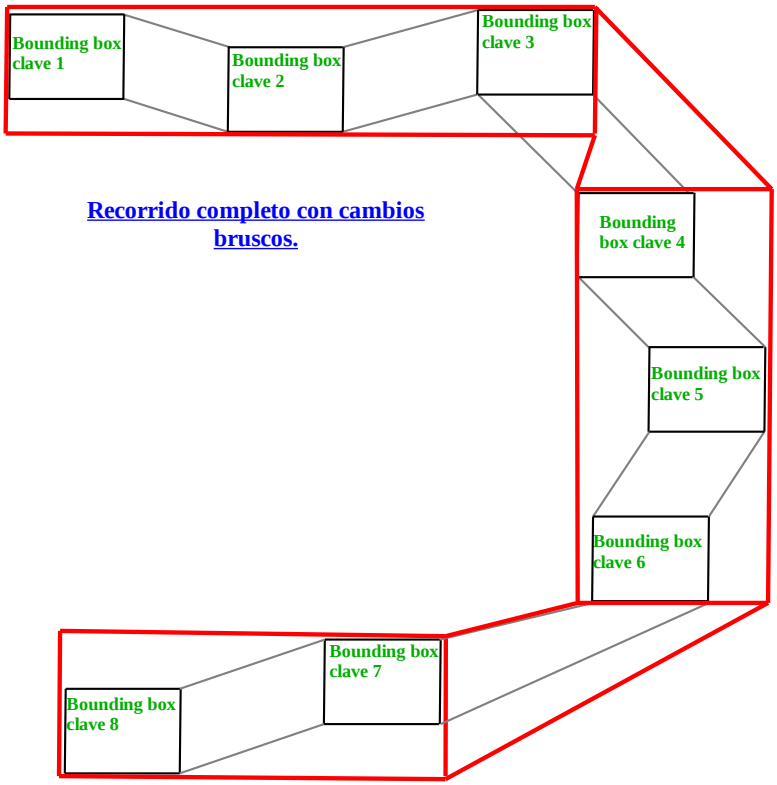

FIGURA 5.21: Representación del recorrido completo de un objeto realizando cambios importantes en su desplazamiento (ver imagen a color en el Anexo B).

Como se ha detallado anteriormente, el recorrido de un objeto viene determinado por una lista de objetos que representan el recorrido completo de la trayectoria realizada por los objetos en la escena. Los objetos que forman esta lista contienen toda la información acerca de la representación del recorrido (véase 5.2.2).

5.3. Submódulo de similitud por trayectoria.

La clasificación de un comportamiento atendiendo a su trayectoria, proporciona al sistema una información muy específica sobre el comportamiento de un objeto en la escena. La trayectoria de un objeto, es la información más valiosa e importante sobre el comportamiento de un objeto. Su correcto análisis proporciona a la aplicación gran información acerca del comportamiento del objeto.

En el *submódulo de similitud por trayectoria* se han elaborado dos algoritmos diferentes que permiten obtener dos visiones de similitud por trayectoria diferentes:

1. **Detección de cambios de dirección**: Permite establecer similitud entre dos trayectorias, tomando como principal argumento la similitud entre BB clave de dos trayectorias. La aparición de BB clave dentro de una trayectoria, implican cambios en la dirección del desplazamiento de un objeto. Por tanto, este algoritmo se centra establecer la similitud entre dos trayectorias comparando la similitud de sus BB clave. Por tanto, mediante la utilización de este algoritmo, dos objetos realizan la misma trayectoria si su trayectoria está formada por el mismo número de BB clave y además los BB clave correspondientes colisionan unos con otros. Se dice que dos BB son similares si y sólo si existe colisión entre ellos en algún punto del espacio. Este algoritmo sólo realiza comparación de similitud con BB clave, no realiza comparación con BB intermedios. Pseudocódigo del algoritmo de detección de cambios de dirección es:

   La explicación de este algoritmo es la siguiente. A medida que se va detectando la aparición de objetos en la escena, este algoritmo es activado para analizar la similitud del comportamiento de cada detección del objeto con respecto a los patrones existentes en la *base de conocimiento de objetos detectados*. Un objeto es detectado en el entorno de monitorización, la detección de éste implica realizar una serie de comprobaciones para detectar si se trata de un BB clave o no según se detalla en la sección 5.2.4. En caso de que no se trate de un BB clave, éste es eliminado. En caso de que si se trate de un BB clave, se activa este algoritmo y se analiza el grado de similitud de cada éste con cada uno de los patrones almacenados en la *base de conocimiento de patrones aprendidos*. Para calcular el grado de similitud se realiza con este algoritmo el cuál permite conocer la similitud existente entre BB clave de la trayectoria del objeto recientemente detectado con respecto a los BB clave de los patrones almacenados en la base de conocimiento. Para obtener el grado de similitud se chequea la existencia de colisión entre el BB clave del objeto detectado y el BB clave correspondiente de cada patrón de la *base de conocimiento de patrones*.

   Por tanto, para obtener el grado de similitud se comprueba si existe colisión entre BB. En caso de que el objeto no haya sido detectado anteriormente, se incorporan a la lista *list_similar_path* los identificadores de todos los patrones que se encuentran almacenados en la *base de conocimiento de patrones*. Posteriormente se calcula si existe colisión entre el BB clave del objeto recién detectado con el BB clave correspondiente de cada patrón. Si no existe colisión entre ellos, se incrementa la variable *num_discarted* en una unidad. El resultado de la colisión es

Algoritmo 14 Similitud atendiendo a los bounding box clave.

Entrada: Requiere la base de conocimiento de patrones aprendidos, el identificador del objeto recién detectado y la base de conocimiento de objetos detectados.

Salida: Porcentaje de similitud por trayectoria.

Se obtiene la posición del nuevo BB detectado por parte del objeto.
si Es la primera vez que se detecta el objeto **entonces**
Añadir a la lista de patrones con trayectorias similares todos los patrones.
para Cada uno de los patrones que hay en la lista de patrones con trayectoria similiares. **hacer**
Comprobar si el nuevo objeto detectado colisiona con el BB de cada patrón correspondiente.
Calcular el porcentaje de colisión entre el patrón y la trayectoria del objeto.
devolver porcentaje de similitud
fin para
si no
si Existe algún patrón nuevo que no haya sido incluido en la lista de patrones con trayectorias similares **entonces**
Se añade el identificador a la lista de patrones con trayectoria similar.
fin si
si La lista de patrones con trayectorias similares no está vacía **entonces**
para Cada uno de los patrones que forman la lista de patrones con trayectorias similares **hacer**
si El BB del objeto detectado tiene una posición inferior a la del patrón **entonces**
Calcular si existe colisión entre el BB del objeto recién detectado y el BB del patrón correspondiente
Almacenar resultado de la colisión.
Calcular porcentaje de similitud
si El número de veces que no ha habido colisión es mayor o igual a 2 **entonces**
Eliminar el patrón de la lista de patrones con trayectoria similares
Añadir el patrón a la lista de patrones descartados.
fin si
si no
Se elimina de la lista de patrones con trayectorias similares.
fin si
devolver porcentaje de similitud
fin para
fin si
fin si

almacenado en una lista denominada *BB_checked_vector* y a partir de los resultados de las comprobaciones de colisión entre BB se calcula el porcentaje de similitud, dependiendo del número de valores almacenados con valor *True* con respecto al número total de BB clave de los que esté formada la trayectoria. En caso de que el objeto detectado ya haya sido detectado anteriormente se calcula la similitud sólo con aquellos patrones con los que existe cierta similitud. Posteriormente se realiza la comprobación de colisión de los BB clave correspondientes. Se almacena el resultado en la lista *BB_checked_vector*. Tras el cálculo de la colisión se calcula el porcentaje de similitud por trayectoria. En caso de que el resultado de la colisión sea negativo, se incrementa en una unidad el número de fallos en la colisión. Si el número de fallos supera o iguala una variable determinada constante (en este caso es 2) significa que el patrón no tiene una alta similitud y éste es descartado y enviado a la lista *list_descarted_path*.

El diagrama de flujo correspondiente a este submódulo se puede ver en la figura 5.22:

2. **Obviando los cambios de dirección**:Este algoritmo proporciona un porcentaje de similitud entre dos trayectorias realizando comparaciones con bounding box clave y con bounding box intermedios. De manera que no se centra en la similitud de los cambios de dirección sino que realiza la comparación de similitud entre todos los BB que forman la trayectoria, tanto BB clave como BB intermedios. Para ello, el algoritmo compara inicialmente con el BB que le corresponde tras la una detección del objeto en la escena. Comprueba si hay colisión con éste. En caso de que no exista colisión con el BB clave correspondiente comprueba si existe colisión con los BB anteriores (si es posible, sino no) y con los BB posteriores(si es posible, sino no) hasta encontrar alguno de ellos en los que exista colisión. Además se almacena el BB con el que ha existido colisión y en la siguiente detección del objeto se comienza a realizar la comprobación desde éste. Pseudocódigo del algoritmo obviando los cambios de dirección del movimiento del objeto:

   Este algoritmo es similar al anterior, la diferencia que existe con el anterior es la forma de realizar la colisión con los BB. En este algoritmo por cada detección de un BB clave, se estudia la existencia de colisión entre el BB recién detectado y el BB clave correspondiente a cada patrón de la lista de de patrones con recorridos similares denominada *list_similar_travel*. La comprobación de un BB clave recién detectado se realiza mediante la comprobación con el último BB más dos posiciones con el que ha habido colisión con él cuya posición se encuentra almacenada en la variable *last_travel_touching*. Se calcula si existe colisión con ésta en caso de que el resultado de

Algoritmo 15 Similitud entre trayectorias atendiendo a todos sus BB.

Entrada: Requiere la base de conocimiento de patrones aprendidos, el identificador del objeto recién detectado y la base de conocimiento de objetos detectados.

Salida: Porcentaje de similitud por trayectoria.

```
Se obtiene la posición del nuevo BB detectado por parte del objeto.
si Es la primera vez que se detecta el objeto entonces
    Parte del pseudocódigo anterior.
si no
    si Lista de patrones con trayectoria similar no está vacía. entonces
        para Cada uno de los patrones que ocupan la lista de patrones con trayectoria similar
        hacer
            Calcular si existe colisión entre el BB recién detectado y el BB del patrón
            correspondiente
            si Existe colisión entonces
                Almacenar si existe colisión entre el BB de la trayectoria del objeto recién
                detectado con el BB de la trayectoria del patrón correspondiente.
                Calcular y almacenar el porcentaje de similitud de la trayectoria del objeto
                recién detectado con la trayectoria del patrón.
            si no
                si Existe el BB-1 en la trayectoria del patrón entonces
                    Calcular colisión entre BB del objeto y el BB-1 del patrón.
                    si Existe colisión entre el BB del objeto y el BB-1 del patrón entonces
                        Eliminar resultados de colisión anteriores.
                        Almacenar el resultado de la colisión.
                        Calcular y almacenar el porcentaje de similitud con los datos actuales de
                        existencia de colisión.
                    si no
                        Ver algoritmo 16
                    fin si
                si no
                    Aumentar el contador de fallo de colisión.
                    si Contador de fallo de colisión es mayor o igual a 2 entonces
                        Eliminar trayectoria.
                    fin si
                fin si
            fin si
        fin para
    fin si
fin si
```

Algoritmo 16 Continuación del algoritmo *Similitud entre trayectorias atendiendo a todos sus BB*.

si Existe BB+1 en la trayectoria del patrón **entonces**
  Calcular colisión entre BB del objeto y el BB+1 del patrón.
  si Existe colisión entre el BB del objeto y el BB+1 del patrón **entonces**
    Eliminar resultados de colisión anteriores.
    Almacenar el resultado de la colisión.
    Calcular y almacenar el porcentaje de similitud con los datos actuales de existencia de colisión.
  si no
    si Existe BB+2 en la trayectoria del patrón **entonces**
      Calcular colisión entre BB del objeto y el BB+2 del patrón.
      si Existe colisión entre el BB del objeto y el BB+2 del patrón **entonces**
        Eliminar resultados de colisión anteriores.
        Almacenar el resultado de la colisión.
        Calcular y almacenar el porcentaje de similitud con los datos actuales de existencia de colisión.
      si no
        Ver algoritmo 17
      fin si
    si no
      Aumentar el contador de fallo de colisión.
      si Contador de fallo de colisión es mayor o igual a 2 **entonces**
        Eliminar trayectoria.
      fin si
    fin si
  fin si
si no
  Aumentar el contador de fallo de colisión.
  si Contador de fallo de colisión es mayor o igual a 2 **entonces**
    Eliminar trayectoria.
  fin si
fin si

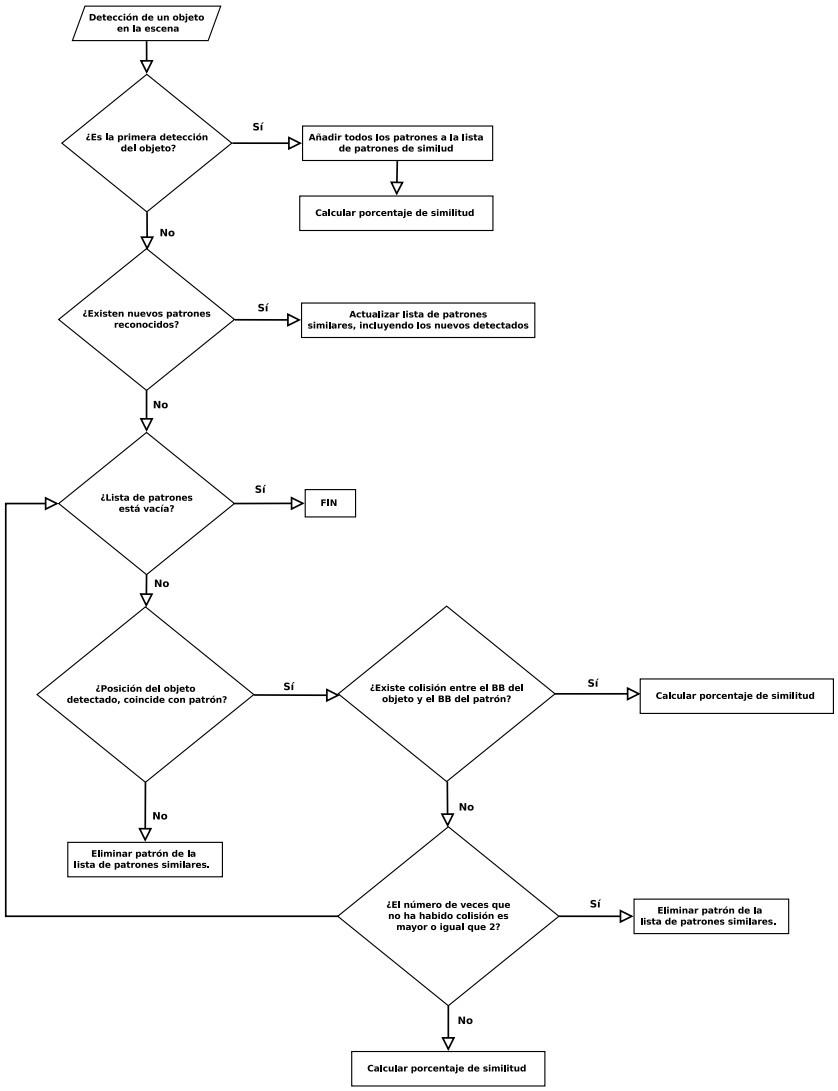

FIGURA 5.22: Diagrama de flujo correspondiente al submódulo de similitud por trayectoria.

la colisión sea positiva, entonces se calcula el porcentaje de similitud con éste patrón. En caso de que el resultado de la colisión sea negativo, se calcula la existencia de colisión con los dos BB anteriores y los dos BB posteriores al BB correspondiente del patrón. Si aún así, el resultado de colisión es negativo, entonces se incrementa la variable contadora del número de veces que no ha habido colisión. Si el número de veces que no ha existido colisión es mayor o igual que dos, entonces se considera se determina que dicho patrón no tiene similitud con el objeto que se desplaza a lo largo de la escena y es eliminado de la lista *list_similar_travel* y añadido en la lista *list_descarted_travel* que contiene los identificadores de los patrones con los que no existe similitud.

Algoritmo 17 Continuación 2 del algoritmo *Similitud entre trayectorias atendiendo a todos sus BB*.

```
si Existe BB1-2 en la trayectoria del patrón entonces
   Calcular colisión entre BB del objeto y el BB-2 del patrón.
   si Existe colisión entre el BB del objeto y el BB-2 del patrón entonces
      Eliminar resultados de colisión anteriores.
      Almacenar el resultado de la colisión.
      Calcular y almacenar el porcentaje de similitud con los datos actuales de existencia
      de colisión.
   si no
      Aumentar contador de fallo de colisión.
      si Contador de fallo de colisión es mayor o igual a 2 entonces
         Eliminar trayectoria.
      fin si
   fin si
si no
   Aumentar el contador de fallo de colisión.
   si Contador de fallo de colisión es mayor o igual a 2 entonces
      Eliminar trayectoria.
   fin si
fin si
```

5.4. Submódulo de similitud por recorrido

La fase de clasificación del comportamiento de los objetos que aparecen a lo largo de la escena es una de las tareas más importantes para llevar a cabo la detección de situaciones anómalas durante el período de monitorización. Establecida la similitud por trayectoria se ha pensado en utilizar la similitud por recorrido para enriquecer la clasificación de comportamientos de los objetos que se desplazan sobre la escena. La similitud por recorrido consiste en obtener le grado de similitud existente entre el recorrido realizado por un objeto con respecto al recorrido realizado por cada uno de los patrones que componen la *base de conocimiento de patrones aprendidos*. El objetivo de la creación de este submódulo es obtener aquellos patrones que tienen mayor grado de similitud utilizando el recorrido (véase la sección 5.2.4) ya que la similitud utilizando el recorrido permite obtener un mayor grado de similitud con patrones, al no se tan restrictivo y preciso que el grado de similitud por trayectoria.

Inicialmente si el objeto detectado en la escena es la primera vez, entonces se introducen todos los identificadores de los patrones de la base de datos en la lista denominada *list_travel*. A continuación, se calcula la similitud con cada uno de los patrones almacenados en esta lista. Se almacena el resultado de la similitud. La similitud en este caso consiste

en calcular la existencia o no de colisión del BB clave que forma el recorrido con el BB clave correspondiente del recorrido de cada patrón. Tanto si existe colisión como si no, el resultado es almacenado. En caso de que no exista colisión se incrementa en una unidad la variable de número de fallos en la colisión. Posteriormente, tras el resultado de la colisión también se calcula el porcentaje de similitud existente entre el patrón y el objeto detectado. En caso de que el objeto detectado en la escena ya haya sido detectado anteriormente, sólo de realizará la comprobación de similitud con aquellos patrones con los que existe similitud. Es decir, se calcula la existencia o no de colisión con aquellos patrones almacenados en la lista *list_travel* que contiene los identificadores de los patrones con los que existe similitud y se comprueba la existencia de colisión con el BB correspondiente. Si el resultado de la colisión es negativo, entonces se incrementa en una unidad el número de fallos de colisión. Tras la comprobación de colisión se calcula el grado de similitud con el patrón en concreto. A continuación, se chequea el número de fallos de colisión si si valor es de dos unidades o más, el patrón es descartado, ya que se considera que existe la similitud suficiente como para realizar la posterior fusión.

El pseudocódigo del algoritmo es el siguiente:

El diagrama de flujo correspondiente a este submódulo se puede ver en la figura 5.23:

5.5. Submódulo de similitud por velocidad

Para realizar una correcta clasificación del comportamiento de un objeto determinado con respecto a los patrones existentes en la *base de conocimiento de patrones aprendidos* es importante realizar una comprobación de la velocidad con aquellos patrones con los que exista una importante afinidad con respecto a su trayectoria y su recorrido. La comprobación de la velocidad de un objeto sobre la escena monitorizada tiene como objetivo informar al usuario de la excesiva o reducida velocidad con la que circula un determinado objeto con respecto a un patrón determinado. Si el objeto que aparece en la escena no tiene similitud con ningún patrón almacenado en la base, no informará sobre la velocidad de éste ya que este submódulo no se habrá activado. Así pues, sólo se activa este módulo si su trayectoria o recorrido es similar (en un grado superior al 80 %) a alguno de los patrones almacenados.

Inicialmente se detecta la posición de un objeto en la escena observada. La posición de este objeto es chequeada para comprobar si se trata de un BB clave o no (para ello, se estudia si existe solapamiento con BB anteriores, o si existe un cambio importante en la dirección del movimiento del objeto). En caso de ser un BB clave, se calcula la velocidad con respecto a otros BB anteriormente detectados, como se ha explicado en la sección 5.2.4.

Algoritmo 18 Similitud atendiendo al recorrido de un objeto.

Entrada: Requiere la base de conocimiento de patrones aprendidos, el identificador del objeto recién detectado y la base de conocimiento de objetos detectados.

Salida: Porcentaje de similitud por recorrido.

Se obtiene la posición del nuevo BB detectado por parte del objeto.
si Es la primera vez que se detecta el objeto **entonces**
Añadir a la lista de patrones con recorridos similares todos los patrones.
para Cada uno de los patrones que hay en la lista de recorridos similares. **hacer**
Comprobar si el nuevo objeto detectado colisiona con el BB del recorrido de cada patrón correspondiente.
Calcular el porcentaje de colisión entre el patrón y el recorrido del objeto.
devolver porcentaje de similitud por recorrido.
fin para
si no
si Existe algún patrón nuevo que no haya sido incluido en la lista de patrones con recorridos similares **entonces**
Se añade el identificador a la lista de patrones con recorrido similar.
fin si
si La lista de patrones con recorridos similares no está vacía **entonces**
para Cada uno de los patrones que forman la lista de patrones con recorridos similares **hacer**
si El BB del objeto detectado tiene una posición inferior que el BB del recorrido perteneciente al patrón **entonces**
Calcular si existe colisión entre el BB del objeto recién detectado y el BB del recorrido del patrón correspondiente
Almacenar resultado de la colisión.
Calcular porcentaje de similitud del recorrido
si El número de veces que no ha habido colisión es mayor o igual a 2 **entonces**
Eliminar el patrón de la lista de patrones con recorridos similares
Añadir el patrón a la lista de patrones descartados.
fin si
si no
Se elimina de la lista de patrones con recorridos similares.
fin si
devolver porcentaje de similitud por recorrido.
fin para
fin si
fin si

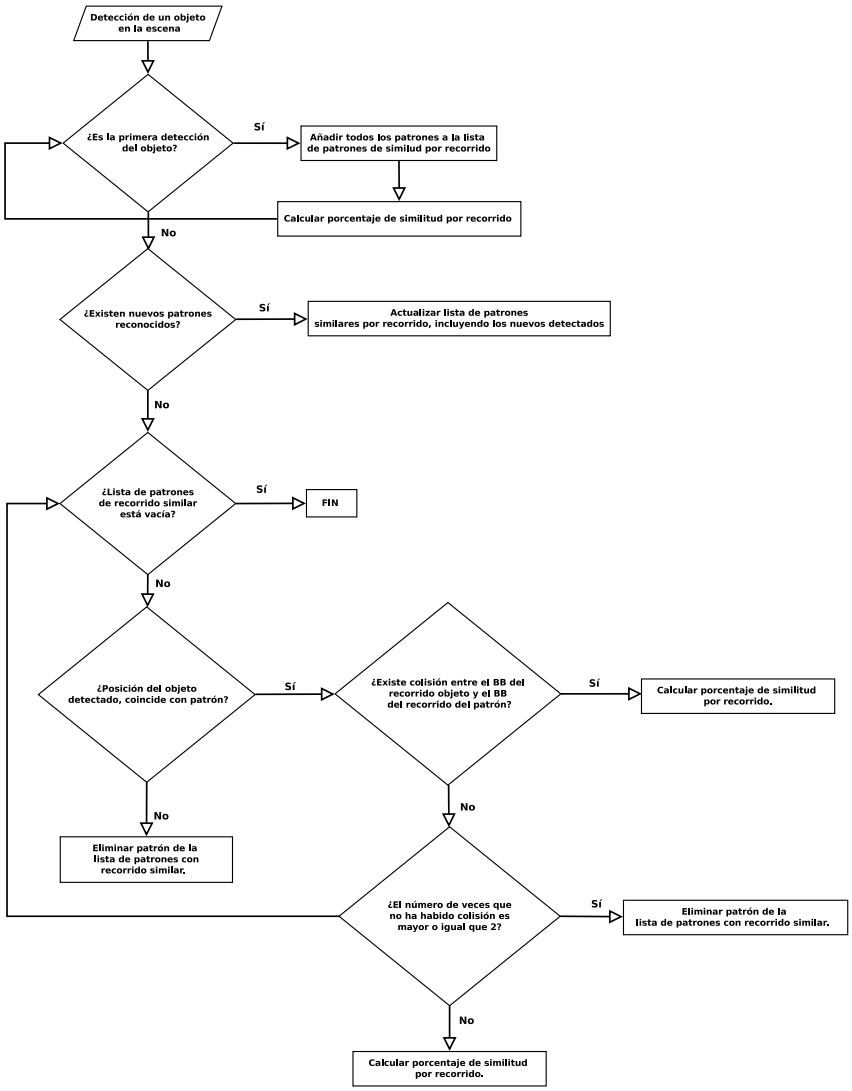

FIGURA 5.23: Diagrama de flujo correspondiente al submódulo de similitud por recorrido.

A continuación, se realiza la comprobación de similitud por trayectoria y recorrido. En caso de existir similitud con un BB clave de un determinado patrón se obtiene la velocidad en dicho fragmento y ésta es comparada con la velocidad con la que se desplaza en este fragmento el objeto recién detectado si ésta es similar entonces se almacena en la lista de resultados de comparación de velocidad el valor de *True* y si no es similar se almacena el valor de *False*. Posteriormente tras almacenar el valor de éste se calcula el porcentaje de similitud con dicho patrón. Para ello, se realiza una comparación de este vector en el que almacena los resultados de similitud por velocidad, obteniendo el número de fragmentos con los que el objeto se desplaza a velocidad similar con respecto a los fragmentos totales que constituyen el movimiento del objeto. En caso de que el objeto tenga una similitud con el patrón del 80 % y no exista una similitud en cuanto a la trayectoria al menos del 80 %

también, se mostrará en la aplicación un mensaje de advertencia, que informará al usuario del exceso o reducida velocidad con la que se desplaza a lo largo de la escena monitorizada.

A continuación, se detalla el pseudocódigo del algoritmo encargado de calcular la similitud por velocidad.

Algoritmo 19 Similitud atendiendo a la velocidad.

Entrada: Requiere la base de conocimiento de patrones aprendidos, el identificador del objeto recién detectado y la base de conocimiento de objetos detectados.
Salida: Porcentaje de similitud por trayectoria.

```
Se obtiene la posición del nuevo BB detectado por parte del objeto.
si Es la primera vez que se detecta el objeto entonces
    Añadir a la lista de patrones con trayectorias similares todos los patrones.
    para Cada uno de los patrones que hay en la lista de patrones con trayectoria similiares.
    hacer
        Obtener la velocidad en el patrón de la posición correspondiente.
        Comprobar si la velocidad del nuevo objeto detectado es similar con la existente en
        el fragmento de la trayectoria correspondiente.
        Almacenar el resultado de la comprobación
    fin para
si no
    si Existe algún patrón nuevo que no haya sido incluido en la lista de patrones con
    trayectorias similares entonces
        Se añade el identificador a la lista de patrones con trayectoria similar.
    fin si
    si La lista de patrones con trayectorias similares no está vacía entonces
        para Cada uno de los patrones que forman la lista de patrones con trayectorias
        similares hacer
            si El BB del objeto detectado tiene una posición inferior a la del patrón entonces
                Comprobar si la velocidad del nuevo objeto detectado es similar con la
                existente en el fragmento de la trayectoria correspondiente.
                Almacenar el resultado de la comprobación
            fin si
        fin para
    fin si
fin si
```

El diagrama de flujo correspondiente a este submódulo se puede ver en la figura 5.24:

5.6. Submódulo de fusión de patrones

Debido a la necesidad de dotar a la aplicación de la capacidad de aprenizaje automático. Es necesario la creación de un algoritmo que realice una fusión de patrones permita definir

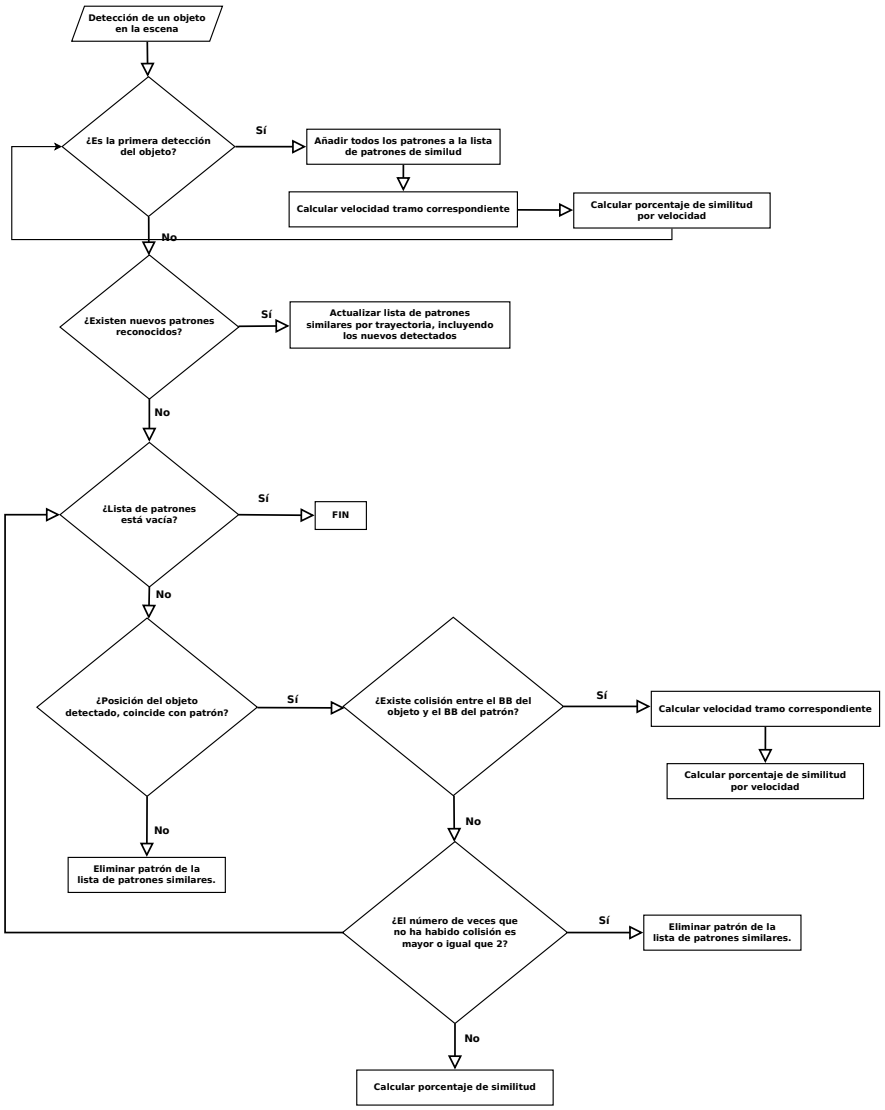

FIGURA 5.24: Diagrama de flujo correspondiente al submódulo de similitud por velocidad.

con gran precisión las distintas rutas existentes en la escena monitorizada. La fusión de patrones tiene como objetivo, dados dos patrones, los cuáles tienen un 90 % de similitud, realizando su fusión se obtiene un sólo patrón más preciso. La principal razón, es que tanto vehículos como peatones circulan por una determinada posición de la escena, pero es probable que alguna de ellas realice una pequeña variación en su desplazamiento. Esto no significa que en el 10 % que no existe similitud entre un objeto recién detectado y un patrón de la *base de conocimiento de patrones aprendidos* no realiza un desplazamiento correcto sobre la escena sino, al contrario. Es interpretado que al variar tan poco (sólo en un 10 %) con respecto al comportamiento del patrón, su desplazamiento es considerado similar o idéntico ya que puede ser que no haya realizado el mismo desplazamiento por que exista algún obstáculo físico en la escena, como puede ser un peatón, un objeto nuevo, etc.

La fusión sólo se va a realizar sobre aquellos patrones que tengan una porcentaje de similitud superior al 80 %. Esto es así por definición ya que ha sido establecido de este modo. Este porcentaje es configurable en caso de que sea necesario se puede definir un nuevo valor más adecuado, dependiendo de la concentración de objetos en la escena, el tiempo de aprendizaje necesario, etc. Dotar de esta capacidad de fusión entre patrones permite como se ha dicho anteriormente dotar de aprendizaje automático a *Atravel*. Aunque entre dos patrones no exista un 100 % de similitud se realiza la fusión de patrones, esto aporta flexibilidad al sistema. De modo, que si se produce una modificación en el entorno de monitorización, y ahora los objetos varían su desplazamiento en un pequeño porcentaje, la aplicación permite aprender de esto.

Además, la aplicación realiza función de aprendizaje, tras la existencia de un importante número de patrones que tienen el mismo comportamiento o similar, se realiza la fusión de éstos. En caso de que aparezca en la escena una modificación urbanística de ésta, no existe problema ya que el sistema se encuentra en aprendizaje continuo y aprende aquéllas nuevas rutas que aparezcan repetidas veces en la escena. Además se puede pensar, y las rutas antiguas cómo son eliminadas o marcadas como anormales. Pues *Atravel* cuenta con una ventana gráfica en la que un usuario puede modificar el comportamiento de uno o más patrones manualmente y pasar de tener un comportamiento *NORMAL* a *ABNORMAL*, o viceversa, además también puede modificarse manualmente si un patrón tiene un comportamiento *NOT_CLASSIFIED* puede pasar a modificarse por un comportamiento *NORMAL* o *ABNORMAL*.

El pseudocódigo del algoritmo utilizado para la creación de la fusión de patrones es el siguiente:

El algoritmo de fusión se realiza o se activa en el momento en el que un objeto abandona la escena, entonces como se ha explicado en la sección 5.2.3 se comprueba el número de patrones tiene similitud por encima del 80 %. Si el número de patrones es superior a cinco, entonces se activa el módulo de fusión de estos cinco patrones con el recién reconocido. Esto es así, ya que como los cinco tienen un alto grado de similitud se interpreta que todos ellos realizan un movimiento similar sobre la escena. A partir, de aquí se considera que el comportamiento de este patrón es *NORMAL*. En cualquier otra situación si el número de similitud es inferior a cinco, entonces no se activará el submódulo de fusión de patrones.

Una vez activado el submódulo de fusión de patrones se realizará pondrá el funcionamiento del algoritmo 20. Este consiste obtener un patrón que tenga una similitud con el objeto superior al 80 % y realizar la fusión con el objeto recién detectado. Posteriormente con el patrón resultante de la fusión, realizar lo mismo con el siguiente

Algoritmo 20 Fusión de patrones.

Entrada: Requiere la base de conocimiento de patrones aprendidos y la base de conocimiento de objetos detectados, identificador del objeto que pasa a ser patrón.

Salida: Porcentaje de similitud por trayectoria.

si Si un elemento pasa a formar parte de la base de conocimiento de patrones **entonces**
si No está vacía la lista de patrones con trayectoria similar o recorrido similar **entonces**
para Todos los patrones con los que existe similitud **hacer**
Obtener el identificador de un patrón de la lista de similitud de patrones
si Si el porcentaje de similitud es al menos superior al 80 % **entonces**
para Todo BB que forman la trayectoria del patrón **hacer**
Realizar fusión BB clave a BB clave.
Calcular BB intermedios.
Añadir la velocidad a cada BB fusionado.
fin para
Calcular la trayectoria del recorrido de la trayectoria fusionada.
Calcular el tipo de trayectoria que era, PEDESTRIAN o VEHICLE.
Asignar categoría a la nueva trayectoria, NOT_CLASSIFIED, NORMAL o UNNORMAL.
fin si
fin para
fin si
fin si
Añadir el patrón reslultante a la base de conocimiento de patrones aprendidos.

patrón con el que exista una similitud superior al 80 %. Así sucesivamente con el resto de patrones que tengan una similitud superior al 80 %.

La fusión de dos patrones consiste en la obtención de la trayectoria de los dos patrones, para cada BB clave que esté formado ambas trayectorias se realiza la fusión aplicando el algoritmo 6. El resultado de la fusión de estos BB se almacenan, en una lista denominada *Trajectory_vector* en la que se encuentran los BB fusionados. Durante el proceso de fusión se obtiene también la velocidad entre cada fragmento existente y se almacena la velocidad media de cada fragmento. Una vez fusionados los BB clave de ambos patrones. Se calculan los BB intermedios de los BB clave calculados. Por último, se calcula el recorrido del movimiento realizado por la nueva trayectoria resultante. Ya se ha obtenido el patrón resultante de la fusión de dos patrones, ahora se realiza la misma operación con el resto de patrones existentes en la lista de patrones con una similitud superior al 80 %. Una vez realizada la fusión de todos los patrones similares se almacena el patrón resultante de la fusión en la *base de conocimiento de patrones aprendidos*.

Módulo de Depuración.

El objetivo de la utilización de este submódulo es la correcta implementación de los diferentes algoritmos que componen la aplicación. De esta forma, que la utilización de este submódulo ha facilitado la obtención de resultados a la hora de comprobar el comportamiento de *Atravel*. *Atravel* es una aplicación implementada utilizando el lenguaje de programación de *Python*. Este lenguaje incorpora un módulo propio de depuración denominado *pdb* (Python DeBugger). Este módulo permite conocer de forma rápida y precisa, cualquier error existente en el código de programación.

Las pruebas utilizadas a cada uno de los diferentes submódulos que componen la aplicación han sido *pruebas de caja negra*. La utilización de estas pruebas consiste en introducir una serie de entradas correspondientes a cada módulo y analizar las salidas que éste genera. Dada la dificultad de comprobar el correcto comportamiento de los diferentes módulos se ha utilizado la herramienta de modelado gráfico *Blender*.

Para comprobar el funcionamiento de los diferentes módulos se han modelado diferentes objetos y escenas utilizando *Blender*. Tales como la generación de un objetos de tipo *Cube*, *Plane*, *Camera*, para comprobar situaciones, como la correcta generación, de un BB intermedio a dos BB, correcto almacenamiento de la trayectoria, correcto almacenamiento del recorrido, fusión de BB, etc.

5.6.1. Módulo de Visualización.

En esta sección se especifica las diferentes funcionalidades que ofrece al usuario. Este módulo hace referencia a las distintas funcionalidades recogidas fundamentalmente por la capa de dominio de la aplicación y más concretamente la interfaz gráfica de usuario.

Debido a que *Atravel* es una aplicación que desempeña las funcionalidades correspondientes a la etapa de análisis de comportamientos, dentro de la estructura de un sistema de videovigilancia. Necesita los resultados de detección, clasificación y seguimiento de los objetos en el entorno de monitorización. De forma meramente informativa y teniendo en cuenta que se trata de un componente importante dentro de un sistema de monitorización. Se ha creado un **submódulo de visualización de flujo de vídeo real**.

El *Submódulo de visualización de flujo de vídeo* se ha creado para poner en contexto al usuario que desee probar o visualizar las funcionalidades de la aplicación. En este submódulo se pueden distinguir dos funcionalidades:

- **Módulo de representación de vídeo real**: En este módulo se realiza la reproducción del flujo de vídeo real almacenado. Dado a que *Atravel* es una aplicación caracterizada en desempeñar las funcionalidades de la etapa de análisis de comportamientos de objetos de forma automática en sistema de vigilancia multicámara permite la opción de reproducir el flujo de vídeo de una escena captada desde diferentes puntos de vista y por dispositivos de vídeo diferentes. El máximo número de cámaras de visualización de la escena que pueden ser utilizadas se ha establecido en tres, por motivos de espacio con la resolución de la pantalla disponible. Por tanto, se puede incorporar el flujo de vídeo de una a tres cámaras. Las diferentes ventanas en las cuáles tiene lugar la visualización del flujo de vídeo son de tipo **OpenCV**. La única funcionalidad de éstos es la de reproducción de un archivo de vídeo.

- **Módulo de representación 3D**: El módulo de representación 3D tiene lugar en una ventana que utiliza la tecnología de **OpenGL**. La funcionalidad de ésta es la representación en 3D de los objetos detectados por las cámaras de seguridad. Es decir, en este componente se realiza la visualización de los BB que son detectados en cada instante de tiempo en la escena de monitorización y es representada su posición en cada momento. Este componente incorpora la capacidad de rotación, zoom y traslación de la representación de la escena. De esta forma el usuario puede rotar el objeto, trasladarlo o aumentar y disminuir el tamaño del objeto para apreciar todos los detalles para su correcta visualización.

Sin embargo, dado que *Atravel* es una aplicación que se encarga de la monitorización de una escena en concreto. Para la representación de las trayectorias de los objetos detectados, su recorrido, y situación de la escena se ha utilizado una ventana que realice la representación de éstos en 3D. Esta ventana está dotada de las funciones de rotación traslación y zoom. Del mismo modo existe una ventana idéntica para la representación en 3D de las características de los patrones que se encuentran en la *base de conocimiento de patrones aprendidos*, tanto su trayectoria como su recorrido. En ambas ventanas son del mismo tipo y están realizadas con *OpenGL*. También ambas tienen las funciones de rotación, traslación y zoom.

Cada uno de los submódulos realizan las siguientes funciones:

- **Submódulo de visualización de patrones**: En este submódulo existe un componente de tipo *OpenGL*, en el cuál se representan las características de cada patrón como son su trayectoria y su recorrido. Además se puede apreciar el tipo de patrón del que se trata, es decir, si se trata de un patrón de tipo *VEHICLE* o de tipo *PEDESTRIAN*. Además, se puede visualizar la categoría de comportamiento de dicho patrón, pudiendo encontrar patrones de tipo *NOT_CLASSIFIED* representado con un logotipo de color ambar, de tipo *NORMAL* representado con un logotipo de color verde y de tipo *ABNORMAL* representado con un logotipo de color rojo. También se puede observar que la representación de las trayectorias viene determinada por líneas continuas que definen el contorno de la trayectoria. Líneas discontinuas que definen el contorno del recorrido del patrón y el color de estas líneas que definen la trayectoria o recorrido de un patrón viene determinado por el tipo de comportamiento. Es decir, si un patrón esta clasificado como *NOT_CLASSIFIED* las líneas que describen el contorno de su trayectoria son de colo ambar, en caso de que sea de tipo *NORMAL* serán de color verder y en caso de que sea *ABNORMAL* las líneas serán de color rojo. También aparece una pequeña descripción con el identificador del patrón. Además se puede determinar la visualización de un patrón en concreto, seleccionando su trayectoria o recorrido, dependiendo de la elección del usuario. Del mismo modo, el usuario puede elegir mostrar en el componente de visualización las trayectorias de todos los patrones aprendido o el recorrido de éstos. Para eliminar ruido en la visualización de patrones, se pueden eliminar la trayectoria o recorridos que el usuario decida, o bien eliminar todas la líneas que existen en el componente quedando vacío.

- **Submódulo de visualización de Objetos**: En este submódulo se puede visualizar toda la información referente a los objetos detectados en la escena en cada instante. Así pues el usuario puede seleccionar un objeto determinado y visualizar

su trayectoria, recorrido el identificador de patrones y el grado de similitud por trayectoria, recorrido y velocidad existente con cada uno de ellos. Además se puede representar en el componente de visualización 3D la información de los patrones similares representando su trayectoria o recorrido, a petición del usuario.

En la representación de la información de cada objeto el usuario, puede visualizar la trayectoria del objeto a lo largo de la escena, actualizándose en cada momento. La representación de la trayectoria viene determinada por la representación del contorno que define el desplazamiento del objeto sobre el entorno de monitorización. Para ello, se han utilizado líneas continuas que definen la trayectoria del objeto en cada momento. Para identificar el recorrido realizado por los objetos se han utilizado líneas discontinuas, para distinguir la representación del recorrido y la de la trayectoria. También al igual que se ha detallado en el *Submódulo de visualización de patrones*, las líneas que representan el recorrido y la trayectoria del objeto pueden variar su color dependiendo de la similitud de su comportamiento con respecto a los patrones almacenados. Así pues, si un objeto detectado en la escena tiene un alto grado de similitud con un patrón que es de tipo *NOT_CLASSIFIED*, entonces el color de las líneas será representado de color *ambar*, si es de tipo *NORMAL* sus líneas serán de color *verde* y si es de tipo *ABNORMAL* serán representadas de color *rojo*. También se representa en cada trayectoria o recorrido el identificador de objeto que describe la trayectoria o el recorrido. Se puede distinguir cuál es el inicio de la trayectoria y cual es el final ya que en la posición actual del objeto, existe una marca junto con el identificador que va cambiando según el desplazamiento del objeto a lo largo de la escena.

El porcentaje de similitud tanto por trayectoria como por velocidad como por recorrido aparece reflejado en la interfaz como una barra junto con el porcentaje de similitud existente. Además el interior de la barrita se colorea de un color diferente según sea inferior a 50 % de color *rojo* o superior a 50 % de color *verde*.

También existen botones que permiten al usuario realizar la visualización concreta de:

- Trayectoria: representa la trayectoria realizada por el objeto en un instante determinado.
- Recorrido: representa el recorrido realizado por el objeto en un instante de tiempo.
- Trayectoria de un patrón con el cuál existe cierta similitud: representa la trayectoria realizada por el patrón con el cuál existe similitud.

- Recorrido de un patrón con el cuál existe cierta similitud: representa el recorrido descrito por el patrón seleccionado.
- Eliminación de su trayectoria: Elimina la trayectoria del objeto selccionado representada en el componente de visualización.
- Eliminación de su recorrido: Elimina el recorrido del objeto seleccionado representado en el componente de visualización.
- Eliminación de la trayectoria de un patrón: Eliminación de la trayectoria de un patrón seleccionado, que estaba representado en el componente de visualización.
- Eliminación del recorrido de un patrón: Eliminación del recorrido de un patrón seleccionado que estaba representado en el componente de visualización.
- Eliminación completa de todos los objetos y patrones que estén representados en el componente de visualización.

5.7. Patrones y diagramas de clases de diseño.

En el desarrollo de los diferentes submódulo que componen la aplicación se ha utilizado los siguientes patrones de diseño:

- *Patrón arquitectónico multicapa*: Este patrón ha sido utilizado para separar la funcionalidad de la aplicación, por un lado aparece se encuentra la capa en la que se desarrolla toda la funcionalidad y algoritmos utilizados (capa de dominio) y en la capa de presentación se encuentran toda la funcionalidad encargada de la visualización e interacción con el usuario. (véase la sección 4.1.
- *Patrón facade*: Este patrón ha sido aplicado para recoger las peticiones realizadas por el usuario, delegando la responsabilidad de cada petición sobre las demás interfaces secundarias. Esto ocurre por ejemplo cuando un usuario realiza una petición ésta es recogida por la capa de presentación (clase main_window), la cuál delega la responsabilidad en la clase GLWidget, encargada de responder a la petición.
- *Patrón iterator*: Se ha utlizado para facilitar el acceso secuencial a listas y abstraerse de la representación interna de las estructuras de datos como listas o diccionarios.
- *Patrón singleton*: Debido a la necesidad de comunicación de las interfaces secundarias con la interfaz principal y evitar la existencia de múltiples puntos de acceso a

ésta se ha aplicado el patrón singleton. De esta forma el acceso a la interfaz principal es realizado de forma global, existiendo una única instancia de la interfaz principal (main_window).

- *Patrón experto*: La utilización de este patrón garantiza la creación de objetos de otras clases, sólo en las estructuras que contengan la información necesaria para la creación de éstas.

El diagrama de clases correspondiente a la capa de presentación se puede observar en la figura 5.25:

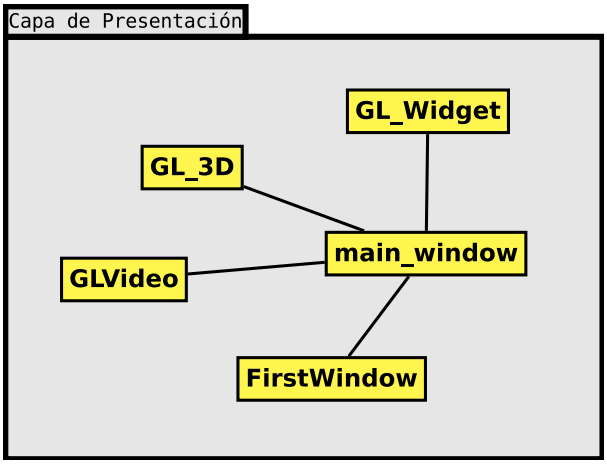

FIGURA 5.25: Diagrama de clases de la capa de presentación.

El diagrama de clases correspondiente a la capa de dominio se puede observar en la figura 5.26:

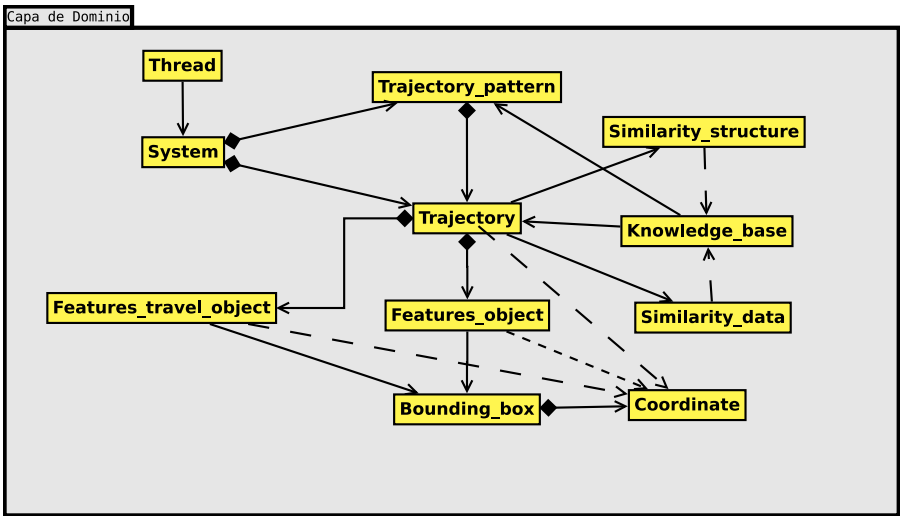

FIGURA 5.26: Diagrama de clases de la capa de dominio.

En la figura 5.27, se puede observar la conexión entre la capa de presentación y de dominio.

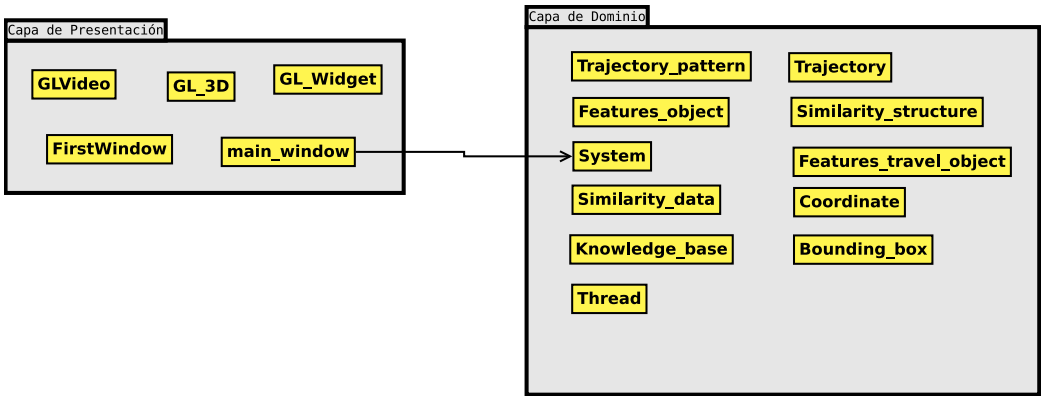

FIGURA 5.27: Diagrama de clases de la capa de dominio.

6

Evolución y Resultados

En este capítulo se especifica la planificación utilizada para la implementación de *Atravel* desde su comienzo hasta su fin. Se trata de situar al lector en la consecución de los distintos objetivos a lo largo del desarrollo del proyecto.

También se incorpora una sección en la que se explican los resultados obtenidos tras las pruebas realizadas con cada uno de los submódulos que componen la aplicación. Además, se detallan los resultados obtenidos tras la realización de pruebas con un entorno real.

6.1. Evolución

Atravel fue elegido para su desarrollo en este proyecto el *20 de Septiembre de 2010*. Durante un tiempo, se fueron estableciendo las diferentes funcionalidades que debía incorporar la aplicación. A medida que se iban incorporando a la aplicación eran especificadas en un diagrama de casos de uso (véase figura 4.2). Una vez especificada toda la funcionalidad que debía incorporar *Atravel* se elaboró un primer diseño preliminar (véase la figura 4.2), el cuál fue el punto de partida, para la generación de las iteraciones realizadas para el desarrollo de la aplicación.

6.1.1. Iteraciones

En esta sección de describen los hitos alcanzados en cada iteración, así como la descripción de los distintos problemas encontrados en la consecución de cada uno de los hitos. El proceso de desarrollo de las iteraciones se comenzó el *10 de Noviembre de 2010*.

Iteración 1.

El primer hito realizado fue la construcción de un script en el lenguaje de programación *Python*, para obtener las propiedades de los objetos necesarias para realizar pequeñas pruebas de la funcionalidad de *Atravel* utilizando la herramienta de diseño y modelado en 3D denominada *Blender*. El motivo de creación de este script es conseguir un archivo de texto, en el cual se almacenen las propiedades de objetos que se desplazan sobre una escena virtual en 3D. El archivo de texto debe contener la misma información, que la información contenida en el archivo resultante tras la etapa de *Tracking*. De manera, que pueda ser utilizado para realizar pruebas con un entorno virtual sin necesidad de tener que establecer un sistema de monitoreo. El archivo de texto resultante debe contener el mismo formato que el especificado en la sección 5.1.1. El script debe transformar la actividad de objetos existente en una escena y escribirla en un archivo de texto siguiendo el formato anteriormente especificado (*tipo, identificador, vértices del bounding box, frame, fecha y hora*).

Para probar el script se ha creado una escena virtual en 3D, formada únicamente por objetos de tipo *Cube* que se desplazan a lo largo de un plano en 3D. Sólo han sido utilizado objetos de tipo *Cube* debido a que todo objeto detectado por el sistema de monitorización es envuelto con en una caja que se ajusta al tamaño real del objeto (bounding box).

Iteración 2.

A partir del archivo de texto (creado a través del script), apareció la necesidad de almacenar toda la información que aparece plasmada en el archivo de texto, en una estructura para realizar un rápido acceso a la información obtenida en cualquier instante. Para ello se elaboró la estructura denominada *Features_object*, la cual almacena la información correspondiente a la detección de un objeto en un instante de tiempo determinado. Esta estructura, que contiene información de *Bounding box, identificador, frame, tiempo, fecha y velocidad* con la que un objeto es detectado en la escena observada. Para el almacenamiento correspondiente del *Bounding box* se ha creado una estructura que almacene la información de éste.

Inicialmente la estructura *Bounding_box* se diseñó utilizando únicamente dos vértices. El vértice correspondiente a la esquina inferior izquierda y a la esquina superior derecha. En un principio, se estableció que utilizando únicamente estos dos vértices bastaría para la representación de un BB.

Además se definió la estructura de *Punto* necesaria para la creación de un BB. Un *Punto* se definió como la estructura formada por tres valores de tipo *float* que hacen referencia a los valores correspondientes a los tres ejes de coordenadas.

La especificación del valor de la velocidad viene determinado por valor de tipo *float*, el cual se calcula mediante la variación de espacio y tiempo con respecto al objeto almacenado anteriormente.

Iteración 3.

Creación de la función encargada de leer los datos especificados en el archivo de texto anteriormente elaborado. Además, toda la información obtenida procedente del archivo de texto debía almacenarse de alguna forma en una estructura. Así que, se ha utilizado una estructura de *diccionario*. Tras barajar diversas estructuras, como estructura en *árbol, grafos, tablas hash* se optó por utilizar un diccionario.

Los principales motivos de la utilización de este tipo de datos, fue la sencillez de implementación en el lenguaje de programación de *Python*, el rápido acceso a los elementos del diccionario y su gran flexibilidad en cuanto a la longitud de un *diccionario*. Los elementos de los diccionarios están caracterizados por tener una clave y un valor. En este caso, los elementos tenían como clave el identificador del objeto detectado en la escena virtual. Y como valor se creó un objeto de tipo *Trajectory* (véase sección 5.2.1). Cada objeto *Trayectory* inicialmente estaba creado por una lista de objetos de tipo *Features_object*, el identificador del objeto y el tipo del objeto. La lista de objetos *Features_object* almacena toda la información correspondiente al objeto detectado en cada instante. Esta lista contiene, los principales datos para la posterior representación de la trayectoria y la velocidad de un objeto determinado. En cuanto a la representación, no se realizaba la generación de la trayectoria del objeto continua, sino que únicamente, se realizaba la representación de los BB clave que forman la trayectoria. La velocidad se representa por fragmentos, es decir, se calcula la velocidad del objeto en cada uno de los fragmentos que definen la trayectoria del objeto.

Iteración 4.

Una vez almacenada la información más relevante de cada uno de los objetos en cada instante de tiempo se decidió realiza un submódulo encargado de optimizar el número de BB almacenados en la lista anteriormente especificada. Ya que el número de BB almacenados era demasiado grande y muchos de ellos se encontraban solapándose y no aportaban

información relevante en cuanto a la dirección acerca de la trayectoria de un objeto. Por lo que se decidió sólo almacenar aquellos bounding box que presentasen un cambio importante en la representación de la trayectoria del objeto.

En la realización de esta función por cada objeto detectado en cada instante de tiempo, se comprobaba si la caja envolvente al objeto (bounding box) se solapaba con el bounding box anteriormente almacenado. Para calcular la existencia de solapamiento o no se realizaba mediante la creación de una función que permitía conocer si existía colisión entre dos BB dados. Esta función se puede ver en 4. En caso de existir colisión entre dos objetos distintos el BB detectado era descartado totalmente. En esta iteración también se desarrolló una función encargada de conocer la existencia de cambios importantes en la dirección del desplazamiento del objeto. Así pues, se introdujo un nuevo concepto, que fue la incorporación de un vector dirección. Este vector tiene como principal misión calcular la dirección del objeto en un instante de tiempo determinado. Teniendo la dirección con la que se desplaza el objeto a lo largo del entorno monitorizado, cuando se detecta el objeto en un frame diferente, se calcula el cambio en primer lugar si existe colisión con el objeto anteriormente almacenado, en caso de no se existir colisión se calcula la diferencia existente en cuanto a la dirección anterior y la actual. Si existe un cambio en la dirección de desplazamiento del objeto lo suficientemente importante, entonces el BB es almacenado en la lista de detección del objeto. En otro caso, el BB es eliminado. Es importante recalcar, que para calcular el cambio de dirección en la trayectoria de un objeto, los vectores han sido normalizados, para que los cálculos sean correctos.

De esta forma la trayectoria correspondiente al desplazamiento del objeto a lo largo de la escena monitorizada se almacena con el mínimo número posible de BB clave.

Iteración 5.

EL siguiente hito a realizar, consistió en la representación de las trayectorias de forma continua. Para ello fue necesario la creación de una función en la que dados dos BB clave (un BB clave es aquel BB que no ha sido elaborado posteriormente sino que es el inicialmente detectado en el entorno monitorizado) calcule el BB que realiza la unión espacial de estos dos BB. La correcta implementación de esta tarea es imprescindible para alcanzar el éxito en la aplicación, ya que gran parte de las comparaciones posteriores se basan en la comparación de este BB resultante y denominado como BB intermedio.

El BB intermedio debe representar lo más fielmente posible el movimiento del objeto. Para ello, dados dos BB es necesario elaborar una función que identifique las dos caras (una cara perteneciente a cada BB) que representen lo más exacto posible el movimiento

del objeto a lo largo de la escena. Posteriormente se debe realizar correctamente la unión de cara con cara, teniendo un cuidado especial a la hora de emparejar vértice con vértice de las dos caras. Se trata de una tarea a priori sencilla pero muy costosa, dadas las diferentes situaciones que se pueden dar.

Como resultado de esta iteración se obtiene un BB denominado *intermedio* (debido a que se encuentra entre dos BB claves), que permita representar una trayectoria continua para la visualización del movimiento del objeto continuo.

Sin embargo, al almacenar y crear el BB intermedio apareció un problema. El problema es que no se ajustaba a la implementación inicial de Bounding box definido por dos puntos. Debido a que las aristas del BB no se encontraban alineadas con los ejes de coordenadas X,Y y Z, es decir, los BB clave sí que se encontraban alineados con los ejes de coordenadas pero los BB intermedios no.

Por tanto, se tuvo que rediseñar la representación de un Bounding box. Ahora la definición de BB viene determinada por el valor de los ocho vértices que los definen, en lugar de dos.

Iteración 6.

El hito número 6 consiste en el cálculo de la velocidad con la que se desplaza el objeto a lo largo de la escena observada. El cálculo de la velocidad se realiza al mismo tiempo en que se obtienen los BB que forman la trayectoria del objeto, después de aplicar el submódulo de optimizar las trayectorias. De modo, que cuando se detecta que un BB es clave, se obtiene su posición en la escena y el tiempo en el que ha sido registrado. A continuación, se obtiene la posición del BB clave anteriormente almacenado y el tiempo. Después de tener los datos de ambas posiciones se realiza el cálculo de la velocidad utilizando la fórmula detallada en la sección 5.2.4, calculando la diferencia entre ambas magnitudes.

El resultado es almacenado en el atributo denominado *velocity* perteneciente a la estructura de datos *Features_object*.

Iteración 7.

El hito de la iteración 7 consiste en calcular la representación del recorrido realizado por un objeto. La generación del recorrido se realiza al mismo tiempo que se calcula los BB clave que forman la trayectoria de un objeto. Cada vez que es detectado un BB clave, tras la comprobación de solapamiento y dirección de éste, se calcula el recorrido seguido por la trayectoria que define el movimiento del objeto. Para ello se ha utilizado una función la cual

comprueba la existencia de un cambio importante en la dirección del movimiento del objeto. Se obtiene el último BB que registró un cambio significativo en la dirección del objeto y se calcula el punto central de éste. Además se obtiene también el punto central del BB recién detectado. Se calcula el vector dirección de éstos entre estos dos puntos y se compara con la dirección que llevaba el objeto hasta esta posición si existe un importante cambio en la dirección, se calcula el BB nuevo del recorrido sólo para el BB recién detectado, dejando intactos los BB anteriores del recorrido. En caso de que no provoque un cambio lo suficientemente fuerte, se recalcula el BB del recorrido anterior ahora incluyendo el nuevo BB detectado. En esta iteración se almacenan todos los BB que forman el recorrido en una lista de BB. Cada vez que se detecta la aparición de un BB clave se actualiza el recorrido del objeto actualizando los BB correspondiente. Esta lista se encuentra almacenada en la estructura de datos denominada *trajectory*. Por tanto, recorrido de un objeto a lo largo de la escena se puede definir como una serie de BB que definen el espacio físico del movimiento del objeto. Se puede ver más detalladamente en la sección 5.2.4.

Iteración 8.

En la iteración número 8 el principal objetivo fue el de calcular cómo aprender un patrón y cómo almacenarlo. Además también se debe especificar la forma de almacenar los patrones reconocidos en el entorno para realizar la clasificación. Las propiedades captadas y almacenadas pertenecientes a la aparición de un objeto en la escena monitorizada se encuentran almacenadas en un diccionario denominado *base de conocimiento de objetos detectados*. Para representar un patrón, en primer lugar es necesario conocer, el instante de tiempo en el que un objeto detectado en el entorno observado deja de aparecer en la escena. Para ello, se ha creado un atributo en la estructura de datos *Trajectory* denominado *number_missing_frames* que actúa como contador, en caso de pasar 3 frames sin recibir información de un objeto, se establece que éste ha abandonado la escena. Una vez que se ha detectado que un objeto ha abandonado la escena, éste pasa a considerarse como patrón. Se ha creado una estructura que permite almacenar cada patrón, considerando todos los datos acerca de su trayectoria, recorrido y velocidad. En la que además un patrón debe venir definido por su categoría, la cual expresa el tipo de comportamiento del patrón, el tipo de patrón, atendiendo a si ha sido realizada por un vehículo o por un peatón, y a la fecha en la que ha sido elaborado el patrón. La estructura utilizada para almacenar cada patrón se denomina *Trajectory_pattern*. Utilizando esta estructura, y obteniendo los datos necesarios, un objeto detectado pasa a ser un nuevo patrón aprendido.

Además es importante definir la estructura donde se almacenan todos los patrones y acceder a ellos de forma directa con la mayor rapidez posible. Para ello en esta iteración también se ha realizado la creación de una *base de conocimiento de patrones aprendidos*. Esta base es una estructura de tipo *diccionario* en la cual se almacenan todos los patrones, la estructura es de tipo *Knowledge_base*. Actúa como estructura de almacenamiento de toda la información acerca de los patrones que forman la aplicación.

Cada patrón recién reconocido, debe adquirir un nuevo identificador de patrón, que debe de ser único. Los patrones son almacenados en la *base de conocimiento de patrones aprendidos* utilizando como clave el nuevo identificador del patrón y como valor la estructura de tipo *Trajectory_pattern*.

Iteración 9.

El hito a realizar en la iteración número 9 consiste en clasificar el comportamiento de un objeto atendiendo a la similitud de la definición de su trayectoria. En este hito el principal objetivo es realizar el proceso de clasificación que consiste en calcular el porcentaje de similitud entre la trayectoria realizada por el objeto detectado en la escena observada y la trayectoria de cada uno de lo patrones existentes en la *base de conocimiento de patrones aprendidos*. Para ello se han elaborado dos algoritmos diferentes.

EL primero de ellos hace hincapié en los cambios de dirección que pueden detectarse en la representación de una trayectoria. Los cambios de dirección en la representación de una trayectoria vienen determinados por los BB claves que la componen. El algoritmo consiste en realizar una comparación entre los BB clave correspondientes de la trayectoria del objeto recién detectada y los BB clave correspondiente a cada uno de los patrones existentes en la *base de conocimiento de patrones*. La comparación consiste en chequear si existe colisión entre ambos BB clave. Dependiendo del número de BB clave con los que exista colisión, así será el porcentaje de similitud. Si existe colisión de todos los BB clave del objeto detectado con el BB clave de un determinado patrón entonces existirá un 100 % de similitud. En caso de que se haya comparado dos veces el objeto detectado con un patrón determinado y no haya existido colisión, automáticamente se descarta el patrón debido a que no existe la similitud suficiente como para clasificarlo como tal.

El segundo método tiene como principal objetivo la similitud de trayectorias sin tener en cuenta los cambios de dirección. Para ello, se ha desarrollado un algoritmo que realiza una comparación con BB clave y BB intermedios. Este algoritmo realiza una comparación del BB recién detectado con un BB de cada patrón, éste BB es dos posiciones mayor a la posición del BB con el que existió colisión. Si no ha existido colisión nunca con el patrón,

entonces se realiza la comparación con el BB clave correspondiente. Si no existe colisión con el BB del patrón elegido entonces se realiza una comprobación con los BB de alrededor al elegido, exactamente con los dos BB anteriores y con los dos BB posteriores al al BB elegido. En caso de no existir colisión con ninguno de estos BB se aumenta en una unidad el contador de fallo en la colisión. Si este contador llega a tener un valor de dos unidades o más, entonces el patrón es descartado. A partir del número de colisiones producidas de la trayectoria del objeto con el patrón en concreto, se calcula el porcentaje de similitud.

Los resultados de esta similitud son almacenados en una estructura de *diccionario* como atributo de la estructura *Similarity_structure*. Tal y como se ha detallado en la sección 5.2.2.

El principal objetivo que tiene la realización de este hito es clasificar el comportamiento de un objeto que se desplaza sobre la escena, según con la categoría del patrón al que tenga mayor similitud. Realizar una buena clasificación del movimiento de los objetos permite tener un buen comportamiento de la aplicación.

Iteración 10.

El hito realizado en la iteración número 10 consiste en calcular el grado de similitud existente entre el recorrido realizado por un objeto en el entorno monitorizado y el recorrido de los patrones existentes en la *base de conocimiento de patrones aprendidos*. El algoritmo se basa en realizar una comparación con los BB clave que componen el recorrido de un objeto. Para cada objeto que es detectado en la escena monitorizada se calcula el BB clave del recorrido existente en ese momento. Posteriormente se realiza una comprobación con el BB correspondiente de cada patrón existente en la *base de conocimiento de patrones aprendidos*. La comprobación consiste en detectar si existe colisión entre el BB clave del recorrido del objeto recién detectado y el BB clave correspondiente al recorrido de cada patrón. Esta comprobación se realiza para todos y cada uno de los BB clave que forman el recorrido del movimiento de un objeto. Al término de la lista de BB que forman el recorrido se encuentran acumulados en una lista los resultados de las comprobaciones de cada uno de los BB. A partir de ella se obtiene el porcentaje de similitud de ésta y es almacenado en el diccionario existente en la estructura de datos *Similarity_structure* cuyo nombre es *similarity_store*.

El principal objetivo de realizar este algoritmo, es proporcionar a la aplicación una clasificación más completa del comportamiento del movimiento de los objetos. Permite una clasificación del movimiento de los objetos por zonas que componen la escena. De esta forma, se puede obtener una clasificación dependiendo de las distintas zonas en las que

se divide la escena de monitorización. Como desventaja, se pierde algo de precisión en la representación del movimiento concreto del objeto.

Iteración 11.

En el hito número 11 se ha llevado a cabo la implementación de una función que calcule el grado de similitud de la velocidad a la que se desplaza un objeto a lo largo de la escena observada. El objetivo de este hito es detectar y avisar de aquellas situaciones que tengan similitud con un patrón determinado pero que la velocidad a la que circula el objeto es excesivamente alta o baja. Es decir, se trata de complementar la información en la clasificación del comportamiento de un objeto determinado, además de por su similitud realizada por la trayectoria o el recorrido. El algoritmo de similitud por velocidad trata de chequear la velocidad de un objeto detectado en la escena con la velocidad existente en el tramo correspondiente de aquellos patrones que realicen una trayectoria similar a la del objeto recientemente detectado. Para ello, se obtiene la velocidad en el fragmento del objeto recién detectado (tal y como se puede observar en la sección 5.2.4). La comparación se realiza por fragmento de la trayectoria del objeto recién detectada con la trayectoria del objeto de los patrones almacenados en la *base de conocimiento de las trayectorias*. Es importante remarcar que el grado de similitud por velocidad sólo se realiza con aquellos patrones con los que existe una alto grado de similitud por trayectoria. El porcentaje de similitud por velocidad viene determinado por el número de tramos en los que existe similitud, de todos los tramos totales que forman la trayectoria del patrón.

Iteración 12.

El hito 12 consiste en la generación de las funciones necesarias para dotar a la aplicación de un algoritmo de fusión de patrones. La principal funcionalidad que aporta es optimizar la base de conocimiento de patrones, con el fin de compactar y reducir las situaciones almacenadas en la base de conocimiento de patrones. Esta funcionalidad se consigue a través de la fusión de patrones que tiene una trayectoria, recorrido y velocidad similares. Una vez que se tienen los datos tras los diferentes análisis de similitud por trayectoria, recorrido y velocidad. Se obtienen aquellos patrones con los que el objeto tiene mayor afinidad. Para dotar al sistema de la capacidad de aprendizaje, se realiza la fusión del objeto que pasa a ser patrón con aquellos patrones con los que tiene mayor afinidad.

El algoritmo de fusión permite unificar dos patrones dando lugar a un patrón resultante con las características de ambos. Dados dos patrones la fusión de ambas trayectorias

se realiza obteniendo los bounding box clave de ambos y para cada par de ellos correspondientes se genera un bounding box nuevo que engloba a ambos. A continuación se generan los BB intermedios para tener una trayectoria continua. La velocidad aplicada a cada fragmento existente entre cada BB clave viene determinado por el valor medio de la velocidad de ambas trayectorias en el fragmento correspondiente. El recorrido de esta trayectoria es generado, una vez que se han generado la trayectoria completa, siguiendo el método explicado en la sección 5.2.4. Por último se almacena la información del nuevo patrón resultante de la fusión en una estructura de tipo *Trajectory_pattern*.

Iteración 13

Una vez terminada la funcionalidad de la aplicación se procedió a realizar el último hito de este proyecto que consiste en realizar un despliegue de la aplicación sobre un entorno real para analizar su comportamiento. El entorno real utilizado ha sido la *calle Darro del municipio Almodóvar del Campo (Ciudad Real)*. Tras las realización de las pruebas del sistema se realizaron diferentes ajustes de los umbrales y valores de configuración, para amoldar la aplicación al entorno de monitorización y sacar el máximo partido a la monitorización de la escena.

El tiempo empleado en la realización de las iteraciones se encuentra detallado en la figura 6.1.

6.1.2. Recursos y costes

El tiempo estimado en la elaboración de *Atravel* ha sido de 48 semanas aproximadamente. Para el desarrollo de todo el trabajo realizado en este proyecto se requerido de personal humano y de recursos hardware, con sus respectivos costes. A continuación, en la figura 6.2 se detallan los recursos y costes utilizados para el desarrollo del proyecto.

En la figura 6.3 se representan el número de archivos, la cantidad de líneas de código utilizadas, la cantidad de líneas comentadas y el número de líneas en blanco empleadas para el desarrollo del proyecto.

6.2. Resultados

En esta sección se detallan los resultados obtenidos con *Atravel* tras las monitorización de un entorno real o virtual. El motivo por el cuál se ha utilizado un entorno virtual es

Número de Iteración	Semanas de Trabajo	Horas de Trabajo
Iteración 1	**7**	**280**
Iteración 2	**3**	**120**
Iteración 3	**5**	**200**
Iteración 4	**5**	**200**
Iteración 5	**4**	**160**
Iteración 6	**1**	**40**
Iteración 7	**4**	**160**
Iteración 8	**3**	**120**
Iteración 9	**3**	**120**
Iteración 10	**5**	**200**
Iteración 11	**3**	**120**
Iteración 12	**5**	**200**
Iteración 13	**4**	**160**
TOTAL	**48**	**1920**

FIGURA 6.1: Tabla de duración de cada iteración.

Recursos	Cantidad	Coste(€/unidad)
Recursos humanos	**3**	**20780**
Cámara Logitech AF-Sphere	**2**	**117,16**
Computadora portátil	**1**	**719,95**
TOTAL		**63294,13**

FIGURA 6.2: Recursos y costes utilizados en la elaboración de *Atravel*.

para probar las funcionalidades implementadas inicialmente sin necesidad de realizar un despliegue del sistema para probar el comportamiento de cada módulo que componen el sistema. También se ha probado la aplicación en un entorno real con el objetivo de obtener un comportamiento más objetivo ante la aparición de factores como el ruido, oclusión, etc.

Para la representación de resultados se ha utilizado **Curvas ROC** (acrónimo de Receiver Operating Characteristics). Se trata de una métrica que permite obtener el ratio existente entre verdaderos positivos (VPR) frente a los falsos positivos (FPR). Para obtener más información acerca de las *Curvas Roc*[1].

La clasificación de resultados en cada uno de los tipos se ha realizado siguiendo:

- **Verdadero Positivo**: Situación normal correctamente clasificada (VP).

[1] http://es.wikipedia.org/wiki/Curva_ROC

Tipo de Archivo	Número de Archivos	Líneas en Blanco	Líneas de Comentario	Líneas de Código	Total Líneas
Python	**19**	**378**	**669**	**4038**	**5085**
Script	**1**	**10**	**9**	**59**	**78**
TOTAL	**20**	**388**	**678**	**4097**	**5163**

FIGURA 6.3: Representación del número de archivos y líneas de código, comentadas y en blanco, utilizadas en el desarrollo de *Atravel*.

- **Verdadero Negativo**: Situación anómala correctamente clasificada como anómala (VN).
- **Falso Positivo**: Situación anómala incorrectamente clasificada como normal (FP).
- **Falso Negativo**: Situación normal incorectamente clasificada como anómala (FN).

Las medidas utilizadas y los resultados obtenidos son:

- *VPR*: Sensibilidad o razón de verdaderos positivos.

$$VPR = \frac{VP}{V} = \frac{VP}{VP+FN} \tag{6.1}$$

- *FPR*: Sensibilidad o razón de falsos negativos.

$$FPR = \frac{FP}{N} = \frac{FP}{FP+VN} \tag{6.2}$$

- *ACC*: Precisión o exactitud.

$$ACC = \frac{(VP+VN)}{(P+N)} \tag{6.3}$$

6.2.1. Resultados con entorno virtual

Para estudiar el comportamiento de *Atravel* es necesario contar con un despliegue previo de monitorización de una escena. Para ello, se necesitan requisitos, como la elección de la escena, colocación, posicionamiento y calibrado de cámaras, segmentación y tracking. Tras el proceso de *Tracking* comienza el funcionamiento de *Atravel*. Sin embargo, el despliegue de todas estas etapas de la monitorización es demasiado costoso y laborioso. De manera, que se ha desarrollado un entorno virtual muy sencillo, para poder estudiar el comportamiento

de la aplicación a medida que se han ido incorporando los diferentes módulos que componen la aplicación.

Para obtener los resultados tras la etapa de *Tracking* se ha empleado un script que permite determinar en un archivo de texto, todos los datos necesarios para el análisis del comportamiento de objetos llevado a cabo por *Atravel* (véase la sección 5.1.1). El modelado de la escena virtual ha sido elaborado utilizando la herramienta *Blender*.

La escena virtual utilizada está compuesta de tres cámaras virtuales, un plano, y un conjunto de *bounding box* o *cajas* que se desplazan a lo largo del plano (véase figura 6.4).

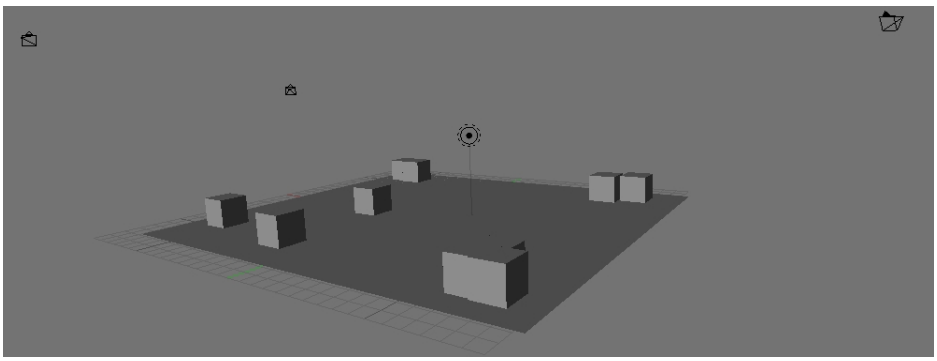

FIGURA 6.4: Representación del entorno virtual 3D.

La prueba virtual ha tenido una duración de 2000 frames, en la cuál se han analizado los comportamientos de los diez objetos que han sido creados para la prueba de los diferentes. Debido a que la realización de esta prueba virtual tiene como objetivo principal comprobar el correcto funcionamiento de cada uno de los módulos, los resultados obtenidos son totalmente correctos, ya que no aparecen factores externos que puedan distorsionar la monitorización y el análisis de comportamientos de los objetos.

Se trata de un escenario ideal, en la que se realiza un desplazamiento de diez objetos por diferentes zonas de la escena. Cinco de los objetos son de tipo vehículos y los otros cinco objetos son de tipo peatón.

En la figura 6.5, se detallan los resultados obtenidos tras la detección de los objetos sobre la escena. Hace referencia al número de trayectorias detectadas correctamente en el entorno. Esto significa, que el sistema ha detectado al objeto sobre la escena con su tipo y dimensiones correctamente, es decir, sin aparecer errores en la detección del objeto, bien por oclusión o por error en etapas anteriores del sistema.

También se ha comprobado el comportamiento del sistema en la representación de la trayectoria, recorrido y velocidad. La representación de éstos submódulos resulta de gran importancia para el análisis del comportamiento de los objetos. Se ha analizado que la representación de estas tres funcionalidades para comprobar si existe correspondencia entre

	Nº Trayectorias Correctamente Detectadas	**Nº Trayectorias Incorrectamente Detectadas**
Vehículo	**5**	**0**
Peatón	**5**	**0**
Porcentaje Vehículo	**100**	**0**
Porcentaje Peatón	**100**	**0**

FIGURA 6.5: Resultados utilizando el entorno virtual 3D.

el movimiento del objeto y la representación de cada funcionalidad. De forma, que una trayectoria es incorrectamente representada si visualmente se aprecia que las uniones entre bounding box que definen la trayectoria del objeto se realiza sin error, también se considera representación incorrecta de trayectoria, la existencia de BB clave en lugares donde no exista una cambio importante en la dirección de la trayectoria representada. De modo, que una trayectoria se considera correctamente representada si reproduce de manera más exacta posible el movimiento de un objeto, utilizando el menor número de BB clave posibles. Se considera una representación incorrecta del recorrido de un objeto, cuando los BB que lo definen no aparecen correctamente unidos, o cuando los BB clave que lo forman son el mínimo número posible para representar el recorrido del objeto teniendo en cuenta los cambios significativos en la dirección de su desplazamiento. En cuanto a el cálculo correcto de la velocidad con la que se desplaza un objeto en la escena, es necesario comprobar el valor calculado con respecto a la velocidad real a la que se desplaza. En la figura 6.6 se pueden observar los resultados obtenidos.

	Representación correcta de la trayectoria	**Representación incorrecta de la trayectoria**	**Representación correcta del recorrido**	**Representación incorrecta del recorrido**	**Representación correcta de la velocidad**	**Representación incorrecta de la velocidad**
Vehículo	**5**	**0**	**5**	**0**	**5**	**0**
Peatón	**5**	**0**	**5**	**0**	**5**	**0**
Porcentaje Vehículo	**100**	**0**	**100**	**0**	**100**	**0**
Porcentaje Peatón	**100**	**0**	**100**	**0**	**100**	**0**

FIGURA 6.6: Resultados utilizando el entorno virtual 3D.

Los resultados obtenidos en el entorno virtual con respecto a la representación de la trayectoria, recorrido y velocidad son idílicas, ya que la escena virtual se ha utilizado para la realización de sucesivas pruebas de cada uno de los distintos submódulos que componen la aplicación. De ahí el fabuloso comportamiento de *Atravel*.

A continuación, en la figura 6.7 se muestran los resultados obtenidos sobre la clasificación de los objetos según su comportamiento con respecto a los patrones existentes

en la base de conocimiento. Esta clasificación permite obtener el tipo de comportamiento del objeto sobre el entorno de monitorización. El comportamiento de un objeto puede ser clasificado como normal, anómalo o no catalogado. La clasificación se realiza atendiendo a la similitud existente entre la trayectoria definida por la detección de los objetos, la similitud existente entre el recorrido realizado por cada objeto y la similitud con respecto a la velocidad con la que se desplzan. El correcto comportamiento de análisis de la similitud por trayectoria, se basa en la realización de una comparación entre la trayectoria del objeto detectado y cada uno de los patrones de la base de datos. Para comprobar, que la similitud por trayectoria se ha realizado de manera correcta se ha chequeado visualizando la representación 3D de la trayectoria del objeto junto con la representación 3D de la trayectoria de cada uno de los patrones. De forma, que se considera el cálculo de la similitud incorrecto cuando en la representación 3D de las trayectorias aparece que tienen una similitud muy significativa y sin embargo el valor de similitud refleja lo contrario, o viceversa. De igual modo, se realiza un chequeo del recorrido de un objeto detectado con los patrones existentes en la base de conocimiento y se comprueba si el valor de similitud se corresponde con el calculado. La similitud por recorrido comprueba únicamente la similitud entre BB clave del recorrido, con lo que puede ocurrir que aunque no exista suficiente similitud por recorrido, debería de encontrar un mayor grado de similitud. Esto es considerado como fallo en el cálculo de la similitud por recorrido. En cuanto a la similitud por velocidad, se comprueba el valor de la velocidad del objeto en cada uno de los tramos que dividen el movimiento del objeto con respecto a los patrones almacenados en la base de conocimiento. El cálculo de la similitud por velocidad es incorrecto, en caso de que el sistema detecte que no se desplazan a la misma velocidad, cuando en realidad sí se desplazan a la misma velocidad, y viceversa.

También es necesario realizar la comprobación de resultados correspondientes a la buena clasificación o no del comportamiento de un objeto, de manera, que puede suceder que un objeto tenga una gran similitud por velocidad y recorrido con respecto a un patrón, pero muy mala similitud por trayectoria, en este caso es importante comprobar el comportamiento de la aplicación y la correcta clasificación, en este caso el comportamiento del objeto debería ser clasificado como no catalogado. Por lo que se han comprobado los resultados tras la clasificación.

Como se puede observar en la figura 6.7 los resultados obtenidos son muy buenos, ya que se trata de una escena virtual elaborada con fin de probar los submódulos de similitud y la clasificación del comportamiento de un objeto.

Por último, también se han obtenido resultado en referencia a la fusión de patrones. Se han estudiado los resultados obtenidos de acuerdo al número de fusiones realizadas, si

	Similitud por trayectoria correcta	Similitud por trayectoria incorrecta	Similitud por recorrido correcta	Similitud por recorrido incorrecta	Similitud por velocidad correcta	Similitud por velocidad incorrecta
Vehículo	**5**	**0**	**5**	**0**	**5**	**0**
Peatón	**5**	**0**	**5**	**0**	**5**	**0**
Porcentaje Vehículo	**100**	**0**	**100**	**0**	**100**	**0**
Porcentaje Peatón	**100**	**0**	**100**	**0**	**100**	**0**

FIGURA 6.7: Resultados utilizando el entorno virtual 3D.

las fusiones realizadas son las correctas o no y también se ha analizado el número total de patrones almacenados. Se ha considerado que una fusión se ha realizado incorrectamente, cuando la representación en 3D resultante se ha realizado de forma correcta y existía suficiente similitud entre la representación de los dos patrones. De forma, que si se realiza la fusión de dos patrones que no alcanzan el grado de similitud correspondiente se considera una fusión incorrecta.

En la figura 6.8 se pueden observar los resultados obtenidos en referencia a la fusión de patrones.

	Nº de patrones fusionados Correctamente	Nº de patrones fusionados Incorrectamente	Nº máximo de patrones almacenados	Nº final de patrones almacenados
Vehículo	**2**	**0**	**3**	**3**
Peatón	**3**	**0**	**2**	**2**
Porcentaje Vehículo	**100**	**0**	-	-
Porcentaje Peatón	**100**	**0**	-	-

FIGURA 6.8: Resultados utilizando el entorno virtual 3D.

Como se puede observar en la figura 6.8 el comportamiento del sistema es bastante bueno.

Además se han obtenido resultados de *Atravel* utilizando 5 instancias positivas y 5 instancias negativas.

Los resultados obtenidos son:

- *VPR*: Sensibilidad o razón de verdaderos positivos.

$$VPR = \frac{VP}{V} = \frac{VP}{VP+FN} = \frac{5}{5+0} = 1.00 \tag{6.4}$$

		Total
Verdadero Positivo (VP) = 5	**Falso Positivo (FP) = 0**	**5**
Falso Negativo (FN)= 0	**Verdadero Negativo (VN) = 5**	**5**
5	**5**	**10**

FIGURA 6.9: Resultados de la aplicación en el entorno virtual.

- *FPR*: Sensibilidad o razón de falsos negativos.

$$FPR = \frac{FP}{N} = \frac{FP}{FP+VN} = \frac{0}{50+0} = 0.00 \tag{6.5}$$

- *ACC*: Precisión o exactitud.

$$ACC = \frac{(VP+VN)}{(P+N)} = \frac{50+50}{100} = 1.00 \tag{6.6}$$

A continuación en la figura B.8, se realiza una representación gráfica de los resultados obtenidos utilizando un entorno virtual.

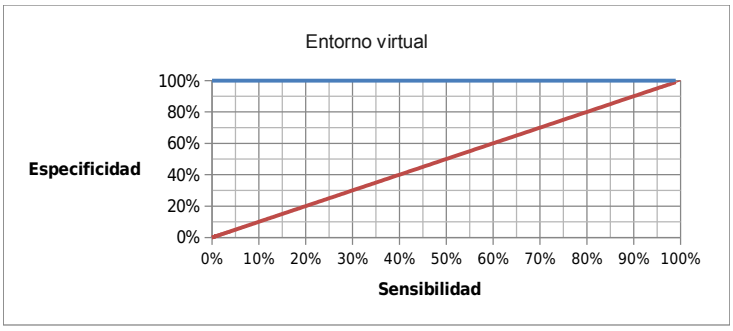

FIGURA 6.10: Gráfico de curva ROC en el entorno virtual (ver imagen a color en Anexo B).

Los resultados obtenidos utilizando *Atravel* son de caracter orientativo y puede que la aplicación se comporte de diferente manera ante circunstancias diferentes.

6.2.2. Resultados con entorno real

La realización de pruebas en un entorno real se ha desplegado el sistema utilizando cámaras sobre una escena que representa un cruce de calles situado en el municipio de *Almodóvar del Campo*. Para la captación de las imágenes se han utilizado dos cámaras

Logitech AF-Sphere colocadas en diferentes perspectivas con el objetivo de complementar el entorno de monitorización. Las tareas de posicionamiento de las cámaras, calibración, segmentación y tracking, se han llevado a cabo utilizando la aplicación **TraceMon** realizada por *David García Bermejo* como proyecto fin de carrera.

Descripción de la escena real

La escena de la figura B.9 representa el cruce de dos de las vías más transitadas de la población. En ella se pueden detectar, tanto movimientos de vehículos como movimientos de peatones. Lo que permite ser una prueba de suficiente complejidad para la obtención de resultados por parte de la aplicación. En la figura B.9 se puede observar la escena real monitorizada.

FIGURA 6.11: Entorno real monitorizado (ver imagen a color en Anexo B).

En ella se puede interpretar los diferentes comportamientos correctos de los objetos atendiendo a su tipología. Dados los dos tipos de objetos que *Atravel* reconoce, su comportamiento puede ser:

- **Vehículos**: El comportamiento normal de todos los objetos de tipo vehículo conlleva a la realización de los siguientes movimientos (tomando como referencia la figura B.11):
  - Desplazamiento de vehículos desde el punto **1** al punto **2**.
  - Movimiento de objetos de tipo vehículo desde el punto **1** al punto **4**.
  - Desplazamiento de vehículos desde el punto **3** al punto **4**.
  - Movimiento de objetos de tipo vehículo desde el punto **3** al punto **2**.

  Cualquier vehículo detectado en la escena debe desplazarse por la calzada de las imágenes y en la dirección especificada anteriormente, ya que la detección de objetos

FIGURA 6.12: Comportamiento de vehículos en la escena monitorizada (ver imagen a color en el Anexo B).

de tipo vehículo en cualquier otra dirección será detectado como comportamiento anómalo.

- **Peatones**: El desplazamiento de objetos a lo largo de la escena monitorizada tiene lugar sobre las aceras y paso de cebra de la escena monitorizada. Un correcto comportamiento de los peatones viene determinado por las siguientes acciones (véase la figura B.12):

FIGURA 6.13: Comportamiento de peatones en la escena monitorizada (ver imagen a color en Anexo B).

  - Desplazamiento del peatón desde el punto **1** al punto **2** y al punto **3**.
  - Movimiento de un peatón desde el punto **1** al punto **2** hasta el punto **5**, pasando por el punto **4**.
  - Desplazamiento de un peatón desde el punto **3** al punto **2** hasta el punto **1**.
  - Movimiento de un peatón desde el punto **3**, hasta el punto **5**, pasando por los puntos **2** y **4**.

- Movimiento de un peatón desde el punto **5**, hasta el punto **3**, pasando por los puntos **4** y **2**.
- Desplazamiento del peatón desde el punto **1** hasta el punto **9**, pasando por los puntos **2** y **4**.
- Movimiento de un peatón desde el punto **5**, hasta el punto **1**, pasando por los puntos **4** y **2**.
- Desplazamiento de un peatón desde el punto **5** hasta el punto **9**, pasando por el punto **4**.
- Desplazamiento de peatones desde el punto **3** hasta el punto **9** pasando por los puntos, **2** y **4**
- Movimiento de peatones desde el punto **9** hasta el punto **5**, pasando por **4**.
- Movimiento de peatones desde el punto **9** hasta el punto **3**, pasando por **4** y **2**.
- Movimiento de peatones desde el punto **9** hasta el punto **1**, pasando por **4** y **2**.
- Desplazamiento de peatones desde el punto **9** hasta el punto **6**, pasando por **7**.
- Desplazamiento de peatones desde el punto **9** hasta el punto **8**.
- Desplazamiento de peatones desde el punto **6** hasta el punto **9**, pasando por **7**.
- Desplazamiento de un peatón desde el punto **6** hasta el punto **8**, pasando por **8**.
- Movimiento de peatones desde el punto **8** hasta el punto **6**, pasando por **7**.
- Movimiento de peatones desde el punto **8** hasta el punto **9**.

Resultados obtenidos utilizando la escena real

A continuación, se van a mostrar los resultados obtenidos tras la realización de una prueba del sistema en un entorno de monitorización real. La prueba ha constado de 15000 frames de vídeo. En ellos, se pueden distinguir el movimiento de 100 objetos entre vehículos y peatones. En concreto se han contabilizado la existencia 67 objetos de tipo peatón y 33 de tipo vehículo.

Al igual que en los resultados obtenidos utilizando una escena virtual, se ha llevado a cabo una obtención de los resultados correspondientes al número de movimientos de objetos detectadas correctamente. Se ha considerado la detección de una trayectoria incorrecta, por ejemplo, cuando una persona sólo es detectada la parte del tronco, con las extremidades inferiores detectadas por otro lado, detección de un objeto por reflejo en los cristales existentes en la escena, etc.

	Representación correcta de la trayectoria	**Representación incorrecta de la trayectoria**
Vehículo	**30**	**3**
Peatón	**61**	**6**
Porcentaje Vehículo	**90,1**	**9,9**
Porcentaje Peatón	**91,04**	**8,96**

FIGURA 6.14: Resultados obtenidos utilizando un entorno real.

Los resultados obtenidos se pueden observar en la figura 6.14.

Como se puede observar en la figura 6.14, existe el doble de detecciones erróneas de objetos de tipo peatón que de tipo vehículo esto es así, debido a que la detección correcta de peatones conlleva más dificultad que la de vehículo. De todas formas, los errores detectados aquí son arrastrados de etapas anteriores a *Atravel*.

Al igual que en los resultados de la escena virtual, se han obtenido resultados acerca de la representación de trayectoria, recorrido y velocidad de los objetos detectados. La determinación de una representación incorrecta que se ha utilizado ha sido la explicada en la sección anterior 6.2.1. La detección de representaciones incorrectas se corresponden por ejemplo, a casos en los que un peatón modifica su trayectoria porque se da media vuelta de repente, o porque se produce oclusión parcial o agrupamiento de personas. En el caso de los vehículos los errores en la representación pueden darse al realizar un aparcamiento y dar marcha atrás, estacionamiento fortuito, etc. En la figura 6.15 se pueden observar los resultados obtenidos sobre esta funcionalidad.

	Representación correcta de la trayectoria	**Representación incorrecta de la trayectoria**	**Representación correcta del recorrido**	**Representación incorrecta del recorrido**	**Representación correcta de la velocidad**	**Representación incorrecta de la velocidad**
Vehículo	**32**	**1**	**32**	**1**	**30**	**3**
Peatón	**62**	**5**	**60**	**7**	**62**	**5**
Porcentaje Vehículo	**96,96**	**3,04**	**96,96**	**3,04**	**90,9**	**9,1**
Porcentaje Peatón	**92,53**	**7,47**	**89,55**	**10,45**	**92,53**	**7,47**

FIGURA 6.15: Resultados obtenidos utilizando un entorno real.

Como se puede observar el comportamiento del sistema para la representación de trayectorias y recorridos es mejor para vehículos que para peatones, esto se debe a que la representación de un vehículo presenta menos dificultad que la representación de la actividad humana. Éste presenta comportamientos más complejos que dificultan la labor de representación 3D. Sin embargo, en la representación de la velocidad, presenta

menor porcentaje en vehículos. A pesar de todo, el comportamiento del sistema para la representación de trayectorias, recorridos y velocidades es bastante bueno.

Los resultados obtenidos con respecto a la funcionalidad de clasificación del comportamiento de los objetos en el entorno de monitorización, se pueden observar en la figura 6.16

	Similitud por trayectoria correcta	Similitud por trayectoria incorrecta	Similitud por recorrido correcta	Similitud por recorrido incorrecta	Similitud por velocidad correcta	Similitud por velocidad incorrecta
Vehículo	**31**	**2**	**31**	**2**	**31**	**2**
Peatón	**58**	**9**	**57**	**10**	**58**	**9**
Porcentaje Vehículo	**93,93**	**6,07**	**93,93**	**6,07**	**93,93**	**6,07**
Porcentaje Peatón	**86,56**	**13,44**	**85,07**	**14,93**	**86,56**	**13,44**

FIGURA 6.16: Resultados obtenidos utilizando un entorno real.

Como se puede apreciar en la figura 6.16 los resultados obtenidos en cuanto a la similitud para la posterior clasificación han sido más bajos que los resultados obtenidos en otras funcionalidades del sistema. Esto es debido a que la clasificación se realiza con respecto a los patrones almacenados en la base de conocimiento. Y la similitud existente con éstos depende de lo buenos que sean estos patrones. Cuanto mayor tiempo de monitorización se realice mejores serán los resultados en este apartado, ya que serán detectados un mayor número de objetos que permitan al sistema aprender de ellos y optimizar los patrones existentes. En la realización de las pruebas no se ha podido realizar una monitorización muy extensa por motivos de tiempo principalmente. Sin embargo, los resultados obtenidos en referencia a la similitud son considerados como bastante buenos. Como principal apreciciación destacar que el sistema se comporta mejor para la clasificación de vehículos que para la clasificación del comportamiento de peatones.

Por útlimo, se han obtenido los resultados correspondientes a la funcionalidad de fusión de patrones. El criterio utilizado para obtención de los resultados ha sido el mismo al detallado en la sección 6.2.1. Los resultados obtenidos se encuentran en la figura 6.17

Conforme a los resultados de la figura 6.17 se puede obtener que el número de fusiones correctas en cuanto a vehículos es muy superior (en proporción) al número de fusiones realizadas en objetos de tipo peatón. Además ninguna de las fusiones realizadas entre patrones de tipo vehículo se ha realizado incorrectamente. Sin embargo, en cuanto a patrones de tipo peatón sí que ha habido algunos errores en la realización de la fusión. De forma, que dados los datos de la tabla se puede extraer como conclusión que los patrones existentes en la base de conocimiento, se encuentran algo más optimizados ya

	Nº de patrones fusionados Correctamente	Nº de patrones fusionados Incorrectamente	Nº máximo de patrones almacenados	Nº final de patrones almacenados
Vehículo	**25**	**0**	**11**	**5**
Peatón	**18**	**4**	**38**	**23**
Porcentaje Vehículo	**100**	**0**	-	-
Porcentaje Peatón	**81,81**	**18,19**	-	-

FIGURA 6.17: Resultados obtenidos utilizando un entorno real.

que se presentan únicamente cinco patrones en la base de conocimiento, mientras que los patrones de tipo peatón almacenados han sido 23.

Al igual que para el cálculo de los resultados obtenidos sobre una escena virtual, se ha utilizado las **Curvas ROC**. Los resultados obtenidos tras la ejecución de *Atravel* sobre la escena real se pueden observar en la figura 6.18.

Considerese que para la obtención de resultados de *Atravel* se han utilizado 50 instancias positivas y 50 instancias negativas.

		Total
Verdadero Positivo (VP) = 45	**Falso Positivo (FP) = 7**	**52**
Falso Negativo (FN) = 5	**Verdadero Negativo (VN) = 43**	**48**
50	**50**	**100**

FIGURA 6.18: Resultados obtenidos utilizando un entorno real.

Los resultados obtenidos son:

- *VPR*: Sensibilidad o razón de verdaderos positivos.

$$VPR = \frac{VP}{V} = \frac{VP}{VP+FN} = \frac{45}{45+5} = 0.9 \tag{6.7}$$

- *FPR*: Sensibilidad o razón de falsos negativos.

$$FPR = \frac{FP}{N} = \frac{FP}{FP+VN} = \frac{7}{7+43} = 0.14 \tag{6.8}$$

- *ACC*: Precisión o exactitud.

$$ACC = \frac{(VP+VN)}{(P+N)} = \frac{45+43}{100} = 0.88 \tag{6.9}$$

El gráfico representativo de los resultados obtenidos tras la ejecución de la aplicación en la escena real se puede ver en la figura 6.19.

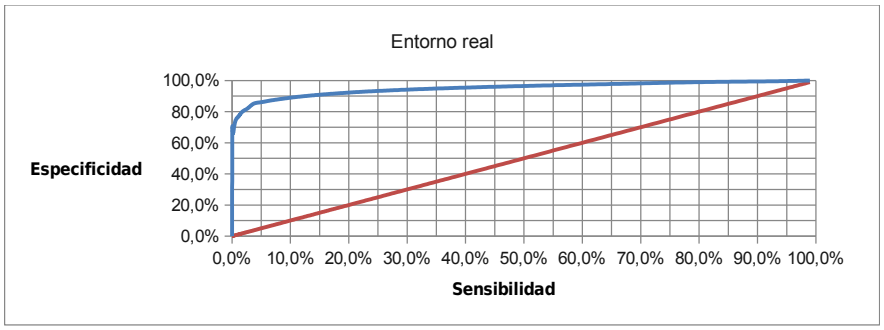

FIGURA 6.19: Curva ROC resultante utilizando entorno real (ver imagen a color en Anexo B).

Anomalías destacadas

- Desplazamiento de un vehículo sobre la calzada en dirección contraria. Por ejemplo, (observando la figura B.11) desplazamiento de un vehículo desde el punto **4** a **3**, desde **4** a **1**, desde **4** a **2**, desde **3** a **1**, desde **1** a **3**, desde **2** a **3**, desde **2** a **1**, desde **2** a **3** o desde **2** a **4**.
- Movimiento de un vehículo por la calzada a velocidad superior que la registrada en la base de conocimiento para ese movimiento.
- Desplazamiento de un peatón por zonas diferentes a las indicadas en la sección 6.2.2. Como por ejemplo y teniendo como referencia la figura B.12, un peatón se desplaza desde el punto **4** al punto **7** (o viceversa), por mitad de la calzada. También se puede dar un desplazamiento desde el punto **7** al punto **2** (o viceversa).
- Desplazamiento de un peatón por mitad de la calzada y a una velocidad superior de lo normal (corriendo). Por ejemplo, desplazamiento de un peatón desde el punto **9** al punto **2** directamente y corriendo.

En general, los resultados obtenidos utilizando *Atravel* han sido bastante buenos (88 %). Como conclusión, se obtiene que se comporta mejor en el análisis de comportamiento de objetos de tipo vehículo que para el comportamiento de peatones.

7

Conclusiones y propuestas

Se ha desarrollado un componente independiente que se encarga del análisis de comportamientos de objetos en un entorno de monitorización. Debido a su independencia con respecto a las distintas etapas que componen un sistema de vigilancia puede ser integrado en cualquier sistema de vigilancia. Además, puede ser utilizado en sistemas de vigilancia multicámara.

Para estudiar el comportamiento de los objetos, *Atravel* realiza un estudio acerca de su trayectoria y velocidad. El análisis de su trayectoria es realizado en 3D, lo que permite obtener con exactitud el comportamiento del objeto sobre el entorno observado y el análisis de velocidades se realiza por fragmentos en los que el objeto es detectado en la escena. Este tipo de análisis dotan a la aplicación de máxima precisión en la detección de situaciones anómalas.

El sistema aprende del comportamiento de los objetos y dependiendo de su frecuencia permite clasificar su comportamiento en tres categorías, *NORMAL*, *ABNORMAL* y *NOT_CLASSIFIED*. Esta clasificación viene precedida por una primera etapa de clasificación del comportamiento de los objetos sobre el entorno de monitorización. Gran parte del correcto funcionamiento de la aplicación, se corresponde con la realización de una buena clasificación. La clasificación de los objetos se realiza atendiendo a la trayectoria 3D y la velocidad del objeto. En cuanto a la trayectoria, además de realizar un estudio estricto de la representación de la trayectoria en 3D se realiza un estudio del recorrido del objeto. El recorrido de un objeto viene determinado por la zona en la que se produce el desplazamiento de un objeto, sin tener en cuenta aquellos desplazamientos suaves en la trayectoria realizada por el objeto. El objetivo de la utilización del recorrido de un objeto es obtener de manera

más rápida, los diferentes comportamientos existentes en la escena de monitorización. Tras la correcta clasificación del comportamiento de un objeto, éste pasa formar parte de la base de conocimiento de patrones. También realiza un estudio de la velocidad con la que se desplazan los objetos por fragmentos o tramos, proporcionando información muy precisa acerca de la velocidad en cada zona del entorno de monitorización.

Además, para optimizar la clasificación y la base de conocimiento, *Atravel* incorpora un mecanismo de fusión de los patrones adquiridos sobre la escena. Con el objetivo de que en la base de conocimiento de patrones aparezcan bien diferenciados los diferentes comportamientos de objetos sobre el entorno, a lo largo del periodo de monitorización. Una vez optimizada la base de conocimiento, el sistema detectará aquellas situaciones anómalas e informará visualmente de su detección.

En conclusión, *Atravel* es una aplicación encargada de llevar a cabo un aprendizaje de trayectorias en 3D y velocidades para detección de situaciones anómalas en entornos reales.

A continuación, se detallan los objetivos alcanzados en el desarrollo de *Atravel* (Aprendizaje Automático de Trayectorias y Velocidades de en entornos monitorizados para la detección de situaciones anómalas) junto con la proposición de algunas mejoras que se pueden incorporar a *Atravel* en el futuro.

7.1. Objetivos alcanzados

1. **Representación multicámara del entorno de monitorización mediante flujo de vídeo**: Para resolver este objetivo se ha creado en la interfaz gráfica de usuario, varios componentes para la visualización de flujo de vídeo, procedentes de la monitorización de una escena desde diferentes puntos de vista. La representación de multicámara del entorno se lleva a cabo en la interfaz gráfica de usuario denominada como *Visualización de vídeo real*. Para la visualización de los flujos de vídeo es llevada a cabo, gracias a la creación del objeto *cvCreateFileCapture* de *OpenCV*. De esta forma el usuario puede visualizar el entorno de monitorización, al mismo tiempo que se detectan los diferentes objetos en la escena. De modo que la visualización de la escena desde los diferentes lugares donde se encuentran situadas las cámaras se lleva a cabo mediante la reproducción de los flujos de vídeo grabados por las cámaras anteriormente.

2. **Representación del entorno de monitorización 3D**: La representación de los objetos que aparecen en el entorno de monitorización se lleva a cabo en un componente situado en la interfaz gráfica de usuario de *Visualización de vídeo real*. El componente de visualización es de tipo *OpenGL*. Permite la representación en 3D de los diferentes objetos detectados en la escena observada. Para ello, se ha dibujado un plano en 3D que hace las funciones de suelo por el que tiene lugar el desplazamiento de los objetos detectados en la escena. Los objetos son representados como cajas tridimensionales que van modificando su posición conforme se desplazan a lo largo de la escena. Este componente incorpora las funciones de *rotación, traslación* y *zoom*. A través, de estas operaciones el usuario puede modificar la perspectida de visualización de la escena y amoldarla a la perspectiva que más desee. Para la rotación de la escena se utiliza el botón izquierdo del ratón presionándolo y moviéndolo en las diferentes posiciones y al soltar el botón se desactiva la rotación. La funcionalidad de *zoom* se consigue moviendo la ruleta del ratón hacia adelante para acercarse a la escena y hacia atrás para alejarse de la escena. Y para la traslación de la escena se ha realizado, pulsando el botón de la ruleta y moviendo el ratón hacia la posición deseada, cuando el usuario encuentre la posición adecuada soltará el botón pulsado.

3. **Almacenamiento de la información de patrones y objetos detectados**: El almacenamiento de la información tanto de los objetos detectados en cada instante de tiempo como de los patrones aprendidos se lleva a cabo en estructuras de datos denominadas *Diccionario*. Los principales motivos de utilización de esta estructura

de datos es su flexibilidad y el rápido acceso a sus elementos. Se ha utilizado un diccionario para albergar a todos los elementos detectados en la escena en cada instante de tiempo, este diccionario hace referencia a la *base de conocimiento de objetos detectados*. En éste se almacena toda la información referente al objeto junto con el grado de similitud existente con cada uno de los patrones que constituyen la *base de conocimiento de patrones aprendidos*. Para el almacenamiento de la información referente a la similitud con los diferentes patrones también se ha utilizada la estructura de *Diccionario*.

En el almacenamiento de los de la información referente a los patrones aprendidos, tras la monitorización de algunos objetos se ha utilizado la estructura de datos de *Diccionario*. En ésta se almacena toda la información referente almacenada en *Base de conocimiento de patrones aprendidos*. En esta base de conocimiento se encuentran almacenados aquellas propiedades necesarias para la creación de patrones.

4. **Definición de patrones**: Este objetivo ha sido llevado a cabo, para poder llevar a cabo la clasificación de los distintos comportamientos de los objetos detectados en la escena. Para ello, se ha definido una estructura de datos denominada *Trajectory_-pattern*, en la cuál se almacena todas las propiedades que debe almacenar un patrón. Entre las propiedades, almacenadas, se encuentra:

   - La trayectoria que representa su movimiento a lo largo de la escena.
   - El recorrido de su desplazamiento sobre el entorno.
   - El tipo de patrón que es, bien *VEHICLE* o *PEDESTRIAN*.
   - El comportamiento que tiene sobre la escena, *NOT_CLASSIFIED*, *NORMAL* o *ABNORMAL*.
   - Fecha de creación del patrón.

5. **Clasificación de patrones**: Este objetivo ha sido alcanzado utilizando el grado de similitud entre un patrón determinado y el comportamiento del objeto sobre el entorno de monitorización. Además, un patrón puede ser automáticamente clasificado dependiendo del número de veces que se repita su comportamiento en la escena. Existen tres posibles clases en la que puede quedar clasificado un patrón, *NOT_-CLASSIFIED, NORMAL, ABNORMAL*. Si existen cinco patrones que tienen un comportamiento muy similar, entonces el sistema clasifica a estos cinco patrones como comportamiento normal. En caso de pasar demasiado tiempo un patrón clasificado como *NOT_CLASSIFIED* es clasificado como *ABNORMAL*. Inicialmente si un patrón no tiene afinidad con ninguno de los patrones existentes en la *base de*

conocimiento de patrones es clasificado del tipo *NOT_CLASSIFIED*. La clasificación de patrones se puede visualizar en la interfaz gráfica de usuario denominada *visualización de patrones y objetos*, en particular en la pestaña de patrones.

6. **Aprendizaje supervisado**: En caso de que el usuario no quiera esperar a que el sistema clasifique a una trayectoria como *NORMAL* o *ABNORMAL*,bien porque no quiere esperar a que aparezcan nuevos objetos que tengan un comportamiento similar o bien porque no quiere esperar el tiempo necesario en que un patrón pasa a ser de tipo *ABNORMAL*. Existe una pestaña en la interfaz gráfica de usuario denominada *visualización de patrones y objetos* que permite modificar la categoría con la que están clasificados lo patrones, pudiendo clasificar los patrones de tipo *NORMAL* y *ABNORMAL*.

7. **Similitud con patrones**: Para la obtención de este objetivo se han cumplido los subobjetivos, de similitud atendiento a la trayectoria, recorrido y velocidad.

   La consecución del subobjetivo *similitud por trayectoria* se ha realizado mediante la implementación de un algoritmo que permite conocer el grado de similitud entre la trayectoria de cada uno de los patrones con la trayectoria del objeto recién detectado en la escena de monitorización cuyo comportamiento se pretende clasificar. El grado de similitud por trayectoria con respecto a los diferentes patrones puede observarse en el campo de similitud por trayectoria situado en la interfza gráfica de usuario denominada *visualización de patrones y objetos* concretamente en la pestaña de visualización de objetos y en cada objeto aparece la barra que contiene el porcentaje de similitud por trayectoria con los diferentes patrones con los que existe similitud.

   La realización del subobjetivo *similitud por recorrido* se ha realizado mediante la implementación de una función que permite conocer el grado de similitud existente entre el recorrido de un patrón de la *base de conocimiento de patrones aprendidos* y el recorrido realizado por un objeto que se desplaza por la escena de monitorización. La visualización de la similitud por recorrido, puede ser visualizada por el usuario en la interfaz gráfica de usuario denominada *visualización de patrones y objetos*, concretamente en la pestaña de objetos y en la columna de similitud por recorrido, aparece una barra donde se incluye el porcentaje de similitud con los patrones de la *base de conocimiento de patrones aprendidos*.

   La obtención del subobjetivo de *similitud por velocidad* se ha llevado a cabo a través de la implementación de un algoritmo que permite obtener el grado de similitud existente entre la velocidad de un patrón con el que existe cierta afinidad y un objeto detectado en la escena monitorizada. Para calcular el grado de similitud por velocidad,

en primer lugar debe existir un alto grado de similitud por trayectoria y recorrido, ya que si no existe similitud en el movimiento del objeto no es posible calcular el grado de similitud por velocidad, ya que si el movimiento del objeto no es similar al de un patrón, entonces no tiene sentido calcular su similitud por velocidad. La similitud por velocidad puede ser comprobada por el usuario visualizando la interfaz gráfica de usuario denominada *visualización de patrones y objetos*, más concretamente en la pestaña de objetos y en la columna de similitud por velocidad. En la cuál se puede observar una barra que contiene el porcentaje de similitud con el patrón correspondiente.

8. **Fusión de patrones**: La consecución de este objetivo se ha obtenido mediante la construcción de las funciones necesarias para realizar la fusión patrones en concreto. Se realiza la fusión de dos patrones cuyo grado de similitud es elevado. Esta función se denomina *add_fusion_path* y se encuentra en la estructura de datos denominada *Knowledge_base*.

9. **Detección de situaciones anómalas**: La resolución de este objetivo está muy relacionado con la clasificación de los patrones en las diferentes categorías. El comportamiento de un objeto es chequeado con cada uno de los patrones existentes en la *base de conocimiento de patrones aprendidos* y dependiendo de su afinidad con cada uno de los patrones, así los clasifica como *NOT_CLASSIFIED*, *NORMAL* o *ABNORMAL*. En caso de que la mayor afinidad la tiene con patrones de tipo *NOT_CLASSIFIED* pues el comportamiento del objeto recién detectado sobre la escena será clasificado de tipo *NOT_CLASSIFIED*. En caso de que el patrón con el que existe mayor grado de similitud es de tipo *NORMAL* pues es clasificado como *NORMAL* y en caso de que el patrón con el que tiene mayor afinidad es de tipo *ABNORMAL*, entonces es clasificado de tipo *ABNORMAL*. La forma de avisar de la existencia de situaciones anómalas es mediante la representación visual de la trayectoria de un objeto en la escena. Ya que el comportamiento de un objeto se asemeja al de un patrón clasificado de tipo *ABNORMAL* es representado de color *rojo*. Mientras que se es de tipo *NOT_CLASSIFIED* se representa de color *ambar* y si es de tipo *NORMAL* es representado de color *verde*. Además *Atravel* también informa del movimiento de un objeto a una velocidad superior o inferior con respecto a la velocidad de los patrones almacenado, informando de la misma mediante una señal de advertencia.

10. **Interfaz gráfica parcialmente desacoplada**: *Atravel* cuenta con una interfaz gráfica de usuario (UI) que permite visualizar al usuario toda la información referente al

análisis del comportamiento de los objetos en el entorno de monitorización. Así pues, cuenta con la implementación de dos ventanas diferentes que constituyen la UI.

- **Ventana de visualización de flujo de vídeo**: Esta ventana está enfocada a la representación y visualización de la escena monitorizada, lo que incluye la visualización del flujo de vídeo del entorno junto con la visualización den 3D de los objetos que se desplazan a lo largo de la escena.
- **Ventana de visualización de patrones y objeto**: En esta ventana el usuario puede observar la representación del comportamiento del objeto a lo largo de su desplazamiento sobre la escena. En esta ventana se puede realizar la visualización de patrones o de los objetos existentes en cada instante de tiempo en la escena.

Además, el módulo de análisis de comportamiento de objetos está desacoplado de la interfaz gráfica pudiendo ser utilizado en otros proyectos.

11. **Escalabilidad**: *Atravel* permite ser utilizado en entornos de monitorización con múltiples dispositivos de captación de vídeo. Además permite ser utilizado en cualquier entorno de monitorización ya sea más complejo o más sencillo.

12. **Estándares libres**: Todas las tecnologías y entornos de desarrollo utilizados en la implementación de *Atravel* tienen licencia *GPL* con el objetivo de declarar la aplicación como de libre utilización. De este modo, *Atravel* puede ser utilizado por la comunidad de usuarios de *Software Libre* para la incorporación de nuevos módulos o introducción de mejoras con el objetivo de hacer cada vez mejor la aplicación.

7.2. Propuestas de trabajo futuro

En esta sección se exponen algunas modificaciones posibles a realizar en un futuro, así como la incorporación de funcionalidad adicional que permita complementar la aplicación mejorando las prestaciones actuales.

- **Indexación y recuperación de información multimedia**: permitiría la reproducción del comportamiento de un objeto determinado a lo largo de la escena. Esta funcionalidad sería totalmente novedosa que sería de gran utilidad, en la aplicación ya que permite reproducir aquellos comportamiento anómalos para revisar y obtener información detallada acerca de posibles sucesos negativos, como robos, accidentes, sanciones de tráfico, etc. Se podrían incluir funciones de parada (pause), terminado

(stop), comienzo (play) y rebobinado de flujo de vídeo. La introduccion de esta mejora en el sistema, permitiría la utilización de la aplicación en sistemas de vigilancia forense, ya que no sería necesario la monitorización por parte de personal de seguridad, sino que las situaciones anormales serían almacenadas en formato de vídeo, para su posterior visualización.

- **Reconocimiento facial**: Permite el reconocimiento de la identidad de los objetos de tipo persona. Para la creación de este módulo sería necesario contar con cámaras de vídeo con una alta resolución que permitan detectar la cabeza de los individuos y realizando zoom sobre éstas poder obtener los rasgos y dimensiones faciales de los objetos. Además al incorporar una base de datos en la que se especifique una fotografía de cada persona podría facilitar la tarea de búsqueda de terroristas, ladrones más buscados, personas que han realizado actos fraudulentos de gran envergadura, etc. Además a nivel de empresa, podría ser utilizado para la inclusión de intrusos realizando un reconocimiento facial de los trabajadores a través de una fotografía almacenada con los datos que definen sus rasgos faciales. El desarrollo de éste módulo es muy complejo ya que no sólo afecta a la etapa de comportamiento de objetos sino que además salpica a otras etapas como la de detección, y clasificación, ya que debe realizar una detección específica de la cabeza de los peatones.

- **Reconocimiento de matrículas de vehículos**: La inclusión de esta funcionalidad a la aplicación implica la incorporación de un nuevo módulo totalmente distinto a las funcionalidades que aporta la aplicación. Ésta debería incorporar un módulo que se encargue de la detección específica de las matrículas de los vehículos con el objetivo de descifrar los dígitos y letras que componen la matrícula de un vehículo determinado. Posteriormente con la detección del número de matrícula se relacionaría con una base de datos en la que se encuentren todas las matrículas y propietarios de los vehículos. Esta funcionalidad puede ser útil para el control del cumplimiento de las normas de tráfico en determinados puntos estratégicos. Además también puede ser utilizado en la identificación de criminales o personas en búsqueda por actos fraudulentos.

- **Exportación e imporación de la base de conocimiento de patrones en un archivo XML**: Esta funcionalidad aportaría a la aplicación la capacidad de almacenar de forma persistente el comportamiento de los objetos detectados en la escena, con el objetivo de poder cargar los patrones reconocidos en la escena para evitar el previo entrenamiento de la aplicación.

- **Creación de un interfaz gráfico para dispositivos móviles**: El objetivo de esta función es permitir la utilización de *Atravel* en dispositivos móviles. Esta funcionalidad aportaría flexibilidad y comodidad a la aplicación.

7.3. Conclusiones personales

Durante el estudio de la titulación de Ingeniería Informática se han estudiado metodologías y técnicas necesarias para el desarrollo de un proyecto, como son: captura de requisitos, análisis, diseño, implementación, pruebas y mantenimiento. De manera, que la realización de un proyecto de esta envergadura, permite la utilización de forma práctica de algunas de estas técnicas.

Durante la realización de la titulación, se han ido estudiando los diferentes conocimientos que abarca la *Ingeniería en Informática*. Sin embargo, durante el estudio de los diversos métodos y técnicas utilizados en el área de la *Inteligencia artifical*, causaron en mí un importante interés sobre este área. Quizás tuvo gran influencia en la elección de este proyecto, el ímpetu de complementar el proyecto *TraceMon*, realizado por mi hermano *David*. De forma, que *TraceMon* se encargue de las etapas de *posicionamiento, calibración, segmentación y tracking* y *Atravel* realice la parte de análisis de comportamientos de los objetos atendiendo a su trayectoria y su velocidad.

Durante el desarrollo de este proyecto se han aprendido métodos y técnicas desconocidas para mí, como representación de la trayectoria y recorrido de un objeto, la creación de patrones, construcción de algoritmos de similitud atendiendo a su trayectoria, recorrido y velocidad, fusión de patrones, etc. Al mismo tiempo que se han puesto en práctica conocimientos adquiridos durante el estudio de la titulación.

A lo largo de la titulación, se van adquiriendo diferentes técnicas y conocimientos muy importantes, para el futuro. Sin embargo, el continuo crecimiento del área relacionado con la informática, hace necesario el continuo estudio de nuevas tecnologías. Pero el aprendizaje de las diferentes tecnologías durante la titulación, permite adaptarnos más rápidamente a las nuevas tecnologías. Es en la realización del proyecto fin de carrera, donde se ponen en práctica los diferentes métodos y técnicas aprendidas, para la creación de un proyecto de mayor envergadura.

Además, supone la culminación de una etapa de aprendizaje, dotando de cierta experiecia al alumno para la inmersión de éste en el mundo laboral.

ANEXOS

A

Manual de referencia

A.1. Manual de usuario

En esta sección se realiza una explicación detallada del uso de la aplicación desarrollada.

En primer lugar, el usuario debe de introducir los vídeos correspondientes en el directorio de la aplicación, *./videos/*. El identificador de los vídeos almacenados debe ser *camera_+número de vídeo.avi*, el número de vídeo viene determinado por la cámara que ha realizado la visuailización del entorno. Por ejemplo, si se trata de un vídeo obtenido por la cámara 1 debe aparecer almacenado con el identificador *camera_1.avi*, si se trata del vídeo obtenido por la cámara 2 se llamará *camera_2.avi*.

Nada más ejecutar la aplicación, la interfaz gráfica que aparece corresponde a la vista real, de los vídeo correspondientes a la escena de monitorización. El aspecto inicial de la aplicación se puede ver en la figura A.1:

A continuación, para poder comenzar la visualización de los vídeos almacenados, es necesario clicar en la pestaña superior denominada *Cameras* y seleccionar el número de cámaras con las que se ha realizado la monitorización. En este caso se realizado con dos. (véase la figura A.2).

Una vez seleccionadas el número de cámaras que han sido utilizadas aparencen en la interfaz los distintos componentes donde se va a mostrar el flujo de vídeo en este caso se realiza en dos (véase figura A.3).

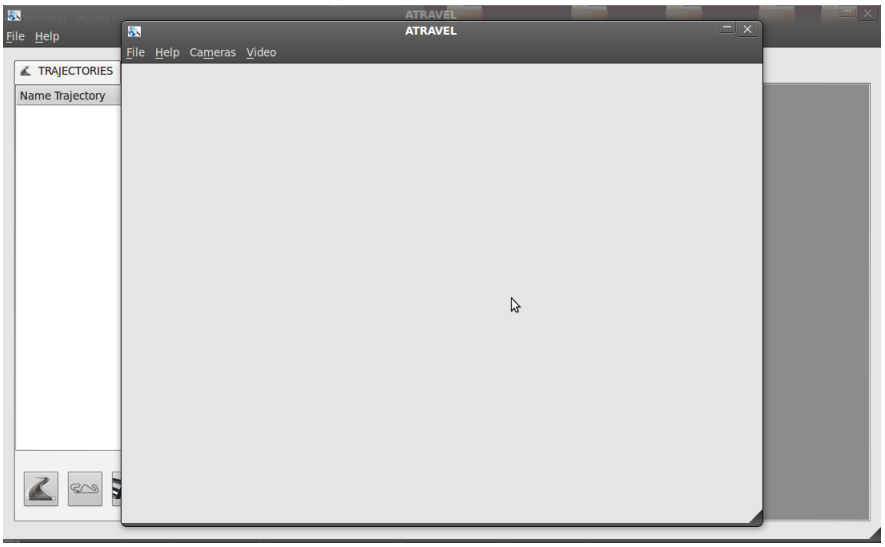

FIGURA A.1: Ventana inicial de *Atravel*.

También aparece un componente en esta interfaz, con el objetivo de poder representar en 3D (incluyendo las opciones de rotación, traslación y zoom) los objetos detectados en la escena y el usuario pueda visualizarlos (véase la figura A.4.

Para que el sistema comience a realizar la monitorización el usuario debe clicar en la pestaña *Vídeo* y seleccionar la opción de *Play Vídeo* (ver figura A.5).

El aspecto de la aplicación tras comenzar la monitorización es el que aparece en la figura A.6. Donde se pueden observar, cómo a medida que aparecen objetos en la escena son representados en 3D en el componente de visualización.

En esta interfaz tiene lugar todo lo relacionado con la visualización del vídeo real y su representación en 3D.

Por otro lado, se encuentra la interfaz principal, la cuál inicialmente presenta este aspecto (ver figura A.7):

En la ventana principal se pueden distinguir dos componentes: uno que el el componente de visualización 3D en el cuál se realiza la representación de todo lo que hace referencia a la detección de los objetos y patrones (ver figura A.8.

El otro componente que se puede diferenciar está compuesto por un arbol en el cuál se realiza la representación de la información perteneciente a los objetos detectados, los patrones reconocidos, la clasificación de patrones y archivo de información detallada. (ver figura A.9).

A continuación, se va a explicar cada una de las funcionalidades de la pestaña de patrones.

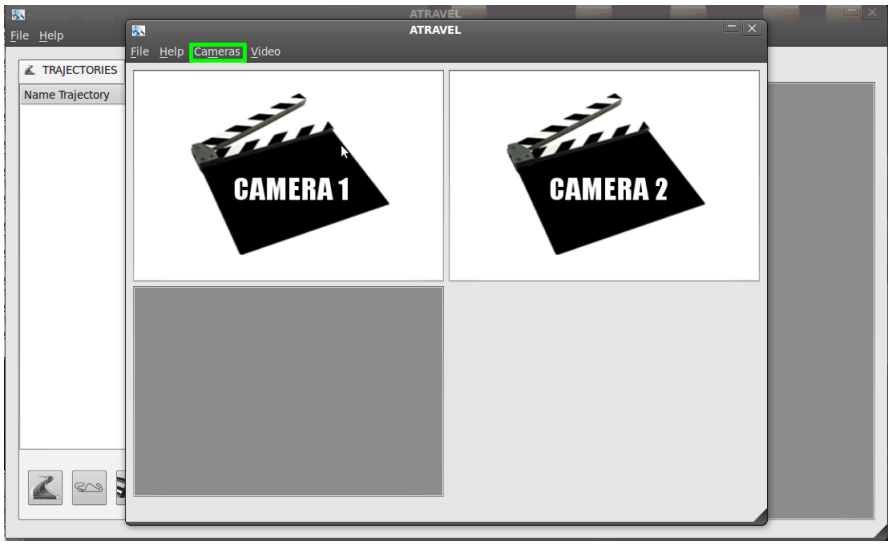

FIGURA A.2: Selección del número de cámaras.

Para la visualización de una trayectoria realizada por un patrón es necesario seleccionar el patrón requerido, y después clicar en el primer botón que aparece más a la izquierda (ver imagen A.10). Además tras clicar en este botón aparecerá en el componente de visualización la representación de la trayectoria del patrón (ésta aparece en línea continua) .

Para la visualización del recorrido realizado por un patrón es necesario seleccionar el patrón elegido y posteriormente clicar en el segundo botón, empezando por la izquierda (ver imagen A.11). Tras clicar en este botón aparecerá en el componente de visualización la representación del recorrido del patrón (en líneas discontinuas).

En esta pestaña también se permite la representación de la trayecotoria de todos los patrones existentes en la base de conocimiento. Para poder visualizarlos, el usuario debe realizar click en el tercer botón comenzando por la izquierda (ver imagen A.12).

Del mismo modo, el usuario puede visualizar todas los recorridos de todos los patrones existentes en la base de conocimiento, clicando en el cuarto botón comenzando por la izquierda (ver imagen A.13).

La aplicación también permite eliminar del componente de visualización aquellas trayectorias representadas anteriormente. Para ello el usuario debe seleccionar el patrón y clicar en el quinto botón comenzando por la izquierda (ver imagen A.14).

Del mismo modo, puede eliminar el recorrido de un patrón anteriormente representado en el componente de visualización, para ello el usuario debe eleccionar un patrón y clicar en el sexto botón (ver figura A.15).

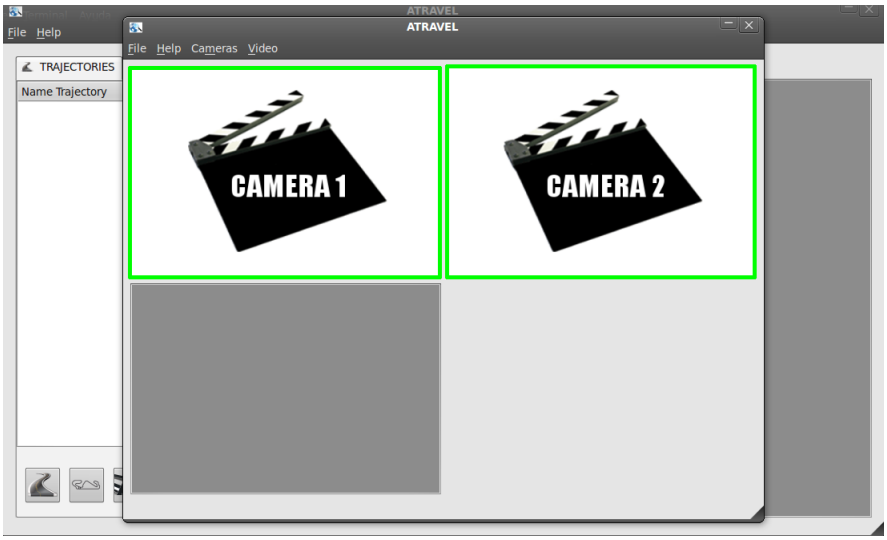

FIGURA A.3: Componentes de flujo de vídeo.

Por último, se ha facilitado al usuario la limpieza total del componente de visualización, para aquellas situaciones en las que exista demasiados patrones representados. Para eliminar todo lo que contiene el componente de visualizacion el usuario debe clicar en el séptimo botón comenzando desde la izquierda (ver figura A.16).

Además en la pestaña de patrones el usuario puede visualiza la información referente al tipo de patrón reconocido (vehículo o peatón), la categoría (comportamiento normal, anormal o no catalogado) y el color con el que es representado (rojo, verde o ámbar) (véase figura A.17).

En la pestaña de objetos detectados se han utilizado las mismas funcionalidades que en la pestaña de patrones y aparecen representadas con el mismo símbolo en cada uno de los botones. Sin embargo, se han añadido dos nuevas funcionalidades con respecto a la pestaña de patrones, ambas funcionalidades corresponden con los dos primeros botones representados.

El primer botón de esta pestaña permite representar en el componente de visualización la trayectoria temporal de un objeto previamente seleccionado (ver figura A.18).

El segundo botón de esta pestaña permite representar en el componente de visualización recorrido temporal de un objeto previamente seleccionado (ver figura A.19).

Además en esta pestaña permite visualizar al usuario características como el porcentaje de similitud con respecto a las trayectorias y recorridos de los patrones de la base de conocimiento (ver figura A.20.

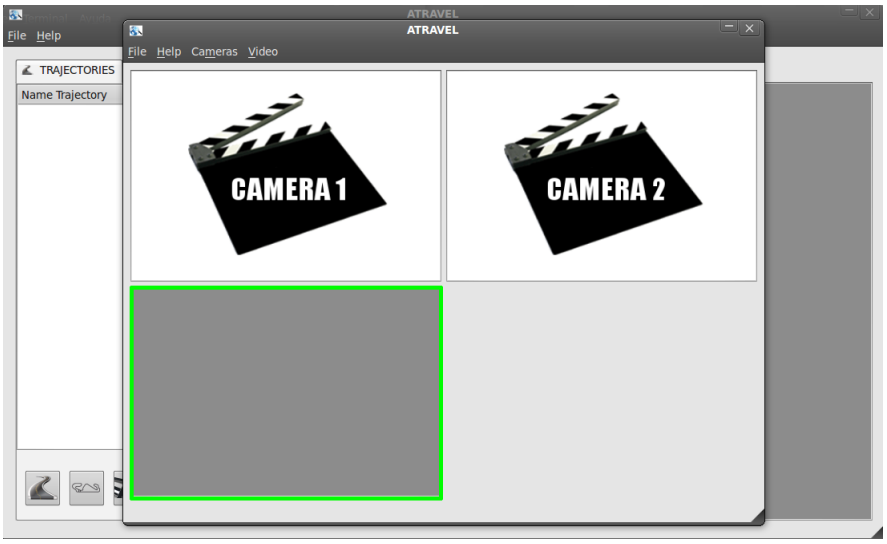

FIGURA A.4: Componentes representación 3D.

En la pestaña de clasificación, se pueden observar los distintos patrones existentes en la base de conocimiento que no han sido clasificados como comportamiento normal o anormal. En esta pestaña se permite catalogar manualmente su comportamiento (ver figura **??**).

En la pestaña de log, el usuario puede obtener información que ocurre tras la monitorización de la escena (ver figura A.21).

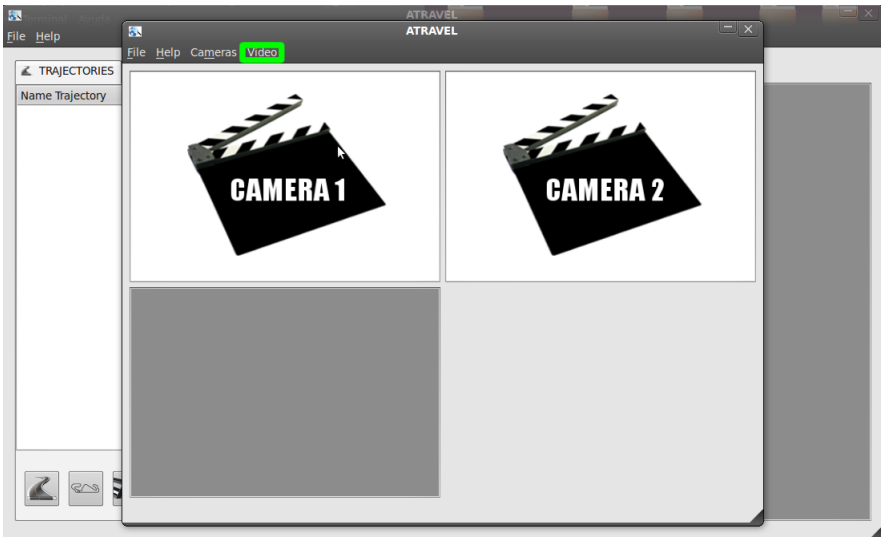

FIGURA A.5: Inicio de la monitorización.

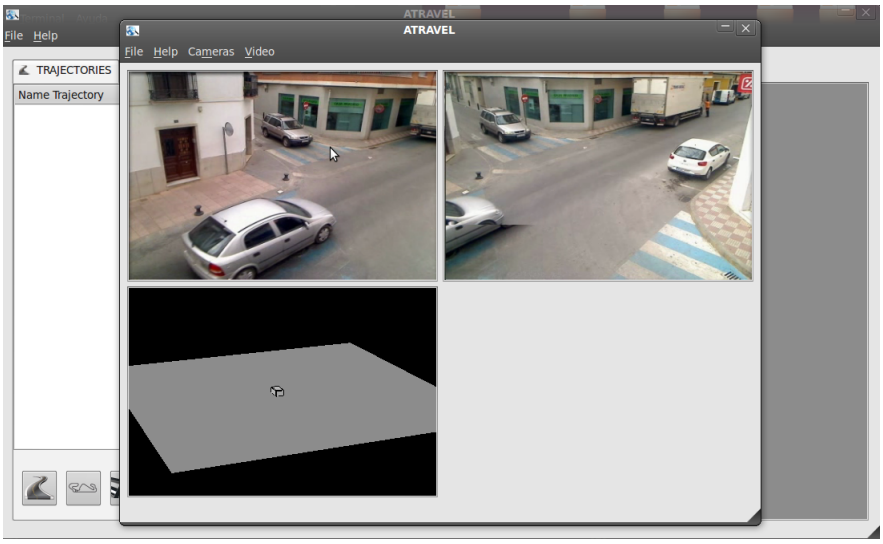

FIGURA A.6: Visualización de monitorización.

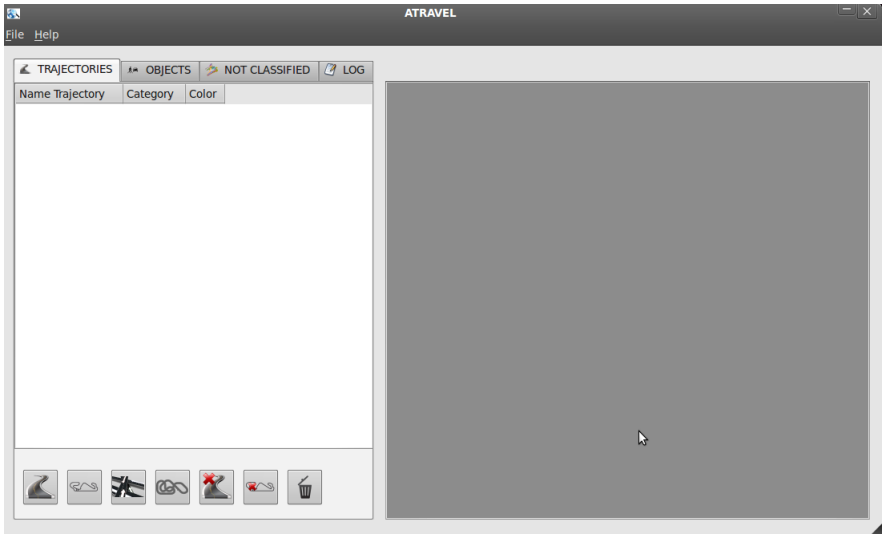

FIGURA A.7: Aspecto inicial de la interfaz principal.

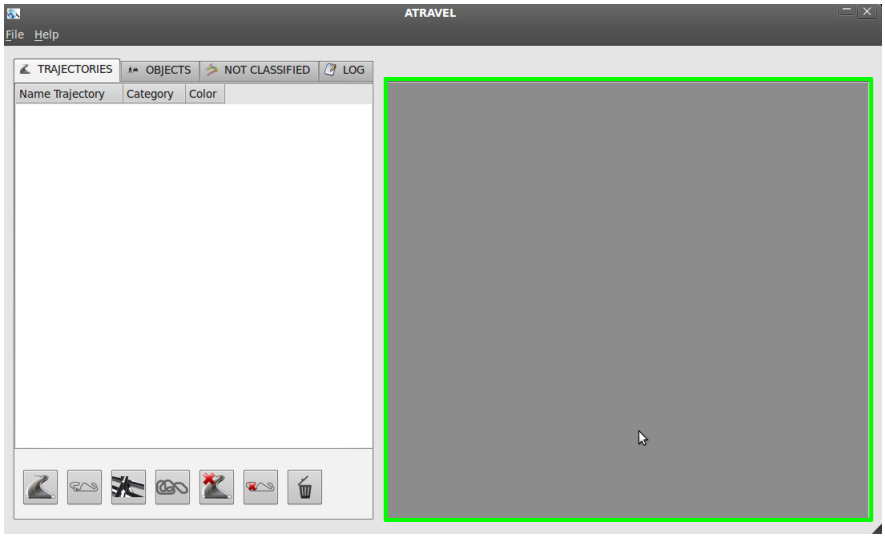

FIGURA A.8: Componente de visualización 3D de la interfaz principal.

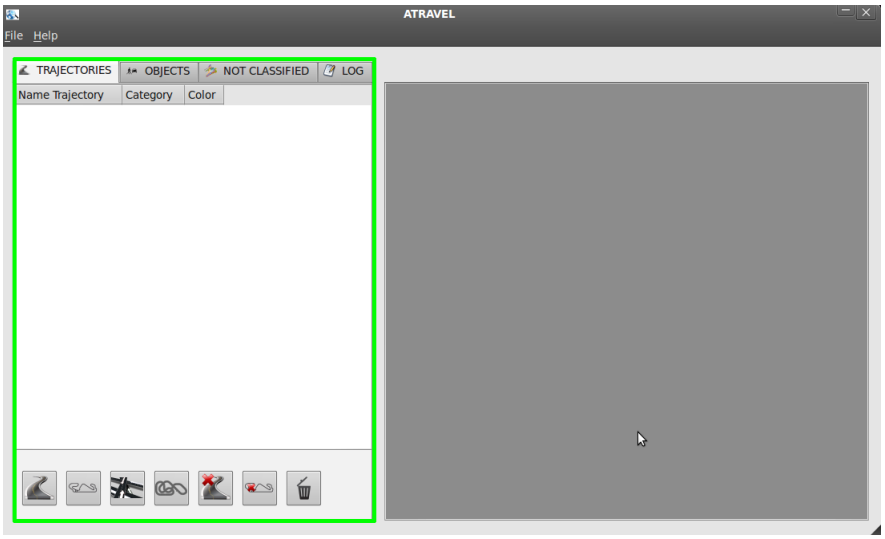

FIGURA A.9: Componente que contiene propiedades de los objetos reconocidos.

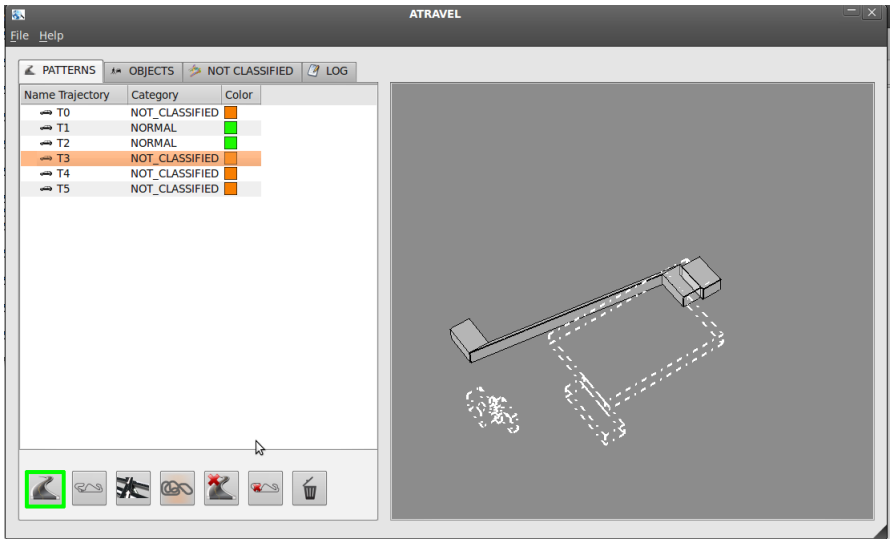

FIGURA A.10: Visualización de una trayectoria del patrón.

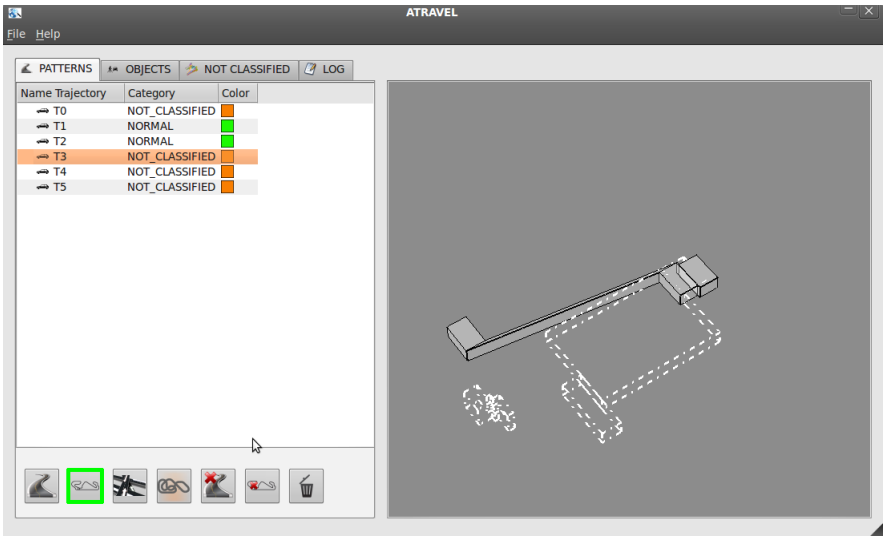

FIGURA A.11: Visualización de recorrido de un patrón.

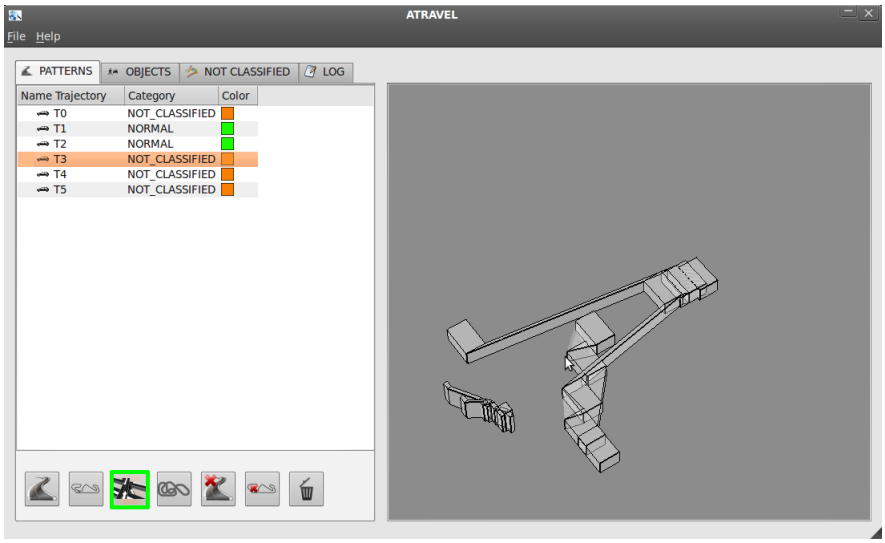

FIGURA A.12: Visualización de la trayectoria de todos los patrones.

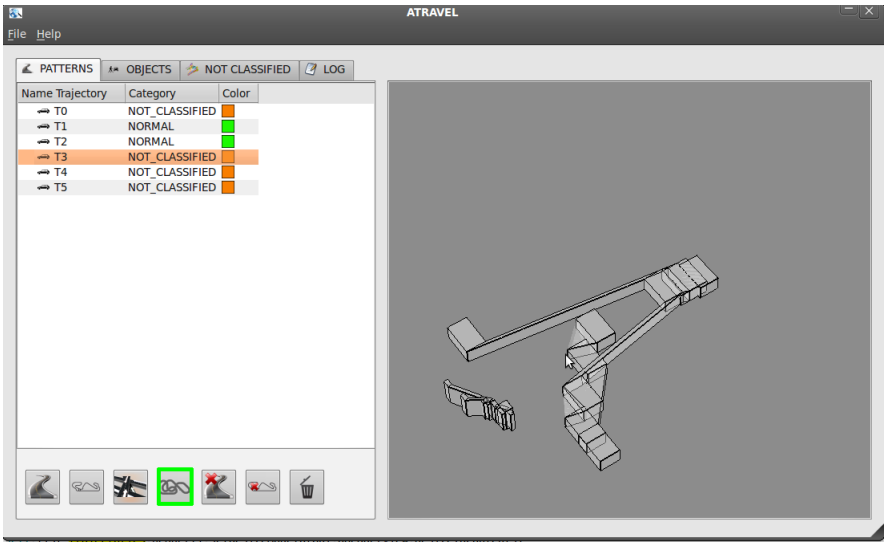

FIGURA A.13: Visualización del recorrido de todos los patrones.

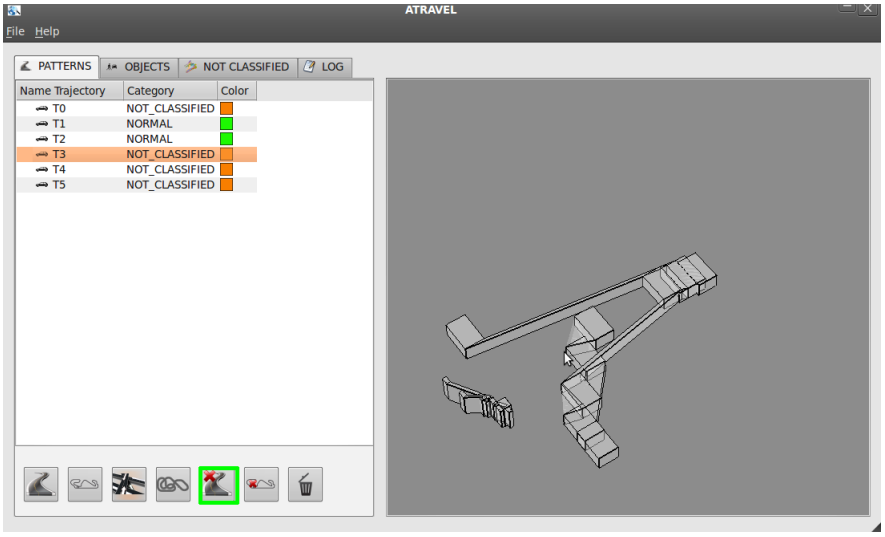

FIGURA A.14: Eliminación trayectoria de un patrón.

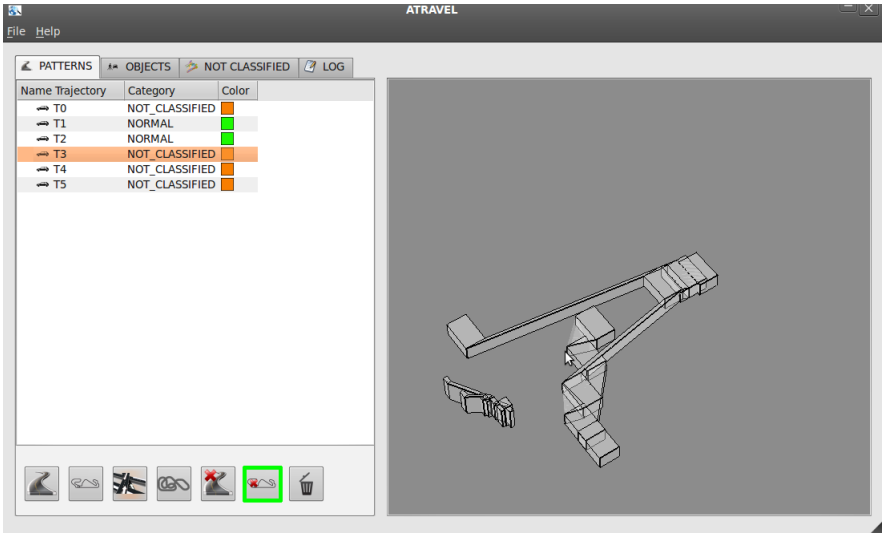

FIGURA A.15: Eliminación recorrido de un patrón.

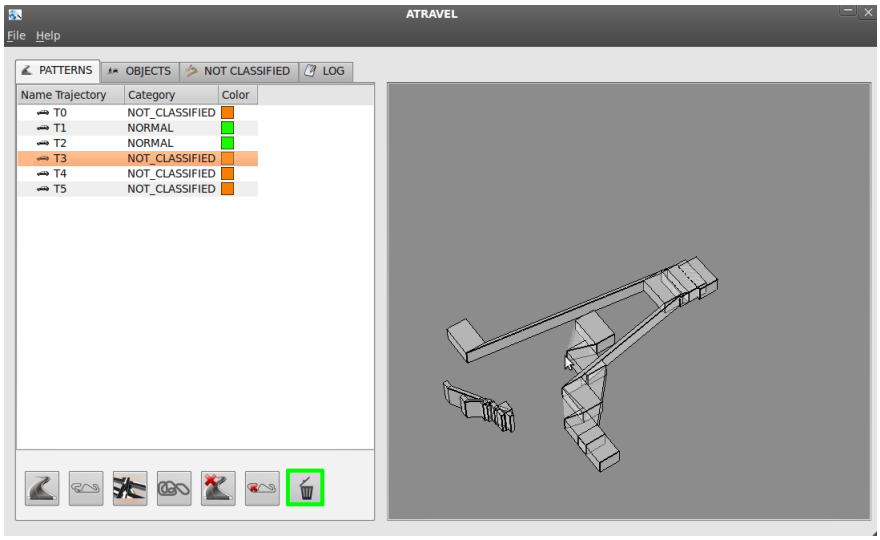

FIGURA A.16: Eliminación de todo lo representado en el componente de visualización.

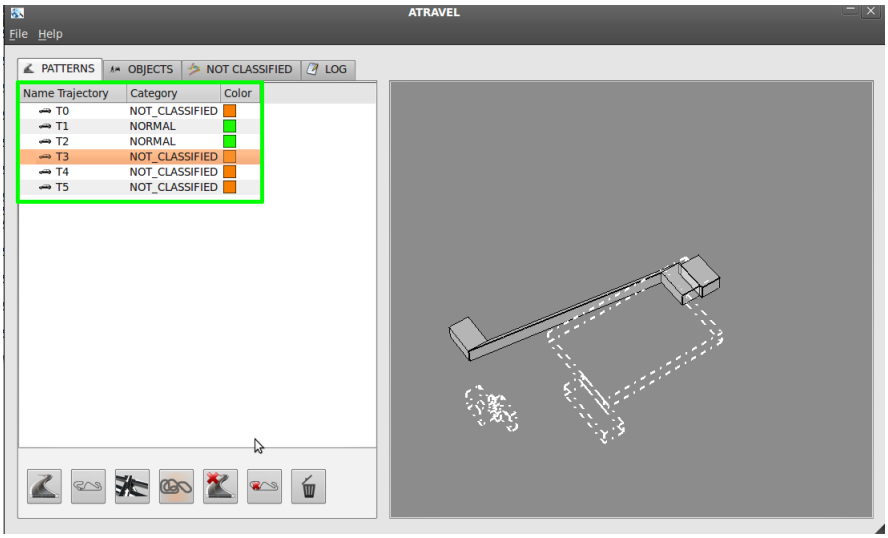

FIGURA A.17: Representación de característica de un patrón.

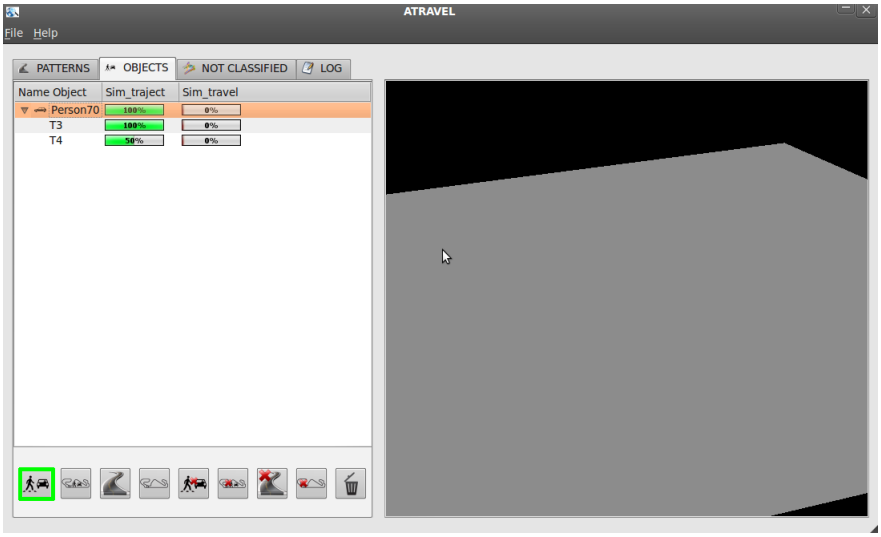

FIGURA A.18: Representación trayectoria de un objeto.

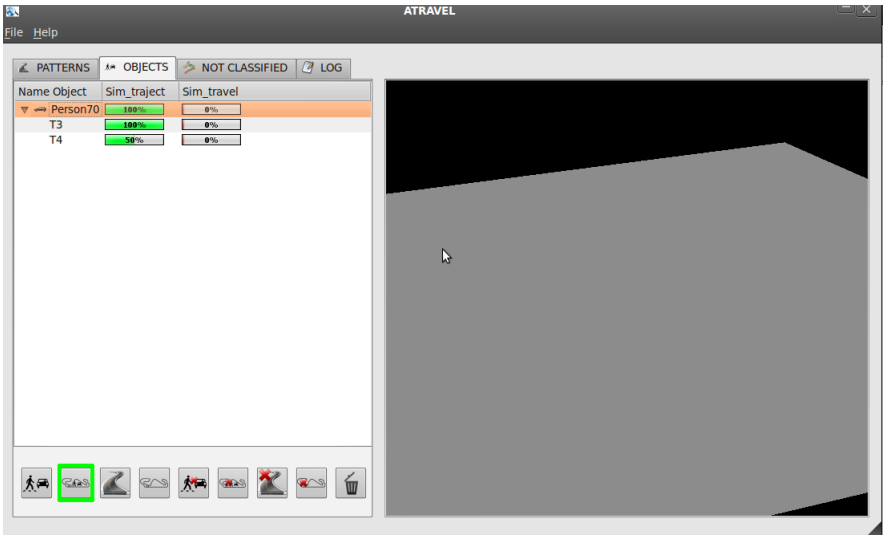

FIGURA A.19: Representación recorrido de un objeto.

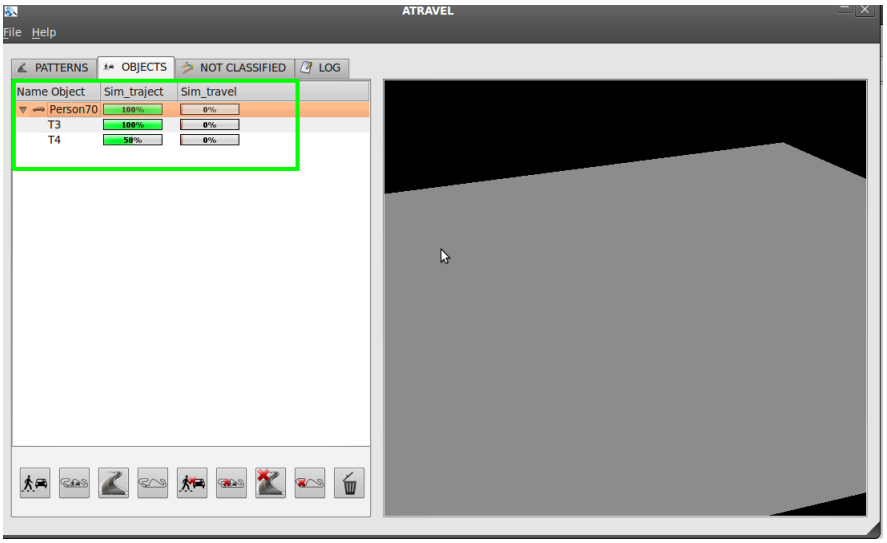

FIGURA A.20: Representación similitud de un objeto.

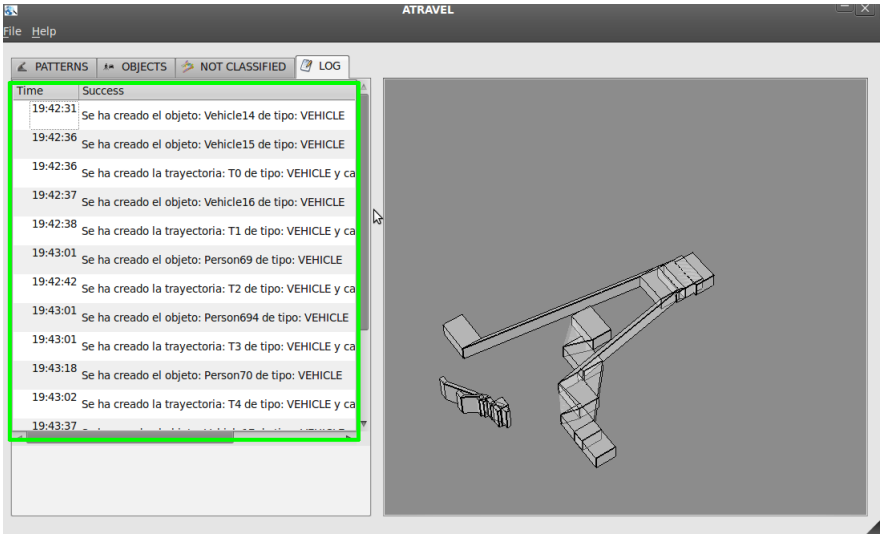

FIGURA A.21: Representación del log.

B

Imágenes a color

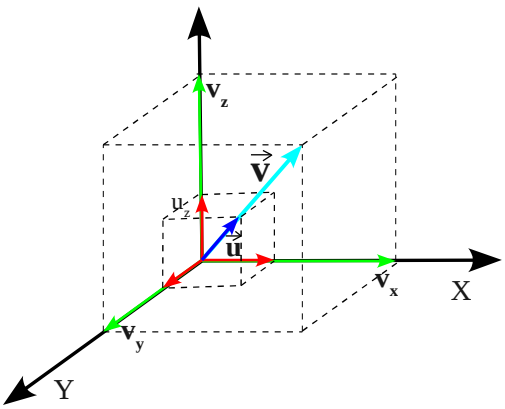

FIGURA B.1: Vector unitario.

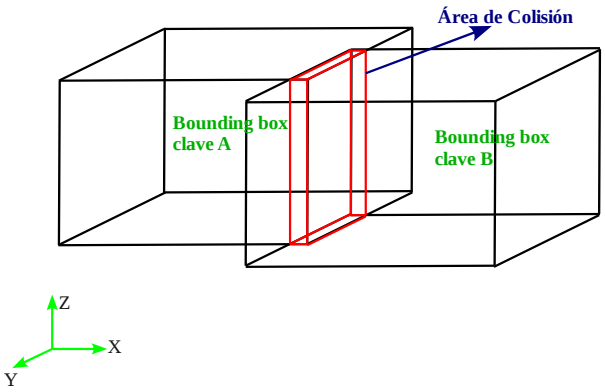

FIGURA B.2: Representación de colisión de dos bounding box.

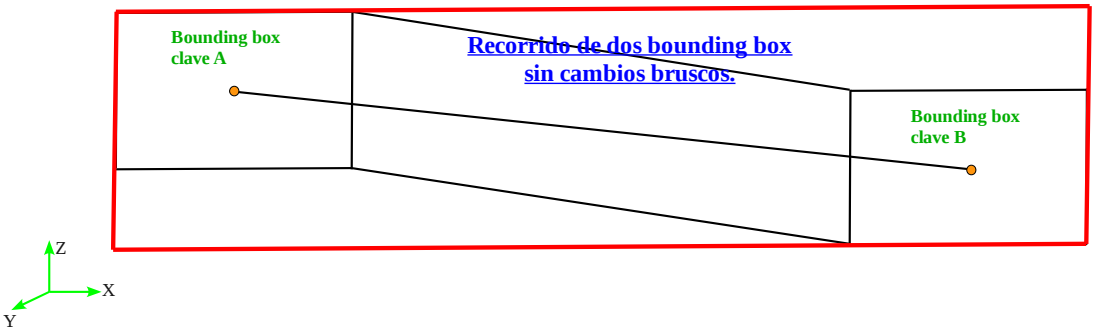

FIGURA B.3: Representación del recorrido de un objeto.

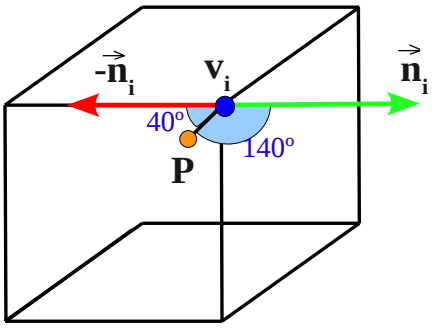

FIGURA B.4: Representación del vector normal de una cara de un BB.

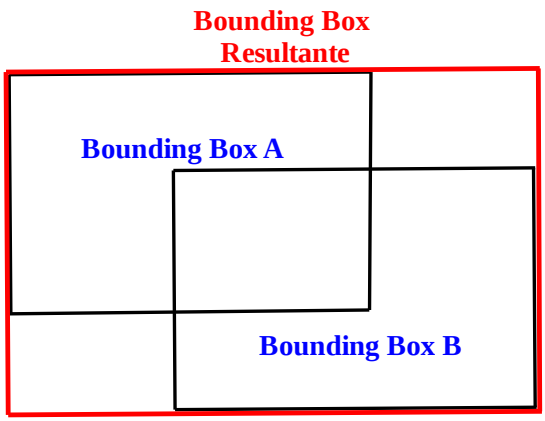

FIGURA B.5: Representación del BB resultante de la fusión de dos BB.

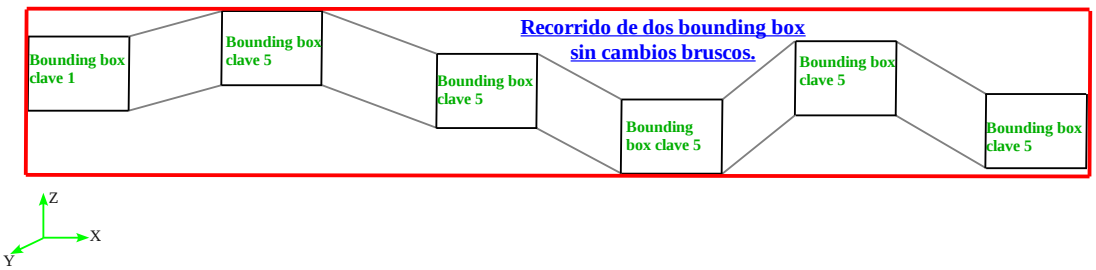

FIGURA B.6: Representación del recorrido completo de un objeto sin realizar cambios importantes en su desplazamiento.

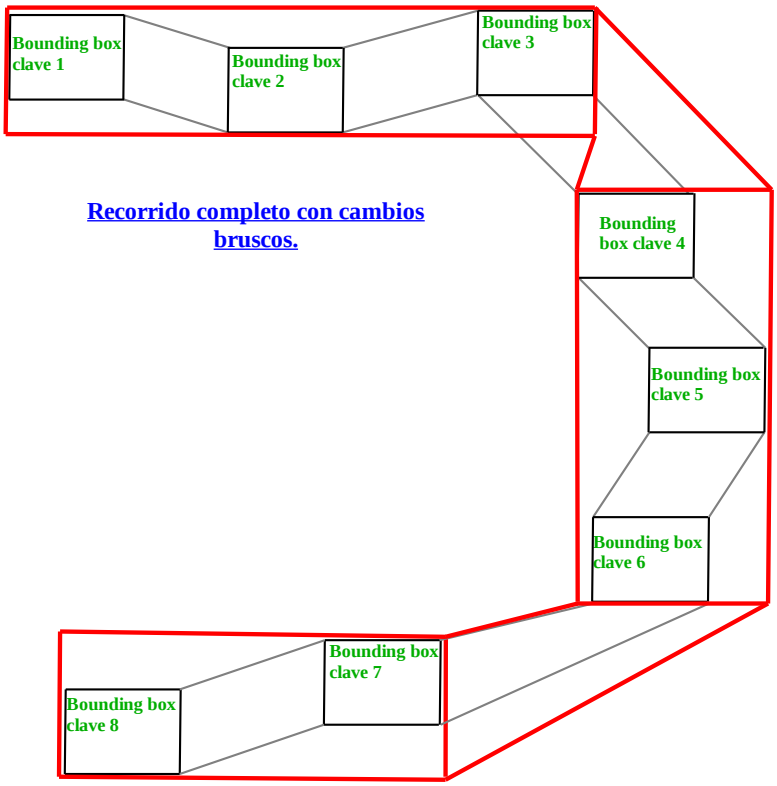

FIGURA B.7: Representación del recorrido completo de un objeto realizando cambios importantes en su desplazamiento.

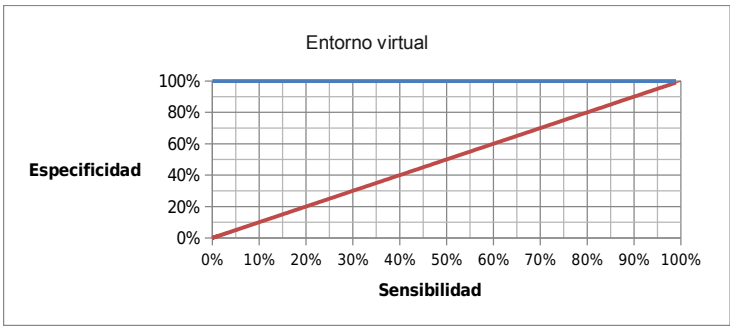

FIGURA B.8: Gráfico de curva ROC en el entorno virtual.

FIGURA B.9: Entorno real monitorizado.

FIGURA B.10: Comportamiento de vehículos en la escena monitorizada.

FIGURA B.11: Comportamiento de vehículos en la escena monitorizada (ver imagen a color en el Anexo B).

FIGURA B.12: Comportamiento de peatones en la escena monitorizada.

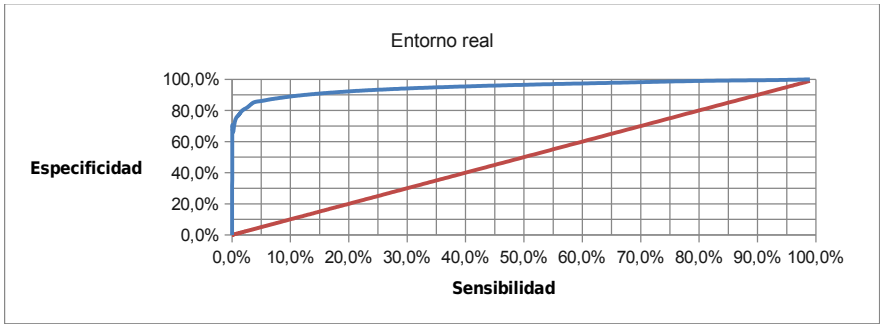

FIGURA B.13: Curva ROC resultante utilizando entorno real.

Bibliografía

[ABS] Alexei Gritai Arslan Basharat and Mubarak Shah. Learning Object Motion Patterns for Anomaly Detection and Improved Object Detection. *Computer Vision Lab, School of Electrical Engineering and Computer Science,University of Central Florida, Orlando, FL, USA*.

[AMV09] Isabel Gracia Luengo Andrés Marzal Varó. *Introducción a la programación con Python.* Publicacions de la Universitat Jaume I. Servei de Comunicació i Publicacions, 2009.

[And89] T. Rist. André, G. Herzog. Natural Language Access to Visual Data: Dealing with Space and Movement; 1st Workshop on Logical Semantics of Time, Space and Movement in Natural Language, 1989.

[BK08] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library.* O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., 2008.

[BV99] Meloche J Boyd, J E and Y Vardi. Statistical tracking in video traffic surveillance. 1999.

[CL99] G. Biswas. C. Li. Clustering Sequence Data using Hidden Markov Model Representation. 1999.

[CX09] Chen Shu-Ching y S. Rubin Chen Xin, Zhang Cheng-cui. Discovering Moving Clusters in Spatio-temporal Data. 2009.

[DM02] T.J. Ellis. D. Makris. Path Detection in Video Surveillance. *Image and Vision Computing Journal.*, 2002.

[DM04] J. Black. D. Makris, T. Ellis. Bridging the Gaps between Cameras; IEEE Conference on Computer Vision and Pattern Recognition. 2004.

[DMB08] Tim Ellis Dimitrios Makris and James Black. Intelligent Visual Surveillance: Towards Cognitive Vision Systems. *The Open Cybernetics and Systemics Journal.*, 2008.

[ES08] A.G.A. Perera. E. Swears, A. Hoogs. Learning Motion Patterns in Surveillance Video using HMM Clustering. 2008.

[ESP] Anthony Hoogs Eran Swears and A.G. Amitha Perera. HMM-Based Clustering for Learning Motion Patterns in Surveillance Video.

[Fer96] Cohn A G Hogg-D C. Fernyhough, J H. Generation of semantic regions from image sequences. 1996.

[FJ02] M. Figueiredo and A. Jain. Unsupervised learning of finite mixture models. *PAMI, IEEE Transactions*, 2002.

[FM98] R. Fraile and S. J. Maybank. ehicle Trajectory Approximation and Classification. 1998.

[HB92] R J Howarth and H Buxton. Analogical representation of spatial events for understanding traffic behaviour. 1992.

[Hea08] Venetianer P. L. Lipton A. Hearing, N. The evolution of video surveillance: anoverview». Machine Vision and Applications. 2008.

[JA11] J.J. Castro-Schez-L. Jimenez-Linares. J. Albusac, D. Vallejo. OCULUS surveillance system: Fuzzy on-line speed analysis from 2D images. 2011.

[JEM04] Anthony J. Tromba Jerrold E. Marsden. *Cálculo Vectorial; Quinta Edición*. Pearson Education S.A., 2004.

[J.F00] D.C.Hogg. J.Fernyhough, A.G. Cohn. Constructing Qualitative Event Model Automatically from Video Input; Image and Vision Computing. 2000.

[JH95] N Johnson and D C Hogg. Learning the Distribution of Object Trajectories for Event Recognition. 1995.

[JH96] N. Johnson and D. C. Hogg. Learning the Distribution of Object Trajectories for Event Recognition. Image and Vision Computing. 1996.

[JLH02] Tieniu Tan Jiangung Lou, Qifeng Liu and Weiming Hu. Semantic Interpretation of Object Activities in a Surveillance System. 2002.

[JMSV04] Alexandre Bernardino José Melo, Andrew Naftel and José Santos-Victor. RETRIEVAL OF VEHICLE TRAJECTORIES AND ESTIMATION OF LANE GEOMETRY USING NON-STATIONARY TRAFFIC SURVEILLANCE CAMERAS. *University of Manchester Institute of Science & Technology, Manchester M60 1QD, UK and Instituto Superior Técnico, Av. Rovisco Pais 1049 - 001 Lisboa, Portugal*, 2004.

[Joh98] N. Johnson. Learning Object Behaviour Models; PhD thesis, School of Computer Studies. 1998.

[JP04] Jian Pei-M. Behzad. J. Pei, J. Han. Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. 2004.

[Kal05] y S. Bakiras. Kalnis, N. Mamoulis. On Discovering Moving Clusters in Spatio-temporal Data. 2005.

[KB04] D.R. Anderson. K.P. Burnham. Multimodel inference: Understanding AIC and BIC in model selection. 2004.

[Lut09] Mark Lutz. *Learning Python, Third Edition.* O'Reilly., 2009.

[MAF81] R. C. Bolles. M. A. Fischler. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Comm. of the ACM.*, 1981.

[ME] Dimitrios Makris and Tim Ellis. Path Detection in Video Surveillance.

[N.98] Johnson N. Learning Object Behaviour Models. 1998.

[NC08] Fredrik Nilsson and Axis Communications. *Intelligent Network Video: Understanding Modern Video Surveillance Systems.* CRC Press - 416 Pages, 2008.

[NS00] A. Bulpitt. N. Sumpter. Learning Spato-temporal Patterns for Predicting Object Behaviour, Image and Vision Computing. 2000.

[PJS] David H. Eberly Philip J. Schneider. *Geometric Tools for Computer Graphics*. Morgan Kaufmann Publishers.

[SB98] N Sumpter and J Bulpitt. Learning spatio-temporal patterns for predicting object behaviour. 1998.

[Shr09] Dave. Shreiner. *OpenGL programming Guide:the official guide to learning OpenGL, versions 3.0 and 3.1, Seventh edition*. Pearson Education, Inc., 2009.

[Smi04] G. J. D. Smith. Behind the screens: Examining constructions of deviance and informal practices among cctv control room operators in the uk. Surveillance and Society. 2004.

[Smy97] P. Smyth. Clustering sequences with Hidden Markov Models. 1997.

[SOM05] IAN SOMMERVILLE. *INGENIERIA DEL SOFTWARE*. ADDISON-WESLEY, 2005.

[VG08] CRIM Valérie Gouaillier. Intelligent Video Surveillance: Promises and Challenges. 2008.

[WIL03] ANTHONY J. WILSON, JERRY D. y BUFFA. *Física, 5ª Edición*. PEARSON EDUCATION, 2003.

[YLdM05] F. Luisier J. Jacot. Y. L. de Meneses, P. Roduit. Trajectory Analysis for Sport and Video Surveillance. *Laboratoire de Production Microtechnique (LPM-IPR), Ecole Polytechnique Fédérale de Lausanne.*, 2005.

[YY12] Jian WU Guangming ZHANG Xuefeng XIAN Yuanfeng YANG, Zhiming CUI. Trajectory Analysis Using Spectral Clustering and Sequene Pattern Mining. 2012.

[Zha08] Monekosso D. N. Remagnino P. Velastin S. A. Xu L.-Q Zhan, B. Crowdanalysis: a survey", Machine Vision and Applications. 2008.

[ZZT06] K. Huang Z. Zhang and T. Tan. Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes. 2006.