UNIVERSIDAD DE LA REPÚBLICA || URUGUAY

FACULTAD DE INGENIERÍA


# TRACKING APLICADO AL FÚTBOL

TESIS PRESENTADA POR
PABLO CHIESA - ERNESTO FERREIRA - NICOLÁS SAMPIETRO
PARA OBTENER EL TÍTULO DE INGENIERO DE LA UNIVERSIDAD DE LA REPÚBLICA

Montevideo, 2011

TUTORES: PABLO CANCELA, ÁLVARO GÓMEZ, ÁLVARO MARTÍN

Instituto de Ingeniería Eléctrica – Instituto de Computación

# Agradecimientos

Esta tesis de grado no hubiera podido realizarse sin el apoyo de algunas personas y organizaciones. Queremos agradecer a los profesores Alvaro Gómez y Pablo Cancela por los valiosos aportes, guía y seguimiento brindado. A la fundación Julio Ricaldoni por confiar en este grupo y por el apoyo económico que hizo posible realizar las pruebas de campo.

También queremos agradecer a Ignacio Oliveri, Gonzalo Arreche y Diego Ferreira quienes nos facilitaron el acceso a los estadios del Club Nacional de Football, del Club Sportivo Miramar Misiones y del Defensor Sporting Club respectivamente. También a las instituciones mencionadas por permitirnos trabajar libremente dentro de sus instalaciones.

No podemos terminar sin agradecer a nuestros familiares, por estar siempre presentes incondicionalmente.

# Índice general

# Índice de figuras

# Índice de cuadros

# Capítulo 1

# Descripción del problema

Los hinchas salen del estadio de fútbol discutiendo cuál fue el mejor jugador del encuentro. La discusión puede durar días, los especialistas deportivos siempre tienen claro cuáles fueron las causas de la derrota. Los técnicos cuentan con la experiencia, conocen a los jugadores, saben quién "anduvo bien" y quién no.

Más allá de cualquier argumento, lo cierto es que todas estas conclusiones son hijas de la intuición, experiencia y percepción. La alta competencia y el profesionalismo del fútbol moderno necesitan de otras herramientas que ayuden a la toma de decisiones y a la evaluación de los jugadores. Estas herramientas son especialmente útiles para los involucrados en el rendimiento del equipo (cuerpo técnico, club, representantes deportivos). En el medio local, surgió recientemente una empresa llamada Kizanaro [12] que brinda este tipo de servicio.

Kizanaro brinda a sus clientes un sistema en el cual es posible repasar y visualizar un gran conjunto de "eventos" que suceden durante un partido de fútbol, como ser: ataques, córners, fueras de juegos, goles, entre otras jugadas. También es posible obtener datos como: numero de pases acertados o errados, porcentaje de dominio de balón por cada cuadro. Estos eventos son indexados en un catálogo y pueden ser accedidos por medio de una interfaz de usuario. Con esta información el técnico de fútbol puede analizar los partidos, su propio equipo y el rival.

En caso de la empresa mencionada, la información es obtenida de forma "manual" por un conjunto de operadores que trabajan sobre las imágenes del partido (es decir, la televisación). Este mecanismo presenta algunas desventajas importantes, entre

las que se pueden destacar: tiempo de procesamiento, costo en recursos humanos , exactitud y completitud de los datos.

Con respecto a la completitud, existe otro valioso conjunto de datos que no puede ser obtenido con la técnica manual. Por ejemplo, la **distancia recorrida en km** es un valor trascendente para los preparadores físicos, les indica el desgaste que tuvo el deportista durante el partido y en consecuencia planear una recuperación acorde. El **área o zona de desplazamiento de un jugador** puede indicarle al técnico si un jugador se movió por la zona donde se suponía debía moverse. El **modelo 2D del campo de juego.** La televisación no cubre todo el terreno de juego, por lo que, cuando el técnico revisa las imágenes de un encuentro, pierde información importante sobre lo que está sucediendo en el resto del cancha. El modelo 2D del campo de juego, le muestra al técnico lo que sucedió en todo el terreno de juego los 90 minutos de partido.

Para obtener este último conjunto de datos mencionado, tendría que implementarse un nuevo mecanismo de obtención de datos. Este mecanismo tendría que calcular la posición (en coordenadas del mundo real 3D) instante tras instante de cada jugador. Además sería necesario calcular la velocidad. Pensar en un procesamiento manual en esta caso, involucraría una o varias personas "etiquetando" sobre imágenes de video, cada uno de los jugadores cuadro a cuadro. Una cámara promedio toma 25 cuadros por segundo, eso es 810.000 cuadros en 90 minutos de juego, multiplicado por 24 que es el numero de jugadores promedio, se tendrían que colocar manualmente 194.400.000 etiquetas para lograr procesar un partido. Por supuesto que esta es una tarea imposible de abordar sin la ayuda de alguna técnica que facilite el procesamiento.

Este es el problema que da nacimiento al proyecto de grado titulado "Tracking aplicado al fútbol" que se desarrolló en la Facultad de Ingeniería de la Universidad de la República. Resolver el problema de obtener: posición 3D, velocidad de los jugadores y el balón en un partido de fútbol, durante los 90 minutos de partido utilizando técnicas de seguimiento e identificación sobre imágenes digitales.

Notar que contando con esta información (posición de jugadores y balón en cada instante) es posible derivar otros valores útiles para el análisis como ser: velocidad media, distancia y zona recorrida de la cancha, porcentaje de posesión del balón,

porcentaje de pases errados o acertados. Estos datos serán obtenidos para analizar ambos equipos en el campo de juego.

## 1.1. Estado del arte en soluciones tecnológicas aplicadas al deporte

Existe una gran cantidad de formas de abordar este problema. Nos centraremos en las soluciones que utilizan mecanismos no invasivos (como podría ser algún dispositivo de radio colocado en los zapatos del jugador) debido al requerimiento de obtener datos para los 22 jugadores, después de todo, no se puede contar con que el equipo rival esté dispuesto a ser analizado.

Obtuvimos variada y extensa documentación relacionada con seguimiento e identificación de múltiples objetos en movimiento y en particular trabajos sobre seguimiento de jugadores de fútbol utilizando varias cámaras fijas o en movimiento.

En el artículo Tracking Football Player Movement From a Single Moving Camera Using Particle Filters (Anthony Deardena, Yiannis Demirisa and Oliver Graub) [5], se utiliza una sola cámara con movimiento y un filtro de partículas, a diferencia de múltiples cámaras fijas que veremos en otras soluciones. La desventaja de esta solución es no poder obtener una representación de toda la cancha. Esto no permite realizar análisis estadísticos que se basen en los 22 jugadores y en los 90 minutos del partido. Sin embargo una cámara en movimiento, tiene como principal ventaja la posibilidad de utilizar la televisación de los encuentros, ahorrando el costo de instalación de cámaras propias en el estadio.

Es necesario diferenciar las soluciones que contemplan el balón de las que no. A priori se podría pensar que el problema del seguimiento del balón es de, por lo menos, la misma complejidad que los demás problemas a solucionar. Sin embargo este punto debe ser resuelto con la ayuda de múltiples cámaras desde distintos ángulos para una reconstrucción 3D del escenario.

En el artículo Tracking and Labelling of Interacting Multiple Targets (Josephine Sullivan and Stefan Carlsson) [22] los autores se ocupan del etiquetado de múltiples objetos en movimiento, para luego realizar un estudio aplicado al fútbol. El problema del etiquetado, se pude dividir en dos etapas: primero localizar los objetos en la

escena y luego etiquetar su identidad (cuadro 1, cuadro 2, juez, balón). En dicho artículo los autores proponen una solución innovadora modelando las trayectorias de los jugadores con una estructura de grafo. Luego el algoritmo permite identificar trayectorias relacionadas que están separadas temporalmente. Esta separación ocurre con las oclusiones. Es necesario obtener una representación estática de todo el campo de juego, para ello se colocan 4 cámaras que logran cubrir todo el campo de juego. Ver figura 1.1

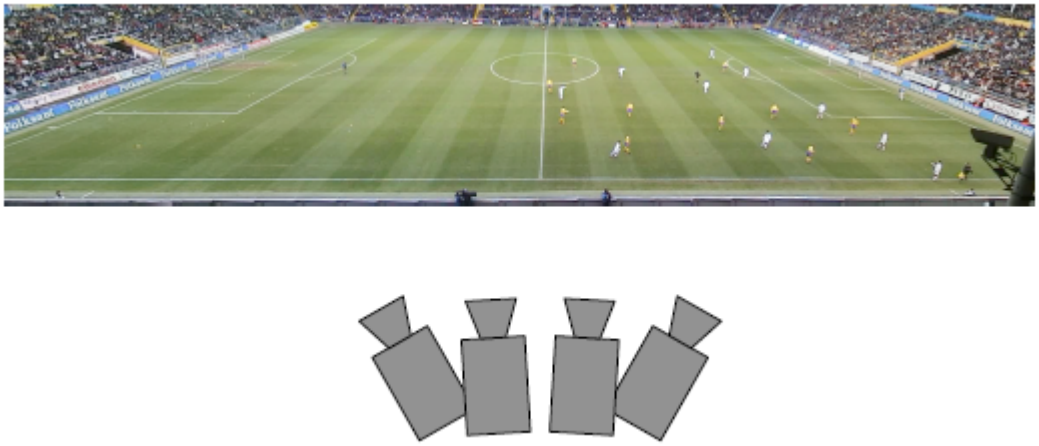

Figura 1.1: Gran angular con 4 cámaras.

El capítulo 8 del libro: IET Professional Applications of Computing Series 5, Intelligent Distributed Video Survelliance Systems (D. Thirde, M. Xu and J. Orwell. Tracking football players with multiple cameras) [8], es una descripción detallada de un sistema y una serie de técnicas de seguimiento y etiquetado aplicado a jugadores de fútbol. Se utilizan seis u ocho cámaras fijas y es capaz de ser ejecutado en tiempo real. La entrada proviene de las cámaras que cubren todo el terreno de juego. La salida son las coordenadas en el mundo real y una estimación estadística de la categoría de los jugadores, balón y jueces.

## 1.2. Soluciones comerciales existentes

Existe en el mercado una gran cantidad de sistemas de análisis de partidos de fútbol. En esta sección se verán los más relevantes a nuestro criterio.

### 1.2.1. ProZone

ProZone [20] trabaja con los principales clubes del fútbol mundial. ProZone ha desarrollado una completa gama de herramientas de análisis que ayudan a los entrenadores a obtener datos sobre el rendimiento y estado físico de sus jugadores. Esto representa una ventaja competitiva sobre sus rivales.

#### Análisis completo del partido

El sistema ProZone analiza el rendimiento del equipo y su nivel físico mediante el trazado de las acciones y movimientos de cada jugador durante todo el partido. Se captan imágenes de video a una frecuencia de 10 cuadros por segundo y el software de ProZone las procesa proporcionando información detallada del partido.

Esta información viene dada en una sencilla interfaz 2D en la que los entrenadores pueden consultar numerosos aspectos del juego. Por ejemplo, puede saber cuál es la distancia que ha corrido un jugador en los 90 minutos.

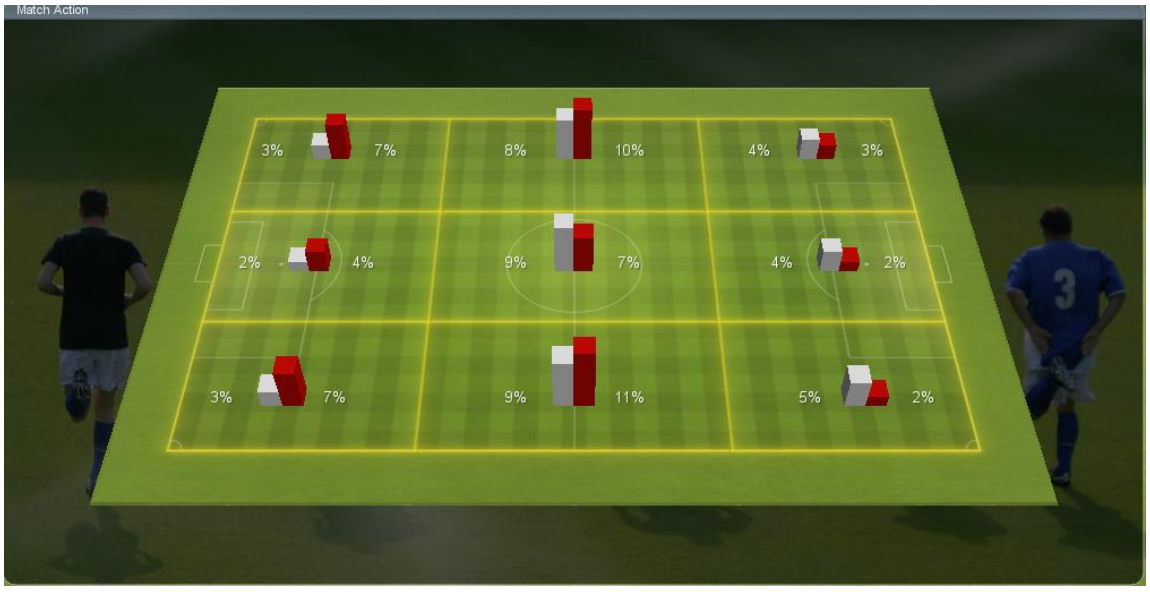


Figura 1.2: Zonas de acción del partido, ProZone Software.


Figura 1.3: Análisis de situaciones del partido, ProZone Software.

### 1.2.2. Amisco

Amisco [3] es el software de la empresa francesa SUP (Sport Universal Process), que se afincó en San Sebastián (porque su primer cliente fue la Real Sociedad) y que tiene entre sus productos AmiscoPro, un programa que descifra un partido de fútbol con todas sus variables estadísticas.

#### ¿Cómo funciona?

Ocho cámaras situadas alrededor del campo graban todo lo que sucede en el césped, estas imágenes se envían a la central, donde se indexan para crear una recreación virtual del terreno de juego durante los 90 minutos. Luego se realiza el seguimiento de cada jugador, los tres árbitros y el balón. De esa manera se pueden contabilizar con exactitud todas las acciones técnicas realizadas por cada jugador (pase, disparo, cabezazo, etc.). Toda esta información es enviada después al club en formato DVD y el cuerpo técnico puede trabajar con ella tantas veces como sea necesario.

### 1.2.3. IMPIRE AG

IMPIRE AG [2] ha operado la mayor base de datos de la Bundesliga (liga de fútbol alemana) durante más de 15 años. No solo registra datos para la Bundesliga sino que también registra información de análisis de partidos para la Champions League, UEFA Cup, DFB Cup, campeonatos mundiales y europeos, la selección de Alemania y la liga de Austria.

El software desarrollado por IMPIRE AG para el seguimiento de jugadores se llama VIS.TRACK [25]. El sistema VIS.TRACK permite analizar y archivar datos de rendimiento de los jugadores en tiempo real. Por tanto, el sistema utiliza los algoritmos informáticos más recientes y la tecnología de cámaras de última generación. Con sólo dos cámaras es posible analizar el partido completo, todos los datos de los jugadores y el balón son capturados. El sistema VIS.TRACK no requiere una instalación permanente se tarda tan solo 30 minutos en instalar el sistema y por lo tanto puede ser utilizado en el estadio y también en el campo de entrenamiento, así como durante la pretemporada.

### 1.2.4. Kizanaro

Kizanaro [12]. Empresa Uruguaya, que ofrece productos relacionados con tecnología aplicada al deporte. Kizanaro ofrece a sus clientes análisis de partidos y jugadores en más de 150 categorías personalizadas. El sistema registra y almacena los eventos de un partido como pases, tiros al arco, etc. Esta información es presentada al usuario por medio de una interfaz que además muestra un resumen del video de cada uno de los eventos registrados. Como ya fue mencionado, la información es obtenida de forma manual, por un grupo de operadores que ingresan los datos al sistema revisando vídeos del encuentro.

## 1.3. Descripción y alcance del proyecto

El objetivo del proyecto es diseñar e implementar un prototipo de software que pruebe la viabilidad técnica de una solución al problema planteado, utilizando técnicas de seguimiento e identificación sobre imágenes digitales.


Figura 1.4: K-Studio, programa desarrollado por Kizanaro.

Dado el requerimiento de procesar los 22 jugadores en todo el terreno de juego, las técnicas que utilizan múltiples cámaras fijas son las que mejor aplican. La entrada del sistema serán los vídeos que provienen de las cámaras fijas y la salida, la posición y velocidad de cada uno de los 22 jugadores durante la secuencia a procesar. La salida también deberá incluir una estimación de la categoría (cuadro 1, cuadro 2, golero 1, golero 2, jueces ) de cada uno de los objetos detectados. Notar que una identificación dentro del equipo (jugadores del mismo equipo) no es posible con la resolución que manejan las cámaras, además no se encontró evidencia de que este problema haya sido resulto por alguna empresa o grupo de investigación.

Es de esperar que ocurran las llamadas oclusiones, esto es cuando dos o más jugadores se "solapan" desde el punto de vista de una cámara. El motivo de utilizar múltiples cámaras fijas es justamente poder resolver este problema. Aun así, es de esperar que existan situaciones en donde el sistema no pueda resolver correctamente la identidad de cada objeto especialmente al salir de una oclusión. En estos casos, es necesario la intervención de un operador que resuelva el problema. También será ne-

cesario, como será visto más adelante, proveer al sistema con datos como: cantidad de jugadores por cada equipo. Nuevamente aquí se espera la presencia de un operador humano. Esto lleva a utilizar el termino "semiautomático" para referirse al funcionamiento del sistema.

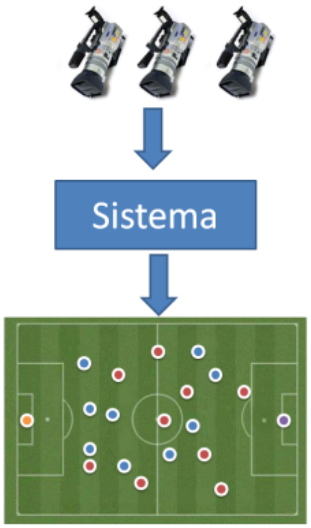


Figura 1.5: Descripción del sistema. La entrada es un conjunto de videos, y la salida es la posición y velocidad de cada uno de los jugadores y árbitros.

El seguimiento del balón, representa un problema aparte y no será abordado en este proyecto principalmente por cuestiones de plazos de tiempo. Notar que dada la observación del balón en una cámara, su posición real en coordenadas 3D es indeterminada (infinitas soluciones), podría estar en el aire o sobre el campo de juego. La técnica de múltiples cámaras fijas que vamos a utilizar, es ideal para resolver este problema, una descripción de cómo resolverlo se encuentra en "A general framework for 3D soccer ball estimation and tracking" [11]. Para los jugadores, sin perder generalidad, asumimos que siempre se encuentran sobre el terreno de juego (z=0).

El proyecto involucra además pruebas de campo para la obtención de un conjunto de datos de entrenamiento propio. Con el apoyo económico de la Fundación Julio Ricaldoni [21], se rentará el equipo necesario.

### 1.3.1. Tiempo de desarrollo del proyecto

El relevamiento de los artículos técnicos y el estado del arte tomo aproximadamente los primeros dos meses de trabajo. Aquí se busco y reviso material bibliográfico así como también juegos de datos completos (videos de múltiples cámaras y datos de calibración). Con respecto al material bibliográfico, luego del relevamiento inicial, nos concertamos en trabajos recientes que utilizaran un conjunto de cámaras fijas por creer que esta es la mejor solución a nuestro problema.

El capitulo 8 del libro: IET Professional Applications of Computing Series 5, Intelligent Distributed Video Survelliance Systems (D. Thirde, M. Xu and J. Orwell. Tracking football players with multiple cameras) [8] brinda una solución ordenada y bien documentada. Además gracias al sitio web que mantienen los autores, contamos con el juego de datos que fue utilizado en el documento original. Tomamos entonces, este articulo como guía de trabajo. Se comenzó por el estudio y comprensión de las técnicas utilizadas. Para esta tarea fue necesario buscar material bibliográfico complementario que acompañara el estudio de cada tema. Además se implementaron pruebas de concepto utilizando el lenguaje MatLab. Estas pruebas de concepto permitieron llegar a comprender las técnicas descriptas y poder incluso probar algunas modificaciones que entendimos aplicaban mejor a nuestro problema. Este proceso junto con el de la adquisición de los datos propios, tomo aproximadamente seis meses de trabajo.

Luego, se comenzó con el diseño software del prototipo. Se definieron las clases, módulos, interfaces, interfaces de usuario. Esta etapa tomo aproximadamente dos meses de trabajo. En la siguiente etapa se implemento el prototipo, incluyendo clases, módulos, interfaces de usuario, archivos de configuración y aplicaciones auxiliares que serán descriptas más adelante en este documento. La etapa siguiente se dedicó a las pruebas unitarias del código y validación. Esto llevó a tareas de re-programación en algún caso. Estas dos etapas tomaron aproximadamente cinco meses de trabajo.

Finalmente se retomó la etapa de documentación que había sido comenzada en paralelo con la etapa de implementación. Esta última etapa tomo aproximadamente otros dos meses de trabajo para un total aproximado de diecisiete meses de proyecto.

# Capítulo 2

# Descripción de la solución propuesta

En este capítulo se brindará una descripción de alto nivel de la solución propuesta. Luego en los capítulos siguientes se describirá con más detalle los algoritmos y técnicas utilizadas. Se propone un procesamiento en dos etapas. La primera encargada del procesamiento por cámara. La entrada es la secuencia de imágenes que llegan desde la cámara y la salida un conjunto de "características". Estas características, describen la posición y categoría de cada uno de los jugadores observados por esa cámara (formalmente, estas características están formadas por: una coordenada en el plano de la cancha, su covarianza asociada y una estimación (probabilidad) de la categoría). Esta información es almacenada cuadro a cuadro en archivos XML con esquema ViPER [13]. El esquema ViPER es muy utilizado en el ámbito académico, especialmente en el área del procesamiento de imagen. Este esquema, nos brinda la posibilidad de poder utilizar aplicaciones como ViPER-GT [13] para visualizar o editar los resultados del proceso o ViPER-PE [13] para obtener datos de rendimiento.

El resultado del procesamiento de cada cámara es utilizado en la segunda etapa llamada "procesamiento multi-cámara". En esta segunda etapa todas las "características" que provienen de las cámaras, son integradas para crear y actualizar un modelo único del estado del juego. Este modelo es finalmente almacenado en un único archivo XML para ser utilizado por aplicaciones de terceros.

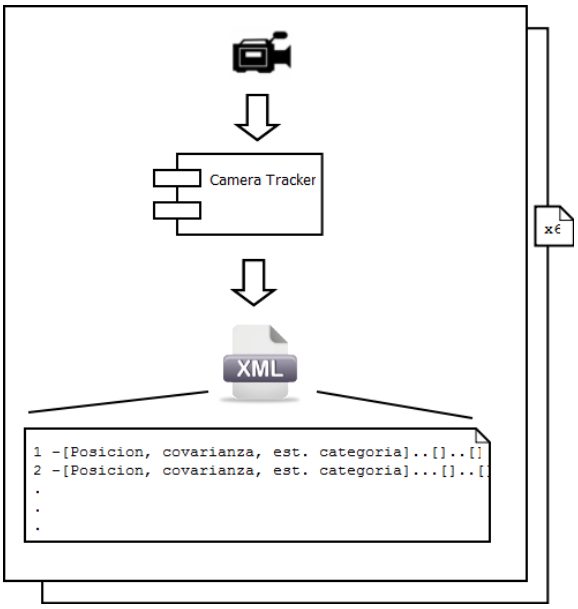


Figura 2.1: Modelo de la primera etapa de procesamiento.

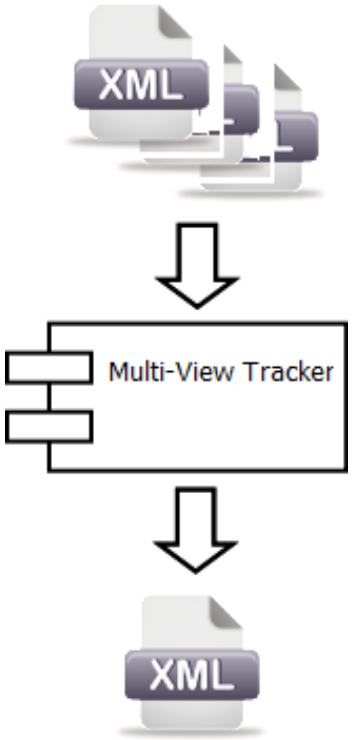


Figura 2.2: Modelo de la segunda etapa de procesamiento.

## 2.1. *Camera Tracker*

Veamos ahora, más en detalle la etapa de procesamiento por cámara (ver figura 2.3). Las imágenes que provienen de la cámara son enviadas a una "tubería" donde se les aplican en forma secuencial una serie de filtros. Es escalada (reducida), rectificada y finalmente se le aplica una máscara que cubre las zonas que no interesan para el procesamiento (tribuna, etc.).

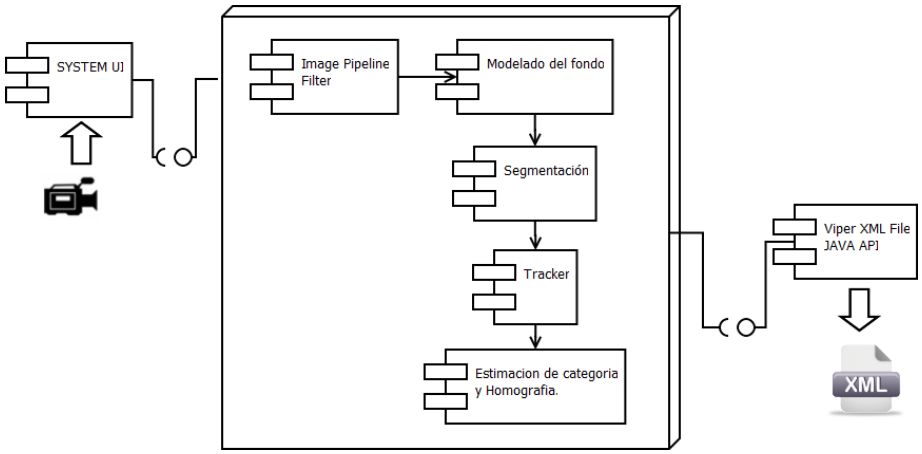


Figura 2.3: Descripción del procesamiento por cámara.

Luego la imagen es utilizada para crear un modelo del fondo. Este modelo es utilizado para separar los objetos móviles (que son los que nos interesan) de los del fondo. En la sección 4.1 se explicará detalladamente la técnica seleccionada para resolver este problema.

Una vez que se tiene el modelo del fondo, el siguiente paso es la segmentación. Esto quiere decir, separar los objetos del frente de los del fondo. Se espera obtener los objetos móviles de la escena (balón, jugadores, jueces). El balón es removido utilizando su forma y tamaño para detectarlo.

En este punto se cuenta con un conjunto de "manchas", regiones de la imagen donde se detecta un objeto. Estas manchas están descriptas por un identificador único, un rectángulo que las contiene y su área. Esta información es enviada al módulo de seguimiento donde cada mancha es asociada con un objetivo si se encuentra dentro de un umbral de distancia establecido o en caso contrario, un nuevo objetivo es iniciado y asociado con esta mancha. Este proceso es explicado con detalle en la

sección 4.2.

El siguiente módulo se encarga de brindar una estimación estadística de la categoría de cada uno de los objetivos que están activos en ese instante. La información del color es utilizada para esta tarea. Técnica abordada en la sección 4.3. Además se calcula la homografía del objetivo que es la transformación de coordenadas en la imagen a coordenadas en el plano de la cancha. Esto se realiza utilizando la matriz de transformación H de la cámara. Finalmente la API de ViPER es utilizada para enviar la salida del proceso a un archivo XML.

## 2.2. *MultiView Tracker*

Veamos ahora con más detalle el procesamiento de la segunda etapa (Ver figura 2.4). Los archivos XML de la primera etapa son utilizados para crear y actualizar un modelo único del campo de juego. El módulo *MultiView Tracker* utiliza las observaciones que brinda cada cámara sobre un objeto para realizar una estimación más precisa sobre su posición y categoría. El capítulo 5 está dedicado enteramente a describir esta técnica. Nuevamente, la API de ViPER es utilizada para enviar la salida del proceso y por lo tanto del sistema a un archivo XML.

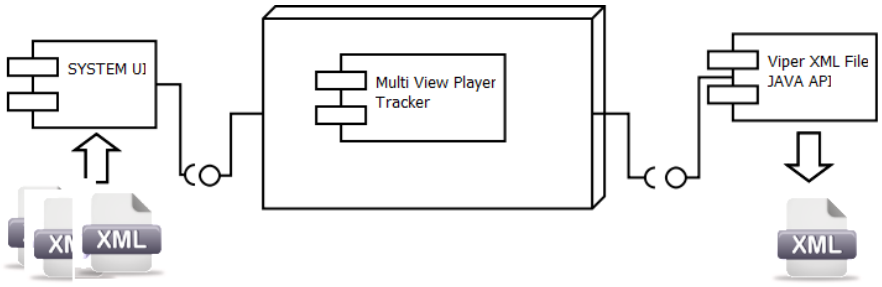


Figura 2.4: Descripción del procesamiento multi-cámara.

## 2.3. Arquitectura de software del sistema

Se creó un diseño modular que hace fuerte uso de los patrones *Template Method*, *Strategy* y *Singleton* [10]. Cada módulo se encuentra encargado de resolver una parte específica del problema en forma independiente del resto y todos colaboran para llegar a la solución global. Se utiliza el patrón *Strategy* para poder intercambiar distintas versiones de un mismo algoritmo y comparar el rendimiento del sistema con cada uno. El orden en que son invocados los módulos, es controlado por el módulo **System**. Utilizando el patrón *Template Method*, **System** delega la responsabilidad del algoritmo principal en los módulos específicos.

Con respecto a la implementación, se decidió utilizar la herramienta MatLab por varias razones: esta herramienta es ideal para prototipos y nos permite fácilmente abordar el problema dejando de lado dificultades relacionadas a los lenguajes de programación, como ser manejo de memoria a bajo nivel, punteros, estructuras de datos, manejo de errores, etc. También influyó en la decisión de utilizar Matlab como herramienta el factor que la misma es conocida por todos los integrantes del grupo. Creemos importante mencionar el grado de dependencia que tiene el código con Matlab. Es decir, el esfuerzo que requeriría portarlo a otros lenguajes. Las funciones de *Image Processing Toolbox* de MatLab que se utilizaron son: **regionprops** detecta y retorna las componentes conexas en una imagen binaria y **bwmorph** operaciones morfológicas sobre imágenes binarias (limpieza, clausura, dilatación, erosión). Este tipo de funciones es común en otros *frameworks* de procesamiento de imágenes como OpenCV [18] u OpenFramework [19]. El resto de las funciones, algoritmos, clases, interfaces, estructuras de datos, etc., fueron enteramente implementadas por el grupo. De esta forma, queda claro que el código podría ser portado a otros lenguajes y/o plataformas sin mayor dificultad.

Además, como fue ya mencionado, se utilizan llamadas a la API del sistema ViPER para almacenar y cargar archivos XML a través de un componente desarrollado por el grupo. A continuación, se presenta un diagrama de clases de la primera etapa (ver figura 2.5) junto con la descripción de cada uno de los módulos.

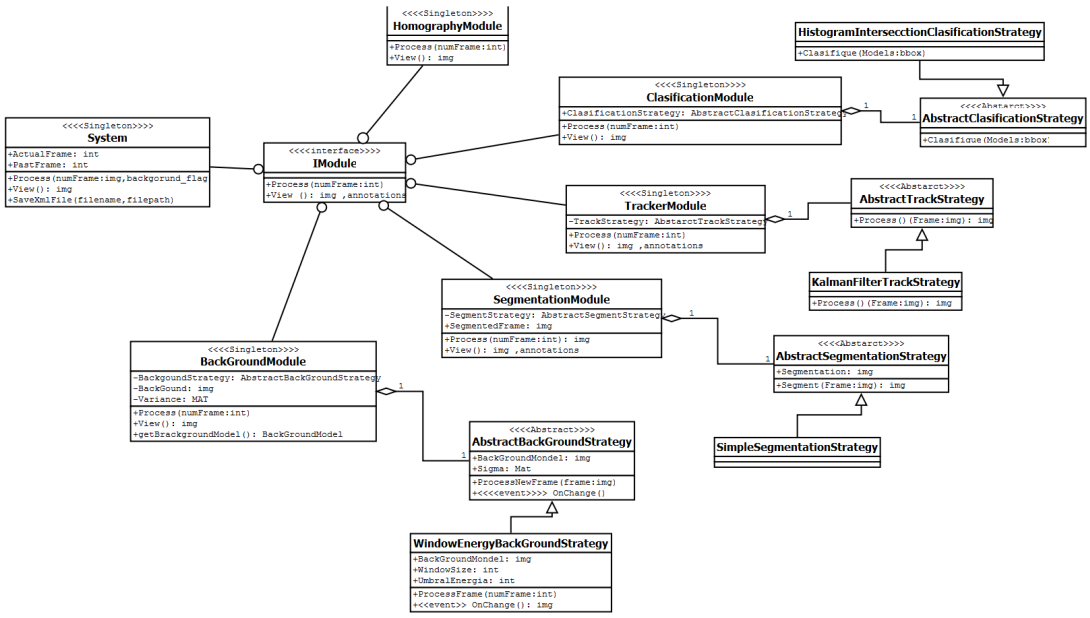


Figura 2.5: Módulos.

### 2.3.1. Módulos

Cada módulo es responsable de resolver un aspecto específico del problema. Los módulos cumplen con el patrón *Singleton* [10] permitiendo que su única instancia sea accesible desde cualquier lugar del sistema. Asimismo los módulos implementan la interfaz **IModule**, que define dos operaciones:

**Process(int frameIndex) , image=View()**

La primera es la encargada de realizar el procesamiento del cuadro indicado y la segunda es utilizada para mostrar al usuario en la UI los resultados parciales en cada paso. También se puede observar en muchas clases el uso del patrón *Template Method* [10]. Por ejemplo la clase **AbstractBackgroundStrategy** define la interfaz de los algoritmos encargados de crear y mantener el fondo. La clase concreta **WindowEnergyBackgroundStrategy**, resuelve este problema utilizando una técnica descrita en la sección 4.1. El módulo **BackgroundModule** mantiene una referencia a una instancia de alguna clase que implemente a **AbstractBackgroundStrategy**. Este mecanismo nos permitió probar e intercambiar distintos tipos de algoritmos que resuelven el mismo problema con facilidad.

La clase **GlobalParameters** (*Singleton*), mantiene el conjunto de parámetros utilizados en el procesamiento.

### System

Este módulo representa un punto de entrada único al sistema, oficia de *Facade* [10]. Este sistema está basado en un procesamiento "por cuadro". También es el encargado de orquestar la ejecución de los módulos en el orden correcto para procesar un cuadro.

### Video data manager

Este módulo es responsable de proveer los cuadros de video. Antes de ser utilizados por lo demás módulos, es necesario aplicar una serie de transformaciones a cada cuadro. Estas transformaciones son encapsuladas en las clases que implementan la clase abstracta **AbstractFilter** (ver figura 2.6) y son aplicadas en forma secuencial al cuadro. El patrón utilizado en este caso es el de Tubería o *pipeline* [10]. De esta forma es posible agregar o remover filtros de forma relativamente simple.

### BackgroundModule

Este módulo es responsable de crear y mantener el fondo del sistema. Como ya fue dicho, el fondo es utilizado para separar los objetos del frente (segmentar) y de esta forma poder detectar las manchas. Existen muchas técnicas para llevar adelante esta tarea. En este caso se utiliza la técnica MoG (*Mixture of Gaussians*) que es implementada en la clase **WindowEnergyBackGroundStrategy** y descripta en la Sección 4.1.

### SegmentationModule

Este módulo es responsable de la etapa de segmentación, separar los objetos móviles del la escena. En este caso la técnica utilizada es encapsulada en la clase **SimpleSegmentationStrategy** y es estudiada a fondo en la Sección 4.2.

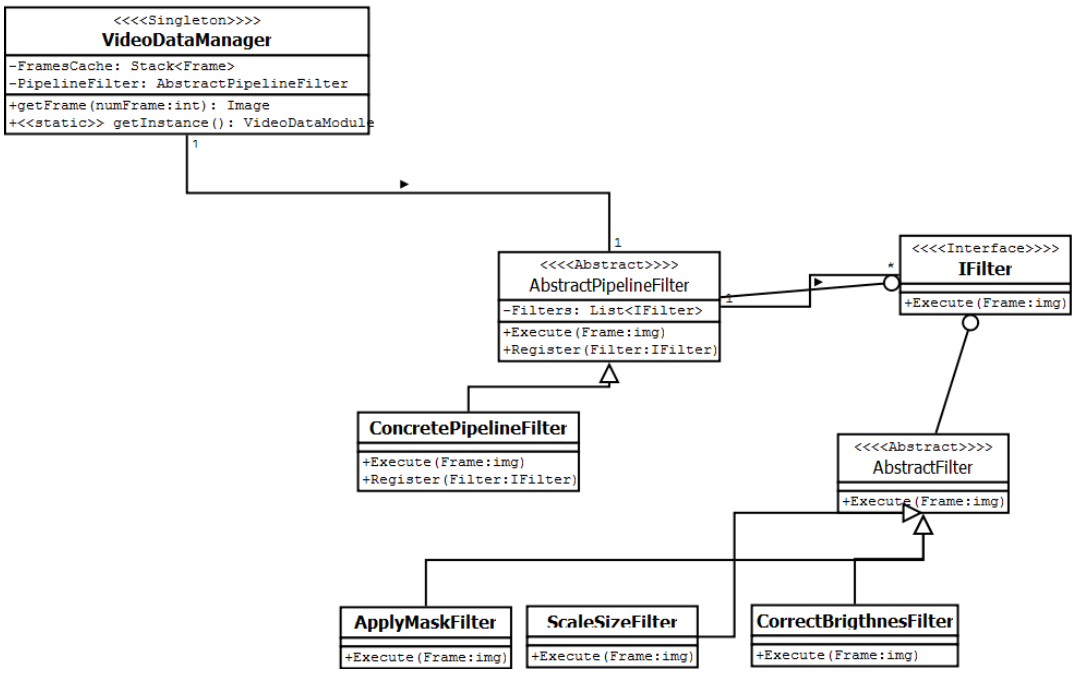


Figura 2.6: VideoDataManager.

### TrackerModule

Una vez identificados las manchas en el cuadro, es necesario realizar el seguimiento que permite determinar que un objeto es el mismo a lo largo de una secuencia de cuadros. La técnica utilizada para resolver este problema es el llamado filtro de Kalman (ver Apéndice B). La lógica se implementa en la clase **KalmanFilterTrackStrategy** y en la Sección 4.2 será detallado su funcionamiento.

### ClasificationModule

Este módulo es el encargado de clasificar los objetivos, es decir, calcular la probabilidad de pertenecer a una clase. El módulo retorna un vector de valores reales (probabilidades) para las 5 clases de objetos en el sistema (Golero 1, Golero 2, Cuadro 1, Cuadro 2, Juez).

La técnica utilizada para esta tarea es la llamada intersección de histogramas de

color que será explicada en la Sección 4.3.4 y es implementada en la clase **HistogramIntersecctionClasificationStrategy.**

### HomographyModule

La salida de la primera etapa contiene una coordenada en el plano de la cancha (mundo real) de cada objetivo. Este es el módulo encargado de realizar esta tarea, utilizando los valores de la matriz H (homografía de la cámara al plano de la cancha). Esta matriz se ingresa como parámetro al sistema (por cámara) y es calculada utilizando técnicas clásicas de calibración como por ejemplo T'sai (Ver **Apéndice A**).

### TargetManager

Un objetivo establecido es un objeto detectado. Idealmente un objetivo representa un jugador. Decimos idealmente porque como veremos en la práctica existen oclusiones, falsas detecciones, solapamiento, etc. El módulo **TargetManager** mantiene la colección de objetivos en todo instante de ejecución del sistema. Es el responsable de dar de alta, baja y conocer el estado de cada objetivo.

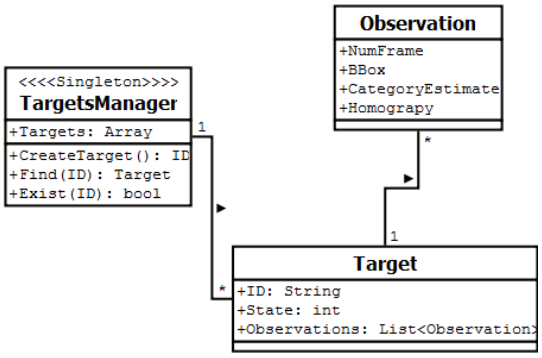


Figura 2.7: Objetivos.

### XMLFileManager

Este módulo es encargado de persistir y cargar los datos que utiliza el sistema. También este módulo realiza una llamada a un componente Java **(GTDataFingCameraTracking.jar)** que a su vez realiza finalmente la llamada a la API de ViPER GT. De la misma forma, este módulo es utilizado en la segunda etapa para cargar y almacenar los datos.

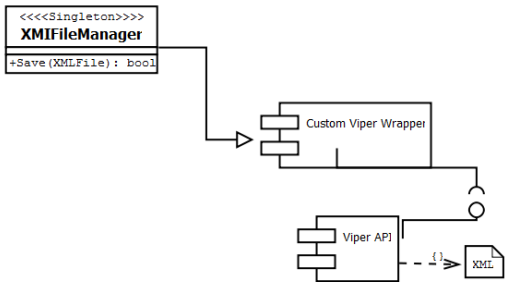


Figura 2.8: XMLFileManager.

### Interfaz de usuario

Se creó una interfaz de usuario con el fin de facilitar la tarea del procesamiento por cámara.

En la imagen puede observarse una captura de pantalla de esta interfaz (ver figura 2.9). Las imágenes que se ven en el centro, de izquierda a derecha y de arriba a bajo, corresponden a: cuadro actual, modelo del fondo, segmentación y homografía. El usuario puede seleccionar el número de la cámara a procesar. El sistema también cuenta con una modalidad desatendida o *batch* de procesamiento en la que es posible procesar todas o algunas de las cámaras. El usuario provee el archivo de parámetros para el juego de datos que se quiere procesar, luego selecciona la cantidad de cuadros a procesar y presiona el botón Start. El resultado del proceso es visualizado cuadro a cuadro en la interfaz. Una explicación más detallada de las funcionalidades y del archivo de parámetros puede encontrarse en el **Apéndice D**.

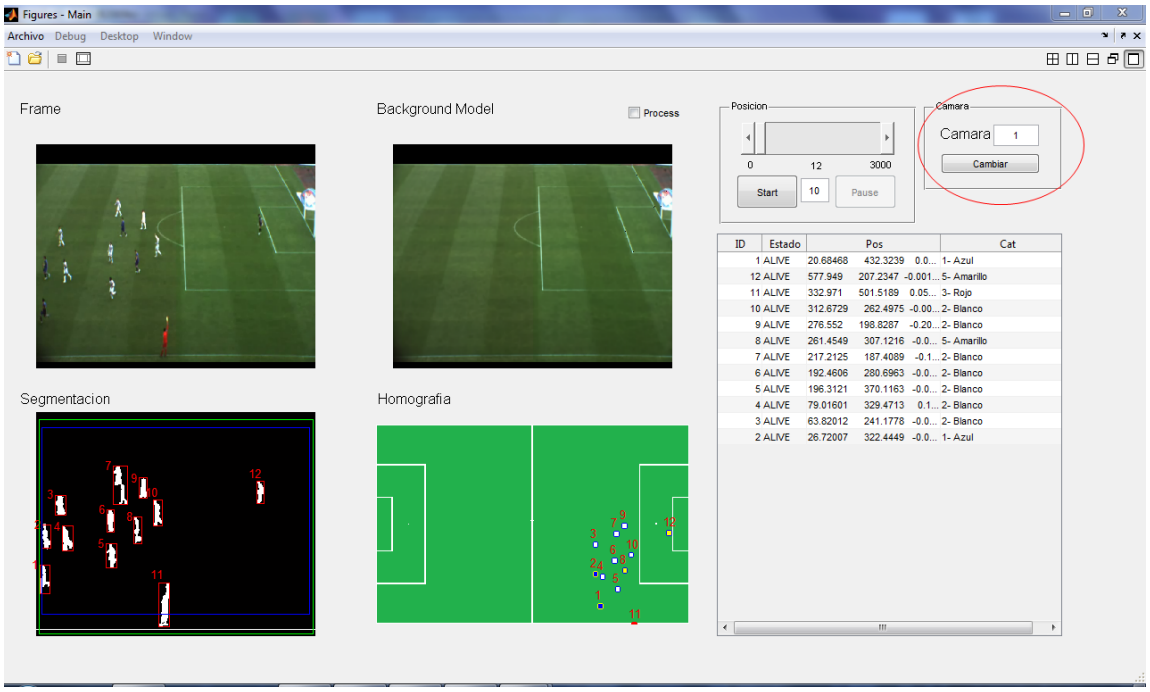


Figura 2.9: Interfaz de usuario de Camera Tracking.

### 2.3.2. Segunda Etapa (seguimiento multi-cámara)

La segunda etapa del sistema hace uso de varios de los módulos ya presentados en la primera, como por ejemplo el de persistencia **XMIFileManager** o **TargetManager**. La lógica principal de la segunda etapa se encuentra en la clase **MultiViewModule**. Esta es la encargada de unificar las observaciones de las cámaras en un modelo común que es actualizado cuadro a cuadro. Un detalle de cómo se realiza esta tarea puede encontrarse en el capítulo 5. Como en la primera, el resultado o salida es un archivo XML con esquema ViPER.

Se creó una interfaz de usuario que permite a un operador fácilmente poner en funcionamiento el sistema (ver figura 2.10). En la imagen puede verse una captura de pantalla de la interfaz. El usuario debe proveer el archivo de parámetros que contiene los XML de la primera etapa. Luego indica la cantidad de cuadros a procesar y presiona el botón Start. La representación del modelo unificado junto con los objetos detectados pueden verse en el centro de la imagen cuadro a cuadro. El color representa la categoría de cada uno de los objetos. El sistema también cuenta con una

modalidad desatendida o *batch* de procesamiento. Para una descripción detallada del funcionamiento ver el **Apéndice D**.


Figura 2.10: UI de Multi-View Tracking.

# Capítulo 3

# Conjunto de datos de prueba

Surge la necesidad de contar con un conjunto de datos de prueba exhaustivo sobre el dominio de interés y suficientemente representativo del conjunto de datos con el en que operara el sistema en producción.

## 3.1. Conjunto de datos ISSIA

En una primera instancia se trabajará con un juego de videos obtenido de ISSIA [9]. El conjunto de datos cuenta con seis videos sincronizados adquiridos por seis cámaras fijas Full-HD (compresión MPG) de 2 minutos de duración. Las cámaras cubren todo el terreno de juego. La información de calibración y los datos de *ground truth* (GT) (archivos XML con la posición real de los jugadores y balón durante la duración de secuencia).

En la figura 3.1 se puede ver un diagrama de la ubicaciones de las seis cámaras y en la figura 3.2 se pueden ver un ejemplo de las imágenes capturadas por cada una de las cámaras. En la tabla 3.1 se muestra un resumen de las características del juego de datos.

A la hora de trabajar con este juego se realizaron algunas modificaciones. En primer lugar se bajó la resolución de los videos de 1920x1080 a 720x576 para reducir los tiempos de procesamiento. Se detecto el problema de espejado en la cámara 2 y en la cámara 6, es decir las imágenes están rotadas horizontalmente. Si bien se cuenta con los archivos de GT estos no son del todo útiles para medir el rendimiento

| | |
|---|---|
| **Resolución** | 1920x1080 |
| **Formato** | MPG |
| **Duración** | 00:02:00 / 3000 cuadros |
| **Datos para calibración** | SI, puntos en el piso |
| **Ground truth** | SI, para todas las cámaras |

Cuadro 3.1: Conjunto de datos ISSIA.

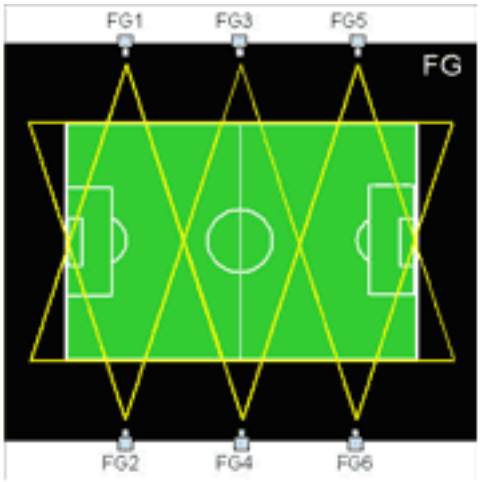


Figura 3.1: Conjunto de datos ISSIA. Las posiciones de las seis cámaras en ambos lados del campo de juego.

dado que solo hacen referencia a la ubicación de los objetos detectados pero no a qué categoría pertenecen (falta información de categoría). Este es una dato mas, de la salida de nuestro sistema. Debido a ello y a que los videos fueron redimensionados hubo que realizar nuevamente el GT a mano agregando una nueva propiedad que indica a qué clase o categoría pertenece cada jugador.

Encontramos favorable la simetría en la ubicación de las cámaras y la altura de las mismas parece ser la adecuada dado que no se detectan gran cantidad de oclusiones como en otros juegos de datos. Por estas razones se decidió utilizar este juego de datos para comenzar el desarrollo y entrenamiento del sistema.

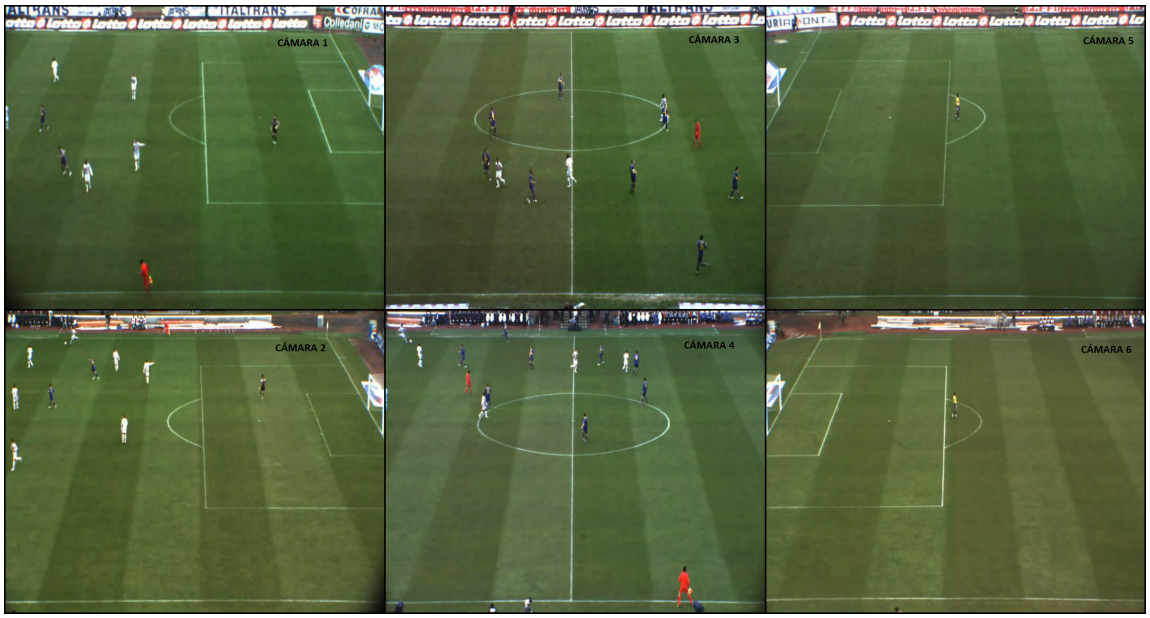


Figura 3.2: Conjunto de datos ISSIA. Imágenes adquiridas por las seis cámaras.

## 3.2. Conjunto de datos SCEPTRE

El juego de datos SCEPTRE, [14] cuenta con ocho videos sincronizados de 2 minutos y medios de duración aproximadamente. Es un partido de la liga inglesa (Fulham vs Manchester United). Los ocho videos han sido obtenidos por ocho cámaras fijas colocadas en el estadio de manera de cubrir todo el terreno de juego. Junto con los videos se obtuvieron los datos de calibración de las cámaras. En la figura 3.3 se puede observar las posiciones de las cámaras y en la figura 3.4 un ejemplo las imágenes capturadas por las mismas. En la tabla 3.2 puede verse un resumen de las características del juego de datos.

| | |
|---|---|
| **Resolución** | 720x576 |
| **Formato** | AVI |
| **Duración** | 00:02:41 / 4025 cuadros |
| **Datos para calibración** | SI, constantes de calibración Tsai |
| **Ground Truth** | NO |

Cuadro 3.2: Conjunto de datos SCEPTRE.

A la hora de trabajar con este juego de datos, encontramos algunas dificultades.

Primero se puede observar el efecto de entrelazado en algunos videos. Esto afecta al proceso de clasificación de categorías. Para tratar de resolver este problema se aplicó el filtro deinterlace (elimina los efectos de scanline del video entrelazado, parte basado en el algoritmo de desentrelazado Yadif por Michael Niedermayer) en el programa VirtualDub, [1]. Esto mejoró bastante la calidad de las imágenes. También surge el problema del movimiento de las cámaras (vibración) esto sucede solo por algunos segundos en algunas de las cámaras. Suponemos se debe al viento o a movimientos en la estructura donde están instaladas las cámaras. Otro de los inconvenientes encontrados está relacionado con el campo visual de las cámaras, por ejemplo una de ellas toma prácticamente toda el terreno de juego lo que genera que se produzcan demasiadas oclusiones.

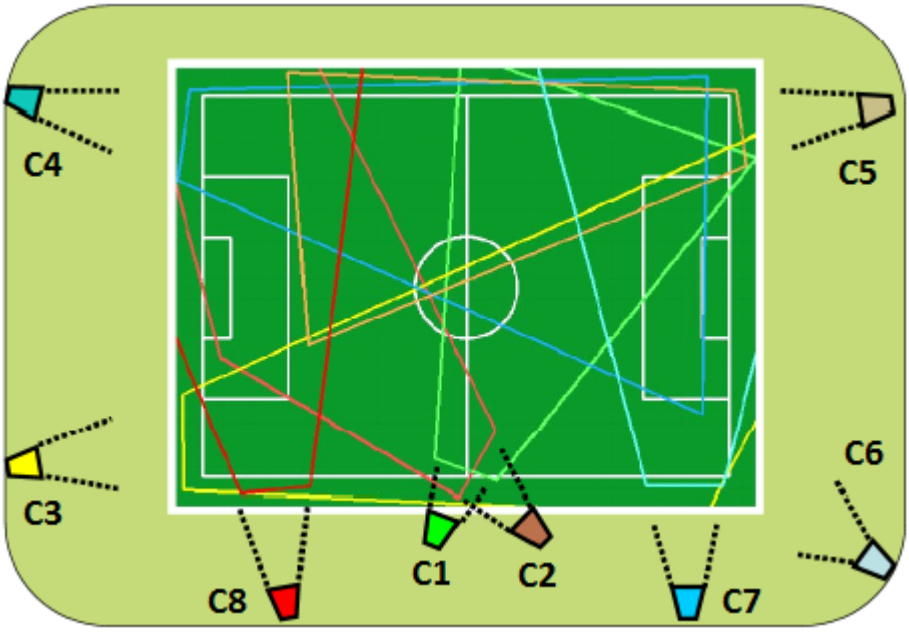


Figura 3.3: Conjunto de datos SCEPTRE. Posiciones de las ocho cámaras en el campo de juego.

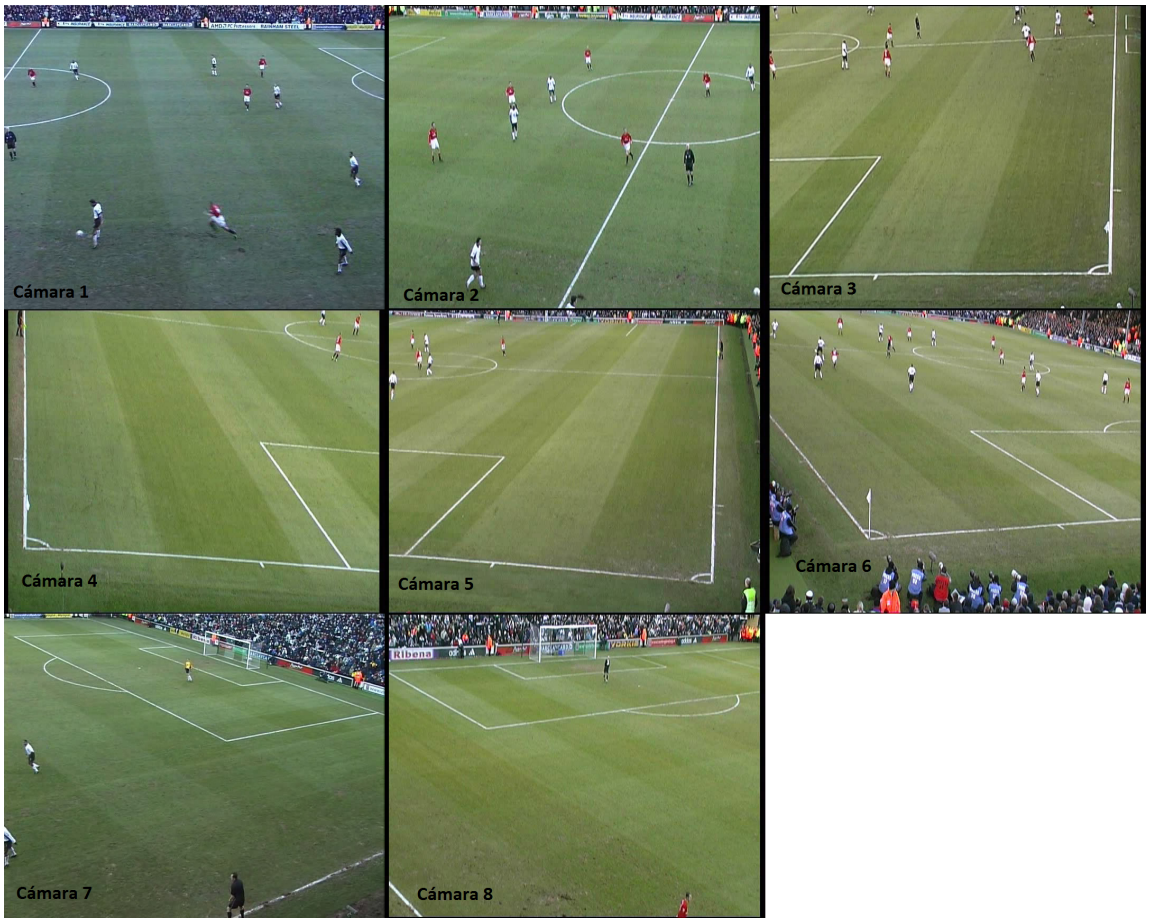


Figura 3.4: Conjunto de datos SCEPTRE. Imágenes adquiridas por las 8 cámaras respectivamente.

## 3.3. Datos obtenidos por el grupo

La Fundación Julio Ricaldoni [21] en su iniciativa "Emprendedores Dinámicos de la Facultad de Ingeniería" ha decidido apoyar este proyecto luego de ser evaluado entre muchas otras propuestas realizadas. El apoyo brindado a los emprendedores incluye la capacitación a medida, la designación de un tutor especialista en un área de conocimiento relevante para el emprendimiento, así como también un subsidio de hasta 2.500 dólares para el desarrollo de su prototipo y/o ensayo. El docente designado por la Fundación Ricaldoni es Pablo Cancela quien ya ha sido mencionado anteriormente dado que es también unos de los tutores de este proyecto por parte

del Instituto de Ingeniería Eléctrica.

El apoyo económico brindado por la Fundación Ricaldoni será destinado para el alquiler de cuatro cámaras idénticas (ya que es la máxima cantidad de cámaras idénticas que se pueden obtener por parte del proveedor, HTV-3 TAJAM [24]). Se grabaron dos partidos con cuatro cámaras y un partido con una sola cámara. El modelo de las cámaras utilizadas es Sony XDCAM EX3 High Definition [6].

### 18 Noviembre 2010: Partido amistoso Uruguay-Argentina Sub 17 en Gran Parque Central

Se filmó con una sola cámara ubicada en diferentes sectores de la tribuna principal. La idea fue grabar distintos transcursos del partido desde diferentes sectores, esto sirve para probar el seguimiento de jugadores desde una vista o cámara. En la figura 3.5 se muestra una captura desde una de las posiciones elegidas para grabar.

Los videos originales tienen una resolución de 1920x1080 la cual se redujo a 720x576 para reducir el tiempo de procesamiento. Si bien la primera experiencia de campo creemos fue satisfactoria, el juego de datos obtenido introdujo dos nuevos problemas. Las imágenes están muy saturadas como claramente se puede ver figura 3.5. La presencia muy marcada de las sombras de los jugadores sobre el campo de juego, atributo que no se encontraba en los anteriores juegos de datos. Esto dificulta el proceso de segmentación como será visto mas adelante.

### 3 Marzo 2011: Partido amistoso Miramar Misiones - Durazno (Parque Méndez Piana)

Las filmaciones se realizaron con 4 cámaras (Sony XDCAM EX3 - HD [6]) idénticas, las cuales se instalaron en una única tribuna (la tribuna de mayor altura).

Las cámaras se ubicaron como se muestra en la figura 1.1. Se pueden ver las imágenes capturadas por cada una de los cámaras en la figura 3.6. Luego se utilizó otra disposición que fue llevar 2 cámaras a cada esquina y dejar 2 en el centro de la cancha. Se filmó un total de 30 minutos con cada disposición.

El inconveniente de este juego de datos es la baja altura de las cámaras lo que genera un gran número de oclusiones, ademas nuevamente aparecen las sombras de los jugadores al igual que en los videos obtenidos en el Parque Central. A estos dos

Figura 3.5: Captura desde una las posiciones elegidas, 18/11/10 Parque Central.

factores, se suma el de que las líneas de la cancha prácticamente no se ven lo cual hace sumamente difícil sino imposible el cálculo de la homografía ya que no hay puntos de referencia a tener en cuenta para hacer el mapeo.

### 26 Marzo 2011: Partido amistoso Defensor Sporting - River Plate (Estadio Luis Franzini)

Este juego datos fue grabado el día 26 de marzo de 2011 en el estadio Luis Franzini en un encuentro amistoso disputado entre el club Defensor Sporting y el conjunto de River Plate.

Se obtuvieron dos conjuntos de datos ya que se jugaron dos partidos, a primera hora se enfrentó la primera división mientras que en el segundo partido se enfrentaron las reservas. Para el primer partido se colocaron las cuatro cámaras como se muestra en la figura 3.7. En la figura 3.8 se pueden apreciar las capturas de cada cámara. Para el partido de segunda hora se quitaron las cámaras 3, 4 y utilizaron solamente las cámaras 1 y 2.

Se grabaron 13 minutos de juego (19500 cuadros) de cada partido con una resolu-


Figura 3.6: Imágenes adquiridas por las 4 cámaras, 03/03/11 Méndez Piana.

ción de 1920x1080. Nuevamente, la altura de las cámaras no es la ideal, la estructura edilicia del estadio no permite ubicarlas a mayor altura. Por momentos se observan sombras de los jugadores sobre el campo de juego. Se detectan pequeños movimientos debido al viento por instantes en algunas de las cámaras. Se escalaron y recortaron las imágenes, es decir se quitó de todo aquello que no era campo de juego y se dejó solo lo que tenía información relevante para el procesamiento. Finalmente se trabajó con una resolución de 1280x430. En las tablas 7.7 y 7.8 se muestran las características finales de los videos con los que se trabajó.

| | |
|---|---|
| **Resolución** | 1280x430 |
| **Formato** | AVI |
| **Duración** | 00:01:54 / 2868 cuadros |
| **Datos para calibración** | plano de la cancha |
| **Ground Truth** | NO |

Cuadro 3.3: Conjunto de datos Franzini_1.

| | |
|---|---|
| **Resolución** | 1280x430 |
| **Formato** | AVI |
| **Duración** | 00:01:59 / 2990 cuadros |
| **Datos para calibración** | plano de la cancha |
| **Ground Truth** | NO |

Cuadro 3.4: Conjunto de datos Franzini_2.

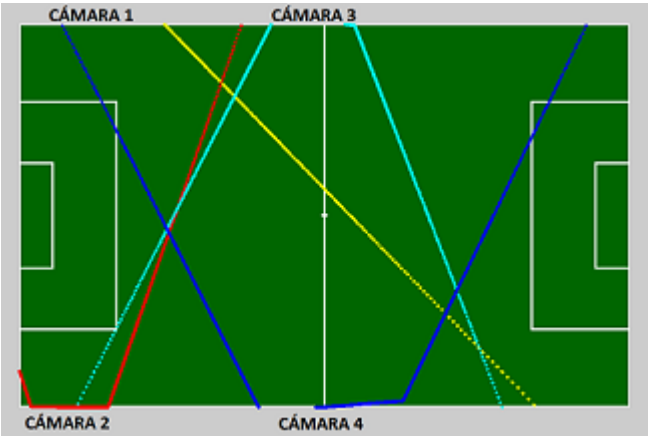


Figura 3.7: Conjunto de datos Franzini. Posiciones de las 4 cámaras en el campo de juego.

Como se puede ver en la figura 3.7 se intenta imitar la disposición de cámaras utilizada en el juego de datos obtenido de ISSIA ya que se consideró que era más práctica y la más eficiente dado que cubre el campo de juego de manera más completa y utilizando menos cámaras que la opción utilizada en los videos de SCEPTRE.

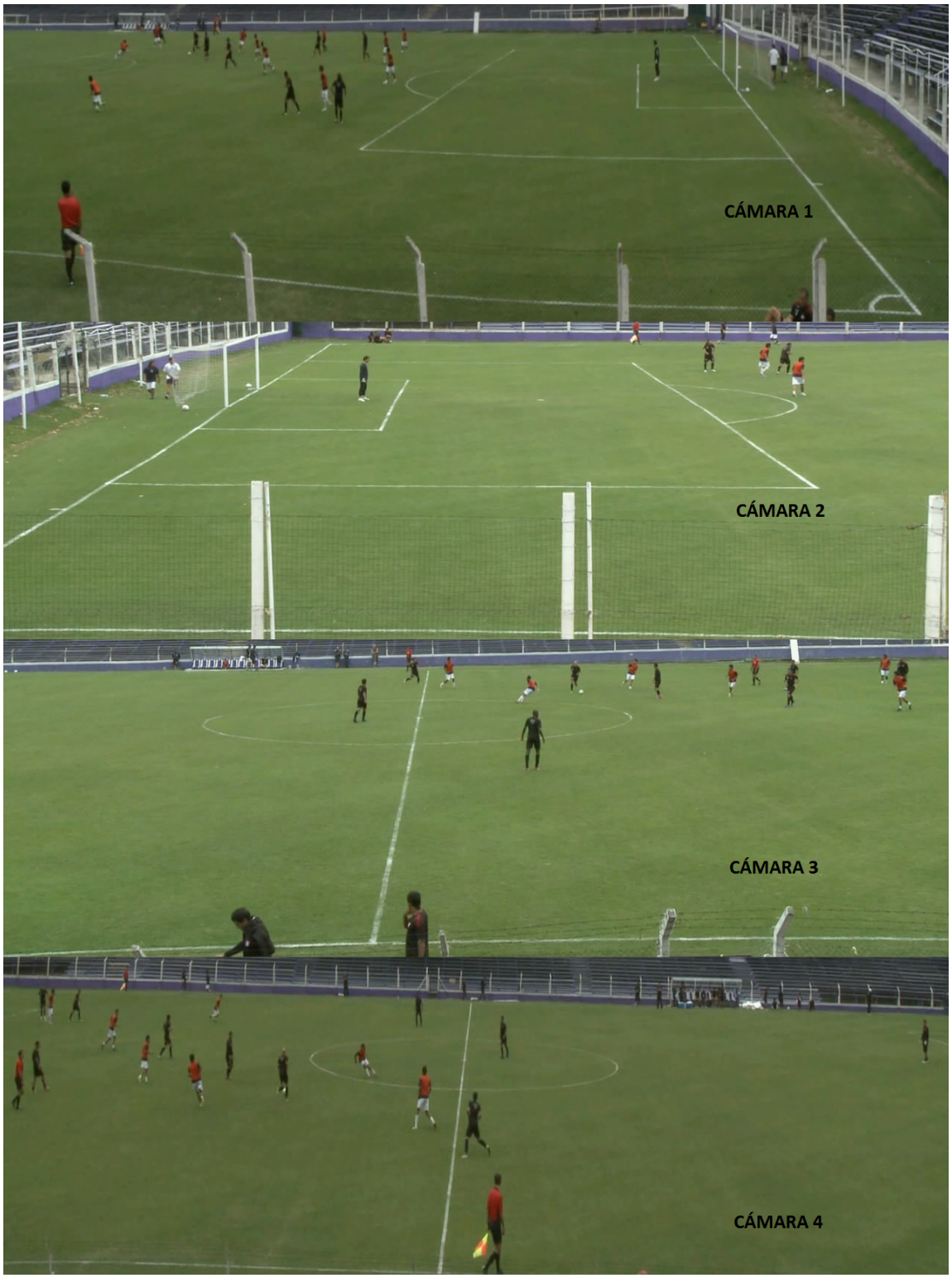


Figura 3.8: Conjunto de datos Franzini. Imágenes adquiridas por las 4 cámaras respectivamente.

## 3.4. Comparación de los juegos de datos

Una vez presentados los juegos de datos con los que se trabajará y luego de haber descripto sus principales características podemos compararlos para determinar cual de todos resulta más útil a la hora de entrenar el prototipo. Los juegos de datos con los que se cuenta son: ISSIA, SCEPTRE, Franzini_1 y Franzini_2.

Como primera diferencia a destacar entre los juegos de datos es que tanto los videos de ISSIA y SCEPTRE cubren todo el campo de juego mientras que los videos grabados en el estadio Luis Franzini en un caso cubren aproximadamente tres cuartos de cancha y en el otro solamente un cuarto de cancha. Por esta razón estos videos se utilizarán para entrenar y probar el procesamiento por cámara y no para el procesamiento multi-cámara como sí se utilizaran los juegos de datos de ISSIA y SCEPTRE.

La altura y disposición de las cámaras es un factor fundamentar. Cuanto mas altura, menos oclusiones vamos a encontrar. La disposición afecta el campo visual de cada cámara. Llegamos a la conclusión de que es preferible que la cámara tome una zona reducida de la cancha pero utilizando mas resolución, en oposición de las cámaras que tomar prácticamente todo el terreno, pero la perspectiva hace que sea muy difícil el procesamiento en zonas alejadas del fondo. Otro de los factores importantes es el movimiento de las cámaras ya que genera detección de las líneas de la cancha a la hora de segmentar.

Teniendo en cuenta estos factores llegamos a la conclusión de que el juego de datos de ISSIA es el mas completo y de mejor calidad. Ninguna de las seis cámaras se mueve durante los 2 minutos de juegos grabados. La altura de las cámaras es la adecuada ya que no se generan demasiadas oclusiones y la ubicación en la cancha de las mismas hace que cada jugador sea visto de por lo menos dos cámaras generando así dos medidas para el procesamiento multi-cámara.

En SCEPTRE se encuentran algunos de los problemas mencionados. Por ejemplo algunas cámaras vibran durante parte del video sufriendo este problema de mayor manera la cámara 5, otras cámaras también vibran como por ejemplo las cámaras 1 y 3 pero por menor lapso de tiempo. Otro problema que afecta a algunos de estos videos es el efecto de entrelazado como se puede ver en las cámaras 1, 3 y 7. En cuanto a la altura de las cámaras se considera que es el correcto, prácticamente similar a la

altura de los videos de ISSIA. En cuanto a la ubicación de las cámaras en el campo de juego se encuentra que quizás hay cámaras que no aportan demasiado. Por ejemplo la vista de la cámara 6 abarca casi toda la cancha lo cual no es lo más deseable ya que si se colocan muchas cámaras lo mejor es que cada una abarque un sector de la cancha para poder resolver mejor las oclusiones en dicho sector.

En cuanto a los videos grabados en el estadio Franzini (Franzini_1 y Franzini_2) el mayor inconveniente que encontramos es la altura de las cámaras, y en consecuencia la cantidad de oclusiones que esto genera. En cuanto a la ubicación de las cámaras en la cancha se intenta imitar el esquema utilizado en los juegos de datos de ISSIA. No hay problemas de movimiento de las cámaras ya que se eligió un tramo de los videos originales donde esto no sucede.

En el juego de datos Franzini_2 dado que solo se tiene dos cámaras enfocando a una de las áreas se puede observar que no hay demasiada actividad en los videos, excepto por un par de jugadas en la mayoría de los cuadros aparece solo el golero.

## 3.5. Generación de datos de *ground truth*

Se llama datos de *ground truth* o simplemente GT, a los datos contra los cuales se compara la salida del sistema determinando cuán cerca están los mismos del valor "verdadero".

Por diferentes razones como hemos explicado anteriormente hemos tenido que generar el GT para todos los juegos de datos. Para los datos de ISSIA porque el GT obtenido no contemplaba la categoría de los jugadores y para los demás datos simplemente porque no los teníamos.

Éste es un trabajo tedioso y que consume mucho tiempo. Ya que cuadro a cuadro se debe crear un rectángulo para encerrar a cada jugador y además indicar su categoría. Para generar los datos de GT utilizamos la herramienta ViPER-GT [13] (ver Apéndice C).

ViPER-GT ayuda al proceso de creación de GT mediante una interfaz gráfica de usuario Java. Está diseñado para permitir cuadro a cuadro el marcado de datos del video y almacenarlos en formato ViPER xgtf o XML. También es útil para la visualización. Para más información, consulte la página del software, [13]. En la

figura 3.9 podemos ver la interfaz gráfica del ViPER-GT y todas las opciones que se presentan para la generación de datos de GT.

En el Apéndice C se explica detalladamente el proceso de creación de los archivos de GT.

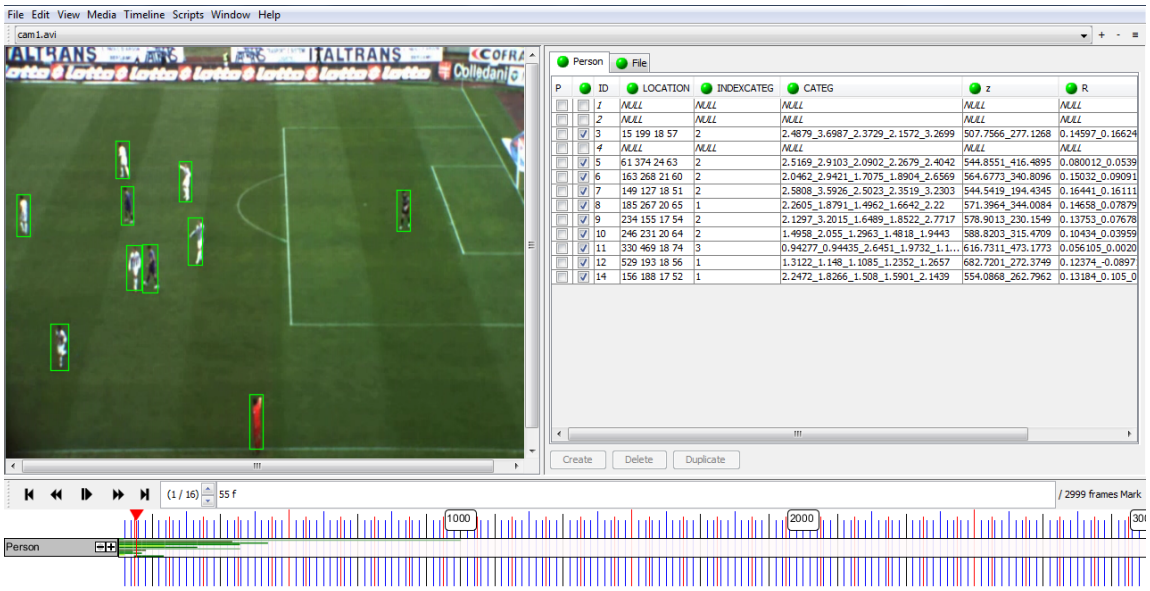


Figura 3.9: Interfaz gráfica del programa ViPER-GT.

# Capítulo 4

# Procesamiento por cámara

## 4.1. Detección del fondo

El primer y fundamental paso del procesamiento por cámara, es la separación entre el fondo y los objetos móviles del frente. En este problema se espera que los objetos del frente sean jugadores, balón, jueces y el fondo simplemente el campo de juego.

El objetivo de la técnica utilizada es crear un modelo del fondo, que sea independiente de los datos de entrenamiento, de las variaciones en las condiciones de luz de la escena, de los objetos móviles (personas). En [15], se propone una técnica no supervisada que combina un análisis temporal de la intensidad por píxel, con una ventana deslizante para poder construir el modelo incluso en presencia de objetos móviles, durante la fase de entrenamiento.

La implementación del algoritmo está basada en dos fases, primero la energía de cada píxel de la imagen es evaluada en una pequeña ventana corrediza que es usada para distinguir entre puntos estáticos y en movimiento (los valores bajos de energía son utilizados para construir el fondo mientras que los altos se descartan por la probabilidad de pertenecer a objetos móviles).

Se toma una ventana de tamaño W cuadros, donde siempre el primer cuadro $B_c(x, y)$ se toma como el modelo "grueso" del fondo a refinar. Formalmente, en cada cuadro (instante $t$) el algoritmo evalúa media y desviación estándar en cada píxel $(x, y)$ de la imagen usando las siguientes ecuaciones:

$$\overline{\mu^t(x,y)} = \alpha\mu^t(x,y) + (1-\alpha)\overline{\mu^{t-1}} \tag{4.1}$$

$$\overline{\sigma^t(x,y)} = \alpha\left|\mu^t(x,y) - \overline{\mu^t(x,y)}\right| + (1-\alpha)\overline{\sigma^{t-1}} \tag{4.2}$$

Solo si la intensidad en ese punto varía poco con respecto al modelo de fondo:

$$\left|I^t(x,y) - B_c(x,y)\right| < th \tag{4.3}$$

Donde $th$ es un umbral determinado experimentalmente y $I^t(x,y)$ es la intensidad del punto (x,y) en el tiempo t.

De esta forma, al final del análisis de los primeros W cuadros, el algoritmo evalúa la componente de energía como:

$$E(x,y) = \int_{t\epsilon W} \left|I^t(x,y) - B_c(x,y)\right|^2 \tag{4.4}$$

El primer modelo "fino" $B_f$ es generado como:

$$B_f(x,y) = \begin{cases} (\mu(x,y), \sigma(x,y)) & \text{si} \quad E(x,y) < th(W) \\ 0 & \text{si} \quad E(x,y) > th(W) \end{cases} \tag{4.5}$$

Un valor bajo de energía significa que el punto en cuestión es de carácter estático y es incluído en el modelo del fondo, mientras que los puntos de alta energía, que corresponden al frente o a objetos del fondo en movimiento no contribuyen al modelo. Todo el procedimiento se repite en otra ventana de W cuadros a partir del cuadro W + 1. El modelo grueso del fondo es ahora el cuadro W + 1 y los nuevos valores estadísticos 4.1 y 4.2 son evaluados por cada punto, al igual que el contenido de la nueva energía 4.4.

La diferencia relevante con 4.5 es que ahora los nuevos parámetros estadísticos se promedian con los valores anteriores, si están presentes, de lo contrario, se convierten en los nuevos valores del modelo estadístico de fondo. Formalmente, la nueva formulación de 4.5 se convierten en:

$$B_F(x,y) = \begin{cases} (\mu(x,y), \sigma(x,y)) & \text{si} \quad E(x,y) < th(W) \wedge B_F(x,y) = 0 \\ \beta * B_F(x,y) + (1-\beta) * (\mu(x,y), \sigma(x,y)) & \text{si} \quad E(x,y) < th(W) \wedge B_F(x,y) \neq 0 \\ 0 & \text{si} \quad E(x,y) > th(W) \end{cases} \tag{4.6}$$

Todo el procedimiento se repite N veces, donde N puede ser un valor predefinido seleccionado experimentalmente para asegurar la cobertura completa de todos los píxeles. De lo contrario, para hacer el sistema menos dependiente de cualquier suposición a priori, se introduce un criterio dinámico de finalización el cual se verifica con facilidad. El procedimiento de modelado se detiene cuando un gran número de puntos del fondo tienen valores significativos:

$$\#(B_F(x,y) = 0) \approx 0 \tag{4.7}$$

En nuestro caso utilizamos ventanas de 2,5 segundos (equivalente a decir que W=63 cuadros en videos a 25 cuadros/segundo). El valor del parámetro $\alpha$ utilizado en 4.1 y 4.2 se ha fijado en 0,1 mientras que el valor de $\beta$ usado en 4.6 se fijo en 0,8. Estos valores se determinaron experimentalmente luego de varias pruebas. Los valores de N y th (umbral de energía) son parámetros de entrada de la función que se usa para crear el fondo y se modifican según el video a procesar. Uno de los factores a tener en cuenta es la actividad que existe en el tramo del video que se utilizará para sacar el fondo, si no hay demasiada actividad serán necesarias menos iteraciones. Ver un ejemplo de modelo en la figura 4.1.

## 4.2. Seguimiento por cámara

Luego de sustraer el fondo y obtener los objetos de interés, se está en condiciones de utilizar el modelo de seguimiento sobre el plano de la imagen.

Cada objeto detectado, será representado por un vector de estado $x_I$ y un vector de observaciones $z_I$:

$$x_I = \begin{bmatrix} r_c & c_c & r'_c & c'_c & \Delta r_1 & \Delta c_1 & \Delta r_2 & \Delta c_2 \end{bmatrix}^T \tag{4.8}$$

Figura 4.1: Resultado de aplicar el método de sustracción de fondo.

$$z_I = \begin{bmatrix} r_c & c_c & r_1 & c_1 & r_2 & c_2 \end{bmatrix}^T \tag{4.9}$$

Donde la letra "r" indica una coordenada que refiere a una fila y la letra "c" indica una coordenada que refiere a una columna dentro de la imágen. Las coordenadas del centroide del objeto son: $(r_c, c_c)$ y la velocidad respectiva de dicho centroide es: $(r'_c, c'_c)$. Las coordenadas $r_1, c_1, r_2, c_2$ representan el vértice superior izquierdo y el vértice inferior derecho del *bounding box* (BB) respectivamente. Mientras que las coordenadas $(\Delta r_1, \Delta c_1)$ y $(\Delta r_2, \Delta c_2)$ representan la posición relativa de los dos vértices opuestos al BB con respecto al centroide.

El filtro de Kalman [4] proporciona un marco para la estimación de una variable de la que se dispone de medidas (observaciones) a lo largo del tiempo (en este caso las medidas de las coordenadas detectadas tras la segmentación del objeto cuadro a cuadro dentro de la secuencia). Se trata de una técnica de estimación bayesiana empleada para modelar sistemas estocásticos dinámicos observados mediante sensores ruidosos (cámaras instaladas en el estadio).

Las ecuaciones que rigen la evolución de los estados y de la medida para dicho proceso son:

$$x_I(k+1) = A_I x_I(k) + w_I(k) \tag{4.10}$$

$$z_I(k) = H_I x_I(k) + v_I(k) \tag{4.11}$$

En donde llamamos $w_I$ y $v_I$ al ruido del proceso y de la medida respectivamente. Las matrices de transición de estados y de medida son:

$$A_I = \begin{bmatrix} 1 & 0 & T & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & T & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.12}$$

$$H_I = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.13}$$

Se espera que los objetos no sufran modificaciones bruscas tanto en la altura como en el ancho de un paso al siguiente. Notamos también que se asume un modelo lineal para cada objeto, un modelo de primer orden, velocidad constante dentro de un tiempo de observación T. La dinámica de movimiento de los jugadores de fútbol muchas veces no encaja bien en este modelo lineal. Especialmente cuando ocurren cambios bruscos de dirección y velocidad. En estos casos es convienente introducir alguna técnica de corrección del *tracking* como la que se verá en la sección Trabajo

a futuro, Corrección del *Tracking*.

Para cada objeto detectado, la predicción del estado siguiente se realiza con las observaciones del estado actual. Esta predicción será luego ajustada con la observación real. El seguimiento de múltiples objetos en movimiento y la perspectiva de las cámaras, da lugar a las llamadas oclusiones o agrupamientos. Es cuando dos o más objetos se unen en uno, desde el punto de vista de una cámara. En este caso, la observación o medida es asociada con múltiples objetos y típicamente el nuevo estado de cada objeto es calculado usando únicamente la estimación del paso anterior. Esto es llamado seguimiento a ciegas, ya que no hay observaciones reales del objeto.

En [26] se introduce una técnica que mejora sustancialmente el resultado del procedimiento descrito anteriormente. La idea es que un objeto puede ser parcialmente observable aun cuando se encuentra en una oclusión. En la imagen 4.2 puede verse un ejemplo de dos objetos agrupados y sus respectivos rectángulos contenedores. Aun así dos de los lados de cada rectángulo (marcados con un mayor grosor en la imagen) pueden ser tratados como observados dada la geometría de la agrupamiento.

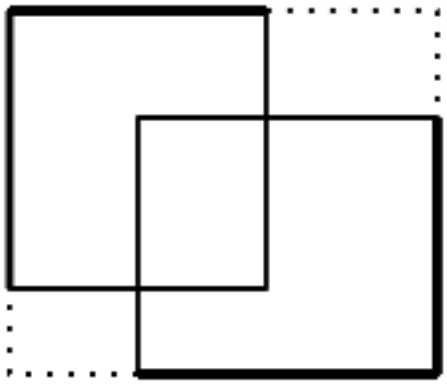

Figura 4.2: Observación parcial de un objeto.

El concepto de observabilidad da buenos resultados y es relativamente sencillo aplicarlo, se define observabilidad según lo expuesto en el apartado 5 de [26]. El resultado se representa por el vector de observabilidad, $m_k$, que tiene la misma dimensión que el vector de medidas del objeto (sus elementos están uno a uno). Cada elemento de $m_k$ tiene sólo dos valores posibles: 1 para observable y 0 para no

observable. El centroide se determina como observable sólo cuando los cuatro bordes de delimitación de su BB $(r_1, c_1, r_2, c_2)$ son observables.

El algoritmo que resuelve el vector $m_k$ está claramente expuesto en la referencia, además de este vector se introduce otro vector de la misma dimensión, vector $n_k$ que vale 1 para puntos que estén dentro del campo de visión de la imagen y 0 para puntos que estén fuera. Se realiza el AND entre estos dos vectores, y es este resultado el que utilizamos como observabilidad.

$$m_k = m_k \,\&\, n_k \tag{4.14}$$

Para un objeto completamente oculto en una agrupación el estado se actualiza utilizando unicamente la predicción, para un objeto parcialmente observable el estado se actualiza combinando la predicción con la observación. Se crea entonces un vector de pseudo-medidas:

$$z_k = Mb_k + (I - M)[\alpha d_k + (1 - \alpha)\hat{z}_k^-] \tag{4.15}$$

Donde M es una matriz diagonal (con $m_k$ como elementos) llamada matriz de observabilidad, $b_k$ es el vector de medidas del objeto del frente, $d_k$ son las medidas deducidas directamente desde el objeto inobservable y $\alpha$ controla los pesos asignados entre medidas observables y no observables, haciendo respetar las hipótesis de alto y ancho constante.

Creamos así para los objetos parcialmente observables un vector de pseudo-medidas condicionado por el modelo y las hipótesis para seguir al objeto durante la oclusión.

Si todas las variables son no observables entonces utilizamos la estimación a priori de la medida y asignamos:

$$d_k = \hat{z}_k^- \tag{4.16}$$

Lo cual sería equivalente a decir que el parámetro de pesos $(\alpha)$ en la ecuación 4.15 es cero.

Con la introducción de la ecuación 4.15 podemos utilizar el concepto de observabilidad para hallar una manera más robusta de seguir a un objeto cuando está en

grupo.

Aquí se presenta una descripción de los pasos del algoritmo de seguimiento.
Para cada cuadro:

- Determinar las medidas del frente $b_k$ y decidir su observabilidad $n_k$.
- Para cada objeto que se está siguiendo determinar su distancia estadística de Mahalanobis con los otros objetos medidos del frente. Si la predicción del centroide cae dentro de un objeto del frente se impone esa distancia a valor nulo. Elegir el mejor apareamiento según la distancia de Mahalanobis (criterio de menor distancia).
- Si varios objetos tienen como mejor distancia a un único objeto medido en el frente nos quedamos con el que tenga menor distancia entre todos (o distancia cero) y a los otros los inhibimos.
- Detectar agrupamiento de objetos, decidir la observabilidad $m_k$ y la medida $z_k$ por cada objeto.
- Para cada objeto con una distancia de Mahalanobis mínima permitida, determinar su estado con la observación si es al menos parcialmente observable, en caso contrario actualizar utilizando la predicción $\hat{x}_k^+ = \hat{x}_k^-$.
- Para cada objeto que no se correspondió con ninguna medida del frente, se incrementa un contador que indica, el numero de cuadros desde la ultima asociación. Si este número esta por encima de un umbral, este objeto será dado de baja.
- Por cada medida que no se corresponde con ningún objeto, se crea un nuevo objeto temporal y se asigna sus características iniciales (velocidad) a cero.
- Determinar la estimación a priori de cada objeto para el cuadro siguiente.

De esta manera podemos realizar un seguimiento a cada objeto detectado en el frente de la imagen. La idea posterior es mediante una homografía pasar la información al plano del modelo de la cancha.

## 4.3. Clasificación de categorías

### 4.3.1. Introducción al problema de la clasificación

El paso siguiente a la detección y seguimiento de cada objeto en la escena, es la estimación de su categoría. Típicamente, se definen 5 categorías:
C1: Equipo A
C2: Equipo B
C3: Juez
C4: Golero A
C5: Golero B

Cada categoría, tiene asociada información de color que la caracteriza. Esta información viene de las camisetas, pantalones y medias de los elementos de cada categoría y es lo que usaremos para reconocer (clasificar).

El objetivo entonces, es dado un objeto detectado, poder inferir una medida de "distancia" a cada una de las categorías. Esta medida de distancia también puede verse como probabilidad de pertenecer a una categoría donde la suma de las probabilidades dan 1.

Para calcular esta distancia, se necesita un "modelo" para cada categoría, contra el cual comparar el objeto de clasificación. Este modelo es obtenido para cada categoría calculando un "histograma promedio". Esto es: trabajando sobre el espacio HSV, se toman 16 pasos en el canal H, 8 pasos en el S y 8 pasos en canal V (histograma de tamaño 16,8,8). El histograma que se asocia con cada categoría, es simplemente el promedio de un conjunto de imágenes de entrenamiento. De esta forma se cuenta con un histograma modelo para cada categoría.

Entonces el proceso es calcular el histograma del objeto que queremos clasificar, para luego determinar la distancia con cada uno de los modelos de cada categoría, utilizando una técnica que describiremos más adelante.

### 4.3.2. Creación de los modelos

Cada secuencia de video, fue dividida lógicamente en dos partes. Una primera que será usada para entrenar al clasificador y la segunda que será utilizada para evaluarlo.

Para cada categoría, se obtuvo un conjunto amplio, (más de 100) objetos, clasificados a mano de la primera parte. Intencionalmente se seleccionaron los mejores ejemplares de cada categoría, es decir que se evitó, falsas detección, jugadores agrupados, imágenes donde se veía parte del jugador, etc. (ver figuras 4.3 y 4.4 ) con el objetivo de lograr extraer las mejores características de color de cada categoría.

Antes de calcular y promediar el histograma para cada imagen del una categoría, se aplica un filtro pasa bajos (filtro gaussiano pasa bajos de tamaño [5x5] y desviación estándar de valor 10) sobre la imagen. Esto hace que la información de color característica de la categoría puede ser mejor capturada en el histograma. Además, para facilitar la normalización de los histogramas, se escalan todas las imágenes a un tamaño promedio.


Figura 4.3: Objetivos no utilizados para calcular los histogramas de cada categoría.

Figura 4.4: Objetivos útiles para el calculo de histogramas.

En la práctica descubrimos que el proceso de entrenamiento de los clasificadores es muy delicado e inestable. Nos referimos a que, un pequeño cambio en alguna variable, puede afectar drásticamente el resultado. El conjunto de variables que pueden afectar al clasificador son:

- El espacio de color: HSV.
- El paso discreto en cada canal [16 x 8 x 8] - resolución del histograma.
- Reducción de las dimensiones del objetivo de entrada (búsqueda de colores útiles).
- Normalización del tamaño del objetivo a 100x40 píxeles.
- Cálculo de los histogramas promedios.

En algunos casos, fue necesario subdividir una categoría en varias y construir más de un modelo. Esto se debe por ejemplo, a que la información de color del frente de una camiseta puede ser muy diferente a la información de color de la parte trasera (ver figura 4.5). Así se construyó un modelo para el frente y otro para la espalda. En general, vamos a tener un modelo por cada agrupación de colores que debamos hacer.

Figura 4.5: Ejemplo de objetivos de las sub-categorías espalda y frente.

Una vez que tenemos los modelos de las categorías, procedemos como sigue:

- Tomar un objeto de entrada, escalar a tamaño promedio, aplicar filtro pasa bajos.
- Calcular su histograma normalizado en el espacio HSV con la resolución de 16x8x8.
- Calcular las distancias a los histogramas modelos.

### 4.3.3. Cálculo de distancias

La técnica utilizada para calcular estas distancias, es la llamada intersección de histogramas [23].

Se define **intersección de histogramas** como:

$$H(I, M) = \frac{\sum_{j=1}^{n} min(I_j, M_j)}{\sum_{j=1}^{n} M_j} \tag{4.17}$$

Donde I es el histograma de la imagen a clasificar y M es el histograma del modelo (ambos de tamaño n) . Como puede verse claramente, se suman los mínimos de cada contenedor para luego normalizar por el número de píxeles en el modelo. Conceptualmente, se pude ver como una medida de cuán "incluida" esta la imagen a clasificar en el modelo.

En la sección de resultados, podrán verse el rendimiento de este clasificador, así como también la comparación con otros métodos que usan por ejemplo la suma de las diferencias en valor absoluto o el coeficiente de Bhattacharyya.

### 4.3.4. Resolución de los histogramas y espacio de color

Optamos por tomar un vector de contenedores = [16 8 8], esto es 16 divisiones en el primer canal, 8 divisiones en el segundo y también 8 en el tercero, pues quisimos darle más importancia (más resolución) al canal angular del *Hue* (matiz) ya que son los píxeles menos afectados por cambios en los niveles de iluminación. Luego se le asignó la mitad (8) contenedores al S (*Saturation*) y el V (*Value*).

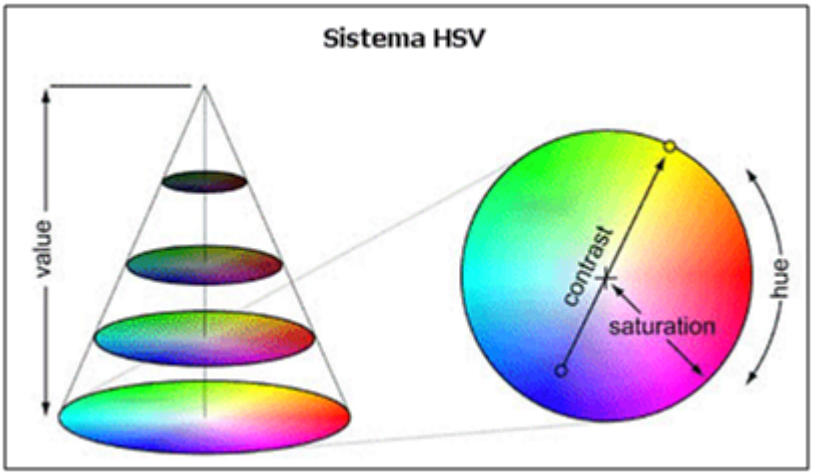


Figura 4.6: Sistema HSV.

Desde el punto de vista computacional, cuanto más grande sea el histograma (más resolución) más tiempo de proceso tomara calcularlo. Entonces se buscó el mínimo tamaño con resolución suficiente para resolver el problema del clasificador.

### 4.3.5. Mapa de probabilidad a priori

Una mejora introducida al clasificación de categoría, vienen dada por la observación de que podemos definir una probabilidad (a priori) de encontrar un determinado elemento en una zona del campo de juego. Por ejemplo, los goleros tendrían una probabilidad muy baja de encontrarse fuera del área y una probabilidad alta de encontrarse dentro de la misma. Al mismo tiempo, para los jugadores de campo asumimos equiprobabilidad en todo el campo de juego.

Se crea así, un mapa de probabilidad, es decir una función que recibe una coordenada 3D (posición) y una categoría para retornar una probabilidad. Entonces, en el proceso de clasificación se agregan los siguientes pasos:

- Se calcula la posición 3D en la cancha del objeto a clasificar (homografía).
- Se combina la distancia a cada modelo con la probabilidad a priori obtenida.

### 4.3.6. Contraste de modelos mediante el coeficiente de Bhattacharyya

En la práctica vimos que la intersección de histogramas o la suma de diferencias en valor absoluto daban buenos resultados, también vimos que cambiando el espacio de colores de trabajo o la resolución en los histogramas además del filtrado gaussiano previo, afectaban el resultado. Pero una gran mejora en los resultados se obtuvieron al utilizar el coeficiente de Bhattacharyya para calcular las distancias:

$$\rho = \sum_{u=1}^{m} \sqrt{p_u . q_u} \tag{4.18}$$

Se puede ver este coeficiente como una aproximación a la medida de superposición, de similitud. Si vale 0 es porque todos los términos de la suma son 0 y esto sería que alguno de los contenedores del modelo o del objeto no contienen píxeles (están vacíos), y el extremo opuesto vale 1 si cada histograma contiene la misma cantidad de píxeles en cada contenedor, es decir si ambos histogramas son idénticos. Entonces el clasificador lo que realizará será el cálculo del coeficiente para cada modelo y asignará la categoría tomando el máximo (el de mayor similitud según Bhattacharyya).

También agregamos la distancia de Hellinger definida como:

$$H(P, Q) = \sqrt{1 - \rho(P, Q)} \tag{4.19}$$

En donde $\rho$ es el coeficiente de Bhattacharyya.

En la sección Resultados, veremos lo obtenido con estas variantes, compararemos con otros métodos. También veremos el concepto de coeficiente de Bhattacharyya normalizado que se expresa como el coeficiente de Bhattacharya dividido la norma del histograma modelo.

$$\rho' = \frac{\rho(P, Q)}{Q.Q^T} \tag{4.20}$$

Vemos que la norma del modelo es el producto escalar representado en el denominador, en donde P es el histograma de la imagen y Q el del modelo.

Opción final:

- El espacio de color: HSV.
- El paso discreto en cada canal [16 x 8 x 8] - resolución del histograma.
- Reducción de las dimensiones del objetivo de entrada (búsqueda de colores útiles).
- Normalización del tamaño del objetivo a 100x40 píxeles.

Utilizando el coeficiente de Bhattacharyya normalizado como método de clasificación y quitando los componentes en los contenedores que tengan mucha información de fondo en la cuenta global del coeficiente.

# Capítulo 5

# Procesamiento multi-cámara, multi-persona

## 5.1. Asociación de objetos a observaciones detectadas

Esta sección describe la segunda etapa del procesamiento multi-cámara. Aquí se integran las observaciones que provienen de las diferentes cámaras. Estas observaciones son integradas en lo que llamaremos una "huella": una representación unificada de la posición y categoría de un jugador en el tiempo.

El mecanismo utilizado es muy similar al de la primera etapa. La diferencia es que para esta segunda, se busca integrar toda la información con la que se cuenta para obtener resultados más precisos.

Los primeros dos pasos de esta sección implementan un proceso de seguimiento utilizando nuevamente el filtro Kalman. El primer paso asocia observaciones obtenidas de la primera etapa a objetivos establecidos y actualiza su estado con la nueva información. El segundo paso es la creación de nuevos objetivos para observaciones que no pudieron ser relacionadas con ningún objetivo establecido.

El resultado es enviado a un proceso de selección en el cual las restricciones de cantidad de elementos para cada categoría son aplicadas, típicamente: 10 jugadores de cancha por cuadro, 1 golero por cuadro y 3 jueces para condicionar la salida

global del sistema. Se asume aquí que este dato es conocido por una fuente externa al sistema. Es decir que un operador debe introducir estos datos al comenzar el procesamiento y luego actualizarlos en caso de que varíen.

## 5.2. Asociación de objetivos y características

Cada jugador es modelado por un objetivo $x_t$ que contiene: un estado $X_t(k)$, la covarianza del estado $P_t(k)$ y una estimación de la categoría $e_t(k)$ en el cuadro $k$. La estimación del estado es actualizada si es posible, por una observación $m_t$ fusionada de al menos una cámara. El $m_t$ comprede: una posición en el plano de la cancha $z_t(k)$, su covarianza $R_t(k)$ y una estimación de categoría $c_t(k)$.

El vector de estados y observación en el plano de la cancha para el filtro de Kalman es:

$$x = \begin{bmatrix} x & y & x' & y' \end{bmatrix}^T \tag{5.1}$$

$$z = \begin{bmatrix} x & y \end{bmatrix}^T \tag{5.2}$$

Tenemos así representada las coordenadas (posición) y velocidad sobre el plano de la cancha. Las ecuaciones de evolución del proceso y de las medidas son:

$$x\left(k+1\right) = A_w x\left(k\right) + w_w\left(k\right) \tag{5.3}$$

$$z\left(k\right) = H_w x\left(k\right) + v_w\left(k\right) \tag{5.4}$$

donde:

$w_w$: es el ruido del proceso.
$v_w$: es el ruido de la medida.

Las matrices de transición de estados y de observación son:

$$A_w = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.5}$$

$$H_w = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{5.6}$$

Los objetivos creados $X_t$ son asociados con las observaciones $(m_j)_i$ de la cámara $i$ y esta asociación es representada en una matriz de asociación $B^{(i)}$. Cada elemento de $(B_{jt})^i$ es 1 para representar una asociación entre el objetivo $t$ y la observación $j$ o 0 de otro modo. La matriz de asociación es creada utilizando la distancia de Mahalanobis $d_{jt}$ entre la observación y la predicción del objetivo.

$$S_{jt}\,(k) = H_w P_t(k|k-1) H_w^T + R_j^{(i)}(k) \tag{5.7}$$

$$d_{jt}^2 = [z_j^{(i)}(k) - H_w \hat{x}_t(k|k-1)]^T S_{jt}(k)^{-1} [z_j^{(i)}(k) - H_w \hat{x}_t(k|k-1)] \tag{5.8}$$

Para una posible asociación, la distancia debe estar por debajo de un umbral. Un umbral muy pequeño puede significar que múltiples huellas son creadas para un mismo jugador y por otro lado, un umbral muy grande puede resultar en objetos diferentes unidos en la misma huella.

Luego se utiliza el algoritmo del vecino más cercano y cada objetivo puede ser asociado con una observación de cada cámara como máximo. Esto es $\sum_j \beta_{jt}^{(i)} \leq 1$. Se asume que un jugador puede dar lugar a una observación como máximo en cada cámara. $\sum_j \beta_{jt}^{(i)} = 0$ corresponde a observaciones no asociadas. Por otro lado, ninguna restricción es impuesta en $\sum_t \beta_{jt}^{(i)}$, es decir el número de objetivos asociados a una observación. Cada observación podría ser asociada a un objetivo, más de uno (para jugadores ocluidos) o ningún objetivo (falsa alarma).

Las observaciones de cada cámara asociadas a cada objetivo se ponderan por las inversas de las incertidumbres o covarianza que tienen asociadas y son integradas en

una observación global de la siguiente manera:

$$R_t = \left[\sum_i \sum_j \beta_{jt}^{(i)} (R_j^{(i)})^{-1}\right]^{-1} \tag{5.9}$$

$$z_t = R_t \left[\sum_i \sum_j \beta_{jt}^{(i)} (R_j^{(i)})^{-1} z_j^{(i)}\right]^{-1} \tag{5.10}$$

$$c_t = \sum_i w_t^{(i)} \sum_j \beta_{jt}^{(i)} c_j^{(i)} \tag{5.11}$$

$$w_t^{(i)} = \frac{\sum_j \beta_{jt}^{(i)} / tr(R_j^{(i)})}{\sum_i \sum_j \beta_{jt}^{(i)} / tr(R_j^{(i)})} \tag{5.12}$$

La ecuación 5.10 indica que la fusión de observaciones es más precisa que las observaciones individuales, mientras que la ecuación 5.11 indica que la observación que tiene más peso es la que tiene menor covarianza.

Cada objetivo asociado a un conjunto de observaciones, es luego actualizado usando las medias integradas:

$$K_t(k) = P_t(k|k-1) H_w^T [H_w P_t(k|k-1) H_w^T + R_t(k)]^{-1} \tag{5.13}$$

$$\hat{x}_t(k|k) = \hat{x}_t(k|k-1) + K_t(k)[z_t(k) - H_w \hat{x}_t(k|k-1)] \tag{5.14}$$

$$P_t(k|k) = [I - K_t(k) H_w] P_t(k|k-1) \tag{5.15}$$

$$\hat{e}_t(k) = (1-\eta)\hat{e}_t(k-1) + \eta c_t(k) \tag{5.16}$$

donde $0 < \eta < 1$.

Si no podemos asociar ninguna observación a un objetivo, entonces su estado es actualizado usando su estimación en el paso anterior solamente. En este caso la covarianza del estado se incrementa linealmente con el tiempo. Una vez que un objetivo no tenga observación asociada por un número determinado de cuadros, la

incertidumbre va a ser mayor a la tolerancia y este objetivo será removido.

## 5.3. Inicialización de los objetivos Tracking

Después de relacionar las observaciones capturadas con los objetivos existentes, es posible que algunas observaciones no correspondan con ningún objetivo. Esas observaciones, cada una obtenida de una cámara diferente, son comparadas entre sí para detectar posibles nuevos objetivos. Si existe una observación $z_{j1}^{i1}$ de la cámara $i_1$, a continuación un nuevo objetivo $X_n$ se establecerá. Todas las matrices de asociación $\beta^{(i)}$ son extendidas en una columna, cada elemento $\beta_{jn}^{i}$ indica la correspondencia entre el nuevo objetivo y la observación $j$ de la cámara $i$. Para la *i-ésima* cámara, la observación $z_{j1}^{i1}$ se asocia automáticamente con el nuevo objetivo:

$$\beta_{jn}^{(i1)} = \begin{cases} 1 & \text{si} \quad j = j_1 \\ 0 & \text{en otro caso} \end{cases} \tag{5.17}$$

Para cada observación sin asociación de las otras cámaras, $z_j^{(i)}$ con covarianza $R_j^{(i)}$, es comparada con la observaciones $z_{j1}^{(i1)}$ con $R_{j1}^{(i1)}$, y se intenta asociar con el nuevo objetivo, si la distancia de Mahalanobis:

$$d_{i_1 i}^2 = [z_j^{(i)}(k) - z_{j1}^{(i1)}(k)]^T [R_j^{(i)}(k) + R_{ji}^{(i1)}(k)]^{-1} [z_j^{(i)}(k) - z_{j1}^{(i1)}(k)] \tag{5.18}$$

se encuentra dentro de un umbral válido. Por lo tanto, el nuevo objetivo se podría asociar con a lo sumo una observación de cada cámara. Todas las observaciones individuales asignadas al nuevo objetivo se integran en una observación general, $Z_n$, $R_n$ y $c_n$, utilizando las ecuaciones representadas en (5.9) - (5.12) y sustituyendo el subíndice $t$ con $n$. El nuevo objetivo es entonces iniciado con la observación integrada.

$$\hat{x}_n(k|k) = [z_n(k)^T \quad 0 \quad 0]^T \tag{5.19}$$

$$P_n(k|k) = \begin{bmatrix} R_n(k) & 0_2 \\ 0_2 & \sigma_v^2 I_2 \end{bmatrix} \tag{5.20}$$

$$\hat{e}_n(k) = c_n(k) \tag{5.21}$$

Una desventaja del esquema de inicialización anterior es que cualquier falsa alarma en una cámara dará lugar a un nuevo objetivo. Una mejor alternativa, la cual fue implementada, es comprobar el número de cámaras que "observan" al nuevo objeto, con el número esperado de cámaras. El número previsto de cámaras se puede obtener al calcular los campos visuales de cada cámara dentro de los cuales se encuentran las observaciones. Un objetivo temporal se establecerá si el número esperado coincide con el obtenido. Los objetivos temporales se convertirán en nuevos objetivos una vez que sean detectados por un número de cuadros determinado.

# Capítulo 6

# Estrategias para la selección y clasificación de las huellas

El sistema de seguimiento multi-cámara que se describe en el capitulo 5 , mantiene un conjunto de datos sobre los objetivos. La estimación del estado de cada objetivo es representada por una función de densidad de probabilidad para su posición (media y covarianza) y una distribución de probabilidad para su categoría (un vector de cinco elementos). En muchas aplicaciones de seguimiento, la salida global del sistema sería directamente estos datos. Pensemos en una sistema de seguimiento de peatones por ejemplo.

Sin embargo, esta salida sin restricciones a menudo da lugar a un número incorrecto de jugadores en cada equipo. Este error puede deberse a varias razones que se manifiesta tanto en el número equivocado de objetivos en el modelo como en la estimación errónea de la categoría. En esta fase es útil volver a examinar los requisitos de la aplicación para intentar mejorar la salida global del sistema.

Esto se lleva a cabo en la Sección 6.1 con la conclusión de que es un requerimiento del sistema que la salida tenga el número correcto de jugadores. En la Sección 6.2 se presenta un marco para satisfacer la restricción de población total, usando la probabilidad máximum a posteriori (MAP) de etiquetado de los objetivos, dado que se tienen sus distribuciones de probabilidad individuales.

En la Sección 6.3 ofrecemos algunas heurísticas simples para el cálculo de estas probabilidades y una técnica sub-óptima (no exhaustiva) para satisfacer la restricción

de población fija.

## 6.1. Requerimientos de la aplicación

Para nuestra aplicación, el número total de personas en el campo de juego es una cantidad conocida. El número por defecto es 25: diez jugadores de campo, un golero por cada equipo y el árbitro con dos asistentes. Si se muestra una tarjeta roja durante el encuentro, este número decrece; otras desviaciones del número por defecto incluyen jugadores que abandonen temporalmente el campo por lesión, personal médico, fanáticos que puedan entrar al campo y sustitutos realizando el calentamiento en el borde del campo de juego. La expulsión de un jugador es un acontecimiento clave y el sistema debe tener un acceso confiable a la cantidad de jugadores que actualmente participan en el juego, se asume que el aviso de este evento estará disponible de alguna fuente externa fiable. Los otros casos son menos importantes y pueden ser manejados con una entrada manual en tiempo real del número actual de personas en el campo de juego.

Podríamos exigir que nuestro sistema limite la salida para que comprenda el número correcto de jugadores de cada categoría. Una restricción similar en el ámbito de reconocimiento óptico de caracteres es el reconocimiento de código postal: un sistema puede ser obligado a dar como salida solamente los códigos postales válidos. Una salida que contiene un número incorrecto de jugadores no se permite, al igual que una secuencia no válida de las letras en un código postal podrá ser rechazada. El modelo puede ser limitado a nivel global como se muestra en la figura 6.1.

## 6.2. Etiquetado MAP

El etiquetado de los objetivos en el modelo consiste en la selección (teniendo en cuenta que las huellas son las auténticas, cuando hay más objetivos en el modelo que jugadores en el campo de juego) y clasificación (teniendo en cuenta qué objetivo pertenece a qué categoría).

En esta sección se describe una metodología para la asignación de la población fija de los jugadores y los árbitros usando etiquetado MAP. Esto se muestra es-

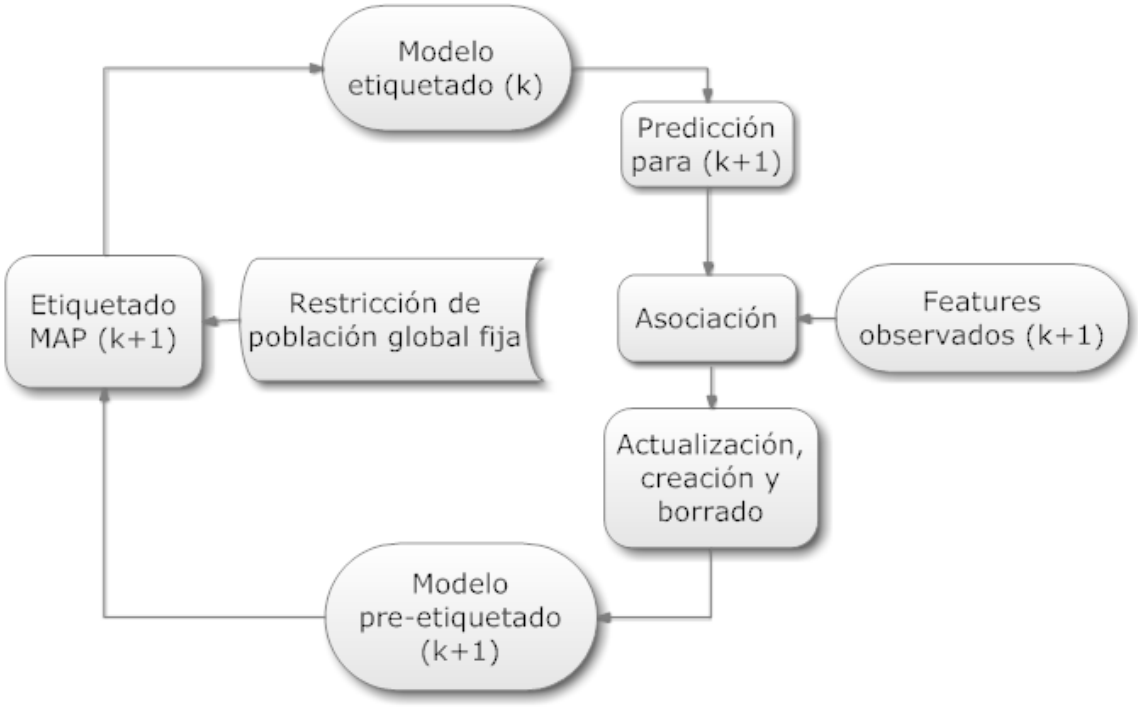


Figura 6.1: Diagrama de flujo para el proceso de seguimiento con la restricción de población global fija. El modelo de etiquetado en el cuadro k se actualiza con las observaciones, para obtener un modelo de pre-etiquetado para el cuadro (k + 1). La restricción de población global fija es re-impuesta en un proceso de etiquetado conjunto de maximun a posteriori (MAP).

quemáticamente en la figura 6.1. Se describe un método para el etiquetado de cada objetivo en el modelo con una de las cinco categorías, sujeto a la restricción de que una etiqueta de la categoría C sólo se puede utilizar Nc veces.

Para seleccionar las huellas, se define un correspondiente vector de etiquetado $q_t$, compuesto por cinco elementos binarios. Fijando $q_c^t(k) = 1$ representa la decisión de asignar una etiqueta de la categoría C la huella t en el cuadro k. Hay dos limitaciones para el etiquetado: debe haber como máximo una etiqueta por huella y la asignación total de etiquetas debe cumplir la restricción de población fija. Teniendo en cuenta la población fija total en cada categoría Nc tenemos

$$\forall c \sum_t q_t^c = N_c \tag{6.1}$$

Del mismo modo, un vector de $p_t(k)$ se utiliza para representar la distribución de probabilidad para cada huella, integrada por los cinco elementos, $p_t^c$, c = 1,..., 5 que representa la probabilidad de que el huella t represente la categoría C. Se supone que estos resultados son exhaustivos (es decir, la huella principal representa exactamente uno o cero jugadores) por lo que estos elementos siempre suman 1.

Suponiendo que las estimaciones de probabilidad son independientes, la probabilidad de que todas las etiquetas estén en lo correcto es el producto de los elementos obtenidos por los vectores de etiquetado:

$$p(\{q\}) = \prod_t p_t q_t \tag{6.2}$$

Este método requiere por tanto un medio para estimar el $\{p_t\}$, y un procedimiento para fijar los elementos de $\{q_t\}$ para obedecer a la restricción de población fija y maximizar el total de una probabilidad a posteriori.

## 6.3. Estimación de la categoría de los objetivos

En primer lugar se describe el método para calcular el $\{p_t(k)\}$. Una probabilidad $\{p'_t(k)\}$ tiene tres elementos, los cuales luego se combinan con uno previo compuesto de $\{p_t(k-1)\}$ y $\{q_t(k-1)\}$.

El primer factor es la probabilidad de que la huella efectivamente represente un jugador, en vez de una falsa alarma de cualquier tipo. El segundo factor es la probabilidad de que represente un jugador de la categoría C (dado que representa un jugador). Por último, incorporamos la probabilidad de que la huella sea la representación principal de este jugador, es decir, no es el caso de que otra huella represente a este jugador mejor. El primer factor incluye tanto la edad de la huella (en cuadros) k1, como el número de cuadros k2 desde la última observación. Estas variables se utilizan en un modelo exponencial de probabilidad de una falsa alarma, con K1 y K2 a escala de k1 y k2 adecuadamente. El segundo factor es simplemente la estimación de la categoría de las observaciones asociadas, $c_t(k)$. Estos se combinan como

$$p'_t(k) = e^{-k2/K2}(1 - e^{-k1/K1})c_t(k) \tag{6.3}$$

El tercer factor es la intención de solucionar situaciones en las que se representa un solo jugador por dos huellas. El problema se aborda aquí contando hasta $N^c(k)$, el número de huellas más probables de representar a la categoría C. Si este es mayor que $N_c$ (la población fija de esa categoría), luego los $N^c(k)$-$N_c$ huellas más cercanas a otro objetivo de la misma categoría se estima que son huellas duplicados, y se consideran como falsas alarmas. Esto es efectivamente un filtro de huellas superfluas a una categoría determinada. Si hay menos objetivos que la población fija de una categoría determinada, entonces la restricción de que haya más de una etiqueta por huella es librada (sin embargo, hemos encontrado en la práctica que se trata de un suceso muy poco frecuente).

El uso de una política de selección de huella cuadro a cuadro, causa discontinuidad temporal en las huellas seleccionadas conduciendo a la situación en la que las huellas maduras y de confianza puedan ser sustituidas por nuevas o anómalas. En particular, las estimaciones de probabilidad dadas las restricciones de población fija se pueden incorporar a través de la inclusión del resultado de la etiqueta anterior. Para incorporar este conocimiento previo se usa un filtro recursivo para actualizar las estimaciones de probabilidad general usando una media conjunta de la forma

$$p_t(k) = \gamma\omega p_t(k-1) + \gamma(1-\omega)q_t(k-1) + (1-\gamma){p'}_t(k) \tag{6.4}$$

donde $\gamma$ y $\omega$ son los pesos que controlan la velocidad de actualización basada en la matriz del cuadro anterior $p_t(k-1)$, la matriz de etiquetado del cuadro anterior $q_t(k-1)$ y la matriz de observación del cuadro actual $p'_t(k)$. Después de que el $p_t(k)$ es evaluado, los elementos de $q_t(k)$ son asignados eligiendo los elementos más probables en primer lugar, hasta que la restricción de la población está satisfecha.

# Capítulo 7

# Resultados

En esta sección se presentarán los resultados obtenidos de las diferentes etapas del proyecto, desde el inicio hasta la solución final presentada. El prototipo fue probado utilizando cuatro juegos de datos: ISSIA, SCEPTRE y datos adquiridos por el grupo Franzini_1 y Franzini_2. Para evaluar el rendimiento utilizamos la herramienta ViPER-PE (ver Apéndice C). Se evaluó el funcionamiento global, así como también el de los módulos individuales.

## 7.1. Esquema de datos XML ViPER GT

Veamos primero la descripción del esquema de datos XML ViPER GT. Este esquema es utilizado para almacenar los resultados tanto del procesamiento por cámara como del procesamiento multi-cámara. La estructura (jerarquía) de los archivos es la siguiente:

1. ViPER
    *a*) config
        - definición del descriptor 1
        - definición del descriptor 2
    *b*) data
        - archivo1

        - descriptor
    - archivo2
        - descriptor

Un descriptor es :

- Un registro que describe un elemento de un video.
- Un objeto que cumple con un esquema de datos definido por el usuario.
- Una composición de una colección de atributos con nombre y tipo.
- Tiene asociado un identificador único y una duración en el que es válido.
- Es uno de estos tres tipos:
    - File: Se refiere a información que habla del video como un todo o información acerca del video como formato y tasa de cuadros por segundo.
    - Content: Instancias de este tipo sólo puede ocurrir de uno en uno y no puede cambiar a lo largo de su vida.
    - Object: Refieren a un objeto que podría tener muchas instancias en un instante de tiempo dado, y estas instancias pueden cambiar a lo largo del tiempo.

Un Atributo ("Attribute") es:

- Cada descriptor tiene varios atributos.
- Un atributo puede ser uno de varios tipos:
    - svalue: cadenas de caracteres.
    - lvalue: un enumerado (una valor de una lista de palabras definida por el usuario).
    - dvalue: atributo de tipo decimal.
    - bbox, polygon, etc: un valor de una lista de formas

```
<data>
    <sourcefile filename="cam2.avi">
        <file id="0" name="Information">
            <attribute name="SOURCEDIR">
                <data:svalue value="C:\Users\pablo\Desktop\Italianos"/>
            </attribute>
            <attribute name="SOURCEFILES">
                <data:svalue value="cam2.avi"/>
            </attribute>
            <attribute name="SOURCETYPE">
                <data:lvalue value="FRAMES"/>
            </attribute>
            <attribute name="NUMFRAMES">
                <data:dvalue value="3001"/>
            </attribute>
            <attribute name="H_FRAME_SIZE">
                <data:dvalue value="720"/>
            </attribute>
            <attribute name="V_FRAME_SIZE">
                <data:dvalue value="576"/>
            </attribute>
        </file>
```

Figura 7.1: Archivo XML, información general.

- reference: referencia a otro descriptor

En la figura 7.1 vemos un ejemplo del descriptor de tipo “File” con los atributos del video, como tamaño del cuadro, largo en cuadros y nombre del archivo .avi

Luego comienzan los descriptores de tipo “Object” . En la figura figura 7.2 vemos un ejemplo.

```
</file>
<object framespan="2254:2254" id="1" name="Person">
    <attribute name="LOCATION">
        <data:bbox framespan="2254:2254" height="45"
            width="15" x="30" y="71"/>
    </attribute>
    <attribute name="INDEXCATEG">
        <data:dvalue framespan="2254:2254" value="2"/>
    </attribute>
    <attribute name="CATEG">
        <data:svalue framespan="2254:2254" value="2.2376_3.1278_1.8666_2.3941_2.3399"/>
    </attribute>
    <attribute name="z">
        <data:svalue framespan="2254:2254" value="486.89_368.1707"/>
    </attribute>
    <attribute name="R">
        <data:svalue framespan="2254:2254" value="0.23766_-0.37389_-0.37389_2.3299"/>
    </attribute>
</object>
```

Figura 7.2: Archivo XML, información por objetos.

Estos descriptores pretenden registrar la ocurrencia de un objeto que se detectó y clasificó en un cuadro del video. Un descriptor de este tipo está compuesto de:

1. id: identificador del objeto
2. framespan: validez en cuadros de este descriptor.
3. LOCATION: es el rectángulo en coordenadas de la imagen que indica la posición del objeto.
4. INDEXCATEG: un decimal que indica el índice de la categoría establecida para este objeto.
5. CATEG: es el vector de 5 elementos de probabilidades de categorías calculado para este objeto.
6. z: Coordenadas de la holografía al plano de la cancha.
7. R: Error relativo a la estimación de la posición para este objeto.

Estos descriptores fueron creados para describir el resultado del procesamiento por cámara y también son utilizados para describir el resultado del procesamiento multi-cámara en la segunda etapa.

***Precision* & *Recall***

Para presentar los resultados obtenidos de las diferentes pruebas nos basamos en las medidas de rendimiento de *Precision & Recall*.

| | resultado correcto / clasificación | | |
|---|---|---|---|
| | | **E1** | **E2** |
| **resultados obtenidos / clasificación** | **E1** | **tp** (verdaderos positivos) | **fp** (falsos positivos) |
| | **E2** | **fn** (falsos negativos) | **tn** (verdaderos negativos) |

Figura 7.3: *Precision & Recall*

*Precision & Recall* se define entonces como: (ver [17])

Precision $= \frac{t_p}{t_p+f_p}$

Recall $= \frac{t_p}{t_p+f_n}$

El valor de *Precision* de una clase es el número de verdaderos positivos (es decir, el número de elementos correctamente etiquetados como pertenecientes a la clase positiva) dividido por el número total de elementos etiquetados como pertenecientes a la clase positiva (es decir, la suma de verdaderos positivos y falsos positivos, que son los elementos incorrectamente etiquetados como pertenecientes a la clase).

*Recall* en este contexto se define como el número de verdaderos positivos dividido por el número total de elementos que en realidad pertenecen a la clase positiva es decir, la suma de verdaderos positivos y falsos negativos, que son elementos que no estaban etiquetados como pertenecientes a la clase positiva pero sí deberían haber estado.

A manera de ejemplo: tenemos un jugador que pertenece a la categoría de azules (en el GT), si el clasificador lo clasifica como azul, este objetivo es un verdadero positivo (correctamente etiquetado), si el clasificador lo clasifica como blanco, entonces es un falso negativo, mientras que los blancos que el clasificador clasifique como azules se consideran como falsos positivos.

Vamos a comparar los valores del GT contra los valores de la salida de nuestro algoritmo. Se calculará *Precision & Recall* como una medida del rendimiento del proceso.

## 7.2. Rendimiento

El GT realizado tiene que ser comparado con los resultados obtenidos, se requiere de una definición clara de los indicadores. Luego, los resultados de la evaluación se combinan para cada cuadro de video para su presentación al usuario. Por lo tanto, podemos distinguir cuatro principales bloques de evaluación, como se muestra en la figura 7.4; la creación de los datos del GT, cómo se dispone de los datos de evaluación conjuntos, las diferentes métricas de rendimiento y por último la presentación de los resultados de la evaluación.

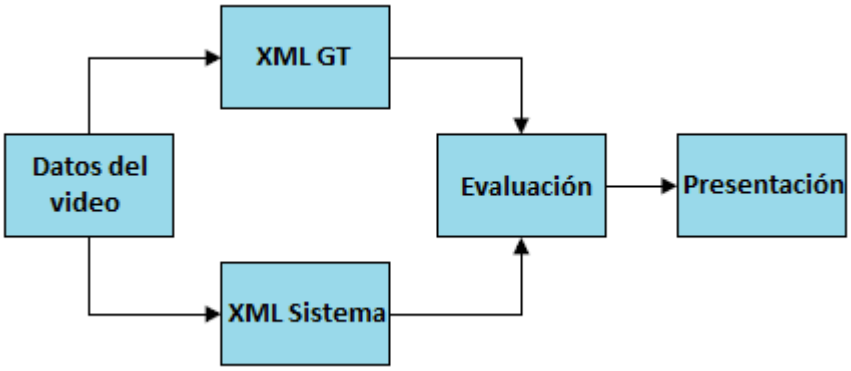


Figura 7.4: Diagrama de bloques de un sistema de evaluación de performance.

Se pueden definir múltiples niveles semánticos para la evaluación:

1. A nivel de píxeles, la segmentación del cuadro de video, los objetos en movimiento y el fondo estático.

2. Basada en objetos de evaluación por cuadro.

3. Basada en objetos de evaluación en un tiempo de vida de los objetos.

4. Características de objetos, como tipo de objeto, velocidad, tamaño (en píxeles, en metros).

5. Comportamiento de los objetos, por ejemplo, agrupaciones, objetos que salen de imagen o que entran, etc.

Nosotros utilizamos objetos de evaluación en un tiempo de vida de los objetos y también estudiamos las características de posición y clasificación de categoría. Armamos los archivos de GT tomando estas consideraciones y utilizamos el ViPER-PE para medir *Precision & Recall* (ver Apéndice C) de cada prueba particular. Realizamos varias pruebas puntuales para cada juego de datos, pruebas de clasificación de categorías, pruebas de solamente detección de posición, pruebas en conjunto, en pocos cuadros, en muchos cuadros, en situaciones particularmente comprometidas, entre otras.

## 7.3. Resultados del procesamiento por cámara

En primer lugar se van a presentar los resultados obtenidos de la clasificación de categorías mostrando algunos casos particulares. Luego se mostrarán resultados generales (teniendo en cuenta la categoría y la ubicación de los jugadores) para cada conjunto de datos.

### 7.3.1. Resultados clasificación de categorías

#### Estudio videos ISSIA

Vamos a estudiar en la cámara 5 un caso en donde se observa una jugada de ataque sobre el área, una aglomeración de jugadores. Primero analizaremos 50 cuadros y luego un total de 100 cuadros dentro del desarrollo de la jugada en el área. Vamos a medir en esa situación desfavorable para el clasificador las distintas técnicas.

La idea será comparar contra el GT en esta situación puntual de una aglomeración en el área (ver figura 7.5) los porcentajes de *Precision & Recall* mediante el uso del ViPER-PE.

Tomamos 50 cuadros (en esta oportunidad se testearon 16 objetos) y calcularemos *Precision & Recall* para poder medir los aciertos de detección y clasificación de categorías.

Hay que destacar que la situación que elegimos en esta primera prueba presenta los dos problemas detectados, falsa asignación de la categoría del golero a los jugadores de campo y falsa asignación de categoría por objetos unidos. Los resultados se

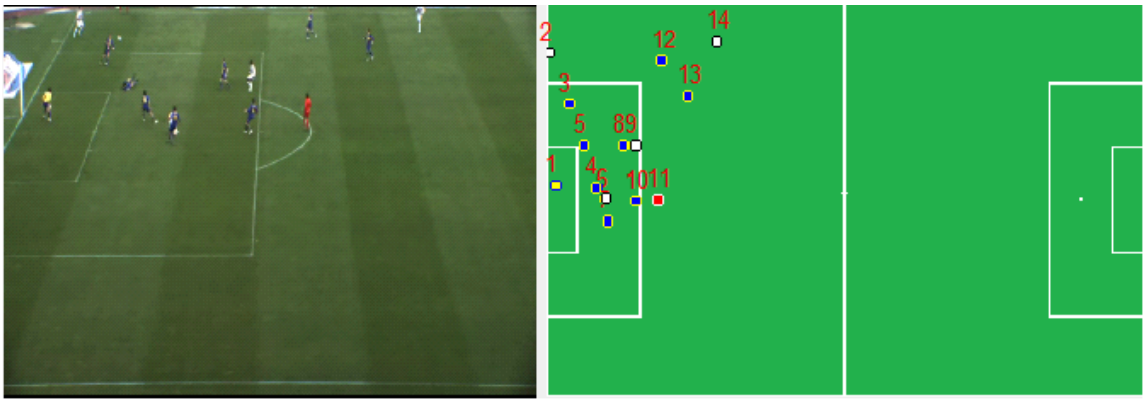


Figura 7.5: Jugada de ataque, aglomeración en el área.

muestran en la figura 7.6.

```
*********************************************************
*                    SUMMARY RESULTS                    *
*********************************************************

For OBJECT Person: Precision is 68 %  (11/16)
For OBJECT Person: Recall is 68 %  (11/16)

For TOTAL: Precision is 68 %  (11/16)
For TOTAL: Recall is 68 %  (11/16)
```

Figura 7.6: Resultado *Precision & Recall* cámara 5 ISSIA

Tomando en cuenta lo visto en la figura 7.5 vemos que los objetos 3, 4, 5, 7, 8 y 10 que pertenecen al equipo azul y están dentro del área durante esta secuencia de cuadros podrían cambiar el color al color del golero. Esta confusión del clasificador se produciría debido a que dentro del área la probabilidad a priori para la categoría del golero no es nula.

En el archivo XML generado por el sistema se pueden ver los objetos candidatos. En él se muestra el "id" del objeto y los cuadros en los que él mismo aparece. Ver figura 7.7.

Dentro de cada objeto se puede distinguir los diferentes atributos como se muestra en la figura 7.8.

Si expandimos uno de los campos respectivos de algún atributo tendremos los

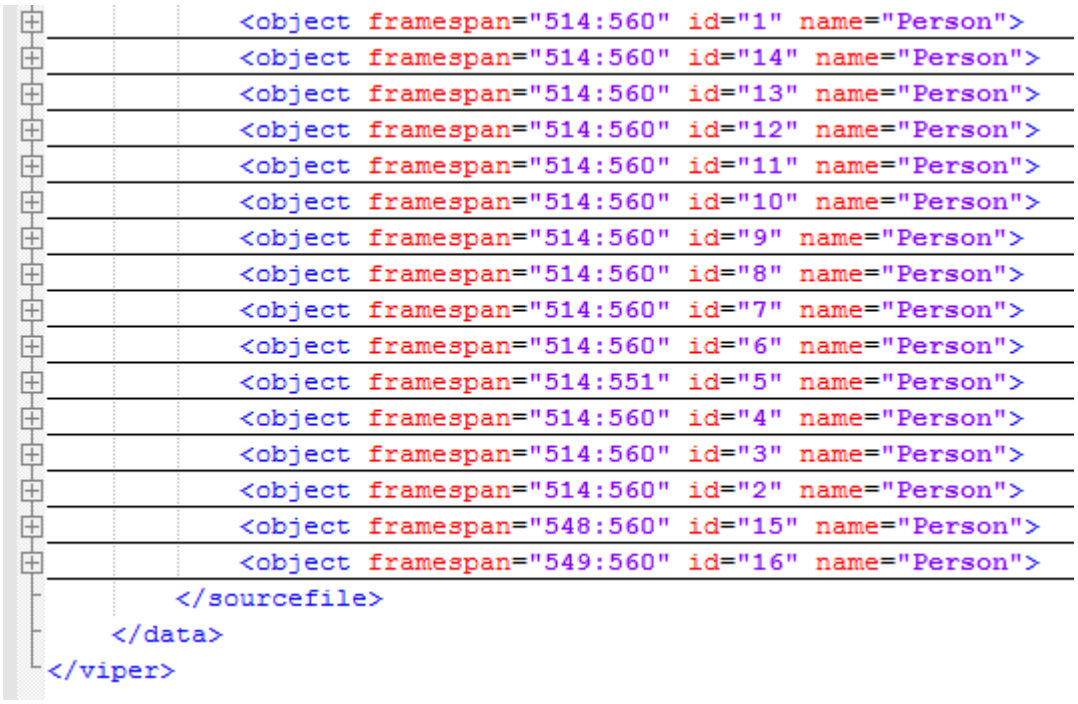


Figura 7.7: Archivo XML, orden por objetos.

valores asignados por el sistema.

En la figura 7.9 el objeto con id=10 aparece entre el cuadro 514 al 560 y por debajo se detalla como varía el índice de su categoría. Se despliega el valor del mismo para todos los cuadros de testeo. Como la métrica utilizada para el atributo INDEXCATEG es la métrica E-equality, el resultado es 0 o falso si es distinta la comparación de índices y 1 o verdadero si es idéntica a la hora de contar los verdaderos positivos (elementos bien etiquetados).

La clasificación se produce utilizando el coeficiente de Bhattacharyya como método de asignación de categorías y el filtrado de aquellos contenedores que contengan mucha información del fondo de la cancha.

Restando aquellos contenedores que contengan el 5 % (0.05) del total de píxeles en el histograma que modela el fondo, se obtuvieron los datos expuestos en el cuadro 7.1.

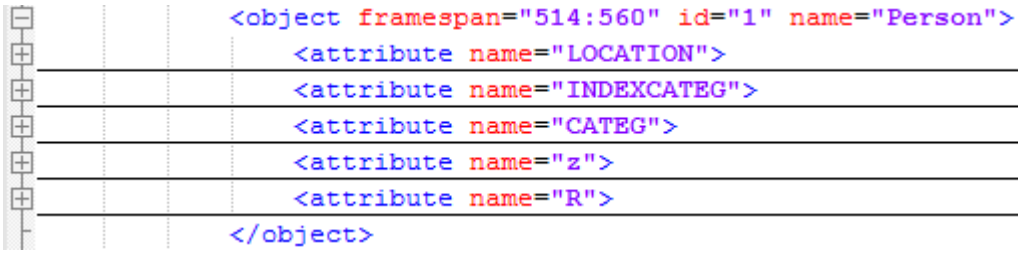


Figura 7.8: Despliegue de un objeto.

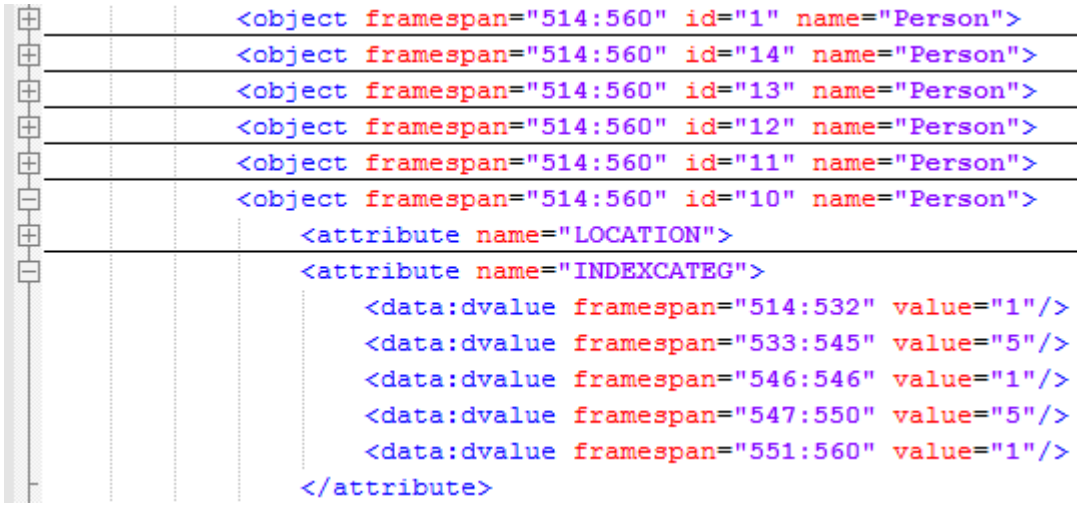


Figura 7.9: Despliegue del atributo INDEXCATEG para un objeto.

| | |
|---|---|
| **Intersección de Histogramas** | 68 % |
| **Coeficiente de Bhattacharyya** | 87 % |

Cuadro 7.1: Cuadro comparativo entre métodos de asignación.

Hay muchas maneras de probar rendimientos de este tipo, lo primordial es elegir un método razonable de comparación (intersección de histogramas, suma de las diferencias en valor absoluto, coeficiente de Bhattacharyya, etc.).

**Conjunto de Testeo: Cámara 5 (ISSIA), cuadros [515:615]**

Durante las pruebas de clasificación de categorías para 100 cuadros de la cámara 5, se encuentra una jugada de aglomeración en el área, el cuadro 7.2 muestra para

estos cuadros de testeo los resultados de *Precision & Recall*.

| **Video5 (ISSIA), cuadros pertenecientes a [515:615]** | ***Precision*** | ***Recall*** |
|---|---|---|
| **Método de Intersección de Histogramas** | 82 % (14/17) | 82 % (14/17) |
| **Método de la suma de las diferencias en valor absoluto** | 82 % (14/17) | 82 % (14/17) |
| **Método del coeficiente de Bhattacharyya** | 76 % (13/17) | 76 % (13/17) |
| **Distancia de Hellinger** | 76 % (13/17) | 76 % (13/17) |
| **Método del coeficiente de Bhattacharyya normalizado** | 88 % (15/17) | 88 % (15/17) |

Cuadro 7.2: Pruebas de clasificación en 100 cuadros de la cámara 5.

**Método de Intersección de Histogramas**
Utilizando una resolución de [16 8 8] en HSV para los objetivos de entrada calculamos el histograma y lo comparamos mediante la ecuación 4.17 por la intersección de los histogramas normalizados.

**Método del coeficiente de Bhattacharyya normalizado**
Con esta configuración buscando los máximos llegamos a un total de 15 objetos bien etiquetados sobre 17 objetos. Es el resultado más alto entre todos los vistos anteriormente, se aplica entre los histogramas la ecuación 4.20.

### 7.3.2. Resultados ISSIA

**PRUEBA 1000 cuadros - cámara 1 ISSIA (cámara que captura una da las áreas)**

Para información sobre las métricas utilizadas ver figura C.3, Apéndice C.

| **Métrica** | ***Precision*** | ***Recall*** |
|---|---|---|
| **LOCATION:Equality [E]** | 64 % (9/14) | 64 % (9/14) |
| **LOCATION:Overlap 0.5** | 78 % (11/14) | 78 % (11/14) |
| **LOCATION:Overlap 0.9** | 78 % (11/14) | 78 % (11/14) |
| **LOCATION:Dice 0.9** | 71 % (10/14) | 71 % (10/14) |

Cuadro 7.3: Cuadro comparativo entre las diferentes métricas.

**PRUEBA 500 cuadros - cámara 4 ISSIA (cámara que captura el centro de la cancha)**

Repitiendo las pruebas realizadas anteriormente para 1000 cuadros de la cámara 1 ahora lo aplicamos a 500 cuadros de la cámara 4.

| **Métrica** | ***Precision*** | ***Recall*** |
|---|---|---|
| **LOCATION:Equality [E]** | 45 % (14/31) | 34 % (14/41) |
| **LOCATION:Overlap 0.5** | 74 % (23/31) | 56 % (23/41) |
| **LOCATION:Overlap 0.9** | 74 % (23/31) | 56 % (23/41) |
| **LOCATION:Dice 0.9** | 74 % (23/31) | 56 % (23/41) |
| **LOCATION:[dice 0.9] INDEXCATEG:[E]** | 54 % (17/31) | 41 % (17/41) |

Cuadro 7.4: Tabla comparativa ejemplo cámara 4.

**PRUEBA 1000 cuadros (1:1000) - cámara 3 ISSIA**

En esta prueba tomamos una cámara del centro de la cancha y tomamos 1000 cuadros como test, hubo oclusiones varias durante el transcurso del video.

| **Métrica** | ***Precision*** | ***Recall*** |
|---|---|---|
| **LOCATION:[overlap 0.5] INDEXCATEG:[E]** | 69 % (48/69) | 63 % (48/76) |
| **LOCATION:[overlap 0.5]** | 88 % (61/69) | 50 % (38/76) |
| **LOCATION:[dice 0.9] INDEXCATEG:[E]** | 66 % (46/69) | 60 % (46/76) |
| **LOCATION:[dice 0.9]** | 86 % (60/69) | 48 % (37/76) |
| **LOCATION:[E] INDEXCATEG:[E]** | 46 % (32/69) | 42 % (32/76) |

Cuadro 7.5: Tabla comparativa ejemplo cámara 3.

Vemos en la tabla 7.5 que la medida para la localización en cuanto al valor de *Precision* es de 60/69 es decir un 86 %, esto se podría interpretar como que de los 69 objetos a testear en el archivo XML de salida del sistema, se pudieron encontrar correspondientes a 60 de ellos con objetos en el archivo de GT.

Mientras que *Recall* es de 37/76, es decir que de los 76 objetos test registrados como verdaderos en el archivo de GT solamente se asignaron 37 de ellos (es decir para 37 de esos objetos test se encontró correspondiente en el archivo XML de salida del sistema).

Como vemos hay una diferencia de objetos entre el conjunto de objetos test descripto en el archivo GT que son 76 elementos y el conjunto correspondiente al archivo XML de salida del sistema que son 69. Estos 9 elementos son elementos que no se detectaron en el seguimiento realizado por el sistema y que sí se enmarcaron a la hora de editar el GT mediante el ViPER-GT (ver apéndice C).

### 7.3.3. Resultados SCEPTRE

**PRUEBA 1000 cuadros (1220:2200) - cámara 1 SCEPTRE**

Volvemos a considerar los conjuntos de 1000 cuadros como prueba ahora en la cámara 1 del juego de datos SCEPTRE. Vemos aquí que hay algunas cosas registradas que pueden ser de nuestro interés estudiarlas, por ejemplo un golero fuera del área, un movimiento mecánico no deseado, los cuales se traducen en falsas detecciones.

| **Métrica** | ***Precision*** | ***Recall*** |
|---|---|---|
| **LOCATION:[overlap 0.5] INDEXCATEG:[E]** | 58 % (23/39) | 95 % (23/24) |
| **LOCATION:[overlap 0.5]** | 61 % (24/39) | 100 % (24/24) |
| **LOCATION:[dice 0.9] INDEXCATEG:[E]** | 58 % (23/39) | 95 % (23/24) |
| **LOCATION:[dice 0.9]** | 61 % (24/39) | 100 % (24/24) |
| **LOCATION:[E] INDEXCATEG:[E]** | 38 % (15/39) | 62 % (15/24) |

Cuadro 7.6: Tabla comparativa ejemplo cámara 1 SCEPTRE.

**PRUEBA 1000 cuadros (1:1000) - cámara 8 SCEPTRE**

En esta prueba pudimos observar cómo aparecen, por pequeños instantes, movimientos ínfimos que hacen que se detecten las líneas de juego como frente, estas son falsas detecciones. A su vez es importante destacar que el golero se encuentra siempre dentro del área.

El resultado obtenido es el siguiente:

For OBJECT Person: Precision is 66 % (4/6)
For OBJECT Person: Recall is 100 % (4/4)

*Recall* es óptimo, es decir, para los 4 objetos test descriptos en el archivo GT, se les encontró asignación con objetivos test candidatos descriptos en el XML de salida del sistema. No quedaron objetos test del archivo GT sin ser correspondidos.

Por otro lado vemos que existe una diferencia de 2 en *Precision*, es decir de los 6 objetos en el GT de salida del sistema fueron correspondidos a objetos del archivo GT 4 de ellos, entonces los otros dos evidentemente fueron falsas detecciones y así lo muestra el archivo de salida de la prueba (ver figura 7.10).

### 7.3.4. Resultados Franzini_1

**PRUEBA 400 cuadros - cámara 2 Franzini_1**

Jugada en donde entran ocluidos los jugadores en la escena, vamos a estudiar cómo repercute esto en el valor de *Recall*.

| **Métrica** | ***Precision*** | ***Recall*** |
|---|---|---|
| **LOCATION:[overlap 0.5] INDEXCATEG:[E]** | 55 % (5/9) | 31 % (5/16) |
| **LOCATION:[overlap 0.5]** | 88 % (8/9) | 43 % (7/16) |
| **LOCATION:[dice 0.9] INDEXCATEG:[E]** | 55 % (5/9) | 31 % (5/16) |
| **LOCATION:[dice 0.9]** | 77 % (7/9) | 37 % (6/16) |

Cuadro 7.7: Tabla comparativa ejemplo cámara 2 Franzini_1.

Vemos en el valor de *Recall* que, de los 16 objetos test del GT, tan solo 6 pudieron

```
*****************************************************
*                      LEVEL 2                      *
*                  FALSE DETECTIONS                 *
*****************************************************
OBJECT Person 4 153:167
          LOCATION : 3*("40 127 63 38"), "54 136 74 43", '
        INDEXCATEG : 15*("3")
             CATEG : "0.48644_0.56493_0.63276_NaN_NaN", "(
                 z : 3*("63.3893_393.869"), "78.0675_395.(
                 R : 3*("0.82141_0.37211_0.37211_0.27171"]

OBJECT Person 5 182:187
          LOCATION : "51 134 86 50", "47 132 86 50", "44 :
        INDEXCATEG : 6*("3")
             CATEG : "0.5181_0.57413_0.5828_NaN_NaN", "0.!
                 z : "82.4522_396.3777", "79.7289_395.935:
                 R : "0.67048_0.31335_0.31335_0.24235", "(
```

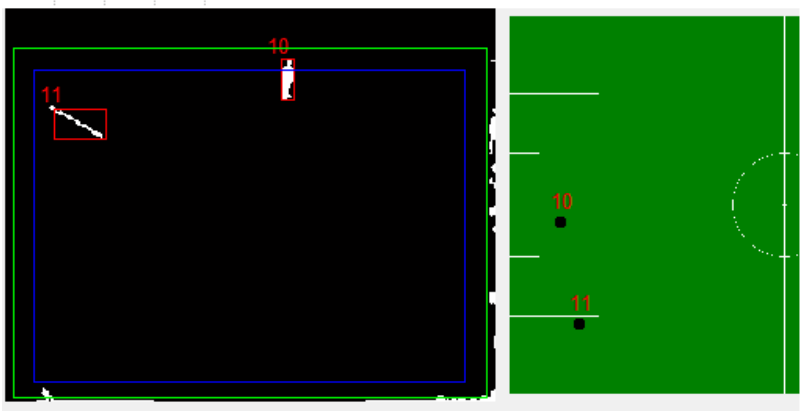


Figura 7.10: Falsas detecciones por movimiento en las cámaras, ocurren entre el cuadro 153 y cuadro 187 de la cámara 8 de SCEPTRE.

ser asignados a algún objeto del XML salida del sistema.

Mientras tanto 7 de los 9 objetos test del XML salida del sistema son encontrados dentro de los objetos test del GT, por lo cual el valor *Precision* es del 77 % (aplica en el caso de medir solo la localización con una métrica de DICE 0.9 , ver tabla 7.7).

Agregando la asignación de categoría con la métrica Equality:

For OBJECT Person: Precision is 55 % (5/9)
For OBJECT Person: Recall is 31 % (5/16)

Vemos la baja en el valor de *Precision*, hay 4 objetos que aparecen como falsa

detección de LEVEL3 (ver Apéndice C), esto es porque no se mantuvo a lo largo del proceso de prueba (400 cuadros elegidos) la identidad de la clasificación de categoría correcta en el objeto test. Ocurre la falsa detección por la norma Equality ya que marca como objeto perdido un objeto que no tenga el mismo INDEXCATEG pero que sí pueda tener, por ejemplo, el mismo LOCATION.

**PRUEBA 500 cuadros (1:500) - cámara 3 Franzini_1**

| **Métrica** | ***Precision*** | ***Recall*** |
|---|---|---|
| **LOCATION:[overlap 0.5] INDEXCATEG:[E]** | 55 % (33/60) | 61 % (33/54) |
| **LOCATION:[overlap 0.5]** | 75 % (45/60) | 75 % (41/54) |
| **LOCATION:[dice 0.9] INDEXCATEG:[E]** | 55 % (5/9) | 31 % (5/16) |
| **LOCATION:[dice 0.9]** | 55 % (33/60) | 61 % (33/54) |

Cuadro 7.8: Tabla comparativa ejemplo cámara 3 Franzini_1.

## 7.4. Resultados procesamiento multi-cámara

En la segunda etapa del proceso, se probó con varios juegos de datos como en la primera. En esta etapa, los principales factores que influyen en el rendimiento son: la posición de las cámaras en el campo (altura, campo visual) y el rendimiento del procesamiento de la primera etapa. Con rendimiento nos referimos a cuán próximo al GT se encuentra la salida del proceso. La distribución de las cámaras, influye en cómo y cuándo se producen las oclusiones. En esta sección se podrá observar como la suma de las observaciones individuales de las cámaras, incrementan la precisión de la salida global del sistema.

En la figura 7.11 puede verse cómo se incrementa la precisión general del sistema, sobre la posición y categoría de los objetos, en relación a la precisión de cada cámara. También la figura 7.11 muestra la distribución de las cámaras del juego de datos de ISSIA. Cada jugador se encuentra a una distancia distinta, con respecto al campo visual de cada cámara. Generalmente una mayor distancia, representa una mayor incertidumbre en el plano de cancha. Esta incertidumbre es representada con la co-

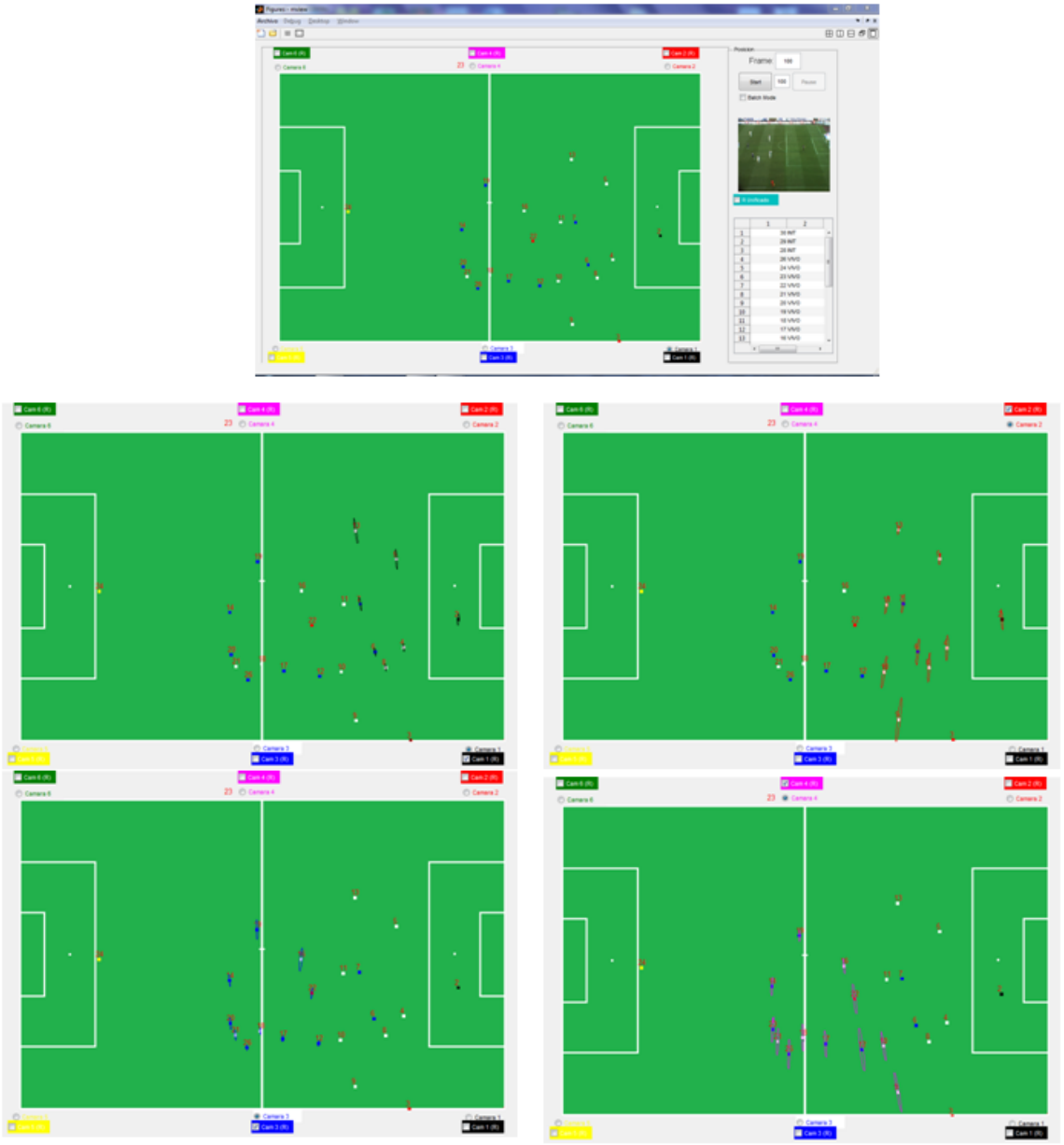

Figura 7.11: Incremento de precisión en la visión unificada. Puede observarse el error que cada cámara tiene sobre la posición de los objetos que detecta. Se observa cómo se incrementa la precisión global del sistema combinando los datos de cada cámara.

varianza de la medida R (graficada como una elipse sobre el objeto). La combinación de medidas utilizadas en este proyecto, integra las medidas utilizando como "pesos" la inversa de la covarianza de cada medida. De esta forma, la medida fusionada y por lo tanto la posición del jugador, es automáticamente condicionada a la medida más precisa de la cámara más cercana.

Para evaluar el rendimiento del sistema, contamos el número de objetos detectados en cada categoría, cuadro a cuadro. Las razones para utilizar este mecanismo se deben a que: 1) cuando un objetivo no es detectado, generalmente se debe a que es ocluido por otro objetivo. Luego de un tiempo, este objeto es terminado y el número total de objetivos disminuye. 2) A menudo un error en el seguimiento causa que un nuevo objeto sea creado, entonces el número de objetivos general se incrementa. Esto nos lleva a pensar que el número de objetos detectados es un buen indicador del rendimiento del sistema.

Vamos a presentar aquí los resultados para el conjunto de datos ISSIA en 3000 cuadros con 6 cámaras fijas. Se cuenta el número de elementos por categoría detectados cuadro a cuadro. Luego, vamos a calcular la media y desviación estándar de esta muestra en cada categoría. Como conocemos los valores "verdaderos", este dato nos va a dar una medida de cuán próximo estuvo el sistema a la salida óptima.

| **Métrica** | **Cuadro 1** | **Cuadro 2** | **Árbitros** | **Golero 1** | **Golero 2** |
|---|---|---|---|---|---|
| **Media** | 12.85 | 12.82 | 2.25 | 0.90 | 1.14 |
| **Desviación estándar** | 2.51 | 2.01 | 1.13 | 0.33 | 0.39 |

Cuadro 7.9: Media y desviación estándar por categoría en el juego de datos ISSIA.

No sorprende que el número medio de jugadores detectados por categoría se encuentre un poco por encima del número esperado para los jugadores de campo, ya que el sistema tiende a "continuar" con aquellos objetos que no tienen medidas asociadas por algún tiempo. Con respecto al árbitro y sus asistentes, hay que notar que muchas veces el campo visual de las cámaras, dejan afuera a los árbitros de línea, lo cual afecta el número de objetos de esta categoría detectados.

# Capítulo 8

# Conclusiones

Se planteó el problema a resolver, sus requerimientos y restricciones. Se relevó el estado del arte en el área del problema, utilizando los trabajos académicos más relevantes. Al mismo tiempo se estudiaron las propuestas comerciales actuales.

En el transcurso de este proyecto se diseñó e implementó en el lenguaje Matlab, un prototipo de software que resuelve los aspectos principales del problema planteado. Esto es, detectar seguir y clasificar los 22 jugadores y los árbitros durante una secuencia de video.

Dada la complejidad del problema, el prototipo fue creado con el objetivo de probar la existencia de una solución y se dejaron de lado aspectos que obviamente tendrían que considerarse para una eventual solución comercial como:

**Tiempo de procesamiento:** se busco diseñar una arquitectura modular que permitiera fácilmente intercambiar y evaluar distintos tipos de algoritmos en las distinta etapas. Este fue el objetivo que nos planteamos porque creemos que el problema del tiempo de procesamiento constituye un proyecto aparte en sí mismo, que involucraría usar lenguajes de programación de bajo nivel como C o C++, ademas involucraría también el uso de arquitecturas de procesamiento paralelo, estructuras de datos y de comunicación entre procesos. Evidentemente esto tendría que encararse una segunda etapa.

**Volumen da datos:** El prototipo implementado maneja sin problemas (en una PC de escritorio estándar 4GB RAM, Micro: Intel Core I5) hasta un volumen de datos que representa aproximadamente 5 minutos de partido con 6 cámaras fijas.

Esta limitación se debe al tipo de estructura de datos que manejamos (archivos XML) y principalmente a los requerimientos de memoria que se necesitan. Nuevamente, en este punto debe entenderse que el objetivo del prototipo no es procesar un partido completo, ese sería el objetivo de un producto comercial. Después de todo, desde el punto de vista técnico el problema se encuentra resuelto, lo que sucede en 85 minutos de partido tiene la misma complejidad que lo que sucede en 5 minutos. La utilización de archivos XML que cumplan con el esquema ViPER, nos permitió evaluar los resultados utilizando ViPER PE o ViPER GT para luego visualizarlos y editarlos, esto además nos permitió fácilmente trasladar la información (simplemente copiar) y por supuesto evitó tener que interactuar con bases de datos u otros dispositivos de almacenamiento masivo.

El prototipo fue probado con juegos de datos propios y externos. El capítulo 7 Resultados, describe cuantitativamente el rendimiento global. Cabe destacar que, en nuestra opinión, los resultados alcanzados son muy alentadores. El número medio de objetos detectados en la secuencia es en general +/- 1 diferente al real y aún más importante, esta diferencia puede ser explicada.

Estos resultados no fueron alcanzados de un día al otro. Fueron alcanzados en un proceso que abarcó prácticamente todo el proyecto, con el paso de los meses progresivamente se adquirió experiencia y conocimiento en las técnicas utilizadas, hasta llegar a dominarlas y poder integrar las soluciones parciales a la solución global.

El prototipo construido nos permitió probar que una solución al problema es técnicamente factible y realizable. Permitió identificar puntos críticos del sistema, así como puntos extensibles o perfectibles. Permitió validar nuestra solución, identificar cuáles fueron buenas y malas decisiones. Permitió enfrentarnos a problemas de campo reales, prácticos. Desde el manejo básico de una cámara de video HD, hasta la estabilización de la posición de un trípode. Desde este punto de vista, estamos completamente satisfechos con el resultado y creemos que el conocimiento adquirido en el área del problema, los obstáculos y problemas que tuvimos que sortear para llegar a este punto, son en definitiva, el real valor del proyecto.

# Capítulo 9

# Trabajo a futuro

Creemos que el prototipo realizado cumple con los principales objetivos que fueron planteados al inicio. Este nos permitió identificar puntos fuertes y débiles de nuestra solución. Con respecto a los puntos débiles, este capítulo abordará los más relevantes a nuestro criterio y describirá una posible solución. También se mencionarán algunos puntos, que bien podrían aportar positivamente a la solución, pero que no fueron abordados principalmente por quedar fuera del alcance de este proyecto.

## 9.1. Flujo de datos

El flujo de datos del prototipo consta de la entrada de la primera etapa, que es el video que proviene de cada cámara, luego se encuentra la salida que es un archivo XML por cámara, el cual contiene los datos del resultado del procesamiento. Este conjunto de archivos XML es utilizado por la segunda etapa para el procesamiento unificado.

Para realizar las pruebas de sistema se utilizaron secuencia de video que varían entre 4 y 6 minutos de duración. Para este lapso de tiempo, el tamaño de los archivos XML de la primera etapa es en promedio de 4 MB para las cámaras que cubren las aéreas y 13MB para las que cubren el centro del campo.

Osea se requiere en promedio (con 6 cámaras) 42 MB (4*4+ 16*2) de datos para 5 minutos de partido. El principal problema es que el "parser" de los archivos XML no accede en forma secuencial a la información, sino que carga en memoria todo el

archivo. Este es el comportamiento normal de los parsers XML.

Quiere decir que, para la segunda etapa, es necesario cargar en memoria los 42MB de datos XML al inicio del proceso. Esto se traduce en cientos de MB en memoria RAM, ya que en general la estructura de datos necesaria para representar un árbol XML ocupa mucho más espacio que los datos planos en el archivo.

La primera solución para este problema es sustituir el "parser" que utiliza ViPER por algún otro que acceda en forma secuencial a la información. Otra solución más comprometida, sería eliminar completamente el uso de archivos XML y utilizar una base de datos en la que, en la primera etapa inserte y en la segunda etapa acceda a la información. Otra opción es adoptar una arquitectura completamente diferente a la nuestra, como la utilizada en [8], que utiliza datagramas IP en "broadcast" para hacer llegar la información del procesamiento por cámara a una unidad central donde se realiza el procesamiento unificado.

## 9.2. Movimiento de cámara

Fue estudiado e implementado un módulo que soluciona los pequeños movimientos de las cámaras. Este módulo utiliza un máscara de las líneas como la que se ve en la figura 9.1.

El principal problema con el movimiento de la cámara es la falsa detección de las líneas del campo de juego. La máscara es utilizada para realizar un resta lógica cuadro a cuadro y poder eliminar las líneas. El problema es que los objetos que se encuentren sobre la línea también serán eliminados o fraccionados. Una solución es aplicar dilataciones para volver a unirlos. En conclusión, vimos que los movimientos de la cámara representan menos del 1 % de nuestros datos y que la posible mejora en esas partes, se ve contrarrestada por la distorsión mencionada anteriormente.

Queda pendiente una solución que lleve a una mejora global y no solo en las partes donde ocurre el movimiento de la cámara .

Figura 9.1: Máscara utilizada para resolver el movimiento de la cámara.

## 9.3. Corrección del seguimiento

El modelo utilizado para el seguimiento, asume velocidad constante y tamaño constante de las manchas. Esto no es del todo cierto, sobre todo tratándose de jugadores de fútbol, donde generalmente podemos observar cambios bruscos de velocidad y dirección. Para resolver este problema, un mecanismo de corrección del seguimiento como el implementado en [8], busca detectar y corregir situaciones en donde el centroide predicho no cae dentro ninguna región de fondo. En este caso se busca corregir la predicción ubicando el centroide dentro de la región más próxima dentro de un umbral.

Con relación al seguimiento a nivel de cámara, creemos que el sistema podría utilizar la estimación de la categoría a la hora de asociar manchas con medidas además de la mínima distancia Mahalanobis. De esta forma, se agregaría un dato más que creemos ayudaría a realizar asociaciones más precisas.

## 9.4. Estimación de la categoría para el procesamiento por cámara

.

En la etapa de procesamiento por cámara, la categoría de cada mancha es estimada cuadro a cuadro. Este trabajo consume mucho procesamiento y es muy propenso a falla cuando hay oclusiones. El sistema podría utilizar un mecanismo similar al utilizado en el procesamiento multi-cámara donde la categoría es actualizada utilizando una combinación de factores como: la edad en cuadros, la estimación de categoría pasada, la última vez que esta mancha fue asociada con una medida. De esta forma, se podría por ejemplo estimar la categoría solo en los cuadros pares, reduciendo a la mitad el procesamiento en este punto.

## 9.5. El balón

La detección y seguimiento del balón, representa un problema aparte que queda por fuera del alcance de este proyecto. Dada la observación del balón en una cámara, su posición real (3D), no puede ser determinada. El balón podría estar rodando por el piso o volando por el aire. Por referencias para métodos de seguimiento del balón ver [11].

## 9.6. Clasificación de categorías

Se notó que debido al método de clasificación (inspección por contenidos de colores de las imágenes miniatura segmentadas) cuando dos o más jugadores se cruzan, el algoritmo de seguimiento descripto anteriormente (en la sección procesamiento por cámara) continua siguiendo a los jugadores aún dentro de la oclusión, se genera un cruce entre dos o más manchas, dando así períodos de tiempo en donde se mezclan los colores de las manchas.

En el instante en que se cruzan dos manchas, A y B intersectan parte de sus características de color de cada mancha individual, cuando queremos determinar la categoría de la mancha A, en esos instantes de cruce hay muchos pixeles pertenecien-

tes al la mancha B que quedan incluidos dentro del rectángulo de la mancha A, por lo cual hay muchos colores agregados a los que originalmente se querían clasificar.

Vemos que, para obtener un buen conjunto de imágenes miniatura representativos por cada categoría y obtener histogramas promedio realmente significativos para dichas categorías, es importantísima la segmentación. Comienza por allí todo, por ejemplo en sus parámetros, como la dilatación que se le realiza a las manchas obtenidas de la sustracción del fondo, cuanto más se dilaten las manchas tendremos imágenes miniatura con más pixeles, pero también con más ruido (pasto, líneas, etc.), también ésta dilatación hace que se produzcan más oclusiones entre manchas y con ello los errores anteriormente mencionados en la clasificación.

En la figura 9.2 vemos como se forma una imagen miniatura con colores mezclados por dos objetivos que se están ocluyendo.

Figura 9.2: imagen miniatura no deseada.

Lo ideal en este sentido es lograr una segmentación que logre formar las manchas justas del tamaño de la silueta de cada objetivo, así los rectángulos contenedores que encuadran perfectamente a cada mancha segmentada y las imágenes miniatura (sub imagen del cuadro, centrada en el centroide del jugador y de dimensiones del rectángulo contenedor del jugador).

Figura 9.3: imagen miniatura deseado.

Otro caso detectado es cuando se regresa de una oclusión y el algoritmo de seguimiento no enmarca bien al jugador que se quiere seguir, la imagen miniatura

queda por unos cuadros, definida más que nada por el fondo (pasto) que por pixeles correspondientes al uniforme del jugador.

En nuestro caso optamos por incluir otro parámetro que es el histograma del fondo del video, este histograma se calcula igual que los otros histogramas promedios de las categorías. Es decir se guarda un conjunto de imágenes miniatura que solo contienen el fondo y se le calcula el promedio.

En una versión posterior a este trabajo, se debería agregar otra categoría más a las 5 definidas, que correspondan a las manchas erróneas, de esta manera estos casos anteriormente comentados, caerían dentro de esta categoría de validez nula.

También se podría mejorar el caso de las falsas detecciones por movimientos en las cámaras como se menciona anteriormente.

En este vínculo radica el equilibrio del buen funcionamiento de la clasificación, si hay una buena segmentación acompañado por un buen armado de datos de entrenamiento, esto se traduce en una buena clasificación posterior de las manchas segmentadas, las imágenes miniatura efectivamente representan objetivos y las comparaciones con los histogramas promedio encontrados para cada modelo funcionan mejor.

# Apéndice A

# Homografía

Se denomina homografía a toda transformación proyectiva que determina una correspondencia entre dos figuras geométricas planas, de forma que a cada uno de los puntos y las rectas de una de ellas le corresponden, respectivamente, un punto y una recta de la otra. Para determinar la homografía entre dos planos se necesitan 4 pares de correspondencias (de las 4 correspondencias, no puede haber 3 puntos colineales).

En nuestro caso es necesario aplicar dicha transformación (homografía entre el plano de la imagen capturada y el modelo de la cancha) a los puntos que representan los pies de los jugadores. De esta manera podremos representar la ubicación de los jugadores en el campo de juego. Utilizaremos el método de Ransac para estimar las homografías correspondientes entre cada cámara y el modelo de la cancha. En la figura A.1 se puede ver un ejemplo.

**Algoritmo de Ransac**

Ransac permite robustizar el ajuste mediante el conjunto de muestras mínimo requerido para ajustar el modelo, ajusta un modelo a este conjunto de datos y comprueba el consenso en este modelo entre las muestras restantes. Así, los modelos que contienen valores extremos son rechazados dado que no generan el suficiente consenso.

**ISSIA Dataset**

Para el juego de videos de ISSIA los datos que se tiene para el cálculo de la

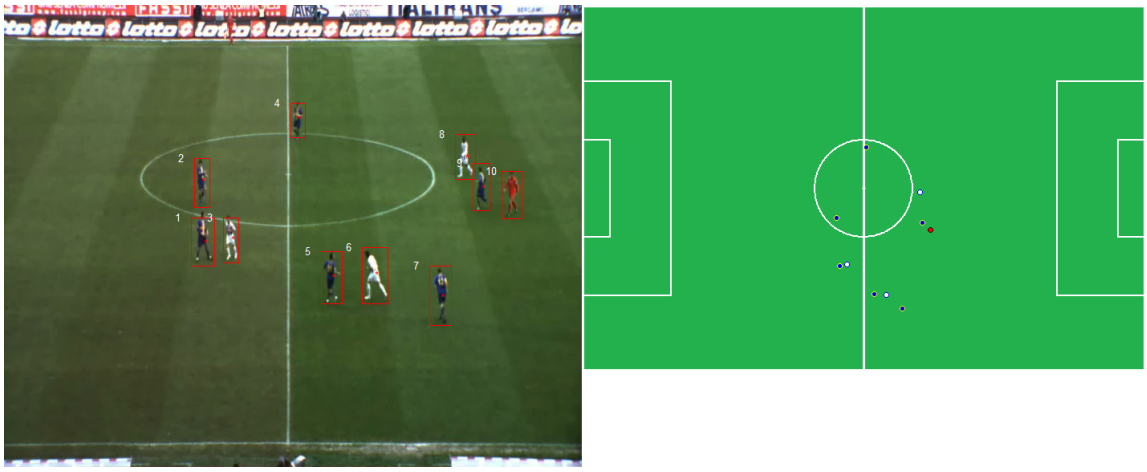

Figura A.1: Representación en el campo de juego de los jugadores detectados en la imagen.

homografía son los puntos marcados en la imagen capturada por la cámara y su representación en el plano de la cancha. De esta manera se tiene un buen número de correspondencias a las que se le aplicará el método de Ransac para estimar la homografía.

**SCEPTRE Dataset**

Para el caso de los datos obtenidos de SCEPTRE la información que se brinda para cada cámara son las constantes cargadas por los archivos de la rutina de calibración según lo definido por el código de calibración de cámara de Tsai [7].

El sistema de coordenadas tiene su origen en el centro del campo de juego, el eje x se alinea con las líneas laterales.

Datos:

* $C_x, C_y$ – intersección del eje Z del sistema de coordenadas de la cámara.
* $s_x$ – factor de escala que tiene en cuenta cualquier incertidumbre en el muestreo del capturador de cuadro en la dimensión horizontal.
* f – distancia focal efectiva de la cámara pinhole.
* k – coeficiente de primer orden de la distorsión radial.

* $T_x, T_y, T_z$ – componentes de traslación de la transformación entre el cuadro de las coordenadas del mundo y el cuadro de las coordenadas de la cámara.

* $R_x, R_y, R_z$ – componentes de rotación de la transformación entre el cuadro de las coordenadas del mundo y el cuadro de las coordenadas de la cámara.

Dado un punto $(X_i, Y_i)$ de la imagen capturada por la cámara lo que queremos es hallar su representación $(X_m, Y_m)$ en el modelo 2D del campo de juego. Para ello es necesario resolver el siguiente sistema de ecuaciones según [7].

Transformación del sistema de coordenadas 3D del mundo $(X_i, Y_i)$ al sistema de coordenadas sin distorsión del plano de la imagen $(X_u, Y_u)$.

$$X_u = f * \left(\frac{X_i}{Z_i}\right) \tag{A.1}$$

$$Y_u = f * \left(\frac{Y_i}{Z_i}\right) \tag{A.2}$$

$$X_u = f * \left(\frac{r_1 X_m + r_2 Y_m + T_x}{r_7 X_m + r_8 Y_m + T_z}\right) \tag{A.3}$$

$$Y_u = f * \left(\frac{r_4 X_m + r_5 Y_m + T_y}{r_7 X_m + r_8 Y_m + T_z}\right) \tag{A.4}$$

Una vez que se tienen las coordenadas de unos cuantos puntos en el plano de la imagen y sus correspondientes coordenadas en el plano del modelo del campo de juego se puede hallar la homografía (aplicando el método de Ransac) que lleva de un plano a otro.

**Franzini Dataset**

Para los videos grabados por el grupo de estudio en el estadio Franzini no se tuvo la oportunidad de marcar los puntos en el campo de juego y medir su distancia respecto a un origen conocido para luego representarlos en el modelo del campo de juego. Por lo tanto se utilizaron puntos conocidos marcados en la cancha como pueden ser los vértices de las aéreas por ejemplo. De esta manera se logró calcular la homografía para las cámaras 1, 2 y 4 ya que se contaba con la cantidad de correspondencias suficientes. Para calcular la homografía de la cámara 3 fue necesario

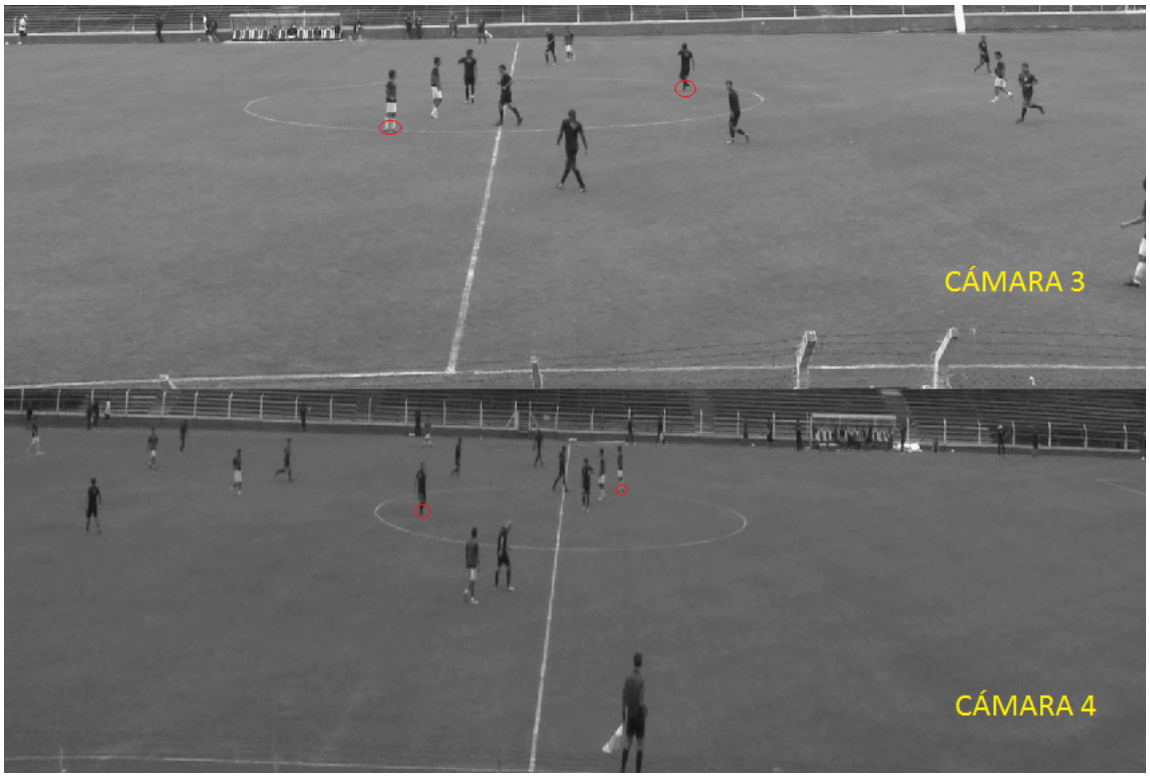


Figura A.2: Ejemplos de puntos que se pueden marcar en ambas cámaras para obtener la homografía de la cámara 3.

marcar un par de puntos en un cuadro determinado, como pueden ser los pies de un jugador, tanto en la cámara 3 como en la 4 (estas cámaras están enfrentadas, ver figura 3.7). Como los videos están sincronizados, ambas cámaras están capturando la misma escena desde diferente lugar. Lo que hacemos es marcar dos puntos en uno de los cuadros capturados por la cámara 4 y hallamos sus correspondientes en el modelo de la cancha. Luego en el mismo cuadro pero en la cámara 3 marcamos los mismos puntos que elegimos en la cámara 4 y los hacemos corresponder con sus respectivos puntos en el modelo. De esta manera obtenemos dos correspondencias más para el cálculo de la homografía. En la figura A.2 se muestra un ejemplo de esta situación.

# Apéndice B

# El filtro de Kalman

**¿Qué es el Filtro de Kalman?**

Es un algoritmo de procesado de datos óptimo recursivo. Óptimo porque minimiza un criterio determinado y porque incorpora toda la información que se le suministra para determinar el filtrado. Recursivo porque no precisa mantener los datos previos, lo que facilita su implementación en sistemas de procesado en tiempo real. Por último, algoritmo de procesado de datos, ya que es un filtro, pensado para sistemas discretos. El objetivo del filtro de Kalman es estimar los estados de una manera óptima, de manera que se minimiza el error cuadrático medio.

En el ámbito de la Visión Artificial el filtro de Kalman es un algoritmo recursivo que se utiliza para estimar la posición de un punto o característica en movimiento y la incertidumbre de la medida, en la siguiente imagen. Se trata de buscar la característica (punto, borde, esquina, región, etc.) en un área determinada de la siguiente imagen alrededor de la posición predicha, en la que estamos seguros de encontrar la característica dentro de un cierto grado de confianza.
El objetivo del filtro es la obtención de un estimador óptimo de las variables de estado de un sistema dinámico, basado en observaciones ruidosas y en un modelo de incertidumbre de la dinámica del sistema.

De manera que su evolución es expresada en espacio de estados por:

$$x(k+1) = Ax(k) + Bu(k) + v(k) \tag{B.1}$$

$$y(k) = Cx(k) + w(k) \tag{B.2}$$

donde,

$u(k) = 0 \; \forall$ k, es la entrada al sistema
$x(k)$ es el estado
$y(k)$ es la salida u observación del sistema
$w(k)$ proceso estocástico que modela el ruido asociado a la medida
$v(k)$ proceso estocástico que modela el ruido asociado al sistema
$A$, $B$, $C$ matrices determinísticas que definen la dinámica del sistema

Se deben asumir las siguientes condiciones:

E[x(0)] = 0
E[w(k)] = 0 $\forall$ k
E[v(k)] = 0 $\forall$ k
E[x(0),w(k)] = 0 $\forall$ k
E[x(0),v(k)] = 0 $\forall$ k
E[v(k),w(j)] = 0 $\forall$ k
E[w(k),w(k)] = R(k)
E[w(k),w(k)] = 0 $\forall$ k $\neq$ j
E[v(k),v(k)] = Q(k)
E[w(k),w(j)] = 0 $\forall$ k $\neq$ j
E[x(0),x(k)] = 0 $\forall$ k

Las matrices de covarianza Q(k) y R(k) son diagonales y por tanto simétricas. En el sistema real podremos observar el valor de y(k) de manera directa con los sensores adecuados. Esta medida incorporará una serie de incertidumbres asociadas: la incertidumbre del sensor y la del sistema. Por otro lado solo podremos acceder al valor y(k). En caso de necesitar la evolución completa del estado x(k) y/o de precisar el valor de la observación ajena a las variaciones provocadas a la incertidumbre, tendremos que estimar de alguna manera indirecta sus valores.

Cuando el filtro de Kalman se aplica a la Visión Artificial, el estado x se corresponde con el vector posición del objeto en la imagen determinado por las coordenadas

de posición $x_x$ y $x_y$, y las coordenadas de velocidad $v_x$ y $v_y$. La observación z en cambio, es únicamente un vector de dos componentes $z_x$ y $z_y$, correspondiente a las coordenadas de la posición observada del objeto de interés.

La matriz A de dimensiones NxM relaciona el estado en tiempo k con el estado en tiempo k+1.

$$\begin{cases} x_{x_{k+1}} = x_{x_k} + v_x t \\ x_{y_{k+1}} = x_{y_k} + v_y t \\ v_{x_{k+1}} = v_{x_k} \\ v_{y_{k+1}} = v_{y_k} \end{cases} \tag{B.3}$$

$$A = \begin{pmatrix} 1 & 0 & t & 0 \\ 0 & 1 & 0 & t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{B.4}$$

En nuestro caso t = 1 (paso temporal entre el cuadro genérico k+1 y k).

El filtro de Kalman propone un método para obtener un estimador óptimo del estado. Si suponemos que $\hat{x}(k)$ es la estimación en el instante k del estado. El filtro de Kalman buscará obtener ese valor de estimación de manera que se minimice el error cuadrático medio. Definiendo el error como la diferencia entre el valor real del estado y la estimación:

$$e(k) = x(k) - \hat{x}(k) \tag{B.5}$$

El objetivo será minimizar:

$$P(n) = E\left\{e(n).e_T(n)\right\} \tag{B.6}$$

A la matriz P(n) se la conoce como matriz de covarianza del error. Dependiendo del valor que tome n en B.6, tendremos distintas representaciones del filtro de Kalman.

Si n = k+1, el filtrado será de predicción, si n = k, el filtrado será de alisado.
El objetivo consiste en determinar los valores de x(k) conocidas las medidas y(0),

y(1), ..., y(k) para que P(k) sea mínima, o visto desde otro punto de vista, el objetivo es determinar los valores de $\hat{x}(k+1)$ a partir de las medidas de la observación y(k+1) para que la matriz P(k+1) sea mínima. El filtrado de $\hat{x}(k+1)$ propuesto por Kalman y Bucy, se realizará a partir del estado anterior y de un factor de corrección que será función del error.

El algoritmo tiene dos pasos que son ejecutados de forma iterativa. Predicción: antes de tener la medida y(k+1) y Corrección o actualización del estado.

**Paso 1: Predicción-Extrapolación-Estimación**

Calculamos una predicción del estado x(k+1), lo notaremos como x'(k+1). El valor de la predicción es calculado a partir del valor más actualizado del estado (posteriormente veremos qué es el estado corregido en la parte final del algoritmo).

$$x'(k+1) = A\hat{x}(k) \tag{B.7}$$

Predecimos el valor de la matriz de covarianza del error previo a la medida P'(k+1) (recordamos que estamos haciendo una previsión del estado). Como se indicó en B.6, el error estará definido por:

$$e'(k+1) = x(k+1) - x'(k+1) \tag{B.8}$$

Por tanto e'(k+1) en esta etapa de predicción vale:

$$e'(k+1) = Ax(k) + v(k) - A\hat{x}(k) \tag{B.9}$$

$$P'(k+1) = AP(k)A^t + Q(k) \tag{B.10}$$

**Paso 2: Innovación-Actualización-Corrección**

Decíamos antes que el valor del estado se va a calcular a través del estado anterior y de una corrección que es función del error. Esto es:

$$\hat{x}(k+1) = x'(k+1) + K(k+1)[y(k+1) - Cx'(k+1)] \tag{B.11}$$

Siendo el factor de corrección:

$$K(k+1)[y(k+1) - Cx'(k+1)] \tag{B.12}$$

donde y(k+1) es el último valor observado y x'(k+1) es el valor más actualizado disponible del estado (calculado en la fase de predicción).

Buscaremos el valor de K(k+1) para conseguir un valor óptimo de $\hat{x}(k+1)$ tal que la matriz de covarianza del error sea mínima.

**Inicialización del algoritmo:**

Para iniciar el algoritmo es necesario conocer las siguientes condiciones de contorno: un valor inicial del estado x(0); el valor de la matriz de covarianza del error, que para P(0) podemos asumir que P(0) = Q y los valores de las matrices de covarianza asociadas al sistema y a la medida: Q y R.

**Funcionamiento iterativo**

En el paso genérico k:

**Predicción:**

$$x'(k+1) = A\hat{x}(k) \tag{B.13}$$

$$P'(k+1) = AP(k)A^t + Q(k) \tag{B.14}$$

**Corrección:**

$$K(k+1) = CP'(k+1)[CP'(k+1)C^t + R(k+1)]^{-1} \tag{B.15}$$

observación y(k+1)

$$\hat{x}(k+1) = x'(k+1) + K(k+1)[y(k+1) - Cx'(k+1)] \tag{B.16}$$

$$P(k+1) = [I - KC]P'(k+1) \tag{B.17}$$

Referencias: [4]

# Apéndice C

# ViPER: The Video Performance Evaluation Resource

**Información general**

En el laboratorio LAMP (Language and Media Processing, [16]), gran parte de la investigación se centra en el análisis de videos en su contenido semántico. Esto incluye el seguimiento de personas, detección de texto, y así más. El software Video Processing Analysis Resource es un conjunto de herramientas de scripts y programas en Java que permiten el marcado de los datos visuales verdaderos (ground truth), y sistemas para evaluar cuán cerca está el conjunto de datos del resultado de los datos verdaderos.

**El problema de la Evaluación de Performance**

Con el fin de evaluar un algoritmo de análisis de video, o un conjunto de algoritmos, es necesario definir una metodología. Ya que hay muchos libros y artículos que describen los métodos para la evaluación de determinados tipos de algoritmos, se decidió desarrollar un marco general para la evaluación. La idea básica común en la mayoría de los tipos de evaluación que hacemos es una comparación entre los resultados generados por computadora y una versión ideal de la "Verdad".

En algunos sub-campos de visión, como el procesamiento de documentos, es posible generar de forma automática los datos de prueba. Sin embargo, para el procesamiento de video, es más común para un ser humano definir un "ground truth" (GT)

para cada video clip. Con el fin de garantizar que los investigadores pueden repetir y verificar las evaluaciones, es importante hacer que los datos del GT estén disponibles para otros investigadores en un formato documentado. Es muy útil disponer de métodos de verificación cualitativa del GT, así como, con metadata browsers y editores. ViPER-GT ofrece herramientas para la creación y edición de video metadata.

Hay muchas maneras de definir cómo corregir un conjunto de datos de resultados con respecto a un conjunto de datos de GT. Un indicador que mira la diferencia de tamaño de los bounding box para la detección de texto puede dar resultados diferentes a una métrica más orientada al objetivo que opera en el número de caracteres o palabras correctamente reconocido. VIPER-PE proporciona herramientas para resolver el problema de la evaluación.

### Herramienta de creación de ground truth - ViPER Ground Truth

ViPER-GT ofrece el proceso de creación de ground truth mediante una interfaz gráfica de usuario Java. Está diseñado para permitir cuadro a cuadro el marcado de datos del video y almacenarlos en formato ViPER (o xml). También es útil para la visualización. Para más información, consulte la página del producto `http://viper-toolkit.sourceforge.net/products/gt/`.

### Herramienta de evaluación de performance - ViPER Performance Evaluation

ViPER-PE es una herramienta de línea de comando de evaluación de performance. Ofrece una variedad de métricas para realizar la comparación entre los archivos de metadatos de video. Con él, el usuario puede elegir entre varios indicadores para comparar el conjunto de datos del resultado con los datos de ground truth. Se pueden dar métricas de precision y recall, realizar evaluaciones cuadro a cuadro y basada en objetos, y también cuenta con un mecanismo de filtrado para evaluar subconjuntos relevantes de los datos. Más información se puede encontrar en la página del producto `http://viper-toolkit.sourceforge.net/products/pe/`.

# C.1. Herramienta ViPER Ground Truth

### Especificaciones de la interfaz de usuario

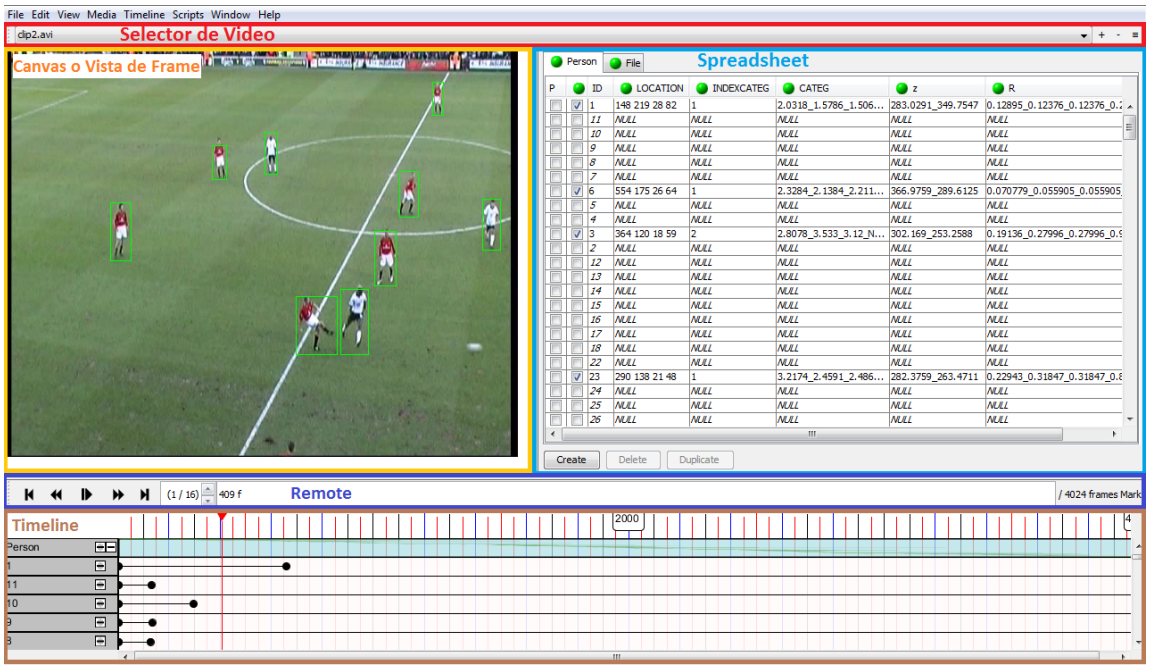


Figura C.1: Interfaz gráfica de usuario completa con un video etiquetado.

La figura C.1 muestra cómo se verá la interfaz de usuario cuando se carga un video con su respectivo archivo de anotaciones (etiquetas). Podemos ver claramente las siguientes regiones en la interfaz.

- Barra de menú: contiene los controles de menú para abrir y etiquetar un video, las operaciones de zoom, la creación de descriptores y el establecimiento de sus atributos y un menú de ayuda.

- Canvas: es la porción de la imagen que muestra el video. Soporta operaciones de zoom y panning sobre el video.

- Spreadsheet: contiene una pantalla tabular de los descriptores y sus atributos. Se puede editar los atributos.

- Timeline: la línea de tiempo está por debajo del Canvas y del Spreadsheet. Contiene un panel de control remoto para navegar por el video. También contiene un panel de línea de tiempo. El panel de línea de tiempo contiene los descriptores y sus líneas de validez. Las líneas de validez indican al usuario en qué cuadros los descriptores son válidos.

**Realizar operaciones en la interfaz de usuario**

***Entrada / Salida***: ViPER-GT tiene dos tipos de datos de entrada: archivos de video (video en formato mpeg) y los archivos de ground truth que describen el video.

***Crear un archivo de datos***: Para crear un nuevo archivo, archivo de datos vacío, utilice la ventana de diálogo que aparece cuando se selecciona el elemento de menú File >> New ViPER File...

***Apertura de archivos de datos existentes***: Para abrir un archivo de datos, utilice la ventana de diálogo que aparece cuando se selecciona File >> Open Existing ViPER File...

***Adición de un nuevo video y extracción de un video***: Se puede notar un botón "+" debajo de la barra de menú. Haga clic en ese botón para abrir un cuadro de diálogo FileChooser. Se puede agregar un nuevo video en el menú desplegable usando este control. Se puede utilizar el botón "-" en forma similar para eliminar un video. Se pueden realizar las mismas operaciones usando el menú Media >> Add Media File y Media >> Remove Media File.

**Edición de descriptores**

***Modificación de un descriptor***: La mayoría de los atributos se pueden editar directamente en la vista de spreadsheet, que contiene los descriptores de contenido, en la primera pestaña si las hay, y los descriptores de objeto en las pestañas a la derecha/debajo de la primera pestaña. Editar descriptores es editar elementos en el spreadsheet. En nuestro caso definiremos descriptores de objetos (OBJECT

descriptors) para etiquetar a los jugadores.

Hay que tener en cuenta que los descriptores de contenido y objeto ambos contienen dos columnas, con la etiqueta P y V. La casilla de verificación P se utiliza para la propagación y la interpolación mientras que la casilla V es para su validez, lo que indica que el descriptor aparece o tiene significado para el cuadro actual. Por ejemplo un descriptor de objeto que representa a jugadores tiene validez en aquellos cuadros donde aparece el jugador, su casilla V debe estar sin marcar en todos los demás cuadros.

Además, hay que tener en cuenta que los descriptores de objetos tienen una columna ID. Esta columna no se puede editar, sino que se generan de forma automática. También hay que tener en cuenta la visibilidad del globo junto al nombre del descriptor en las pestañas. Cuando este es de color rojo, los atributos están ocultos y no pueden ser seleccionados. Al hacer clic en el globo alterna entre visible (verde) e invisible (rojo). Los atributos de los descriptores de objetos pueden ser estáticos o dinámicos, los atributos estáticos tienen un asterisco antes de sus nombres en los encabezados de la columna y sólo pueden tener un valor durante la vida del descriptor, pero los atributos dinámicos pueden tener un valor diferente para cada cuadro donde el contenido del descriptor es válido.

***Agregar descriptores de objetos***: Para crear un objeto nuevo con todos los atributos inicializados en NULL, hay que pulsar el botón "Create" en la pestaña del descriptor de objeto adecuado.

***Duplicar un descriptor de objeto***: Para copiar un descriptor de objeto, seleccione la fuente haciendo clic en la casilla correspondiente de la columna "ID" y haga clic en el botón "Duplicate" en la parte inferior de la vista de spreadsheet.

***Eliminar un descriptor de objeto***: Para eliminar un objeto, primero hay que hacer clic en la casilla de la columna "ID" y luego en el botón "Delete".

***Atributos***

Los atributos se dividen en dos categorías: texto (cadena de texto) y espaciales (formas). Los atributos de texto sólo se pueden editar en la tabla de datos, mientras que los espaciales también se pueden editar en el Canvas. Antes de editar un atributo, se debe estar seguro de que el objeto cuyo atributo se está editando es válido para el cuadro actual, es decir, asegurarse de que la casilla de la columna "V" está marcada para el descriptor cuyo atributo desea editar.

***Atributos de texto***

Los tipos de los atributos texto son bvalues, dvalues, fvalues, svalues y lvalues. Para modificar los atributos de texto, a excepción de lvalue y bvalue, seleccione el atributo en la tabla de datos con un clic en su casilla. Con un solo clic selecciona la casilla para editar y coloca el cursor al final del texto actual, mientras que un doble clic selecciona una palabra en la casilla y un triple clic selecciona todo el texto en la casilla. Atributos del tipo lvalue ofrecen una lista desplegable en lugar de un cuadro de texto, y pueden ser editados solamente con el mouse.

***Atributos espaciales***

Atributos espaciales, que en la versión actual son todas las formas, son atributos que se pueden extraer y manipular en el Canvas. Los tipos espaciales de atributos incluyen bboxes, oboxes, elipses, polígonos abiertos y cerrados, puntos y círculos. En nuestro caso utilizaremos los del tipo bboxes para etiquetar a los jugadores.

Un atributo espacial pasa por dos fases. En principio, el atributo espacial debe ser creado. Esta creación típicamente se inicia al hacer clic en una celda con NULL, y asegurándose de que la casilla de la columna "V" está seleccionada. Una vez que la forma se ha elaborado, comienza la segunda fase donde se puede editar. Los atributos espaciales pueden ser editados de dos maneras. Pueden modificarse los valores en el spreadsheet, o editar en la vista de cuadro. Por lo general es más fácil de editar en la vista de cuadro, donde hay apoyo para la manipulación de interfaz gráfica de usuario.

La manipulación de los atributos espaciales requiere saber cuándo se está en el modo de creación o en modo de edición. El modo de creación ocurre cuando un atributo espacial NULL válido ha sido seleccionado en la vista de spreadsheet, o cuando

se cambia de cuadro y el atributo espacial seleccionado es NULL. A continuación se describe el atributo espacial bbox.

***Atributo espacial Bbox***: El Bbox es una forma de rectángulo no orientada. Sólo se pueden colocar con los lados paralelos al eje del Canvas. La representación escrita de un Bbox usada en la vista de spreadsheet, es x y h w donde x e y son las coordenadas de la esquina superior izquierda del rectángulo, h es la altura y w es el ancho.

***Crear atributos espaciales en el Canvas***

Para crear un nuevo atributo espacial, se debe hacer clic en un atributo espacial NULL en la vista de spreadsheet. Asegúrese de que el atributo es válido en el cuadro. A continuación, seguir con las siguientes instrucciones. Una vez que la forma se ha creado en el Canvas, se entra en el modo de edición, que se describe en la siguiente sección.

***Crear Bbox***: Para crear un Bbox, presione el botón izquierdo del mouse en el Canvas. Esto establecerá una de las esquinas del Bbox. Arrastre el mouse, cuando se suelta el botón del mouse se define la otra esquina del Bbox.

**Dibujar y editar**

Se pueden crear y editar formas en la vista de cuadro y en la vista de spreadsheet. Para dibujar o modificar las formas en el panel de datos, se debe seleccionar la casilla del atributo y el descriptor que se desea editar y escribir un valor apropiado para el tipo de datos.

***Bbox***: Si un Bbox es seleccionado (el mismo estará en rojo), se puede mover el mouse sobre el borde, esquina o el interior del Bbox. Si el cursor está cerca de una esquina, se dibuja un círculo, lo que indica que puede arrastrar esa esquina para redimensionar el Bbox. Al soltar el mouse cambia el tamaño del Bbox. Si el cursor

está cerca de un borde, el borde se resalta (se vuelve más grueso y de color magenta), y se puede arrastrar ese borde. Al arrastrar un borde conserva altura o anchura dependiendo del borde que se arrastra. Si el cursor se encuentra dentro del Bbox, los cuatro lados se destacan. Si se arrastra el mouse se traslada todo el Bbox. Pulse el botón "Delete" para quitar la caja.

### Operaciones avanzadas: la propagación y la interpolación

#### *Información general*

ViPER-GT permite a los usuarios realizar dos operaciones sobre los valores de los atributos de los descriptores de objetos seleccionados en un rango de cuadros. Las operaciones son la propagación y la interpolación. Es posible invocar estas operaciones directamente de la tabla haciendo clic en el descriptor, o por medio de la línea de tiempo arrastrando de un lugar a otro. En la figura C.2 se puede ver un ejemplo de atributos seleccionados con la opción de propagación.

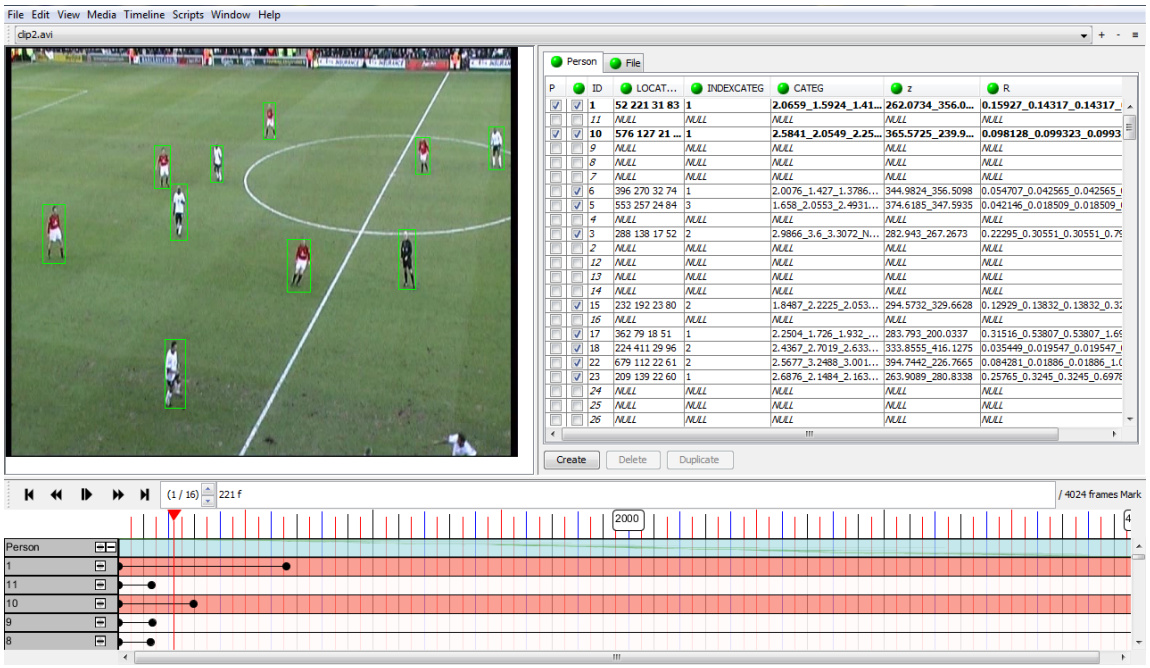


Figura C.2: Los descriptores 1 y 10 tienen los atributos establecidos para propagar.

**Propagación en Detalle**

La propagación copia el valor del cuadro actual de los descriptores de objeto seleccionados a todos los cuadros en el rango de propagación. Todos los valores de los descriptores de objeto elegidos son reemplazados con los valores en el cuadro actual.

Para propagar el valor de un descriptor, se debe ir al cuadro que contiene el valor que se desea copiar y marcar las casillas en la columna "P" para el descriptor o descriptores que desea propagar. A continuación, puede ver el video o arrastrar el cursor hasta el cuadro final deseado para la propagación. La segunda opción es simplemente cambiar los cuadros de forma incremental. Todos los cuadros entre el cuadro actual y el cuadro que se eligió para que tome el valor actual de todos los descriptores han sido marcados para la propagación.

Se debe tener cuidado ya que una vez que se haya marcado la casilla, esta permanecerá marcada hasta que se desactive o se salga del programa, el cambio de pestaña no es suficiente, ya que es posible que desee propagar o interpolar dos tipos de descriptores a la vez. Esto es importante, es una de las herramientas más fácil y más peligrosa de usar, la manera de propagar un valor es simplemente dejar la casilla "P" marcada y pasar por un video de un cuadro a la vez, si se decide ir un centenar de cuadros hacia atrás, pero se olvida guardar y desmarcar la propagación primero, se pueden borrar fácilmente un centenar de cuadros de trabajo.

## C.2. Herramienta ViPER Performance Evaluation

Es bueno comentar que podemos medir la performance parcial del sistema al generarse el tracking y la clasificación por cada cámara, esto lo podemos hacer mediante la herramienta ViPER-PE y con la ayuda de un archivo de datos "verdaderos" GT (Ground Truth) contra el cual se compararan los resultados obtenidos por el sistema. Primero vamos a describir las características básicas del ViPER-PE y cómo lo utilizamos en nuestro caso.

VIPER utiliza el archivo de parámetros de evaluación (EPF) en combinación con un conjunto de propiedades para configurar una evaluación específica entre los datos

a testear y el archivo de Ground Truth (GT).
VIPER-PE dispone de dos formatos de salida: un archivo de texto legible por humanos, y un espacio delimitado para la lectura de la máquina. La herramienta makeGraph puede convertir el formato de salida legible por máquina en gráficos, mientras que la herramienta RunEvaluation puede comparar varias series de experimentos.
La evaluación de la performance consiste en comparar los datos generados, llamado el candidato o los datos de resultados, con el ground truth, o los datos a testear. Teniendo en cuenta los descriptores (grupos de candidatos) y los datos del objetivo a testear, la herramienta de evaluación del rendimiento genera una descripción de lo bien que se aproximan los candidatos (los resultados) con los objetivos (la verdad) dados los parámetros de configuración y los descriptores.

**Análisis de objetos: Matching entre candidatos y objetivos**

El sistema ViPER-PE intenta hacer coincidir objetos candidatos a objetos de destino. Se trata de responder a la pregunta: ¿Qué tan cerca están dos descriptores? Con esta idea, se determina que los objetivos y los candidatos están muy próximos (según alguna métrica configurable), y se generan los valores de presicion and recall de la cantidad de objetos comparados. Se trata de definir un espacio para los descriptores de distancia, de preferencia una métrica. En ViPER-PE, todas las distancias se han normalizado entre cero y uno. Una lista completa de los indicadores disponibles se incluye en la página del producto.
El análisis de objetos se divide en varias fases: detección, localización, y la comparación estadística. Cada fase es un mayor nivel de análisis.
En la figura C.3 se indican las diferentes métricas configurables para buscar "similitudes" o "emparejamientos" en los atributos testeados entre los objetivos y los candidatos.

Como vemos para la clasificación de la categoría nos interesaría utilizar una métrica de igualdad, pues en definitiva lo que queremos es comprobar dos índices de categorías y preguntar si son iguales. Esto es en el supuesto de que el GT no contiene ninguna variación en cuanto a la posición de los blobs, pero si queremos también medir la performance de la posición de los mismos con respecto a la verdadera posición, tendríamos que utilizar una métrica de solapamiento.

| Métrica | Definición |
|---|---|
| Equality | Dos atributos están cerca si son iguales. |
| Dice | El doble del área compartida sobre la suma. Esto evita la asimetría de la superposición métrica. |
| Overlap | Se utiliza como distancia la fracción del objetivo que se solapa con el candidato. |
| Maximum Deviation | La máxima distancia en que el objetivo y candidato se superponen. |
| String Edit Distance | Edición de otra forma diferente que se requiera. |

Figura C.3: Métricas configurables en ViPER-PE.

### C.2.1. Niveles de comparación utilizados

Como decíamos anteriormente tenemos tres niveles diferentes: detección, localización, y la comparación estadística. Cada fase es un mayor nivel de análisis.

#### Nivel 1 - Detección

Durante la fase de Detección, VIPER-PE examina los tipos de descriptores y los cuadros en los que estos se encuentran. Si un candidato se superpone a cualquier descriptor de un objetivo durante un determinado número de cuadros este se cuenta como detectado. Esto es lo que se busca en este primer nivel de detección, si queremos comparar una detección, empezamos por saber si fue este objetivo detectado y registrado como un posible candidato. Este primer nivel puede ser utilizado por ejemplo en sistemas que solo se quieren contar objetos sin importar el clasificarlos entre sí en diferentes categorías, simplemente en este primer nivel contamos si el objeto está o no, veremos a continuación que en los niveles sucesivos va aumentando el nivel de análisis.

#### Nivel 2 - Localización

La segunda fase de la localización funciona muy parecida a la detección. En lugar de buscar en todos los cuadros de los objetos, se compara solamente los cuadros donde los atributos son similares. La similitud de atributos es definida por el usuario. Para un par candidato-objetivo, los cuadros diferentes se cuentan como pérdidas o falsos a la hora de calcular la distancia. Entonces el resultado devolverá el número de cuadros en donde concuerdan los atributos del par candidato-objetivo. Cada distancia entre los atributos se calcula por separado, y se utiliza las propiedades y el

archivo de parámetros de evaluación (EPF).

**Nivel 3 - Comparación estadística**

El tercer nivel es la comparación estadística. Para las parejas candidato-objetivo localizadas, se calcula la distancia promedio, mediana, mínima o máxima y luego valores umbrales contra otros valores de tolerancia, se especifica como la propiedad level3_tool. Esto es útil para los descriptores que se cruzan varios cuadros. La distancia entre los descriptores se establece como ésta propiedad.

Luego de ver estos tres niveles de comparación, tomar las medidas de performance para un grupo de cuadros y ver qué resultados obtenemos al variar las entradas, por ejemplo al introducir XML de los mismos cuadros, pero con objetos diferentes, incluyendo falsas detecciones, situaciones de cambios de categoría transitoria (errores en la clasificación), etc.
Estudiaremos para ese conjunto de prueba los resultados en cada uno de los tres niveles anteriormente explicados y luego detallaremos el estudio final comentando los resultados de Presicion y Recall generales.
El archivo **object-Person.epf** será nuestro archivo de parámetros de evaluación (EPF) que comentábamos anteriormente, a continuación sus definiciones de equivalencias y los objetos de evaluación:

```
#BEGIN_EQUIVALENCE
LOCATION: LOCATION
INDEXCATEG: INDEXCATEG
#END_EQUIVALENCE
#BEGIN_OBJECT_EVALUATION
OBJECT Person [dice 1]
LOCATION: [dice .9]
INDEXCATEG: [E]
#END_OBJECT_EVALUATION
```

Vemos que el atributo LOCATION quedó configurado con un umbral de dice 0.9

(dice = Dice coefficient (intersection / sum of candidate and "Target")), mientras que la métrica para el atributo INDEXCATEG es la "E" - Equality.

En el archivo **properties.pr** se definen otro tipo de atributos para la clasificación de las métricas, tales como:
level3_metric = median
level3_tool = 0.99
range_tool = .99
Parámetros del funcionamiento del Nivel 3 y tolerancias para matcheos entre otros parámetros.

# Apéndice D

# Manual de usuario y juego de datos de prueba

En esta sección se dará una explicación de cómo operar los sistemas que forman este prototipo. Se crearon dos interfaces de usuario, una para el procesamiento a nivel de cámara y la otra para el procesamiento multi-cámara. También se explicará la estructura de los archivos de parámetros de entrada, utilizados en cada una de las etapas.

## D.1. Procesamiento por cámara

**1)** El script que ejecuta la interfaz de usuario para el procesamiento por cámara se llama Argos.m

```
>> argos
```

Se presenta al usuario la UI que se muestra en la figura D.1.

**2)** Se debe seleccionar el ID de la cámara a procesar (por defecto está seleccionada la 1). Luego este parámetro se podrá modificar con el botón cambiar.

**3)** Cargar parámetros: Archivo - Cargar Parámetros, seleccionar el archivo parameters.m.

**4)** El usuario presiona Start para comenzar a procesar, el número que se encuentra entre el botón de **Start** y el de **Pause** indica la cantidad de cuadros que se procesan en cada ejecución (ver figura D.2).

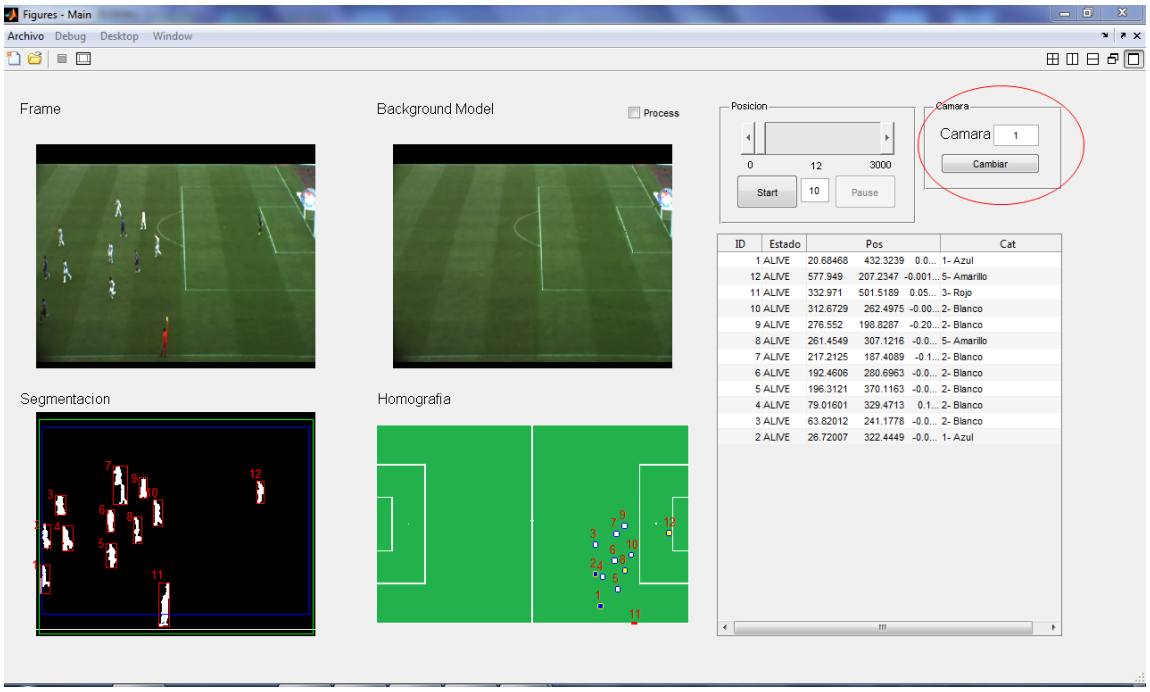


Figura D.1: UI del procesamiento por cámara.

**5)** Una vez finalizado ir a **Archivo - Guardar XML** para guardar el resultado del procesamiento en un archivo XML.

**6)** Modo batch: Se agregó una funcionalidad más a la UI que permite procesar todas las cámaras de forma desatendida. Para ello ir a Archivo - Modo batch. El sistema comenzará a procesar un número indicado de cuadros por cámara y almacenará el resultado en archivos. Para más información sobre cómo están configurados estos parámetros, ver la sección D.3.

## D.2. Procesamiento multi-cámara

Una vez que procesamos cada cámara (paso anterior), podemos pasar a la segunda etapa. Procesamiento multi-cámara.

**1)** Ejecutar el script MultiView.m

```
>> MultiView.m
```

Se presenta la UI al usuario (ver figura D.3).

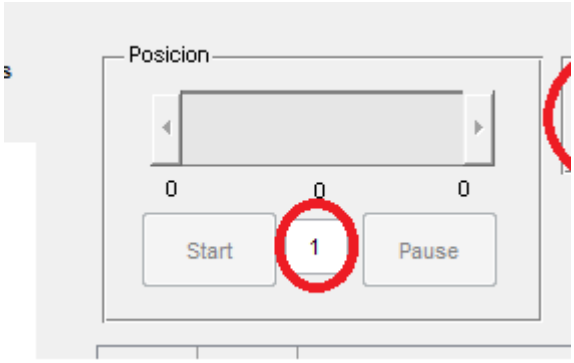


Figura D.2: Cantidad de cuadros que se procesan en cada ejecución.

**2)** Seleccionar **Archivo - Cargar Parámetros** y buscar el archivo, por ejemplo: parametersMView6cam.m.
**3)** El usuario presiona **Start** para comenzar a procesar. Se procesará la cantidad de cuadros que se encuentra a la derecha del botón **Start** (del mismo modo que funciona en la UI del procesamiento por cámara). El check box "modo batch", permite procesar sin refrescar la UI lo que acelera el proceso.
**4)** Es posible visualizar el error parcial de cada cámara o el error unificado, activando los check boxes que se encuentran debajo del número de cada cámara o debajo del cuadro donde se muestra el cuadro actual (marcados con rojo en la imagen D.3).
**5)** Luego de finalizar el procesamiento, en el menú **Archivo - Guardar XML**, se podrá almacenar la salida del sistema.

## D.3. Parámetros procesamiento por cámara

Se entregó junto con el código, los archivos que contienen los parámetros necesarios para procesar los datos. A continuación se dará una breve descripción de su estructura y parámetros. En este caso veremos como ejemplo el archivo "parameters" del juego de datos de ISSIA.

parameters.mat

Archivo de parámetros utilizado en procesamiento por cámara (Argos.m). Este


Figura D.3: UI del procesamiento unificado.

archivo contiene todos los parámetros de las 6 cámaras. A continuación se describirá la estructura de este archivo.

**parameters =**

**1x6 struct array with fields:**

**background**

**segmentation**

**homography**

**category_classification**

**media**

**traking**

**params_batch**

## D.3.1. Background

Parámetros para el proceso de substracción del fondo por cámara.

```
>> parameters(1).background
ans =
window_size: 100
energy_threshold: 95
a: 0.1000
b: 0.8000
threshold_pixel_intensity_change: [10 10 10]
initial_model: [1x1 struct]
```

| | |
|---|---|
| window_size | Mixture of Gaussians param [15]. |
| energy_threshold | Mixture of Gaussians param [15]. |
| a | Mixture of Gaussians param [15]. |
| b | Mixture of Gaussians param [15]. |
| threshold_pixel_intensity_change | Mixture of Gaussians param [15]. |
| initial_model | Modelo inicial (se trata de buscar una imagen con la menor cantidad de jugadores posible). |

Cuadro D.1: parameters(1).background

### D.3.2. Segmentation

Parámetros utilizados en el proceso de segmentación.

```
>> parameters(1).segmentation
ans =
difference_threshold: 11
dilate_quantity: 1
minimum_area: 200
pitch_mask: [576x720 uint8]
line_mask: [576x720 double]
```

### D.3.3. Homography

Sub sección dedicada a almacenar los parámetros utilizados para el cálculo de la homografía.

| | |
|---|---|
| difference_threshold | Umbral de segmentación, (desviación de la media). |
| dilate_quantity | Cantidad de dilataciones que se aplican a los objetos luego de la segmentación. |
| minimum_area | Filtrado de área mínima (en píxeles), cualquier objeto por debajo de este valor será descartado. |
| pitch_mask | Máscara de la tribuna, para evitar detectar falsos positivos. |
| line_mask | Máscara de las líneas, para resolver los pequeños movimientos de la cámara. |

Cuadro D.2: parameters(1).segmentation

```
>> parameters(1).homography
ans =
matrix: [3x3 double]
camera_number: 1
model: [475x745x3 uint8]
```

| | |
|---|---|
| matrix | Matriz de homografía (3x3) de la correspondiente cámara. |
| camera_number | Identificador de la cámara. |
| model | Modelo a escala del plano de la cancha. |

Cuadro D.3: parameters(1).homography

### D.3.4. Category_classification

Parámetros para el proceso de clasificación de los objetos detectados.

```
>> parameters(1).category_classification
ans =
Bins: [8 8 4]
ClassNames: 5x1 cell
Histogramas: [1x1 struct]
MapaDeProbabilidadAPriori: [1x1 struct]
```

| | |
|---|---|
| Bins | Cantidad de bins utilizados para el cálculo del histograma. |
| ClassNames | Nombres de las 5 clases de elementos que se encuentran. |
| Histogramas | Histogramas de las 5 clases (modelos). |
| MapaDeProbabilidadAPriori | Mapa de probabilidad a priori, da valores altos en las áreas a los goleros y bajos en el resto de la cancha. |

Cuadro D.4: parameters(1).category_classification

### D.3.5. Media

Contiene el path del video para esa cámara.

### D.3.6. Tracking

Parámetros utilizados para el proceso de seguimiento utilizando el filtro de Kalman.

```
>> parameters(1).tracking
ans =
max_distance: 50
max_no_match: 5
field_of_view: [1x1 struct]
```

| | |
|---|---|
| max_distance | Distancia máxima para la asociación, "blob" - "Target". |
| max_no_match | Numero máximo de cuadros, para dar por terminado un objeto. |
| field_of_view | Campo visual interno y externo. Es utilizado para la creación y destrucción de los objetos cuando entran y salen de la escena. |

Cuadro D.5: parameters(1).tracking

### D.3.7. Params_batch

Contiene los parámetros necesarios del procesamiento en batch.

```
>> parameters(1).params_batch
ans =
cant_camaras: 6
cuadros: [3000 3000 3000 3000 3000 3000]
filename: [1x6 cell]
PathName: ['D:" D:" D:" D:" D:']
```

| | |
|---|---|
| cant_camaras | Cantidad de cámaras a procesar en el modo batch. |
| cuadros | Cantidad de cuadros a procesar por cámara. |
| filename | Nombre de los archivos de salida. |
| PathName | Ubicación de los archivos de salida. |

Cuadro D.6: parameters(1).tracking

## D.4. Parámetros procesamiento unificado

parametersMView.mat

Archivo que contiene los parámetros necesarios para la segunda etapa. Procesamiento unificado de las 6 cámaras.

**parametersMView =**

**XML: [1x6 struct]**
**Modelo: [475x745x3 uint8]**
**CantidadObjetosClase: [10 10 3 1 1]**
**UbralFramesParaMuerto: 80**
**UbralFramesParaMuertoINI: 5**
**UbralFramesParaVIVO: 30**
**UbralDistanciaMahalanobis: 20**
**AgeFactor: 10**
**LastObservationFactor: 20**
**CategoryUpdateFactor1: 0.9000**

**CategoryUpdateFactor2: 0.1000**

| | |
|---|---|
| XML | Contiene el path de los 6 archivos XML en formato VIPER. Esta es la salida del proceso anterior (procesamiento por cámara). |
| Modelo | Modelo a escala de cancha. |
| CantidadObjetosClase | Cantidad de objetos que se debería encontrar en cada clase. |
| UbralFramesParaMuerto | Cantidad de cuadros para dar por finalizado un objeto. |
| UbralFramesParaMuertoINI | Cantidad de cuadros para dar por finalizado un objeto que aun no está establecido. |
| UbralFramesParaVIVO | Cantidad de cuadros para dar por establecido un objeto. |
| UbralDistanciaMahalanobis | Umbral de distancia mínima para asociar medidas. |
| AgeFactor | Factor utilizado junto con LastObservationFactor para estimar la antigüedad de un objeto. |
| LastObservationFactor | Factor utilizado junto con AgeFactor para estimar la antigüedad de un objeto. |
| CategoryUpdateFactor1 | Peso utilizado junto CategoryUpdateFactor2 para balancear el peso de la estimación actual con respecto a las pasadas, en el cálculo de la categoría. |
| CategoryUpdateFactor2 | Peso utilizado junto CategoryUpdateFactor1 para balancear el peso de la estimación actual con respecto a las pasadas, en el cálculo de la categoría. |

Cuadro D.7: parametersMView.m

# Bibliografía


[1] virtualdub.org. URL http://www.virtualdub.org/. VirtualDub is a video capture/processing utility for 32-bit and 64-bit Windows platforms (98/ME/NT4/2000/XP/Vista/7), licensed under the GNU General Public License (GPL).

[2] IMPIRE AG. URL http://www.bundesliga-datenbank.de/en/homede.

[3] Amisco. URL http://www.sport-universal.com.

[4] Grewal Mohinder S. Andrews, Angus P. *Kalman Filtering Theory and Practice Using MATLAB*. 2008.

[5] Yiannis Demirisa Anthony Deardena y Oliver Graub. *Tracking Football Player Movement From a Single Moving Camera Using Particle Filters*. Department of Electrical and Electronic Engineering Imperial College London, Exhibition Road, London, SW7 2BT, 2006.

[6] Sony Cameras Broadcast y XDCAM Production. URL http://pro.sony.com/bbsc/ssr/cat-broadcastcameras/cat-xdcam/product-PMWEX3.

[7] Tsai Camera Calibration. URL http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/DIAS1/.

[8] M. Xu D. Thirde y J. Orwell. *Intelligent Distributed Video Surveillance Systems (Professional Applications of Computing)*, cap. 8, págs. 225–252. Institution of Engineering and Technology, 2006.

[9] Istituto di Studi sui Sistemi Intelligenti per lÁutomazione. Issia cnr. 2004. URL http://www.issia.cnr.it/htdocs%20nuovo/progetti/bari/soccerdataset.html.

[10] Ralph Johnson Erich Gamma, Richard Helm y John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1995.

[11] G. Jones J. Ren, J. Orwell y M. Xu. *A general framework for 3D soccer ball estimation and tracking.* 2004.

[12] Kizanaro. URL http://www.kizanaro.com/web/.

[13] Language y Media Processing Laboratory. Viper: The video performance evaluation resource. URL http://viper-toolkit.sourceforge.net/.

[14] Kingston University London. URL http://sceptre.king.ac.uk/sceptre/default.html.

[15] Paolo Spagnolo Marco Leo Tiziana D Orazio Nicola Mosca y Massimiliano Nitti. *A Background Modelling Algorithm for Motion Detection.*

[16] University of Maryland. Language and media processing laboratory. URL http://lamp.cfar.umd.edu/.

[17] Delen Dursun Olson, David L. *Tecnicas avanzadas de Data Mining.* ISBN 3540769161, 2008.

[18] OpenCV. URL http://opencv.willowgarage.com/wiki/.

[19] OpenFramework. URL http://www.openframeworks.cc/.

[20] Prozone. URL http://www.prozonesports.com.

[21] Fundación Julio Ricaldoni. URL http://www.ricaldoni.org.uy.

[22] Josephine Sullivan y Stefan Carlsson. *Tracking and Labelling of Interacting Multiple Targets.* Royal Institute of Technology, Stockholm, Sweden, 2006.

[23] Michael Swain y Dana Ballard. *Color Indexing*, cap. 7(1), págs. 11–32. Internacional Journal of Computer Vision, 1991.

[24] HTV-3 TAJAM. URL http://www.tajam.com.uy/. ALTA DEFINICION EN URUGUAY.

[25] VIS.TRACK. URL http://www.bundesliga-datenbank.de/en/tracking-system.

[26] Ming Xu y Tim Ellis. *Partial Observation vs. Blind Tracking through Occlusion*. Information Engineering Centre, London, EC1V 0HB, 2002.