

Interpretación y Análisis Automático de Imágenes de Partidos de Fútbol

Civile, Juan Pablo

Crespo, Álvaro

Ordano, Esteban

Instituto Tecnológico de Buenos Aires

Proyecto Final Ingeniería en Informática

Director: Dra. Juliana Gambini

1 de agosto de 2014

RESUMEN

El seguimiento de objetos en una secuencia de imágenes (video) puede ser aplicado a eventos deportivos. En esta aplicación el objetivo es detectar y seguir la posición de los jugadores y la pelota a medida que se desarrolla el juego, obteniendo información útil para distintas aplicaciones, como puede ser brindar soporte informático para los árbitros (detección automática de pases, goles, posiciones adelantadas, etc...), colaborar con el cuerpo técnico en el entrenamiento de los jugadores, el análisis y el estudio de las tácticas de un contricante, entre otros.

En el presente trabajo se propone un método de seguimiento de jugadores que puede utilizarse para cumplir algunos de los objetivos del problema, se analizan las dificultades de realizar el seguimiento de jugadores en tiempo real con reducida o nula supervisión humana a partir de una sola fuente de video que consiste en una cámara de alta resolución (HD) fija capaz de encuadrar todo el campo de juego.

En base al análisis de las dificultades, se selecciona una técnica de seguimiento y se detalla un algoritmo construido sobre ésta para la tarea específica de detección y seguimiento de jugadores en partidos de fútbol. Se presentan los resultados de aplicar este algoritmo, se comparan con otro método existente en la literatura y se explicita qué información se podría proveer para el análisis de imágenes de partidos de fútbol.

Índice general

Introducción	4
1. Estado del arte	6
1.1. Algoritmos de seguimiento	6
1.1.1. Predictores Lineales	6
1.1.2. Aprendizaje Local	7
1.1.3. Basado en Grafos	10
1.2. Homografía	17
1.2.1. Aplicación y Utilización	17
1.2.2. Estimación de una Homografía	18
1.2.3. Distorsión de la Lente de la Cámara	20
1.3. Análisis de Partidos de Fútbol	21
1.3.1. Análisis Utilizando 6 Cámaras	21
1.3.2. Análisis Utilizando 8 Cámaras	25
1.3.3. Análisis de Deportes con Múltiples Cámaras	26
1.3.4. Análisis de Deportes con Video Televisado	27
2. Descripción del Problema	30
2.1. Análisis en Tiempo Real	30
2.2. Supervisión del Usuario	30
2.3. Sistema de Cámaras	31
2.4. Dificultades	31
2.4.1. Corrección de la Perspectiva de la Cámara	31
2.4.2. Sistema de Cámaras	31
2.4.3. Complejidad del Análisis en Tiempo Real	32
2.4.4. Distorsión de la Lente	32
2.4.5. Oclusiones entre Jugadores	33
3. Descripción del Método	34
3.1. Contornos Activos	34
3.2. Implementación Numérica	35

3.3. Elección	36
3.3.1. Parámetros	37
3.4. Limitaciones	37
4. Solución Propuesta	39
4.1. Eliminación de Fondo	39
4.1.1. Tribuna y Publicidades	39
4.1.2. Substracción de Fondo por Valor de Energía	40
4.1.3. Eliminación de Líneas	40
4.1.4. Eliminación del Césped	41
4.2. Contornos activos	42
4.2.1. Características	42
4.2.2. Función de Probabilidad	44
5. Resultados	47
5.1. Aplicación	47
5.2. Material Utilizado	49
5.3. Evaluación del Método	52
5.4. Comparación con IFTrace	53
5.4.1. Evaluación de Comportamiento	55
6. Conclusiones	56
6.1. Trabajo Futuro	56
Apéndice A. Captura de Pantalla de Aplicación	60
Apéndice B. Videos Utilizados	61

Introducción

Dentro del campo del análisis y tratamiento de imágenes un problema que se estudia es el del reconocimiento y seguimiento de objetos en secuencias de imágenes. Identificar correctamente un objeto en un video es útil para un amplio rango de aplicaciones, como ser aplicaciones médicas (donde se ofrece soporte para diagnósticos y cirugías), la industria cinematográfica (captura de movimiento, *post-producción*), vigilancia automática con cámaras y juegos interactivos. También tiene una gran influencia en el reconocimiento de caras y el seguimiento de ojos, técnicas que actualmente son muy utilizadas y estudiadas, y que tienen una gran potencialidad para ser implementadas en nuevos campos.

En el ámbito deportivo existen muchas aplicaciones para el seguimiento de objetos. Puede resultar útil para validar o reemplazar las decisiones de los jueces o árbitros del partido, permitir a deportistas de alto rendimiento analizar y mejorar sus movimientos, ayudar a los entrenadores a decidir estrategias y evaluar a los deportistas, otorgar estadísticas a fanáticos del juego, entre otras aplicaciones.

El seguimiento de objetos en video consiste en detectar objetos de interés en el primer cuadro de un video y seguirlo a lo largo de toda la secuencia de imágenes. Existen varios tipos de métodos que se pueden utilizar para llevar a cabo esta tarea, los cuales se diferencian en la forma en que representan a los objetos a seguir o en la manera en que localizan a cada objeto en una imagen de la secuencia. Por ejemplo, existen métodos de seguimiento que representan a los objetos de interés por marcas en la imagen y realizan el seguimiento de las mismas, asociando las marcas entre dos cuadros consecutivos. El método utilizado en este trabajo representa los objetos de interés por medio de curvas que representan los contornos de los objetos a seguir y realiza el seguimiento modificando estas curvas cuadro a cuadro, intercambiando puntos entre dos listas de píxeles vecinos.

En este trabajo se estudia el problema de realizar el seguimiento de todos los jugadores de un partido de fútbol, utilizando una única cámara, con su posición fija y cuyo campo visual abarca todo el campo de juego. Se busca que el seguimiento sea en tiempo real, es decir, que el procesamiento sea lo suficientemente rápido para que se vean los resultados mientras se ejecuta el video. Teniendo en cuenta la posición a través del tiempo, se puede informar la velocidad de un jugador en un determinado momento, distancia recorrida, zonas más frecuentadas por el jugador, entre otros.

Se plantea el uso de una técnica de seguimiento existente, basada en una técnica denominada *contornos activos* (ver [1]), para el seguimiento en tiempo real de los jugadores con una única computadora. Esta técnica determina el contorno de un objeto dentro del video y a medida que el objeto se mueve el contorno se adapta tomando decisiones basadas en variaciones de color de los píxeles de los objetos de interés y de sus alrededores. Este trabajo propone alternativas para mejorar el seguimiento especializán-

dolo a las características de un video de fútbol.

El trabajo se encuentra compuesto de la siguiente manera: en el Capítulo 1 se describe el estado del arte actual en el área de seguimiento de objetos, y en particular se examinan soluciones aplicadas al análisis automático de eventos deportivos. En el Capítulo 2 se detalla el problema a solucionar y se discuten las dificultades del seguimiento automático. En el Capítulo 3 se resume el marco teórico y el funcionamiento del algoritmo de seguimiento utilizado. El contenido del Capítulo 4 describe la solución planteada, estrategias que otorgaron mejores resultados, así como variaciones que fueron implementadas y posteriormente descartadas por no mejorar los resultados. El Capítulo 5 muestra los resultados obtenidos al ejecutar el algoritmo en videos filmados en partidos de fútbol profesional de clubes argentinos, y ofrece una comparación de la performance de la solución propuesta con otro algoritmo de seguimiento existente que no fue optimizado para seguimiento de jugadores de fútbol. Para finalizar, en el Capítulo 6 se presentan las conclusiones del trabajo.

Capítulo 1

Estado del arte

1.1. Algoritmos de seguimiento

A continuación se describen los algoritmos de tres familias dentro de la rama de análisis de imágenes para seguimiento de objetos.

1.1.1. Predictores Lineales

Un *Predictor Lineal* es una función que toma los valores de una serie de variables aleatorias y predice el valor de una variable dependiente. La función tiene la forma $f(x_1, \dots, x_n) = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n$, donde x_i es una variable aleatoria y β_i una constante. Se dice que el predictor es lineal debido a la condición $\beta_i \in \mathbb{R}$ ¹.

En los artículos [2, 3] los autores utilizan predictores para hacer seguimiento de objetos en video. El objetivo es relacionar el cambio de valor de varios píxeles cuadro a cuadro con el movimiento del objeto entre dichos cuadros. Para soportar cualquier clase de movimiento, este se representa mediante una homografía (ver [4]), $H \in \mathbb{R}^{3 \times 3}$.

Los predictores se representan mediante una matriz $A \in \mathbb{R}^{9 \times n}$ donde cada fila representa una función de predicción. De dos cuadros consecutivos se obtiene el vector de cambios de los valores de píxeles $X = (x_1, \dots, x_n)$ y se computa:

$$\begin{aligned} AX &= B \\ B &= (h_{1,1}, h_{1,2}, h_{1,3}, h_{2,1}, h_{2,2}, h_{2,3}, h_{3,1}, h_{3,2}, h_{3,3}) \end{aligned}$$

Donde B contiene los valores de la homografía H de movimiento entre los cuadros.

¹Cualquier x_i no lineal, puede expresarse como $x'_i = g(x_i)$ y tratarse como lineal

Cálculo de los predictores

Para obtener la relación A se utiliza un paso inicial de entrenamiento. Primero se determina la posición del objeto en un cuadro de manera supervisada. Una vez conocida la posición, se toman n muestras aleatorias del valor de píxeles en ese cuadro.

Luego se crean m perturbaciones de la posición del objeto, y se computa la homografía H_i (para $i = 1, 2, \dots, m$) que representa la perturbación. Se denomina B_i al vector que representa cada homografía. Además, se toman muestras de valores de píxeles (X_i) con sus posiciones alteradas por la homografía H_i . Se plantea:

$$A(X_1 | \dots | X_m) = (B_1 | \dots | B_m) \quad (1.1)$$

El cual es un sistema de ecuaciones lineales que puede ser resuelto de manera sencilla.

Mejoras

Para mejorar la efectividad de los algoritmos, se calculan varios predictores y se disponen en capas. Cada capa se construye para buscar cambios de posición cada vez más finos o pequeños. Es decir, la primer capa busca cambios bruscos y la última pequeños cambios.

Holzer et al. (ver [2]) introducen una nueva técnica para manejo de oclusiones parciales de manera eficiente. Para esto se divide el objeto en pequeñas secciones cuadradas, llamadas *templates*, y se computa la matriz A de la siguiente manera:

$$\begin{aligned} Y &= (B_1 | \dots | B_m) \\ H &= (X_1 | \dots | X_m) \\ A &= YH^T(HH^T)^{-1} \end{aligned}$$

Esto permite actualizar A rápidamente para remover y agregar *templates*. Es decir, permite ignorar secciones ocultas del objeto hasta que éstas vuelven a ser visibles.

1.1.2. Aprendizaje Local

El aprendizaje local es otra técnica que permite resolver el problema del seguimiento de objetos en secuencias de imágenes, que ha sido también un tema de investigación en el área del aprendizaje automático (ver [5]) y usado principalmente en la clasificación de imágenes, recuperación y reconocimiento de objetos.

A diferencia del aprendizaje global, que entrena un modelo basado en todos los datos de entrenamiento, en *local learning* se consideran varios modelos locales, cada uno extraído de solamente un subconjunto de los datos de entrenamiento. Este tipo de enfoque enfatiza la idea de que un modelo local puede caracterizar mejor las propiedades intrínsecas y discriminativas de su correspondiente subconjunto de datos, que un modelo global para el conjunto completo de datos. De esto se desprende que

utilizar múltiples modelos locales puede ofrecer mejores resultados cuando los datos están distribuidos de manera complicada o poco clara.

Si bien se puede pensar al seguimiento de objetos como una tarea de clasificación binaria que apunta a separar el objeto del resto de la imagen o fondo, en realidad, el principal problema reside en relacionar las apariciones del objeto entre dos cuadros consecutivos. Esto se logra mediante una función de distancia que establece el grado de la similitud entre los puntos característicos ² de cada cuadro.

La elección de la medida de distancia es de suma importancia. A modo de ejemplo, la distancia Euclídeana puede llevar a algoritmos de seguimiento inestables a la hora de diferenciar el objeto del fondo de la imagen. Por lo tanto, se requiere una mejor y más compleja medida de distancia. Esta es la idea principal que proponen Li y Lu (ver [6]).

La función de distancia que usan es una combinación lineal de distancias elementales, como se presenta en [7]. Definen a la función de distancia entre dos puntos característicos, a los que llaman *ejemplares*, e y z como:

$$D_e(z) = w_e \cdot d_{ez} \quad (1.2)$$

donde w_e es el vector de pesos de e , y d_{ez} es el vector de distancia n -dimensional entre e y z , cuya n -ésima componente es la distancia L_2 entre la característica n -ésima de e y z . Cabe destacar que la función del vector de pesos, w_e , es asignar una importancia relativa a la distancia de cada componente de d_{ez} . De esta forma, se ponderan las distancias entre ciertas características por sobre otras.

Cada ejemplar está también asociado con un vector binario B_e , cuyos elementos no nulos implican que los ejemplares correspondientes son similares a e . La longitud de B_e es igual al número de ejemplares con el mismo rótulo de e ³. Asumiendo que el aprendizaje de cada una de las funciones de distancia es independiente del resto, se puede aprender w_e y B_e , en un problema de aprendizaje formulado de la siguiente manera:

$$f_1(w, B) = \sum_{i \in C} B_i L(-w \cdot d_i) + \sum_{i \notin C} L(w \cdot d_i) \quad (1.3)$$

$$w^*, B^* = \arg \min_{w, b} f_1(w, b) \quad (1.4)$$

$$w \geq 0, B_i \in \{0, 1\}, \sum_i B_i = M \quad (1.5)$$

Nótese que se descarta el subíndice e para tener una mayor claridad. El conjunto C es el conjunto de todos los ejemplares con el mismo rótulo que e y M es el mínimo número de ejemplares similares a e (predefinido). La función L puede ser cualquier función de costo o pérdida ⁴, estrictamente positiva. El

²Los puntos característicos son puntos que se diferencian del resto por poseer o mostrar ciertas características específicas y representativas. En el caso del seguimiento de objetos, son los puntos que mejor representan las características del objeto, y que tienen mayor resistencia a variar con los diferentes cambios de escala, rotación, iluminación, etc...

³Se dice que un conjunto de ejemplares tiene un mismo rótulo si pertenecen a un mismo objeto. En el caso de seguimiento de objetos más sencillo, existirían 2 objetos: el objeto a seguir, y el fondo

⁴Una función de pérdida o función de costo es simplemente una función que mapea un evento o los valores de una o más variables a un número real que representa algún "costo" asociado al evento.

vector de pesos w se requiere que sea positivo para asegurar que una mayor distancia elemental (entre alguna de las características) no pueda nunca llevar a una menor distancia total, lo que implicaría una mayor similitud.

Previo al seguimiento se deben especificar manualmente algunos parámetros iniciales, como ser: punto central, ancho, altura y rotación del objeto. En el proceso de entrenamiento, se aplica un Análisis de la Componente Principal ⁵ de forma incremental en los primeros F cuadros. En su trabajo, Li y Lu usan $F = 5$ (ver [6]).

Los mejores resultados se seleccionan como muestras de entrenamiento positivas, mientras que los peores resultados, se toman como muestras negativas. Se obtienen las características RGB y LBP, como se presentan en [8], de todas las muestras, tanto positivas como negativas. Para cada muestra de entrenamiento, se calculan las distancias elementales RGB y LBP con respecto a todas las restantes muestras.

Para obtener el vector de pesos óptimo, w^* , se calcula iterativamente B , en función de w , y w en función de B , asegurando que el valor de f_1 nunca crezca, para encontrar un mínimo local. Este proceso iterativo se puede modelar de la siguiente forma:

$$B^k = \arg \min_B \sum_{i \in C} B_i L(-w^k \cdot d_i) \quad (1.6)$$

$$w^{k+1} = \arg \min_w \sum_{i: B_i^k=1} L(-w \cdot d_i) + \sum_{i \notin C} L(w \cdot d_i) \quad (1.7)$$

Dado un w^k , que inicialmente podría ser aleatorio, se minimiza la ecuación 1.6 fijando $B_i = 1$ para los M valores más pequeños de $L(-w \cdot d_i)$ y 0 para el resto. Dado B^k , se puede obtener fácilmente w^k , resolviendo la ecuación 1.7. El aprendizaje termina cuando $B^{k+1} = B^k$.

En el proceso de seguimiento, para cada cuadro nuevo, se determinan cuáles son los candidatos a ser ejemplares. En el área del aprendizaje automático, siempre se seleccionan candidatos para luego verificar si cumplen las condiciones requeridas. En este caso, se quiere ver si los candidatos cumplen con las condiciones necesarias para ser puntos característicos. Para ello, se seleccionan aleatoriamente aplicando un filtro de partículas ⁶ sobre el resultado del cuadro anterior. Luego, se extraen las características RGB y LBP como muestras de testeo. Después de calcular las distancias elementales entre las muestras de testeo y de entrenamiento, se forma una matriz de distancia, $D \in R^{N_{test} \times N_{train}}$ a través de las funciones de distancia entrenadas. Es decir, el elemento (i, j) de la matriz D , contiene la distancia entre la muestra de testeo i , y la muestra de entrenamiento j .

Finalmente, se utiliza la matriz D para localizar al objeto como sigue:

⁵El Análisis de la Componente Principal, es una técnica utilizada para reducir la dimensionalidad de un conjunto de datos. Sirve para hallar las causas de la variabilidad de los datos y ordenarlas por importancia. Técnicamente, busca la proyección según la cual los datos queden mejor representados en términos de cuadrados mínimos. Involucra el cálculo de la descomposición en autovalores de la matriz de covarianza, normalmente tras centrar los datos en la media de cada atributo.

⁶El filtro de partículas es un método empleado para estimar el estado de un sistema que cambia a lo largo del tiempo. Más concretamente es un método de Montecarlo(secuencial). Se compone de un conjunto de muestras (las partículas) y valores, o pesos, asociados a cada una de esas muestras. Las partículas son estados posibles del proceso, que se pueden representar como puntos en el espacio de estados de dicho proceso.

$$T = \arg \min_t c \sum_i D_i(t) + (1 - c) \sum_j D_j(t) \quad (1.8)$$

$$\forall i \in \{i : i \in S^+, D_i(t) \leq Thr_D\} \quad (1.9)$$

$$\forall j \in \{j : j \in S^+, D_j(t) > Thr_D\} \quad (1.10)$$

Donde t es un candidato y T es el objeto. S^+ es el conjunto de muestras de entrenamiento positivas. La idea consiste en que las muestras cuyas distancias sean menor que el umbral Thr_D contribuyan más a localizar el objeto, entonces la constante c debería tener un valor acorde, $c > 0,5$. Los autores Li y Lu utilizaron $c = 0,7$ (ver [6]).

El algoritmo de seguimiento se complementa con una idea más, que contribuye a atacar el problema de los cambios de poses y las oclusiones. Para esto es necesario actualizar el modelo que se tiene para que efectivamente pueda manejar estas dificultades. Haciendo uso del umbral de distancia Thr_D para prevenir malas actualizaciones, se realizan correcciones al modelo, en este caso, las funciones de distancia. Esto es, una distancia menor a Thr_D indica que el candidato pertenece a la misma clase, mientras que una distancia mayor indica lo contrario. Utilizando la ecuación 1.11, se agregan candidatos, como muestras positivas de entrenamiento y cada 5 cuadros, se utiliza el conjunto actualizado de entrenamiento para reentrenar todas las funciones de distancia.

$$t_{label} = \begin{cases} 1, & D_{S^+}(t) \leq Thr_D \\ 0, & D_{S^+}(t) > Thr_D \end{cases} \quad (1.11)$$

$$D_{S^+}(t) = \frac{1}{N_{S^+}} \sum_{i \in S^+} D_i(t) \quad (1.12)$$

En la ecuación 1.12, N_{S^+} es el número de muestras positivas de entrenamiento y $D_{S^+}(t)$ es el promedio de distancia del candidato t a las muestras de entrenamiento positivas.

1.1.3. Basado en Grafos

El algoritmo conocido como *IFTrace* (ver [9]) tiene como objetivo seguir un objeto en una secuencia de imágenes. Este algoritmo realiza algunas asunciones sobre el objeto a seguir:

- consiste de una o más regiones conexas,
- tiene un borde bien definido
- sus propiedades intrínsecas pueden variar con el tiempo.

Los objetos a seguir son inicialmente marcados de forma interactiva por el usuario en el primer cuadro, y luego se seleccionan automáticamente varios marcadores en el interior y los alrededores del objeto. Estos marcadores se localizan en el siguiente cuadro utilizando en forma conjunta el algoritmo de seguimiento de características KLT (ver [10]) y extrapolación de movimiento. Los bordes del objeto

son entonces identificados a partir de estos marcadores por la IFT, *Image Foresting Transform* (ver [11]). Otra característica central del algoritmo es el operador de detección de borde que se adapta gradualmente a cambios en el color y la textura del objeto.

El método IFT trata a la imagen como un grafo, en el cual los píxeles son los nodos, y estos están conectados mediante aristas si son adyacentes y tiene un peso o costo. Este costo debe ser una medida de la probabilidad de que los puntos estén separados por el borde del objeto de interés. En problemas sencillos puede ser tan simple como el valor absoluto de la diferencia de color entre los píxeles. El método combina el gradiente del color con el gradiente de una función de clasificación difusa (*fuzzy matching*), que consiste en cuantificar la similitud entre el píxel y el conjunto de píxeles del objeto segmentado en el frame anterior.

La IFT encuentra caminos de costo mínimo desde los marcadores, tanto internos como externos, a cada píxel de la imagen. Los píxeles son agrupados en particiones, que resultan ser árboles de camino óptimo disjuntos,⁷ donde cada árbol tiene a uno de los marcadores como raíz. La proyección del objeto es entonces tomada como la unión de los árboles cuyas raíces son marcadores internos.

El algoritmo IFTrace

Algoritmo 1 IFTrace

Input: Video $I : D \times \{1..n_f\} \rightarrow V$
donde $D = \{1..n_x\} \times \{1..n_y\}$
Mascara Binaria $O^{(1)} : D \rightarrow \{0,1\}$
Output: Mascaras Binarias $O^{(t)} : D \rightarrow \{0,1\}$ para $t \in \{2..n_f\}$

```

for  $t = 2, 3, \dots, n_f$  do
   $(R_o^{(t-1)}, R_x^{(t-1)}) \leftarrow \text{SelectMarkers}(I^{(t-1)}, O^{(t-1)})$ 
   $(S_o^{(t)}, S_x^{(t)}) \leftarrow \text{TrackMarkers}(R_o^{(t-1)}, R_x^{(t-1)}, I^{(t-1)}, I^{(t)})$ 
  if  $S_o^{(t)} \neq \emptyset$  then
     $O^{(t)} \leftarrow \text{IFTSegment}(I^{(t)}, C^{(t-1)}, S_o^{(t)}, S_x^{(t)})$ 
     $C^{(t)} \leftarrow \text{BuildClassifier}(I^{(t)}, O^{(t)})$ 
  else
     $C^{(t)} \leftarrow C^{(t-1)}$ 
     $(O^{(t)}, C^{(t)}) \leftarrow \text{RecoverObj}(I, t, O, C)$ 
return  $O$ 

```

El algoritmo 1 muestra el algoritmo de IFTrace desde el más alto nivel. IFTrace toma como input un video I , modelado como un array 3D de píxeles, siendo las dimensiones el alto y ancho de la imagen, y número de cuadro de la secuencia de video, y una máscara binaria $O^{(1)}$, correspondiente al objeto marcado en el primer cuadro, y devuelve un array de máscaras binarias $O^{(t)}$, con las proyecciones del objeto para cada cuadro del video.

⁷Dado un grafo simple, no dirigido y conexo G , un árbol de camino óptimo, o árbol de camino más corto, es un árbol recubridor de G , tal que la distancia o costo del camino desde la raíz v a cualquier otro vértice u es la distancia de camino mínima de v a u en G .

Para cada cuadro de la secuencia, el algoritmo elige dos conjuntos de puntos o píxeles: los que pertenecen a la máscara del objeto R_o , y los que pertenecen a sus alrededores R_x . Luego se procede a localizarlos en el siguiente cuadro a través del procedimiento *trackMarkers*.

Si el conjunto de píxeles que representan al objeto $S_0^{(t)}$ no resulta vacío para este nuevo cuadro, entonces el seguimiento fue exitoso. A continuación, se utiliza el procedimiento *IFTSegment* para obtener la máscara del objeto para el cuadro actual, denominado $O(t)$. Este procedimiento toma los conjuntos de píxeles pertenecientes al objeto $S_0^{(t)}$ y a sus alrededores $S_x^{(t)}$, el clasificador de color del cuadro anterior $C^{(t-1)}$ y el cuadro actual $I(t)$.

Como paso final, se obtiene el clasificador de color del cuadro actual $C(t)$, a partir de la máscara del objeto $O(t)$ y el cuadro actual $I(t)$.

Por otro lado, si el seguimiento del objeto falla para el nuevo cuadro, es decir si el conjunto de píxeles resultantes luego de aplicar *trackMarkers* $S_0^{(t)}$ resulta vacío, entonces se mantiene el mismo clasificador de color del cuadro anterior $C(t-1)$, y se procede a intentar recuperar el objeto en el cuadro actual mediante *recoverObj*. Si la recuperación es exitosa, se obtiene una máscara del objeto $O(t)$, y un clasificador de color para el cuadro actual $C(t)$, sino, todos los conjuntos se dejan vacíos y el clasificador se mantiene inalterado.

Algoritmo RecoverObj

Algoritmo 2 RecoverObj - Intento de recuperar un objeto perdido de vista

Input: Video I , índice de cuadro t , tope cuenta hacia atrás m_f , máscaras del objeto O^k y clasificadores de colores C^k para los k cuadros previos.

Output: Mascara del objeto recuperada O^t (puede estar vacia) y su correspondiente clasificador de color C^t

```

for  $k = t - 1, t - 2, \dots, \max 1, t - m_f$  do
  if  $O^k$  no esta vacio then
     $M \leftarrow \text{CandidateMask}(I^{(t)}, C^{(k)})$ 
    Sea  $\kappa$  la lista de regiones conexas en  $M$ 
    for region  $K$  en  $\kappa$  do
       $(S_o, S_x \leftarrow \text{SelectMarkers}(I^{(k)}, K)$ 
       $K' \leftarrow \text{IFTSegment}(I^{(t)}, C^{(k)}, S_o, S_x)$ 
      if  $K'$  es suficientemente similar a  $O^k$  then
         $C' \leftarrow \text{BuildClassifier}(I^{(1)}, O^{(1)}, D O^{(1)})$ 
        return  $(K', C')$ 
return  $(\emptyset, \emptyset)$ 

```

Si en algún paso del algoritmo IFTrace, el seguimiento del objeto falla, se procede a utilizar la heurística *recoverObj* (Algoritmo 2), que sirve para localizar una región conexa cuya forma y colores sea similar a la del objeto en alguno de los cuadros anteriores.

La heurística se aplica utilizando un cuadro de referencia, e iterando hacia atrás hasta encontrar un cuadro en el que se pueda recuperar el objeto, evitando utilizar cuadros previos en los que no se pudo

recuperar el objeto anteriormente.

Una vez ubicado el cuadro de referencia en el cual el seguimiento fue exitoso, se construye una “máscara candidata” M que indica las posibles ubicaciones del objeto en el cuadro actual t . Esto es lo que hace el procedimiento *CandidateMask*. En este paso se utiliza el clasificador de color del cuadro de referencia k , $C^{(k)}$. La máscara etiqueta los píxeles del cuadro como “posiblemente objeto”(1) o “probablemente fondo”(0).

Esta máscara M estará compuesta de cero o más regiones conexas. Si el objeto está visible en el cuadro actual, su proyección debería coincidir con alguna de estas regiones. Entonces se procede a analizar cada una de estas regiones, para las cuales se obtienen los dos conjuntos de marcadores $S_0^{(t)}$ y $S_x^{(t)}$, utilizando la misma técnica que en el algoritmo 1.

Luego, para cada región K , se utiliza nuevamente el algoritmo IFT para conseguir una proyección K' de la región y compararla con la máscara del objeto en el cuadro k . En caso de ser similares, se construye el clasificador de color C' y se retorna como resultado, junto a la proyección K' que es la máscara representativa del objeto para el actual cuadro.

Cabe destacar que, por simplicidad, para la comparación de formas se utilizan las Invariantes de Momento de Maitra (ver [12]), y las formas se consideran similares si la distancia Euclideana entre sus vectores de invariantes es menor a 2. Sin embargo se podrían utilizar otros descriptores de formas.

Si ninguna de las regiones resulta similar a la máscara del objeto, *recoverObj* intenta el mismo procedimiento utilizando como cuadro de referencia el cuadro anterior. Luego de un número especificado de intentos, o de agotar todos los cuadros, la heurística termina. En este caso, se retorna una máscara vacía y un clasificador nulo, señalando que el objeto se perdió en el cuadro actual. *IFTrace* intentará entonces recuperar en el siguiente cuadro.

Selección de Marcadores

El procedimiento *selectMarkers* es el encargado de elegir dos conjuntos de marcadores $R_0^{(t)}$ y $R_x^{(t)}$, los que están dentro de la máscara del objeto, y los que están a su alrededor, respectivamente.

Los marcadores internos se eligen aplicando una erosión morfológica ⁸ a la máscara del objeto, con radio δ_o , y luego seleccionando los píxeles de la región resultante que tengan mayor probabilidad de ser seguidos por el algoritmo KLT. Específicamente, se construye la matriz:

$$H[p] = \begin{bmatrix} \sum_q (\frac{\partial L}{\partial x}[q])^2 & \sum_q (\frac{\partial L}{\partial x}[q])(\frac{\partial L}{\partial y}[q]) \\ \sum_q (\frac{\partial L}{\partial x}[q])(\frac{\partial L}{\partial y}[q]) & \sum_q (\frac{\partial L}{\partial y}[q])^2 \end{bmatrix} \quad (1.13)$$

donde $L[p]$ es la luminancia del píxel p , y las sumatorias incluyen a todos los píxeles q en una ventana de 9×9 centrada en p .

Se define el grado de “similaridad” $\lambda[p]$ como el valor mínimo singular de $H[p]$. Solo los píxeles con $\lambda > 1$ son escogidos para utilizarlos en la propagación KLT. Esto coincide con los autores Tomasi y

⁸La erosión morfológica es una de las dos operaciones fundamentales de la matemática morfológica. Consiste en reducir o “erosionar” los bordes de un determinado objeto, reduciendo su tamaño.

Karade (ver [10]) quienes afirman que los píxeles con mayor valor de λ tienen más posibilidades de ser localizados por el algoritmo KLT.

Los marcadores externos son escogidos aplicando una dilatación morfológica⁹ a la máscara del objeto 0, con radio δ_x , y tomando los píxeles a lo largo del borde del objeto.

Seguimiento de Marcadores

El procedimiento *trackMarkers* se utiliza para localizar los marcadores internos en el cuadro actual, que se corresponden con los puntos internos del objeto en el cuadro anterior. Utiliza el algoritmo de seguimiento KLT, descrito por Tomasi y Karade (ver [10]).

El algoritmo KLT recibe un punto p en un cuadro I^{t-1} , una posición estimada q_0 en el próximo cuadro I^t , y busca un punto q tal que los vecinos de q en I^t sean similares a los de p en I^{t-1} . Este algoritmo tiene varios parámetros de configuración que alteran su comportamiento: ℓ , la cantidad escalas consideradas; κ , el factor de reducción entre las sucesivas escalas; y ω , el ancho de la ventana usada para comparar vecinos en cada escala. La implementación de IFTrace está configurada para usar $\ell = 2, \kappa = 4$ y $\omega = 9$, como en la implementación de KLT de Birchfield (ver [13]).

Los marcadores exteriores no son seguidos con KLT, ya que no tiene sentido debido a que la mayoría se perdería por oclusión con el objeto o se seguiría el fondo, en vez del objeto. En cambio, lo que se hace es trasladarlos de acuerdo a los desplazamientos medios de los marcadores internos más cercanos. Esto tiende a mantener estos marcadores fuera del objeto, pero cerca de su borde, aún cuando hay rápidos cambios de tamaño o de forma (por ejemplo, rotaciones o movimiento de extremidades).

Segmentación de objetos - IFT

La IFT interpreta a la imagen I como un grafo G , cuyos nodos son los píxeles y cuyas aristas unen dos nodos si los píxeles que representan son adyacentes. IFT toma como parámetros un conjunto de marcadores (píxeles) S , una función de costo de arista w , y una función de conectividad f que asigna un costo de camino $f(\pi)$ a todo camino π en G , dependiendo del costo de sus aristas.

Para cada píxel p , IFT encuentra un camino directo óptimo (de costo mínimo), que lo conecta con su raíz $R(p)$, perteneciente al conjunto S de marcadores. Estos caminos forman un bosque de caminos óptimos. Cada árbol en este bosque de caminos óptimos tiene como raíz a algún marcador r , y agrupa a todos los píxeles de la imagen para los cuales existe un camino a r de costo menor a cualquier otro camino a otro marcador en S . IFT también le asigna a cada píxel p un costo $V(p) = \pi(p)$, de valor igual al costo del camino mínimo para llegar a p partiendo desde su raíz, y un marcador raíz $R(p)$, el marcador que se encuentra en la raíz del árbol al cual pertenece.

IFTrace depende de la IFT para segmentar los objetos a seguir. Usa la función de conectividad f_{max} , que asigna a un camino π el máximo costo para todas las aristas en π . Los píxeles marcadores utilizados son tanto los marcadores interiores, obtenidos por el algoritmo KLT, como los exteriores que se ubican alrededor del objeto.

⁹La dilatación morfológica es otra de las operaciones básicas de la matemática morfológica. Generalmente utiliza un elemento estructurador para expandir alguna forma contenida en la imagen.

La segmentación obtenida al usar f_{max} tiene una importante propiedad *minimax*. Sea la *frontera* del objeto de interés el conjunto de todas las aristas cuyos nodos pertenecen a distintos segmentos, dicha segmentación maximiza el costo mínimo de todas las aristas en la frontera de la segmentación. Esto significa que la segmentación IFT/ f_{max} es esencialmente la misma que la famosa segmentación *divisoria* (ver [11]). Esto hace que sea particularmente apropiada para casos en los que el objeto está mejor caracterizado por su conectividad que por su forma o tamaño. Es por esto que se elige una función de costo de arista $w(p, q)$ que se basa en la probabilidad de que p y q estén en diferentes lados del borde del objeto.

Algoritmo IFT

Para algunas funciones de conectividad, como f_{max} , el bosque de caminos óptimos se puede computar eficientemente mediante una variante del famoso algoritmo de Dijkstra (ver [11]). Por cuestiones de eficiencia, el algoritmo retorna un mapa de raíces R , que almacena, para cada nodo p , su raíz $R(p)$.

Algoritmo 3 Algoritmo IFT con f_{max}

Input: Grafo G_1 , conjunto de semillas $S = S_o \cup S_x$
Output: Bosque de camino optimo P , mapa de conectividad V
y mapa de raíz R .
Auxiliary: Cola de prioridades Q , variable tmp

```

for  $p \in G_1$  do
     $P(p) \leftarrow \text{nil}$ ,  $R(p) \leftarrow p$ ,  $V(p) \leftarrow +\infty$ 
    if  $p \in S$  then insertar  $p$  en  $Q$  y setear  $V(p) \leftarrow 0$ 
while  $Q \neq \emptyset$  do
    Remover  $p$  de  $Q$  tal que  $V(p)$  sea minimo.
    for  $q$  4-vecino de  $p$ , tal que  $V(q) > V(p)$  do
        Computar  $tmp \leftarrow \max\{V(p), w(p, q)\}$ 
        if  $tmp < V(q)$  then
            if  $V(q) \neq +\infty$  then remover  $q$  de  $Q$ 
             $P(q) \leftarrow p$ ,  $R(q) \leftarrow R(p)$ ,  $V(q) \leftarrow tmp$ 
            Insertar  $q$  en  $Q$ 

```

Se agregan los nodos que representan marcadores a la cola Q . El ciclo *while* principal computa un camino óptimo desde las raíces a cada nodo p . En cada iteración, un camino de valor $V(p)$ mínimo se encuentra cuando removemos el último píxel p de la cola. Luego se evalúa si el camino que alcanza el píxel adyacente q a través de p es más barato que el actual camino que termina en q y se actualizan Q , $V(q)$, $R(q)$ y $P(q)$.

Este algoritmo se puede modificar para recomputar el bosque de camino óptimo en forma incremental, a medida que se van agregando o quitando marcadores, generalmente en tiempo sub-lineal.

Comparación con Corte de Grafos

Boykov and Funka-Lea [5,12] han propuesto otra alternativa para la segmentación de imágenes basada en grafos, conocida como Corte de Grafos. En este enfoque, el grafo de píxeles se modela como una red de flujo, donde el objeto es la fuente y el fondo el sumidero. Se puede probar que el máximo flujo total de todas las fuentes hacia todos los sumideros es igual a la mínima capacidad total de todo corte (conjunto de aristas) que separa fuentes de sumideros. Este flujo máximo y su corte mínimo asociado se puede encontrar con el clásico algoritmo Ford Fulkerson (ver [14]).

En comparación con la IFT, *Corte de Grafos* tiene 2 grandes desventajas: tiene un mayor costo computacional (IFT es $O(n)$ mientras que el mejor algoritmo *Corte de Grafos* es $O(n^{2.5})$), y además tiende a minimizar el número de aristas en el corte, en vez de sus capacidades. Esto último quiere decir que muestra una preferencia a segmentar basada en la longitud del borde, en lugar de basarse en la importancia de la conexión entre el objeto y el fondo (ver [15]).

Se puede introducir una corrección al algoritmo de *Corte de Grafos* para remediar este segundo problema, pero lo único que se obtiene es el mismo resultado de la IFT, con un costo computacional mucho mayor (ver [15]).

Costo de arista de IFT

Como se explicó anteriormente, IFT opera con una función de costo de arista w , la cual resulta crítica para la segmentación. Idealmente, el costo para aristas que cruzan el borde del objeto debe ser alto, y bajo para todo el resto. En otras palabras, la función de costo debe ser un detector de bordes del objeto.

Un ejemplo puede ser la distancia Euclideana entre los colores RGB de los píxeles en ambos extremos de una arista. Desafortunadamente, este detector resulta demasiado simplista para casos prácticos, en los que un objeto puede tener diferentes colores y texturas. Por lo tanto, se requieren detectores de bordes más sofisticados.

En la implementación de *IFTrace*, se utiliza un detector de bordes basado en una combinación lineal

$$w(p, q) = \gamma w_f(p, q) + (1 - \gamma) w_o(p, q) = \gamma |\nabla I| + (1 - \gamma) |\nabla M| \quad (1.14)$$

donde γ es un parámetro definido por el usuario, ∇I es el gradiente del color de la imagen y M es un mapa de clasificación de color.

La primer componente $w_f(p, q)$ es la distancia Euclideana entre los colores de los extremos de la arista, como se menciona anteriormente pero con una particularidad: se mide en el espacio de colores *Lab* de *CIE*¹⁰ en lugar del tradicional RGB. La justificación para esto es que las distancias entre colores se aprecian más de esta forma.

La segunda componente $w_o(p, q)$, es el gradiente del mapa de clasificación de colores M , una imagen en escala de grises donde cada píxel $M[p]$ es la probabilidad de que un píxel con color $v = I[p]$

¹⁰Un espacio de colores *Lab* es un espacio de colores de 3 dimensiones, L para la luminosidad o claridad, a para la posición entre rojo y verde y b para su posición entre amarillo y azul. Existen dos espacios muy similares *Hunter Lab* y *CIE Lab*, que se diferencian en la forma de calcular las coordenadas de color a partir de los datos: el primero utiliza raíces cuadradas y el segundo raíces cúbicas.

pertenezca a la proyección del objeto. Los colores que ocurran solo dentro del objeto deberían estar asociados al valor 1, los que solo ocurran en el fondo deberían estar asociados al 0, y los que puedan ocurrir en ambos, deberían mapearse a valores intermedios. El propósito de incluir este término es resaltarle importancia a los bordes entre colores que son internos del objeto, o totalmente ajenos al objeto, y enfatizar los bordes que efectivamente delimitan el objeto del fondo de la imagen.

El mapa de clasificación de colores M se obtiene de la imagen I por medio de una función de clasificación difusa C , una función no lineal $C : \mathbb{V} \rightarrow [0, 1]$. En *IFTrace*, la función C se implementa como una variante del clasificador del vecino más cercano, *nearest neighbor* (NN) y determina dos conjuntos de colores \mathbb{U}_0 y \mathbb{U}_x , que se asumen son representativos del objeto de interés y del fondo, respectivamente. Para evaluar $C(v)$ para cierto color v , se debe buscar 2 colores representantes $u_0 \in \mathbb{U}_0$ y $u_x \in \mathbb{U}_x$ que son los más cercanos a v . La probabilidad de que un píxel de color v sea parte del objeto es entonces estimada por la fórmula

$$C(v) = \frac{|v - u_0|}{|v - u_0| + |v - u_x|} \quad (1.15)$$

En teoría, se podría usar todos los píxeles para construir los conjuntos \mathbb{U}_0 y \mathbb{U}_x . Pero en la práctica, se deben usar muestras de menor tamaño por razones de eficiencia, de otra forma, el costo de encontrar los representantes u_0, u_x aumenta demasiado. De hecho, la construcción del mapa M resulta ser el paso que tiene mayor costo computacional de *IFTrace*, más aún que el seguimiento de marcadores y que la segmentación de la IFT.

Es por esto que el procedimiento para construir el clasificador, *buildClassifier*, comienza seleccionando dos conjuntos de píxeles R_0, R_x , respectivamente los que están dentro y fuera de la actual máscara del objeto. Pero si alguno de estos conjuntos tiene más de 200 elementos, entonces se realiza un muestreo aleatorio para reducirlos a ese tamaño.

1.2. Homografía

1.2.1. Aplicación y Utilización

La técnica de la homografía se utiliza para poder transformar la vista tridimensional de la secuencia de imágenes (la imagen del estadio de fútbol), en una vista de dos dimensiones (el campo de fútbol visto desde arriba). Esto facilita el análisis y los cálculos de las velocidades y posiciones de los jugadores, lo cual permite sacar conclusiones y efectuar juicios complejos como en el caso de la regla del fuera de juego. Además, visualmente elimina ambigüedades que pueden suceder en el caso de la vista tridimensional por razones de perspectiva, lo cual facilita el entendimiento al ser humano.

Considérese dos imágenes, $f(x, y)$ y $f'(x, y)$, relacionadas por una transformación geométrica. Sean p_k y p'_k para $k = 0 \dots N$ puntos de las imágenes f y f' respectivamente, y dadas correspondencias tentativas $p_k \rightarrow p'_k$, se quiere estimar la transformación T , tal que

$$f(x, y) = f'(T(x, y)) \quad (1.16)$$

Esta transformación T se puede modelar como una transformación de coordenadas lineales

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = H \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \quad (1.17)$$

en donde H es una matriz de 3×3 que representa la proyección, rotación, escalamiento, sesgo y perspectiva.

Se puede observar que para aplicar H se extiende el vector de dos dimensiones $[x, y]^T$ a tres componentes. El valor de la tercera componente puede variar, y define una clase equivalencia tal que

$$\begin{bmatrix} zx \\ zy \\ z \end{bmatrix} \equiv \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (1.18)$$

Este tipo de coordenadas se denominan homogéneas.

La forma de H determina el tipo de transformación geométrica representada. Por ejemplo,

$$H = \begin{bmatrix} sa \cos(\theta) & -sb \sin(\theta) & t_x \\ sa \sin(\theta) & sb \cos(\theta) & t_y \\ p_0 & p_1 & 1 \end{bmatrix} \quad (1.19)$$

Representa una rotación de ángulo θ , una traslación dada por t_x y t_y (notesé el uso del “1” de coordenadas homogéneas), un escalamiento dado por el factor s , un sesgo introducido por a y b y un cambio de perspectiva dado por p_0 y p_1 .

En total, son 8 los parámetros que definen la matriz H , y sus elementos son

$$H = \begin{bmatrix} H_{00} & H_{01} & H_{02} \\ H_{10} & H_{11} & H_{12} \\ H_{20} & H_{21} & 1 \end{bmatrix} \quad (1.20)$$

1.2.2. Estimación de una Homografía

Se pueden estimar los 8 parámetros desconocidos de la matriz de transformación H , basándose en correspondencias de puntos conocidas. La transformación de una coordenada x en coordenadas homogéneas ($z = 1$), a una coordenada objetivo $x' = Hx$ da como resultado

$$\begin{aligned} x' &= H_{00}x + H_{01}y + H_{02} \\ y' &= H_{10}x + H_{11}y + H_{12} \\ z' &= H_{20}x + H_{21}y + H_{22} \end{aligned}$$

Dividiendo las primeras 2 ecuaciones por z' para convertirlas en coordenadas Euclidianas, se puede llegar a

$$\begin{aligned} \frac{x}{z'}(H_{20}x + H_{21}y + H_{22}) - H_{00}x - H_{01}y - H_{02} \\ \frac{x}{z'}(H_{20}x + H_{21}y + H_{22}) - H_{10}x - H_{11}y - H_{12} \end{aligned}$$

Teniendo varias correspondencias como la anterior, se las puede escribir en forma matricial:

$$Ah = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & \frac{x'x}{z'} & \frac{x'y}{z'} & \frac{x'}{z'} \\ 0 & 0 & 0 & -x & -y & -1 & \frac{y'x}{z'} & \frac{y'y}{z'} & \frac{y'}{z'} \\ & & & \vdots & & & & & \end{bmatrix} \begin{bmatrix} H_{00} \\ H_{01} \\ H_{02} \\ H_{10} \\ H_{11} \\ H_{12} \\ H_{20} \\ H_{21} \\ H_{22} \end{bmatrix} = 0 \quad (1.21)$$

Cada correspondencia de puntos agrega dos filas a la matriz A , por lo que n correspondencias generan una matriz de $2N \times 9$.

Como consecuencia, se tiene el siguiente sistema de ecuaciones lineales para resolver

$$Ah = 0 \quad h \neq 0 \quad (1.22)$$

Con 4 puntos de correspondencias, el sistema es *compatible determinado* y la única solución es el espacio nulo de A . Para más correspondencias, el sistema pasa a ser *compatible indeterminado* y de las infinitas soluciones, se busca la de menor norma 2.

$$\arg \min_{\|h\|=1} \|Ah\| = \arg \min_{\|h\|=1} h^T A^T A h = \lambda_{min} \quad (1.23)$$

donde λ_{min} es el menor autovalor de $A^T A$. Esto se desprende de las propiedades de la norma 2 de matrices y de la propiedad de las matrices cuadradas simétricas que dice que dado $B = A^T A$, λ_i y q_i sus autovalores y autovectores respectivamente, las matrices

$$Q = [q_0 q_1 \dots q_{n-1}] \quad (1.24)$$

y

$$D = \begin{bmatrix} \lambda_0 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_{n-1} \end{bmatrix} \quad (1.25)$$

cumplen $B = QDQ^T$.

$$\arg \min_{\|h\|=1} h^T QDQ^T h = \arg \min_{\|y\|=1} y^T D y = \arg \min_{\|y\|=1} \lambda_1 y_1^2 + \dots + \lambda_n y_n^2 \quad (1.26)$$

Con $\lambda_i = \lambda_{min}$, se alcanza un mínimo cuando todas las componentes de y se anulan excepto por $y_i = 1$. Como $y = Q^T h$, $h = Qy = q_{min}$, el autovector de B correspondiente al mínimo autovalor.

El problema se reduce entonces a hallar este autovector. Para ello, se recurre a la Descomposición en Valores Singulares

$$A = U\Sigma V^T \quad (1.27)$$

donde las columnas de U contienen los autovectores de AA^T y las columnas de V los autovectores de $A^T A$, mientras que Σ es una matriz diagonal con los autovalores de

$$A^T A$$

en su diagonal. Nótese que los autovalores de $A^T A$ son iguales a los autovalores de A al cuadrado.

La Descomposición en Valores Singulares se calcula de tal forma que los autovalores en la diagonal de Σ aparecen en orden decreciente. Por lo tanto se toma como solución la última columna de V , correspondiente al menor autovalor de $A^T A$.

1.2.3. Distorsión de la Lente de la Cámara

El material utilizado fue obtenido con una lente con una baja distancia focal, lo que genera una distorsión de tipo *barril*, donde las líneas rectas se curvan hacia el exterior. El modelo de *Brown* plantea una solución general para múltiples niveles de distorsión, tanto de tipo radial como tangencial.

Para obtener los valores de la imagen sin distorsión se utilizó la siguiente corrección:

$$\begin{aligned} x_u &= (x_d - x_c)(1 + Kr^2) \\ y_u &= (y_d - y_c)(1 + Kr^2) \end{aligned}$$

donde x_u y y_u son los puntos en la imagen sin distorsión, x_d y y_d son puntos en la imagen original, x_c y y_c son los puntos centrales de la imagen (se puede asumir que el punto central de la cámara es el punto central de la imagen), K es un coeficiente de distorsión radial, y $r = \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2}$, la distancia del punto al centro de la imagen.

1.3. Análisis de Partidos de Fútbol

1.3.1. Análisis Utilizando 6 Cámaras

D’Orazio et al. estudian la viabilidad de juzgar si en un partido de fútbol ocurrió una posición adelantada, utilizando seis cámaras dispuestas en ambos laterales de la cancha para reducir errores de medición y perspectiva. Las imágenes obtenidas por las cámaras son sincronizadas y procesadas para obtener la posición de cada jugador y de la pelota en todo momento, y a partir de estos datos detectar pases de pelota para finalmente detectar si se cometió una posición adelantada.

El sistema consiste de seis cámaras de alta resolución, tres en cada lado de la cancha, con sus ejes ópticos paralelos a las líneas de llegada de la cancha con el objetivo de reducir errores de perspectiva. Cada cámara envía sus imágenes a una de seis computadoras dispuestas con el objetivo de analizar las imágenes y detectar la posición de los jugadores, la pelota, y posibles pases entre jugadores de un mismo equipo. Una computadora central recibe estos datos y valida que las mediciones de distintas cámaras sean congruentes.

A continuación se presentan las técnicas de análisis de imagen aplicadas.

Extracción del fondo

Un algoritmo de eliminación de fondo basado en una técnica que analiza el nivel de energía (ver [17]) en una ventana temporal (una secuencia de cuadros consecutivos) para detectar aquellas áreas candidatas a contener a alguno de los jugadores, la pelota, el árbitro o los jueces de línea. Se analiza la conexidad de las áreas candidatas de acuerdo a su topología para descartar sombras que puedan llegar a generar errores en análisis posteriores.

Este análisis utiliza la técnica de “ventana deslizante”, ya que se basa en una cantidad de imágenes consecutivas de la secuencia (se referirá a esta subsecuencia como W y se la denominará “ventana temporal” o simplemente “ventana”). D’Orazio et al. utilizan una ventana de 60 cuadros, equivalente a 2,5 segundos, valor obtenido empíricamente y suficiente para determinar que los objetos en primer plano (principalmente los jugadores) no se mantenían durante tanto tiempo en posición completamente estática.

Inicialmente, la energía de un punto (x, y) en una imagen es definida por la ecuación:

$$E(x, y) = \sum_{t \in W} \|I^t(x, y) - B_C(x, y)\|^2 \quad (1.28)$$

donde $I^t(x, y)$ es la intensidad del punto en coordenadas (x, y) para la imagen t , y $B_C(x, y)$ es una estimación de la intensidad del fondo para el punto (x, y) . Para determinar B_C se aplican sucesivos filtros gaussianos al primer cuadro de W .

Para las siguientes ventanas se refina el modelo del fondo de la imagen. Se calcula la función B_F , definida para todo punto de un cuadro, cuya imagen es \mathbb{R}^2 y corresponde a la media μ y la desviación estándar σ de $E(x, y)$.

Se define B_F inicialmente mediante la ecuación:

$$B_F(x, y) = \begin{cases} \mu(x, y), \sigma(x, y) & \text{if } E(x, y) < th(W) \\ \phi & \text{if } E(x, y) > th(W) \end{cases} \quad (1.29)$$

donde $th(W)$ es un valor de umbral obtenido empíricamente, proporcional al tamaño de W . Un bajo nivel de energía corresponde a un punto estático, por lo tanto se considera que ese punto pertenece al fondo y B_F contendrá la media y la desviación estándar para ese punto. Si la imagen tiene un alto valor de energía, se considera al punto un candidato a contener un objeto en movimiento y por lo tanto el valor de B_F es indefinido.

Una vez que este primer B_F contiene un modelo que identifica zonas en primer plano y en el fondo, para sucesivas ventanas se utiliza esa información y se la complementa con nuevas observaciones, utilizando un parámetro de actualización β (D’Orazio et al. utilizaron un valor de 0,1). De este modo, luego de la primer aproximación a B_F , éste será calculado en base a su valor anterior y a los nuevos valores de energía ($E(x, y)$), utilizando la ecuación 1.30:

$$B_F(x, y) = \begin{cases} \mu(x, y), \sigma(x, y) & \text{if } E(x, y) < th(W) \wedge B_F(x, y) = \phi \\ \beta B_F(x, y) + (1 - \beta)\mu[\mu(x, y), \sigma(x, y)] & \text{if } E(x, y) < th(W) \wedge B_F(x, y) \neq \phi \\ \phi & \text{if } E(x, y) > th(W) \end{cases} \quad (1.30)$$

De esta manera, B_F contiene ahora información para determinar si un punto pertenece al primer plano o es un punto del fondo o estático. Un punto en la siguiente ventana temporal pertenece al primer plano si $B_F(x, y) = \phi$ o bien $E(x, y) - \mu(x, y) > N\sigma(x, y)$, es decir, la energía en ese punto difiere del valor esperado por más de N desviaciones estándar (los autores utilizaron un valor de 2 para N).

Por último, se realiza un análisis de conectividad de los puntos de primer plano y los conjuntos de puntos conexos son denominados sectores candidatos a ser personas en la cancha o la pelota. Luego, se descartan sectores cuyo tamaño no supere un umbral (determinado experimentalmente con el objetivo de descartar sectores muy chicos) y se realiza un análisis de la morfología de los conjuntos conexos, para eliminar la sombra proyectada por los jugadores.

Clasificación de Jugadores y Árbitros

Luego de identificar los sectores candidatos, éstos son clasificados para determinar a que clase pertenecen (jugadores o arquero de uno u otro equipo o árbitros). Dado que los colores de las camisetas de los jugadores no se conocen previamente, es necesario utilizar un mecanismo de clasificación no supervisado. Dicho mecanismo consiste de dos pasos. En primer lugar, se crean las clases de objetos con un algoritmo de *clustering* basado en una versión modificada del algoritmo BSAS (ver [18]). En segundo lugar, el clasificador obtenido es utilizado para determinar la clase de cada objeto candidato obtenido del análisis anterior.

El algoritmo BSAS sólo requiere para funcionar una medida de semejanza $d(x, C)$, y un valor de umbral denominado th para generar las clases de objetos. Se realizan varias iteraciones, unificando

clases de objetos si no difieren por más de th .

Para clasificar a los objetos en su clase correspondiente se toma la clase con menor *distancia Manhattan* al prototipo, y se actualiza la clase ganadora mediante la fórmula:

$$C_k = \frac{1}{w_k + 1}(w_k C_k + V)$$

donde C_k es el prototipo de la clase k , V es el vector de características asociadas a la clase k , y w_k es el número de objetos clasificados dentro de la clase k de acuerdo a las últimas dos ventanas temporales W .

Seguimiento de Jugadores

Se utiliza una técnica de seguimiento representando al área donde un jugador se encuentra mediante una “caja delimitadora”. Dos “cajas delimitadoras” pueden estar en estado de colisión si sus lados se intersectan. El vector de seguimiento de un jugador está definido por la tupla $x_{ti} = (p_i, v_i, d_i, l_i, c_i, s_i)$, donde:

- p_i, v_i, d_i son la posición, velocidad, y dimensiones (ancho y alto del caja) para el jugador i .
- s_i define el estado del seguimiento, para resolver colisiones entre cuadros y la aparición o desaparición de sectores candidatos en el área.
- c_i es la clase a la que pertenece el jugador.
- l_i es una etiqueta identificadora del jugador, o un conjunto de etiquetas si el cuadro está en estado de colisión.

Para cada nuevo cuadro, se actualiza la posición de la caja que contiene a los sectores candidatos analizando el movimiento de sectores dentro de la caja para actualizar la velocidad, posición y tamaño, respecto a otras cajas para actualizar su estado (si entró en contacto). Para resolver situaciones en la que se esté terminando la colisión, se utilizan datos correspondientes a la clase de los objetos detectados y la velocidad de cada jugador previo a la colisión de las cajas.

Para la detección de la posición de la pelota, se utiliza un clasificador entrenado con un conjunto de entrenamiento de imágenes de pelotas en distintas posiciones y de distintos tamaños. Se elige el candidato a ser la pelota a aquél sector cuyo coeficiente de correlación con el clasificador sea el mayor.

La velocidad de la pelota es determinada a partir de las observaciones de la cámara por la ecuación:

$$V_x = \frac{(P_{x_t} - P_{x_{t-n}})}{n}, V_y = \frac{(P_{y_t} - P_{y_{t-n}})}{n}$$

donde P_{x_t}, P_{y_t} es la posición de la pelota en el cuadro t , y n es la cantidad de cuadros entre la imagen actual y la última ubicación correcta de la pelota.

Sin embargo, para que una posición candidata para la pelota sea seleccionada como un verdadero positivo, debe recaudarse información sobre más de un cuadro. Para esto, se genera un mapa de probabilidades de que la pelota se encuentre en cada punto de la imagen, basado en observaciones de cuadros

anteriores. Esta posibilidad está definida como sigue:

$$P(x, y) = \exp \left[-\left(|x - |\tilde{x} + V_x \text{sign}(\cos \theta)| + (y - |\tilde{y} + V_y \text{sign}(\sin \theta)|)^2 / 2\theta^2 \right) / \sigma \sqrt{2\pi} \right] \quad (1.31)$$

donde \tilde{x}, \tilde{y} es la última posición conocida de la pelota, y

$$\sigma = \frac{R_p V_{max} n}{R_{cm} T}$$

donde V_x y V_y representan la velocidad de la pelota en las coordenadas x e y , θ es $\arctan(\frac{V_y}{V_x})$, R_p es el radio de la pelota en píxeles, V_{max} es el máximo valor admitido para la velocidad de la pelota, R_{cm} es el radio de la pelota en centímetros, T es la cantidad de cuadros por segundo, y n la cantidad de cuadros desde que la pelota fue encontrada en la posición \tilde{x}, \tilde{y} .

En sucesivos cuadros, se utiliza esta información para evitar buscar la pelota en todo el campo de juego, y sólo se analizan los píxeles con alta probabilidad de contener la pelota (la probabilidad disminuye exponencialmente con la distancia según la ecuación 1.31).

Detección de Pases y Posición Adelantada

Los datos analizados en una computadora por cada cámara son entonces transmitidos a un servidor (llamado “supervisor”) que unifica la información recibida de cada cámara, considera sus posiciones relativas, y arma el estado del juego, compuesto por el tiempo desde el inicio del partido, la posición de todos los jugadores, el árbitro, jueces de línea, sus respectivas velocidades y estimación de la aceleración, si la pelota fue detectada o no, la posición y la confianza en la posición de la pelota. Los jugadores son posicionados a través de una transformación homográfica y el supervisor correlaciona los datos provenientes de distintas cámaras para formar un estado consolidado.

Para la detección de una jugada de posición adelantada, se requiere:

- *Determinar qué jugador pateó la pelota:* esta es la tarea más compleja del análisis. Se requiere determinar la posición de la pelota en tres dimensiones. Para lograr esto, se detecta la posición de puntos conocidos del campo de juego en la cámara y se utiliza la homografía para calcular la posición estimada de la pelota. Se hace uso también de la posición relativa de las cámaras en la cancha: dado que están enfrentadas, las mediciones de la posición de la pelota de dos cámaras enfrentadas debería ser similar. Al detectar un cambio repentino en la velocidad de la pelota, se estima que el jugador más cercano a la pelota es el que la pateó.
- *Determinar si es una posición adelantada activa:* Se analiza la posición de jugadores en estado de “posición adelantada pasiva”. De las reglas del juego, si un jugador está adelantado pero no recibe la pelota, la jugada puede seguir y no se ha cometido una falta. Durante los tres segundos posteriores a un pase largo (tres segundos es una estimación del tiempo que la pelota está en el aire desde que es pateada hasta que es recibida) el supervisor evalúa si los jugadores en posición adelantada interceptan la pelota, en cuyo caso, se anuncia que el jugador está en falta.

1.3.2. Análisis Utilizando 8 Cámaras

Xu et al. proponen un sistema que consta de 8 módulos, conectados a cámaras, que recolectan y envían información a un módulo supervisor, encargado del mantenimiento de la posición de los jugadores. El sistema supervisor no tiene acceso a los datos crudos de la imagen obtenida, sino que reciben información procesada de cada nodo.

Pre-procesamiento del Video

Cada nodo realiza un preprocesamiento de la imagen para estimar la posición de la cámara y detectar áreas de interés (candidatas a ser los jugadores en pantalla). El primer paso es la eliminación del fondo. Para ello, se utilizan dos máscaras, una geométrica, que aproxima la geometría de la cancha, y otra que extrae información acerca de los píxeles y genera un histograma para detectar el color verde del pasto de la cancha. Para la primera máscara, se pasa la imagen del espacio de colores *RGB* a *HSI* y se analiza el histograma de *hue* para los valores de intensidad. Luego, los píxeles que serán considerados son aquellos que no pertenezcan a un determinado rango de *hue*. Luego, la máscara que elimina valores de acuerdo a su color queda definida por:

$$M_c = (\{(u, v) | H(u, v) \in [H_l, H_h]\} \oplus B) \ominus B$$

Donde $H(u, v)$ es el valor de *hue* del píxel en la posición (u, v) , B es un elemento estructurador cuadrado y \oplus y \ominus son los operadores morfológicos de erosión y dilatación, aplicados para separar a los jugadores y eliminar las líneas de campo blancas. La máscara geométrica comprende los siguientes puntos:

$$M_g = \{(u, v) | E(u, v) \in P\}$$

Donde P es el rango de coordenadas en el mundo real correspondiente a la cancha y $E(u, v)$ es la correspondencia del punto (u, v) en el mundo real, de acuerdo a una corrección utilizando ángulos de Euler. Por último, la máscara que determina aquellos píxeles que corresponden al pasto de la cancha son los que pertenecen a la intersección de ambas máscaras:

$$M = M_c \cap M_g$$

El segundo paso consiste en un proceso de seguimiento local. Cada nodo determina pequeñas “cajas delimitadoras” alrededor de las áreas donde es posible que haya jugadores. Estas cajas son representadas por la posición en el plano de la imagen \mathbf{x}_l y su error en la medición \mathbf{z}_l en un filtro de *Kalman* (ver [20]):

$$\mathbf{x}_l = [r_c \ c_c \ \dot{r}_c \ \dot{c}_c \ \Delta r_1 \ \Delta c_1 \ \Delta r_2 \ \Delta c_2]^T$$

$$\mathbf{z}_l = [r_c \ c_c \ r_1 \ c_1 \ r_2 \ c_2]^T$$

Donde $r_1 < r_2$ y $c_1 < c_2$ son los límites de la caja y r_c y c_c sus centroides. Estos valores son actualizados cuadro por cuadro, asumiendo que la variación en alto y ancho es baja. Se utilizan ángulos de Euler para traducir estos valores a la posición esperada dentro del plano de la cancha. La varianza es estimada utilizando el jacobiano de la matriz de Euler.

Por último, se analiza la categoría de los jugadores de acuerdo al histograma de colores, clasificando de acuerdo a los cinco posibles uniformes (uno para cada equipo, uno para cada arquero, y los árbitros).

Seguimiento multi-cámara

El módulo supervisor recibe los datos pre-procesados de los demás módulos. Se unifican los valores obtenidos de todos los módulos y se los asocia utilizando una matriz que es actualizada de acuerdo a la distancia de Mahalanobis ¹¹. Si esta distancia queda por debajo de un determinado umbral, se empieza a considerar que las mediciones de dos cámaras distintas para dos “cajas delimitadoras” pasan a ser el mismo jugador.

1.3.3. Análisis de Deportes con Múltiples Cámaras

Beetz et al. analizan, desde el punto de vista de la inteligencia artificial, el problema de detectar el movimiento de los jugadores dentro de la cancha. Para eso, proponen un sistema con dos módulos, uno que extrae características e información de videos provenientes de múltiples cámaras, y un módulo de análisis y seguimiento del comportamiento de cada objeto. Los objetos de interés son los jugadores, la pelota y los árbitros.

Encuentran una gran dificultad en detectar la posición de jugadores más alejados de la cámara, dado que el material con el que trabajan en general son cámaras situadas a no más de 18 metros de altura. Esto genera una imagen borrosa de los jugadores que están en el otro extremo de la cancha, dificultando el análisis.

El primer módulo se encarga de procesar los cuadros de video y producir información de la posición de las regiones que potencialmente contienen los objetos de interés. Se utiliza la segmentación por colores como herramienta principal de detección. Se conoce de manera supervisada las clases de colores que representan el césped, los jugadores de un equipo, los jugadores del otro equipo, los árbitros, las líneas de la cancha y la pelota. Con esta información, es posible armar filtros complejos a la hora de buscar una clase de objeto particular. Por ejemplo, para encontrar jugadores, se toma la región que queda de eliminar los árbitros, las líneas, el césped y la pelota, y de esto, se toman las partes que cumplen con la representación por colores de los jugadores. Para evitar problemas de iluminación, cada cierto número de cuadros se re-aprende la clase de color que corresponde al verde del césped.

Para detectar las regiones de interés, se utilizan las clases de color para aislar potenciales regiones. A partir de estas, se utilizan las características morfológicas de cada clase de objeto para eliminar regiones inválidas. Se asume que los jugadores siempre van a estar parados, y por lo tanto tienen regiones

¹¹La distancia de Mahalanobis es una medida de distancia que sirve para determinar la similitud entre dos variables aleatorias multidimensionales. Se diferencia de la distancia euclídea en que tiene en cuenta la correlación entre las variables aleatorias y posee invariancia de la escala.

rectangulares. De la misma manera, se espera que la región que contiene a la pelota sea circular. Nótese que no se espera que los resultados obtenidos en este paso sean finales. La información producida por este módulo es procesada por el segundo módulo para producir los resultados finales de seguimiento.

Finalmente, la última tarea asignada al primer módulo es la de estimar los parámetros de la cámara. Esto es necesario para poder asignar una ubicación en el plano de la cancha a un objeto representado por una región en el plano de la imagen. De manera supervisada y previa a la ejecución del algoritmo, se genera un modelo de la cancha. Este modelo incluye líneas de la cancha, arcos y los carteles de propaganda. Luego, durante la ejecución se aplica un algoritmo iterativo de 3 pasos:

1. Proyectar el modelo sobre la cancha
2. Utilizar algoritmos de detección de líneas para calcular el error de la proyección
3. Ajustar los parámetros para minimizar el error de proyección

Como existe la posibilidad de que no haya suficiente información en un cuadro para estimar utilizando el método anterior, se utiliza además un algoritmo de extracción de características y seguimiento. En particular, la implementación de Shi y Tomasi (ver [22]).

La información de regiones y sus posiciones es entregada al segundo módulo. Este utiliza el algoritmo **Multiple Hypothesis Tracking (MHT)** (ver [23, 24]) mejorado por Schmitt et al. (ver [26]). Este algoritmo utiliza un modelo de cámara y movimiento probabilístico para estimar la posición de los jugadores y actualizarlas en cada observación. El algoritmo fue adaptado para tener en cuenta algunas restricciones encontradas en el fútbol:

- Todos los objetos de interés se encuentran en el mismo plano
- Los objetos de interés solo pueden desaparecer de la imagen por los bordes

Una vez obtenida la posición de cada objeto de interés, se puede hacer un análisis del estado del partido. Sólo se observa al jugador que tiene la pelota en un dado cuadro. Para este jugador se construyen eventos de movimiento. Un evento de movimiento es un cambio de posición del jugador tal que puede ser representado por una función lineal. Por lo tanto, se termina describiendo el movimiento de un jugador a lo largo de una jugada, de manera aproximada, como una serie de pequeñas líneas rectas.

Con estos eventos de movimiento, se construye un episodio. Esto es la serie de eventos de movimiento que realiza un jugador desde que recibe la pelota en un cuadro t_i , hasta que la pierde en un cuadro t_f . Como no siempre se conoce con precisión la posición de la pelota, se ingresa de manera supervisada el momento en el que un jugador toma contacto con la pelota, y cuando lo pierde.

Finalmente, se intenta clasificar los episodios según su resultado: un pase, un tiro al arco, una evasión o pérdida de pelota. Para esto se utiliza un árbol de clasificaciones, con reglas simples para los primeros 3 casos. En el caso de la pérdida de pelota, se utiliza un árbol de decisión más complejo.

1.3.4. Análisis de Deportes con Video Televisado

En [27] se plantea una técnica para obtener las posiciones de los jugadores y la pelota utilizando la transmisión televisada de un partido de fútbol. El algoritmo se puede dividir en las siguientes etapas:

- Estimación de la relación entre puntos en la imagen y coordenadas en la cancha
- Estimación de la posición de la cámara en las coordenadas del mundo
- Detección y seguimiento de la pelota
- Detección de la cancha
- Detección de los jugadores

Para estimar la relación entre los puntos en la imagen y las coordenadas en la cancha se calcula una homografía. Como la perspectiva de la cámara cambia cuadro a cuadro, ésta debe ser calculada en cada cuadro nuevamente. Si en la imagen actual se encuentran 4 puntos conocidos (esquinas de la cancha o de las áreas), la homografía puede ser calculada directamente. De no ser así, se estima considerando que la homografía del cuadro actual H_t se puede relacionar con H_{t-1} mediante un *Global Motion Parameter* P . Este parámetro $P_{t-1,t}$ es una homografía que se calcula tomando una serie de puntos en la imagen y relacionando su posición entre el cuadro $t - 1$ y t . Luego $H_t = H_{t-1}P_{t-1,t}$. Se utiliza el método *KLT* (ver [10]) para encontrar los puntos de referencia a seguir para el cálculo de P .

Para obtener la posición de la cámara se descompone la homografía en 2 matrices:

$$H = K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix}$$

$$K = \begin{pmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

Donde α, β representan la amplitud focal de la cámara, γ representa el sesgo y (u_0, v_0) es la coordenada del punto principal. Se asume que la amplitud focal se mantiene constante, $\gamma = 0$ y el punto principal es el centro de la imagen. r_1, r_2 son parte de una rotación $R = (r_1, r_2, r_1 \times r_2)$, y t son las coordenadas del origen de la cancha en coordenadas de la imagen. Conocidas H y K se puede calcular la posición de la cámara como $R^{-1}t$. Para el cálculo de K se necesitan α y β que se calcula usando P de la siguiente manera:

$$\begin{bmatrix} p_{11}p_{21} & p_{12}p_{22} \\ p_{11}p_{31} & p_{12}p_{32} \\ p_{11}p_{31} & p_{22}p_{32} \end{bmatrix} \begin{bmatrix} \alpha^2 \\ \beta^2 \end{bmatrix} = \begin{bmatrix} -p_{13}p_{23} \\ -p_{13}p_{33} \\ -p_{23}p_{33} \end{bmatrix} \quad (1.32)$$

Seguimiento de la pelota

El algoritmo planteado para seguimiento de la pelota tiene 2 etapas. La primer etapa es de detección y utiliza un algoritmo basado en *Viterbi*. La segunda etapa es de seguimiento y utiliza un filtro de *Kalman* (ver [20]).

El algoritmo de la primer etapa, crea una imagen binaria, segmentando según la característica blanca de la pelota. Una vez segmentada, se eliminan los candidatos teniendo en cuenta las distintas características morfológicas de la pelota. Esta operación se realiza en T cuadros distintos, y a partir de esto se construye un grafo pesado como se explica a continuación. En este grafo los nodos representan posibles candidatos de la posición de la pelota en los T cuadros. Cada nodo tiene asignado un peso según el grado de similitud entre el candidato y una pelota ideal. Luego, se coloca un vértice entre un nodo del cuadro t y otro del cuadro $t + 1$ si son parecidos en características y cercanos en posición. Sobre este grafo se hace una búsqueda de camino óptimo. Los nodos que forman parte del camino óptimo se consideran entonces como la pelota real.

Una vez detectada la pelota, se utiliza un filtro de *Kalman*. Cuando el filtro devuelve una posición con un margen de error mayor a un cierto umbral, se considera pérdida. En esa instancia se vuelve a detectar la pelota con el algoritmo de detección detallado anteriormente.

Detección de la cancha

Para detectar la cancha y aislar el resto de los objetos, se computa un histograma de colores de la imagen. El color de la cancha representa el mayor pico en el histograma. Por lo tanto, se busca este pico en el histograma y se toman todos los colores adyacentes en el histograma que esten dentro de un rango de ocurrencia relativo al pico principal. Todo píxel cuyo valor caiga dentro de ese rango se considera parte del fondo.

Detección de jugadores

Una vez que se determina el rango de colores de la cancha, se construye una imagen binaria separando la cancha de todo otro objeto. Construida esta imagen, se aplica la misma técnica que se usa para seguir la pelota, adaptada a la morfología de los jugadores. Una vez que se conoce la posición en el cuadro t de los jugadores, utilizando la homografía H_t se puede calcular la posición en la cancha de los jugadores.

Capítulo 2

Descripción del Problema

Este trabajo busca ofrecer una solución para obtener de manera semi-supervisada información del estado del juego de fútbol en tiempo real. Esto involucra conocer la posición de cada jugador para cada cuadro del video, la posición de la pelota, cuál es el puntaje actual, entre otros. Para el trabajo actual, se acota el problema únicamente a la determinación de la posición de los jugadores.

Soluciones más avanzadas podrían otorgar información con un alto nivel de abstracción, si se conocen datos como la posición de la pelota. Por ejemplo, se pueden determinar automáticamente casos en los que los jugadores cometen una falta por estar fuera de juego (denominado en inglés una falta por *off-side*), lo cual requiere, al menos, la posición de cada jugador (y a qué equipo pertenece), la posición de la pelota, y detectar cuándo un jugador realiza un pase. Este último punto requiere un análisis de alto nivel que es descrito en [16]. Se discuten en el Capítulo 4 las dificultades encontradas para replicar estos resultados.

2.1. Análisis en Tiempo Real

Se plantea que el algoritmo debe poder realizar el seguimiento en tiempo real, tomando ventaja de que contornos activos (la técnica utilizada para obtener la posición de los jugadores) es lo suficientemente eficiente para funcionar en tiempo real.

Realizar el seguimiento en tiempo real agrega restricciones al problema, debido a que el algoritmo debe ser lo suficientemente eficiente para procesar la imagen en una fracción de tiempo. Toda técnica tiene un costo de procesamiento, por lo tanto se debe tener cautela al momento de seleccionar qué procesos de análisis de imagen pueden ser realizados en los aproximadamente 40 milisegundos que separan un cuadro de otro al mostrar un video de 24 cuadros por segundo.

2.2. Supervisión del Usuario

Respecto a la supervisión necesaria, se busca extraer automáticamente información sobre un partido que un operador (o grupo de operadores) podría obtener del video; así como complementar esto con más información proveniente de un análisis que sólo se podría lograr mediante un cómputo automático.

En el presente trabajo, un supervisor selecciona inicialmente las posiciones de los jugadores en la cancha. El algoritmo de contornos activos es luego ejecutado para obtener la posición de cada jugador para los cuadros siguientes. El supervisor debe corregir actualizaciones incorrectas de contornos activos. Es deseable que estas falsas detecciones sean mínimas o inexistentes.

2.3. Sistema de Cámaras

El trabajo se enfoca en imágenes obtenidas de un sistema de una única cámara fija, posicionada en la cancha de forma tal que todo el campo de juego entra dentro del cuadro de la cámara. Al utilizar una única cámara, la resolución adquiere un rol determinante, dificultando o impidiendo el uso de muchas de las técnicas de seguimiento.

2.4. Dificultades

Bajo las suposiciones mencionadas, esta Sección describe los problemas que deben ser solucionados para obtener datos precisos respecto a la posición de los jugadores.

2.4.1. Corrección de la Perspectiva de la Cámara

La imagen capturada por la cámara es una representación en 2 dimensiones de la realidad. El modelo de datos de este trabajo representa cada jugador y la pelota como un punto en un campo de dos dimensiones, es decir, se descarta el valor de la altura de cada objeto seguido, ya que esa información no es de interés para este estudio. Para esto, se aplica una homografía para convertir las coordenadas de un punto de la imagen a coordenadas en el plano donde se encuentra la cancha.

El cálculo de una homografía involucra reconocer por lo menos 4 puntos de la cancha en la imagen (ver [4]). Esto puede hacerse de manera supervisada, seleccionando en la imagen proveniente del video distintos puntos y luego correspondiéndolos con su posición en la cancha en dos dimensiones. También puede lograrse de manera automática mediante un algoritmo de detección de líneas que permita comparar las líneas en la imagen con las que se encuentran en la cancha.

Ignorar el valor de altura de los objetos seguidos podría eventualmente causar errores en la medición de la posición y velocidad de la pelota (ver [27]). Sin embargo, dado que la posición de los jugadores en altura no varía notablemente durante el partido, es una buena aproximación a la hora de determinar las posiciones de los jugadores proyectadas sobre el plano del campo de juego.

2.4.2. Sistema de Cámaras

Un sistema de múltiples cámaras se ve beneficiado por una mejor resolución (se obtiene una mejor precisión al determinar la posición de objetos), pero requiere un sistema de sincronización que coordine la obtención de información.

Por otro lado, un sistema constituido por una única cámara no tiene la complejidad extra que implica la sincronización de la información de las diferentes cámaras, pero sufre de una menor resolución y

menor precisión.

Esta reducción de resolución puede tornarse prohibitiva para algunas técnicas o algoritmos de seguimiento ya que algunos objetos (como por ejemplo la pelota) tendrán unos pocos píxeles en cada imagen, lo cual dificulta la tarea de segmentación y seguimiento al contar con una imagen de peor calidad. En particular la detección de texturas se torna más difícil. Esto puede observarse en las imágenes de las figuras 2.1 y 2.2.

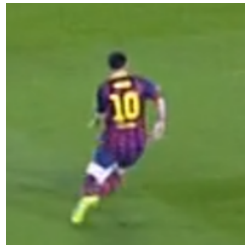


Figura 2.1: En la imagen se observa la falta de resolución. Los bordes se ven difusos.

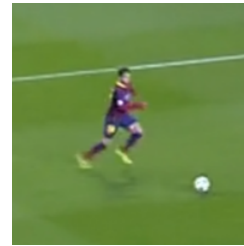


Figura 2.2: Se observan dos de las dificultades del problema: la baja resolución y el diminuto tamaño de la pelota con respecto al jugador.

2.4.3. Complejidad del Análisis en Tiempo Real

Al tener una resolución de *1080p* (aproximadamente dos millones de píxeles por cuadro), el procesamiento de cada píxel debe tomar a lo sumo 20 nanosegundos. Para lidiar con esta restricción se puede utilizar información adicional de la que se disponga respecto al video con el objeto de evitar procesar píxeles de poca o nula utilidad para el seguimiento. Un ejemplo de esto es descartar píxeles que estén fuera de la cancha, ya que es probable que la cámara encuadre más que el campo de juego, abarcando las gradas, el público espectador, publicidades alrededor del campo de juego, entre otros.

Muchos autores han desarrollado algoritmos automáticos de seguimiento de objetos en secuencias de imágenes (ver [9, 2, 6, 24]). Todos ellos están basados en soluciones de ecuaciones diferenciales en derivadas parciales y proveen resultados precisos, pero tienen severas restricciones que impiden que se utilicen para aplicaciones en tiempo real.

Para este trabajo, se utiliza el algoritmo de contornos activos (ver [1]), el cual no utiliza ecuaciones diferenciales (haciéndolo apto para aplicaciones en tiempo real) y además hace un análisis local de los objetos seguidos en la imagen, lo cual hace que el tiempo de análisis de un cuadro sea dependiente de la cantidad de píxeles que abarque la silueta de un jugador e independiente del tamaño de cada cuadro del video.

2.4.4. Distorsión de la Lente

Al utilizar una única cámara para captar la cancha entera se corre el riesgo de que los puntos de la imagen más alejados al foco de la cámara sufran una distorsión. Este es el llamado “efecto de ojo de buey” e introduce mucho error, por ejemplo en la aplicación de la homografía, por lo tanto se debe aplicar una corrección. Una lente apropiada y bien calibrada puede reducir este error, pero nunca puede

ser eliminado totalmente. La Figura 2.3 muestra un cuadro con el efecto de distorsión y la Figura 2.4 muestra el mismo cuadro, pero con la corrección por software aplicada.



Figura 2.3: Imagen afectada por distorsión de lente



Figura 2.4: Misma imagen, con una corrección por software aplicada

2.4.5. Oclusiones entre Jugadores

En un partido es muy común que los jugadores se crucen entre sí, ocultándose parcial o totalmente de la cámara, y por lo tanto ocurran oclusiones entre los jugadores. El sistema debe poder tolerar la oclusión parcial o total de los jugadores. Esto puede llevar a situaciones muy difíciles de detectar automáticamente. Por ejemplo, una situación particularmente difícil de resolver se da cuando dos jugadores del mismo equipo (con vestimenta muy similar) se encuentran alineados con respecto a la cámara. Una posible solución es utilizar información de cuadros anteriores para estimar la velocidad de cada uno y estimar sus nuevas posiciones, pero esto es poco efectivo si los jugadores cambian de velocidad mientras uno ocluye al otro, o si la velocidad era muy similar al momento de generarse la oclusión.

Una situación problemática similar es una jugada de tiro de esquina (córner), donde las oclusiones entre varios jugadores son muy numerosas, lo que agrega a la restricción de tiempo real mayor complejidad, ya que la resolución de oclusiones debe ser muy eficiente en tiempo.

Capítulo 3

Descripción del Método

En este Capítulo se describe el método de seguimiento seleccionado para este trabajo. En primer lugar, la Sección 3.1 explica el algoritmo utilizado (*Contornos Activos*), y en la Sección 3.2 su implementación. Luego, en la Sección 3.3 se justifica la elección del algoritmo. Finalmente, en la Sección 3.4 se analizan las limitaciones del algoritmo para esta aplicación.

3.1. Contornos Activos

La segmentación basada en contornos activos se basa en definir una región en base a su contorno. El contorno o borde de una región Ω_i se representa por una curva paramétrica dada por:

$$C_i(s) = (x_i(s), y_i(s)) \quad (3.1)$$

donde $0 \leq s \leq S_i$, y $C_i(0) = C_i(S_i)$, siendo S_i el total de puntos que conforman la curva. Es decir, C_i es una curva cerrada.

Se definen tantos contornos como objetos de interés haya en la secuencia (i es un entero entre 1 y el número de objetos). Adicionalmente, se considera una región Ω_0 que contiene a todo punto que no forma parte de ninguna otra región. Se la denomina *región de fondo*.

Se declara una función de probabilidad p que permite saber que tan probable es que un píxel forme parte de una dada región. Para esto, es necesaria una función v que dado un píxel devuelve un vector $v(x)$ de características, por ejemplo los valores RGB del píxel. Esto permite calcular $p(v(x)|\Omega_i)$, la probabilidad de que un píxel x forme parte de una región Ω_i . Estas funciones dependen de la secuencia particular a ser analizada, por lo que varían dependiendo del caso de estudio. Como ejemplo, se toman los valores RGB como vector de características y $p(v(x)|\Omega_i) = \|v(x) - v_i\|$, donde v_i es el valor promedio RGB de todo píxel en la región Ω_i .

Se define la función de energía de los contornos:

$$E = - \sum_{m=0}^M \int_{\Omega_m} \log p(v(x)|\Omega_m) dx + \lambda \int_{C_m} ds \quad (3.2)$$

Los algoritmos basados en contornos activos buscan minimizar esta ecuación. Si el valor de E es mínimo para un cuadro, se considera que se encontró la mejor aproximación de la región de los objetos interés. De 3.2 se deriva la ecuación de evolución de cualquier curva C_m :

$$\frac{dC_m}{dt} = (F_d + F_s) \vec{N}_{C_m} \quad (3.3)$$

$$F_d = \log p(v(x)|\Omega_m)/p(v(x)|\Omega_0) \quad (3.4)$$

$$F_s = \lambda \kappa_m \quad (3.5)$$

F_d y F_s son fuerzas derivadas de la ecuación de energía. F_d representa la competencia entre regiones y F_s una función que produce el efecto de suavizado. κ_m es la curvatura de la curva C_m .

3.2. Implementación Numérica

Se toma de referencia la implementación según Shi y Karl (ver [1]). En la implementación, el borde de un contorno C_m se representa usando dos conjuntos de píxeles correspondientes a los bordes interno y externo, L_{in} y L_{out} respectivamente. Entonces, la evolución se realiza intercambiando píxeles entre estos dos conjuntos.

A continuación se detalla la técnica para la segmentación de un solo objeto de interés que se puede fácilmente extrapolar y utilizar para la segmentación de múltiples objetos ¹.

El objeto de interés Ω_1 y el fondo Ω_0 cumplen $\Omega_1 \cup \Omega_0 = I_k$, donde I_k es la imagen del cuadro k de la secuencia de imágenes, y $\Omega_1 \cap \Omega_0 = \emptyset$. Cada una de las regiones está caracterizada por su vector característico $v_m, m = \{0, 1\}$.

Se define una función $\phi(x)$ que indica si un píxel x pertenece a una región o al fondo de la siguiente manera:

$$\phi(x) = \begin{cases} 3 & \text{si } x \in \Omega_0 \text{ y } x \notin L_{out} \\ 1 & \text{si } x \in L_{out} \\ -1 & \text{si } x \in L_{in} \\ -3 & \text{si } x \in \Omega_1 \text{ y } x \notin L_{in} \end{cases} \quad (3.6)$$

Los conjuntos L_{in} y L_{out} se definen como

$$L_{in} = \{x \text{ es un píxel} \mid \phi(x) < 0 \text{ y } \exists y \in N_4(x) \text{ de modo que } \phi(y) > 0\} \quad (3.7)$$

$$L_{out} = \{x \text{ es un píxel} \mid \phi(x) > 0 \text{ y } \exists y \in N_4(x) \text{ de modo que } \phi(y) < 0\} \quad (3.8)$$

donde $N_4(x) = \{y \text{ es un píxel} \mid |x - y| = 1\}$, son los píxeles vecinos del píxel x .

¹Para la segmentación de múltiples objetos basta con marcar las distintas regiones con un identificador distinto y seguir las por separado.

El algoritmo de segmentación se compone de dos etapas, ya que luego de la especificación inicial de la curva en forma supervisada ² se intercambian los píxeles de L_{in} y L_{out} en dos ciclos. En el primero, se aplica la fuerza $F_d(x)$, y en el segundo se aplica $F_s(x)$ para la regularización.

En el primer ciclo, se ejecutan los siguientes pasos N_a veces, donde $0 < N_a < \max(filas, columnas)$.

1. Para cada $x \in L_{out}$, si $F_d(x) > 0$ entonces borrar x de L_{out} y agregarlo a L_{in} .
Luego, $\forall y \in N_4(x)$, con $\phi(y) = 3$, agregar y to L_{out} y hacer $\phi(y) = 1$.
2. Después del paso 1 algunos de los píxeles x en L_{in} pasan a ser píxeles internos.
Por lo tanto, se sacan de L_{in} y se hace $\phi(x) = -3$.
3. Para cada $x \in L_{in}$, si $F_d(x) < 0$ entonces, borrar x de L_{in} y agregarlo a L_{out} .
Luego, $\forall y \in N_4(x)$, con $\phi(y) = -3$, agregar y a L_{in} y hacer $\phi(y) = -1$.
4. Después del paso 3 algunos de los píxeles x en L_{out} pasan a ser píxeles externos.
Por lo tanto, se sacan de L_{out} y se hace $\phi(x) = 3$.

En el segundo ciclo, la curva se suaviza utilizando un filtro Gaussiano, de tal forma que la fuerza de evolución es $F_s(x) = G \otimes \phi(x)$. Para aplicar F_s se usan los mismos pasos que para F_d . El resultado final es análogo a modificar la curva de acuerdo a la definición formal dada anteriormente (Ecuación 3.5).

En cada cuadro, el borde del contorno del objeto es actualizado de acuerdo al resultado obtenido por el algoritmo en el cuadro anterior. En el caso de la primera imagen, se puede dar de forma supervisada o semi-supervisada por el usuario, o bien puede ser obtenida de forma automática mediante algoritmos de aprendizaje complejos basados en características predefinidas.

El algoritmo termina cuando se alcanza la condición de corte, dada por las ecuaciones 3.9 o cuando se alcanza el numero de iteraciones N_a

$$\begin{aligned} F_d(x) &\leq 0 \quad \forall x \in L_{out} \\ F_d(x) &\leq 0 \quad \forall x \in L_{in} \end{aligned} \tag{3.9}$$

3.3. Elección

Se escogió el algoritmo *Contornos Activos* por varios motivos:

- No depende de disponer de bloques de $n \times n$ píxeles para su correcto funcionamiento. Una de las principales restricciones de este trabajo es utilizar una sola cámara para obtener el video, y, como se explica en la Sección 2.4.2, un jugador está formado por un número muy limitado de píxeles, dificultando el uso de algoritmos que no tengan esta característica.

²Podría no ser supervisada. Existen variantes con determinaciones semi-supervisadas del objeto de interés, así como también detección automática basada en ciertas características predefinidas.

- Su tiempo de ejecución no depende del tamaño del cuadro, sino del de los jugadores. Esto hace que el tiempo de procesamiento de un cuadro sea muy pequeño, posibilitando el análisis en tiempo real.
- Se cuenta con métodos de manejo de oclusiones para *Contornos Activos*[28].

3.3.1. Parámetros

Se seleccionaron experimentalmente los siguientes valores para las constantes del método:

- $N_a = 500$
- Para el filtro G , se usa una máscara de 7×7 con un $\sigma = 0,7$

3.4. Limitaciones

El correcto funcionamiento del algoritmo de contornos activos depende de una buena selección de la función característica, para poder distinguir claramente a un jugador respecto a otros objetos o respecto del fondo (ver [1]). En casos como el ilustrado por la Figura 3.1, elegir una función característica resulta sencillo, ya que con elegir el color de la camiseta se asegura una buena descripción del contorno a seguir. Sin embargo, la imagen de la Figura 3.2 introduce uno de los problemas en la selección de esta función. Se puede ver que las diferencias entre ambos colores del objeto hacen difícil identificarlos a ambos con una misma característica. Si se toma el promedio del valor RGB , considerando que el color negro tiene valor $(0, 0, 0)$ y el rojo $(255, 0, 0)$, el promedio de ambos sería $(127, 0, 0)$ (un marrón oscuro). Este valor muestra distinto a ambos colores, y no describe bien a ninguno. Es por eso, que en la Figura 3.2 el algoritmo de contornos activos podría determinar que uno de los dos colores de la camiseta es más similar al fondo que al promedio, y en consecuencia actualizar el contorno siguiendo únicamente uno de los dos colores.



Figura 3.1: Camiseta de color lisa. El color de la camiseta es claramente distinguible del fondo.



Figura 3.2: Camiseta de 2 colores a rayas. Uno de los dos colores es más similar al color de fondo que al promedio.

Esto es dificultoso debido a dos problemas:

- Selección de colores: si la cancha tiene un color muy similar a la camiseta de un equipo, ¿cómo será posible distinguirlos? En este trabajo se exploran distintas alternativas, como utilizar distintas codificaciones de color.

- Textura de los jugadores: esto representa un gran problema por varias razones. La primera es que la técnica de contornos activos depende de la selección de una o varias características del objeto a seguir. Como la característica más distintiva es el color, la técnica se basa en ella. Pero para camisetas con más de un color, encontrar un único color característico no es posible. Esta dificultad se ilustra en las imágenes de las figuras 3.3 y 3.4. Por otro lado, dada la escasa cantidad de píxeles que representan a un jugador, debido al enfoque de única cámara de este trabajo, cualquier tipo de análisis se torna complejo cuando sus características no están lo suficientemente definidas como para diferenciarlas (ya sea de otros jugadores o del fondo).



Figura 3.3: Desde este punto de vista, se puede observar al menos 3 fuertes características (colores) en la camiseta del jugador, rojo, azul y amarillo.

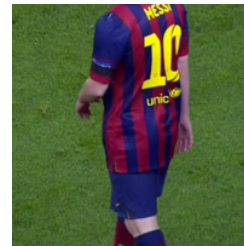


Figura 3.4: Incluso con mayor resolución, las distintas características (como ser los colores azul, amarillo, y rojo) dificultan el seguimiento con contornos activos.

Capítulo 4

Solución Propuesta

En esta Sección se describen las técnicas aplicadas para lograr el seguimiento de los jugadores. En primer lugar, se detallan algoritmos utilizados para ignorar los elementos del fondo de la imagen en la Sección 4.1. Luego, se detalla cómo fue utilizado el algoritmo contornos activos y las modificaciones que se le hicieron en la Sección 4.2.

4.1. Eliminación de Fondo

Se evaluó que el análisis por contornos activos se beneficiaría de un análisis previo que detecte y otorga información al proceso de actualización de contornos activos sobre qué sectores de los cuadros del video no corresponden a las siluetas de los objetos de interés para el seguimiento.

Con ese fin, se analizaron distintos métodos para extraer información adicional de la imagen y detectar e ignorar sectores de la imagen que no correspondan a jugadores. A continuación se describen los métodos evaluados.

4.1.1. Tribuna y Publicidades

La técnica más simple de eliminación de sectores es una técnica de *recorte* que elimina de la imagen todo píxel ajeno a un polígono (como ser un cuadrilátero) que bordea la cancha. En las Figuras 4.1 y 4.2 se muestra el resultado de aplicar esta técnica a uno de los videos utilizados.



Figura 4.1: Un cuadro del video de un partido.

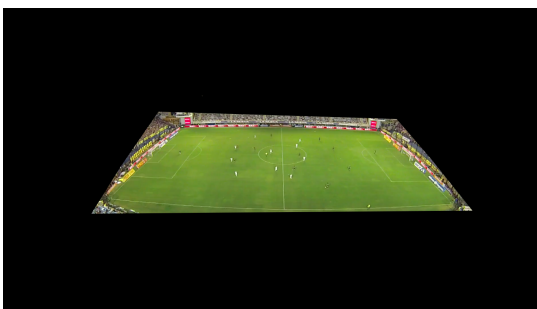


Figura 4.2: El mismo cuadro, luego de aplicar la operación *recorte*.

4.1.2. Substracción de Fondo por Valor de Energía

Se implementó el método de eliminación de fondo descrito en [16] para eliminación de sectores que corresponden al verde del césped de la cancha o las líneas blancas pintadas sobre el mismo, basado en una medición de la variación del color de cada píxel, a la cual se refiere como energía.

Este no resultó ser un método apropiado debido a que el sistema de codificación del video genera muchos falsos negativos, sobre todo *glitches*. Un *glitch* es un error (de encodificación o filmación) que genera cambios en los valores de colores (aumentando la energía de ese píxel). La gran cantidad de *glitches* alrededor de las líneas de la cancha les otorga a estos puntos un mayor valor de energía del que realmente tienen. En la Figura 4.3 se muestra un recorte del primer cuadro del video de un partido entre Boca e Independiente. En las figuras 4.4 y 4.5 se muestra el resultado de aplicar este algoritmo en ese mismo recorte. Se puede ver que en el cuadro #27 detecta muy bien el fondo, pero los cambios que se producen en el cuadro siguiente confunden al método, eliminando algunos píxeles que estaban bien detectados.



Figura 4.3: Recorte del cuadro #1 del video del partido entre Boca e Independiente



Figura 4.4: Recorte del cuadro #27 del video. El color azul denota los píxeles detectados como fondo por el algoritmo.

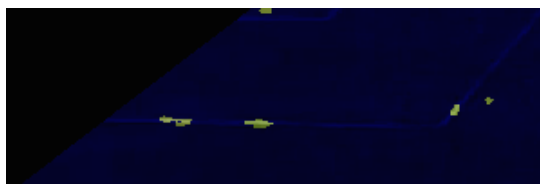


Figura 4.5: Recorte del cuadro #28 del video. En este caso, el efecto de *glitch* causa un falso negativo y no detecta parte de la cancha como fondo.

Otro problema encontrado con este algoritmo es que, al basarse en cambios de color, puede considerar que un jugador que se encuentra estático durante varios cuadros es parte del fondo. Si bien los jugadores suelen estar en movimiento durante el partido, esto no es tan cierto en el caso del arquero.

Por los motivos detallados, se descartó el uso de este algoritmo en favor de las técnicas descritas en la próximas Secciones, 4.1.3 y 4.1.4.

4.1.3. Eliminación de Líneas

Las líneas blancas que delimitan la cancha y sus distintas partes presentan un problema para el seguimiento de los equipos con camisetas blancas o de colores claros. Para evitar que el algoritmo de

contornos activos considere que las líneas son parte de un jugador, como en la figura 4.6, se considera otro método para detectarlas e ignorar estos puntos al momento de realizar el paso de actualización de contornos.

Se desarrolló un método que detecta los tramos pintados de blanco en el césped de la cancha. El mismo funciona aplicando un detector de bordes (se utilizaron tanto el método de Roberts como el de Canny), umbralizando el resultado, y haciendo un análisis morfológico de las componentes conexas obtenidas luego de la umbralización. Este método está basado en la idea detrás del algoritmo de detección de líneas de Hough.

El análisis morfológico considera que una componente conexas entre píxeles detectados como bordes es un jugador si se cumplen las siguientes condiciones:

- El ancho de la región es menor a un valor de umbral M_{width} .
- El alto de la región es menor a un valor de umbral M_{height} .
- El área cubierta es menor a un valor de umbral M_{area} .

Estos tres valores son parámetros del algoritmo y dependen de la resolución y *zoom* del video (a mayor resolución, los jugadores abarcan mayor área y las regiones que ocupan son más altas y anchas).



Figura 4.6: Resultado de aplicar el algoritmo sin eliminación de líneas. El color celeste delimita el contorno de un jugador, incluyendo erróneamente parte de las líneas de la cancha.

4.1.4. Eliminación del Césped

Para la eliminación del césped del campo de juego se requiere caracterizar los píxeles correspondientes al verde del campo. Para esto, se realiza un análisis de los colores de la imagen recortada, es decir la imagen resultante luego aplicar la técnica de *recorte*, detallada en 4.1.1, una vez seleccionados los contornos iniciales de los jugadores. El análisis consta de calcular el valor promedio y desvío estándar del conjunto de píxeles que no forme parte de los jugadores o un elemento ya conocido del fondo (por ejemplo, las líneas de la cancha sobre el césped).

Con este valor, se considera parte del césped cualquier punto que no tenga las características de un jugador (según definidas en la Sección 4.2.1), y se encuentre a menos de 3 desvíos del color promedio calculado.

4.2. Contornos activos

Contornos Activos requiere para su implementación que se definan dos funciones. Una es la función de probabilidad p y la otra la función característica v (ver Sección 3.1). En esta Sección se describe la elección de ambas funciones para este trabajo.

4.2.1. Características

Como se explica en la Sección 3.4, cuando los objetos de interés son complejos, la utilización del color promedio como única característica distintiva no alcanza. Es por esto que varias de las mejoras planteadas al algoritmo de contornos activos giran en torno a la selección de características para representar a los objetos de interés. A continuación, se describen los cambios que se han incorporado para hacer el algoritmo más efectivo ante un video correspondiente a un partido de fútbol.

Si bien el color promedio resulta insuficiente para caracterizar correctamente al objeto, puede utilizarse en complemento con otras características. Una opción es la utilización del desvío estándar de color en el objeto de interés. Este se calcula de la manera descrita en el Algoritmo 4, donde la rutina *DesvioLocal* se define en el Algoritmo 5.

Algoritmo 4 Característica Desvío Estándar

Input: Cuadro c , Contornos $\omega_1 \dots \omega_n$
Output: $(r_{dev}, g_{dev}, b_{dev})$

```
 $\Omega \leftarrow \cap \omega_1 \dots \omega_n$ 
 $count \leftarrow 0$ 
 $r_{sum} \leftarrow 0$ 
 $g_{sum} \leftarrow 0$ 
 $b_{sum} \leftarrow 0$ 
for  $p \in c$  do
  if  $p \notin \Omega$  then
     $count \leftarrow count + 1$ 
     $(r_{dev}, g_{dev}, b_{dev}) \leftarrow \text{DesvioLocal}(p, c)$ 
     $r_{sum} \leftarrow r_{dev} + r_{sum}$ 
     $g_{sum} \leftarrow g_{dev} + g_{sum}$ 
     $b_{sum} \leftarrow b_{dev} + b_{sum}$ 
return  $(\frac{r_{sum}}{count}, \frac{g_{sum}}{count}, \frac{b_{sum}}{count})$ 
```

Algoritmo 5 DesvioLocal

Input: Posición p , Cuadro c ,**Output:** $(r_{dev}, g_{dev}, b_{dev})$

```
 $r_{avg} \leftarrow 0$ 
 $g_{avg} \leftarrow 0$ 
 $b_{avg} \leftarrow 0$ 
for  $p'$  8-conectado a  $p$  do
     $(r, g, b) \leftarrow c_{p'}$ 
     $r_{avg} \leftarrow r + r_{avg}$ 
     $g_{avg} \leftarrow g + g_{avg}$ 
     $b_{avg} \leftarrow b + b_{avg}$ 
 $(r_{avg}, g_{avg}, b_{avg}) \leftarrow (\frac{r_{sum}}{count}, \frac{g_{sum}}{count}, \frac{b_{sum}}{count})$ 
 $r_{dev} \leftarrow 0$ 
 $g_{dev} \leftarrow 0$ 
 $b_{dev} \leftarrow 0$ 
for  $p'$  8-conectado a  $p$  do
     $(r, g, b) \leftarrow c_{p'}$ 
     $r_{dev} \leftarrow r_{dev} + (r - r_{avg})^2$ 
     $g_{dev} \leftarrow g_{dev} + (g - g_{avg})^2$ 
     $b_{dev} \leftarrow b_{dev} + (b - b_{avg})^2$ 
return  $(\sqrt{\frac{r_{dev}}{8}}, \sqrt{\frac{g_{dev}}{8}}, \sqrt{\frac{b_{dev}}{8}})$ 
```

Entonces la función v queda defininida como:

$$\begin{aligned}(r, g, b) &= c_x \\(r_{dev}, g_{dev}, b_{dev}) &= \text{DesvioLocal}(x, c) \\v(x) &= (r, g, b, r_{dev}, g_{dev}, b_{dev})\end{aligned}$$

Y para cada contorno se define su vector característico v_i como el resultado de aplicar los Algoritmos 6 y 4.

Aprendizaje de valores

Debido a que a lo largo del campo de juego la iluminación no es uniforme, y que durante el partido la misma puede cambiar, se realiza un aprendizaje simple de las características de cada contorno. En cada cuadro, siendo \mathbf{v} un vector con las características de un contorno dado, se calcula el promedio de los valores de las características para todos los píxeles pertenecientes al contorno y se lo denomina $\hat{\mathbf{v}}$. A continuación se actualiza \mathbf{v} y se lo reemplaza por un nuevo valor \mathbf{v}_n calculado de acuerdo a:

$$\mathbf{v}_n = (1 - \alpha) \mathbf{v} + \alpha \hat{\mathbf{v}}$$

Se toma un valor de α del orden de 10^{-2} , es decir, se toma el 1 % de $\hat{\mathbf{v}}$. Esto evita la posibilidad de

Algoritmo 6 PromedioRGB

Input: Cuadro c , Contorno ω **Output:** (r_a, g_a, b_a)

```
count  $\leftarrow$  0
rsum  $\leftarrow$  0
gsum  $\leftarrow$  0
bsum  $\leftarrow$  0
for  $p \in \omega$  do
    count  $\leftarrow$  count + 1
     $(r, g, b) \leftarrow c_p$ 
    rsum  $\leftarrow$   $r + r_{sum}$ 
    gsum  $\leftarrow$   $g + g_{sum}$ 
    bsum  $\leftarrow$   $b + b_{sum}$ 
return  $(\frac{r_{sum}}{count}, \frac{g_{sum}}{count}, \frac{b_{sum}}{count})$ 
```

que se memoricen cambios no deseados, como podría ser una oclusión parcial, antes de su corrección.

4.2.2. Función de Probabilidad

Si bien la función de probabilidad sugerida en la Sección 3.1 se puede aplicar en este caso, se modificó para tener en cuenta la morfología de los objetos a seguir.

Se plantea que los jugadores estarán contenidos dentro de un rectángulo de mayor altura que ancho. Se estima el máximo valor posible del ancho y alto de un rectángulo que contenga a un jugador, y se limita el tamaño del contorno, impidiendo que se expanda y no pueda ser contenido de un rectángulo de este tamaño.

Dado que el algoritmo de contornos activos es utilizado en las líneas del césped del campo de juego para evaluar si un punto pertenece al fondo o no, no se añade la restricción de tamaño a estas líneas. Se sugieren dos funciones de probabilidad, una denominada f utilizada para la detección del fondo y una función f_j para el seguimiento de los jugadores, definidas como:

$$\begin{aligned} f(p) &= 1 - \frac{\|v(p) - v_i\|}{255 * 6} \\ f_j(p) &= 1 - \min(1, f(p) + \text{distance_weight}(p_x, p_y, \omega_i)) \end{aligned}$$

Donde v_i es el valor característico para el contorno ω_i contra el que se evalúa el punto p . Y la rutina *distance_weight* se define en el Algoritmo 7, donde $maxX$ y $maxY$ son el semi-ancho y la semi-altura máxima del rectángulo, respectivamente.

Seguimiento por Complemento

Como se explica en la Sección 3.4, no resulta evidente cómo construir una función v para camisetas que presenten más de un color. Si bien el agregado del Desvío Estandar contribuye a mejorar el segui-

Algoritmo 7 distance_weight

Input: Posición (x, y) , Contorno ω
Output: Un número

```
centroidx  $\leftarrow$  0
centroidy  $\leftarrow$  0
for  $p \in \omega$  do
    centroidx  $\leftarrow$  centroidx +  $p_x$ 
    centroidy  $\leftarrow$  centroidy +  $p_y$ 
centroidx  $\leftarrow$   $\frac{\text{centroid}_x}{\#\omega}$ 
centroidy  $\leftarrow$   $\frac{\text{centroid}_y}{\#\omega}$ 
res  $\leftarrow$  0
distx  $\leftarrow$   $\|p_x - \text{centroid}_x\|$ 
disty  $\leftarrow$   $\|p_y - \text{centroid}_y\|$ 
if distx > maxx then
    res  $\leftarrow$  res +  $\min(1, \frac{\text{dist}_x - \text{max}_x}{\text{max}_x})$ 
if disty > maxy then
    res  $\leftarrow$  res +  $\min(1, \frac{\text{dist}_y - \text{max}_y}{\text{max}_y})$ 
return res
```

miento en estos casos, no soluciona totalmente el problema. Para poder seguir correctamente a equipos cuyas camisetas presentan este desafío, se ofrece la posibilidad de descartar el algoritmo de contornos activos para el seguimiento de este equipo y determinar que los espacios que por sus características no son definidos como césped, líneas de la cancha, o jugadores de otro equipo, son jugadores del equipo cuyas características son difíciles de determinar. Este método solo es efectivo si uno de los equipos puede ser correctamente identificado por contornos activos.

Con este fin, se ejecuta el algoritmo de contornos activos sobre el primer cuadro, con los jugadores del equipo identificable seleccionados. Luego, se aplica el procedimiento *complemento*, detallado en el Algoritmo 8. La salida de este procedimiento marca todo punto que no fue identificado bajo ningún conjunto con el color cyan, y aplica un operador morfológico de expansión sobre todos estos puntos con el objeto de unificar conjuntos que están cercanos.

Algoritmo 8 complemento

Input: Cuadro c , Contornos $\omega_1 \dots \omega_n$, cespel y líneas ω_0
Output: Cuadro c

```
 $\Omega \leftarrow \cap \omega_0 \dots \omega_n$ 
for  $(x, y) \in c$  do
    if  $(x, y) \notin \Omega$  then
         $c_{x,y} \leftarrow \text{CYAN}$ 
        for  $(x', y')$  4-conectado a  $(x, y)$  do
             $c_{x',y'} \leftarrow \text{CYAN}$ 
return  $c$ 
```

Luego de esto, se utiliza la técnica de contornos activos siendo la función de probabilidad de los contornos del segundo equipo una función que devuelve 1 si el píxel es color *cyan*, y 0 en caso contrario.

Capítulo 5

Resultados

Este Capítulo presenta un análisis de los resultados obtenidos a partir de la ejecución del algoritmo utilizando distintos videos. A lo largo de esta Sección se refiere al *algoritmo* como la versión modificada de contornos activos que fue descripta en el Capítulo 4. Las palabras *aplicación* y *programa* serán utilizadas indistintamente para referirse al programa que implementa dicho algoritmo, junto con una interfaz de usuario para el operador. Se refiere a la *implementación* como la parte del programa que implementa el algoritmo.

En primer lugar, la Sección 5.1 describe la aplicación, los datos que deben ser provistos por el operador, y el funcionamiento de la misma. La Sección 5.3 presenta los resultados obtenidos de la ejecución del programa y las métricas utilizadas para evaluar la performance de la implementación. Por último, en la sección 5.4 se presenta la comparación con otro método de seguimiento.

5.1. Aplicación

El programa incorpora una interfaz gráfica utilizada tanto para dar instrucciones como para recibir información acerca del partido y del funcionamiento del algoritmo. La misma cuenta con una sección donde se muestra una lista de los jugadores que están siendo seguidos, una donde se puede visualizar el video original con un recuadro en cada jugador seguido, un mapa de calor que muestra las posiciones más frecuentes, y una imagen que permite al operador evaluar cualitativamente el correcto funcionamiento del algoritmo. Se puede ver una captura de pantalla de la aplicación en la Figura 5.1.

Para comenzar la ejecución del algoritmo, el programa requiere que el operador identifique la posición de los jugadores en la imagen inicial de la secuencia. En este paso, también debe agregar información acerca del número de camiseta que viste, el equipo al cual pertenece y el nombre de cada jugador. Se puede obtener información de un jugador específico mediante la lista de jugadores seguidos en la esquina superior derecha. La Figura 5.2 muestra el detalle con información de un jugador luego de unos segundos de seguimiento.

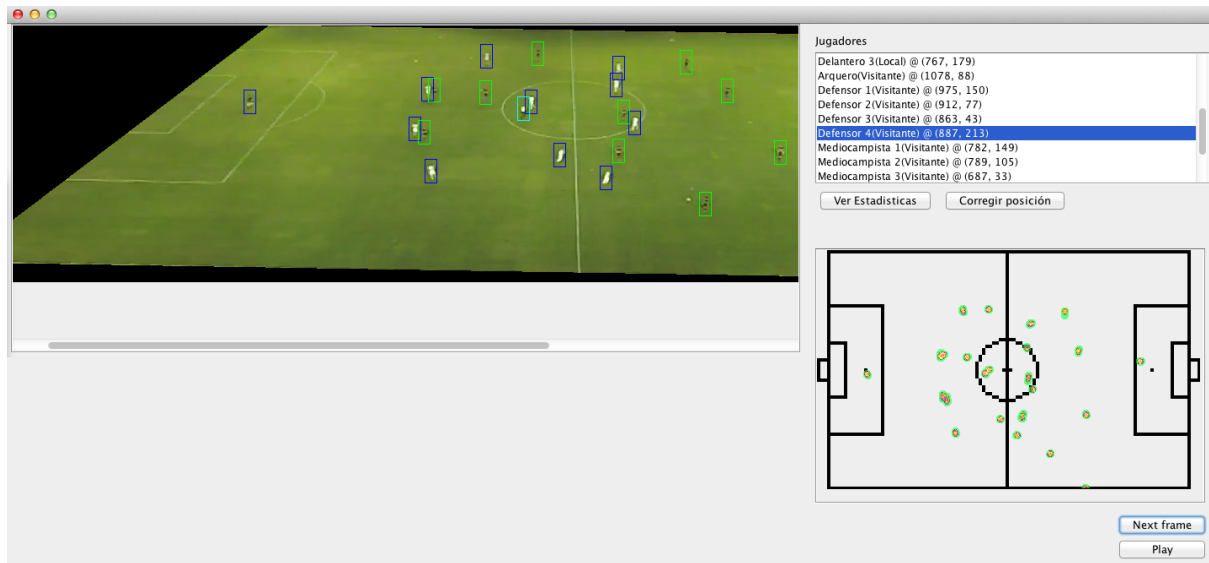


Figura 5.1: Captura de pantalla de la aplicación, mostrando el seguimiento en un video de fútbol.



Figura 5.2: Captura de pantalla de la aplicación, que muestra información del seguimiento de un jugador.

Un paso adicional que debe ejecutar el operador es determinar los puntos a utilizar para el cálculo de la homografía. Los datos necesarios para ello son cuatro parejas de puntos, como se explica en la Sección 1.2. De estas parejas, el primer punto es una coordenada de la imagen en perspectiva y el otro punto es la coordenada en un plano bidimensional. Con cuatro de estas relaciones, se puede calcular la matriz que resuelve la homografía para cualquier otro punto.

Una vez que el programa tiene estos datos, el algoritmo puede comenzar su ejecución. Existen dos modos: en un modo se puede avanzar un cuadro sólo ante la indicación del operador (en una mecánica de tipo “cuadro por cuadro”) y en otro modo se puede avanzar automáticamente cada vez que se computa un cuadro (mecánica de tipo “tiempo real”).

Para cada cuadro, se informa:

- Posiciones actualizadas de los jugadores
- Velocidad actual, promedio y máxima de un jugador
- Mapa de calor del recorrido de cada jugador
- Mapa de calor del recorrido de todos los jugadores

Finalmente, en un archivo se guarda la posición de cada jugador para cada momento. Y de manera opcional, en cada cuadro se guarda en disco duro una copia del estado actual del seguimiento para referencia futura.

5.2. Material Utilizado

Se utilizaron principalmente dos videos, uno correspondiente a un partido entre los equipos argentinos de los clubes Boca Juniors e Independiente; y un segundo video en el cual se enfrentan los equipos Independiente y San Lorenzo. Se detalla a continuación las características de las imágenes extraídas de esos videos.

- **Boca vs. Independiente:** El video cuenta con una resolución de *1080p* (1920 píxeles de ancho y 1080 de alto). La cancha se muestra en su totalidad, y se puede observar las gradas del lado opuesto y cielo por encima de ellas. Luego de descartar esas regiones del video, la resolución pasa a ser de 1459 píxeles de ancho por 304 de alto. Se puede apreciar en la Figura 5.3 un cuadro del video, luego de la extracción de las gradas y corrección del efecto de curvatura de la lente.

Los jugadores de Independiente usan remera y shorts de color blanco, fácilmente identificables respecto al fondo de color verdoso. El árbitro, de amarillo, también contrasta respecto al fondo. Los jugadores de Boca, por otro lado, son difíciles de identificar a la distancia y son confundidos con el color del césped de la cancha, como se ilustra en la Figura 5.4.

- **Independiente vs. San Lorenzo:** También filmado en resolución de *1080p*, las esquinas del campo de juego quedan fuera del campo visual. El video fue editado con anterioridad y el alto de un cuadro es menor al alto de un video en resolución *1080p*. Luego de descartar las gradas, la resolución final del video es de 1920 píxeles de ancho y 540 de alto. La Figura 5.5 muestra un cuadro del video procesado.

Los jugadores de ambos equipos, al utilizar los valores RGB de cada píxel, contrastan contra el césped y el algoritmo de contornos activos funciona correctamente al analizarlo cualitativamente.



Figura 5.3: Cuadro del video del partido entre Boca e Independiente.



Figura 5.4: Acercamiento a dos jugadores en el video de Boca vs. Independiente. De acuerdo a la iluminación, se puede notar que el contorno de un jugador puede resultar muy difícil de delimitar.



Figura 5.5: Cuadro del partido en el que Independiente enfrenta a San Lorenzo.

Además, se tomaron pequeños cortes de tres videos de fútbol televisado, en los cuales la cámara se encuentra prácticamente estática y se analizó la correctitud del seguimiento en estos casos, contando con

mayor resolución pero sin poder efectuar exitosamente la eliminación de fondo o la aplicación de una homografía para determinar las posiciones correctamente, ya que cada cuadro donde la cámara cambia su orientación, la matriz de la homografía debería ser recalculada con nuevos datos.

Los videos televisados utilizados pertenecen a tres partidos disputados durante el año 2014. De la liga UEFA, se tomaron recortes de los partidos **Manchester City vs Barcelona FC**, **Real Madrid vs Borussia Dortmund**, y de la **FIFA World Cup** se tomaron fragmentos del partido de cuartos de final de **Argentina vs Suiza**.

Del primer video televisado se extrajeron tres escenas en particular. En la Figura 5.6 se muestran algunos cuadros de estos fragmentos. El video cuenta con una resolución de 720p (1280 píxeles de ancho por 720 píxeles de alto). Los jugadores de Manchester City, con su vestimenta deportiva de color blanco, son confundidos por sus características con el color de las líneas de la cancha. En el primer recorte, los jugadores son correctamente identificados durante los 120 cuadros, exceptuando aquellos que atraviesan una línea de cancha. Se corrigen correctamente dos oclusiones parciales entre jugadores. En el segundo fragmento, los jugadores son correctamente localizados a pesar de la alta velocidad de *panning* de la cámara. En el tercer video todos los jugadores, al igual que en el primero, son correctamente localizados durante todo el video exceptuando aquellos que se intersectan con las líneas marcadas en el césped.

En los otros dos videos se observaron similares características al video de Manchester City, el seguimiento funciona correctamente bajo la asunción de que los jugadores no estén ocluyendo una línea en el césped. La Figura 5.7 muestra un cuadro donde se puede apreciar los colores de la camiseta de ambos equipos en el partido **Real Madrid vs Borussia Dortmund** y en la Figura 5.8 se observa una captura del partido de **Argentina vs Suiza**.

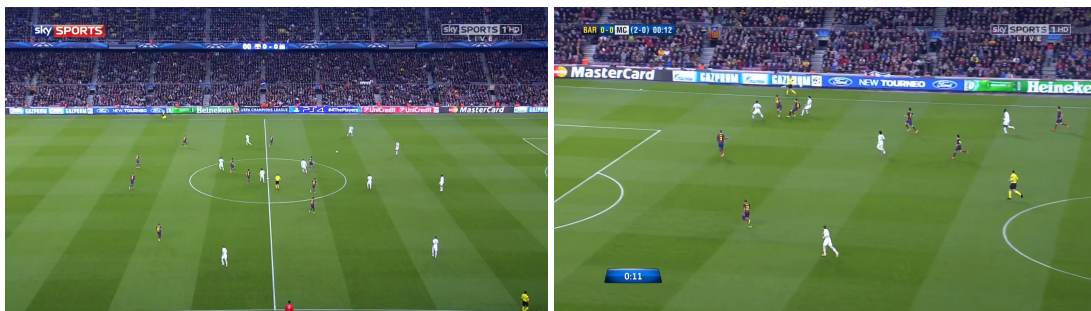


Figura 5.6: Fragmentos del video televisado de un partido entre los equipos Manchester City y Barcelona FC.



Figura 5.7: Real Madrid vs Borussia Dortmund, UEFA Champions League

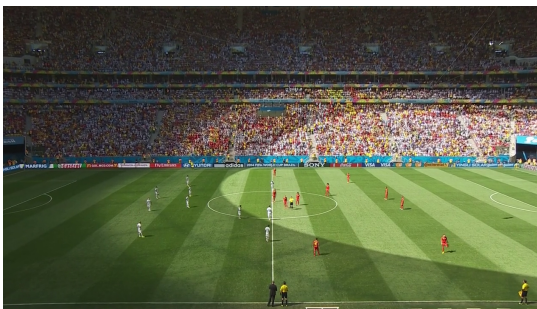


Figura 5.8: Argentina vs Suiza, FIFA World Cup 2014

5.3. Evaluación del Método

La principal evaluación del correcto funcionamiento del algoritmo es llevada a cabo por el operador de forma cualitativa, como ocurre en otros trabajos citados en el estado del arte (ver [16]). El resultado que arroja esta evaluación es positivo, consiguiéndose seguir correctamente a todos los jugadores en casi todo momento. Sin embargo, existen algunas situaciones en las que el algoritmo falla en el seguimiento y no logra recuperarse por sí mismo, requiriendo una corrección manual por parte del operador para continuar su funcionamiento. Esto se atribuye a la baja calidad de los videos utilizados como material y a la poca capacidad de procesamiento en comparación (el poder de procesamiento de una sola computadora limita algunas operaciones o técnicas).

Las situaciones en que se producen estos errores están bien identificadas y ocurren porque el algoritmo, en una situación de proximidad entre dos jugadores, a veces confunde a la sombra del otro jugador con el jugador al cual se encuentra siguiendo, y comienza a seguir a la sombra, perdiendo al jugador.

Para evaluar estos inconvenientes se define una métrica que mide exactamente la cantidad de errores por unidad de tiempo. Esta métrica, resumida como la cantidad de errores del algoritmo por cada cien cuadros, se intenta minimizar durante el estudio de las variantes de la aplicación.

La ejecución del algoritmo utilizando el video entre Boca e Independiente resulta en 7 errores en 766 cuadros, dando aproximadamente 0.91 errores cada cien cuadros, lo que significa un poco menos de un error cada 4 segundos.

En el video del partido entre Independiente y San Lorenzo el algoritmo incurre en 6 errores en 213 cuadros, dando 2.81 errores cada cien cuadros. Es decir, algo menos de 3 errores cada 4 segundos.

Otro aspecto importante a la hora de evaluar la solución propuesta es su velocidad, medida en el tiempo de ejecución que requiere para llevar a cabo su trabajo. Para medir esto se define la métrica del tiempo que le toma al algoritmo procesar un solo cuadro, o, lo que es lo mismo, la cantidad de cuadros procesados por segundo. El objetivo es minimizar esta métrica intentando acercarse lo más posible a los 24 cuadros por segundo (equivalente a 42 milisegundos por cuadro), la velocidad de los videos utilizados.

5.4. Comparación con IFTrace

El algoritmo de seguimiento IFTrace, propuesto por Minetto et al., es un algoritmo robusto que soporta cambios de iluminación y forma, oclusiones y se centra en hallar características representativas de la textura de los objetos a seguir. Es capaz de recuperarse de errores menores y permite el seguimiento de múltiples objetos a la vez.

Un algoritmo de este tipo podría proporcionar una solución al problema. Para comprobarlo, se llevaron a cabo algunas pruebas utilizando un video sintético creado para este fin, y un video real de un partido de fútbol. En la Figura 5.1 de la sección anterior se observa un video real. La figura 5.9 muestra el primer cuadro del video sintético. Pueden apreciarse a simple vista las marcadas diferencias entre ambos videos, como por ejemplo la resolución de la imagen y el tamaño y la complejidad de los objetos de interés.

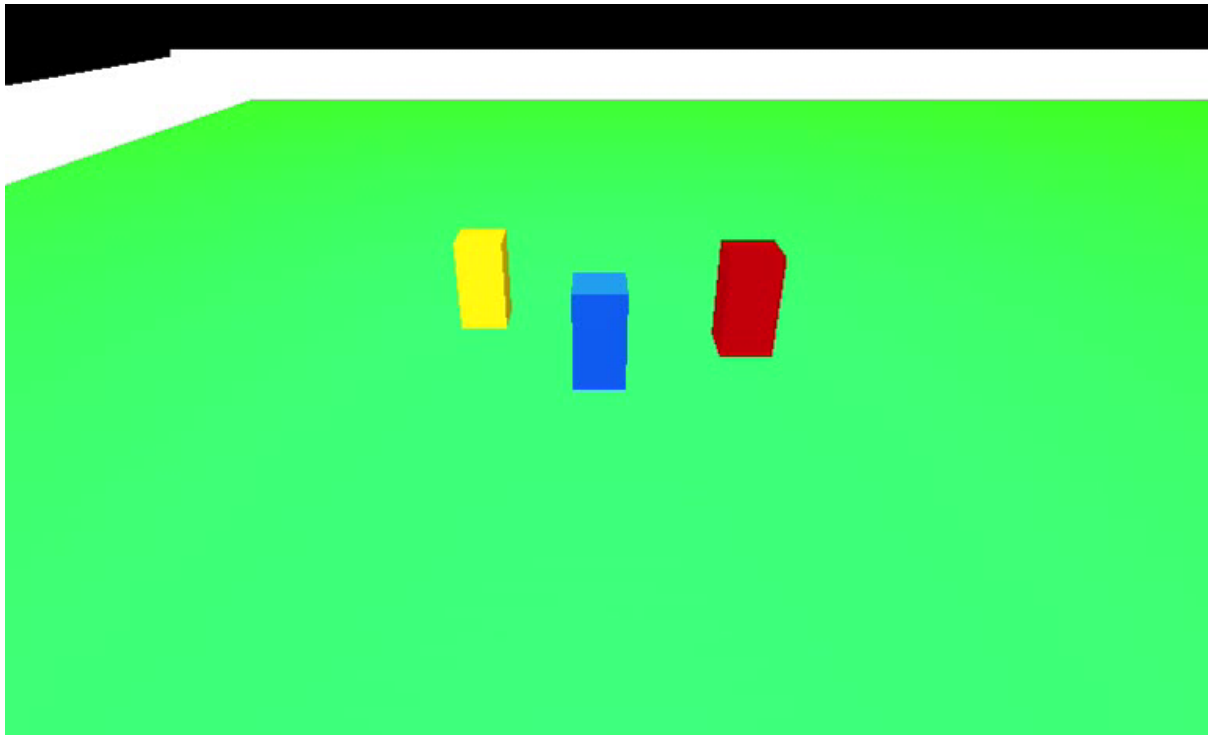


Figura 5.9: Muestra de un cuadro del video sintético de prueba.

Como se puede ver en la Figura 5.10, IFTrace logra un correcto seguimiento de múltiples objetos en el video sintético. También puede observarse, en la Figura 5.11, como sigue correctamente a un jugador en el video real. Sin embargo, el seguimiento sólo es exitoso durante unos pocos cuadros, ya que, en el cuadro 17, el algoritmo cae en un error del cual sólo una corrección manual puede sacarlo. Este tipo de corrección semi-supervisada no está contemplada en el algoritmo de IFTrace.

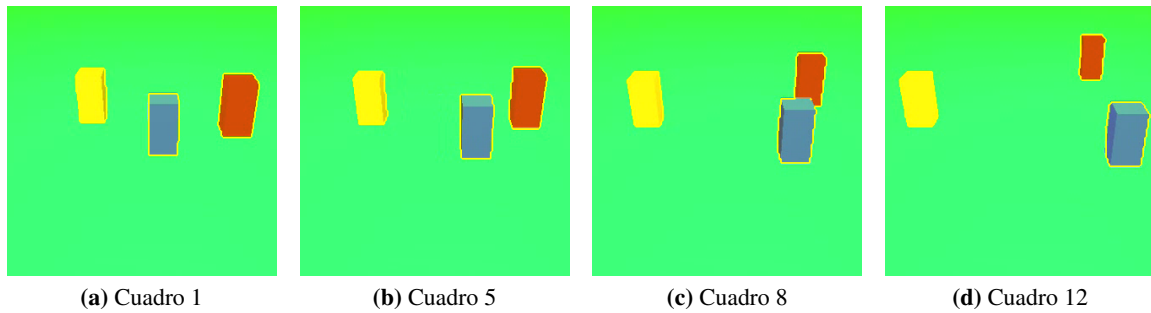


Figura 5.10: IFTrace funcionando en una secuencia de cuadros de video sintético.

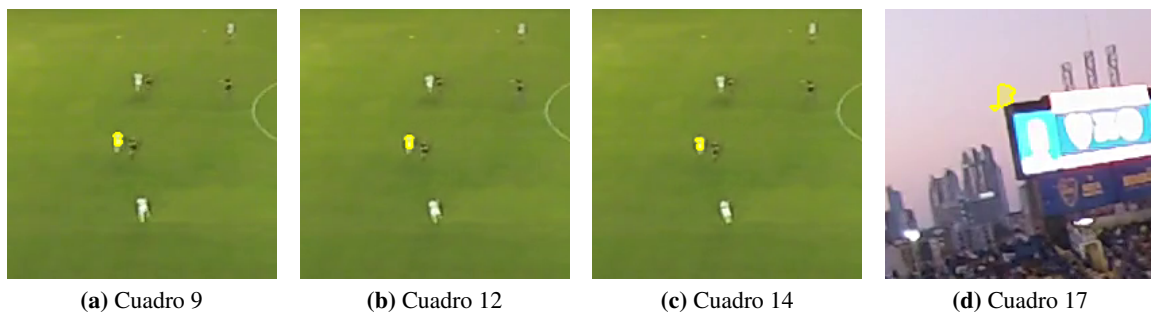


Figura 5.11: Seguimiento de un jugador en un video real utilizando IFTrace.

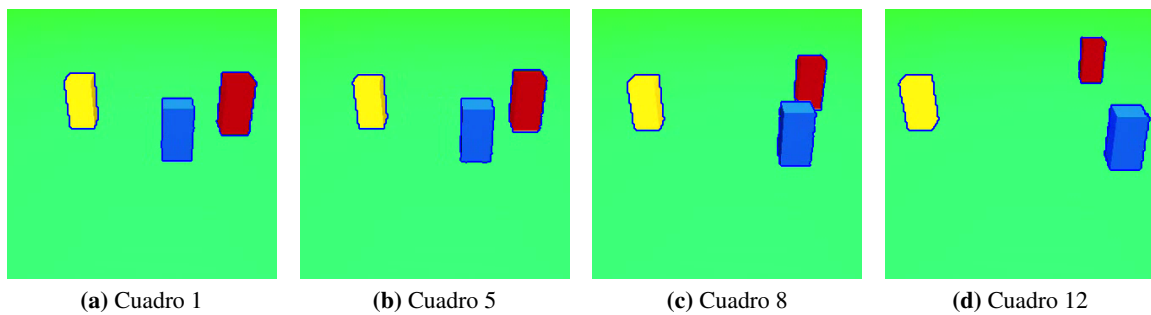


Figura 5.12: El algoritmo de contornos activos modificado en funcionamiento en un video sintético.

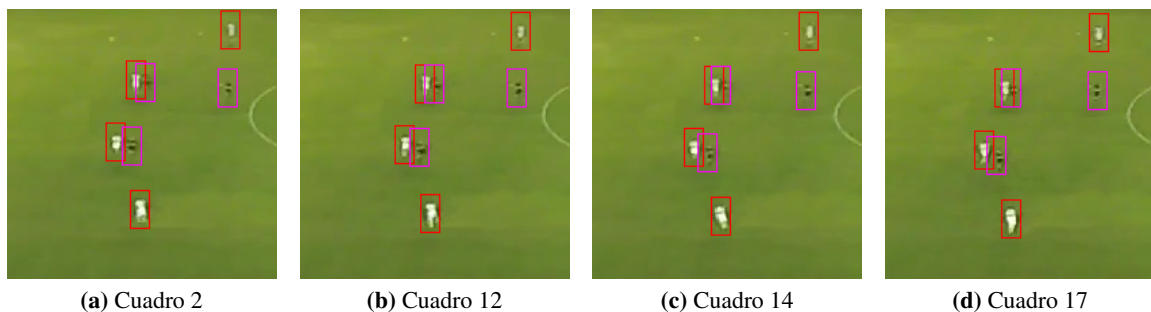


Figura 5.13: Seguimiento de los jugadores en un video real mediante el algoritmo de contornos activos modificado.

Como se puede observar en la Figura 5.12, el algoritmo propuesto en este trabajo logra seguir con

éxito a los objetos de interés en el video sintético. Además, también se obtiene un resultado positivo en el video real en la situación en que IFTrace pierde al jugador, como puede observarse en la Figura 5.13.

5.4.1. Evaluación de Comportamiento

Otro punto importante de comparación entre los dos algoritmos es su tiempo de ejecución, es decir el tiempo que tarda en llevar a cabo su trabajo. De acuerdo a las mediciones realizadas con un video real de un partido de fútbol, siguiendo a un solo jugador, el tiempo promedio que tarda IFTrace por cuadro es 6.962 segundos, mientras que el algoritmo implementado tiene un tiempo promedio de 0.638 segundos. Se puede observar que se encuentra un orden magnitud por debajo de IFTrace, incluso antes de realizar optimizaciones.

Cabe destacar que ambos algoritmos podrían verse beneficiados de ciertas optimizaciones, como ser por ejemplo la programación en GPU y la reducción de operaciones de *Input/Output* al almacenamiento secundario (disco duro).

Capítulo 6

Conclusiones

El Capítulo 2 muestra porque el problema en cuestión es de gran complejidad y porque ha sido objeto de interés y de estudio con anterioridad. Diversos autores han estudiado soluciones a este problema en el pasado, y en general las soluciones existentes requieren de muchos recursos para lograr buenos resultados.

Teniendo en cuenta estas dificultades, sumadas a las restricciones del problema, se concluye que la solución propuesta realiza un buen trabajo al lograr el seguimiento de los jugadores con el material disponible. Puede observarse como la solución propuesta resulta flexible y capaz de recuperarse de forma semi-supervisada, algo que otras técnicas no contemplan, como por ejemplo la técnica de *IFTrace*.

Uno de los grandes impedimentos y fuente de problemas es el material utilizado, de mala calidad debido a la baja resolución de la imagen y la elección para la ubicación de la cámara. Conseguir filmaciones de mejor calidad es dificultoso debido a los contratos y derechos de transmisión que actualmente rigen el fútbol argentino. Para utilizar videos televisados de fútbol, se debe resolver el problema de manejar los movimientos de cámara, lo que escapa al objetivo del trabajo. Una desventaja de este tipo de material es que sólo ofrecen una visión parcial del campo de juego (no se tiene información de todos los jugadores en la cancha).

Un enfoque multi-cámara podría resolver el problema de la baja resolución de la imagen y ofrecer mejores resultados (el ejemplo más sencillo serían 2 cámaras que tomen, cada una, una mitad del campo de juego). Además, en el caso de contar con suficientes cámaras, se podría disponer de mayor información a la hora de manejar oclusiones, al tener disponible más de un punto de vista.

6.1. Trabajo Futuro

Este estudio se vería beneficiado de utilizar videos con mayor resolución, con una cámara ubicada sobre el campo del juego apuntando perpendicularmente al plano del campo de juego, o múltiples cámaras para poder hacer una comparación justa del método de contornos activos contra los métodos utilizados por Xu, Orwell y Jones y D'Orazio et al. en los trabajos vistos en el Capítulo 1.

Uno de los objetivos de la investigación de este trabajo es lograr un seguimiento en tiempo real, es decir procesar por lo menos 24 cuadros por segundo. La implementación de referencia no cumple con

este objetivo, pero es posible avanzar. Por ejemplo, el trabajo de preprocesamiento de un cuadro y de aplicación de *Contornos Activos* puede distribuirse en 2 equipos distintos. Esto permite tener una menor latencia entre cuadros, pudiendo duplicar la velocidad de procesamiento.

Existe también la posibilidad de optimizar el algoritmo de *Contornos Activos*. El algoritmo original no está diseñado para seguir 24 contornos en tiempo real. La implementación procesa cada contorno de manera secuencial, desperdiciando la capacidad de multiprocesamiento de las computadoras modernas. Se podría adaptar el algoritmo para aprovechar esta capacidad y mejorar el tiempo de ejecución notablemente.

Bibliografía

- [1] Y. Shi y W.C. Karl. «Real-time tracking using level sets». *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. 2005.
- [2] S. Holzer, S. Ilic y N. Navab. «Multilayer Adaptive Linear Predictors for Real-Time Tracking». *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2013).
- [3] F. Jurie y M. Dhome. «Hyperplane approximation for template matching». *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2002).
- [4] S. Van Der Walt. *Homography Estimation*. 2010.
- [5] C.G. Atkeson, A.W. Moore y S. Schaal. «Locally weighted learning» (1997).
- [6] X. Li y H. Lu. «Object tracking based on local learning». *19th IEEE International Conference on Image Processing (ICIP)*. 2012.
- [7] T. Malisiewicz y A.A. Efros. «Recognition by Association via Learning Per-exemplar Distances». *CVPR*. 2008.
- [8] F. Yang et al. «Visual tracking via bag of features». *IET-IPR* (2012).
- [9] R. Minetto et al. «IFTrace: Video segmentation of deformable objects using the Image Foresting Transform». *Computer Vision and Image Understanding* (2012).
- [10] C. Tomasi y T. Kanade. *Detection and Tracking of Point Features*. Inf. téc. International Journal of Computer Vision, 1991.
- [11] A.X. Falcão, J. Stolfi y R. de Alencar Lotufo. «The image foresting transform: theory, algorithms, and applications». *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2004).
- [12] S. Maitra. «Moment invariants». *IEEE* (1979).
- [13] S. Birchfield. *KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker*. 2010.
- [14] T.H. Cormen et al. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009.
- [15] P.A.V. Miranda y A.X. Falcão. «Links Between Image Segmentation Based on Optimum-Path Forest and Minimum Cut in Graph.» *Journal of Mathematical Imaging and Vision* (2009).
- [16] T. D’Orazio et al. «An Investigation Into the Feasibility of Real-Time Soccer Offside Detection From a Multiple Camera System». *IEEE Transactions on Circuits and Systems for Video Technology* (2009).

- [17] T. Kanade et al. «Advances in Cooperative Multi-Sensor Video Surveillance». *Darpa Image Understanding Workshop*. 1998.
- [18] J. Kainulainen y J. J. Kainulainen. *Clustering Algorithms: Basics and Visualization*.
- [19] M. Xu, J. Orwell y G.A. Jones. «Tracking football players with multiple cameras.» *ICIP*. IEEE, 2004.
- [20] N. Funk. «A study of the kalman filter applied to visual tracking» (2003).
- [21] M. Beetz, B. Kirchlechner y M. Lames. «Computerized Real-Time Analysis of Football Games». *IEEE Pervasive Computing* (2005).
- [22] J. Shi y C. Tomasi. «Good features to track». *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*. 1994.
- [23] I.J. Cox y S.L. Hingorani. «An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking». *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1996).
- [24] D.B. Reid. «An algorithm for tracking multiple targets». *IEEE Transactions on Automatic Control* (1979).
- [25] T. Schmitt et al. «Watch their Moves: Applying Probabilistic Multiple Object Tracking to Autonomous Robot Soccer». in *AAAI National Conference on Artificial Intelligence*. 2002.
- [26] T. Schmitt et al. *Cooperative Probabilistic State Estimation for Vision-based Autonomous Mobile Robots*. 2002.
- [27] Y. Liu et al. «Extracting 3D information from broadcast soccer video». *Image and Vision Computing* (2006).
- [28] J. Gambini et al. «Occlusion Handling for Object Tracking Using a Fast Level Set Method». *Computer Graphics and Image Processing, 2008. SIBGRAPI '08. XXI Brazilian Symposium on*. 2008, págs. 61-68.

Apéndice A

Captura de Pantalla de Aplicación

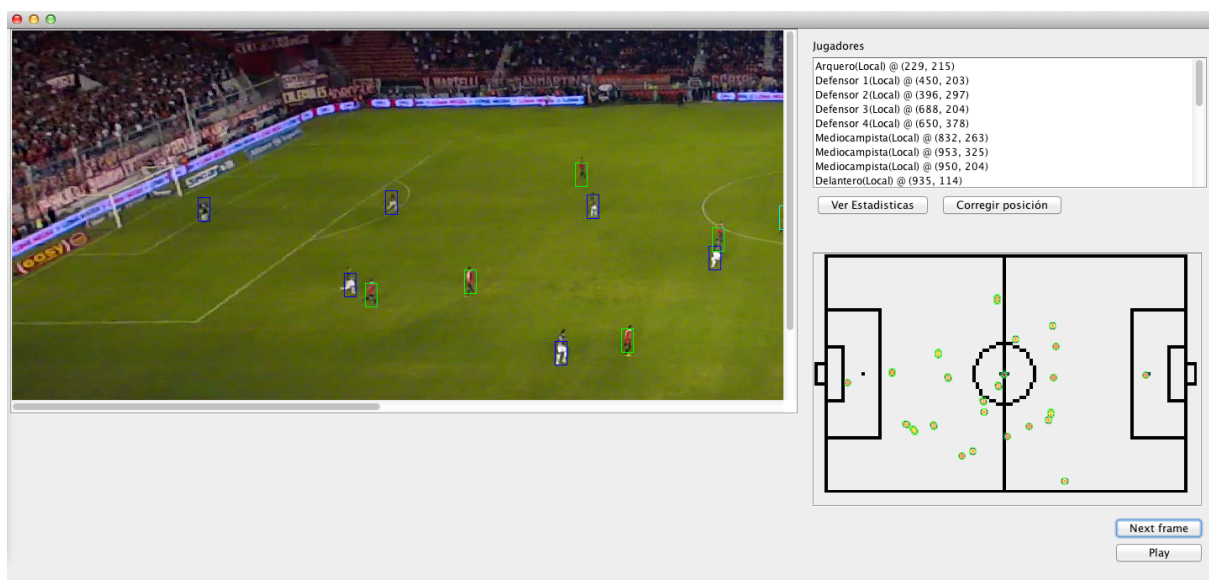


Figura A.1: Captura de pantalla utilizando el video de Independiente.

Apéndice B

Videos Utilizados



Figura B.1: Cuadro del video del partido entre Boca e Independiente.



Figura B.2: Cuadro del video del partido entre Independiente y San Lorenzo.



Figura B.3: Cuadro del video del partido entre Real Madrid y Borussia Dortmund.

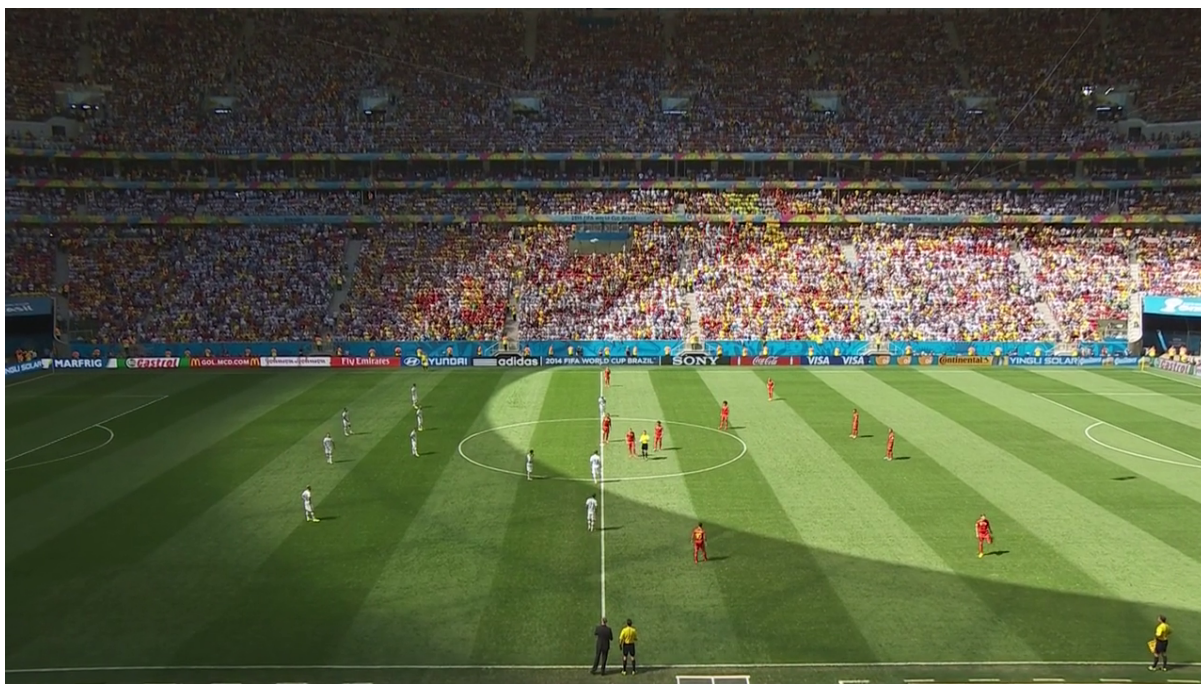


Figura B.4: Cuadro del video del partido entre Argentina y Suiza.



Figura B.5: Cuadro del video del partido entre Manchester City y Barcelona FC.