UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Obtención de "*siteswap*" de malabares mediante análisis de vídeo

Autor: Alejandro Alonso García
Tutor: José María Martínez Sánchez

junio 2023

Agradecimientos

Sinceramente, no suelo ser una persona que valore demasiado las secciones de “agradecimientos”, creo que son siempre muy similares entre sí (independientemente del autor) y que hay gente que termina dentro porque “tiene que estar”. De cualquier forma, eso no significa que no esté agradecido a muchas personas y, aunque intento que lo tengan siempre presente de primera mano, quiero aprovechar y dejarlo aquí por escrito, aunque sea por si acaso.

Por su involucración con este trabajo desde el día que lo cogió, la persona a la que estoy más agradecido es evidentemente mi tutor Chema. Creo que los tutores tienen gran influencia tanto en el resultado final del trabajo, como en que el estudiante disfrute (o no) del proceso de realizarlo. Conozco de primera mano casos en que, por una mala tutorización, un tema que podría dar lugar a algo interesante termina siendo un trabajo hecho rápido para quitarlo del medio y pasar el trámite. Yo solo puedo pensar que al final llevo más de 9 meses sacando el mejor resultado posible, y es gracias al trato tan cercano, tan accesible y al final tan humano que he tenido con Chema; me has ayudado muchísimo y ha sido un verdadero placer tratar contigo durante este tiempo, tanto a nivel académico, como a nivel personal.

Mi otro pilar a nivel académico es sin duda mi tutora PAT Idoia, tuve la oportunidad de conocerla y hablar con ella antes de entrar en la carrera y de alguna forma me ayudó a tomar ciertas decisiones que han provocado que esté en el lugar donde estoy hoy en día. La conversación de aquel día es, de cualquier forma, simplemente la punta del iceberg, pues he tenido la suerte de poder charlar con ella varias veces por curso, dedicando Idoia su tiempo tanto en guiarme sobre en qué tenía que centrar mis esfuerzos hasta llegar al próximo objetivo que hubiera, como en ayudarme con cualquier problema que haya podido surgir por el camino. Tanto por el apoyo que ha supuesto, como por la confianza que creo que hemos compartido, considero haber conocido a Idoia una de las mayores suertes que he tenido en la carrera.

Por otro lado (y por suerte también), hay vida más allá del mundo académico, y ahí tengo una gran lista de familiares y amigos/as que de alguna forma son los causantes de que los días suelan ser soleados y de que haya una recompensa continua por transitar este camino que es la vida. Podría ponerme a enumerar nombres y más nombres, pero creo que tampoco tiene sentido, en algún punto tendría que trazar la línea de quién queda dentro y quién fuera, y creo que sería injusto. Además, no me cabe duda de que, para cualquier persona que leyendo estas palabras y se pueda preguntar si me habría acordado de el/ella, su nombre ya ha pasado por mi cabeza. A todas las personitas de esta lista imaginaria que en realidad está más definida de lo que pueda parecer, gracias de corazón por darme un empujoncito cuando lo he necesitado, un hombro cuando el empujoncito no habría llevado a ningún sitio y, sobre todo, por haberme dado siempre el máximo cariño posible.

Resumen

El *siteswap* es con diferencia la notación más extendida a la hora de designar patrones de malabares, y tiene la característica de que, además de hacer de “nombre”, define matemáticamente la secuencia de lanzamientos que se producen.

Estos fundamentos matemáticos han permitido que a lo largo de los años salgan al mercado varias aplicaciones capaces de, desde un *siteswap*, realizar una simulación del patrón, lo que permite visualizarlo detalladamente y ayudar a que cualquier persona que lo quiera aprender pueda comprenderlo. De cualquier forma, a pesar de lo extendidas que están estas aplicaciones dentro del ámbito de los malabaristas, no hay aplicaciones accesibles que hagan el proceso contrario, desde un vídeo extraer el *siteswap*.

Este Trabajo de Fin de Grado propone justamente el desarrollo de un sistema que realice esta función, desde la grabación de un malabarista ser capaz de extraer el *siteswap* que esté realizando. Para ello, hay dos tareas principales a cumplir, obtener la ubicación de las bolas de malabares en todo momento, y con esta información ser capaz de averiguar a qué *siteswap* corresponden estas detecciones.

Para la primera tarea se han explorado distintas técnicas de detección y seguimiento de varios objetos en vídeo, con el objetivo de seleccionar la que mejor funcione y desarrollar desde ahí la funcionalidad correspondiente a la parte de extracción del *siteswap*, probando distintos acercamientos capaces de obtener el orden en que se ha lanzado cada bola para poder extraer la secuencia matemática desde ahí.

Simultáneamente, ha sido necesario definir un *dataset* de vídeos de un malabarista realizando un *siteswap* sobre el que se ha hecho tanto el desarrollo como las pruebas.

Con ello, el sistema final desarrollado mejora considerablemente los resultados que obtienen otros sistemas de características similares en la parte detección y seguimiento de bolas de malabares, además de suponer el primer acercamiento público en lo relativo a la extracción del *siteswap*.

Palabras clave

Malabares, *siteswap*, patrón, secuencia, periodo, detección, *tracking*, *GroundTruth*, falso positivo, falso negativo, métrica, *dataset*, filtro de Kalman.

Abstract

The siteswap is by far the most widely used notation for designating juggling patterns, and it has the characteristic of not only serving as a “name” but also mathematically defines the sequence of throws that occur.

These mathematical foundations have allowed the publication of multiple applications capable of, from a siteswap, create a simulation of the pattern, which allows for a detailed visualization and assists anyone who wants to learn the pattern in understanding it. However, even if these applications are widespread among jugglers, there isn’t any application accessible that performs the reverse process, from a video extracting the siteswap.

This Bachelor Thesis precisely proposes the development of a system that performs this function, from a recording of a juggler being able to extract the siteswap being performed. To achieve this, there are two main tasks, obtaining the location of the juggling balls in all times, and with this information determining which siteswap these detections correspond to.

For the first task, different techniques of multi-object detection and tracking have been explored, with the aim of selecting the one which is performing better and developing the functionality for the extraction of the siteswap from there, testing various approaches to obtain the order in which each ball has been thrown, and extracting the mathematical sequence from there.

Simultaneously, it has been necessary to define, create and annotate a dataset of videos of a juggler performing a siteswap, which has been used for both the development and testing.

As a result, the final developed system significantly improves the results obtained by other similar systems in the part of ball detection and tracking, and it represents the first public approach regarding the extraction of the siteswap.

Keywords

Juggling, siteswap, pattern, sequence, period, detection, tracking, GroundTruth, false positive, false negative, metric, dataset, Kalman filter.

Índice

1 Introducción ... 1

1.1. Motivación ... 1

1.2. Objetivos ... 2

2 Estado del arte ... 3

2.1. *Vanilla Siteswap* ... 3

2.2. Sistemas de reconocimiento de video en el ámbito de los malabares ... 9

2.3. Simuladores de *siteswap* ... 15

3 Estado del arte ... 19

3.1. *Sistemas base* ... 19

3.2. *Diseño del sistema general* ... 20

4 Desarrollo y pruebas V0 ... 25

4.1. *Desarrollo* ... 25

4.2. *Pruebas* ... 33

5 Desarrollo y pruebas V1 ... 37

5.1. *Desarrollo* ... 37

5.2. *Pruebas* ... 37

6 Desarrollo y pruebas V2 ... 41

6.1. *Desarrollo* ... 41

6.2. *Pruebas* ... 44

7 Versión final y pruebas ... 47

7.1. *Descripción* ... 47

7.2. *Pruebas finales* ... 48

8 Conclusiones y trabajo futuro ... 51

8.1. *Conclusiones* ... 51

8.2. *Trabajo futuro* ... 51

Bibliografía ... 53

Apéndices ... 57

A - Módulos auxiliares ... 59

A.1. *Kalman_utils* ... 59

A.2. *Excel_utils* 59
A.3. *Mot16_utils* 60
A.4. *Visualizer* 60
A.5. *Metrics* 61
B - Filtros de Kalman 63
C - Parámetros utilizados en las pruebas 67
C.1. *Versión 0* 67
C.2. *Versión 1* 68
C.3. *Versión 2* 69
C.4. *Versión final* 69

Índice de figuras

2.1 Diagrama representación cascada de 3 bolas. *Siteswap* 3. 3
2.2 Diagrama representación paridad *siteswap*. 6
2.3 Representación alturas intuitivas *siteswap*. Extraído de: [11]. 7
2.4 Representación colisión tras un lanzamiento N valores menor N turnos después (con N=1). 8
2.5 Ejemplos del *siteswap* 3 en otros tipos de notación. Extraído de: Libro de Ben Beever [12], [13] 9
2.6 Captura de la aplicación de Danielo Amaya [14]. 10
2.7 Captura de la aplicación de Gaya Goredecki [15]. 11
2.8 Captura de la aplicación de Nathaniel Guy [16]. 11
2.9 Captura de la aplicación de Barton Chittenden [18]. 12
2.10 Captura de la aplicación de Rasmus Åkerlund [21]. 14
2.11 Captura de la aplicación de Stephen Meschke [24]. 14
2.12 Captura de la aplicación de James Cozens [25]. 15
2.13 Captura de los primeros simuladores de *siteswap*, disponible en [29]. 16
2.14 Captura de los simuladores actuales, por orden: *Siteswap animator*, *Juggling simulator* y *Gunswap*. 16
2.15 Captura de *Juggling Lab* en su versión para Windows (Izquierda y centro) y Android (Derecha). 17
3.1 Diagrama de diseño del sistema. 20
3.2 Imágenes ejemplo *dataset*. 22
4.1 Visualización de la ejecución del programa de *tracking* manual. 28
4.2 Ejemplo supresión de no máximos en una etapa temprana del desarrollo. 29
4.3 Visualización ejecución programa Barton Chittenden. 30
4.4 Análisis posición de una bola para los *siteswaps* 5 y 441. 32
5.1 Ejemplo optimización distancia para considerar mismo identificador. 38
6.1 Mapa de calor *siteswap* 5. 42
6.2 Mapa de calor *siteswap* 5 marcado. 43
A.1 Captura ejecución *tracking_visualizer* 60
B.1 Diagrama base filtro de Kalman. Extraído de: [44]. 63
B.2 Diagrama completo filtro de Kalman. Extraído de: [44]. 65

Índice de tablas

4.1 Resultados *color_tracking* con preprocesado V0. 33
4.2 Resultados *color_tracking* sin preprocesado V0. 34
4.3 Resultados *bg_substraction* V0. 35
4.4 Resultados *seq_extraction* basado en coordenadas desde el *GroundTruth* V0. 35
4.5 Resultados *seq_extraction* basado en coordenadas desde el mejor resultado de *tracking* V0. 36
5.1 Resultados *color_tracking* con preprocesado V1. 38
5.2 Resultados *color_tracking* sin preprocesado V1. 39
5.3 Resultados *seq_extraction* basado en coordenadas desde el mejor resultado de *tracking* V1 40
6.1 Resultados *seq_extraction* basado en cuadrantes desde el *GroundTruth* 45
6.2 Resultados *seq_extraction* basado en cuadrantes desde el mejor resultado de *tracking* V2 45
7.1 Resultados finales del sistema. 48

Índice de ecuaciones

Ecuación B.1..... 64
Ecuación B.2..... 64
Ecuación B.3..... 64
Ecuación B.4..... 65

1

Introducción

1.1. Motivación

El *siteswap* [1] es la notación más extendida para describir cualquier patrón de malabares, y aunque es cierto que a veces hay trucos que pueden recibir nombres con palabras, estos cambian en función del país o incluso del propio malabarista, de forma que el *siteswap* es además la única notación existente suficientemente extendida como para poder asumir que se puede producir una comunicación técnica con cualquier otro malabarista.

Como notación, tiene la ventaja de que define matemáticamente la secuencia de lanzamientos que sigue el patrón, de forma que ya existen aplicaciones (y están bastante extendidas dentro de la comunidad) [2] que a partir del *siteswap* generan simulaciones de los patrones, permitiendo a los malabaristas estudiarlos para aprender a realizarlos, investigar trucos nuevos al obtener secuencias que matemáticamente son correctas, o incluso teorizar sobre nuevas corrientes dentro del malabarismo con la simulación de trucos que en un principio son meramente teóricos.

De cualquier forma, no hay ninguna aplicación accesible capaz de, desde el vídeo de un malabarista, extraer el *siteswap* que está realizando. Esto podría ser de bastante utilidad para la comunidad, pues a nivel internacional se realizan un gran volumen de publicaciones en redes sociales e intercambio general de contenido audiovisual sobre malabares que en muchas ocasiones no van acompañados del propio *siteswap*, lo que dificulta el proceso de poder imitar ciertos patrones que puedan resultar más atractivos.

Este trabajo se centra en desarrollar precisamente esta idea y crear un sistema capaz, tanto de hacer un seguimiento de las bolas del malabarista, como de extraer la información necesaria de ese seguimiento para calcular el *siteswap* correspondiente.

Idealmente, el sistema debería ser capaz de poder clasificar cualquier vídeo de uno o más malabaristas con todos los *siteswaps* que se realicen, pudiendo ser a su vez *siteswaps* que incluyeran variaciones y lanzamientos síncronos. Sin embargo, el alcance real propuesto (abarcable dentro del marco de un TFG) es que en el vídeo de entrada haya solo un malabarista que realice un único siteswap con todos los lanzamientos asíncronos (solo se lanza un objeto a la vez), internos (la trayectoria es de dentro a fuera) y sin variaciones (no hay lanzamientos por detrás de la espalda o similares), de forma que el sistema solo tenga que calcular un *siteswap* que sea lo más claro posible.

1.2. Objetivos

Objetivos:

- Realizar un análisis en profundidad tanto de la situación del *siteswap* en la actualidad como de otros sistemas de características similares.
- Diseñar e implementar un sistema capaz de extraer el *siteswap* que está ejecutando un malabarista en un vídeo. Concretamente, deberá ser capaz de:
  - Realizar tanto detección como seguimiento (*tracking*) de las bolas de malabares presentes en un vídeo en condiciones ideales que serán definidas durante el desarrollo del proyecto.
  - A partir de la información de la posición de las bolas a lo largo del vídeo, ser capaz de extraer el *siteswap* que se está realizando y devolverlo.
- Diseñar, grabar y anotar un *dataset* especializado en *siteswaps* de malabares según las condiciones ideales definidas.

2

Estado del arte

En la presente sección se abordará una revisión de los temas más representativos de este trabajo; por un lado, el *Siteswap* (*Vanilla Siteswap* realmente, que es la base del *Siteswap* actual y sobre la que se va a trabajar), por otro, los distintos proyectos de reconocimiento de vídeo en malabares que han tenido mayor éxito y se encuentran más cerca del objetivo propuesto, y finalmente, se introducirán los simuladores de *siteswap*, aplicaciones bastante extendidas dentro del sector que desde el *siteswap* son capaces de representar visualmente el patrón en vídeo.

2.1. *Vanilla Siteswap*

2.1.1. Consideraciones previas

Tanto en este apartado como en el resto de la presente memoria se van a tomar dos licencias con el fin de que el texto sea lo más legible posible. La primera de ellas es hablar de bolas de malabares en todo momento, en vez de "objetos", "material" o "*props*", aunque a nivel teórico todo lo que se exponga en este trabajo es posible con otro tipo de elementos tales como mazas, aros, sombreros… La segunda de ellas es que dentro del *siteswap* la mayor parte de las veces no se va a hacer la distinción de que un carácter de un *siteswap* puede ser un número o una letra, simplemente se dirá número y en caso de que sea una letra se asumirá su valor correspondiente (Sección 2.1.3).

Cuando se habla de "cascada de N bolas" como truco, es el patrón más sencillo que se puede hacer con N bolas (su notación en *siteswap* simplemente es un carácter con el valor de N). El patrón más característico de malabares es la cascada de 3 bolas, por ser el primer patrón que se suele enseñar (ver figura 2.1).

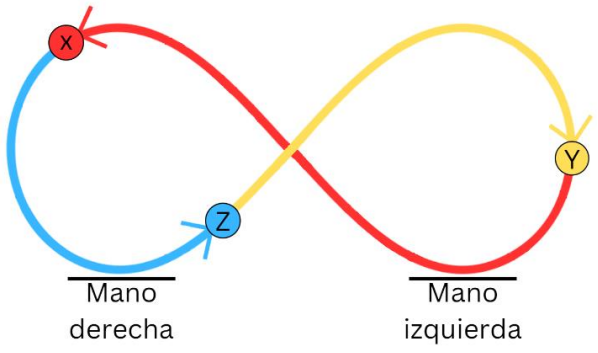

Figura 2.1: Diagrama representación cascada de 3 bolas. *Siteswap* 3.

Por último, mencionar la existencia de simuladores de *siteswap* [2],[3] (Ver sección 2.3) que pueden servir de apoyo al texto de la presente memoria en caso de tener dificultades para visualizar alguna secuencia. Dichos simuladores son abordados en futuros apartados, pero vale con introducir el *siteswap* correspondiente para ver una simulación de una persona realizando ese patrón de malabares.

2.1.2. Origen

La invención del *siteswap* [4] se atribuye a Paul Klimek de California en 1981, aunque por aquel momento se llamaba "*Quantum Juggling*" debido a una analogía que hizo entre los átomos con su núcleo rodeado de nubes de electrones que subían y bajaban como si fuera un malabarista con sus bolas.

Paul Klimek se dio cuenta de que al mirar a un malabarista desde el lateral sin fijarse en las manos era capaz de observar ciclos. Trabajando en esa idea sumó el tiempo de vuelo al de sujetar la bola en la mano y se dio cuenta de que era un múltiplo integral del tiempo entre lanzamientos [5]. Desde ahí desarrolló la notación que conocemos ahora.

De cualquier forma, Paul no fue el único en hacer este descubrimiento. En algún momento antes de 1986 Paul comenta a un amigo suyo su notación "*Quantum Juggling*", y este le pone en contacto con Don Hatch, pues había descubierto algo parecido. Efectivamente ambos usaban y comprendían el mismo sistema, y Don Hatch también lo había descubierto en torno a 1981 [6].

También de forma independiente, en 1985 en Cambridge, Mike Day, Colin Wright y Adam Chalcraft llegan a un sistema muy similar, salvo porque consideraban todas las cifras con un valor menos, aunque probablemente de haber seguido más tiempo con su sistema sin saber del "*Quantum Juggling*" habrían añadido ese valor para poder representar los huecos y que se cumplieran ciertos principios matemáticos [7].

Sobre esos años, Bruce Tiemann, un malabarista americano descubre un sistema para generar trucos al que llama *siteswap* (que no tenía nada que ver con la notación que conocemos hoy día), que se basaba en el concepto de que cada permutación distinta de números implica un truco distinto de malabares para cada número posible de objetos. Este concepto lo usaba para generar trucos nuevos mediante cambios de sitio en esas permutaciones, de ahí *site-swap* [8]. No servía como notación, aunque Bruce tenía su propio sistema de escribir los trucos basado en los giros de las mazas y en si los lanzamientos cambiaban o no de mano [7].

En 1986 Paul y Bruce se conocen, e intercambian sus descubrimientos. Paul deja a Bruce explicar primero su sistema, para asegurarse de que lo que le contaba no estaba influenciado por sus propios descubrimientos [5]. Una vez Bruce contó su forma de generar los trucos, Paul procedió a contarle su forma de anotarlos, cosa que para Bruce fue un descubrimiento, y de inmediato lo consideró superior a su propia notación [7].

Tras esta conversación Paul fue a buscar al mejor malabarista del momento (y por muchos considerado el mejor malabarista de la historia), Anthony Gatto, pues llevaba

años queriendo hablar con él para tratar de popularizar su sistema de notación (que de momento seguía llamándose "*Quantum Juggling*"). Tras contárselo, Anthony Gatto le contesta que está bien, pero que otra persona le acababa de contar lo mismo de los números [5]. Esta persona fue el propio Bruce, que habló con Anthony y le contó el sistema de Paul pero sin dejar claro que no era su propia invención, aunque de cualquier forma Anthony le despachó porque no le interesaba mucho, ya que tenía "sus propias ideas sobre cómo hacer trucos" [7].

De cualquier forma, desde estas conversaciones se empieza a extender este sistema de definir los patrones con números, pero lo hace bajo el nombre de *siteswap*, en vez de "*Quantum Juggling*", ya que Bruce no hace grandes esfuerzos por desmentir que no fue invención suya [9].

Dado a que (como se discutirá más adelante) el *siteswap* no puede definir todos los patrones, en 1990 se completó la notación, naciendo así el *Aumented Siteswap*, que terminó de completarse por Ben Beever en la misma década [4].

2.1.3. Elementos

Como se ha mencionado anteriormente, los *siteswaps* son cadenas alfanuméricas que representan los patrones de malabares y hacen las veces de partitura para los malabaristas. Algunos ejemplos podrían ser "3", "645" o "db97531".

En estas secuencias cada carácter representa un lanzamiento, y su valor representa el número de tiempos que tarda esa bola en volver a ser recogida. En el caso de que el carácter sea una letra, simplemente representa un número de más de una cifra, pues el alfabeto que se utiliza es {1,2,3,4,5,6,7,8,9,a,b,c,d,...}. Se entiende como un tiempo el periodo entre un lanzamiento y el siguiente (de cualquiera de las manos). A nivel teórico todos los tiempos miden lo mismo, aunque a nivel práctico si se analiza el vídeo de un malabarista se pueden encontrar pequeñas variaciones entre los tiempos (debido a imprecisiones por no poder replicar ejecuciones perfectas).

El *siteswap* presupone además que los lanzamientos se realizan alternando las manos. De esta forma, siendo R la mano derecha y siendo L la mano izquierda, todos los *siteswaps* se efectúan siguiendo [R,L,R,L,R,L,R,L,…] (o cualquier secuencia en la que R y L se alternan). Aunque tanto a nivel teórico como práctico se suele considerar una notación para dos manos, también es válida para un número mayor, de forma que si un malabarista tuviera tres manos hábiles (o usara un pie y las dos manos, por ejemplo) se podrían seguir anotando los trucos igualmente. Otro ejemplo es el *passing* (malabares entre varias personas), que para su notación utiliza una ampliación del *Vanilla Siteswap* con pequeñas ampliaciones, pero la misma base.

Así, si tuviéramos 3 bolas "x", "y" y "z" y quisiéramos hacer la cascada de 3 bolas, primero lanzaríamos "x" con la mano derecha, luego "y" con la izquierda, luego "z" con la derecha, después "x" con la izquierda y así sucesivamente. Escribiendo la secuencia queda como [x,y,z,x,y,z,x,y,z,...], pudiendo verse que cada bola se lanza 3 tiempos después de la última vez que fue lanzada, es decir, cada bola hace un 3 en notación

siteswap. Otro ejemplo un poco más complicado sería tratar de lanzar las 3 bolas en la secuencia [x,y,z,z,x,y,y,z,x,x,y,z,z…]. Podemos observar que la bola “x” se lanza por segunda vez 4 tiempos después de haber sido lanzada, la bola “y” igual, y la bola “z” un solo tiempo después. Por ello, en notación *siteswap* ese patrón sería 441, de forma que la bola “x” se lanza cada 4,4,1,4,4,1 tiempos, la bola “y” cada 4,1,4,4,1,4,4,1,4 tiempos y la bola “z” cada 1,4,4,1,4,4,1,4,4,1. Como regla general “Lanzar un (valor de *siteswap*) N, significa que esa misma bola será lanzada de nuevo N lanzamientos después” [10].

Habiendo visto este último ejemplo (441), es destacable cuando una bola hace un 4 siempre vuelve a la misma mano que la lanzó, mientras que cuando una bola hace un 1 cambia de mano. Esto se debe a que, en el *siteswap*, todos los lanzamientos representados por un número par vuelven a la misma mano que las lanzó, mientras que los lanzamientos representados por un número impar cambian de mano. De forma analítica, teniendo en cuenta que las manos derecha e izquierda se alternan, si la próxima vez que lanzas una bola corresponde a un número par, la bola tendrá que volver a caer en la misma mano, mientras que en caso de que fuera impar tiene que caer en la otra mano (ver Figura 2.2)

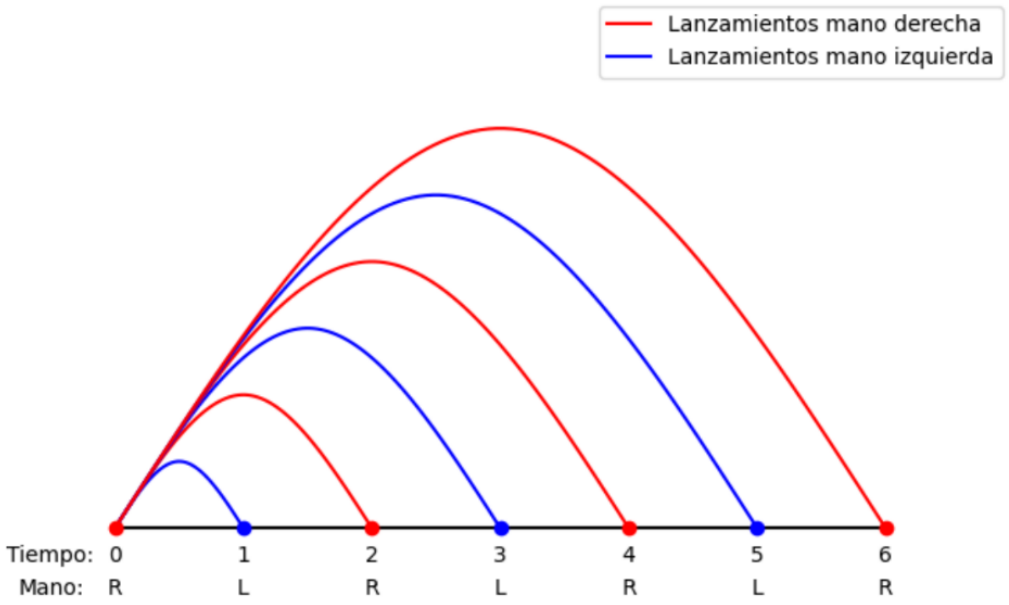

Figura 2.2: *Diagrama representación paridad siteswap.*

Intuitivamente, dado que valores mayores representan un mayor número de tiempos para recoger las bolas, esto se suele traducir en usar mayor altura en lanzamientos de cifras más grandes (ver Figura 2.3). De cualquier forma, cabe destacar que, aunque útil en la práctica, técnicamente esta es una idea equivocada bastante generalizada, ya que los números hacen referencia a una cantidad de tiempos, no a una altura. Por ejemplo, un 5 podría traducirse en colocar una bola encima de una mesa con una mano y recogerla con la otra en el transcurso de 5 tiempos, lo que no tiene relación a ninguna altura.

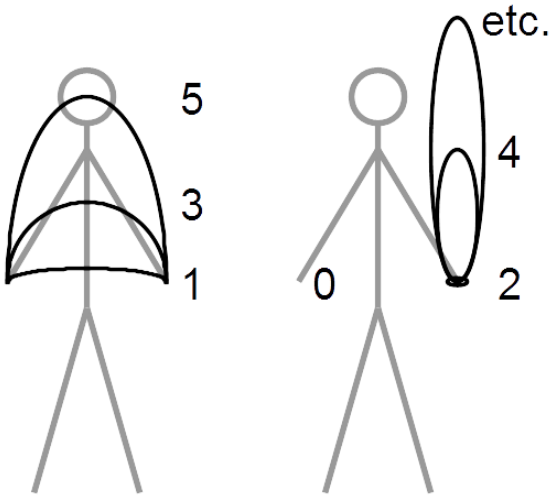

Figura 2.3: *Representación alturas intuitivas siteswap. Extraído de: [11].*

Aun sabiendo cómo se comportan valores pares e impares, hay dos valores con características concretas que se deben matizar. Por un lado, tenemos los 2's que, dado que van a ser lanzados con la misma mano en el siguiente lanzamiento que haga (ya que tarda 2 tiempos en volver a la mano y se alternan R y L), no es necesario soltarlos de la mano, de forma que generalmente simplemente se sostienen (2 pasivo), aunque si se quiere se pueden lanzar (2 activo, a veces denotado como 2T). El otro caso especial son los 0's, que significaría que "la siguiente vez que lanzas esta bola es en 0 lanzamientos". Dado que no puedes realizar un lanzamiento "0 tiempos después", en realidad termina representando la ausencia de bolas en una mano. Así, si en un patrón se lanzan dos bolas seguidas con la misma mano, aunque pueda parecer que va en contra del principio de alternar R y L que se había discutido, no es el caso, ya que la mano que no hace un lanzamiento estará haciendo un 0. Este comportamiento se puede encontrar en *sitwswaps* como 50505 o 55550.

Dado que un patrón se puede repetir infinitamente, la forma que tenemos de anotarlo es usar la menor cantidad de caracteres posibles que represente todo el patrón de forma que no se repita, por ello, en vez de 3333… escribimos 3, en vez de 441441441 escribimos 441 y así sucesivamente. Desde aquí es fácil entender el concepto del periodo de un *siteswap*, que es el número de lanzamientos que se producen antes de que se repita el patrón, de forma que por ejemplo el 441 tiene un periodo de 3. Cabe distinguir también el "periodo completo", que sigue la misma idea pero distingue que realmente sólo se repite el patrón si todas las bolas hacen los mismos lanzamientos repetidos, de forma que 441 tiene un periodo completo de 9 (ya que el patrón hace [x,y,z,z,x,y,y,z,x] antes de repetirse completamente).

2.1.4. Reglas y propiedades

Aun sabiendo que los *siteswaps* son cadenas alfanuméricas que representan patrones de malabares, no todas las cadenas alfanuméricas representan un patrón que sea posible. Para ello debe seguir dos reglas necesarias, pero no suficientes:

• Si se suman los valores de cada carácter y se divide entre el número de caracteres, el resultado debe ser un valor entero, que corresponde con el número de bolas con las que se realiza el *siteswap*. Es decir, al realizar la media de la secuencia se debe obtener un número natural.

• Ninguna cifra puede estar seguida N turnos después por otra cifra que sea N valores menor. Esto es así porque en caso de producirse caerían las dos bolas a la vez; por ejemplo, si lanzamos un “43” el 3 caería en el mismo tiempo y mano que el 4 (ver figura 2.4).

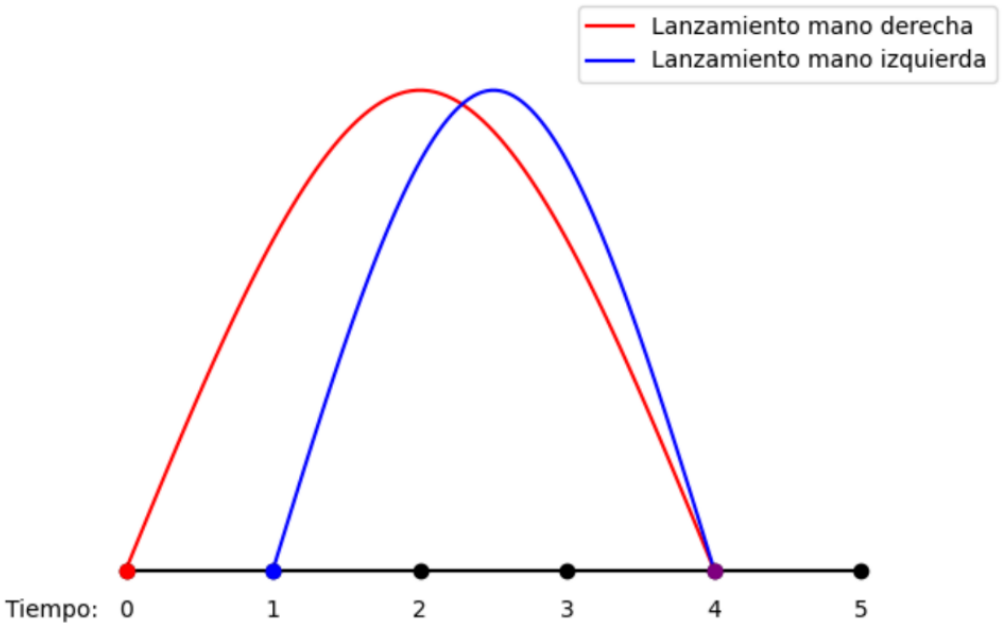

Figura 2.4: *Representación colisión tras un lanzamiento N valores menor N turnos después (con N=1).*

Estas dos propiedades aseguran que el *siteswap* sea válido, pero no necesariamente asegura que puedan ser iniciados desde una cascada, es decir, necesitan de lanzamientos que les hagan de “puente” para entrar y/o salir del *siteswap*. Estos patrones se conocen como patrones excitados, nombre que reciben del “*Quantum Juggling*” en referencia a los átomos excitados.

2.1.5. Posibilidades y limitaciones

A pesar de que el *Vanilla Siteswap* define una infinidad de patrones, tiene algunas limitaciones, como pueden ser el hecho de no poder definir todos los patrones o no definir todas las variaciones que pueden suceder en el mismo patrón.

Sobre no poder definir todos los patrones, el V*anilla Siteswap* solo define aquellos en los que se gestiona una bola por tiempo, lo que deja fuera trucos en los que las dos manos lanzan a la vez (síncronos), una mano lanza dos o más bolas a la vez (*multiplexes*) o aquellos en los que dos o más bolas caen a la vez en la misma mano (*squeeze catches*). Para solucionarlo entran en juego notaciones que amplían el *siteswap*, como el *siteswap* extendido [4].

Por otro lado, es verdad que el V*anilla Siteswap* no define todos los aspectos de un patrón, ya que no ofrece información sobre con qué se hacen los malabares (e.g., bolas, mazas, aros), lugar por donde se hacen los lanzamientos (e.g., debajo de las piernas, detrás de la espalda), la trayectoria de las bolas (e.g., externas, internas), rotaciones que puedan hacer los objetos (e.g., giros según el eje vertical, según el eje horizontal) u otros elementos añadidos al patrón, como puede ser realizarlo mientras se equilibra otro objeto o añadir giros del cuerpo. De cualquier forma, esto no está considerado como un problema, ya que los tiempos del patrón sí que quedan perfectamente definidos, por lo que los malabaristas pueden explorar y a partir de un truco (por ejemplo 441) hacer variaciones (por ejemplo 441 pero los 4's lanzarlos por detrás del hombro).

2.1.6. GS y otras notaciones

Cabe también mencionar la existencia de otras notaciones, tales como el GS creado por Ben Beever [10], consistente en una matriz que, a diferencia del *siteswap*, define múltiples aspectos de los patrones en vez de quedarse solo en los tiempos, o los diagramas de escalera, en los que se representa gráficamente los tiempos y trayectorias de las bolas.

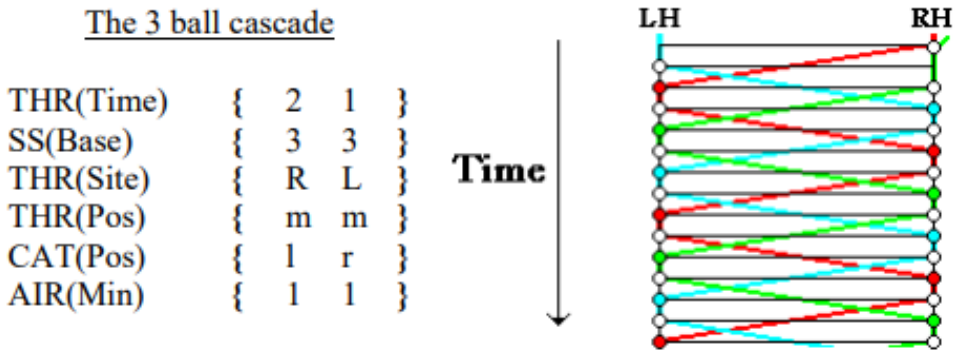

Figura 2.5: *Ejemplos del siteswap 3 en otros tipos de notación. Extraído de: Libro de Ben Beever [12], [13] .*

De cualquier forma, existe una gran variedad de otros sistemas de notación de patrones de malabares que han sido menos influyentes o simplemente no han llegado a popularizarse en absoluto.

2.2. Sistemas de reconocimiento de video en el ámbito de los malabares

Desde hace años la comunidad del malabarismo ha ido desarrollando distintos sistemas para potenciar esta disciplina y tratar de servir de apoyo para todos los malabaristas del mundo. Cabe destacar la influencia de los simuladores de *siteswaps*, cuyos primeros desarrollos se remontan a la década de los 90 y que desempeñan un funcionamiento inverso al que se propone en este proyecto; a partir de un *siteswap*, generar una simulación en la que se pueda ver el patrón que describe. Estas herramientas han ido evolucionando, y hoy en día permiten ajustar parámetros y ver tanto detalle que

para muchos malabaristas se han convertido en un pilar fundamental de aprendizaje (ver sección 2.3).

Sobre sistemas más directamente relacionados con este Trabajo de Fin de Grado, se han encontrado varios proyectos que están orientados en direcciones similares, y aunque la mayoría de ellos únicamente implementan detección de bolas de malabares, en algunos casos llegan a intentar catalogar el patrón. A continuación, se detallan algunos de los más relevantes.

2.2.1. Detección de bolas – Danielo Amaya

Danielo Amaya es un artista circense colombiano que durante 2017 estuvo desarrollando distintos sistemas de detección de bolas principalmente en C++ [14].

Aunque para sus sistemas tenía que definir manualmente el color de las bolas de malabares para poder aplicar una máscara, sus sistemas parecen funcionar considerablemente bien incluso para grabaciones en vivo, en especial en entornos con buenas condiciones de iluminación.

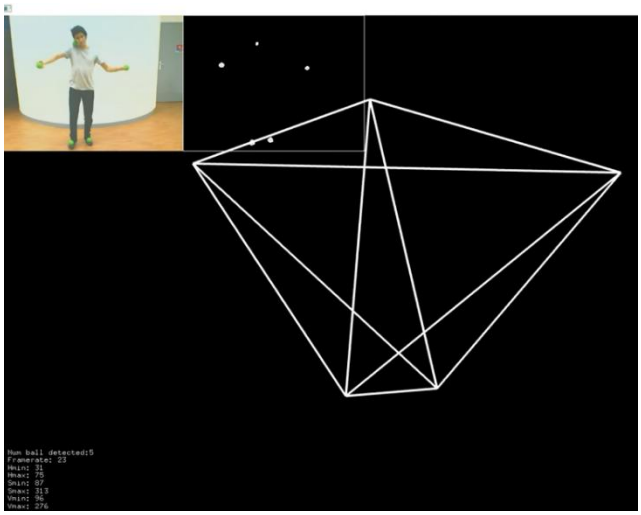

Figura 2.6: *Captura de la aplicación de Danielo Amaya [14].*

2.2.2. Seguimiento de bolas y extracción velocidad – Gaya Goredecki

Como proyecto final de un curso de Visión Artificial, Gaya Goredecki diseñó y codificó un proyecto sobre un *notebook* capaz de seguir tres bolas de malabares sobre un vídeo y extraer sus respectivas velocidades [15].

Para ello utiliza umbrales de color para cada bola (que deben ser de colores distintos), y desde la información obtenida muestra sobre cada *frame* la ubicación de la bola y una estela de su trayectoria. Para las velocidades simplemente calcula la distancia desde el punto anterior al actual y lo divide por los *fps* (*frames per second*) como tiempo unidad. Además, almacena las velocidades para poder mostrar las aceleraciones dentro del patrón.

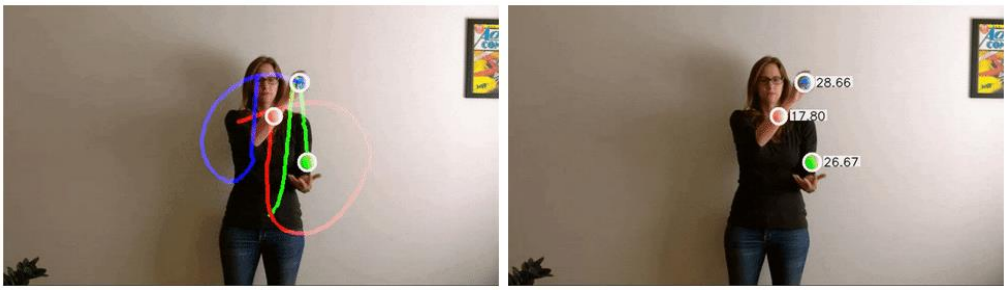

Figura 2.7: *Captura de la aplicación de Gaya Goredecki [15].*

2.2.3. Seguimiento de bolas y predicción trayectoria – Nathaniel Guy

Nathaniel Guy es un ingeniero aeroespacial con un máster doble en ciencias de la computación e ingeniería aeronáutica/astronáutica. En 2015 desarrolló un sistema que, además de *trackear* las bolas de un malabarista (de las que también tenías que indicar el color manualmente), era capaz de predecir su trayectoria, tanto en tiempo real como en vídeos pregrabados [16], [17].

Para ello, por cada *frame* primero hacía substracción de fondo y después al resultado le aplicaba umbrales de color para extraer los puntos de interés y aplicaba varias transformaciones como erosión, dilatación y rechazo de valores atípicos. Con ello, y a través de *clustering* de k-vecinos estimaba los centros de las bolas y sus posiciones relativas.

Una vez tenía localizadas las bolas en cada *frame*, y a través de un sistema de estados actuales y predicciones propagadas, usa el método de Euler para mostrar en el vídeo una trayectoria futura estimada.

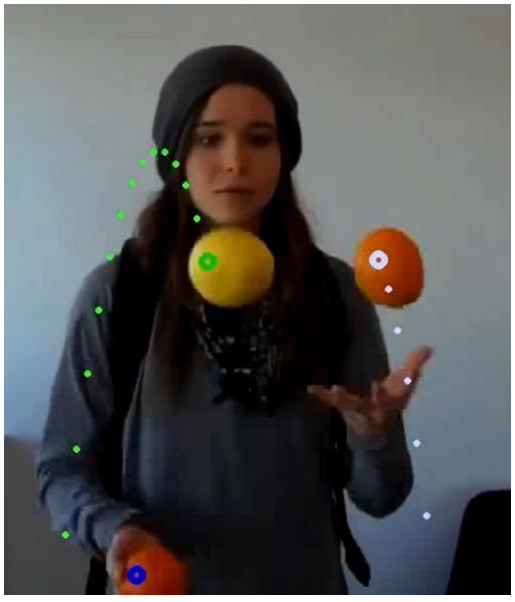

Figura 2.8: *Captura de la aplicación de Nathaniel Guy [16].*

2.2.4. Seguimiento de bolas – Barton Chittenden

Barton Chittenden es un ingeniero de *DevOps* estadounidense que durante 2022 estuvo trabajando en un sistema de seguimiento de bolas [18] que incluye una gran cantidad de opciones de configuración con el objetivo de poder adecuar el seguimiento al vídeo lo máximo posible. Algunas de estas opciones son elegir *frames* de inicio y fin del *trackeo* dentro del vídeo, delimitar la zona de *tracking* para eliminar ruido exterior y cambiar la distancia máxima entre *frames* que puede haber para considerar dos detecciones bajo el mismo identificador.

Para las detecciones a cada *frame* le aplica substracción del fondo, y pasando el resultado por umbrales de tonos de gris y aplicándole detección de contornos llega a una lista de objetos *trackeados* sobre los cuales tiene las coordenadas y consecuentemente el resultado deseado.

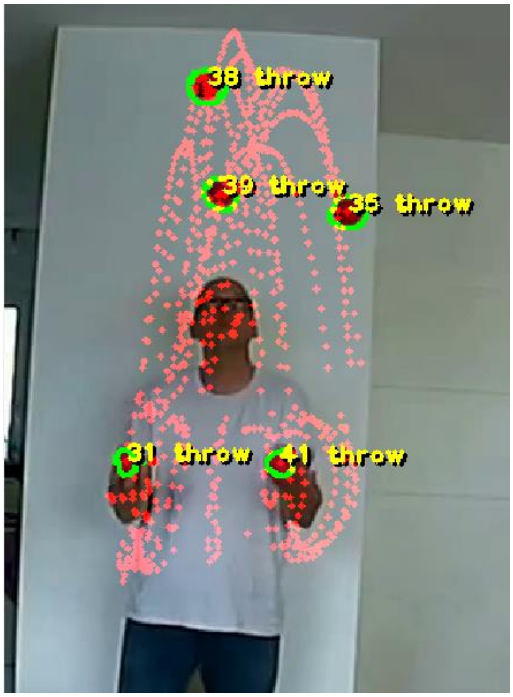

Figura 2.9: *Captura de la aplicación de Barton Chittenden [18].*

2.2.5. Seguimiento de bolas y extracción del *siteswap* – Tim Dresser

Tim Dresser se enfrentó a un proyecto con exactamente el mismo planteamiento que este Trabajo de Fin de Grado: a partir del vídeo de un malabarista extraer el *siteswap* que está ejecutando. De cualquier forma, por desgracia no se dispone de imágenes ni código fuente, simplemente de la información que ha compartido [19]. Para enfrentarse al proyecto, identificó dos subproblemas distintos; *trackear* las bolas, y generar el *siteswap* desde la información obtenida.

A la hora de *trackear* las bolas, Tim se encontró con la dificultad de que, para cualquier patrón moderadamente complejo, había involucrados lanzamientos considerablemente altos, lo que implicaba tener que hacer planos más amplios y

consecuentemente que las bolas ocuparan menos píxeles. Sus acercamientos tenían algunos errores por esta razón, no podía *trackear* las bolas cuando eran pequeñas.

En la parte de extraer los *siteswaps*, tuvo que lidiar con el problema de que a veces las manos esconden las bolas, por lo que perdía su identidad. En su caso, lo solucionó requiriendo bolas de distintos colores y que el malabarista las identificara al comienzo de la secuencia dibujando un rectángulo por delante con la mano.

Ahora bien, de cualquier forma, se encontró con otro problema, los 2's pasivos generan ambigüedades a la hora de predecir el *siteswap*, de forma que 522 es muy similar a 3, en especial cuando cambias los tiempos de permanencia en la mano (*dwell time*) para el 522. Por ello, Tim asegura que, si volviera a hacer un desarrollo similar, definiría una función de coste para medir la distancia entre las trayectorias obtenidas y un *siteswap* válido, y luego encontrar el *siteswap* minimizando esa función de coste.

2.2.6. Seguimiento de bolas, manos, creación de *dataset* y extracción de velocidad – Rasmus Åkerlund

Rasmus Åkerlund inició hace 4 años un proyecto con el objetivo de crear y entrenar una red neuronal convolucional que pudiera localizar las manos de un malabarista y un número conocido de bolas en tiempo real sin aceleración de *GPU*. Dicha red debería funcionar para combinaciones razonables de bolas, ropas, fondos e iluminación. Para ello, además creó un *dataset* de malabarismo [20] y entrenó pequeñas redes neuronales para catalogar los 36 patrones de malabares que componían el *dataset* a partir de la información obtenida de la red neuronal convolucional [21], [22].

En mayor detalle, el *dataset* consiste en 54000 *frames* obtenidos de 180 vídeos de 10 segundos y contiene cantidades iguales de vídeos con una, dos y tres bolas con las anotaciones de las coordenadas x e y de cada bola y mano en cada *frame*. El *dataset* se divide en un conjunto de entrenamiento de 144 vídeos, un conjunto de validación de 18 y un conjunto de test de otros 18, tratando siempre de que cada conjunto tenga una variedad similar de bolas, ropas, malabaristas, fondos e iluminación.

El proyecto terminó con un sistema que cumplía los objetivos, y era capaz de clasificar patrones de malabarismo comunes con una, dos y tres bolas basándose en la localización de las bolas y las manos, a pesar de que tenía algunas limitaciones como los lanzamientos síncronos, o encontrar coincidencias en algún patrón del *dataset* cuando se usaba algún patrón que la red neuronal desconocía.

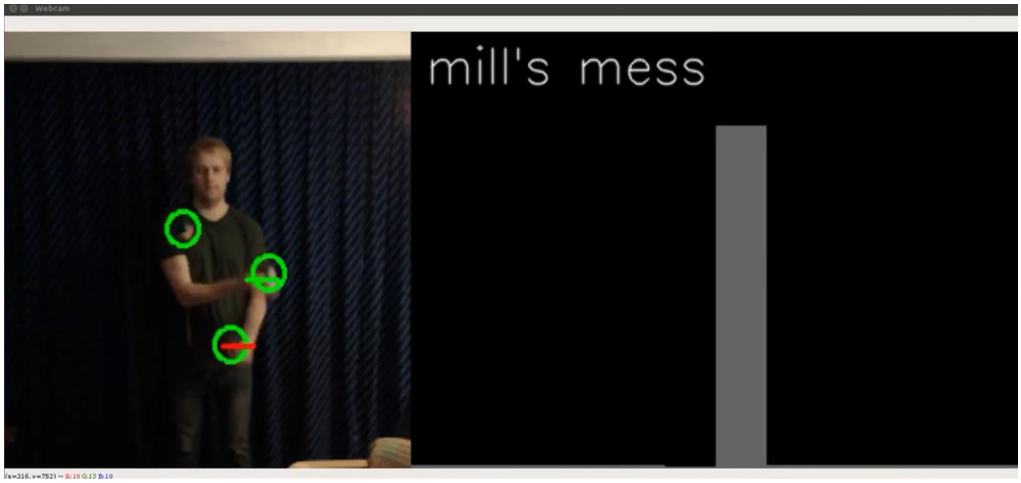

Figura 2.10: *Captura de la aplicación de Rasmus Åkerlund [21].*

2.2.7. Seguimiento de bolas, creación de *dataset*, extracción de trayectorias y del nombre del truco – Stephen Meschke

Stephen ha explorado varios acercamientos para el *trackeo* de bolas, como es el caso de usar el color de las bolas (que en su opinión es la forma más sencilla y precisa, pero requiere de alta resolución y movimientos lentos, además de buena iluminación), *Optical Flow Tracking* (que requiere de altas tasas de *fps*, al menos 60 *fps* para *trackear* 5+ bolas) o distintos acercamientos híbridos que pretenden solucionar los problemas de los anteriores [23], [24].

Además, Stephen es el autor de otro *dataset* sobre detección de patrones de malabares, en este caso también de 180 vídeos, pero esta vez con la mayoría de unos 1000 *frames*. A partir de este *dataset* ha sido capaz de crear un sistema que fijándose en la forma de las trayectorias que han seguido las bolas indica el nombre del patrón que sigue el malabarista de entre 5 posibles.

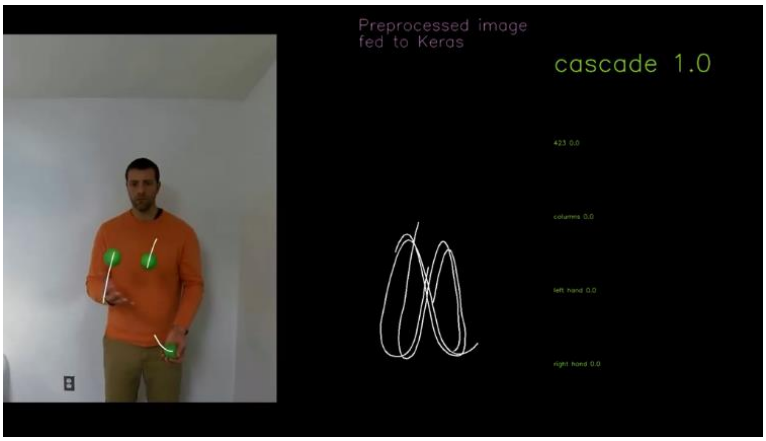

Figura 2.11: *Captura de la aplicación de Stephen Meschke [24].*

2.2.8. Seguimiento de bolas y extracción del *siteswap* – James Cozens

James Cozens es un estudiante de doctorado en la universidad de Cambridge con estudios en Ingeniería de la Información, particularmente en *Machine Learning*, procesamiento de señales y visión artificial.

Comenzó a investigar sobre la detección de *siteswaps* para uno de sus proyectos de máster hace más de un año, y desde entonces ha continuado ese proyecto que es capaz de, desde un vídeo de un malabarista efectuando *siteswaps* distintos de forma continua, ir haciendo el seguimiento de las bolas y sacando el valor de *siteswap* de cada lanzamiento individual.

La primera muestra de su proyecto tuvo lugar el 8 de octubre de 2022, en el que muestra una detección perfecta de *siteswaps* sobre un vídeo de 47 segundos en el que se efectúan mezclados 3 *siteswaps* distintos en condiciones óptimas [25].

Según el propio James Cozens [26], para vídeos de esas características (compuestos por *siteswaps* de 4 pelotas en condiciones óptimas) su sistema logra un 90% de acierto en las detecciones, aunque el límite de su funcionalidad parece estar en siteswaps de 7 bolas, lo que por otra parte podría ser su principal uso. No obstante, tiene otra muestra [27] en la que se ve un seguimiento que parece ser perfecto del récord mundial actual de 11 bolas con 34 recogidas, aunque solo hace la parte del seguimiento de bolas, no la obtención del *siteswap* (que sería “b” en este caso).

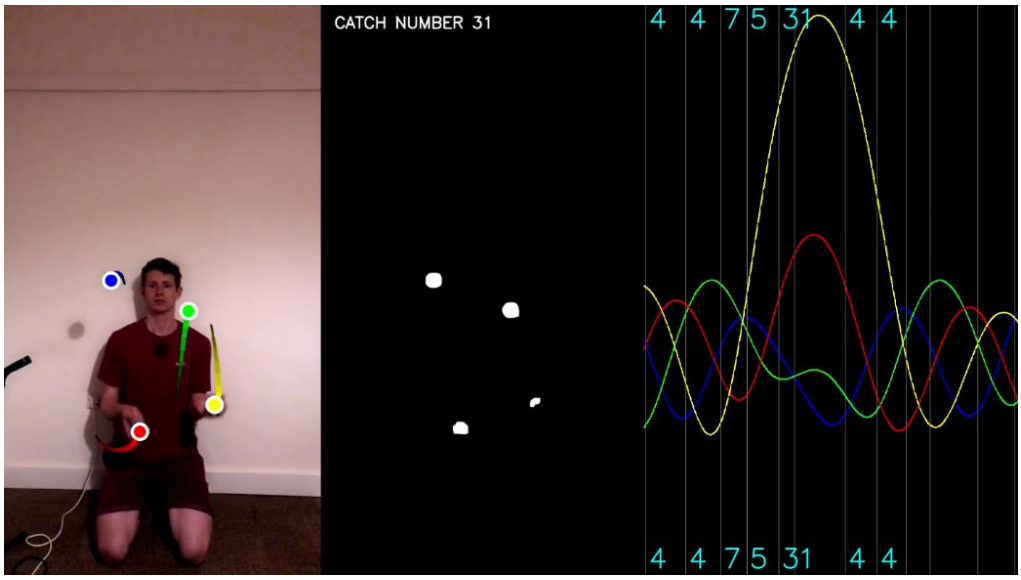

Figura 2.12: *Captura de la aplicación de James Cozens [25].*

2.3. Simuladores de *siteswap*

A partir de la década de los noventa los malabaristas comenzaron a explorar formas de representar por ordenador los distintos *siteswaps*. Así es como surgen distintos simuladores que, hasta el día de hoy, han permitido a los malabaristas poder observar los distintos patrones sin que sea necesario si quiera que nadie en el mundo sea capaz de reproducirlo físicamente (Lo que sirvió en su momento para descubrir nuevos patrones, y ahora sigue sirviendo para teorizar trucos que son meramente teóricos). En esta sección se aborda un repaso a la evolución de este tipo de aplicaciones.

2.3.1. Primeros simuladores

La mayoría de los primeros simuladores que existieron han dejado de funcionar en los sistemas operativos más modernos, aunque muchos siguen estando disponibles para su descarga [28]. Sus funcionalidades eran limitadas, variando entre la simulación de todos los *vanilla siteswap* con cierto número de bolas máximas hasta permitir variaciones limitadas como seleccionar tipo de lanzamientos (e.g., exteriores, por detrás de la espalda) o el propio objeto (e.g., mazas o aros).

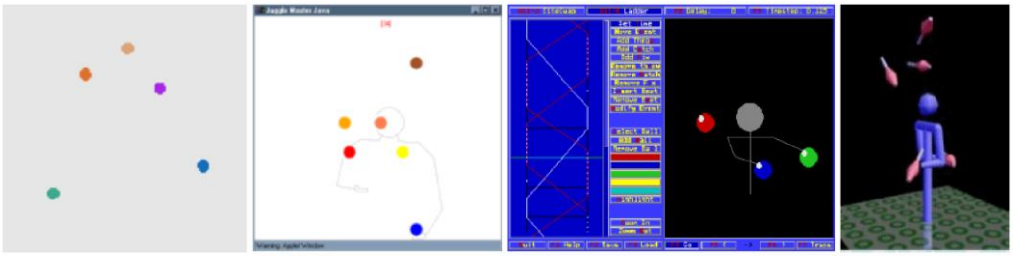

Figura 2.13: *Captura de los primeros simuladores de siteswap, disponible en [29].*

2.3.2. Simuladores actuales

El desarrollo de simuladores ha continuado hasta el día de hoy, incorporando todas las características necesarias para poder simular cualquier *siteswap* que sea correcto con cualquier modificación desde tipo de lanzamiento o tipo de objeto hasta cuánto tiempo se mantienen los objetos en las manos, cuando tiempo pasa entre cada lanzamiento… Permitiendo además en muchos casos incluso parar la simulación, verla hacia atrás, verla desde distintos ángulos…

La distribución de estos simuladores es variada, aunque la mayoría se pueden descargar como ejecutable. Muchos cuentan además con versión web o hasta descargas al código del proyecto.

Algunos de los ejemplos de estos simuladores más recientes son el *Siteswap animator* de *jugglingedge.com* [30], *Juggling Simulator* de la *World Juggling Federation* [31] o *Gunswap* por Eric Gunther [32].

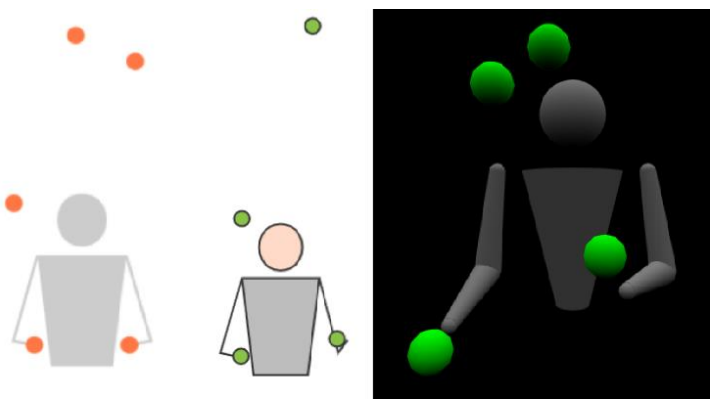

Figura 2.14: *Captura de los simuladores actuales, por orden: Siteswap animator, Juggling simulator y Gunswap.*

2.3.3. Juggling Lab

De cualquier forma, el simulador más extendido hoy en día es *Juggling Lab* [3], creado por Jack Boyce, pero en el que han colaborado decenas de personas desde que comenzó su desarrollo en 1997.

El principal atractivo de *Juggling Lab* respecto al resto de simuladores es que se encuentra disponible para todas las plataformas principales, lo que incluye Windows, macOS, Linux, Android y versión web. Además, mantiene todas las opciones de configuración mencionadas anteriormente y añade tanto enlace al repositorio y documentación como material especializado para comprender el funcionamiento interno del simulador.

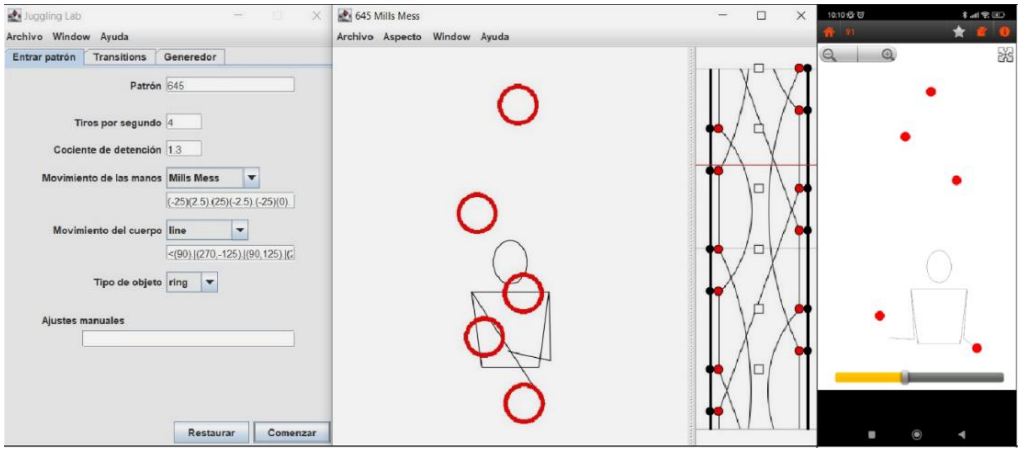

Figura 2.15: *Captura de Juggling Lab en su versión para Windows (Izquierda y centro) y Android (Derecha).*

3

Diseño

Esta sección presenta el diseño general del sistema implementado, justificando la presencia de cada módulo y explicando su funcionamiento de forma general tanto para los módulos propios como para aquellos sistemas tomados de partida para el desarrollo. Una descripción más detallada del funcionamiento de todos los subsistemas se encuentra en las secciones 4, 5 y 6.

3.1. *Sistemas base*

De la subsección 2.2 se han utilizado dos proyectos para tomar funcionalidad base a la hora de desarrollar este sistema. A continuación, se listan estos programas y se explican sus funcionalidades principales.

3.1.1. Stephen Meschke – *manual_tracking*

Dentro del proyecto de Stephen Meschke [33] se puede encontrar un programa que permite el *tracking* de una bola de forma manual, haciendo *click* en su posición en cada *frame* y obteniendo al final todas sus coordenadas. Si bien esta forma de *tracking* no es directamente útil en el marco del sistema a desarrollar, es necesaria para poder establecer *GroundTruths* contra los que comprobar cada sistema de *tracking* que se implemente.

3.1.2. Stephen Meschke – *colorspaces_tracking*

También dentro del proyecto de Stephen Meschke [33] se puede encontrar este programa que, a partir de un rango de color en valores del espacio de color HSV introducido manualmente, aplica una máscara a cada *frame* y devuelve la posición del contorno más grande encontrado. De esta forma, para videos con una bola, devolvería la posición de la bola en cada *frame*. No obstante, en caso de tener varias solo encuentra una en cada *frame*, y si se modifica el código del programa para devolver N contornos, dado que no tiene ninguna funcionalidad de *tracking* (solo detección), simplemente devuelve los N contornos más grandes.

3.1.3. Barton Chittenden – juggling_detector

Esta vez en el proyecto de Barton Chittenden [18] se encuentra un programa que, dada la entrada de un vídeo, pinta encima la detección de cada bola y manos, principalmente. Si bien es cierto que este programa tiene más de un par de decenas de parámetros opcionales, su funcionalidad base consiste en aplicar substracción de fondo

a cada *frame*, y dado que el movimiento principal entre cada *frame* lo provocan las manos y las bolas, devuelve sus contornos (y ruido que se pueda producir en el proceso). De cualquier forma, este programa tiene una funcionalidad muy básica de *tracking*, basada en distancias euclídeas desde el anterior contorno, lo que, añadido al ruido, genera decenas de identidades extra.

3.2. *Diseño del sistema general*

El sistema general se ha diseñado siguiendo el diagrama de la figura 3.1. Está formado principalmente por un módulo de detección y *tracking*, encargado de extraer las coordenadas (e identidad) de cada bola, otro de extracción del *siteswap*, encargado de obtener el *siteswap* desde las coordenadas, y un conjunto de otros módulos más pequeños que se encargan de tareas auxiliares. Los módulos/submódulos que no son obligatorios para el funcionamiento del sistema se marcan con líneas punteadas.

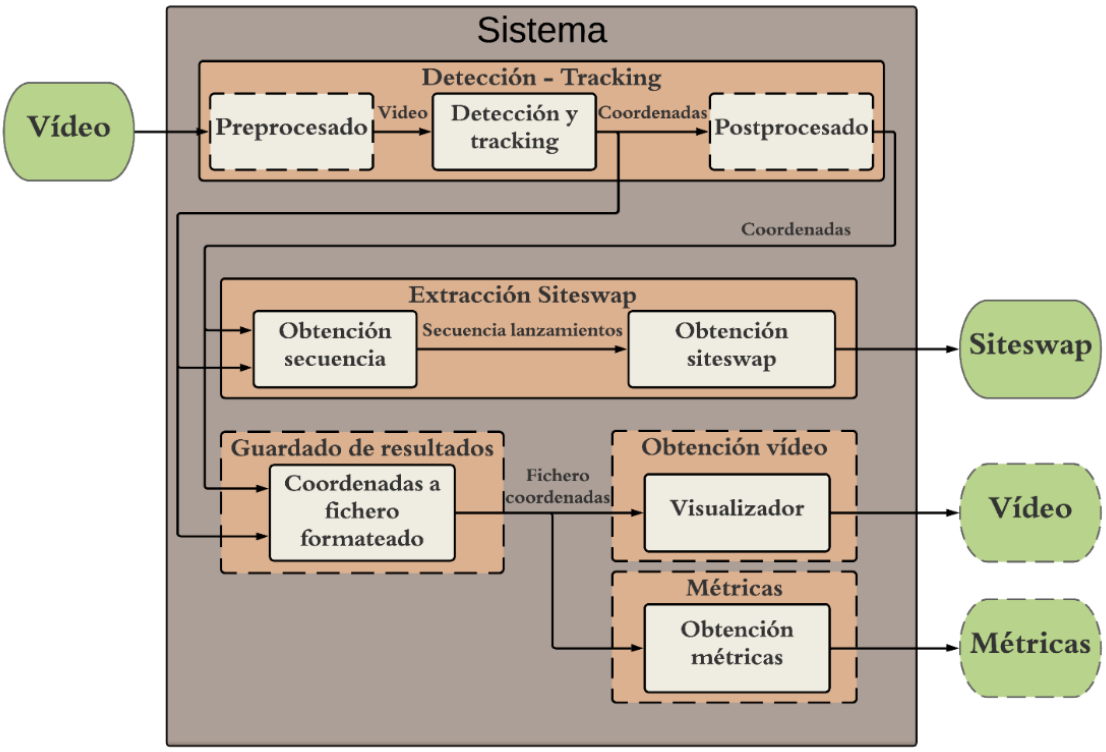

Figura 3.1: *Diagrama de diseño del sistema.*

3.2.1. Módulo "Detección-*Tracking*"

Para llevar a cabo su función, el módulo de detección y *tracking* se divide en los siguientes tres submódulos:

- Preprocesado: Este submódulo es opcional, y su función es asistir al propio submódulo encargado de la detección y *tracking* para que obtenga mejores resultados, ya sea modificando el propio vídeo de entrada, o bien facilitando alguna información que el usuario no tiene por qué conocer.

- Detección y *tracking:* Este submódulo es el que realmente se encarga de obtener qué pelota estaba en qué posición para cada *frame.* Para ello puede utilizar distintos métodos que serán discutidos en la siguiente sección.

- Postprocesado: Este submódulo también es opcional, y su función es tratar de mejorar las detecciones del módulo anterior, ya sea interpolando detecciones perdidas, eliminando ruido o suavizando las trayectorias detectadas.

3.2.2. Módulo "Extracción *siteswap*"

Dado que el *siteswap* realmente representa cuántos tiempos tarda cada bola en volver a ser recogida, una vez se tienen las coordenadas de las pelotas en cada momento, necesitamos información intermedia antes de poder calcular el *siteswap* que están siguiendo, es por ello que este módulo está compuesto por los siguientes dos submódulos:

- Obtención secuencia lanzamientos: A partir de las coordenadas, este submódulo detecta en qué momento ocurren los lanzamientos, y sobre qué objeto, de forma que a su salida saca el orden en el que se ha lanzado cada bola.

- Obtención *siteswap*: Desde la secuencia de lanzamientos que se ha seguido, este submódulo se encarga de encontrar una periodicidad y calcular a qué *siteswap* corresponde, lo que supone la salida final del sistema.

3.2.3. Módulos auxiliares

Además de los módulos directamente involucrados en la extracción del *siteswap* desde el vídeo, hay otros módulos que otorgan funcionalidades extra al sistema. Algunos no se muestran en el diagrama general, pues proveen asistencia directa a otro submódulo, mientras que otros incluso generan salidas del sistema distintas al *siteswap* obtenido, lo que sirve para tener otras referencias sobre el funcionamiento de este.

La lista con los módulos auxiliares y una descripción de estos se puede encontrar el apéndice A.

3.2.4. Marco evaluación

- **Creación y características del dataset**

Tanto para el desarrollo como para la evaluación ha sido necesario crear un *dataset* propio, ya que, aunque existen algunos de otros malabaristas, ninguno estaba centrado en la ejecución de *siteswaps*, si no que consisten en vídeos de trucos más sencillos para probar sistemas centrados en el *tracking*.

Los vídeos del *dataset* se ha intentado que sean en las condiciones más ideales posibles, por lo que están bien iluminados, con fondo negro y la ropa del malabarista toda negra también, además de tener todas las bolas del mismo color (rojo en este caso). La duración de los vídeos es siempre suficiente para que se repita el *siteswap* al completo varias veces, y oscila en torno los 10-15 segundos (algunos más, algunos menos) en función de la dificultad.

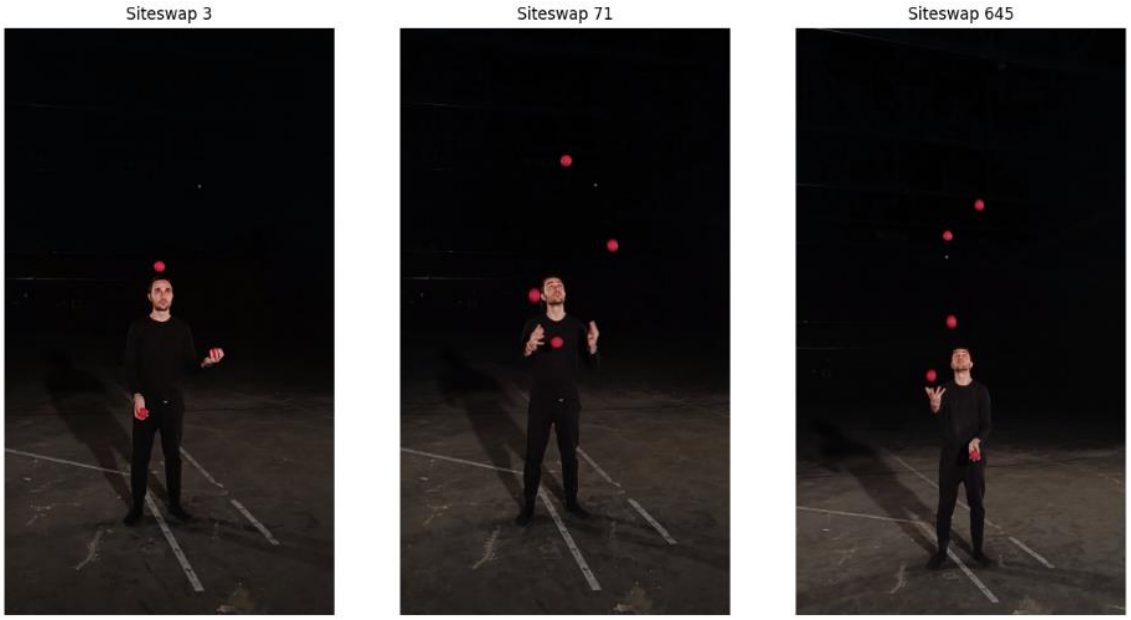

Figura 3.2: *Imágenes ejemplo dataset.*

El conjunto de vídeos se ha intentado que sea lo más representativo posible de los patrones que suelen aprender la mayoría de los malabaristas con cierto nivel técnico. De esta forma, se encuentran vídeos de 1 a 6 bolas, con cifras lo más repartidas posible entre 0 y 9 y periodos generalmente cortos.

Para el desarrollo, se ha trabajado con un subconjunto del *dataset* compuesto por los vídeos de los *siteswaps* 1, 3, 441, 423 y 5. El primero de ellos es, además del *siteswap* más sencillo, un caso particular que permite ver que la cifra “1” se esté detectando correctamente. Tanto el *siteswap* 3 como el 5 se han considerado las cascadas más representativas, además de servir como guía para vídeos de malabares sencillos y con mayor cantidad de objetos respectivamente. Sobre los dos que restan, son de los primeros *siteswaps* que se suelen enseñar, y además cubren otros casos concretos como los 2’s y los 4’s.

Para la evaluación se ha completado el *dataset* hasta tener un total de 22 vídeos, que añaden patrones más complejos, con alguna bola más y algunos casos más extremos (la lista completa se puede encontrar en la evaluación del sistema final en la sección 7.2).

Cabe destacar que el nivel de ejecución técnica del *siteswap* depende del nivel del malabarista en el vídeo, por lo que a *siteswaps* más complejos, peor nivel de ejecución, resultados un poco más “sucios” y vídeos con una longitud ligeramente inferior, lo que podría afectar a los resultados obtenidos por los sistemas de reconocimiento de *siteswap* que hiciesen uso de este *dataset*.

- **Métricas seleccionadas**

A la hora de hacer las evaluaciones se han diseñado y escogido métricas para cada uno de los módulos principales.

Concretamente, para el módulo de detección y *tracking*, las métricas elegidas son las de mot16 [34], que permiten evaluar múltiples aspectos de la detección y *tracking* de varios objetos. Concretamente, se ha usado la librería *motmetrics [35]*, y de todos los atributos que ofrece se han considerado como relevantes los siguientes:

- *Num_fragmentations:* Al valorar el número de fragmentaciones lo que se comprueba es la cantidad de veces que se pierde el seguimiento de un objeto.

- *Num_misses:* El número de fallos indica cuántas detecciones no se han llegado a producir.

- *Num_falses_positives:* Los falsos positivos en este caso hacen referencia al ruido que se tenga en las detecciones.

- *Num_switches:* Mide uno de los factores que repercuten más negativamente a la hora de extraer la secuencia de lanzamientos desde las detecciones; los casos en los que el identificador de dos objetos se intercambia.

- *Num_unique_objects:* En caso de estar perfecto corresponde al número de bolas.

- *Mostly_lost, partially_tracked, mostly_tracked:* Dado que hay *siteswaps* en los que no todas las bolas hacen el mismo recorrido, puede pasar que haya alguna en concreto para la que el sistema funcione mejor o peor. Estas métricas miden el porcentaje de tiempo que se sigue cada bola (<20%, 20-80%, >80% respectivamente).

- *Recall:* Mide cuántos de los objetos que deberían haber sido detectados fueron detectados correctamente.

- *Precision:* Mide cuántos de los objetos detectados son los que realmente se buscaban.

- *MOTP:* Muestra el desempeño del sistema en obtener posiciones precisas de objetos sin tener en cuenta su capacidad para mantener trayectorias constantes [36].

- *MOTA:* Tiene en cuenta todos los errores cometidos por el *tracker* tales como falsos positivos, fallos, errores de asignación de id… Evalúa si las trayectorias se mantienen de forma precisa sin tener en cuenta la precisión de las detecciones [37].

Sobre el módulo de extracción del *siteswap*, ha surgido la necesidad de diseñar una métrica desde 0. Un primer acercamiento fue usar distancias de edición (Levenshtein) [36], pero pierde bastante sentido al tener en cuenta que comparar 441 con 91 (por ejemplo) da una distancia de edición de 2 cuando son *siteswaps* completamente distintos. Otro acercamiento similar fue mirar la distancia de edición sobre el orden en el que se lanzaban las pelotas (según su identificador), pero tampoco funciona correctamente, ya que la secuencia [x,y,z,x,y,z,…] representa el mismo *siteswap* que [z,y,x,z,y,x,…] y su distancia de edición no solo no es 0, si no que crece a medida que aumenta la secuencia. Por ello, este acercamiento se descartó, y finalmente se optó por evaluar los siguientes aspectos:

- Presente***:*** Es un *booleano* que valora si el *siteswap* real forma al menos parte del *siteswap* predicho.

- Predicción: A pesar de que no supone una métrica por si sola, a la hora de mostrar los resultados se ha decidido mostrar el resultado del *siteswap* predicho para poder evaluar “a ojo” el funcionamiento del sistema.

- Presencia general: De la secuencia del *siteswap* obtenido sin reducir (sin “cortar por el periodo”) comprueba el porcentaje que pertenece al *siteswap* real.

- Periodo: Es un *booleano* que comprueba si el *siteswap* real tiene el mismo periodo que el *siteswap* predicho.

- Fallos: En caso de que el submódulo empleado pueda obtener este valor, es la tasa de lanzamientos que no se detectan del vídeo.

- Coincide: Es un *booleano* que valora en última instancia si el resultado obtenido es el correcto.

- **Parámetros utilizados en las pruebas**

Los valores de los parámetros configurables de cada submódulo utilizados para cada prueba se pueden consultar en el apéndice C.

4 Desarrollo y pruebas V0

En esta sección se entra en detalle de cada módulo mencionado en la anterior sección de diseño, describiendo tanto la funcionalidad e implementación como las pruebas realizadas.

4.1. Desarrollo

A continuación, se aborda tanto el funcionamiento como las decisiones de implementación de los componentes que suponen el primer sistema funcional. Dado que el único material de partida de esta iteración son los sistemas base, el objetivo es probar distintos acercamientos para, en la posterior evaluación, comprobar qué es lo que mejor funciona e irlo perfeccionando.

4.1.1. Módulo "Detección-*Tracking*"

- **Preprocesado**

El preprocesado que se realiza depende de las necesidades que tengan los módulos de detección y *tracking* que vengan detrás. En este caso, si bien es cierto que ninguno de los sistemas base necesita preprocesado de la imagen obligatoriamente, la detección aplicando máscaras de color tiene la contrapartida de que necesita que el usuario introduzca (y previamente obtenga) el rango de color que tienen las bolas. Con el objetivo de eliminar esta necesidad, se propone el diseño de un submódulo que calcule automáticamente este rango de color.

  - **Obtención color**

Este preprocesado trata de servir de apoyo al funcionamiento de uno de los sistemas base, el basado en máscaras de color. Para ello, el acercamiento seguido para cumplir el objetivo de este submódulo es emplear el funcionamiento base del programa de Barton Chittinden [18], basado en substracción de fondo, por lo que no necesita la información del color de antemano. La idea clave de este programa es que se obtienen los contornos de todo aquello que haya producido movimiento (en mayor o menor medida) entre distintos *frames*. La estructura general de este submódulo es conseguir la mayor cantidad de detecciones correspondientes a bolas, ver el color medio que suelen tener, y devolver un cierto rango que lo contenga. El problema de usar substracción de fondo para las detecciones es que gran parte de estas no corresponden a las bolas, si no a otros movimientos que pueda haber en el plano como son los brazos y las manos del

malabarista. Sin embargo, es importante tener en cuenta que no es necesario tener todas las detecciones de las bolas, simplemente tratar de asegurar que no haya falsos positivos. Por ello, cada detección es sometida a una serie de filtros relativamente estrictos que pretenden evitar todas las detecciones que no pertenezcan a bolas, aún a costa de perder muchas detecciones que sí que lo son. Para la implementación de estos filtros se han usado funciones auxiliares de *OpenCV* [38] para obtener la información base de cada contorno, de forma que luego se han podido crear las restricciones de forma manual.

Dichos filtros son:

- Un filtrado por área del contorno: Contornos demasiado pequeños los suponemos ruido (probablemente pequeños movimientos de la cabeza o cuerpo).

- Un filtrado por el número de lados del polígono que más se asemeja al contorno: Por cada contorno se obtiene el polígono aproximado que más se le asemeja, de forma que para considerar el contorno como válido se requiere que ese polígono tenga al menos N lados (y consecuentemente que tenga una forma aproximadamente circular).

- Que el polígono que más se asemeja sea convexo: Por último y con el mismo fin, se requiere que el polígono que más se asemeja sea convexo, por lo que contornos que no sean relativamente circulares pero que hayan sido aproximados por polígonos de más de N lados quedan descartados.

Una vez se tienen todas las bolas, el siguiente paso es extraer el rango de color. El color más repetido es el que se supone pertenece a las bolas (esto provoca la limitación de que todas las bolas tienen que ser del mismo color). Para devolver el rango, simplemente tomamos ese valor en el espacio de color HSV, y se varían ligeramente el valor de H (para tomar tonos similares de color), y V (para tomar distintas sombras de esos tonos) pero sin llegar a los límites, que implicaría devolver tonos blancos o negros, y se devuelven los límites inferiores y superiores que conforman el intervalo.

- **Detección y *tracking***

A la hora de realizar las detecciones y el *tracking*, sin entrar en el ámbito del aprendizaje automático, los tres sistemas de partida sirven de base para cada una de las tres posibilidades más viables encontradas; manual, por máscaras de color, y por substracción de fondo. Por ello, se han realizado tres implementaciones para este submódulo, cada una de las cuales parte de uno de los sistemas base. A continuación, se detalla el funcionamiento y las posibles adaptaciones de cada uno de los tres:

○ *Tracking* manual a partir del programa de Stephen Meschke

A pesar de que realizar *tracking* manual no es una vía coherente a la hora de realizar las detecciones para un sistema de estas características, es necesaria su implementación para establecer un *GroundTruth* y poder evaluar el resto de los sistemas.

El programa *manual_tracking* de Stephen Meschke [33] implementa una primera versión que permite obtener todas las coordenadas de una bola en forma de lista. Para ello, itera sobre cada *frame* del video, permitiendo al usuario hacer *click* sobre una bola, que será de la que se vayan guardando las coordenadas (correspondientes al *click*, una coordenada por *frame*). El programa además hace *zoom* sobre la bola que se está anotando, y va pintando por pantalla las últimas "detecciones" para que el usuario tenga una perspectiva general de la trayectoria.

Al planteamiento inicial de este programa se le han añadido las siguientes tres características con el fin de mejorar su usabilidad:

- Se muestra por pantalla el número de *frames* que le quedan al vídeo. Para ello, simplemente se lleva la cuenta del número de *frames* recorridos y se resta al número total que contenga el vídeo.

- En vez de hacer seguimiento de una sola bola, y luego tener que juntar manualmente las coordenadas, el programa implementado permite indicar el número de bolas que se pretenden anotar en el vídeo, de forma que en vez de obtener una lista de coordenadas se tiene un diccionario de listas donde cada clave pertenece a una bola.

- Dado que los procesos de anotación pueden ser largos, se ha incorporado al programa un *buffer* que permite deshacer las últimas anotaciones, de forma que, si se produce algún error, se entra en "modo *backtracking*" y se retrocede en el vídeo hasta llegar al punto deseado para retomar la anotación desde ahí.

Con ello, el programa final permite la anotación de vídeos de N bolas haciendo un *click* en cada *frame* por cada bola. El formato de la salida del programa depende de un parámetro, que en función de su valor desechará el resultado, lo sacará en un Excel mediante el módulo auxiliar "*excel_utils*" o lo sacará en formato mot16 mediante el módulo auxiliar "*mot16_utils*".

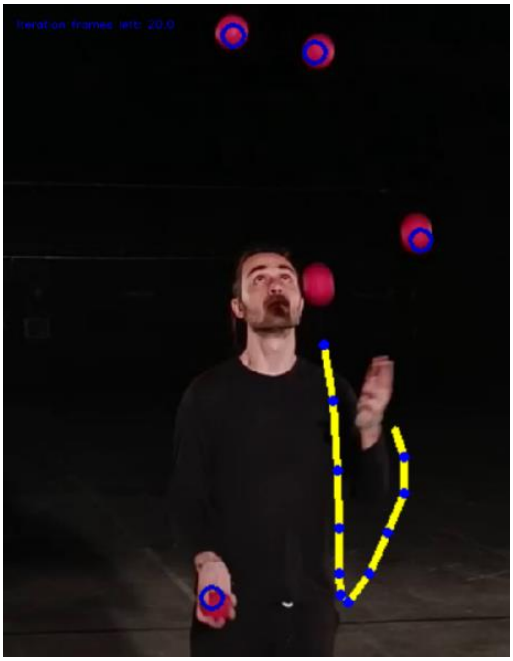

Figura 4.1: *Visualización de la ejecución del programa de tracking manual.*

- ***Tracking* por máscaras de color a partir del programa de Stephen Meschke**

Hacer las detecciones por máscaras de color supone el primer acercamiento real para el sistema. El programa de partida [33], al igual que pasaba con la versión manual, tiene el problema de que solo saca detecciones de una única bola. Esto es así porque, internamente, lo que hace el programa es, por cada *frame*, sacar todos los contornos correspondientes a aplicar la máscara de color para luego quedarse con el más grande, que es del que se guarda las coordenadas. Esto lleva implícito no solo que haya que modificar el programa para hacer N detecciones, sino que además hay que implementar alguna forma de asignar identificadores a las detecciones y asociarlos entre *frames*.

El primer cambio realizado es permitir que se hagan N detecciones. No obstante, por el método empleado, esto conlleva que se hacen muchas más detecciones de las que se buscan, y además se suelen superponer unas entre otras. Para solucionarlo, se ha implementado una función que realiza un proceso de supresión de no máximos, de forma que ordena los contornos y los compara unos con otros, eliminando en el proceso aquellos que estén muy cerca (en función de un umbral) de algún otro más grande.

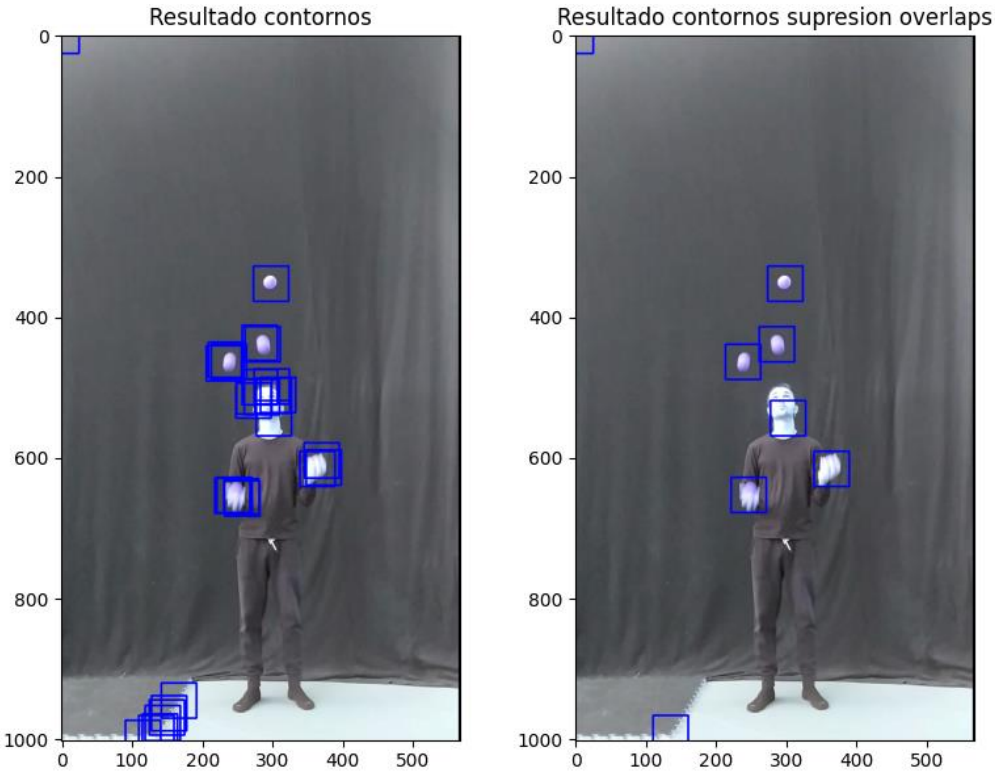

Figura 4.2: *Ejemplo supresión de no máximos en una etapa temprana del desarrollo.*

Una vez se ha reducido el número de detecciones a las que (presumiblemente) son realmente las bolas, se usa la funcionalidad implementada en el módulo auxiliar "*kalman_utils*", de forma que se hace una asociación entre las detecciones del *frame* anterior y las detecciones actuales (excepto si es el primer *frame* que se inicializa la estructura de identificadores).

Con ello finalmente se obtiene un diccionario con los identificadores como claves y una lista de detecciones como valores. En caso de haber detecciones perdidas, habrá más identificadores que bolas, y además las listas de coordenadas podrán tener menos *frames* que el video. Posibles soluciones se discuten en el postprocesado.

- ### *Tracking* por substracción de fondo a partir del programa de Barton Chittenden

Por su parte, Barton Chittenden implementa un programa basado en substracción de fondo [18] que es mucho más completo en posibilidades de configuración y en estructura, pues el planteamiento que tiene es permitir gran cantidad de opciones de configuración para mejorar las detecciones. Por lo general, estos parámetros se han desechado, pues su propósito suele ser mejorar los resultados para un vídeo fuente en concreto (definiendo manualmente el área en el que quieres realizar las detecciones por ejemplo).

El funcionamiento general de este programa se basa en el uso de un objeto BackgroundSubstractorMOG2 de OpenCV [39], que a nivel general funciona de tal forma que se le van aplicando *frames* y devuelve una máscara con aquellos píxeles en los que

haya habido cierto movimiento en función de otros parámetros. Con ello, itera sobre los *frames* y va sacando los contornos correspondientes al movimiento.

Uno de los problemas de este acercamiento es que se genera gran cantidad de ruido, pues, al haber un malabarista, su cuerpo produce gran cantidad de falsos positivos. Para tratar de mitigar este problema, Barton filtra por área las detecciones, para que contornos pequeños no se tengan en cuenta. Posteriormente, aplica un procedimiento sencillo para tratar de mantener los identificadores entre *frames*, basado en considerar dos detecciones entre *frames* la misma si están cerca.

Aunque sea un programa más completo, es cierto que su objetivo es ir pintando sobre el propio video las detecciones, y al ejecutarlo (figura 4.3) se puede comprobar que tiene dos problemas considerables. El primero es la cantidad de falsos positivos que genera, y el segundo es que el sistema de *tracking* genera muchos identificadores nuevos, de forma que para videos de 3 bolas el resultado final puede llegar a contar con más de 50 identificadores nuevos.

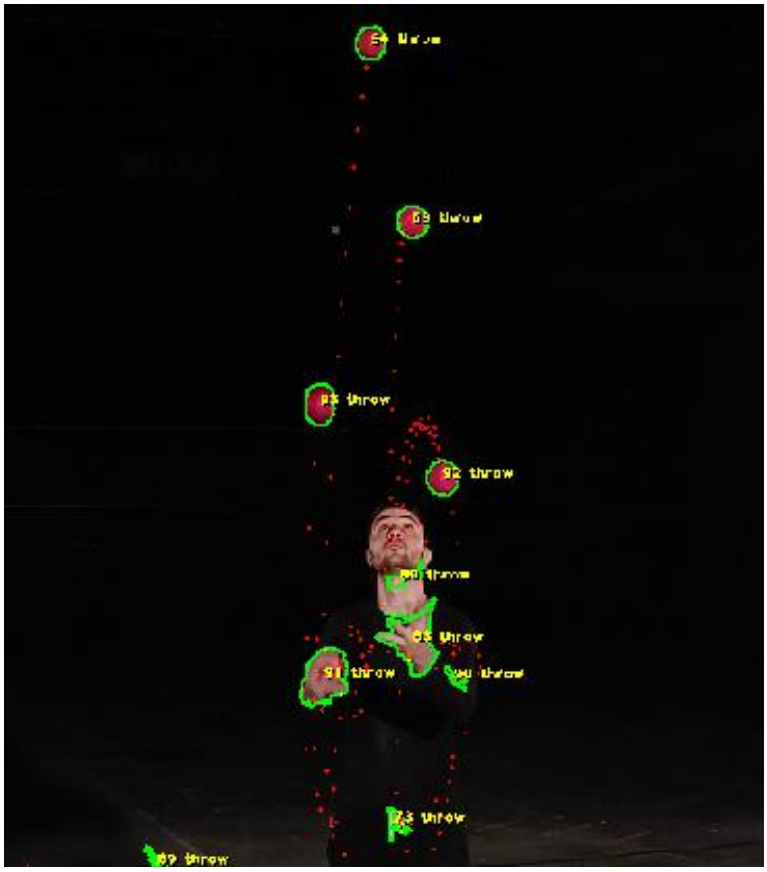

Figura 4.3: *Visualización ejecución programa Barton Chittenden.*

En la implementación propia, para tratar de solventar el primer problema, a los contornos se les pasa por un proceso más exhaustivo para tratar de determinar si son verdaderos positivos. Para ello, no solo se comprueba que tengan un área mínima, sino que además se comprueba que, o bien el mínimo círculo que se pueda hacer alrededor del contorno tenga un área similar, o bien que el polígono más aproximado al contorno tenga al menos cuatro lados y sea convexo. Esto pretende filtrar las formas circulares, por lo que detecciones de brazos y manos deberían quedar descartadas.

Respecto al segundo problema, simplemente se han añadido las llamadas respectivas al módulo auxiliar "*kalman_utils*", detallado en el apéndice A y que busca precisamente hacer una mejor aproximación al *tracking* de las bolas.

- **Postprocesado**

Dado que para la primera iteración aún no se han detectado los problemas del módulo de detección y seguimiento que se pueden mejorar con postprocesado, no se ha llegado a hacer ninguna implementación en esta parte. Sin embargo, quedan definidas las siguientes ideas para posiblemente realizar mejoras en iteraciones futuras o revisiones del trabajo:

- Eliminar ruido: Una vez se tienen los identificadores y sus coordenadas, todas aquellas detecciones que no se muevan del sitio corresponderán a ruido estático, y aquellas que se muevan, pero estén fuera de cierta zona de mayor densidad de detecciones, probablemente correspondan a ruido dinámico.

- Unir identificadores que realmente correspondan al mismo objeto: Teniendo en cuenta que el movimiento de cada bola es periódico, es posible que, viendo el movimiento que ha realizado la bola de cierto identificador y, si termina antes de tiempo, comprobando los movimientos de otra que comience después se pueda determinar que realmente son la misma y se puedan unir.

- Rellenar detecciones perdidas: Tanto si el programa ha perdido la detección en algún momento o si por postprocesado se han unido dos identificadores, habrá *frames* donde las coordenadas de algún objeto no tengan valor. Viendo los valores anteriores y posteriores se podría inferir cuánto valen para ese *frame.*

4.1.2. Módulo "Extracción *siteswap*"

Como se ha mencionado anteriormente, a la hora de extraer el *siteswap* se ha decidido hacerlo en dos pasos. El primero de ellos se encarga de, desde las detecciones, obtener la secuencia en la que se lanzan las bolas según su identificador. Una vez se tiene esa secuencia, el siguiente paso es calcular el número de tiempos que tarda cada objeto en volver a ser lanzado, y con ello recuperar la secuencia del *siteswap* completo, de la que se extraerá su periodo para poder retornar el *siteswap* mínimo resultado del procesado.

- **Obtención secuencia**

Dado que la información obtenida consiste en las posiciones de cada bola, graficando las mismas se observa el resultado mostrado en la figura 4.4.:

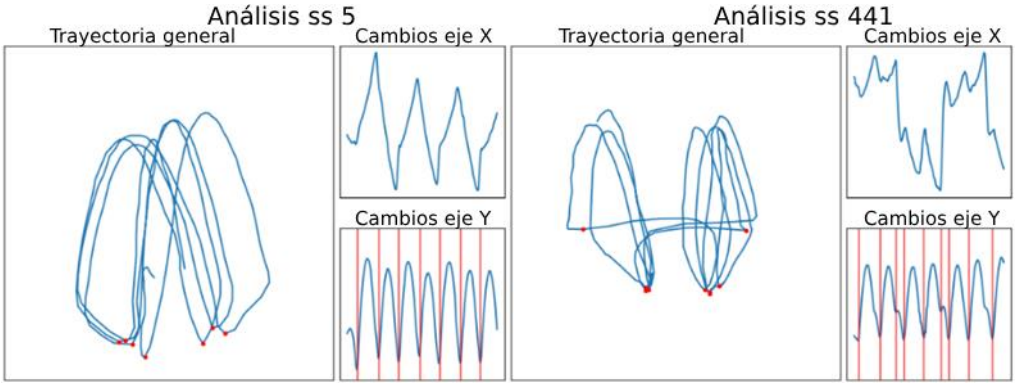

Figura 4.4: *Análisis posición de una bola para los siteswaps 5 y 441.*

Viendo la posición de cada bola, y sus cambios en cada eje, por el gesto que hace un malabarista para lanzar una bola, se puede observar que en la mayoría de los casos los lanzamientos corresponden a mínimos relativos en el eje Y (marcados en rojo). Aun así, hay tres lanzamientos en los que esto no tiene por qué cumplirse:

- 0's: Al no haber ninguna bola en esa mano, no hay información en la secuencia de detecciones de que se produce este "lanzamiento".

- 1's: Este lanzamiento, al producirse en horizontal, a nivel teórico podría no tener un mínimo relativo. Aun así, debido al cambio de trayectoria, el malabarista suele bajar la mano antes de lanzarlo, lo que debería producir la detección.

- 2's: Pasa algo parecido que con los 1's, en este lanzamiento el malabarista sujeta la bola con la mano, lo que podría ocasionar un mínimo relativo, o varios si el malabarista agita la bola, por ejemplo. De cualquier forma, dado que estamos suponiendo restricciones sobre la ejecución del *siteswap*, se puede suponer que los 2's se mantienen en la mano de forma estable.

Viendo cómo quedan marcados los mínimos relativos del eje Y en las trayectorias (puntos en rojo), se puede intuir que este acercamiento puede dar buenos resultados.

- **Obtención *siteswap***

La extracción del *siteswap* desde la secuencia de lanzamientos, si bien puede parecer otro punto de fallo para el sistema, es el único módulo que, en caso de recibir una entrada perfecta, produce siempre una salida perfecta. Esto es debido a la naturaleza matemática del *siteswap*, y funciona de la siguiente manera:

Dado que el *siteswap* cuenta el número de tiempos desde que una bola se lanza hasta que se vuelve a lanzar, si se tiene de partida la secuencia de lanzamientos perfecta, obtener la secuencia *siteswap* es relativamente sencillo, pues solo hay que ir contando este número de tiempos. De esta forma, para la secuencia de lanzamientos [x,x,y,z,z,y,x,y,z] se obtiene que la bola x se lanza cada [1,5] tiempos, la bola y cada [3,2] tiempos y la bola z cada [1,4], lo que al ordenarlo según el momento temporal en el que se producen los lanzamientos daría la secuencia [1,5,3,1,4,2,…] (indicar que el *siteswap*

"153142" sin añadir otros lanzamientos detrás es físicamente imposible de realizar por las propias normas de la notación).

De esta forma, si la secuencia de lanzamientos de entrada es perfecta, la secuencia *siteswap* obtenida es perfecta. El siguiente paso sería reducir al mínimo periodo, ya que, por ejemplo, la secuencia *siteswap* perfecta podría ser [4,4,1,4,4,1,4,4,1,…,4,4,1] por lo que el resultado que se busca es [4,4,1], es decir, el *siteswap* 441. Una vez se tiene un *siteswap* candidato, se comprueba que sea válido según las normas de la notación, y en caso afirmativo se devuelve.

4.1.3. Módulos auxiliares

Además, para esta iteración ha tenido lugar el desarrollo de los módulos auxiliares detallados en el apéndice A, encargados de aportar algún tipo de asistencia al sistema, ya sea sirviendo de apoyo a otro submódulo o bien generan sus propias salidas.

4.2. *Pruebas*

Esta sección recopila todas las pruebas y resultados obtenidos para los distintos submódulos implementados en esta iteración. En caso de los sistemas de detección los resultados serán los logrados directamente sobre el conjunto de desarrollo del *dataset* especificado en la sección 3 mientras que en el caso de las pruebas sobre el submódulo de extracción del *siteswap* se harán tanto sobre el *GroundTruth* de los videos como sobre los mejores resultados del módulo anterior, simulando así la ejecución del sistema completo.

4.2.1. Módulo "Detección-Tracking"

- **Color_tracking con preprocesado**

SS	Track Perdido	Detec. Perdidas	Detec. Ruido	ID Swap	Num Balls	<20%	20-80%	>80%	Recall	Precision	MOTA	MOTP
1	5	6	56	1	1	0	0	1	0.986	0.885	0.856	0.627
3	52	121	1	4	3	0	0	3	0.896	0.999	0.891	0.639
423	79	141	7	4	3	0	0	3	0.922	0.996	0.916	0.623
441	85	176	7	5	3	0	0	3	0.896	0.995	0.889	0.633
5	152	396	35	21	5	0	1	4	0.86	0.986	0.84	0.565

Tabla 4.1: *Resultados color_tracking con preprocesado V0.*

Los resultados obtenidos con esta configuración del módulo son considerablemente buenos para un primer acercamiento, en particular teniendo en cuenta que se están realizando las detecciones aplicando máscaras de color con un rango de color que ha sido obtenido de forma automática. Las puntuaciones de *recall*, precisión y MOTA son

relativamente altas, aunque MOTP tiene margen de mejora. Cabe destacar que, a mayor número de bolas, el sistema tiende producir mayor número de cambios de identificador, lo que resulta bastante problemático a la hora de extraer el *siteswap* posteriormente.

- **Color_tracking sin preprocesado**

SS	Track Perdido	Detec. Perdidas	Detec. Ruido	ID Swap	Num Balls	<20%	20-80%	>80%	Recall	Precision	MOTA	MOTP
1	*0*	*1*	*49*	*1*	*1*	*0*	*0*	*1*	*0.998*	*0.899*	*0.883*	*0.843*
3	*26*	*67*	*3*	*0*	*3*	*0*	*0*	*3*	*0.942*	*0.997*	*0.94*	*0.809*
423	*24*	*64*	*14*	*0*	*3*	*0*	*0*	*3*	*0.965*	*0.992*	*0.957*	*0.81*
441	*19*	*66*	*1*	*0*	*3*	*0*	*0*	*3*	*0.961*	*0.999*	*0.96*	*0.804*
5	*74*	*306*	*5*	*2*	*5*	*0*	*0*	*5*	*0.892*	*0.998*	*0.889*	*0.764*

Tabla 4.2: *Resultados color_tracking sin preprocesado V0.*

Al eliminar el preprocesado y establecer el rango de color manualmente (que se puede obtener por ejemplo desde el programa *hsv_color_picker* presente en el proyecto de Stephen Meschke [33]), nos aseguramos de que sea suficientemente restrictivo para no generar falsos positivos, pero que a su vez no haga falsos negativos (o los menores posibles en ambos casos). Como se puede observar, esto mejora los resultados anteriores, de forma que todas las bolas de todos los *siteswaps* son detectadas con éxito al menos más del 80% del tiempo (antes había una que estaba entre 20 y 80%), produciéndose pocos cambios de ID (3 en total respecto a los 35 anteriores) y llegando a puntuaciones que dejan cada vez menos margen de mejora en las métricas principales (mejoras del 5,34% en los resultados ya altos de MOTA, y del 30,54% para los resultados de MOTP).

Cabe mencionar que las puntuaciones de las métricas (particularmente de MOTP) se ven afectadas por decisiones de diseño y por lo tanto nunca van a ser perfectas. Un ejemplo de esto es el hecho de que se hagan supresiones de no máximos al haber varias detecciones suficientemente juntas, lo que provoca que se pierda puntuación debido a que al principio y al final de cada vídeo, cuando el malabarista tiene todas las bolas recogidas en las manos, se eliminan todos los identificadores en cada mano menos uno, por lo que ahí se tienen detecciones perdidas y consecuentemente baja la puntuación. No obstante, esta decisión mejora el resultado mientras se ejecuta el patrón, que al final es el momento que realmente importa a la hora de extraer el *siteswap*.

- ***Bg_substraction***

SS	Track Perdido	Detec. Perdidas	Detec. Ruido	ID Swap	Num Balls	<20%	20-80%	>80%	Recall	Precision	MOTA	MOTP
1	45	186	276	30	1	0	1	0	0.573	0.475	-0.128	0.614
3	125	441	50	19	3	0	3	0	0.619	0.935	0.56	0.691
423	129	549	36	45	3	0	3	0	0.696	0.972	0.651	0.762
441	142	362	57	58	3	0	3	0	0.786	0.959	0.719	0.698
5	212	755	23	43	5	0	5	0	0.733	0.989	0.709	0.742

Tabla 4.3: *Resultados bg_substraction V0.*

Por su parte, usar substracción de fondo para las detecciones devuelve unos resultados bastante distintos. Por la puntuación de MOTP y precisión se puede deducir que las detecciones (cuando las hay) son relativamente certeras, y quitando en el caso del *siteswap* 1 no hay demasiado ruido. Sin embargo, viendo las detecciones perdidas, se puede ver que tenemos gran cantidad de falsos negativos, lo que provoca que el sistema tenga que apoyarse más en los filtros de Kalman sin poder reajustarlos en nuevas iteraciones y consecuentemente que se produzcan muchos cambios de identificador, de ahí que haya bajado tanto los resultados de MOTA, llegando a obtener un valor negativo.

4.2.2. Módulo "Extracción *siteswap*"

Con los resultados obtenidos en la anterior subsección, el módulo de extracción de *siteswap* se va a comprobar tanto contra el *GroundTruth* como contra los resultados obtenidos con máscaras de color sin preprocesado, pues ha sido la configuración que mejores resultados ha aportado.

- ***Seq_extractor***

Respecto al *GroundTruth*:

Siteswap	*Siteswap* predicho	Presente	Presencia general	Periodo	Coincide
1	1	Verdadero	0.944	Verdadero	Verdadero
3	35223	Falso	0.44	Falso	Falso
423	Sin identificar	Falso	0.077	Falso	Falso
441	441	Verdadero	0.566	Verdadero	Verdadero
5	5	Verdadero	0.584	Verdadero	Verdadero

Tabla 4.4: *Resultados seq_extraction basado en coordenadas desde el GroundTruth V0.*

Respecto a los resultados obtenidos en el anterior módulo, los de la extracción del *siteswap* tienen mayor margen de mejora. Mientras que el *siteswap* 1 tiene un rendimiento muy bueno, en los otros dos que se identifican correctamente la secuencia general tiene gran parte de contenido que no corresponde correctamente al *siteswap*. Esto es relativamente normal en el caso del 441 (A mayor periodo te hacen falta un mayor número de detecciones correctas consecutivas para que la secuencia aparezca completa), pero para el 5 sería esperable tener

una presencia general más alta. Aún peor es el caso del *siteswap* 3, que no se llega a identificar correctamente, a pesar de no tener gran cantidad de bolas, un periodo alto o complejidad particular por ser un "caso extremo". Aun así, su presencia general no es mucho más baja que la de los *siteswaps* 441 o 5, por lo que podría ser que "hubiera quedado cerca" de ser identificado. Por su parte, el *siteswap* 423 queda lejos de tener un buen resultado. En este caso sin embargo tenemos un periodo relativamente alto, y además un caso particular que añade complejidad, los 2's (que como recordatorio significa que el malabarista retiene la bola en la mano y no la llega a lanzar), por lo que se puede sospechar que no se están detectando correctamente.

Respecto al *tracking* por máscaras de color:

Siteswap	***Siteswap* predicho**	**Presente**	**Presencia general**	**Periodo**	**Coincide**
1	1	Verdadero	0.733	Verdadero	Verdadero
3	3	Verdadero	0.625	Verdadero	Verdadero
423	Sin identificar	Falso	0.091	Falso	Falso
441	441	Verdadero	0.66	Verdadero	Verdadero
5	5	Verdadero	0.738	Verdadero	Verdadero

Tabla 4.5: *Resultados seq_extraction basado en coordenadas desde el mejor resultado de tracking V0.*

Curiosamente, el resultado sobre las detecciones del módulo anterior es mejor a rasgos generales que el obtenido sobre el *GroundTruth*. Esto probablemente se deba a que el *GroundTruth*, aunque tenga todas las detecciones correctas, es posible que la propia coordenada que se tiene no corresponda exactamente al centro de la pelota (por estar anotado a mano), mientras que, en el otro caso, si se ha identificado correctamente el contorno, el centro calculado es más preciso a pesar de que pueda haber alguna detección que no sea correcta. Además, hay que tener en cuenta que antes de modelizar la función sobre la que se sacan los mínimos relativos se pasan todas las coordenadas por un proceso de interpolación, para cubrir huecos con detecciones perdidas. Por ello, es posible que los mínimos relativos de la función obtenida sean más fáciles de identificar si las coordenadas que se tienen son el centro exacto de la bola.

Sobre los resultados, aunque el *siteswap* 1 tenga menor presencia en la secuencia general, el resto (menos el 423) mejoran considerablemente, llegando a identificarse correctamente el *siteswap* 3 (lo que puede confirmar la teoría de que en el caso anterior estuvo cerca de poderse identificar). Sobre el 423, sigue obteniendo una puntuación muy baja, lo que da mayor peso a la suposición de que los 2's no se identifican correctamente.

5 Desarrollo y pruebas V1

Un posible problema identificado en la anterior versión es que todos los parámetros han sido puestos “a ojo”, de forma que es probable que no sean óptimos. La versión 1 pretende mejorar esto, comprobando el rendimiento que obtiene el sistema para cada posible parámetro y así proponer cambios que mejoren el rendimiento general.

5.1. *Desarrollo*

Con la propuesta de optimizar la detección y el *tracking* se ha desarrollado un nuevo módulo auxiliar “*Optimizer*”, que incorpora varios *scripts* que pretenden encontrar experimentalmente los mejores valores de los parámetros de forma que, tras actualizarlos, se obtengan mejores resultados en las pruebas.

5.1.1. Módulo “Detección-*Tracking*”

- **Detección y *tracking***

Los cambios se han hecho sobre el submódulo de *tracking* por máscaras de color, ya que es el que mejor resultado estaba dando. Además, dado que parte de la optimización consiste en encontrar los mejores parámetros para los filtros de Kalman, estos valores pueden variar en función del submódulo que los esté usando.

Concretamente se han llevado a cabo dos ajustes. El primero de ellos ha sido optimizar todos los parámetros a los que se les había asignado algún valor manualmente, según los resultados obtenidos por el módulo “*Optimizer*”. Por otro lado, se ha añadido un nuevo parámetro que indica el número de bolas que tiene el vídeo, de forma que, dentro del código, se limitan las detecciones a ese número, lo que pretende reducir el ruido y reducir así el número de falsos positivos, reduciendo en el proceso la cantidad de cambios de identificador en el *tracking*.

5.1.2. Módulo “Extracción *siteswap*”

En este módulo no se ha realizado ningún cambio.

5.2. *Pruebas*

5.1.3. Módulos auxiliares

Se ha añadido un nuevo módulo auxiliar correspondiente a la optimización.

- ***Optimizer***

El resultado de las métricas no solo se ha utilizado para la evaluación, también se han usado para el ajuste de parámetros de los distintos submódulos. Para ello se han utilizado los ficheros de "*Optimizer*", que lanzan en paralelo múltiples ejecuciones probando distintos valores para los parámetros y sacando a un fichero los resultados obtenidos para cada valor del parámetro, lo que permite obtener los que mejor funcionan.

Un ejemplo de los resultados de este módulo está en la figura 5.1 en la que se pueden comprobar cuál es la distancia óptima a tener en cuenta entre las bolas para considerarlas (o no) la misma detección en función los cambios de id que se producen para distancias entre 0 y 200 en los *siteswaps* del conjunto de desarrollo.

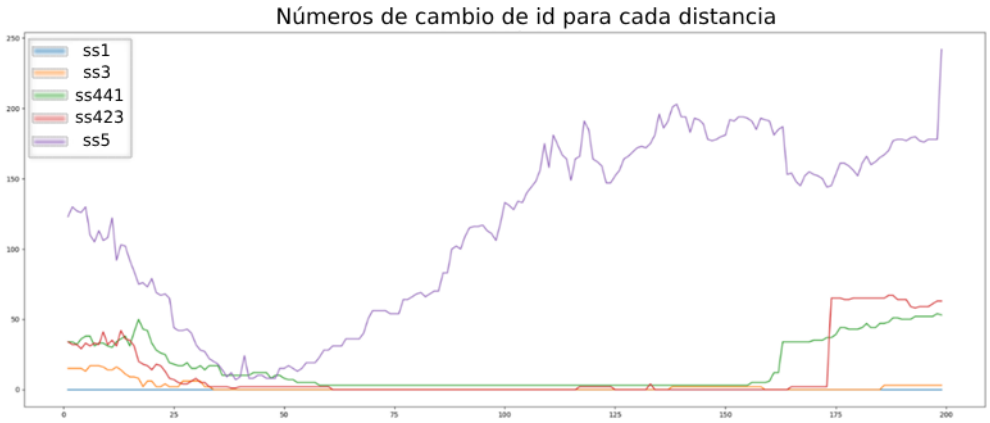

Figura 5.1: *Ejemplo optimización distancia para considerar mismo identificador.*

5.2.1. Módulo "Detección-*Tracking*"

Se han repetido las pruebas de la iteración 1 pero con los cambios en los parámetros. Dado que se ha tomado la decisión de centrarse en el *tracking* mediante máscaras de color, las pruebas se ejecutan únicamente sobre este submódulo.

- **Color_tracking con preprocesado**

SS	Track Perdido	Detec. Perdidas	Detec. Ruido	ID Swap	Num Balls	<20%	20-80%	>80%	Recall	Precision	MOTA	MOTP
1	2	3	2	0	1	0	0	1	0.993	0.995	0.989	0.628
3	29	80	3	5	3	0	0	3	0.931	0.997	0.924	0.637
423	54	111	10	2	3	0	0	3	0.938	0.994	0.932	0.621
441	63	145	11	6	3	0	0	3	0.914	0.993	0.904	0.633
5	66	249	26	22	5	0	0	5	0.912	0.99	0.895	0.559

Tabla 5.1: *Resultados color_tracking con preprocesado V1.*

Se puede observar cómo las optimizaciones han mejorado y homogeneizado los resultados de la anterior iteración, de forma que los resultados de *recall*, precisión¸ MOTA y MOTP o bien son mejores o bien son muy cercanos al valor anterior y están todos relativamente próximos. De cualquier forma, al obtener el rango de color de forma automática, el resultado de MOTP sigue siendo mejorable. También cabe mencionar que, aunque se ha reducido tanto las detecciones perdidas como el ruido, se produce un número de cambios de identificador similar (a veces más, a veces menos).

- **Color_tracking sin preprocesado**

SS	Track Perdido	Detec. Perdidas	Detec. Ruido	ID Swap	Num Balls	<20%	20-80%	>80%	Recall	Precision	MOTA	MOTP
1	0	1	0	0	1	0	0	1	0.998	1.0	0.998	0.843
3	2	28	20	0	3	0	0	3	0.976	1.0	0.976	0.809
423	0	32	0	0	3	0	0	3	0.982	1.0	0.982	0.81
441	0	37	0	0	3	0	0	3	0.978	1.0	0.978	0.804
5	4	197	0	2	5	0	0	5	0.93	1.0	0.93	0.76

Tabla 5.2: *Resultados color_tracking sin preprocesado V1.*

De igual forma, los resultados obtenidos seleccionando el rango de color manualmente aumentan aún más los resultados de la anterior iteración, a destacar la reducción de detecciones perdidas y ruido, que sumado a la calidad de las detecciones que ya se tenían provocan puntuaciones bastante altas en las métricas de las últimas cuatro columnas, con precisiones "perfectas", *recall* y MOTA con puntuaciones por encima del 0.9 en todos los casos y un MOTP que, teniendo en cuenta que no puede ser perfecto, mantiene puntuaciones relativamente elevadas.

Estos resultados se acercan mucho a la versión anotada a mano, de hecho, aunque no se pueda mostrar a través de las tablas de métricas, si se comprueban visualmente los resultados de las detecciones sobre el vídeo original, durante la ejecución del patrón es complicado observar comportamientos que apunten a que se tratan de detecciones automáticas y no del *GroundTruth*.

5.2.2. Módulo "Extracción *siteswap*"

En este caso, dado que no se han hecho cambios en la forma de obtener la secuencia de lanzamientos y/o *siteswap*s, a continuación se muestran los resultados obtenidos de realizar pruebas partiendo del *tracking* que ha aportado el mejor resultado.

- ***Seq_extractor***

Respecto al *tracking* por máscaras de color:

Siteswap	***Siteswap* predicho**	**Presente**	**Presencia general**	**Periodo**	**Coincide**
1	1	Verdadero	0.957	Verdadero	Verdadero
3	3	Falso	0.56	Verdadero	Verdadero
423	24	Falso	0.214	Falso	Falso
441	441	Verdadero	0.66	Verdadero	Verdadero
5	5	Verdadero	0.769	Verdadero	Verdadero

Tabla 5.3: *Resultados seq_extraction basado en coordenadas desde el mejor resultado de tracking V1.*

Dado que en esta iteración no se han realizado cambios sobre este módulo, los resultados son similares. No obstante, dado que los datos de entrada son más precisos, los resultados obtenidos también lo son, y mientras que las coincidencias exactas siguen siendo las mismas, la presencia general de todos los *siteswaps* mejora.

6 Desarrollo y pruebas V2

Viendo que el principal punto de fallo está situado en el módulo de "Extracción *siteswap*", se plantea dar un nuevo enfoque a la forma de extraer la secuencia de lanzamientos.

6.1. *Desarrollo*

En esta iteración, solo se van a realizar modificaciones en la forma de obtener la secuencia, por lo que el resto de los módulos se mantienen igual.

6.1.1. Módulo "Extracción *siteswap*"

A pesar de que obtener la secuencia de lanzamientos según los mínimos relativos del movimiento de cada bola en el eje horizontal funciona dentro de unos mínimos, en esta iteración se plantea un nuevo enfoque al procedimiento; En vez de fijarnos en los movimientos de un solo eje, mirar los dos ejes a la vez.

Con esta idea, se busca obtener un punto que separe en cuadrantes el video, de forma que la línea vertical delimita las zonas en las que se mueve cada mano y la línea horizontal las zonas en las que una bola, o está en una mano, o está en el aire. En caso de que una bola cruce la línea vertical en cualquier dirección y/o la línea horizontal de abajo a arriba, sabemos que ha habido un lanzamiento.

- **Obtención punto de separación en cuadrantes**

Para obtener dicho punto se ha implementado un programa que procesa el vídeo antes de que se intente obtener la secuencia, y devuelve únicamente esas coordenadas.

Conceptualmente, el punto separa la imagen en cuadrantes, de forma que los dos superiores tienen las zonas en las que menos movimiento se producen, el de abajo a la derecha contiene la zona en la que se suele mover el brazo izquierdo, y la de abajo a la izquierda igual, pero con los movimientos del brazo derecho. Por ello, se ha decidido hacer una implementación que lo primero que hace es recorrer todo el vídeo aplicando substracción de fondo, y todas las detecciones que hace (sin filtrarlas de ninguna forma) se van guardando en un mapa de calor. Un ejemplo del mapa de calor que se obtiene se muestra en la figura 6.1.

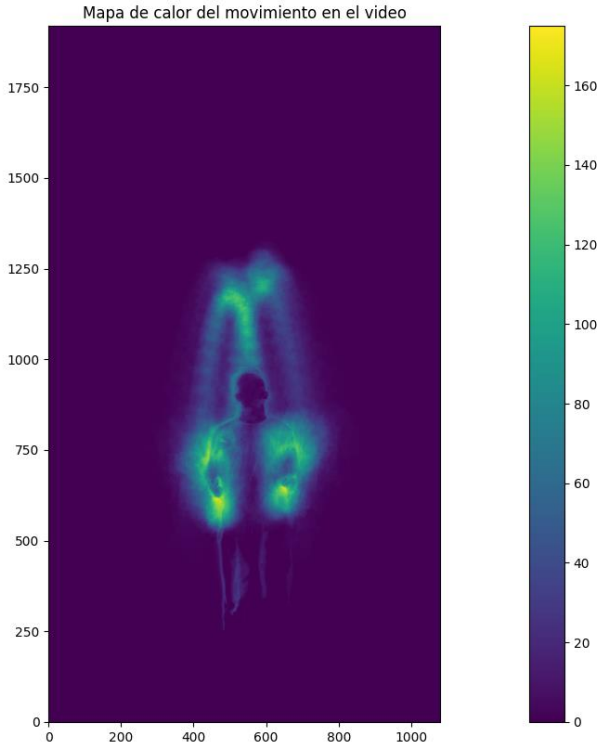

Figura 6.1: *Mapa de calor siteswap 5.*

Una vez se tiene el mapa de calor, se suman todos los valores correspondientes a cada columna, obteniendo un *array* que se puede entender como los distintos valores que toma una función que modeliza la cantidad de detecciones que ha habido en cada columna de píxeles. Este array se pasa por un proceso de suavizado, y se quitan los valores más bajos (Correspondientes a los laterales donde no hay detecciones). Con ello, se busca el mínimo relativo que tiene un valor más bajo, que en teoría siempre corresponde al punto central del patrón.

Respecto a la coordenada Y, el procedimiento es bastante parecido, solo que se suman los valores correspondientes a cada fila, y se recorre el vector de arriba abajo hasta llegar a un valor por encima de cierto umbral.

De esta forma, para el mapa de calor anterior se obtiene un resultado como el de la figura 6.2.

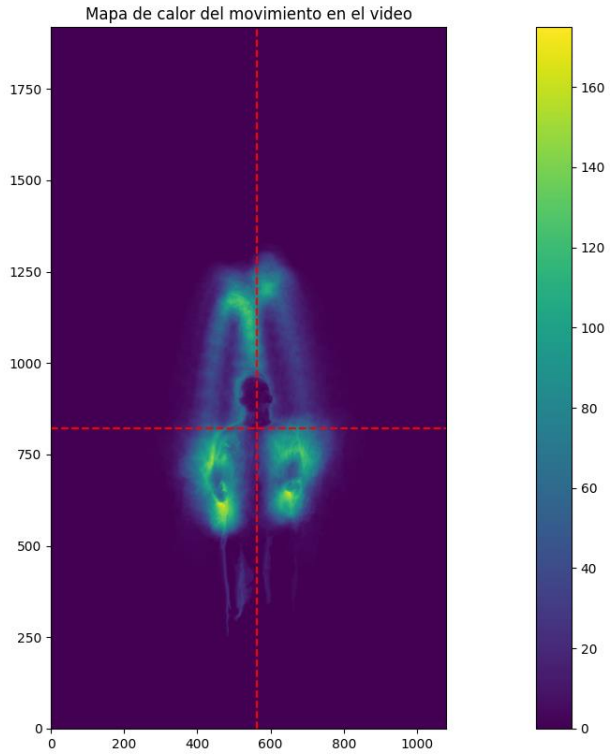

Figura 6.2: *Mapa de calor siteswap 5 marcado.*

- **Obtención secuencia**

Para obtener la secuencia, el objetivo de la implementación es controlar cada bola cuándo cruza cada eje, y obtenerlas ordenadas cronológicamente.

Para ello, el algoritmo se fija en cada bola por separado, comprobando en primer lugar si el lanzamiento en el que está se ha contabilizado ya o no. Para tener en cuenta cuándo una bola es lanzada de nuevo, se fija en la coordenada X, pues cuando hay un cambio de dirección en esa coordenada significa que la bola ha sido recogida para volver a lanzarse.

Con ello, en cada lanzamiento se comprueba si cruza el eje vertical (en cualquier dirección) o el horizontal (de abajo a arriba) establecido por el punto de separación en cuadrantes. La primera vez que se da esta condición por lanzamiento se marca el lanzamiento como comprobado y se guarda el identificador de la bola, el *frame* del lanzamiento y la mano que lo ha producido (información que se obtiene comprobando el cuadrante que acaba de abandonar).

En este punto, si se obtiene la lista de identificadores ordenada según el *frame* en el que se produce el lanzamiento se podría obtener la secuencia de lanzamientos buscada. No obstante, este planteamiento tiene un problema fundamental, que es la imposibilidad de detectar 0's o 2's, ya que incluso en situaciones ideales son los únicos lanzamientos que no cambian de cuadrante.

Para tratar de solventar esta situación se recorre toda la secuencia de lanzamientos comprobando qué mano lo ha realizado. En caso de que se repita dos veces seguidas la misma mano, se sabe que ha habido un error en algún punto, por lo que en la secuencia se introduce un id "-1" que se procesará posteriormente y se contabiliza el error (esto además permite obtener la tasa de errores).

Otro problema es que puede pasar que una bola cruce uno de los ejes sin que realmente haya sido lanzada (por ejemplo, si se va a lanzar un número par, digamos un 4, el gesto de la mano es hacer un círculo hacia el interior del patrón, por lo que la bola en la mano puede cruzar el eje vertical, pero ser lanzada hacia el exterior, de forma que no debería tenerse en cuenta como lanzamiento que cambia de cuadrante). Para solventarlo, en vez de contabilizar los lanzamientos en el momento en que se cruza uno de los ejes, se guardan como "posibles lanzamientos", que pasan a considerarse lanzamientos definitivos cuando la bola ha penetrado finalmente en un cuadrante por encima de cierto umbral.

- **Obtención *siteswap***

La parte de extracción del *siteswap* también ha tenido que ser modificada, pues ahora tiene que tratar con los identificadores "-1" que le pueden llegar del anterior submódulo.

Dado que cada aparición de uno de estos identificadores significa que se ha perdido una detección, podemos asumir que ha habido un 0 o un 2 en esa posición (en caso de haber sido otro número, la detección del lanzamiento habría fallado igual por alguna otra razón, por lo que no hay "penalización" por suponer otra cosa). De esta forma, de esta lista de identificadores se generan dos, una que entiende que donde hay -1's es porque hay 0's en la secuencia del *siteswap*, y otra que entiende que es porque hay 2's. Esto deja fuera posibles *siteswaps* donde en función de la posición puede haber cualquiera de los dos, pero los casos donde esto ocurre son un porcentaje tan bajo que se ha considerado como despreciable.

Con ello, se intenta obtener el *siteswap* con la lista con 2's (son más comunes), y si encuentra un resultado matemáticamente correcto se devuelve, si no, se intenta con la lista con 0's, que sigue la misma lógica, y en caso de tampoco encontrarse se devuelve "*NotFound".* Esta idea deja un posible punto de fallo, que sería el caso en el que el patrón contenga realmente 0's pero al suponer primero que son 2's el *siteswap* que se obtiene es técnicamente correcto, por lo que se devolvería. De cualquier forma, es un caso bastante concreto que con el diseño actual no es posible de controlar.

6.2. *Pruebas*

En este caso, se repetirán las pruebas del módulo "Extracción *siteswap*", pero aplicando siempre la forma de obtener secuencias de lanzamientos basada en los cuadrantes.

6.2.1. Módulo "Extracción *siteswap*"

- ***Seq_extractor basado en cuadrantes***

Respecto al *GroundTruth:*

Siteswap	*Siteswap* predicho	Presente	Presencia general	Periodo	Fallos	Coincide
1	1	Verdadero	0.952	Verdadero	0.0	Verdadero
3	3	Verdadero	0.87	Verdadero	0.0	Verdadero
423	234	Verdadero	0.733	Verdadero	0.312	Verdadero
441	414	Verdadero	0.879	Verdadero	0.0	Verdadero
5	5	Verdadero	0.726	Verdadero	0.066	Verdadero

Tabla 6.1: *Resultados seq_extraction basado en cuadrantes desde el GroundTruth*

Con este nuevo sistema de extracción de *siteswap* se consigue por primera vez obtener correctamente los 5 ejemplos de subconjunto de desarrollo. Por un lado, cabe destacar la baja tasa de fallos obtenidos (aunque realmente la tasa debería ser un poco más alta en todos los casos, porque si no la presencia general tendría que estar muy próxima a 1, sirve para dar un indicio de la calidad del funcionamiento), y por otro destacar también que la presencia general obtenida es bastante mayor a las que se conseguían con el otro acercamiento, siendo capaz este sistema además de detectar los 2's correctamente. En resumen, parece una forma más fiable de tratar de extraer la secuencia de lanzamientos.

Respecto al *tracking* por máscaras de color:

Siteswap	*Siteswap* predicho	Presente	Presencia general	Periodo	Fallos	Coincide
1	1	Verdadero	0.952	Verdadero	0.0	Verdadero
3	3	Verdadero	0.87	Verdadero	0.0	Verdadero
423	234	Verdadero	0.733	Verdadero	0.312	Verdadero
441	414	Verdadero	0.776	Verdadero	0.017	Verdadero
5	5	Verdadero	0.882	Verdadero	0.014	Verdadero

Tabla 6.2: *Resultados seq_extraction basado en cuadrantes desde el mejor resultado de tracking V2*

Los resultados son bastante similares respecto a la ejecución desde los datos del GroundTruth. Esto por un lado indica que este sistema no necesita de detecciones perfectas para funcionar, y por otro realmente demuestra lo bien que funciona el módulo anterior.

7

Versión final y pruebas

7.1. *Descripción*

Para la versión final del sistema se ha creado un programa cuya idea general es lanzar una ejecución en paralelo para cada vídeo que se quiera comprobar. A su vez, en cada ejecución, se pasa la entrada por un submódulo de detección y *tracking*, luego se extrae el *siteswap* con el submódulo de los cuadrantes y, en caso de que el resultado no parezca ser concluyente, se intenta con el submódulo de las coordenadas. Con ello, se puede devolver ese resultado o hacer una evaluación más exhaustiva en la que se calcule MOTA, MOTP y presencia general. Una vez están todas las ejecuciones finalizadas el programa devuelve una tabla por terminal con todos los resultados.

El programa cuenta con un archivo en el que se pueden establecer las siguientes configuraciones:

- Información básica de directorios y formatos de nombres que usa el programa.
- Lista de *siteswaps* (vídeos) a comprobar con el programa.
- Submódulo de detección y *tracking* a utilizar.
- Un *flag* que marca si se quiere hacer todo el *tracking* o directamente cargar los resultados desde ficheros ya existentes.
- Rango de color en caso de querer introducirlo manualmente para el *tracking* por máscaras de color.
- Un *flag* que marca que en vez de meter el rango de color manualmente se quiere obtener por preprocesado.
- La tasa de fallos que se admiten para tratar de obtener el *siteswap* por cuadrantes antes de pasar a coordenadas.
- Un factor que marca la longitud máxima que se permite al resultado para tratar de obtener el *siteswap* por cuadrantes antes de pasar a coordenadas.

- El número de índices que se usan para comprobar que una secuencia se repite antes de devolverla como *siteswap*.

- Un *flag* que marca si se quiere hacer la evaluación completa.

- El número de decimales que se muestran.

7.2. *Pruebas finales*

A continuación, se muestra el resultado de ejecutar el conjunto completo del *dataset* creado en el sistema final usando para el *tracking* el submódulo basado en máscaras de color con el rango establecido manualmente.

Núm. bolas	SS	MOTP	MOTA	Presencia	Predicción	Sistema usado	Fallos (Cuadrantes)	Funciona
1	1	0.843	0.998	0.952	1	Cuadrantes	0.0	Verdadero
2	31	0.802	0.997	0.923	31	Cuadrantes	0.0	Verdadero
	40	0.834	0.977	0.939	40	Cuadrantes	0.49	Verdadero
	330	0.826	0.997	0.0	123	Cuadrantes	0.312	Falso
3	3	0.809	0.976	0.87	3	Cuadrantes	0.0	Verdadero
	423	0.81	0.982	0.733	243	Cuadrantes	0.312	Verdadero
	441	0.804	0.978	0.776	414	Cuadrantes	0.017	Verdadero
	531	0.77	0.96	0.675	315	Cuadrantes	0.024	Verdadero
	51	0.786	0.946	0.808	54	Cuadrantes	0.074	Verdadero
4	4	0.805	0.949	0.615	4	Coordenadas	---	Verdadero
	633	0.775	0.906	0.783	633	Coordenadas	---	Verdadero
	5551	0.776	0.977	0.564	5155	Cuadrantes	0.087	Verdadero
	525	0.797	0.958	0.909	525	Cuadrantes	0.324	Verdadero
	534	0.803	0.93	0.871	534	Coordenadas	---	Verdadero
	66611	0.768	0.922	0.0	4	Cuadrantes	0.18	Falso
	561	0.794	0.943	0.056	---	Coordenadas	---	Falso
	75314	0.79	0.844	0.0	---	Coordenadas	---	Falso
5	5	0.76	0.93	0.882	5	Cuadrantes	0.014	Verdadero
	645	0.79	0.853	0.0	---	Coordenadas	---	Falso
	744	0.775	0.815	0.409	474	Coordenadas	---	Verdadero
	91	0.753	0.843	0.308	91	Cuadrantes	0.125	Verdadero
6	6	0.743	0.706	0.524	6	Coordenadas	---	Verdadero
Media		0.792	0.927	0.573	---	---	0.14	17/22

Tabla 7.1: *Resultados finales del sistema.*

El primer detalle a comentar de los resultados es que a nivel general parece funcionar peor a mayor cantidad de bolas, pero realmente es posible que se deba (como se comentó en la subsección 3.2.4) al hecho de que a mayor complejidad los videos fuente están técnicamente peor ejecutados y tienden a ser más cortos, por lo que hay menor secuencia en la que buscar el *siteswap*.

Sobre los casos que fallan hay que destacar por un lado que dos de ellos tienen periodo igual a 5, lo que complica la extracción del *siteswap* (a mayor periodo tiene que procesarse mayor proporción de la secuencia completa de forma perfecta para encontrarlo) y por otro que en el caso concreto del *siteswap* 330 cuando se busca la secuencia por cuadrantes teniendo en cuenta que los fallos pueden ser 0's el resultado se obtiene correctamente, pero al comprobar primero si los fallos pueden ser 2's encuentra un resultado válido y es el que devuelve.

De cualquier manera, los resultados del módulo de detección y *tracking* son bastante buenos, en particular teniendo en cuenta que en un porcentaje considerable del vídeo (el segundo que hay al principio y al final del vídeo en el que el malabarista no está realmente ejecutando el patrón) por la supresión de no máximos se pierden detecciones de partida. Sobre el módulo de extracción del *siteswap* funciona bien en un 77.27% de las veces, que teniendo en cuenta que es el primer acercamiento a un sistema de este estilo no es un mal resultado, en particular teniendo en cuenta que en la mayoría de los casos en los que ha fallado hay algún factor que los hace "casos extremos".

8 Conclusiones y trabajo futuro

8.1. *Conclusiones*

Como se ha mencionado en la sección anterior, el resultado final obtenido es que, para videos grabados en condiciones ideales, el sistema funciona bien en un 77.27% de las ocasiones. Teniendo esto en cuenta, y a pesar de que el *dataset* empleado es tan representativo que seguramente la mayoría de los malabaristas experimentados sean capaces de identificar los *siteswaps* a vista, resultados de este nivel podrían ser útiles para personas introduciéndose en la materia, o incluso para el resto de la comunidad en caso de que se mantuvieran para vídeos en peores condiciones.

Es por ello que revisiones futuras de este trabajo podrían añadir ciertas mejoras para funcionar en casos menos ideales, pudiendo llegar al lanzamiento de una aplicación muy utilizada dentro del nicho de los malabaristas (igual que los simuladores de *siteswap*), ya que cada vez hay un mayor número de malabaristas técnicos subiendo el listón de lo que se creía posible, y -de momento- hasta ejecutando *siteswaps* de 9 bolas de periodo mayor a 3 [40], que dejan de ser fácilmente identificables por la mayoría de malabaristas experimentados.

De cualquier forma, se han cumplido los objetivos inicialmente propuestos, y aunque no se haya entrado en soluciones basadas en aprendizaje automático debido a que escapaban del planteamiento inicial del trabajo por requerir un proceso de definición, obtención y anotado del *dataset* mucho más extenso, se han mejorado completamente los sistemas ya existentes de detección y *tracking* de bolas de malabares, que hasta ahora no hacían especial hincapié en mantener las identidades de las bolas, y además se ha añadido toda la funcionalidad relativa a la extracción de secuencias y obtención del *siteswap*, siendo el primer proyecto de estas características accesible en internet [41].

8.2. *Trabajo futuro*

Desde el propio planteamiento de este proyecto, surgen dos vías principales de mejora:

- Hacer un sistema de estas características que pueda detectar también aros, mazas de malabares y cualquier otro tipo de objeto.

- Hacer que, además de poder obtener *siteswaps vanilla*, pueda detectar patrones que requieran de notaciones más complejas como el *extended siteswap*.

De cualquier forma, es interesante notar que para ambas ampliaciones (una referente a la detección, y la otra a la extracción del *siteswap*), se puede sustituir algún módulo del sistema actualmente implementado y usar el resto.

De cualquier forma, durante el desarrollo e implementación de los distintos módulos, han surgido más vías de mejora que se podrían explorar en el futuro. Además de hacer módulos que, manteniendo el planteamiento de este trabajo mejoren resultados, podría ser interesante probar desarrollos basados en aprendizaje automático, lo que muy previsiblemente podría mejorar todos los módulos actuales, requiriendo eso sí un alto esfuerzo de ampliación del *dataset*. Además, aun manteniendo el punto de vista actual tanto en alcance del sistema como en implementación queda abierta la posibilidad de la implementación del módulo de postprocesado, y, teniendo en cuenta que el módulo de detección y *tracking* que mejores resultados ha dado necesita de antemano el número de bolas y su rango de color, otra vía de trabajo futura podría ser eliminar esta necesidad, o ser capaz de proveerla de forma automática y eficiente sin que esto afecte a los resultados obtenidos.

Bibliografía

[1] B. Beever. “Siteswap Ben’s Guide to Juggling Patterns”. (2000). [Online]. Disponible: https://www.jugglingedge.com/pdf/BenBeeversGuidetoJugglingPatterns.pdf. (Accedido: Nov 16, 2022).

[2] York Jugglers. “Juggling Simulators”. Yorkjugglers.org. [Online]. Disponible: http://www.yorkjugglers.org/links/juggling-simulators. (Accedido: Nov. 16, 2022).

[3] J. Boyce. “Juggling Lab”. (1.6.3). [Online]. Disponible: https://jugglinglab.org/. (Accedido: Nov. 16, 2022).

[4] A. Lubker. “History: Siteswap”. Juggle.fandom.com. [Online]. Disponible: https://juggle.fandom.com/wiki/Siteswap. (Accedido: Nov. 16, 2022).

[5] P. Klimek, comentario foro online. (Dic. 27, 2012). [Online]. Disponible: https://groups.google.com/g/rec.juggling/c/SHYhB9OYOuM/m/djC71AGeRzUJ. (Accedido: Nov 16, 2022).

[6] P. Klimek, comentario foro online. (Dic 30, 2012). [Online]. Disponible: https://groups.google.com/g/rec.juggling/c/SHYhB9OYOuM/m/c3w0TFo5LKoJ. (Accedido: Nov. 16, 2022).

[7] B. Tiemann, comentario foro online. (Dic. 28, 2022). [Online]. Disponible: https://groups.google.com/g/rec.juggling/c/SHYhB9OYOuM/m/_zj7icmejN0J. (Accedido: Nov 16, 2022).

[8] B. Tiemann, comentario foro online. (Dic 15, 2022). [Online]. Disponible: https://groups.google.com/g/rec.juggling/c/SHYhB9OYOuM/m/QKj7JT1WqFUJ. (Accedido: Nov. 16, 2022).

[9] P.Klimek, comentario foro online. (Dic. 28, 2022). [Online]. Disponible: https://groups.google.com/g/rec.juggling/c/SHYhB9OYOuM/m/GPLDI-S-VrkJ. (Accedido: Nov 16, 2022).

[10] B. Beever, “SITESWAP NOTATION” en Siteswap Ben’s Guide to Juggling Patterns. Reino Unido: 2000 pp. 15-15. [Online]. Disponible: https://www.jugglingedge.com/pdf/BenBeeversGuidetoJugglingPatterns.pdf. (Accedido: Nov 16, 2022).

[11] Hyacinth. “Siteswap relative visualized”. (2014). Wikimedia commons. [Online]. Disponible: https://commons.wikimedia.org/wiki/File:Siteswap_relative_visualized.png. (Accedido: Abril 03, 2023).

[12] B. Beever, “WHAT MAKES A PETTERN?” en Siteswap Ben’s Guide to Juggling Patterns. Reino Unido: 2000 pp. 5-9. [Online]. Disponible: https://www.jugglingedge.com/pdf/BenBeeversGuidetoJugglingPatterns.pdf. (Accedido: Nov 16, 2022).

[13] B. Beever, “DESIGNING SITESWAPS” en Siteswap Ben’s Guide to Juggling Patterns. Reino Unido: 2000 pp. 28-29. [Online]. Disponible: https://www.jugglingedge.com/pdf/BenBeeversGuidetoJugglingPatterns.pdf. (Accedido: Nov 16, 2022).

[14] D. Amaya. “Dots & Lines. Juggling Balls Tracking”. (Abr. 11, 2017). Youtube.com. [Online]. Disponible: https://www.youtube.com/watch?v=bwmcXeKU4YI&ab_channel=DanieloAmaya. (Accedido: Nov. 16, 2022)
[15] G. Gorodecki. “JugglingTracker_Python”. Github.com. [Online]. Disponible: https://github.com/GayaGorodecki/JugglingTracker_Python. (Accedido: Nov. 16, 2022)
[16] N. Guy. “Ball Trackier and Predictor for Juggling #2”. (Ago. 10, 2015). Youtube.com. [Online]. Disponible: https://www.youtube.com/watch?v=O4fVQKigutk&list=PL3n7ZCdaE2EPLa47BZ6NNGX0-zex62W6p&index=6&ab_channel=NattyBumppo. (Accedido: Nov 16, 2022)
[17] N. Guy. “Basic State Estimator to Tracki Juggling Balls in Video Data”. (Ene. 22, 2016). [Online]. Disponible: https://www.natguy.net/juggling_paper.pdf. (Accedido: Nov. 16, 2022).
[18] B. Chittenden. “juggling_detector”. Github.com. [Online]. Disponible: https://github.com/bartonski/juggling_detector (Accedido: Nov. 16, 2022).
[19] T. Dresser (comunicación privada), 2022.
[20] R. Åkerlund. “Balls and Hands in Videos of Juggling.” Distribuido por Kaggle. [Online]. Disponible: https://www.kaggle.com/datasets/rasmuspeterakerlund/balls-and-hands-in-videos-of-juggling (Accedido: Nov. 16, 2022).
[21] R. Åkerlund. “Pattern Categorization Demo). (May. 1, 2019). Youtube.com. [Online]. Disponible: https://www.youtube.com/watch?v=nsTGB06gu40&ab_channel=Rasmus%C3%85kerlund. (Accedido: Nov. 16, 2022).
[22] R. Åkerlund. “Real-time localization of balls and hands in videos of juggling using a convolutional neural network”. (Mar. 22, 2019). [Online]. Disponible: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1297966&dswid=1529. (Accedido: Nov. 16, 2022).
[23] S. Meschke. “Neural Network Pattern Detection”. Openjuggle.com. [Online]. Disponible: https://openjuggle.com/neural-net-pattern-detection. (Accedido: Nov. 16, 2022).
[24] S. Meschke. “Model”. (Oct. 26, 2017). Youtube.com. [Online]. Disponible: https://www.youtube.com/watch?v=s1DHupsWpNE&ab_channel=StephenMeschke. (Accedido: Nov. 16, 2022)
[25] J. Cozens. “Realtime Juggling Siteswap Detection Demo”. (Oct. 8, 2022). Youtube.com. [Online]. Disponible: https://www.youtube.com/watch?v=0dtM_QP0luY&ab_channel=JamesCozensJuggling. (Accedido: Nov. 16, 2022).
[26] J. Cozens (comunicación privada), 2022.
[27] J. Cozens. “Tom Whitfield 11 Ball Juggling WR | Tracking and Analysis”. (Oct. 14, 2022). Youtube.com. [Online]. Disponible: https://www.youtube.com/watch?v=Ba4ul1q8VwI&ab_channel=JamesCozensJuggling. (Accedido: Nov. 16, 2022).
[28] B. Apps. “IBM-PC Programs”. Juggling.org. [Online]. Disponible: http://www.juggling.org/programs/ibm-pc/. (Accedido: Ene. 31, 2023).
[29] E. Colin. “Juggling Simulators – An Introduction”. Jugglingdb.com. [Online]. Disponible: http://web.archive.org/web/20120412141438/http://www.jugglingdb.com/compendium/geek/software/simulators.html?lang=en. (Accedido: Ene.31, 2023).
[30] JugglingEdge. “Siteswap animator”. [Online]. Disponible:

https://jugglingedge.com/help/siteswapanimator.php?Pattern=(6x%2C4)*. (Accedido: Ene. 31, 2023).
[31] WJF. “Juggling simulator”. [Online]. Disponible: https://www.thewjf.com/siteswap/. (Accedido: Ene. 31, 2023).
[32] E. Gunther. “Gunswap”. Github.com. [Online]. Disponible: https://github.com/ydgunz/gunswap. (Accedido: Ene. 31, 2023).
[33] S. Meschke. “juggling”. Github.com. [Online]. Disponible: https://github.com/smeschke/juggling. (Accedido: Abr. 15, 2023).
[34] A. Milan, L. Leal-Taixé, I. Reid, S. Roth y K. Schindler. “MOT16: A benchmark for multi-object tracking”. (May. 3, 2016). arXiv preprint arXiv:1603.00831.
[35] C. Heindl. “py-motmetrics”. Github.com. [Online] Disponible: https://github.com/cheind/py-motmetrics. (Accedido: Abr. 15, 2023).
[36] K. Bernardin, A. Elbs y R. Stiefelhagen. “Multiple Object Tracking Performance Metrics and Evaluation in a Smart Room Environment”. [Online]. Disponible: https://cvhci.anthropomatik.kit.edu/~stiefel/papers/ECCV2006WorkshopCameraReady.pdf. (Accedido: Abr. 15, 2023).
[37] A. Fawzy. “Measuring Text Similarity Using the Levenshtein Distance”. [Online]. Disponible: https://cvhci.anthropomatik.kit.edu/~stiefel/papers/ECCV2006WorkshopCameraReady.pdf. (Accedido: Abr. 15, 2023).
[38] OpenCV. “The OpenCV Reference Manual”. [Online]. Disponible: {http://docs.opencv.org/. (Accedido: Jun. 4, 2023).
[39] OpenCV. “cv::BackgroundSubstractorMOG2 Class Reference.”. OpenCV Documentation. [Online]. Disponible: https://docs.opencv.org/3.4/d7/d7b/classcv_1_1BackgroundSubtractorMOG2.html.(Accedido: Jun. 1, 2023).
[40] D. La Terra. “aaaaaaaa1”. (Oct. 11, 2022). Instagram.com. [Online]. Disponible: https://www.instagram.com/p/CjkjUM9gCD_/. (Accedido: Abr. 15, 2023).
[41] A. Alonso. “tfg_jugglingTrackingSiteswap”. Github.com. [Online]. Disponible: https://github.com/AlejandroAlonsoG/tfg_jugglingTrackingSiteswap. (Accedido May. 29, 2023).
[42] R. Sadli. “2-D-Kalman-Filter”. Github.com. [Online]. Disponible: https://github.com/RahmadSadli/2-D-Kalman-Filter. (Accedido: May. 27, 2023).
[43] R. E. Kalman. “A New Approach to Linear Filtering and Prediction Problems”. [Online]. Disponible: https://www.cs.unc.edu/~welch/kalman/media/pdf/Kalman1960.pdf. (Accedido May. 27, 2023).
[44] A. Becker. “Kalman Filter in One Dimension”. Kalmanfilter.net. [Online]. Disponible: https://www.kalmanfilter.net/kalman1d.html. (Accedido: May. 27, 2023).

Apéndices

A

Módulos auxiliares

A.1. *Kalman_utils*

Este módulo implementa toda la lógica de tracking del programa. Se basa el funcionamiento de los filtros de Kalman descrito en el apéndice B para tratar de predecir las posiciones futuras de las detecciones y así asignarles el id correspondiente. Concretamente tiene funciones para:

- Inicializar la estructura de datos que se va a usar, que será un diccionario con los identificadores como claves, y para cada identificado se medirá, entre otros, la última coordenada, el historial de posiciones, su propio filtro de Kalman, la posición futura predicha y el historial de predicciones.

- Desde los identificadores y una lista de detecciones, crear las asignaciones correspondientes. Para ello, crea una matriz de distancias e irá asignando los identificadores previos a las detecciones que estén más cerca y dentro de cierto umbral. En caso de que haya detecciones que sobren se crearán nuevos identificadores.

- Actualizar detecciones perdidas de forma que, si hay un identificador al que no se le ha asignado una nueva detección, se considera su nueva coordenada la predicción previa, para ver si en la siguiente iteración se puede recuperar. En caso de que lleve 10 *frames* sin tener asignada ninguna detección real, se da por perdido, pues las predicciones dejan de ser realistas.

Aunque la implementación de la funcionalidad previamente descrita forma parte de este proyecto, se ha usado el módulo auxiliar *2-D-Kalman-Filter* [42] para toda la funcionalidad relativa a los propios filtros de Kalman

A.2. *Excel_utils*

Este módulo gestiona una de las formas de guardar la información relativa a la detección y *tracking*. Concretamente, contiene todas las funciones necesarias para crear, guardar y cargar libros de Excel donde la primera página contendrá información general como el sistema utilizado y video analizado, y en la segunda página contendrá todas las

detecciones, de forma que cada fila corresponderá a un *frame* y cada columna al eje X o Y de un identificador.

Sirve principalmente para poder visualizar de forma sencilla el funcionamiento del sistema, y en caso de querer ver alguna presentación de estos.

A.3. *Mot16_utils*

Este es el otro módulo encargado de gestionar la información de detección y *tracking*. Contiene toda la funcionalidad para manejar archivos de texto en formato mot16 [34].

El uso de este formato tendrá dos funcionalidades. La primera de ellas es que es mucho más ligero que el Excel, por lo que para el guardado y cargado de datos dentro del sistema será la opción escogida. La segunda de las funcionalidades es que, al ser un formato más estandarizado, es el formato elegido a la hora de realizar métricas sobre las distintas opciones de detección y seguimiento para poder evaluarlas, ver cómo optimizarlas y compararlas.

A.4. *Visualizer*

El *visualizer*, como su nombre indica, será el módulo responsable de la visualización del *tracking*. Recibiendo el video de partida y los resultados del del módulo de detección y *tracking* muestra por pantalla (y opcionalmente guarda a disco) el vídeo con las detecciones por encima, y al lado lo mismo, pero con fondo negro, de forma que se puede comprobar visualmente el funcionamiento del sistema.

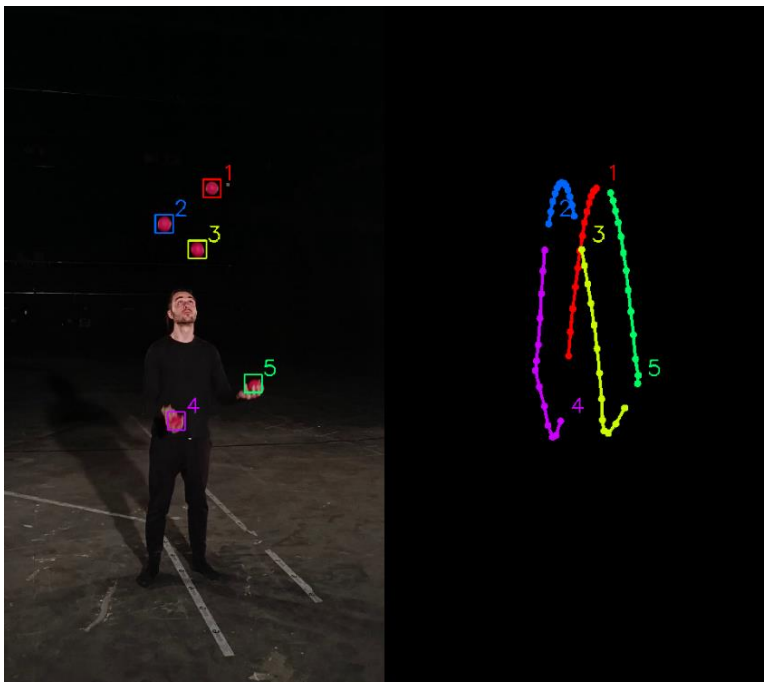

Figura A.1: *Captura ejecución tracking_visualizer*

A.5. *Metrics*

Parte de los programas implementados corresponden a métricas para evaluar cada uno de los subsistemas. Las métricas utilizadas están detalladas en la sección 3.2.4, pero a nivel general se han implementado ficheros con métricas para los módulos de "Detección-*Tracking*" y "Extracción *siteswap*".

B

Filtros de Kalman

Un Filtro de Kalman [42], [43] es un algoritmo que permite estimar el siguiente estado de un sistema dinámico, y en este proyecto ha sido usado para, a partir de las posiciones anteriores de una bola, tratar de predecir cuál va a ser su siguiente posición y así poder asignarle su identificador. Conceptualmente siguen el siguiente diagrama:

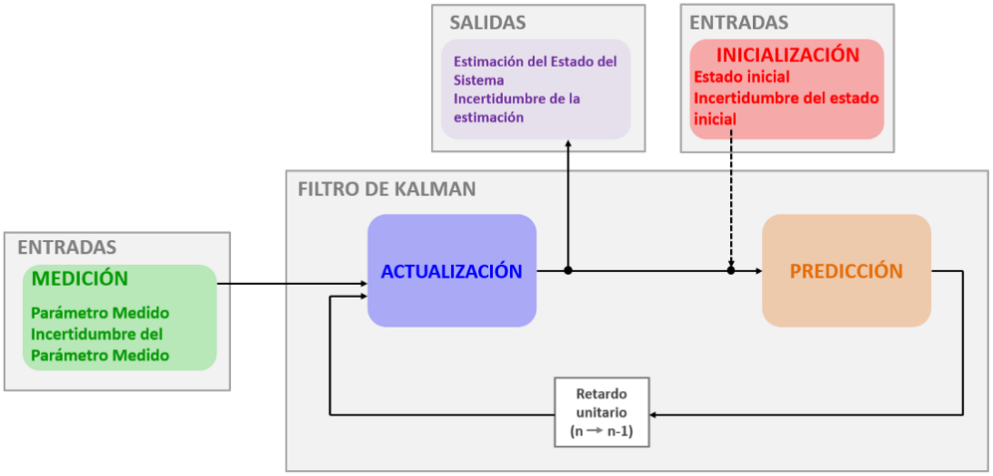

Figura B.1: *Diagrama base filtro de Kalman. Extraído de: [44].*

Primero se parte de una entrada de la que se tiene un estado inicial $\hat{x}_{1,0}$ y una incertidumbre $p_{1,0}$ (Que pueden estar proporcionadas por otro sistema o inicializadas a mano). A dicha entrada se le somete a un proceso de actualización y predicción en el cual pasa por ciertas ecuaciones que definen el funcionamiento de los Filtros de Kalman y el resultado que se obtiene se considera la predicción del estado siguiente.

Cuando se tiene la primera medición, se usa en conjunto a la predicción que se había obtenido para ajustar los parámetros internos de las ecuaciones, de forma que el siguiente estado predicho sea más preciso, y así sucesivamente.

Concretamente, las funciones que definen el funcionamiento de los Filtros de Kalman son las siguientes:

- Ganancia de Kalman:

Considerando que el error de cada medición es aleatorio, se pueden describir por su varianza, de forma que la varianza de los errores es la incertidumbre de medición (r_n). A medida que pasan las iteraciones, el error entre predicción y medición se debería hacer más pequeño, y aunque no se puede saber cuál es, se puede aproximar su incertidumbre de estimación ($p_{n,n}$).

La Ganancia de Kalman (k_n) es el parámetro que modela cuánto nos fiamos de una medición respecto a la predicción que hubiéramos generado para ajustar el estado actual. Es decir, con si se tuviera un sensor muy impreciso (alta incertidumbre de medición) y la predicción no coincide con la medición se tendría más en cuenta el estado predicho (y viceversa en el caso contrario). La forma de esta ecuación está definida en la ecuación B.1:

$$Kn = \frac{Incertidumbre\ estimación}{Incertidumbre\ estimación + Incertidumbre\ medición} = \frac{p_{n,n-1}}{p_{n,n-1}+r_n} \qquad Ec.\ B.1$$

- Actualización de covarianza:

Al ir teniendo más información sobre el sistema a medida que pasan las iteraciones, la incertidumbre de medición cambia (y tiende a 0). Este comportamiento es el que modeliza la actualización de la covarianza, que se rige según la ecuación B.2:

$$p_{n,n} = (1 - K_n)p_{n,n-1} \qquad Ec.\ B.2$$

Es importante resaltar que según la ganancia de Kalman sea más alta (hay menor incertidumbre sobre la medición) la convergencia es más rápida.

- Extrapolación de covarianza:

La extrapolación de la covarianza añade a la incertidumbre del siguiente estado el ruido de las mediciones (q_n), que puede provocar imprecisiones. Para sistemas estáticos tiene la forma de la ecuación B.3.

$$p_{n+1,n} = p_{n,n} + q_n \qquad Ec.\ B.3$$

- Extrapolación de estado:

Las ecuaciones de extrapolación de estado son las que definen propiamente los estados en los que puede estar el sistema. Por ello, estas ecuaciones cambian tanto en forma como en número en función del sistema sobre el que se realicen las predicciones. En un sistema como el de este proyecto las ecuaciones que definen

el estado (la posición de la bola) son las que definan su posición, velocidad y aceleración para un momento dado.

- Actualización de estado:

La última ecuación es la que modela, en función de la ganancia de Kalman, el peso que se le da al estado predicho y a la medición respectivamente para asumir el estado actual. Se rige según la ecuación B.4:

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + k_n\left(z_n - \hat{x}_{n,n-1}\right) \qquad Ec.\ B.4$$

Teniendo en cuenta estas ecuaciones y actualizando el diagrama anterior, un Filtro de Kalman sigue la siguiente estructura:

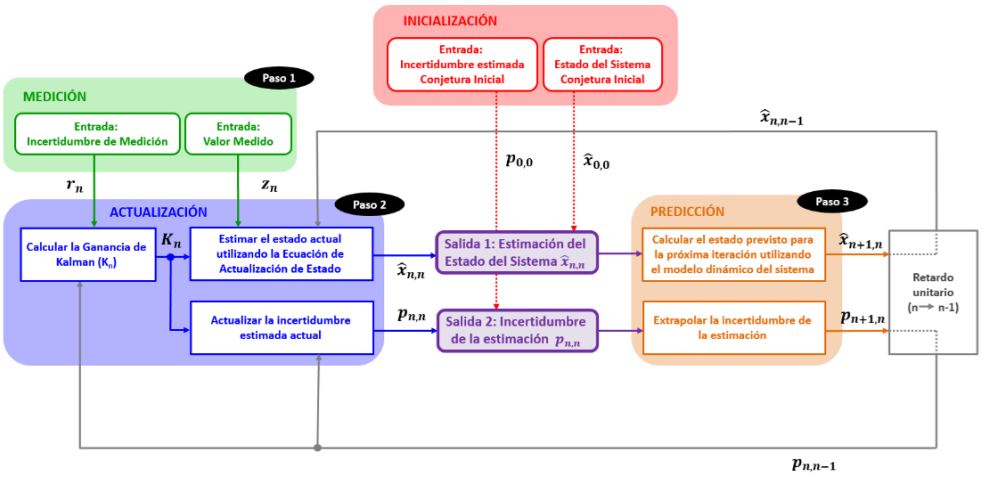

Figura B.2: *Diagrama completo filtro de Kalman. Extraído de: [44].*

Como anotación fuera de la explicación correspondiente a este apéndice, toda la funcionalidad aquí descrita tiene en cuenta filtros de Kalman en una dimensión. No obstante, el funcionamiento para N dimensiones sigue la misma línea teórica. Lo único que puede cambiar (además de la forma de las ecuaciones), es que se puede usar una notación matricial para optimizar el rendimiento. Detalles más concretos se pueden encontrar en [44].

C

Parámetros utilizados en las pruebas

En este apéndice se indica el valor que se le ha dado a cada parámetro configurable que genera diferencias en el resultado de los submódulos empleados en las pruebas.

C.1. *Versión 0*

C.1.1. Módulo "Detección-*Tracking*"

- **color_tracking:**

  - *color_extractor* (si con preprocesado)*:*
    - *min_contour_area* = 1000.
    - *h_range* = 2.
    - *sv_range1* = 75.
    - *sv_range2* = 175.
    - *size* = 5.
  - *hsv_range (si sin preprocesado)* = 168,140,69,175,255,198.
  - *non_max_suppresion_threshold* = 100.
  - Filtros de Kalman*:*
    - *dt* = 0.1.

- *u_x* = 15.
- *u_y* = 30.
- *std_acc* = 30.
- *x_std_meas* = 0.1.
- *y_std_meas* = 0.1.

- **Bg_substraction:**
  - *min_contour_area* = 1000.
  - *enclosing_area_diff* = 0.5.
  - *arc_const* = 0.1.

C.1.2. Módulo "Extracción *siteswap*"

- **Seq_extractor:**
  - (No requiere de parámetros adicionales).
- **Prediction:**
  - *num_balls* = Número de bolas del *siteswap*.
  - *test_numbers* = 5.

C.2. *Versión 1*

C.2.1. Módulo "Detección-*Tracking*"

- **color_tracking:**
  - *color_extractor* (si con preprocesado)*:*
    - *min_contour_area* = 1000.
    - *h_range* = 2.

- *sv_range1* = 75.
- *sv_range2* = 175.
- *size* = 5.

- *hsv_range (si sin preprocesado)* = 168,140,69,175,255,198.
- *non_max_suppresion_threshold* = 39.
- Filtros de Kalman*:*
  - *dt* = 0.3.
  - *u_x* = 1.
  - *u_y* = 3.
  - *std_acc* = 8.5.
  - *x_std_meas* = 0.1.
  - *y_std_meas* = 0.1.

C.2.2. Módulo "Extracción *siteswap*"

Mismos valores que en la anterior iteración.

C.3. *Versión 2*

C.3.1. Módulo "Detección-*Tracking*"

Mismos valores que en la anterior iteración.

C.3.2. Módulo "Extracción *siteswap*"

- **point_extractor:**
  - *min_contour_area* = 2500.
  - *y_mul_threshold* = 0.21.

- **seq_extraction_cuadrants:**
  - *point* = Punto extraído desde point_extractor.
  - *img_threshold* = 0.
- **prediction:**
  - *num_balls* = Número de bolas del *siteswap*.
  - *test_numbers* = 5.

C.4. *Versión final*

- *evaluate = True.*
- *tracking_system = "ColorTrackingMaxBalls".*
- *color_range* = 168,140,69,175,255,198.
- *max_balls* = Número de bolas del *siteswap*.
- *tracking_preprocessing = False.*
- *max_cuadrant_misses* = 0.49.
- *ss_test_numbers* = 5.
- *max_period_threshold* = 1.5.
- *decimal_round* = 3.