Tesis de Licenciatura

Detección y Reconocimiento de las Señas del Juego del Truco en Tiempo Real

Un algoritmo basado en Templates Temporales

Integrante	LU	Correo electrónico
Avendaño, Santiago	113/06	savendano@dc.uba.ar
Castillo, Gonzalo	164/06	gcastillo@dc.uba.ar

30 de Julio de 2012

Directores

Marta Mejail marta@dc.uba.ar

Norberto Goussies ngoussie@dc.uba.ar

DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (54 11) 4576-3359

http://exactas.uba.ar/

Resumen

En este trabajo presentamos un método para detectar y reconocer las señas del juego de Truco, las cuales conforman un subconjunto de gestos faciales. El método utiliza Templates Temporales para representar movimiento y posteriormente extraer características. El método propuesto funciona en tiempo real, lo que permite utilizarlo como un mecanismo de interacción entre un humano y una computadora, por ejemplo, en el contexto de una partida de Truco. De acuerdo a nuestro conocimiento este es el primer trabajo que utiliza la detección de gestos faciales en el contexto de un juego.

Palabras Claves: *Gesto Facial, Templates Temporales, Truco (juego de cartas)*

Abstract

In this work we present a method to detect and recognize the signs of the card game of Truco which are a subset of facial gestures. The method uses Temporal Templates to represent motion and later extract features. The proposed method works in real time, allowing to use it as an human-computer interface, for example, in the context of the card game of Truco. To the best of our knowledge, this is the first work that uses detection of facial gestures in the context of a game.

Keywords: *Facial Gesture, Temporal Templates, Truco (card game)*

Índice general

1. Introducción **1**
- 1.1. Contexto de trabajo . . . 1
- 1.2. Motivación . . . 2
- 1.3. Objetivos . . . 2
- 1.4. Trabajos Relacionados . . . 3
- 1.5. Organización del trabajo . . . 4

2. Conceptos Iniciales **5**
- 2.1. Representación de Imágenes . . . 5
  - 2.1.1. Representación de imágenes mediante matrices . . . 6
- 2.2. Representación de Color . . . 7
  - 2.2.1. Espacio de color *Red Green Blue (RGB)* . . . 7
  - 2.2.2. Espacio de color *Hue Saturation Value(HSV)* . . . 8
  - 2.2.3. Conversión a escala de grises . . . 9
- 2.3. Captura de Video . . . 9
- 2.4. Conclusiones del capítulo . . . 10

3. Detección y Seguimiento de Caras **11**
- 3.1. Detección de rostros . . . 12
  - 3.1.1. Definición del modelo . . . 12
  - 3.1.2. Clasificación utilizando haar-like features . . . 14
- 3.2. Seguimiento de rostros en capturas de video . . . 16
  - 3.2.1. Coherencia Temporal . . . 16
  - 3.2.2. Seguimiento de rostros por color: *Camshift* . . . 17
- 3.3. Conclusiones del capítulo . . . 19

4. Representación de movimiento **21**
- 4.1. Representación de movimiento usando Templates Temporales . . . 21
- 4.2. Motion History Image y Motion Energy Image . . . 23
  - 4.2.1. Trabajos Relacionados . . . 23
- 4.3. Descripción del método . . . 24
  - 4.3.1. Detección de movimiento . . . 26
- 4.4. Consideraciones al cálculo del MHI . . . 27
  - 4.4.1. Algoritmo . . . 28
- 4.5. Variantes del Método . . . 30
  - 4.5.1. El problema de la oclusión de movimientos . . . 30
  - 4.5.2. MHI Direccionales (DMHI) . . . 31
- 4.6. Conclusiones del capítulo . . . 34

5. Extracción de Características **35**
- 5.1. Extracción de características en trabajos previos . . . 35
- 5.2. Segmentación de Imágenes de Movimiento . . . 36
  - 5.2.1. Segmentación Cuadrícula . . . 36
  - 5.2.2. Segmentación por Regiones Faciales . . . 36
  - 5.2.3. Segmentación Quadtree . . . 38

5.2.4. Segmentaciones Adicionales . . . 40
5.2.5. Consideraciones Adicionales . . . 40
5.3. Cálculo de características de movimiento . . . 40
5.3.1. Proporción de Movimiento . . . 40
5.3.2. Momentos Geométricos . . . 41
5.4. Relación con los métodos de clasificación . . . 42
5.5. Conclusiones del capítulo . . . 43

6. Clasificación de Gestos **45**
6.1. Clasificadores . . . 45
6.1.1. Método de entrenamiento de un sistema de clasificación supervisado . . . 46
6.2. Algoritmos de Aprendizaje Supervisado . . . 47
6.2.1. K-Vecinos más Cercanos (K-NN) . . . 47
6.2.2. Máquinas de Vectores de Soporte (SVM) . . . 48
6.3. Aplicación de los algoritmos de clasificación . . . 53
6.4. Algoritmos de rechazo . . . 54
6.4.1. SVM de una clase *(One Class SVM)* . . . 55
6.4.2. Umbralización de probabilidades . . . 55
6.4.3. SVM binario . . . 55
6.5. Conclusiones del capítulo . . . 55

7. Detección de Intervalos Temporales de Movimiento **57**
7.1. Detección de intervalos . . . 57
7.1.1. Proporción de movimiento . . . 58
7.1.2. Algoritmo . . . 58
7.2. Rechazo de intervalos . . . 59
7.3. Integración del sistema . . . 60
7.4. Conclusiones del capítulo . . . 61

8. Experimentos **63**
8.1. Experimentos sobre parámetros de Motion History Image . . . 63
8.1.1. Parámetro α . . . 63
8.1.2. Parámetro β . . . 63
8.1.3. Parámetro μ . . . 64
8.2. Experimentos sobre reconocimiento de gestos . . . 64
8.2.1. Bases de Videos utilizadas . . . 65
8.2.2. Experimento intra-base . . . 66
8.2.3. Experimento inter-base . . . 66
8.2.4. Selección de métodos . . . 67
8.3. Comparación de algoritmos de rechazo . . . 68
8.4. Performance . . . 71
8.4.1. Tiempos variando el tamaño de los *frames* . . . 72
8.4.2. Tiempos del algoritmo . . . 73

9. Conclusiones Finales **75**
9.1. Limitaciones del sistema desarrollado . . . 75
9.1.1. Reconocimiento de gestos mediante *Templates Temporales* . . . 75
9.1.2. Algorimo de detección de intervalos temporales . . . 76
9.1.3. Lenguaje utilizado . . . 76
9.1.4. Sensibilidad en los parámetros seleccionados . . . 76
9.1.5. Restricciones definidas . . . 76
9.2. Objetivos Alcanzados . . . 77
9.3. Trabajo Futuro . . . 77

Apéndices **79**

Ejecución del cálculo de la MHI **81**

Agradecimientos

Esta tesis no sólo refleja el trabajo realizado por nosotros como alumnos de la carrera, sino que es el resultado del aporte en mayor o menor medida de personas que nos acompañaron en el arduo pero gratificante camino que significó llevar a cabo este proyecto.

En primer lugar no queremos dejar de lado a nuestros directores Marta Mejail y Norberto Goussies quienes no sólo nos dieron su aporte académico en el transcurso de la tesis sino que realmente los consideramos parte un grupo humano excelente el cual nos brindó su preocupación, paciencia y apoyo para desempeñar adecuadamente nuestro trabajo. Tampoco queremos olvidarnos de Julio Jacobo quien inicialmente formó parte de este equipo y no podemos dejar de darle palabras de agradecimiento por el trabajo realizado durante el tiempo que estuvo con nosotros.

No nos vamos a olvidar de todas las personas que integran el grupo de Imágenes del Departamento de Computación quienes nos permitieron compartir su espacio y tiempo cada vez que nos hizo falta.

Queremos agradecer también a todas las autoridades del Departamento de Computación quienes nos permitieron transitar la carrera de Licenciatura en Ciencias de la Computación manera ordenada y eficiente y por supuesto a todos los profesores que tuvimos a lo largo de la misma quienes nos hicieron crecer académicamente.

Por último ponemos nuestro sincero agradecimiento a todos los argentinos que mediante el pago de sus impuestos hacen posible que siga subsistiendo la posibilidad de formar profesionales académicos con reconocimiento mundial de manera gratuita en una sobresaliente institución como la Universidad de Buenos Aires.

A todos ellos, muchas gracias.

Agradecimientos de Gonzalo

Es una sensación muy satisfactoria poder estar escribiendo estas líneas, significa que una etapa muy linda ya pasó y está por venir una mucho mejor. Sin embargo, quiero expresar mi agradecimiento particular a cada una esas personas cercanas que justamente me permitieron mirar con ojos esperanzadores todo el trayecto realizado.

A vos mamá por estar siempre ahí, pendiente de cualquier detalle, interesada en ver cómo todo se iba desarrollando. Gracias por darme palabras de aliento para seguir y seguir aún cuando mi propio instinto me lo impedía.

A vos viejo, siempre fuiste un cable a tierra para permitirme abrir la cabeza y no ahogarme un vaso de agua. Siempre pudiste encontrar una forma de despejarme para disfrutar del momento.

A mis hermanos por estar ahí cada vez que los necesité. Espero que vos Hernán en breve te conviertas en uno de los nuestros, futuro computador.

A mi novia Natalia, de la cual debería escribir una tesis de agradecimientos. Realmente agradezco tu paciencia, tu adaptación a mis tiempos y por sobre todo por ver a través de mis ojos aceptando cualquier comportamiento muchas veces incorrecto por parte mía.

A mis amigos del barrio, compañeros de la vida, de los cuales nunca recibí un cuestionamiento por falta de tiempo en las juntadas, es más me hicieron pasar cada momento como si fuera el mejor. También a mis otros amigos, a los que conocí hace siete años atrás al empezar la carrera. Gracias por cada jornada de estudio, salida y por sobre todo por seguir viéndonos hoy en día.

A Santi, que hiciste que todo este proyecto fuese increíblemente cómodo. Envidio sanamente tu frialdad para decir las palabras justas en los momentos calientes. Agradezco sinceramente lo que me hiciste crecer como persona en tan poco tiempo, marcándome respetuosamente los errores que comúnmente cometo. Fue un lujo haberte tenido de compañero de tesis.

Finalmente a mis compañeros de trabajo, por aceptar cada tiempo que necesité dedicarle a esta tesis, sin poner impedimento alguno.

Todos de algún modo me ayudaron a que hoy pueda estar acá, así que muchas gracias!!

Agradecimientos de Santiago

Me toca en este momento, dar las gracias a aquellas personas que también fueron parte de este trabajo, y aportaron, cada uno a su forma, algo para que el mismo hoy esté finalizado.

Primero que nada tengo que agradecer a mis viejos. A Tata por demostrarme que con actitud todo se puede lograr y por transmitirme sus ganas de superarse. Por su sencillez y sus valores. A mami por su gran amor, sus fuerzas y su manera de hacerme sentir siempre cerca. Por haberme visto dejar mi casa hace 7 años atrás y alentarme siempre a seguir haciendo lo que más me gusta.

A Pauli, por hablarme con sus silencios y por mostrarme una forma diferente de ser.

A toda mi familia, por recibirme cada vez que vuelvo y hacerme sentir siempre acompañado. Especialmente, a la abu Tina, por sus mates, sus charlas y sus consejos, y a mis abuelos que siguen cuidándome desde arriba.

A mis amigos y amigas de Chivilcoy, el pilar donde me apoyo cada vez que trastabillo. Por ser un grupo único, por sus consejos, por sus comidas y por estar en las buenas y en las malas.

A mis amigos y amigas de Buenos Aires (de la facu, del fútbol y de mis trabajos) por abrirme las puertas de sus casas, recibirme y hacerme sentir como en la mía. No saben cuánto aportó su compañía para que yo pueda seguir adelante y abrirme camino en Buenos Aires.

A Chivilcoy, mi lugar en el mundo, mi cable a tierra.

A mi escuela y mis profesores, por transmitirme esas ganas interminables de seguir aprendiendo.

A Gonza mis gracias infinitas. Por elegirme desde un principio para realizar este trabajo y mantener esta elección durante este año y medio. Por bancarse mis locuras y mi forma de ser. Hicimos un gran trabajo y es muy bueno hoy compartir estos logros con vos.

Finalmente, a la Vida, por haberme dado tanto.

Capítulo 1

Introducción

Entendemos por gesto al movimiento de una parte del cuerpo con la intención de transmitir un mensaje. El reconocimiento de gestos es uno de los mayores desafíos para lograr desarrollar sistemas avanzados de interacción no-verbal entre un humano y una computadora.

Dentro de los diferentes tipos de gestos que podemos encontrar, el presente trabajo abarca aquellos caracterizados por ser voluntarios y porque la persona con la que se está interactuando conoce su significado (por ejemplo, asentir con la cabeza). Nos enfocamos en la detección y reconocimiento de gestos faciales, es decir, aquellos que pueden ser logrados mediante la contracción o relajación de uno o varios músculos faciales. Paul Ekman [1, 2], ha estudiado los posibles movimientos faciales que una persona puede hacer, clasificándolos en 46 *"Action Units (AU)"*. En el presente trabajo presentamos un algoritmo para detectar, con cierto grado de confiabilidad, la realización de un subconjunto de estos movimientos en tiempo real.

Este capítulo se organiza de la siguiente manera: en la sección 1.1 describimos el contexto a partir del cual se desarrolló este trabajo. En la sección 1.2 comentamos las motivaciones que nos incentivaron a realizar el mismo. En la sección 1.3 definimos los objetivos concretos del sistema implementado. En la sección 1.4 se hace referencia a los trabajos en los cuales basamos nuestra implementación. Finalmente en la sección 1.5 comentamos el esquema del sistema implementado y cómo el mismo es analizado en el resto de los capítulos que componen esta tesis.

1.1. Contexto de trabajo

El *"Truco"* es un juego de cartas de origen español, difundido con variantes por España y el sur de América Latina. Uno de sus principales objetivos es engañar al contrincante con el fin de obtener más puntos. Se puede jugar de 2, 4 o 6 personas. Los jugadores se dividen en 2 equipos. Para poder comunicar las cartas que cada jugador tiene a sus compañeros de equipo, sin que el contrario se entere, se dispone de un conjunto de señas faciales. Así los jugadores de un mismo equipo pueden transmitirse aquellas cartas de mayor valor y elaborar una estrategia común, sin que el equipo contrario pueda percibirlo.

Con algunas variaciones se aceptan las señas que se describen en la tabla 1.1. La tercer columna de la tabla indica el gesto correspondiente según la notación utilizada por Ekman en [2]. Para los gestos correspondientes a siete de espada y siete de oro no existe una *Action Unit* correspondiente en el sistema FACS.

Carta	Seña	FACS
Ancho de Espada (As de Espada)	Levantar ambas cejas	AU1+AU2
Ancho de Basto (As de Basto)	Cerrar el ojo derecho	AU46
Siete de Espadas	Estirar hacia derecha la comisura de los labios	n/a
Siete de Oro	Estirar hacia izquierda la comisura de los labios	n/a
Cualquier 3	Morder el labio inferior	AU32
Cualquier 2	Colocar los labios como si se tirase un beso	AU18
Anchos Falsos (As de Copa/Oro)	Abrir y cerrar la boca	AU25+AU27

Tabla 1.1: Descripción de los gestos que queremos detectar junto con su correspondiente AU según la notación del método FACS definido por Ekman en [2].

Debido a que el sistema FACS no se ajusta correctamente a las señas que se quieren detectar en nuestro trabajo, decidimos referenciar a las mismas con nuestra propia notación, asignándoles el nombre de la carta que representan. En la figura figura 1.1 se muestran cada una de las señas asociadas a las cartas que representan.

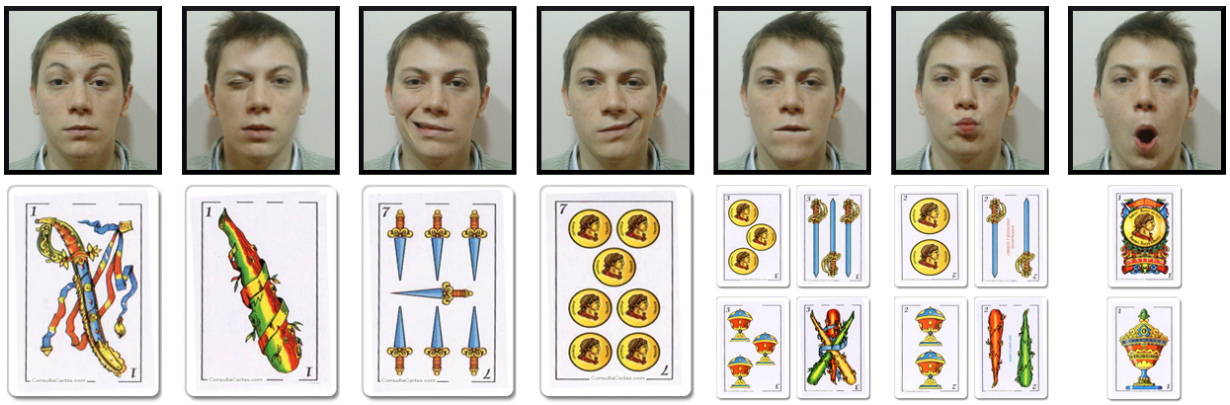

Figura 1.1: Señas del Truco identificando las cartas que representan.

1.2. Motivación

Como se comentó anteriormente el reconocimiento de gestos es uno de los mayores desafíos para lograr desarrollar sistemas avanzados de interacción no-verbal entre un humano y una computadora. En la sección 1.4 se analizan varios trabajos relacionados con el reconocimiento de gestos. El primero de los motivos que nos llevó a realizar esta tesis fue la posibilidad de interiorizarnos en un problema difícil y nuevo para nosotros.

Por otro lado, en nuestro país y en gran parte de Latinoamérica el juego del Truco es popular y muchas veces motivo de reunión entre pares y amigos. Desde este punto de vista consideramos que integrar tecnología y cultura se corresponde con una motivación personal para llevar a cabo este trabajo. Existen implementaciones del juego del Truco que permiten jugar partidas contra una computadora. Sin embargo la complicidad para conformar una estrategia en un equipo es tanto o más interesante que la partida de Truco en sí misma. Para ello, es necesario la comunicación no verbal mediante las señas del Truco entre los integrantes del equipo.

A partir de esto, llevar a cabo una implementación que permita generar esa interacción no verbal entre un participante y una computadora en el contexto de este juego, reúne las condiciones necesarias para ser un sistema novedoso en nuestro país. En particular, nosotros enfocamos nuestro trabajo en la primera parte de dicha interacción, el reconocimiento de las señas generadas por un participante a una computadora, que además funcione en tiempo real, lo que permite que una partida sea tan fluida como cuando participan personas. Consideramos que la integración posterior a una estrategia común es factible dado el reconocimiento.

1.3. Objetivos

El sistema que implementamos tiene como objetivos detectar el subconjunto de gestos faciales, dados por las señas del Truco, satisfaciendo las siguientes características:

- Ser capaz de detectar gestos que se realizan frente a una cámara web o algún dispositivo de captura de video, en un lapso corto de tiempo (menos de medio segundo).
- Poder decidir entre un conjunto de gestos predefinidos con cual se corresponde un gesto, o si no se corresponde con ninguno de ellos.
- Realizar las tareas de detección y clasificación de los gestos en tiempo real.

Entendemos por *tiempo real* a la capacidad del sistema de procesar al menos 15 *frames* por segundo.

Los gestos son detectados en caras frontales, permitiendo pequeñas variaciones en la posición, escala, rotación e inclinación de la cabeza (ver Figura 1.2). Asimismo, el rostro deberá tener un tamaño considerable para que los movimientos sean detectados con claridad.

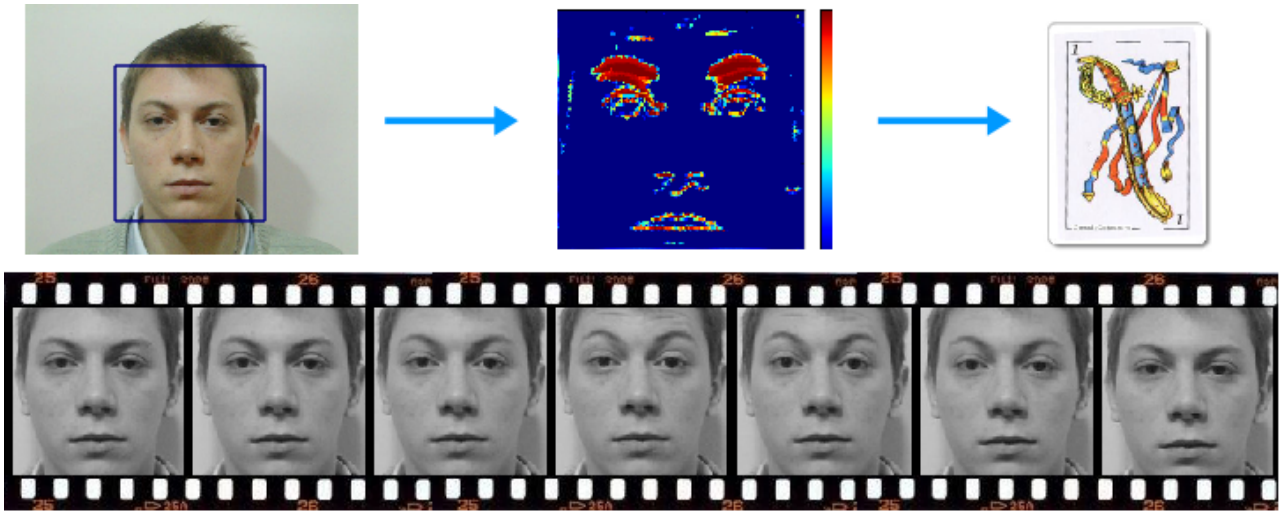

Figura 1.2: Detección del gesto del Ancho de Espada a partir de una captura de video.

1.4. Trabajos Relacionados

En este apartado enumeramos los trabajos relacionados con el reconocimiento de gestos y sus diferentes variantes.

Inicialmente en [3] (Zelinsky, Heinzmann) se utilizaron filtros de Kalman para predecir y seguir la posición de ciertas características faciales en tiempo real. A partir de estas posiciones se pudieron detectar gestos como: sí, no, tal vez, mirar hacia arriba o abajo, etc.

En [4, 5, 6, 7] se describieron algunos métodos para detectar gestos dentro de un rostro humano. En [4] (Algorri, Escobar) se construyeron espacios de gestos faciales usando el método de *eigen-faces*. Usando estos espacios, los gestos faciales pudieron ser representados por un número limitado de descriptores de gestos faciales. En [5] (Naghsh-Nilchi) se estudió un método basado en flujo óptico para detectar y reconocer las 6 expresiones faciales universales (felicidad, tristeza, sorpresa, miedo, enojo y disgusto). En [6] (La Cascia, Valenti y Sclaroff) se construyó un modelo cilíndrico en 3D con el que se obtuvo la posición y se realizó el seguimiento de la cara. A partir de este modelo, se asumió que el error residual podía ser modelado como una combinación lineal de *templates* de movimiento facial y a partir de esto deducir que expresión se estaba realizando. Asimismo, en [7] (Liao, Cohen), también se utilizó un modelo 3D, para detectar la posición y la inclinación de la cara. Además se estudió el problema de detectar gestos faciales en presencia de movimiento de la cabeza.

En [8, 9, 10] se estudió el problema de reconocimiento de las *Action Units* definidas por Ekman en [1]. En [8] (Valstar, Patras y Pantic) se abarcó este problema a partir la noción de *Templates Temporales*, utilizados inicialmente en [11] (Davis, Bobick) para reconocer movimientos corporales. Para clasificar se utilizó el método de vecinos más cercanos en conjunto a un sistema de reglas. En [9] (Valstar, Pantic) se detectaron *Action Units*, así como sus etapas de desarrollo y duración, a partir de características espacio-temporales calculadas mediante el seguimiento de 20 puntos faciales. Estos puntos eran detectados y trackeados automáticamente. Tanto los gestos, como sus segmentos temporales, fueron reconocidos mediante *Supports Vectors Machines*. En [10] (Koelstra,Pantic y Patras) se propusieron dos enfoques basados en texturas dinámicas para reconocer *Action Units* y sus modelos temporales. El primero es una extensión del método de *Templates Temporales* basados en *Motion History Image*. El segundo método se basa en *Nonrigid Registration* utilizando *Free-Form Deformations*.

Finalmente en [12] (Mitra, Acharyase) se analizaron las herramientas disponibles y los diferentes enfoques para realizar el reconocimiento de gestos, ya sean faciales como manuales.

Nuestro enfoque sigue lo propuesto en [8, 10] con el uso de los *Templates Temporales* basados en *Motion History Images*. Se utiliza además el concepto de *Directional Motion History Images* descripto en [13] para complementar el método. Si bien las señas del Truco se corresponden con un subconjunto de *Actions Units*, nuestro desafío está enfocado en lograr detectar las mismas, de manera confiable, mediante un método que funcione en tiempo real. Para ello adaptamos las distintas metodologías a este contexto de trabajo particular, a fin de lograr los objetivos propuestos.

1.5. Organización del trabajo

El sistema implementado se descompone en módulos con funcionalidades bien definidas descriptos en la figura 1.3. Esto nos permite analizar variantes en cada uno de ellos de manera independiente y en pos de alcanzar los mejores resultados en conjunto.

Como primer paso existe un módulo que a partir de una captura de video detecta la región facial (módulo *"Seguimiento del Rostro"*). El siguiente módulo *"Actualización de Templates Temporales"* se encarga de obtener los movimientos generados en la captura, distinguiendo sólo aquellos que se ubican en la región del rostro detectado. A partir de los movimientos detectados se identifican segmentos temporales en donde ocurre un movimiento significativo, en el módulo *"Detección de Intervalos Temporales de Movimiento"*. Tomando en cuenta los movimientos detectados dentro del segmento se extraen características (módulo *"Extracción de Características"*) que, mediante un proceso de clasificación, se pueden asociar a una seña del Truco (módulo *"Clasificación"*). La combinación de todos los módulos en conjunto permite satisfacer la funcionalidad mencionada. La figura 1.3 resume la interacción de los módulos en el sistema general.

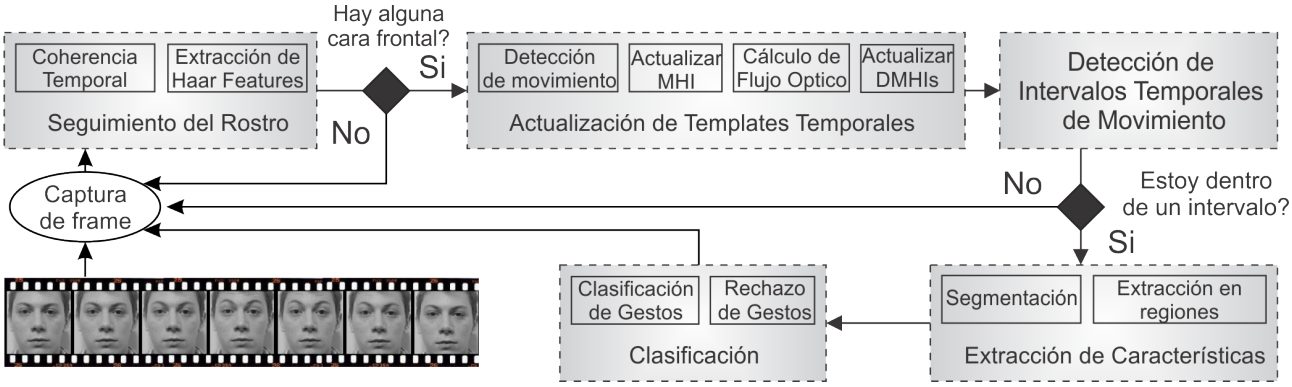

Figura 1.3: Esquema de composición del sistema general a partir de módulos.

El presente trabajo está organizado a modo de analizar cada módulo que compone al sistema de la siguiente forma:

- En el capítulo 2 definimos algunos conceptos básicos que son tenidos en cuenta en el resto de los capítulos.
- En el capítulo 3 analizamos el algoritmo implementado para la detección de un rostro en una imagen y seguimiento del mismo a lo largo de la secuencia de *frames* (módulo *"Seguimiento del Rostro"*).
- En el capítulo 4 describimos la forma de capturar los movimientos que se producen en el rostro detectado (módulo *"Actualización de Templates Temporales"*).
- En el capítulo 5 presentamos las diferentes variantes para extraer características de los movimientos capturados (módulo *"Extracción de Características"*).
- En el capítulo 6 estudiamos los métodos de clasificación utilizados para reconocer el gesto que se está realizando a partir de las características extraídas (módulo *"Clasificación"*).
- En el capítulo 7 desarrollamos un método para localizar los segmentos temporales de un video en donde es factible que se esté ejecutando un gesto (módulo *"Detección de Intervalos Temporales de Movimiento"*).
- En el capítulo 8 se realizan diversos experimentos para mostrar la eficacia y eficiencia del método.
- En el capítulo 9 se extraen las conclusiones sobre el trabajo total realizado en esta tesis.

Capítulo 2

Conceptos Iniciales

En este capítulo definimos una serie de conceptos básicos referidos al procesamiento de imágenes y video, que se utilizan a lo largo de este trabajo.

En la sección 2.1 definimos el concepto de imagen digital y la representación que utilizamos para la misma en el desarrollo de la tesis. En la sección 2.2 profundizamos en la forma de describir imágenes digitales en color y analizamos los diferentes modelos de representación del mismo. Si bien se explican los conceptos básicos del modelo de color HSV(*Hue Saturation Value*) el mismo es tenido en cuenta posteriormente en un algortimo estudiado pero descartado en la implementación, con lo cual su lectura es prescindible. En la sección 2.3 analizamos la forma de representación de una captura de video y los elementos que la componen. Finalmente en 2.4 se extraen algunas conclusiones sobre los conceptos definidos.

2.1. Representación de Imágenes

En esta sección se formaliza el concepto de imagen digital y la forma de representación que utilizamos para la misma a lo largo de esta tesis. Como se explica en [14]:

Def. 2.1. Una *imagen* puede ser definida como una función bidimensional $f(x, y)$ donde x e y son coordenadas en el plano espacial. El valor de f en cualquier par de coodenadas (x, y) es llamada *intensidad* o *nivel de gris* de la imagen en ese punto.

La definición 2.1 es válida para imágenes monocromáticas (en escala de grises). En la sección 2.2 analizamos cómo representar colores en las imágenes.

Def. 2.2. Llamamos *imagen digital* a una imagen $f(x, y)$ que cumple que x, y y los valores de f son todos valores discretos.

La definición 2.2 permite notar que una imagen digital está compuesta por un número finito de elementos, cada uno de los cuales tiene una posición y un valor. Estos elementos son llamados *pixeles*.

Como se comentó anteriormente, en una imagen monocromática, cada valor de f se corresponde a un determinado valor en una escala de grises. La misma está comprendida en un rango de 0 a 255 en el cual cada valor está asociado a distintas graduaciones de gris, desde el negro cuyo valor es el 0, hasta el blanco cuyo valor es 255 como puede verse en la figura 2.1.

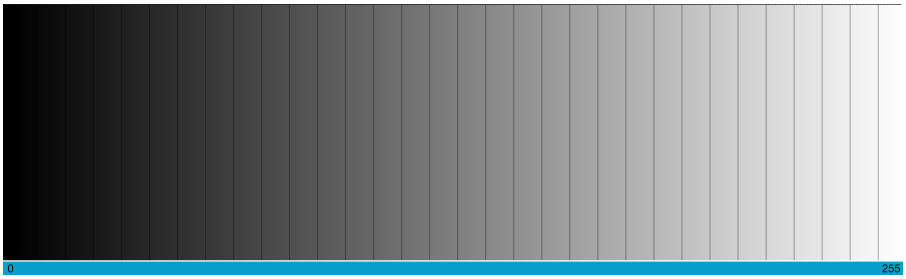

Figura 2.1: Imagen de todos los valores en la escala de grises. Las líneas divisorias agrupan valores cada 8 niveles de gris.

A continuación, se formaliza el concepto de imagen digital monocromática teniendo en cuenta los conceptos mencionados.

Def. 2.3. Una imagen digital monocromática de tamaño $m \times n$, con $m, n \in \mathbb{N}$, puede definirse mediante una función $f : \mathbb{A} \longrightarrow \mathbb{Z}$, con $\mathbb{A} = \{x, y : 1 \leq x \leq m, 1 \leq y \leq n\}$ tal que:

$$0 \leq f(x, y) \leq 255 \quad \forall (x, y) \in \mathbb{A}$$

2.1.1. Representación de imágenes mediante matrices

Una matriz es un conjunto finito de elementos dispuestos en filas y columnas donde cada fila es una entrada horizontal en una matriz y cada columna una entrada vertical en la misma (ver Figura 2.2). Una matriz de tamaño $m \times n$, con $m, n \in \mathbb{N}$, es aquella que tiene m filas y n columnas, lo que determina una cantidad de $m * n$ elementos en la matriz. Vamos a denotar al conjunto de matrices de tamaño $m \times n$ como $\mathbb{K}^{m \times n}$ en donde $\mathbb{K}$ es el cuerpo o campo de los elementos de la matriz.

$$A \in \mathbb{Z}^{3\times 3} = \overbrace{\left.\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right\} Filas}^{Columnas}$$

Figura 2.2: Matriz de tamaño 3 x 3 con valores enteros.

Algunos conjuntos de matrices que utilizamos en la tesis son los siguientes:

- $\mathbb{N}^{m \times n}$ o conjunto de matrices de tamaño $m \times n$ de naturales.
- $\mathbb{Z}^{m \times n}$ o conjunto de matrices de tamaño $m \times n$ de enteros.
- $\mathbb{R}^{m \times n}$ o conjunto de matrices de tamaño $m \times n$ de reales.
- $\mathbb{B}^{m \times n}$ o conjunto de matrices de tamaño $m \times n$ de valores de verdad *true* o *false* (verdadero a falso) más comúnmente denominados valores booleanos. Como $\mathbb{B}$ no representa un cuerpo algebraico, no son tenidas en cuenta las operaciones de adición y multiplicación sobre estas matrices.

Cada elemento que forma una matriz $A \in \mathbb{K}^{m \times n}$ lo vamos a denotar como $A(i, j)$ con $1 \leq i \leq m$, $1 \leq j \leq n$, donde i corresponde a la fila y j a la columna donde se encuentra el elemento dentro de la matriz A.
Algunas operaciones entre matrices que utilizamos en este trabajo son:

- *Adición:* Si A y B son matrices tales que $A, B \in \mathbb{K}^{m \times n}$, entonces vale que la suma de A y B es una matriz $C \in \mathbb{K}^{m \times n}$ tal que:

$$C(i, j) = A(i, j) + B(i, j) \qquad \forall i, j : 1 \leq i \leq m, 1 \leq j \leq n \tag{2.1}$$

- *Producto escalar:* Si $A \in \mathbb{K}^{m \times n}$ y $s \in \mathbb{K}$, entonces vale que el producto escalar de s y A es una matriz $C \in \mathbb{B}^{m \times n}$ tal que:

$$C(i, j) = s * A(i, j) \qquad \forall i, j : 1 \leq i \leq m, 1 \leq j \leq n \tag{2.2}$$

Una imagen digital puede representarse mediante una matriz $I \in \mathbb{Z}^{m \times n}$ en donde n representa el ancho de la imagen y m el alto de la misma. Los índices $1 \leq i \leq m$, $1 \leq j \leq n$ determinan la ubicación de cada pixel dentro de la imagen y el valor correspondiente al elemento de la matriz en esa posición se corresponde con el nivel de gris de dicho pixel. La imagen 2.3 muestra la representación de una imagen mediante una matriz.

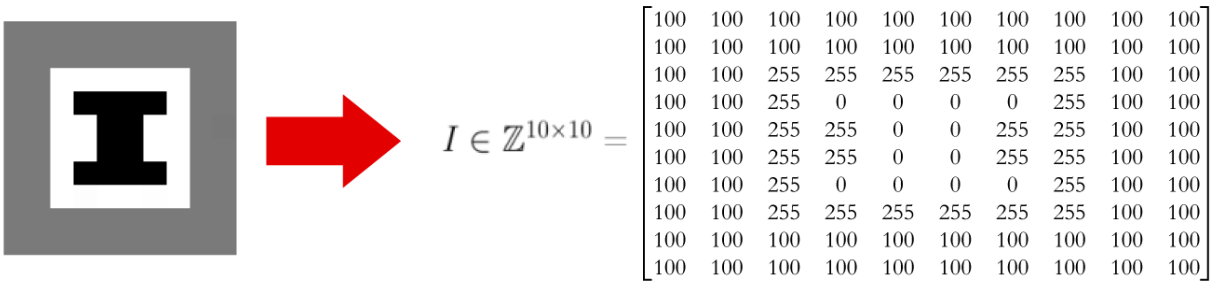

Figura 2.3: Representación de una imagen en escala de grises mediante una matriz.

2.2. Representación de Color

Un espacio de color proporciona un método para representar colores en imágenes digitales. Por lo general un espacio de color está representado por uno o más canales cuya combinación lineal genera todo el espacio de color. Una imagen en escala de grises tiene un solo canal que comprende los distintos niveles de gris en un rango de 0 a 255.

A continuación definiremos el concepto de imagen digital en un espacio de color.

Def. 2.4. Una *imagen digital* de tamaño $m \times n$, con $m, n \in \mathbb{N}$, en un espacio de color γ compuesto por $c \in \mathbb{N}$ canales puede definirse mediante una función $f : \mathbb{A} \longrightarrow \mathbb{Z}^c$, con $\mathbb{A} = \{x, y \in \mathbb{N} : 1 \leq x \leq m, 1 \leq y \leq n\}$ tal que:

$$f(x, y) \in < Rango_1^\gamma, \ldots, Rango_i^\gamma, \ldots, Rango_c^\gamma >$$

en donde cada $Rango_i^\gamma$ con $1 \leq i \leq c$ representa la escala de valores asociadas al canal i en el espacio γ.

A su vez, una imagen digital de tamaño $m \times n$ en el espacio de color γ con c canales, puede ser representada por una tupla $\mathbf{I}^\gamma =< I_1^\gamma, \ldots, I_c^\gamma >$ de c matrices en valores enteros de tamaño $m \times n$, es decir, $(\forall j : 1 \leq j \leq c)\ I_j^\gamma \in \mathbb{Z}^{m\times n}$. Además $(\forall x, y \in \mathbb{N} : 1 \leq x \leq m, 1 \leq y \leq n)\ I_j^\gamma(x, y) \in Rango_j^\gamma$

En este trabajo vamos a concentrarnos en los espacios de colores RGB y HSV que se describen en las secciones subsiguentes.

2.2.1. Espacio de color *Red Green Blue (RGB)*

El espacio de color RGB está compuesto por tres canales denominados rojo, verde y azul asociados a los que vamos a considerar colores primarios. Este espacio permite la composición del color en términos de la intensidad de los colores primarios lumínicos con los que se forma. Es decir, es posible representar un color mediante la mezcla de estos tres canales.

Cada uno de los valores de los canales que conforman el espacio RGB varían en un rango de 0 a 255. En la tabla 2.1 puede observarse la representación de algunos colores mediante este espacio.

Color	*Canal Rojo*	*Canal Verde*	*Canal Azul*
Rojo	255	0	0
Verde	0	255	0
Azul	0	0	255
Amarillo	255	255	0
Cian	0	255	255
Magenta	255	0	255
Negro	0	0	0
Blanco	255	255	255

Tabla 2.1: Representación de colores mediante el espacio RGB

En la figura 2.4 podemos observar la intensidad lumínica en escala de gris de cada uno de los tres canales del espacio RGB por separado para una imagen en color.

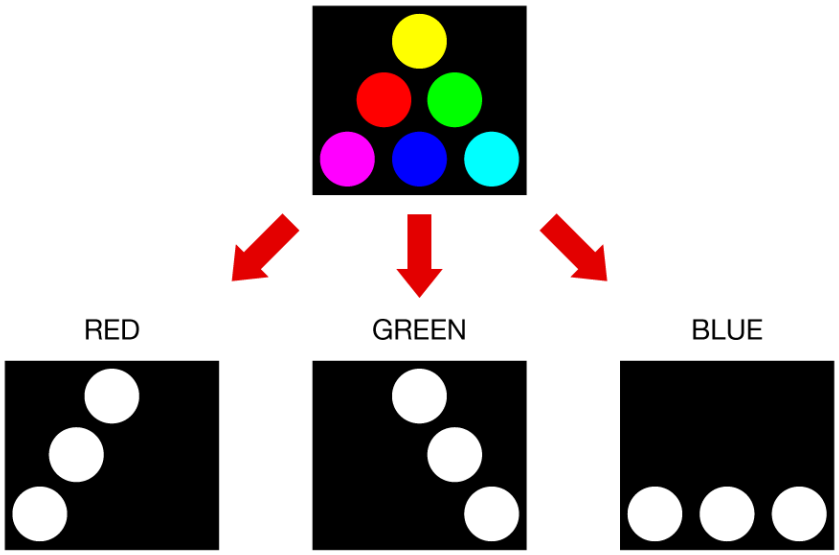

Figura 2.4: Separación de una imagen en los tres canales del espacio de color RGB.

2.2.2. Espacio de color *Hue Saturation Value(HSV)*

La descomposición HSV (del inglés *Hue, Saturation, Value*) describe un modelo de color a partir de las siguientes componentes:

- **Tonalidad (*Hue*):** representa el tipo de color (como rojo, azul o amarillo). Se refiere a la frecuencia dominante del color dentro del espectro visible. Se representa como un grado de ángulo cuyos valores posibles van de 0 a 360° (aunque para algunas aplicaciones se normalizan del 0 al 100). Cada valor corresponde a un color. Algunos ejemplos son: 0 es rojo, 60 es amarillo y 120 es verde.
- **Saturación (*Saturation*):** se refiere a la pureza relativa o cantidad de luz blanca mezclada con una tonalidad. Los colores puros del espectro están totalmente saturados. Colores como el rosa (blanco y rojo) están menos saturados, siendo su grado de saturación inversamente proporcional a la cantidad de luz blanca añadida.
- **Valor (*Value*):** es la intensidad de luz de un color. Dicho de otra manera, es cuan claro o cuan oscuro es el mismo. Los valores posibles van del 0 al 100.

En la Figura 2.5 se muestra una representación del espacio de color HSV.

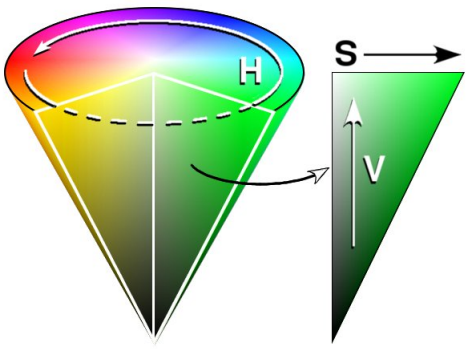

Figura 2.5: Espacio de Color HSV.

Luego, una imagen HSV está compuesta por tres canales, uno para cada componente, cuya combinación genera el color en cada pixel de la imagen.

2.2.3. Conversión a escala de grises

Las imágenes obtenidas por un dispositivo digital como un cámara fotográfica o de video se encuentran generalmente en el espacio de color RGB. Esto implica que para poder procesar una imagen se deben procesar cada uno de los canales que conforman la misma. En términos matriciales, se deben efectuar operaciones sobre las tres matrices que representan una imagen en el espacio de color RGB.

Uno de los objetivos primordiales de esta tesis es lograr un método que funcione en tiempo real. Para ello se busca reducir el procesamiento de tal manera que el objetivo sea alcanzado. Si el resultado de procesar imágenes de varios canales es equivalente al de procesar imágenes con uno único, entonces debemos contemplar estas últimas para reducir los tiempos.

En la mayoría de los algoritmos que empleamos en esta tesis nos bastará trabajar con imágenes en escala de grises compuestas por un único canal. El color no es un factor importante pero sí la intensidad de un determinado pixel con respecto al resto dentro de una imagen.

Para ello, debemos convertir las imágenes obtenidas mediante un dispositivo de captura de un espacio RGB a un espacio en escala de grises. Utilizando la representación matricial de imágenes, si consideramos $I_R, I_G, I_B \in \mathbb{Z}^{m\times n}$ como los tres canales de una imagen I de tamaño $m \times n$ en el espacio RGB, la imagen $I_Y \in \mathbb{Z}^{m\times n}$, que representa la conversión de I a escala de grises puede calcularse como:

$$I_Y = 0,299 * I_R + 0,857 * I_G + 0{,}114 * I_B \tag{2.3}$$

(redondeando los valores para que la imagen siga manteniendo valores enteros).

Si bien para obtener I_Y se podría calcular el promedio de los tres canales, se utilizan las constantes definidas, debido a que, a iguales cantidades de color el ojo humano es más sensible al verde, luego al rojo y finalmente al azul. Con lo cual para que la percepción humana de una imagen en color sea semejante a una imagen en escala de grises deben ponderarse estos valores.

En la figura 2.6 puede observarse la conversión de una imagen según el método comentado anteriormente.

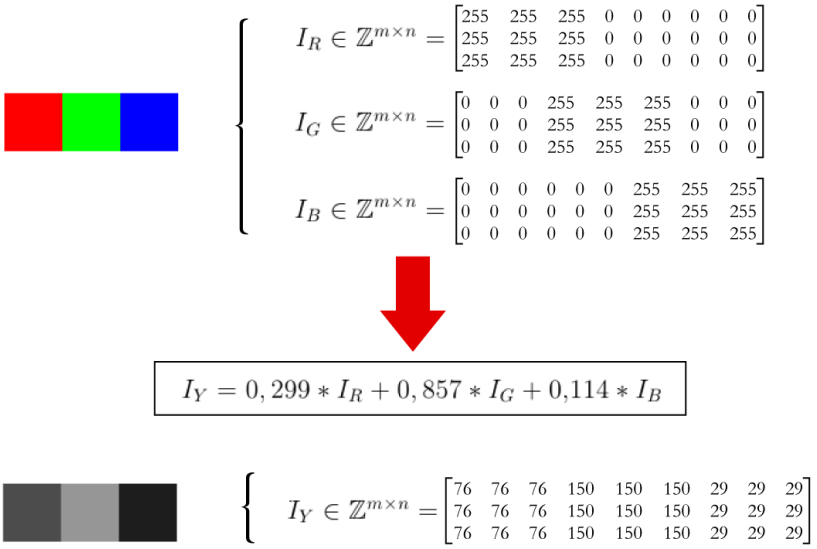

Figura 2.6: Conversión de una imagen del espacio de color RGB a escala de grises.

2.3. Captura de Video

Para este trabajo definimos una captura de video como una secuencia finita de imágenes de igual tamaño. Consideramos a la secuencia de imágenes como monocromáticas. De este modo notamos a una captura de video como $\mathbf{C} \in [\mathbb{Z}^{m\times n}]_t$ que representa una secuencia $\mathbf{I} = [I_1, \ldots, I_t]$ de t imágenes en valores enteros de tamaño $m \times n$, es decir, $(\forall j : 1 \leq j \leq t)\, I_j \in \mathbb{Z}^{m\times n}$.

Con la representación mencionada podemos determinar que una captura de video está compuesta por tres elementos fundamentales: *ancho, alto y cantidad de imágenes*. Además las capturas de video están caracterizadas por una medida de frecuencia denominada *Frame Rate* comúnmente expresada en frames por segundo *(Frames Per Second, FPS)*. Los FPS asociados a una captura nos indican la cantidad de imágenes o *frames* de la misma que se capturan o reproducen en un segundo. Básicamente la lógica de reproducción genera que una simple secuencia de imágenes reproducidas a una cierta velocidad den la ilusión de un movimiento fluido en una captura, similar a lo que se hace cuando se producen dibujos animados.

2.4. Conclusiones del capítulo

En este capítulo se introdujeron algunos conceptos que serán utilizados durante el resto de las secciones que conforman este trabajo. Se definió el concepto de imagen y su representación práctica para poder ser manipulada y procesada. Se tuvieron en cuenta los espacios de color utilizados para representar imágenes en colores y la transformación correspondiente a escala de grises. A partir del concepto de imagen, se definió una captura de video como una secuencia de las mismas. Esta definición permite observar que el hecho de procesar una captura de video básicamente significa procesar cada una de las imágenes que componen la misma de manera ordenada. El concepto de captura de video definido es de gran importancia ya que representa la entrada al sistema implementado en esta tesis.

Capítulo 3

Detección y Seguimiento de Caras

En el capítulo 2 se definieron, entre otras cosas, los conceptos de imagen y captura de video y se mencionaron las formas en que las representaremos en este trabajo. Esto es importante ya que una captura de video, es decir, una secuencia de imágenes, conforma la entrada de nuestro sistema.

En este capítulo nos enfocamos en el primer paso en la detección de las señas del juego de Truco. Las mismas tienen como ubicación el rostro de la persona que las genera. Para ello abordamos el tema de detección de rostros en capturas de video y el seguimiento de la cara detectada en la secuencia de *frames* que componen la captura utilizada como entrada. Esto nos permite reducir tamaño del problema a la región de una cara como puede observarse en la figura 3.1. Además se detallan los algoritmos utilizados para llevar a cabo estas tareas, las diferentes variantes evaluadas y se discuten las mejoras implementadas para el contexto de esta tesis.

Al reducir el problema a la región del rostro se descarta cualquier tipo de información adicional al mismo, es decir, todas aquellas regiones de la captura que no son la cara de una persona. Esto nos permite entre otras cosas no procesar información que no es tenida en cuenta para detectar señas del Truco. Como resultado, se reduce el tiempo de cómputo de manera considerable, siguiendo el objetivo de lograr que el sistema funcione en tiempo real.

Repasando el contexto de nuestro trabajo, el mismo se caracteriza por:

- Se cuenta con una captura de video.
- La captura contiene el rostro de una persona.
- El área en la que se ubica el rostro tiene un tamaño considerable.
- El rostro no se mueve excesivamente.
- Se detectan gestos solo cuando el rostro está frontal al dispositivo de captura.

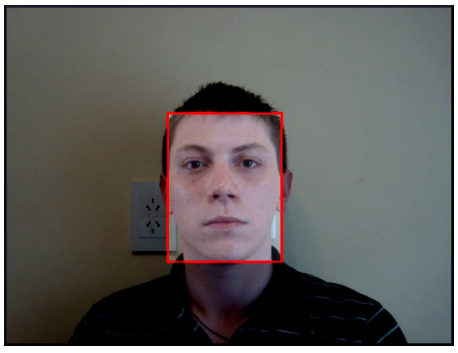

Figura 3.1: Detección del rostro en una imagen. El área de mayor oscuridad denota la región irrelevante para el sistema.

El capítulo se organiza de la siguiente forma: en la sección 3.1 se detallan los algoritmos de detección tenidos en cuenta durante la implementación de esta tesis, incluyendo sus variantes y determinando cuáles de ellas se utilizan como bases para trabajar. En la sección 3.2 se abordan diferentes formas de seguir un rostro detectado en

una captura de video y de define el concepto de *Coherencia Temporal* como la mejora utilizada para optimizar la performance descartando el algoritmo *Camshift* lo cual hace que su lectura sea prescindible. Se extraen algunas conclusiones finales en la sección 3.3.

3.1. Detección de rostros

Para realizar la detección de una cara en una imagen utilizamos el algoritmo de *Viola-Jones* definido en [15] y la extensión del mismo presentada por Lienhart en [16]. Debido a que este algoritmo funciona en tiempo real puede ser utilizado para realizar el seguimiento del rostro *frame* a *frame*.

3.1.1. Definición del modelo

El algoritmo de Viola-Jones es el primer algoritmo de detección de rostros que funciona en tiempo real. Su funcionamiento se basa en la extracción de un conjunto de características denominadas *haar-like features* inspiradas en las *haar-basis functions* propuestas por Papageorgiou en [17]. La ventaja principal de utilizar *haar-like features* es su gran velocidad de cómputo en diferentes posiciones y escalas sobre una misma imagen.

Una *haar-like feature* considera regiones rectangulares adyacentes, suma la intensidad de los pixeles que las mismas contienen y calcula las diferencias entre estas sumas. Estas diferencias de intensidades son usadas para describir subregiones de la imagen. El trabajo de Viola-Jones propuso tres tipos de *features*, que pueden observarse en la figura 3.2:

- **Dos Rectángulos:** El resultado se calcula a partir de la diferencia entre la sumatoria de los valores de los pixeles de dos rectángulos adyacentes (vertical u horizontalmente) que tienen el mismo alto y ancho.
- **Tres Rectángulos:** El valor de esta *feature* se calcula mediante la suma de los dos rectángulos externos a los cuales se le sustrae el valor del rectángulo interno.
- **Cuatro Rectángulos:** El resultado es la diferencia entre los pares de rectángulos diagonales.

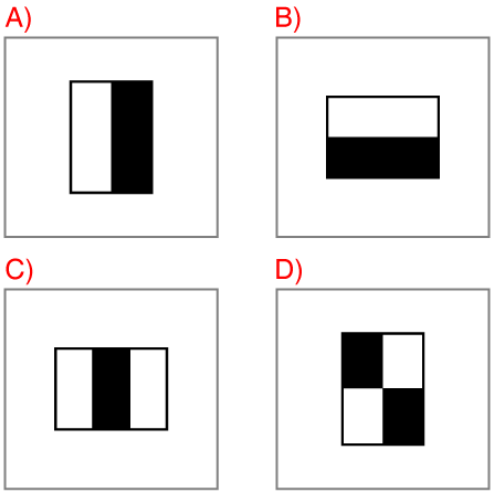

Figura 3.2: Ejemplo de los tres tipos de *features* propuestos por Viola-Jones. Las *features* en A) y B) se corresponden con dos rectángulos, C) con tres rectángulos y D) con cuatro.

La rapidez de este método se logra a partir del cálculo de una imagen intermedia denominada *imagen integral*. Esta imagen permite el cómputo de una *haar-like feature* a cualquier escala y en cualquier posición en tiempo constante.

En su trabajo, Lienhart extendió el conjunto de *features* definido por Viola-Jones. Para ello propuso utilizar, además de los rectángulos convencionales, rectángulos orientados 45°. A partir de esta idea define 14 prototipos de *features* que pueden verse en la figura 3.3. El conjunto de *features* se compone por: cuatro *Edge Features*, ocho *Line Features* y dos *Center-Surround Features*.

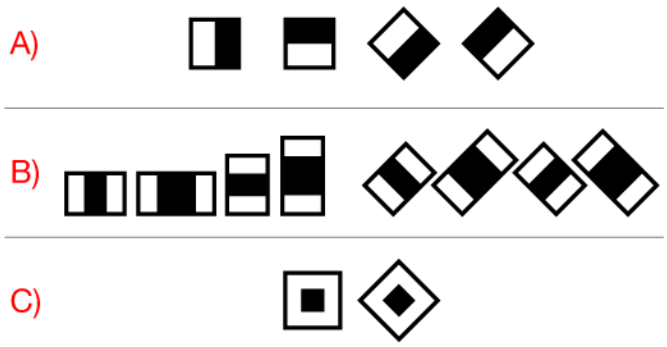

Figura 3.3: Conjunto de 14 prototipos de *fearures* propuestos por Lienhart: A) *Edge Features*, B) *Line Features*, C) *Center-Surround Features*.

De manera equivalente a Viola-Jones, Lienhart propuso una forma de calcular la suma de pixeles correspondiente a los rectángulos de las *features*. Para los rectángulos con orientación 0° definió una imagen auxiliar denominada *SAT(Summed Area Table)* equivalente a la imagen integral propuesta por Viola-Jones:

$$SAT(x,y) = \sum_{i \leq x, j \leq y} I(i,j) \tag{3.1}$$

Una representación gráfica del concepto de imagen integral o SAT puede verse en la figura 3.4.

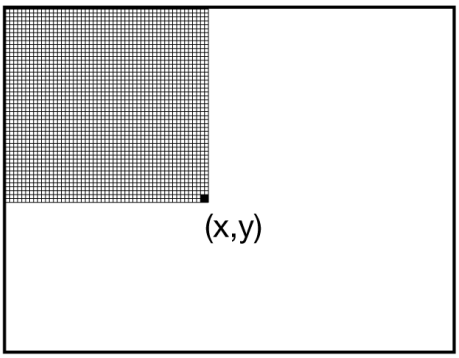

Figura 3.4: La imagen auxiliar(SAT) en la posición (x, y) se corresponde con la suma de los pixeles arriba y a la izquierda de (x, y) inclusive.

Lienhart propone una forma de calcular esta imagen mediante un barrido de izquierda a derecha y de arriba hacia abajo, utilizando la siguiente ecuación:

$$SAT(x,y) = SAT(x,y-1) + SAT(x-1,y) + I(x,y) - SAT(x-1,y-1) \tag{3.2}$$

siendo:

$$SAT(-1,y) = SAT(x,-1) = SAT(-1,-1) = 0 \tag{3.3}$$

Luego, la sumatoria de los valores de los pixeles para cualquier rectángulo $r = (x, y, w, h)$, con (x, y) la posición del extremo izquierdo superior y w, h el ancho y alto del rectángulo, puede ser calculada mediante la función *RecSum* definida como:

$$RecSum(r) = SAT(x-1,y-1) + SAT(x+w-1,y+h-1) - SAT(x-1,y+h-1) - SAT(x+w-1,y-1) \tag{3.4}$$

Es decir, una vez obtenida la imagen SAT, las diferencias pueden computarse en tiempo constante. Esto es posible ya que la suma de los pixeles de cualquier rectángulo en la imagen puede calcularse mediante cuatro operaciones básicas 3.5).

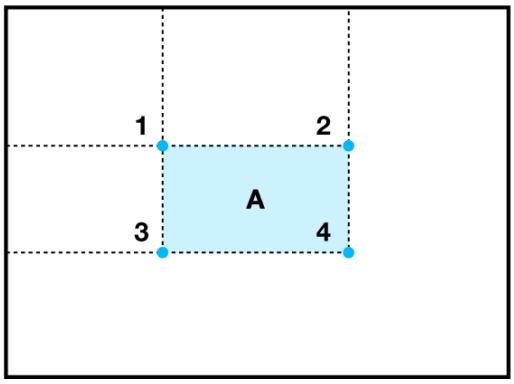

Figura 3.5: El valor del rectángulo A puede ser calculado mediante $SAT(4) + SAT(1)$ - $(SAT(2) + SAT(3))$

Por otra parte para los rectángulos orientados 45° definió otra imagen auxiliar *RSAT (Rotated Summed Area Table)* cuya definición es la siguiente:

$$RSAT(x, y) = \sum_{j \leq y, j \leq y - |x - i|} I(i, j) \tag{3.5}$$

La misma también puede ser calculada mediante una barrida sobre la imagen original de izquierda a derecha y de arriba hacia abajo:

$$RSAT(x, y) = RSAT(x - 1, y - 1) + RSAT(x + 1, y - 1) - RSAT(x, y - 2) + I(x, y) + I(x, y - 1) \tag{3.6}$$

siendo:

$$RSAT(-1, y) = RSAT(x, -1) = RSAT(x, -2) = RSAT(-1, -1) = RSAT(-1, -2) = 0 \tag{3.7}$$

De este modo, la suma del valor de los rectángulo de la forma $r = (x, y, w, h)$ orientados 45° puede calcularse como se muestra en la figura 3.6 de la siguiente manera:

$$RecSum(r) = RSAT(x-h+w, y+w+h-1)+RSAT(x, y-1)-RSAT(x-h, y+h-1)-RSAT(x+w, y+w-1) \tag{3.8}$$

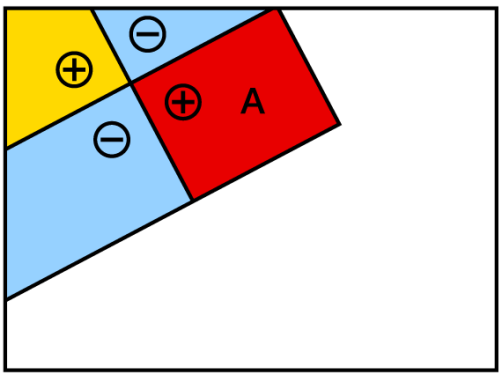

Figura 3.6: Esquema de cálculo de la suma del valor de los pixeles para el rectángulo A orientado 45°.

3.1.2. Clasificación utilizando haar-like features

Una ventana es un rectángulo de altura h y ancho w contenido en el tamaño total de una imagen. Las ventanas en una imagen comprenden rectángulos a todas las escalas posibles dentro de la misma. Para poder

localizar rostros en una imagen es necesario determinar si cada posible ventana de la misma se corresponde o no con un rostro. Para lograrlo, es necesario implementar un algoritmo de clasificación para generar una correcta asociación de una ventana a un rostro.

El cálculo de las *haar-like features* en todas las ventanas de una imagen dificulta mantener el método funcionando en tiempo real. Con lo cual, para realizar de manera correcta y computacionalmente eficiente la tarea de detección de rostros en ventanas, es necesario encontrar el subconjunto finito de *haar-like features* que mejor caracterice al objeto que se desea encontrar, en este caso un rostro. En sus trabajo Viola-Jones y Lienhart utilizaron el método *Boosting* [18] para lograr este objetivo (ver Algoritmo 1).

Los algoritmos de *Boosting* seleccionan de un conjunto de clasificadores simples (algoritmos de clasificación que funcionan ligeramente mejor que clasificar al azar) aquellos que arrojan mejores resultados para combinarlos en un único clasificador fuerte. Esta elección se efectúa mediante una votación sobre los pesos asignados a cada clasificador. El desafío es asociar pesos grandes a buenas funciones de clasificación y pesos bajos a funciones pobres.

Viola-Jones y Lienhart asocian una *haar-like feature* a un clasificador simple. Luego mediante *Boosting* obtienen el subconjunto de estas *features* que mejor caracterizan un rostro. Para encontrar dicho subconjunto, cada clasificador simple se diseña de tal manera que permite seleccionar las *features* que mejor separan los ejemplos a clasificar positivos de los negativos, en este caso rostros contra imágenes que no lo son.

Algoritmo 1 Algoritmo de Boosting genérico

Sea $(x_1, y_1) \dots (x_n, y_n)$ un conjunto de n pares en donde
x_i se corresponde con una ventana aplicada a la imagen
e $y_i \in \{-1, +1\}$ indica si la muestra es (+) o no (-) un rostro
Se definen un conjunto de pesos w_i $\forall 1 \leq i \leq n$
Se generan T clasificadores
for $t = 1 \dots T$ **do**
    Se entrena el clasificador f_t a partir de los pesos y los datos de entrenamiento.
    Se computa el error e_t asociado a la clasificación
    Se actualizan los pesos w_i $\forall 1 \leq i \leq n$ considerando e_t
end for
El resultado del clasificador final se forma a partir
de los T clasificadores generados y una función g basada en el error.
return $F = \text{Signo}(\sum_{t=1}^{T} g(e_t) f_t)$

Para seleccionar este conjunto de características, Viola y Jones utilizan el método *AdaBoost*. Por su parte, Lienhart mejora estos resultados usando el algoritmo *Gentle AdaBoost*.

La combinación de pocas *haar-like features* reduce considerablemente el tiempo de cómputo. Sin embargo la aplicación de todas ellas a las ventanas de una imagen puede continuar siendo inaceptable. Para corregir esto, los autores emplean un modelo de clasificación en cascada como puede verse en la figura 3.7. Este modelo en cascada se utiliza con el objetivo de descartar la mayor cantidad de ventanas en las primeras etapas para refinar la detección sobre el final de las mismas. En cada etapa se busca mantener una tasa de acierto del 100 % descartando una cantidad considerable de ventanas que no son consideradas rostro, de modo tal que al final del procesamiento la misma tienda a 0.

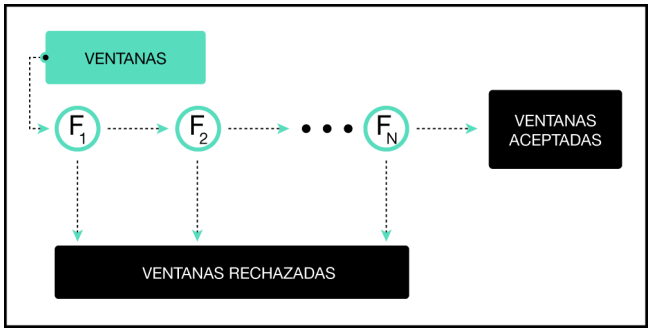

Figura 3.7: Esquema de clasificación en cascada.

Viola y Jones utilizan un modelo de 38 etapas mientras que Lienhart utiliza 20 para su modelo. Los resultados obtenidos por Lienhart en [16] demuestran que su método mejora en un 10 % la tasa de falsos positivos bajo la misma tasa de verdaderos positivos con lo cual en la tesis se utiliza su implementación. La misma puede encontrarse en [19].

3.2. Seguimiento de rostros en capturas de video

Dado que nuestro sistema se encarga de procesar una captura de video buscando gestos en un rostro, necesitamos conocer la ubicación del rostro en cada uno de los *frames* que componen la misma. Una primera aproximación para esta tarea es aplicar el detector de rostros definido en la sección 3.1 a cada uno de los *frames* de la captura. Si bien este enfoque se mantiene dentro de la restricción de funcionamiento en tiempo real, deja poco margen para las restantes tareas que deben ejecutarse *frame* a *frame*. Con lo cual optimizar este proceso es un requerimiento indispensable que nos permite ampliar los tiempos disponibles para las demás tareas.

En las secciones 3.2.1 y 3.2.2 se detallan las mejoras analizadas e implementadas con el objetivo de reducir los tiempos del proceso de seguimiento de un rostro en una captura de video.

3.2.1. Coherencia Temporal

En [20] (Castrillón, Déniz, Guerra y Hernández) se propusieron una serie de mejoras para optimizar la eficacia y la eficiencia de la detección de rostros. La mejora que se implementa en nuestro trabajo fue la integración de la *coherencia temporal*. Este concepto se refiere a que una vez detectada una cara en un *frame* es esperable que en el *frame* siguiente el rostro esté ubicado en un área cercana a la anterior y con una tamaño similar (ver figura 3.8). Para esto se define un área de búsqueda expandiendo el área de detección del rostro localizado en el *frame* anterior. Con esta optimización se logra mejorar los tiempos manteniendo la misma tasa de detección. La proporción de expansión del área de búsqueda se configura de acuerdo a los experimentos realizados en este trabajo, pero podría variar en otras implementaciones. La mejora de los tiempos radica en el hecho de que la búsqueda de un rostro no se realiza sobre la imagen completa, sino en un área de búsqueda de menor tamaño cuya ubicación depende de la detección realizada en el *frame* anterior. Por otro lado como el algoritmo de detección no es modificado en su funcionamiento la tasa de aciertos sigue manteniéndose.

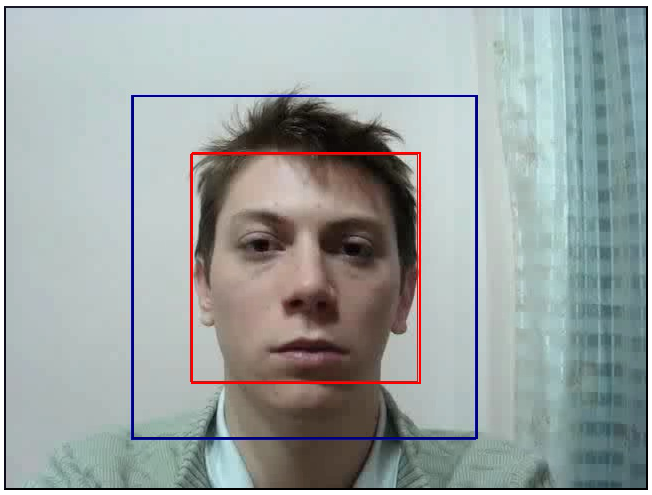

Figura 3.8: Area de búsqueda con una ampliación del 1.5 en el tamaño de la detección. El rectángulo rojo se corresponde con la detección obtenida, mientras que el rectángulo azul se corresponde con el área de búsqueda para el *frame* siguiente.

Las demás mejoras propuestas en [20] fueron implementadas pero luego se desestimaron ya que no producían grandes avances dadas las características de nuestro contexto. Estas mejoras eran:

- Detección de hombros y cabeza (definido como contexto local) para determinar la posición del rostro.
- Detección de los ojos y utilización del conocimiento que se tiene de un rostro para determinar la posición de la cara.

3.2.2. Seguimiento de rostros por color: *Camshift*

Adicionalmente se estudió la posibilidad de utilizar un algoritmo basado en color para llevar a cabo el seguimiento de un rostro a lo largo de una secuencia de *frames*. Para ello se estudió el comportamiento del algoritmo *camshift (Continously Adaptive Mean Shift)* propuesto por Bradski en [21]. Esta implementación luego fue desestimada por las razones que se detallarán al final de la sección.

La ventaja que tiene este algoritmo es la velocidad en el seguimiento, ya que está basado únicamente en cálculos de histogramas y comparaciones mediante color. La idea fundamental de éste método es poder hacer el seguimiento de un área a evaluar posteriormente.

Descripción del método

Utilizamos fuertemente la descomposición HSV (ver sección 2.2.2) de cada uno de los *frames* de la secuencia para realizar el seguimiento por color de una cara. En particular, se utiliza el canal *hue* (tonalidad), ya que el color de la piel de cualquier raza puede ser identificado unívocamente por dicho canal [21].

El primer paso del algoritmo es la selección de una ventana de búsqueda inicial a la cual se le hace el seguimiento en el resto de los *frames* de la secuencia. En nuestro caso esta ventana inicial está dada por el rostro detectado en el primero de los *frames* de una captura.

Posteriormente, se calcula un histograma de color sobre la ventana de búsqueda en la imagen trasladada al espacio de color HSV sobre el canal *hue*[1]. Un histograma, es una representación que nos permite dado un conjunto de muestras con ciertas etiquetas, y una cantidad finita de esas etiquetas, indicar la proporción del conjunto asignada en cada una de ellas (ver figura 3.9). En el caso de un histograma basado en color, considerando sólo dos colores, dados todos los valores de color de los pixeles en una imagen en blanco y negro, el resultado del cálculo del histograma es la cantidad de pixeles de color negro y la cantidad de color blanco.

Formalmente un histograma de color asociado al canal *hue* puede definirse como un vector $H \in \mathbb{Z}^k$, en donde k es el rango de valores o matices en la escala del canal *hue* de la siguiente manera:

$$H_i = \sum_{x,y \in R} (\delta(I(x,y), i)) \quad \forall 1 \leq i \leq k \tag{3.9}$$

En donde R se corresponde con el área de búsqueda e $I(x,y)$ es el valor de la posición (x,y) en el canal *hue* de la imagen original. La función $\delta : \mathbb{Z}^2 \to \mathbb{Z}$ devuelve 1 si dos números son iguales y 0 en caso contrario.

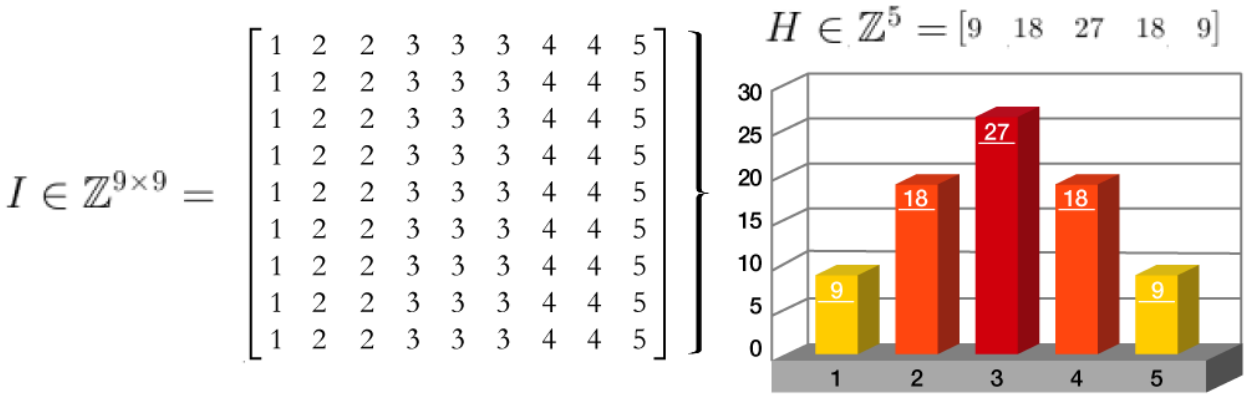

Figura 3.9: Cálculo de un historama a partir de una imagen $I \in \mathbb{Z}^{9\times 9}$ en una escala de 5 valores.

A partir del histograma de cada *frame* de la secuencia se calcula una imagen de retroproyección asociada a cada uno de ellos (ver figura 3.10). Esta operación consiste en generar una imagen en escala de grises de la siguiente forma: se obtiene el valor en el histograma, asociado a la intensidad de cada pixel en el *frame* a procesar para, posteriormente, convertirlo al nivel de gris de la imagen de retroproyección. De este modo, el valor de cada pixel en esta imagen identificará la probabilidad de que dicho pixel en el *frame* procesado pertenezca al objeto identificado en el área de búsqueda.

[1] Para transladar una imagen de un espacio de color RGB a HSV se utiliza una operación sobre canales similar a la explicada en 2.2.3 para la conversión a escala de grises.

Formalmente, considerando N la cantidad de pixeles en el área de búsqueda e I la imagen a procesar, la imagen de retroproyección P queda definida como:

$$P(x,y) = \frac{H_{[I(x,y)]} * 255}{N} \tag{3.10}$$

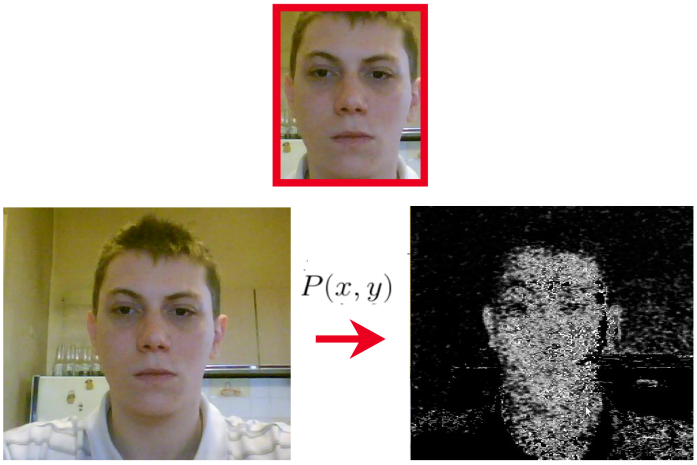

Figura 3.10: Imagen de retroproyección sobre una imagen utilizando como área de búsqueda inicial la detección de un rostro.

Mediante la utilización del algoritmo *Mean Shift* definido en [22] se obtiene una nueva área de búsqueda sobre la imagen de retroproyección para el siguiente *frame*. Este algoritmo es iterativo y consta de la siguiente secuencia de pasos:

1. Se elige un área de búsqueda. En este caso se toma el área de búsqueda actual.
2. Se calcula el centro de masa (x_c, y_c) sobre la imagen de retroproyección dentro del área de búsqueda. Para ello se utilizan los momentos geométricos[2] de orden 0.
3. Se centra la ventana de búsqueda en el punto (x_c, y_c).
4. Se repiten los pasos 2 y 3 hasta que el centro de la ventana se desplace menos de un determinado umbral.

Una vez obtenida el área de búsqueda, el proceso (ver figura 3.11) se repite *frame* a *frame* durante toda la captura de video para realizar un seguimiento continuo del rostro.

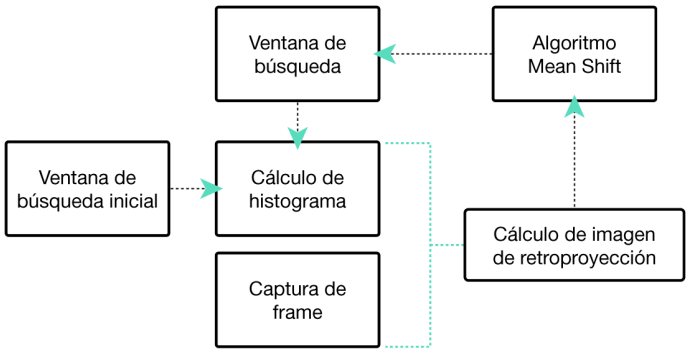

Figura 3.11: Esquema de cálculo del algoritmo *Camshift*.

[2]Una explicación más detallada sobre el concepto de momentos geométricos (aplicados a otro contexto) será realizada en la sección 5.3.2.

Si bien este algoritmo cumple con el objetivo de funcionar en tiempo real, lo distintos tipos de iluminación ambiental varían el comportamiento del mismo lo que lo hace inestable, por ejemplo en el caso de que el color del fondo sea similar al del rostro. Es por ello, que decidimos desestimar este método y utilizar la coherencia temporal para el seguimiento de rostros.

3.3. Conclusiones del capítulo

Se consideraron las variantes de Viola-Jones y Lienhart para la detección de rostros. Si bien ambas implementaciones son confiables, decidimos utilizar la de Lienhart por considerarlo una mejora por sobre la de sus predecesores y funcionar también en tiempo real.

Posteriormente se evaluaron diversas alternativas para el seguimiento de los rostros en la secuencia de *frames* que componen una captura de video. Se descartó el método de seguimiento por color por ser inestable a cambios de iluminación. Finalmente se utilizó el seguimiento de rostro mediante la detección del mismo *frame* a *frame* utilizando la mejora basada en coherencia temporal por mantener una buena tasa de reconocimiento y ser computacionalmente eficiente. Las demás mejoras como la detección de hombros u ojos para la localización del rostro también fueron desestimadas por no ser consideradas mejoras sobre el método elegido y agregar complejidad computacional.

Capítulo 4

Representación de movimiento

En el capítulo 3 se analizó el problema de detección y seguimiento de una cara en una secuencia de *frames*. Solucionado este problema nos enfocamos en estudiar la forma de capturar y describir los movimientos que se realizan en el área detectada, descartando cualquier otro movimiento que no esté contenido en la misma. Para ello buscamos una forma de representar los movimientos que se van produciendo a lo largo de una secuencia de *frames*. Esta representación debe permitirnos distinguir en qué zonas se detecta movimiento y cómo el mismo evoluciona en el transcurso de tiempo.

Para esta tarea se utiliza entonces el método de *Templates Temporales* basado en las imágenes *Motion History Image (MHI)* y *Motion Energy Image (MEI)* y algunas derivaciones y mejoras del mismo.

En la sección 4.1 del presente capítulo definimos la noción de *Templates Temporales* y sus diferentes aplicaciones. En la sección 4.2 se explican los conceptos teóricos relacionados a *Motion History Image (MHI)* y *Motion Energy Image (MEI)*. En la sección 4.3 se describe el método para el cálculo del MHI. En la sección 4.4 se analizan cuestiones particulares relacionadas al cálculo de la MHI y se presenta una especificación del algoritmo implementado. En la sección 4.5 se analizan diferentes variantes y mejoras realizadas al algoritmo, para adaptarlo al contexto de reconocimiento de gestos faciales. Por último, en la sección 4.6 se extraen algunas conclusiones del capítulo.

4.1. Representación de movimiento usando Templates Temporales

Definimos un *Template Temporal* como un vector-imagen en donde el valor en cada posición del mismo es una función de las propiedades del movimiento en la correspondiente ubicación en una secuencia de imágenes. A continuación se definen formalmente estos conceptos.

Def. 4.1. En lo subsiguiente, vamos a notar $\mathbf{I} \in [\mathbb{Z}^{m\times n}]_t$ a una secuencia $\mathbf{I} = (I_1, \ldots, I_t)$ de t imágenes en valores enteros de tamaño $m \times n$, es decir, $(\forall j : 1 \leq j \leq t)\, I_j \in \mathbb{Z}^{m\times n}$. Dado $\mathbf{I} \in [\mathbb{Z}^{m\times n}]_t$ y $j \leq t$ notamos como $\mathbf{I}_j$ a la subsecuencia de las primeras j imágenes de $\mathbf{I}$, es decir, $\mathbf{I}_j = (I_1, \ldots, I_j)$.

Def. 4.2. Sea $\mathbb{B}^{m\times n}$ el conjunto de matrices booleanas de $m \times n$. Definimos el operador $\phi : [\mathbb{Z}^{m\times n}]_t \longrightarrow \mathbb{B}^{m\times n}$ que dada una secuencia de imágenes $\mathbf{I} \in [\mathbb{Z}^{m\times n}]_t$ le asigna una matriz booleana $\psi \in \mathbb{B}^{m\times n}$ que indica los pixeles en que se detectó movimiento en la imagen I_t. La matriz ψ cumple que:

$$\psi(x, y) = true \qquad \text{si se detectó movimiento en la imagen } I_t \text{ en la posición } (x, y)$$

$$\psi(x, y) = false \qquad \text{si no se detectó movimiento en la imagen } I_t \text{ en la posición } (x, y)$$

Def. 4.3. A partir del operador ϕ y una secuencia de imágenes $\mathbf{I} \in [\mathbb{Z}^{m\times n}]_t$ definimos la secuencia de imágenes booleanas $\mathbf{\Psi} \in [\mathbb{B}^{m\times n}]_t$ tal que $\mathbf{\Psi} = (\psi_1, \ldots, \psi_t)$ donde

$$\psi_i = \phi(\mathbf{I}_i) \qquad \forall i, 1 \leq i \leq t \tag{4.1}$$

Notaremos $\mathbf{\Psi}(x, y) \in \mathbb{B}^t$ a la secuencia de valores booleanos obtenidos por el operador ϕ en la posición (x, y) a lo largo de las imágenes de la secuencia $\mathbf{I}$. Formalmente,

$$\mathbf{\Psi}(x, y) = (\psi_1(x, y), \ldots, \psi_t(x, y)) \qquad \forall (x, y), 1 \leq x \leq m, 1 \leq y \leq n \tag{4.2}$$

$\mathbf{\Psi}(x, y)$ indica para cada momento de la secuencia $\mathbf{I}$ si se detectó o no movimiento en la posición (x, y).

Def. 4.4. Sea $\mathbf{\Psi} \in [\mathbb{B}^{m\times n}]_t$ tal que $\mathbf{\Psi} = (\psi_1, \ldots, \psi_t)$ una secuencia de matrices de detección de movimiento (ver Def. 4.3). Sea $f : \mathbb{B}^t \longrightarrow \mathbb{V}$, con $\mathbb{V}$ un conjunto de valores. Definimos el *template temporal* $T \in \mathbb{V}^{m\times n}$ para la secuencia de matrices $\mathbf{\Psi}$ como:

$$T(x, y) = f(\mathbf{\Psi}(x, y)) \qquad \forall (x, y), 1 \leq x \leq m, 1 \leq y \leq n \tag{4.3}$$

Como se ve, la función f extrae una propiedad del movimiento, en una posición determinada, tomando en cuenta la sucesión de detecciones de movimiento que se registra en esa posición.

En la figura 4.1 se puede observar una forma de expresar gráficamente la definición de la función ϕ. En este figura $\mathbf{I}$ representa una secuencia de imágenes, ϕ es la función de detección de movimiento (según lo definido en Def. 4.2) y $\mathbf{\Psi}$ es la secuencia de matrices de movimientos detectados (ver Def. 4.3)

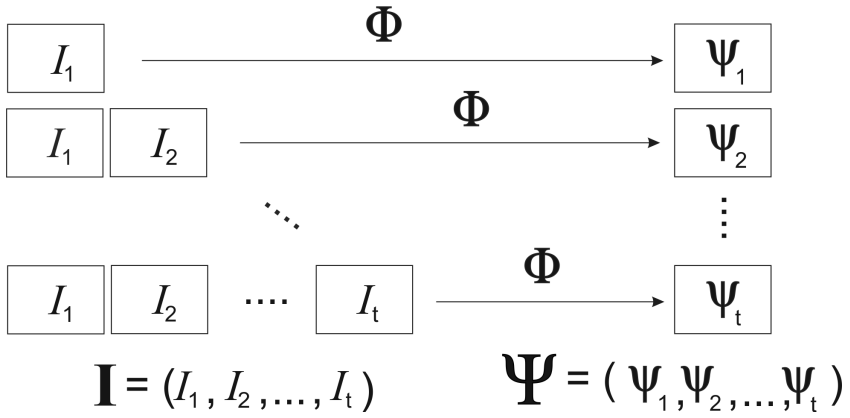

Figura 4.1: La función ϕ expresada gráficamente de acuerdo a la definición 4.3

En la figura 4.2 se expresa gráficamente la definición de *template temporal*. En esta figura, $\mathbf{\Psi}$ es la secuencia de matrices de movimientos detectados (ver Def. 4.3), f es la función que extrae las propiedades del movimiento y T es el *template temporal* obtenido.

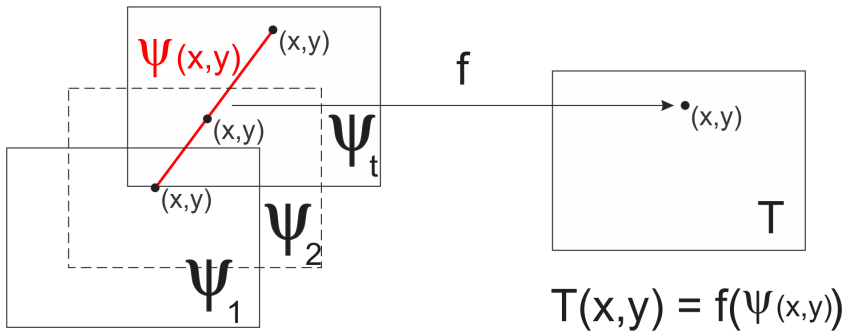

Figura 4.2: Definición gráfica de un *template temporal* de acuerdo a la definición 4.4

La gran ventaja de representar movimiento mediante *Templates Temporales* es que podemos sintetizar el movimiento que se produce en una secuencia de imágenes en una única imagen que contiene toda la información relevante que necesitamos.

La representación de movimiento mediante *Templates Temporales* además nos permite:

- Observar y analizar fácilmente como se desarrolla el movimiento.
- Aplicar filtros al *template temporal* que nos permitan descartar información que no sea de interés (ruido, errores de captura, etc.)
- Trabajar sobre una imagen en lugar de sobre una secuencia de imágenes.

Utilizamos un *template temporal* para reconocer el movimiento que se está realizando, buscando en una base de templates ya calculados aquel que mejor lo adapte o que más se le parezca.

4.2. Motion History Image y Motion Energy Image

En el presente trabajo, para describir el movimiento que se produce en una secuencia de imágenes se analizan dos aspectos: *dónde* se produce el movimiento y *cómo* se produce el mismo. En [11] Bobick y Davis describen dos estructuras que encapsulan estos aspectos. La primera de ellas es una imagen de energía de movimiento (*Motion Energy Image, MEI*), que indica en cuáles pixeles se detectó movimiento a lo largo de la secuencia de *frames*, describiendo así dónde ocurre el movimiento. La segunda es la imagen de historia de movimiento (*Motion History Image, MHI*) que contiene en cada posición el último momento en que se detectó movimiento y describe cómo éste fue realizado.

Ambas estructuras pueden definirse utilizando el concepto de *Templates Temporales* de la siguiente forma:

Def. 4.5. Sea $\mathbf{\Psi} \in [\mathbb{B}^{m\times n}]_t / \mathbf{\Psi} = (\psi_1, \ldots, \psi_t)$ una secuencia de matrices de detección de movimiento. Llamaremos *Motion Energy Image* al *template temporal* $E \in \mathbb{B}^{m\times n}$ definido por la función $f : \mathbb{B}^t \longrightarrow \mathbb{B}$

$$f(\mathbf{B}) = \bigvee(\mathbf{B}) = \bigvee_{i=1}^{t} b_i \quad \text{con } \mathbf{B} = (b_1, \ldots, b_t) \tag{4.4}$$

donde $\bigvee$ se entiende como la disyunción lógica de valores booleanos.

Luego, por las definiciones 4.3 y4.4 vale que

$$E(x,y) = f(\mathbf{\Psi}(x,y)) = \bigvee_{i=1}^{t} \psi_i(x,y) \qquad \forall (x,y), 1 \leq x \leq m, 1 \leq y \leq n \tag{4.5}$$

Def. 4.6. Sea $\mathbf{\Psi} \in [\mathbb{B}^{m\times n}]_t / \mathbf{\Psi} = (\psi_1, \ldots, \psi_t)$ una secuencia de matrices de detección de movimiento. Llamamos *Motion History Image* al *template temporal* $M \in \mathbb{Z}^{m\times n}$ definido por las función $f : \mathbb{B}^t \longrightarrow \mathbb{Z}$

$$f(\mathbf{B}) = \begin{cases} 0 & \text{si } \forall i / 1 \leq i \leq t, b_i = false \\ \text{máx}\{i : 1 \leq i \leq t, b_i = true\} & \text{sino} \end{cases} \qquad \text{con } \mathbf{B} = (b_1, \ldots, b_t) \tag{4.6}$$

Luego, por la definición 4.4 vale que

$$M(x,y) = \begin{cases} 0 & \text{si } \forall i / 1 \leq i \leq t, \psi_i(x,y) = false \\ \max\limits_{i}\{i : 1 \leq i \leq t, \psi_i(x,y) = true\} & \text{sino} \end{cases} \tag{4.7}$$

$\forall (x,y), 1 \leq x \leq m, 1 \leq y \leq n$

La MHI M calculada para una secuencia de matrices de movimiento, nos da como resultado una imagen en la cual las posiciones en que se haya detectado movimiento en forma más reciente tienen un valor mayor.

4.2.1. Trabajos Relacionados

La noción de *Motion History Image (MHI)* fue descripta en forma inicial por Bobick y Davis en [11] con el objetivo de reconocer movimientos de personas. En el trabajo mencionado se cuenta con capturas de personas realizando una actividad (ejemplo: sentarse, agacharse, mover los brazos) y el objetivo es distinguir cuál actividad se está ejecutando asociándola con un *template* que representa su ejecución. El *input* para este tipo de algoritmos es un video de una persona ejecutando un movimiento particular. Asociar dicho movimiento con alguno de los posibles movimientos conocidos por el autor se reduce a comparar, mediante algún método, la MHI calculada contra las MHIs propias de cada movimiento conocido.

En [23] Hsieh, Liou y Cheng propusieron el método MHI como *template* adicional a otras técnicas utilizadas en el reconocimiento de patrones en una mano para generar la integración con el puntero (comúnmente manejado por el mouse) en una computadora.

En [24] Valstar, Pantic y Patras utilizaron la noción de MHI para la detección de expresiones en un rostro mediante la utilización de *Action Units* (señales faciales atómicas) descriptas en [1] y considerando algunas alternativas para los problemas de oclusión del movimiento (este problema se analizará en la sección 4.5).

En [13] y [25] (Ahad, Ogata, Kim, Tan, Ishikawa) se abordaron los problemas asociados con la oclusión del movimiento con mayor robustez y se propusieron varios métodos para mitigar dichos inconvenientes, entre ellos, la técnica de MHI direccionales (DMHI) que consiste en calcular la MHI para cada una de cuatro posibles

direcciones básicas del movimiento, arriba, abajo, izquierda y derecha. Para ello se consideró el gradiente de la MHI con el objeto de estimar dichas orientaciones. Esto genera la noción de dirección para un movimiento en particular. Además, se propusieron algunos algoritmos útiles para el trabajo con dichas MHI's, como por ejemplo el cálculo de histogramas de movimiento.

Finalmente en [26] y [27] (Ahad, Ogata, Kim, Tan, Ishikawa) se hizo un recorrido sobre la técnica de MHI y la mayoría de sus variantes utilizadas por los distintos autores.

En nuestro trabajo utilizamos muchas de las nociones consideradas en las publicaciones comentadas anteriormente. Realizamos una adaptación del método de *Motion History Image*, introduciéndole algunas mejoras para nuestro objetivo particular que es la detección y reconocimiento de gestos de Truco. Nos apoyamos sobre la suposición de que los resultados son similares a los obtenidos por Bobick y Davis, teniendo en cuenta que contamos con la captura de un rostro y deseamos obtener los movimientos que en él se producen.

4.3. Descripción del método

En esta sección describimos un método para calcular la MEI E y la MHI M de una secuencia de imágenes. Para este propósito reformulamos las definiciones 4.5 y 4.6 expresándonlas en forma recursiva. Las definiciones recursivas nos permiten extraer fácilmente un método que, tomando secuencialmente un conjunto de imágenes, les calcule su matriz de movimiento y con estas matrices obtenga los *Templates Temporales parciales*.

En primer lugar, basándonos en las definiciones 4.5 y 4.6 podemos observar que la imagen E puede calcularse a partir de la imagen M de la siguiente forma:

$$E(x,y) = \begin{cases} false & \text{si } M(x,y) = 0 \\ true & \text{si } M(x,y) > 0 \end{cases} \qquad \forall (x,y), 1 \leq x \leq m, 1 \leq y \leq n \tag{4.8}$$

Con este resultado se demuestra que para calcular ambas matrices, sólo debemos calcular la matriz M y luego obtener E marcando con el valor *true* las posiciones de M que tengan un valor mayor a 0 y las demás con $false$. Luego de aquí en adelante sólo nos ocupamos del cálculo de la matriz M.

A continuación se obtiene una definición recursiva de la matriz M derivada de la definición 4.6.

Prop. 4.7. Sea $\mathbf{\Psi} \in [\mathbb{B}^{m \times n}]_t$ tal que $\mathbf{\Psi} = (\psi_1, \ldots, \psi_t)$, una secuencia de matrices de detección de movimiento (ver Def. 4.3). La *Motion History Image* $M \in \mathbb{Z}^{m \times n}$ asociada a la secuencia $\mathbf{\Psi}$, puede calcularse como $M = M_t$, donde $M_i \in \mathbb{Z}^{m \times n}$ se define recursivamente como:

- Caso base $i = 1$:

$$M_1(x,y) = 0 \qquad \forall (x,y), 1 \leq x \leq m, 1 \leq y \leq n \tag{4.9}$$

- Caso recursivo $1 < i \leq t$:

$$M_i(x,y) = \begin{cases} i & \text{si } \psi_i(x,y) = true \\ M_{i-1}(x,y) & \text{sino} \end{cases} \qquad \forall (x,y), 1 \leq x \leq m, 1 \leq y \leq n \tag{4.10}$$

Un problema que nos surge, es que tanto la definición 4.6 como la propiedad 4.7, toman en cuenta todas las matrices que componen la secuencia $\mathbf{\Psi}$. Es posible que si la secuencia es larga, podamos estar incluyendo movimientos que no se consideran recientes. Para solucionar este problema incluimos el parámetro α que nos indica cuántos de los últimos *frames* son considerados como recientes y tenidos en cuenta para calcular los *Templates Temporales*. Notamos a estas nuevos *Templates Temporales* como E^α y M^α y los definimos de la siguiente forma:

Def. 4.8. Sea $\mathbf{\Psi} \in [\mathbb{B}^{m \times n}]_t / \mathbf{\Psi} = (\psi_1, \ldots, \psi_t)$ una secuencia de matrices de detección de movimiento. Dado $\alpha \in \mathbb{N}$, llamaremos *Motion Energy Image* al *template temporal* $E^\alpha \in \mathbb{B}^{m \times n}$ definido por las función $f^\alpha : \mathbb{B}^t \longrightarrow \mathbb{B}$

$$f^\alpha(\mathbf{B}) = \bigvee^{\alpha}(\mathbf{B}) = \bigvee_{i=max(1,t-\alpha)}^{t} b_i \quad \text{con } \mathbf{B} = (b_1, \ldots, b_t) \tag{4.11}$$

donde $\bigvee$ se entiende como la disyunción lógica de valores booleanos.

Luego, por la definición 4.4 vale que

$$E^\alpha(x,y) = f^\alpha(\mathbf{\Psi}(x,y)) = \bigvee_{i=max(1,t-\alpha)}^{t} \psi_i \qquad \forall (x,y), 1 \leq x \leq m, 1 \leq y \leq n \tag{4.12}$$

Def. 4.9. Sea $\mathbf{\Psi} \in [\mathbb{B}^{m\times n}]_t / \mathbf{\Psi} = (\psi_1, \ldots, \psi_t)$ una secuencia de matrices de detección de movimiento. Dado $\alpha \in \mathbb{N}$, llamaremos *Motion History Image* al *template temporal* $M^\alpha \in \mathbb{Z}^{m\times n}$ definido por las función $f^\alpha : \mathbb{B}^t \longrightarrow \mathbb{Z}$

$$f^\alpha(\mathbf{B}) = \begin{cases} 0 & \text{si } \forall i / max(1, t-\alpha) \leq i \leq t, b_i = false \\ \max\limits_i \{i : \max(1, t-\alpha) \leq i \leq t, b_i = true\} & \text{sino} \end{cases} \tag{4.13}$$

con $\mathbf{B} = (b_1, \ldots, b_t)$.

Luego, por la definición 4.4 vale que

$$M^\alpha(x,y) = \begin{cases} 0 & \text{si } \forall i / 1 \leq i \leq t, \psi_i(x,y) = false \\ \max\limits_i \{i : max(1, t-\alpha) \leq i \leq t, \psi_i(x,y) = true\} & \text{sino} \end{cases} \tag{4.14}$$

$\forall (x,y), 1 \leq x \leq m, 1 \leq y \leq n$

Las definiciones 4.8 y 4.9 nos dan la pauta que a partir de que t supera el valor de α se empiezan a dejar de tener en cuenta los primeros *frames* de la secuencia para el cálculo de ambos *Templates Temporales*.

Notar que la ecuación (4.8) sigue siendo válida para esta definición. En este caso,

$$E^\alpha(x,y) = \begin{cases} false & \text{si } M^\alpha(x,y) = 0 \\ true & \text{si } M^\alpha(x,y) > 0 \end{cases} \qquad \forall (x,y), 1 \leq x \leq m, 1 \leq y \leq n \tag{4.15}$$

Prop. 4.10. La MHI M^α puede ser calculada a partir de la MHI M, de la siguiente forma:

$$M^\alpha(x,y) = \begin{cases} 0 & \text{si } M(x,y) \leq t-\alpha \\ M(x,y) & \text{si } M(x,y) > t-\alpha \end{cases} \qquad \forall (x,y), 1 \leq x \leq m, 1 \leq y \leq n \tag{4.16}$$

Calculando recursivamente M de la forma definida en Prop. 4.7, podemos calcular M^α y a partir de esta última calcular E^α.

Podemos observar que la MHI M^α tendrá valores cada vez más grandes a medida que transcurre el tiempo. Esta situación puede evitarse, una vez definido el parámetro α, haciendo que la imagen M^α sólo contenga valores entre 0 y α (donde los pixeles que contengan valores más cercanos a α serán aquellos en los que se detectó un movimiento más reciente). Para eso redefiniremos la matriz M^α como $M^\alpha = M^\alpha_t$, donde $M^\alpha_i \in \mathbb{Z}^{m\times n}$ se calcula recursivamente como:

$$M^\alpha_i(x,y) = \begin{cases} min(i, \alpha) & \text{si } \psi_i(x,y) = true \\ M^\alpha_{i-1}(x,y) & \text{si } \neg\psi_i(x,y) \text{ y } i < \alpha \\ max(0, M^\alpha_{i-1}(x,y) - 1) & \text{si } \neg\psi_i(x,y) \text{ y } i \geq \alpha \end{cases} \qquad 1 \leq x \leq m, 1 \leq y \leq n, 1 < i \leq t \tag{4.17}$$

Dónde $\neg\psi_i(x,y)$ es la negación lógica del valor booleano $\psi_i(x,y)$.

Se observa que para valores de $i < \alpha$ la definición se mantiene. Para valores de $i \geq \alpha$ se van decrementando en 1 las posiciones en las que no se detectó movimiento (siempre y cuando tengan un valor mayor a 0) y seteando el valor α en aquellas en las que sí se detectó movimiento. De lo anterior se deduce que los valores de M^α se encuentran entre 0 y α.

Si bien la matriz M^α calculada de esta manera no tiene los mismos valores que la matriz con el mismo nombre definida en Def. 4.9, pero la información que contiene es análoga, es decir, M^α sigue cumpliendo que las posiciones en que se haya detectado un movimiento más reciente tienen un valor más alto.

Notar que la ecuación (4.15) sigue valiendo para esta definición de M^α.

Por último nos surge un problema relacionado al hecho de que dos MHIs generadas a partir de secuencias de imágenes de diferente longitud no son comparables. En efecto, si tenemos dos secuencias de imágenes $\mathbf{I} \in [\mathbb{Z}^{m\times n}]_t$ y $\mathbf{J} \in [\mathbb{Z}^{m\times n}]_s$, tal que $t > s$, podemos observar que la MHI generada a partir de los movimientos detectados en $\mathbf{I}$ podrá contener valores entre 0 y t, mientras que la generada por los movimientos de $\mathbf{J}$ podrá tener valores entre 0 y s. De esta forma, ambas MHI no son comparables, por lo tanto, es necesario normalizarlas para que las características extraídas de cada uno sean comparables.

Def. 4.11. Sea $\mathbf{\Psi} \in [\mathbb{B}^{m\times n}]_t / \mathbf{\Psi} = (\psi_1, \ldots, \psi_t)$ una secuencia de matrices de detección de movimiento. Sea M^α la MHI calculada a partir de la secuencia $\mathbf{\Psi}$ tomando en cuenta la ventana de tiempo $\alpha \in \mathbb{N}$. Definimos la

Motion History Image Normalizada (NMHI) $NM^\alpha \in \mathbb{Z}^{n\times m}$ de M^α como

$$NM^\alpha(x,y) = \begin{cases} 0 & \text{si } t = 0 \\ M^\alpha(x,y)/t & \text{si } 1 < t < \alpha \\ M^\alpha(x,y)/\alpha & \text{si } t \geq \alpha \end{cases} \qquad \forall (x,y), 1 \leq x \leq m, 1 \leq y \leq n \tag{4.18}$$

Como se observa la NMHI tendrá valores reales que variaran entre 0 y 1, independientemente del valor de t, lo que nos permite comparar capturas de diferente duración.

Recapitulando podemos calcular la *Motion History Image* M^α para una secuencia de matrices de detección de movimiento $\mathbf{\Psi} \in [\mathbb{B}^{m\times n}]_t / \mathbf{\Psi} = (\psi_1, \ldots, \psi_t)$ de la siguiente forma:

$$M^\alpha = M^\alpha_t \tag{4.19}$$

donde M^α_i puede calcularse recursivamente como:

$$M^\alpha_i(x,y) = \begin{cases} min(i,\alpha) & \text{si } \psi_i(x,y) = true \\ M^\alpha_{i-1}(x,y) & \text{si } \neg\psi_i(x,y) \text{ y } i < \alpha \\ max(0, M^\alpha_{i-1}(x,y) - 1) & \text{si } \neg\psi_i(x,y) \text{ y } i \geq \alpha \end{cases} \tag{4.20}$$

La *Motion Energy Image* E^α puede calcularse a partir de M^α como:

$$E^\alpha(x,y) = M^\alpha(x,y) > 0 \tag{4.21}$$

En ambos cálculos, el valor $\alpha \in \mathbb{N}$ define la cantidad de *frames* que serán considerados como recientes para analizar el movimiento (ventana de tiempo).
Finalmente, para que la MHI M^α sea comparable con otras MHI generadas por secuencias de distinto tamaño debemos normalizarla, obteniendo la NMHI NM^α mediante el siguiente cálculo:

$$NM^\alpha(x,y) = \begin{cases} 0 & \text{si } t = 0 \\ M^\alpha(x,y)/t & \text{si } 1 < t < \alpha \\ M^\alpha(x,y)/\alpha & \text{si } t \geq \alpha \end{cases} \tag{4.22}$$

4.3.1. Detección de movimiento

Por lo definido en Def. 4.2, la función $\phi : [\mathbb{Z}^{m\times n}] \longrightarrow \mathbb{B}^{m\times n}$ le asigna a una secuencia de imágenes $\mathbf{I} \in [\mathbb{Z}^{m\times n}]_t$, una matriz booleana $\psi \in \mathbb{B}^{m\times n}$, que indica los pixeles en que se detectó movimiento en la imagen I_t. Una primera implementación simple de esta función consiste en obtener la diferencia absoluta entre cada posición (x,y) de un *frame* y su inmediato anterior y determinar que existe movimiento en la misma si la diferencia supera un umbral μ.

Def. 4.12. Dada una secuencia $\mathbf{I} \in [\mathbb{Z}^{n\times m}]_t$ y un umbral de movimiento $\mu \in \mathbb{Z}$, podemos definir el resultado ψ de la función ϕ^μ aplicada a $\mathbf{I}$:

$$\psi(x,y) = \begin{cases} true & \text{si } \Delta(x,y) \geq \mu \\ false & \text{si } \Delta(x,y) < \mu \end{cases} \qquad \forall (x,y), 1 \leq x \leq m, 1 \leq y \leq n \tag{4.23}$$

donde la matriz $\Delta \in \mathbb{Z}^{m\times n}$ se define por :

$$\Delta(x,y) = \begin{cases} 0 & \text{si } t = 1 \\ |I_t(x,y) - I_{t-1}(x,y)| & \text{si } t > 1 \end{cases} \qquad \forall (x,y), 1 \leq x \leq m, 1 \leq y \leq n \tag{4.24}$$

La definición anterior de la función ϕ^μ puede arrojar malos resultados de detección de movimiento si los cambios entre un *frame* y el siguiente son graduales. Para solucionar este problema se agrega el parámetro β al cálculo de la matriz de comparación Δ. Este parámetro indica con qué *frame* (de los anteriores) se deberá realizar la comparación para determinar si se produjo o no movimiento.

Def. 4.13. Dada una $\mathbf{I} \in [\mathbb{Z}^{n\times m}]_t$, un umbral de movimiento $\mu \in \mathbb{Z}$ y un valor de separación entre *frames* $\beta \in \mathbb{N}$ definimos el resultado ψ de la función ϕ^μ_β para la entrada $\mathbf{I}$, es decir $\psi = \phi^\mu_\beta(\mathbf{I})$ como:

$$\psi(x,y) = \begin{cases} true & \text{si } \Delta^\beta(x,y) \geq \mu \\ false & \text{si } \Delta^\beta(x,y) < \mu \end{cases} \qquad \forall (x,y), 1 \leq x \leq m, 1 \leq y \leq n \tag{4.25}$$

donde

$$\Delta^{\beta}(x,y) = \begin{cases} |I_t(x,y) - I_1(x,y)| & \text{si } t \leq \beta \\ |I_t(x,y) - I_{t-\beta}(x,y)| & \text{si } t > \beta \end{cases} \qquad \forall (x,y), 1 \leq x \leq m, 1 \leq y \leq n \tag{4.26}$$

Usando este método de detección de movimiento, solo es necesario mantener almacenadas las últimas β imágenes de una secuencia, para poder calcular el movimiento en la imagen actual y en las subsiguientes imágenes.

4.4. Consideraciones al cálculo del MHI

En esta sección analizamos algunas decisiones tomadas para el cálculo de la MHI.

Como comentamos en la sección 4.3 el cálculo de la MHI requiere que todos las imágenes de la secuencia tengan el mismo tamaño. Frente a esta restricción nos encontramos con dos posibilidades:

1. Realizar el cálculo de la MHI sobre el *frame* completo que obtiene nuestro dispositivo de captura y luego extraer los movimientos que se produjeron en el área detectada por el detector de caras.

2. Detectar la posición de la cara en cada *frame* capturado y calcular la MHI sobre la secuencia de imágenes formada por la extracción del área detectada en cada *frame*.

La segunda opción sufre de varios problemas originados en el hecho de que el algoritmo de detección de la cara arroja (tal como vimos en el Capítulo 3) áreas de detección que pueden variar de tamaño y posición *frame* a *frame*, incluso si la cara se mantiene en un mismo lugar. Esto hace necesario la implementación de un algoritmo que escale y traslade la detección, para que coincida en tamaño y posición.

Para simplificar este cálculo se eligió utilizar la opción 1, realizando algunas asunciones sobre la captura. En principio, vamos a considerar que para la cantidad de *frames* definida por la ventana de cálculo de la MHI (parámetro α), la posición de la cara no variará demasiado. Esto nos permite establecer que, si analizamos el área donde se detectó la cara en el último *frame* procesado, estaremos capturando todos los movimientos producidos en la cara durante los últimos α *frames*.

Si bien la detección es calculada *frame* a *frame* (perder la cara implicaría que no es necesario calcular movimiento) sólo se utiliza como detección la del último *frame* obtenido, y la MHI se calculará sobre la totalidad de la imagen (ver figura 4.3). El resultado final es la extracción de dicha área de detección a la MHI calculada para obtener solamente la región de interés (ver figura 4.4). Cabe destacar que las mismas consideraciones se aplican al cálculo del MEI.

Figura 4.3: Ejemplificación del proceso de cálculo de la MHI sobre una secuencia de 25 *frames*. La imagen de la izquierda muestra el *frame* final de la secuencia de *frame* con la detección calculada. La imagen central muestra la MHI calculada. La imagen de la derecha muestra la MHI reflejando la detección del último *frame*.

Figura 4.4: Extracción del área de interés del MHI a partir de la detección de la cara (área recuadrada en rojo en la figura 4.3

Otra decisión importante a considerar, sobre la forma de procesar los *frames* capturados para trabajar con ellos, es si vamos a utilizar los *frames* originales (representados en modelo de color RGB) o aplicaremos un filtro para modificar su modelo de color (escala de grises, HSV).

En general realizar operaciones sobre pixeles (como las realizadas en el cálculo de la MHI) en imágenes que contienen tres canales suele triplicar los tiempos de procesamiento. Por ejemplo, el cálculo de las diferencias absolutas entre *frames* debería hacerse canal a canal y luego sintetizar los resultados obtenidos en cada canal en un único canal. Por este motivo decidimos trabajar con *frame* en escala de grises, disminuyendo así los tiempos de procesamiento. Esto implica que previamente debamos convertir el *frames* capturado de un espacio de colores RGB a un espacio de colores escala de grises. Si bien esto produce un pequeño costo de tiempo inicial, éste luego se compensa al realizar las posteriores operaciones sobre imágenes de un solo canal. Finalmente, el cálculo de la MHI arroja resultados similares al trabajar con imágenes de uno o varios canales, sin embargo, los tiempos al procesar imágenes de un solo canal se reducen considerablemente, así como el consumo de memoria.

Como consideración final, aclaramos que por una cuestión de comodidad en la sintaxis cada vez que nos referimos al cálculo de la MHI consideramos la normalización de la misma NMHI.

4.4.1. Algoritmo

En esta sección describimos la estructura general del algoritmo de cálculo de la MHI para una secuencia de *frames*, explicitando las cuestiones analizadas en apartados anteriores.

El pseudocódigo que exponemos a continuación no pretende ser formal, sino dar una idea global del funcionamiento completo del algoritmo de cálculo de la MHI. Internamente existen estructuras de datos que permiten entre otras cosas: tener en cuenta la cantidad de *frames* procesados, guardar los últimos movimientos detectados, almacenar una cantidad de *frames* anteriores para comparaciones futuras y diferentes optimizaciones que permiten realizar este cálculo en forma eficiente. Sin embargo, no detallamos estas estructuras ya que son elementos propios del código fuente y no son necesarias para entender el método.

Es por ello que no definimos de manera algorítmica cada una de las funciones de cálculo de MHI. Sin embargo consideramos que este pseudocódigo, en conjunto al marco teórico definido para cada una de las funciones, completa una idea general del objetivo del algoritmo.

Algoritmo 2 Algoritmo de cálulo de MHI y MEI

Input: Secuencia finita de imágenes de un mismo tamaño (w,h)
Output: MHI y MEI final de la secuencia de imágenes

```
1:  mhi = nuevaImagenVacia(w,h)
2:  while exista el frame do
3:    frame ← obtenerFrameActual()
4:    frameGris ← obtenerEscalaDeGrises(frame)
5:    deteccion ← obtenerDetecciónDeCara(frameGris)
6:    if se detecto cara then
7:      movimiento = detectarMovimiento(frameGris)
8:      actualizarMHI(mhi, movimiento)
9:      nmhi ← normalizarMHI(mhi)
10:   else
11:     return Error("No se detectó una cara en el frame actual")
12:   end if
13:   avanzar a siguiente frame
14: end while
15: mhiValido ← extraerArea(nmhi, deteccion)
16: mei ← obtenerMEI(mhi)
17: meiValido ← extraerArea(mei, deteccion)
18: return mhiValido, meiValido
```

La función $actualizarMHI$ es la definida por la ecuación 4.20, mientras que la función $detectarMovimiento$ se corresponde con la función ϕ definida en Def. 4.13.

En la figura 4.5 se muestra el resultado de aplicar el algoritmo especificada a la secuencia de 25 *frames* que se muestra en la figura 1 (ver apéndice).

Figura 4.5: MEI y MHI obtenido sobre la secuencia de 25 *frames* de la figura 1

En el apéndice 9.3 se muestran las imágenes MHI y MEI parciales que se van calculando para la misma secuencia de *frames*. Además se muestra los resultados obtenidos por la función de detección de movimiento para cada *frame* de la secuencia.

4.5. Variantes del Método

En esta sección analizamos una variante del algoritmo propuesto. Esta variante surge con el fin de mitigar el principal problema que sufre la representación de movimiento mediante una *Motion History Image*: la oclusión de movimientos.

El objetivo de ésta sección es abordar el problema de la oclusión citado por varios autores [13, 25] y considerar algunas alternativas para su solución. Para ello nos adentramos en la causa del problema y la relación que el mismo tiene en nuestro trabajo. Finalmente se propone la forma de solucionarlo (o al menos mitigar algunos de sus efectos) incluyendo algunas de las ideas de los trabajos mencionados anteriormente, adaptándolas al contexto específico de esta tesis.

4.5.1. El problema de la oclusión de movimientos

El problema de la oclusión del movimiento ocurre cuando en una secuencia de *frames* existen movimientos que se superponen en una misma área. Como explicamos en las secciones anteriores, el cálculo de la MHI se hace sobre cada uno de los píxeles o puntos de todos los *frames*, con lo cual, si a lo largo de una secuencia de *frames* se producen movimientos diferentes en una misma región, el cálculo de la MHI actualiza cada uno de los píxeles de dicha región con los valores del movimiento más recientes. Esto hace que el movimiento que se realiza con anterioridad sea ocluido por el más reciente.

Consideremos una secuencia de 20 *frames*, representada por las imágenes que vemos en las figuras 4.6 y 4.7

Figura 4.6: *Frames* 0 al 10 de un secuencia de 20 *frames*.

Figura 4.7: *Frames* 11 al 20 de un secuencia de 20 *frames*.

Como podemos ver, existe movimiento, es decir, cambios en los valores de los pixeles durante la primer decena de *frames*. Si tomamos el cálculo de la MHI en dicha secuencia el resultado es el que se muestra en la figura 4.8. Sin embargo si tomamos la MHI de toda la secuencia completa obtenemos la imagen que se muestra en la figura 4.9.

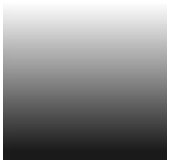

Figura 4.8: MHI calculada sobre la secuencia de 10 *frames* presentada en la figura 4.6.

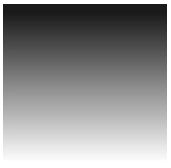

Figura 4.9: MHI de la secuencia completa de 20 *frames* conformada por las imagenes de mostradas en las figuras 4.6 y 4.7.

El resultado obtenido en la figura 4.9 se debe a que la MHI se actualiza con el movimiento de la segunda decena de *frames* (la más reciente) tapando por completo el movimiento generado durante la primer decena. Como muchas veces es necesario conocer el movimiento de toda una secuencia, este comportamiento se convierte en un problema para dicho análisis.

Para nuestro trabajo específicamente un gesto está compuesto por dos acciones primitivas. En primer lugar tenemos la *formación* del gesto, es decir, la acción conformada por los movimiento que hacen que una cara pase de un estado neutral hasta generar la forma propia del gesto en sí. Por ejemplo, en el gesto de un "Ancho Falso" (ver tabla 1.1) el momento en que la boca queda abierta. Por otro lado tenemos la acción de *relajación* en la que se pasa de una cara marcada con el gesto propiamente dicho a una cara en estado neutro (en el caso del "Ancho Falso", cuando la boca vuelva a cerrarse).

Se puede observar que los movimientos que se producen en cada acción son inversos (en cuanto al sentido de movimiento) y se desarrollan en instantes de tiempo disjuntos. Es por eso que los movimientos de relajación ocluirán a los movimientos de formación.

Una opción propuesta para solucionar este problema es detectar cuándo la acción de formación de un gesto concluye y se comienza con la acción de relajación. Este punto donde se cambia de acción se denomina *ápice* o *vértice*. La solución considerada proponía calcular dos MHI diferentes (una para los *frames* anteriores al vértice y otro para los posteriores). La dificultad para determinar la posición del vértice hizo que desestimáramos esta opción.

Luego la opción considerada consiste en analizar la secuencia completa que contiene al gesto y tratar de dividir los movimientos en MHIs diferentes según su dirección. La variante que encontramos es el método de MHI Direccionales, que describiremos a continuación.

4.5.2. MHI Direccionales (DMHI)

El método de MHI Direccionales, comúnmente denotado como DMHI, es un técnica propuesta en [25], entre varias otras, para mitigar los efectos de la oclusión de movimiento en el cálculo del MHI. El modelo no sólo toma en cuenta la descripción del movimiento plasmada en el cálculo de la MHI sino también la dirección en el cual el mismo es realizado.

La idea principal de éste método se basa en la definición de una función de ángulo de movimiento ρ que indica, dada una secuencia de imágenes, para cada posición, en que ángulo se está realizando el movimiento. A continuación lo definimos formalmente.

Def. 4.14. Para las definiciones siguientes notamos como $\mathbb{A}$ al conjunto de ángulos sexagecimales expresados en notación decimal, es decir,

$$\mathbb{A} = \{a \in \mathbb{R} / 0 \leq a < 360\} \tag{4.27}$$

Def. 4.15. Definimos la función de ángulo de movimiento $\rho : [\mathbb{Z}^{m \times n}]_t \longrightarrow \mathbb{A}^{m \times n}$ que dada una secuencia de imágenes $\mathbf{I} \in [\mathbb{Z}^{m \times n}]_t$ le asigna una matriz $\omega \in \mathbb{A}^{m \times n}$ que indica el ángulo de movimiento en cada una de las posiciones para la imagen I_t.

Def. 4.16. A partir del operador ρ y una secuencia de imágenes $\mathbf{I} \in [\mathbb{Z}^{m \times n}]_t$ definimos la secuencia de matrices de ángulos de movimiento $\mathbf{\Omega} \in [\mathbb{A}^{m \times n}]_t$ tal que $\mathbf{\Omega} = (\omega_1, \ldots, \omega_t)$ donde

$$\omega_i = \rho(\mathbf{I}_i) \qquad \forall i, 1 \leq i \leq t \tag{4.28}$$

Notaremos $\mathbf{\Omega}(x, y)$ a la secuencia de ángulos obtenidos por el operador ρ en la posición (x, y) a lo largo de las imágenes de la secuencia $\mathbf{I}$. Formalmente,

$$\mathbf{\Omega}(x, y) = (\omega_1(x, y), \ldots, \omega_t(x, y)) \qquad \forall (x, y), 1 \leq x \leq m, 1 \leq y \leq n \tag{4.29}$$

$\mathbf{\Omega}(x, y)$ indica para cada momento de la secuencia $\mathbf{I}$ en qué ángulo se esta produciendo movimiento en la posición (x, y).

A continuación definimos el concepto de dirección de movimiento.

Def. 4.17. Sea $\mathbb{A} = \{a \in \mathbb{R} / 0 \leq a < 360\}$. Se define una dirección de movimiento D_{d_i, d_f}, con $d_i \in \mathbb{A}$ y $d_f \in \mathbb{A}$, al conjunto infinito de valores reales tal que:

- Si $d_i \leq d_f \Rightarrow D_{d_i, d_f} = \{a \in \mathbb{A} / d_i \leq a \leq d_f\}$
- Si $d_i > d_f \Rightarrow D_{d_i, d_f} = \{a \in \mathbb{A} / d_i \leq a < 360 \vee 0 \leq a \leq d_f\}$

Teniendo en cuenta un conjunto de direcciones, el método de MHI direccionales calcula una *Directional Motion History Image DM* diferente para cada una de ellas, teniendo en cuenta los pixeles en los que se detectó movimiento y el ángulo del mismo. De este modo, ya no sólo observamos la evolución del movimiento en forma general, sino que vemos cómo éste evoluciona en cada una de las direcciones.

Formalmente lo definimos como:

Def. 4.18. Sea $\mathbf{\Psi} \in [\mathbb{B}^{m\times n}]_t/\mathbf{\Psi} = (\psi_1, \ldots, \psi_t)$ una secuencia de matrices de detección de movimiento. Sea $\mathbf{\Omega} \in [\mathbb{A}^{m\times n}]_t/\mathbf{\Omega} = (\omega_1, \ldots, \omega_t)$ una secuencia de matrices de ángulos de movimiento. Sea γ una dirección de movimiento. Definimos la *Directional Motion History Image* $DM^\gamma \in \mathbb{A}^{m\times n}$ para la dirección γ como $DM^\gamma = DM_t^\gamma$ donde DM_i^γ se define recursivamente como:

- Caso base $i = 1$:

$$DM_1^\gamma(x,y) = 0 \qquad \forall (x,y), 1 \leq x \leq m, 1 \leq y \leq n \tag{4.30}$$

- Caso recursivo $1 < i \leq t$:

$$DM_i^\gamma(x,y) = \begin{cases} t & \text{si } \psi_i(x,y) = true \text{ y } \omega_i(x,y) \in \gamma \\ DM_{i-1}^\gamma(x,y) & \text{sino} \end{cases} \qquad \forall (x,y), 1 \leq x \leq m, 1 \leq y \leq n \tag{4.31}$$

De la misma forma que lo hicimos en la sección 4.3 podemos incorporar el parámetro α a la definición para determinar cuántos *frames* son tenidos en cuenta. Utilizando estas consideraciones podemos definir la DMHI $DM^{\gamma,\alpha}$ como $DM^{\gamma,\alpha} = DM_t^{\gamma,\alpha}$ donde $DM_i^{\gamma,\alpha}$ se define por:

$$DM_i^{\gamma,\alpha}(x,y) = \begin{cases} min(i,\alpha) & \text{si } \psi_i(x,y) = true \wedge \omega_i(x,y) \in \gamma \\ DM_{i-1}^{\gamma,\alpha}(x,y) & \text{si } (\psi_i(x,y) = false \vee \omega_i(x,y) \notin \gamma) \text{ y } i < \alpha \\ max(0, DM_{i-1}^{\gamma,\alpha}(x,y) - 1) & \text{si } (\psi_i(x,y) = false \vee \omega_i(x,y) \notin \gamma) \text{ y } i \geq \alpha \end{cases} \tag{4.32}$$

para $x, y, i \,/\, 1 \leq x \leq m, 1 \leq y \leq n, 1 < i \leq t$

Finalmente la *Directional Motion Energy Image* $DE^{\alpha,\gamma}$ puede obtenerse a partir de la $DM^{\gamma,\alpha}$ como:

$$DE^{\alpha,\gamma}(x,y) = DM^{\gamma,\alpha}(x,y) > 0 \qquad \forall x, y \,/\, 1 \leq x \leq m, 1 \leq y \leq n \tag{4.33}$$

Análogamente a la imagen MHI, consideramos la normalización de cada una de las imágenes direccionales DMHI.

Determinación de la función de ángulo de movimiento

Las variaciones en el método de MHI direccionales difieren en la forma de definir la función de ángulo de movimiento en cada posición, es decir, la implementación de la función ρ.

Inicialmente se pensó una implementación de la función ρ basada en el método de Lucas-Kanade [28]. Esta implementación fue desestimada ya que tenía una baja performance, haciendo que el algoritmo deje de funcionar en tiempo real.

La segunda opción considerada, fue utilizar la MHI y calcular el gradiente en cada pixel. Debido a que los pixeles que registran movimientos más recientes tienen un valor mayor en la MHI, la aplicación del gradiente sobre el mismo, nos indica en que dirección se están produciendo los movimientos más recientes. Para implementar esta solución buscamos una forma de estimar el gradiente en cada posición de una imagen. La forma que utilizamos es mediante el operador de Sobel.

En [29] se propone una variante que consiste en encontrar el vecino más cercano de mayor valor (sin cruzar por pixeles que integran el fondo). Sin embargo no tenemos en cuenta esta implementación en la tesis.

En términos prácticos el algoritmo que calcula el MHI se va mantener igual que lo descripto en la sección 4.3. Además por cada *frame* procesado, actualizamos cada uno de los MHI direccionales de acuerdo a la orientación del movimiento.

Como dijimos, una forma de obtener la orientación de cada uno de los puntos que conforman el MHI es mediante el cálculo del gradiente para cada uno de ellos.

Formalmente el gradiente de una imagen $f(x,y)$ en la posición (x,y) es el vector:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial x} \end{bmatrix} \tag{4.34}$$

El vector gradiente indica la dirección de la máxima variación de f en (x, y). La *dirección* del vector gradiente es una magnitud importante que vamos a considerar dentro del cálculo del mismo. Si tomamos $\alpha(x, y)$ la representación del ángulo de dirección del vector ∇f en (x, y), podemos derivar en la siguiente fórmula:

$$\varpi(x, y) = \arctan(\frac{G_y}{G_x}) \tag{4.35}$$

donde el ángulo obtenido se mide con respecto al eje x.

Una buena aproximación para G_x y G_y puede ser obtenida mediante la aplicación del operador *Sobel* en la MHI tanto en orientación horizontal (para el cálculo de G_x en todo punto) como en orientación vertical (cálculo de G_y). Además de ser una buena aproximación tiene un bajo costo computacional. En la definición 4.15 definimos ρ aplicado a una secuencia de imágenes. En este caso utilizamos ρ'_t para definir la función de ángulo de movimiento sobre la MHI en el instante t, M_t^α. De este modo, aplicando la fórmula 4.35 podemos obtener la orientación de la MHI para cada uno de sus puntos.

Teóricamente:

$$\rho'_t(x, y) = \arctan(\frac{G_y(M_t^\alpha(x, y))}{G_x(M_t^\alpha(x, y))}) \tag{4.36}$$

Sólo nos queda definir en qué rango angular vamos a considerar a cada una de las direcciones. Por cuestiones prácticas definimos cuatro rangos angulares y asociamos cada uno de los ángulos comprendidos en dicho rango a cada una de las direcciones (Ver Figura 4.10) definidas siendo:

- $\mathbf{(45° - 135°]} \rightarrow$ arriba
- $\mathbf{(135° - 225°]} \rightarrow$ izquierda
- $\mathbf{(225° - 315°]} \rightarrow$ abajo
- $\mathbf{(315° - 45°]} \rightarrow$ derecha

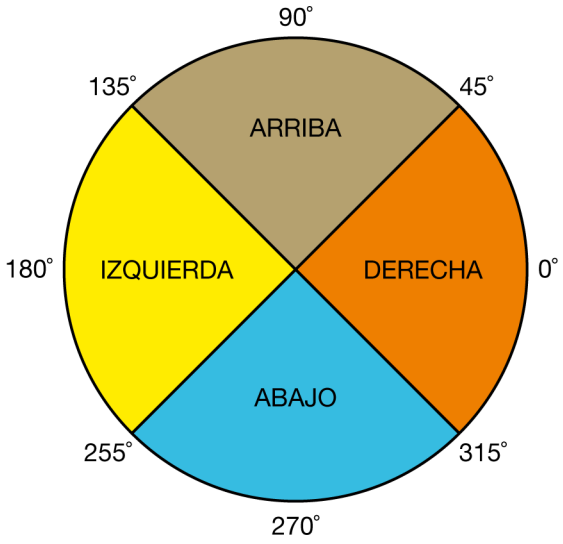

Figura 4.10: Rangos angulares asociados a cada una de las direcciones.

Entonces a esta altura ya conocemos la MHI general de una secuencia de *frames*, la MEI indicándonos donde se genera movimiento y las direcciones en las que cada movimiento evoluciona con el correr de la secuencia, plasmadas en el cálculo de las DMHI.

A continuación dejamos las DMHI para la secuencia inicial del 20 *frames* presentada en la introducción a este segmento 4.5.1.

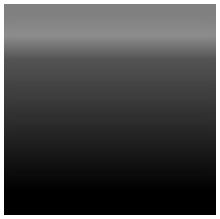

Figura 4.11: DMHI asociada a la dirección arriba (movimiento ocurrido durante los primeros 10 *frames* de la secuencia).

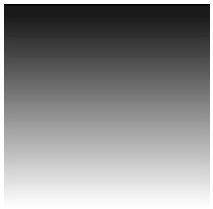

Figura 4.12: DMHI asociada a la dirección abajo (movimiento ocurrido durante los segundos 10 *frames* de la secuencia).

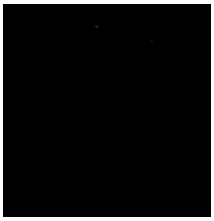

Figura 4.13: DMHI asociada a las direcciones izquierda y derecha

4.6. Conclusiones del capítulo

En este capítulo nos enfocamos en encontrar una forma de representar el movimiento que ocurre en la región facial en el transcurso del tiempo. Para representar movimiento se utilizó un método basado en *Templates Temporales* utilizando las imágenes MHI y MEI. Estas técnicas no fueron presentadas por nosotros, sino que son el resultado de varios trabajos de los cuales extrajimos ideas adaptándolas al contexto de esta tesis. También se abarcó el método de DMHI lo que nos permitió identificar la dirección en que es generado un movimiento y mitigar el problema de la oclusión existente en las señas del Truco. A partir de la representación del movimiento, en los siguientes capítulos abordaremos la extracción de características del mismo en pos de asociarlo a una seña de Truco.

Capítulo 5

Extracción de Características

En los capítulos 3 y 4 definimos el proceso para detectar y seguir un rostro en un video, así como la forma de representar los movimientos que se producen en el mismo. A partir de esa representación debemos encontrar los aspectos significativos del movimiento en la región del rostro que distinguen un gesto de otro. El objetivo luego es generar una asociación entre determinadas características de movimiento facial y las señas del juego del Truco.

En este capítulo definimos las características que queremos extraer de los movimientos y cómo las mismas son obtenidas para posteriormente ser utilizadas en el proceso de clasificación (ver capítulo 6).

El proceso de extracción de características se divide en dos etapas:

1. Segmentación del área del rostro en regiones en las cuáles se quiere extraer características.

2. Cálculo de características de movimiento sobre las regiones.

Es decir, para extraer características faciales se convierte una imagen (MHI, MEI o DMHI) en un vector de atributos descriptores de dicha imagen. Cada elemento del vector se corresponde con el cálculo de características de movimiento asociado a cada una de las regiones del rostro resultantes de la segmentación.

En el presente capítulo analizamos las dos etapas del proceso por separado, definiendo las diferentes variantes consideradas para realizarlas. En la sección 5.1 comentamos la forma de extracción de características en trabajos previos. En 5.2 se presentan cuatro tipos de segmentación sobre las imágenes de movimiento analizadas en el capítulo 4 en la región del rostro detectada con el algoritmo presentado en el capítulo 3. En 5.3 presentamos dos tipos de extracción de características a obtener a partir de las regiones definidas. En 5.4 detallamos como se combinan las características extraídas para luego ser utilizadas en la clasificación. Finalmente en 5.5 extraemos conclusiones sobre lo mencionado en este capítulo.

5.1. Extracción de características en trabajos previos

El proceso de extracción de características a partir de imágenes de movimiento fue abarcado por distintos autores en trabajos anteriores. La idea de segmentación facial y extracción de propiedades a partir de regiones localizadas es comúnmente usada como proceso de obtención de características. En [5] (Naghsh-Nilchi) la región facial es fraccionada en seis subregiones para posteriormente obtener características basadas en la dirección en que el movimiento se produce en cada una de ellas. En [7](Liao y Cohen) se descompone el rostro en nueve regiones para, mediante el cálculo de flujo óptico residual, obtener el grado de deformación de cada región. Por otro lado, trabajando con imágenes de movimiento como en esta tesis, en [11](Davis y Bobick) no se utiliza una segmentación, sino que se extraen características asociadas a las imágenes de movimiento de posturas humanas mediante el método de *Hu Moments* para el reconocimiento de formas. En [29] (Pantic, Rothkrantz) se segmenta el rostro en regiones localizadas para extraer propiedades en dichas regiones en una imagen de movimiento similar a uno de los métodos empleado en esta tesis. Finalmente en [10] (Koelstra, Pantic, Pantras) se emplea el método de descomposición *quadtree* sobre imágenes de movimiento para posteriormente extraer características asociadas a la dirección y la magnitud del mismo en las regiones segmentadas.

En este trabajo utilizamos cinco formas de segmentación facial y dos formas de extracción de características que se presentan en las siguientes secciones.

5.2. Segmentación de Imágenes de Movimiento

El objetivo de esta sección es encontrar una manera de segmentar la región del rostro detectada en el capítulo 3. Dicha segmentación es aplicada sobre las imágenes MHI, MEI y DMHI. La idea es que a partir de cada segmento se pueda extraer una característica relevante para el contexto de esta tesis.

Nos enfocamos en cinco algoritmos de segmentación que son descriptos a continuación. En el capítulo 8 evaluamos cada una de las alternativas utilizadas para segmentar.

5.2.1. Segmentación Cuadrícula

El primer algoritmo considerado para segmentar imágenes de movimiento consiste en dividir la zona facial utilizando un cuadrícula de f filas y c columnas, siendo f y c parámetros de este método de segmentación que son definidos empíricamente (ver figura 5.1). Así el área en la que se detecta el rostro queda dividida en $f * c$ regiones del mismo tamaño, no solapadas. Esta primera aproximación fue utilizada por Naghsh-Nilchi y Roshanzamir en [5]. Sin embargo la segmentación utilizada en dicho trabajo se limitaba a una cuadrícula de 3 filas y 2 columnas en las cuales el tamaño de cada uno de los recuadros variaba de acuerdo a la región facial (ver figura 5.2).

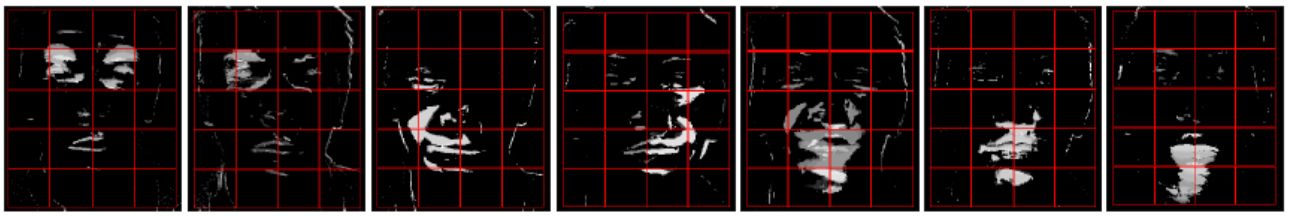

Figura 5.1: Segmentación en cuadrícula de 5 filas y 4 columnas aplicada a las MHI de los gestos Ancho de Espada, Ancho de Basto, Siete de Espada, Siete de Oro, Tres, Dos, Ancho Falso.

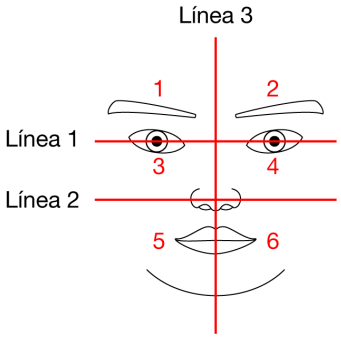

Figura 5.2: Segmentación en cuadrícula utilizada en [5] para detectar las seis emociones básicas: Alegría, Tristeza, Enojo, Sorpresa, Disgusto, Miedo.

Este algoritmo es simple y no requiere de cálculos adicionales. Además es independiente de cualquier tipo de rostro para el cual se deseen extraer características y de las señas que en el mismo se producen.

5.2.2. Segmentación por Regiones Faciales

La segunda variante que presentamos considera las regiones faciales en las cuáles tenemos el conocimiento de que se produce movimiento que nos permite diferenciar una seña de otra. Este enfoque también fue considerado en [7, 8] (ver figura 5.3). Sin embargo, las posiciones de las regiones establecidas en estos trabajos varían con respecto a las definidas en el nuestro debido a que los gestos que queremos reconocer no son los mismos. En [29] Pantic y Rothkrantz utilizaron regiones específicas para poder extraer características que le permitían identificar *Action Units*. Por otro lado en [7] Liao y Cohen utilizaron una segmentación que cubría la totalidad del rostro para poder determinar las seis expresiones básicas.

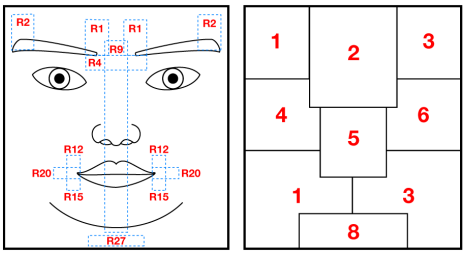

Figura 5.3: La figura de la izquierda se corresponde con la segmentación facial aplicada por Pantic y Rothkrantz en [29] en donde cada R_i representa la región i-ésima definida. La figura de la derecha se corresponde con la segmentación de Liao y Cohen en [7].

En nuestro caso las regiones son definidas de tal manera de abarcar todos los movimientos generados por las señas del juego del Truco, evitando comprender zonas sin relevancia para nuestro método como las del pelo, oídos, nariz, etc.

Para ello se divide la cara en 9 regiones cuadriculadas específicas y disjuntas entre sí:

Región Nro.	Zona de relevancia abarcada
1	Frente
2	Ceja Izquierda
3	Ceja Derecha
4	Ojo Izquierdo
5	Ojo Derecho
6	Región Superior de la Boca
7	Region Inferior de la Boca
8	Región Izquierda de la Boca
9	Región Derecha de la Boca

En en la Figura 5.4, puede observarse la segmentación implementada aplicada sobre las MHIs correspondientes a las todas las señas del Truco.

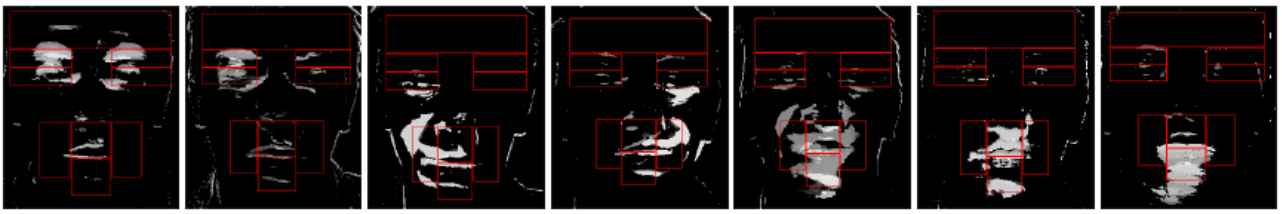

Figura 5.4: Segmentación facial aplicada a las MHI de los gestos Ancho de Espada, Ancho de Basto, Siete de Espada, Siete de Oro, Tres, Dos, Ancho Falso.

Para poder calibrar correctamente cada una de las regiones, se probó la segmentación sobre una base de datos de imágenes de personas generada por nosotros y sobre la *MUCT Landmarked Face Database*[30]. De este modo las zonas se ajustaron empíricamente para abarcar correctamente las regiones faciales asociadas a cualquier rostro humano en las bases de datos.

Este algoritmo requiere de un proceso de calibración previo y continúa siendo una implementación independiente al tipo de seña que se produzca en la cara.

5.2.3. Segmentación Quadtree

La tercera opción considerada es la descomposición *quadtree* presentada Pantic, Koelstra y Pantras en [10]. En este método de segmentación, la división en regiones del rostro está relacionada directamente con cada una de las señas del juego del Truco. El algoritmo divide el área de la cara de forma tal que las regiones que presenten mayor movimiento son divididas en una gran cantidad de pequeñas subregiones, mientras que las de menor movimiento son divididas en un menor número de subregiones de mayor tamaño. Debido a que las diferentes señas producen movimientos distintos en la cara se genera una descomposición *quadtree* por cada una de ellas.

El algoritmo de descomposición *quadtree* es iterativo. Su nombre se deriva del hecho de que en cada paso iterativo, una región a evaluar es descompuesta en cuatro subregiones rectangulares de igual tamaño si se cumplen una serie de condiciones. Para trabajar se utilizan dos parámetros definidos empíricamente:

- El parámetro τ se corresponde con el ancho o el alto mínimo que debe tener una región para poder descomponerse en cuatro subregiones. El parámetro es utilizado como criterio de parada y definido de tal modo de lograr extraer características más específicas en regiones donde se produce movimiento. Por otro lado debe tener un tamaño suficiente como para no dividir de manera excesiva en regiones que no aporten información relevante. Es decir, si este parámetro no existiera las regiones serían divididas hasta llegar al tamaño de un pixel de la imagen lo que deja de tener sentido para una correcta extracción de características.

- El parámetro σ determina una proporción de valores distintos a 0 que debe tener la región. Como se comentó anteriormente, la idea del algoritmo es dividir la región del rostro en partes más pequeñas en aquellas zonas en donde el movimiento sea mayor. Para eso, sólo se dividirá una región siempre y cuando la cantidad de pixeles distintos a 0 supere la proporción σ con respecto a la cantidad de pixeles totales de la región. Como la segmentación es aplicada sobre las imágenes de movimiento en escala de grises MHI, MEI y DMHI, el 0 representa la ausencia de movimiento.

El algoritmo comienza con la región del rostro en su totalidad y en cada paso iterativo segmenta, si es posible, las regiones obtenidas en la iteración anterior hasta el momento en que no haya más divisiones posibles. El pseudo código definido en 3 muestra el esquema de funcionamiento del algoritmo. Asimismo la figura 5.5 muestra cada uno de los pasos del algoritmo aplicados a una imagen de ejemplo.

Algoritmo 3 Algoritmo de descomposición quadtree

```
1: Se define τ como el mínimo ancho o alto de una región a dividir
2: Se define σ como la proporción de pixeles distintos a 0
3: Se inicializa R como un conjunto de regiones (Inicialmente la región del rostro)
4: while Existan en R regiones a dividir do
5:   for r ∈ R do
6:     Se define c_r como la cantidad de pixeles totales en r
7:     Se define p_r como la cantidad de pixeles distintos de 0 en r
8:     Se define w_r y h_r como el ancho y el alto de r respectivamente
9:     if p_r > σ * c_r y w_r > τ y h_r > τ then
10:      Se define regiones_r como la partición de r en cuatro regiones de igual tamaño
11:      Se quita r de R
12:      Se agregan regiones_r a R
13:    end if
14:  end for
15: end while
16: return R
```

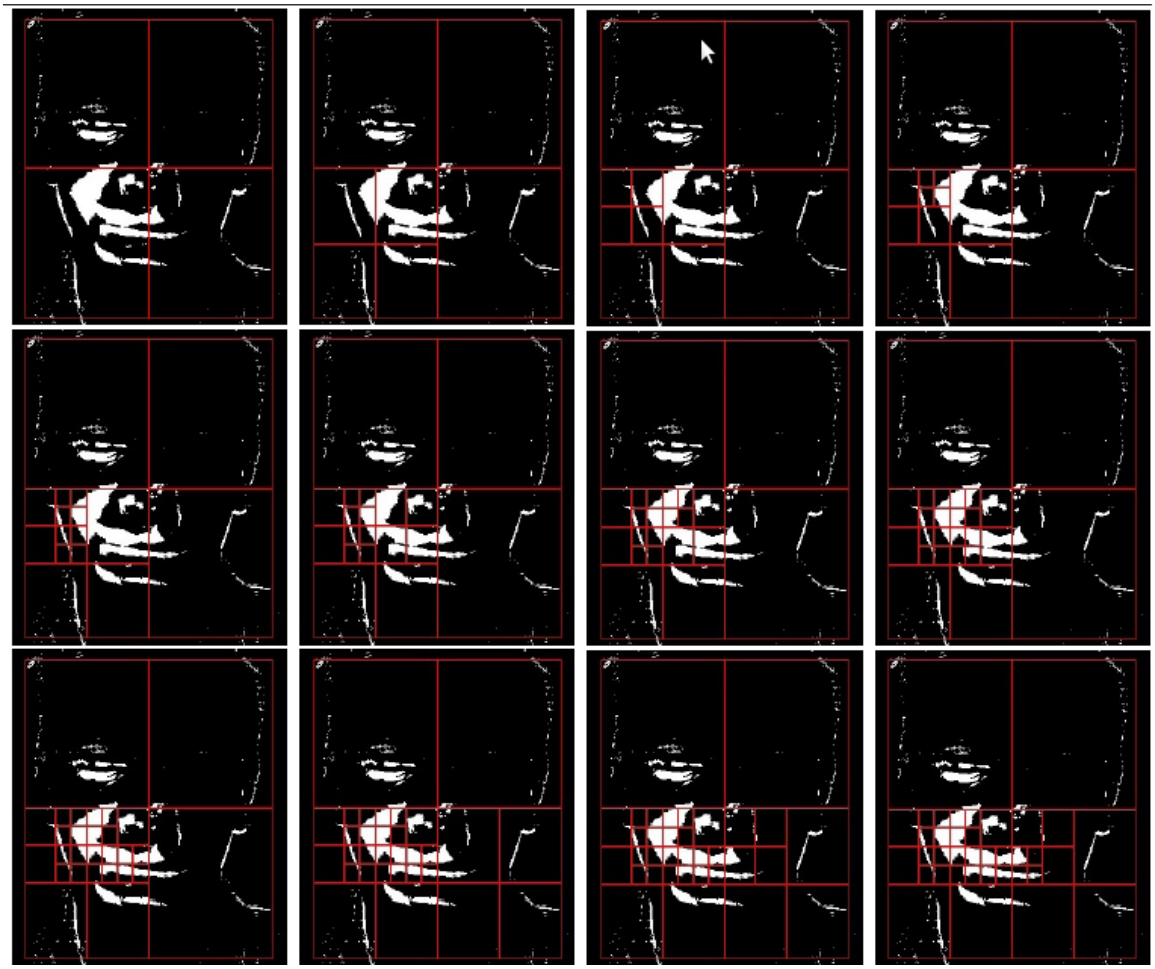

Figura 5.5: Proceso de descomposición *quadtree* utilizando como parámetro τ un 20 % del tamaño de la región del rostro y una proporción σ de 0.01.

Para poder emplear correctamente este tipo de segmentación, es necesario realizar un paso previo de calibración. Por más que se ejecute la segmentación *quadtree* sobre dos rostros distintos generando el mismo gesto, la cantidad de segmentos resultantes puede ser distinto, lo que hace incomparables ambas segmentaciones. Para ello en primer paso se elige una MHI representativa por cada uno de los gestos de alguna base de datos de gestos, y se aplica la segmentación *quadtree* a cada una de ellas. Como resultado se tienen siete segmentaciones genéricas (una por cada gesto) que serán luego aplicadas a las imágenes MHI, MEI y DMHI asociadas a cualquier rostro. La elección de las MHI asociadas a todas las señas del Truco se realiza manualmente. En la Figura 5.6 se muestra la descomposición *quadtree* para los gestos estudiados en este trabajo.

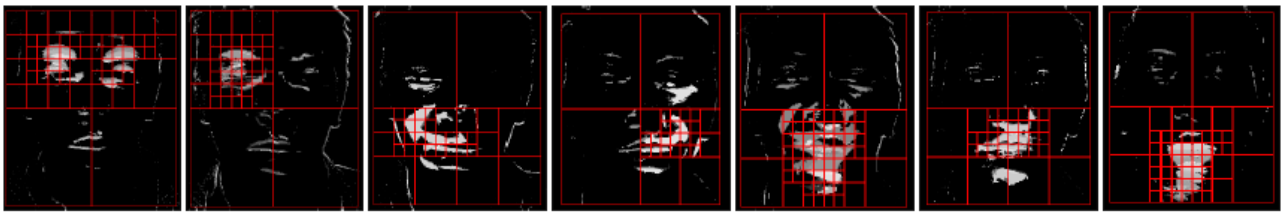

Figura 5.6: Segmentación quadtree aplicada a las MHI de los gestos Ancho de Espada, Ancho de Basto, Siete de Espada, Siete de Oro, Tres, Dos, Ancho Falso.

Al igual que la segmentación en regiones faciales, la segmentación *quadtree* requiere de un procesamiento previo. Además en este caso la descomposición sí es dependiente de la seña que se genere.

5.2.4. Segmentaciones Adicionales

Se utilizan también en de este trabajo dos tipos de segmentación adicionales cuya importancia es explicada en la sección 5.3.2.

La primera de ellas es un caso particular de la segmentación cuadrícula definida en 5.2.1. La segmentación se realiza utilizando dos filas y dos columnas en donde cada una de las regiones tienen el mismo tamaño (ver figura 5.7). De este modo la región facial queda dividida en cuatro regiones no solapadas.

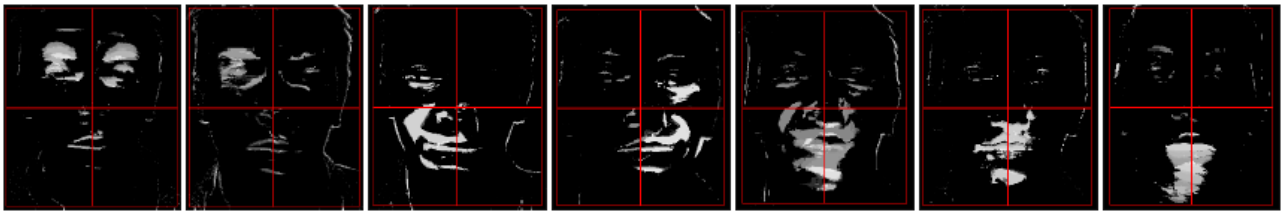

Figura 5.7: Segmentación en cuatro regiones aplicada a las MHI de los gestos: Ancho de Espada, Ancho de Basto, Siete de Espada, Siete de Oro, Tres, Dos, Ancho Falso.

El segundo tipo de segmentación es similar a la primera con la diferencia que se divide la región del rostro en dos filas, para posteriormente dividir en dos columnas sólo la fila superior. De este modo la región del rostro queda dividida en tres partes no solapadas en donde la subregión inferior tiene el doble del ancho de cada una de las subregiones superiores (ver figura 5.8).

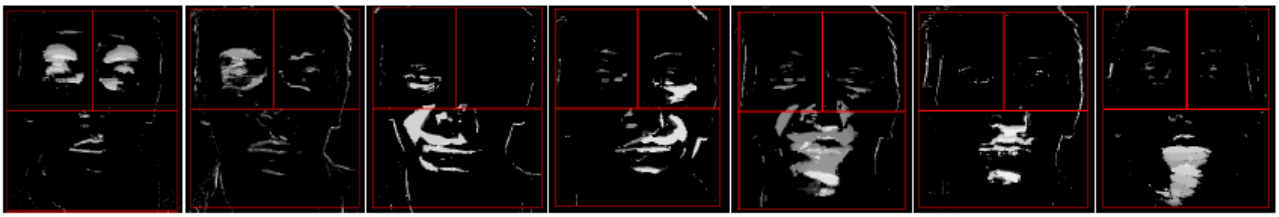

Figura 5.8: Segmentación en tres regiones aplicada a las MHI de los gestos: Ancho de Espada, Ancho de Basto, Siete de Espada, Siete de Oro, Tres, Dos, Ancho Falso.

5.2.5. Consideraciones Adicionales

Se presentaron cinco tipos de segmentación cuyo desempeño puede observarse en el capítulo 8. Las características varían de acuerdo a cada uno de los métodos comenzando con particiones simples como segmentación cuadrícula a más complejas como segmentación *quadtree*. Sin embargo existe una aspecto común a todos ellos: las regiones siempre son rectangulares. Tomamos esta decisión porque consideramos que permiten ahorrar tiempo de cómputo con respecto a otro tipo de particiones como círculos o elipses. Además concluimos en que el cálculo de características posterior no generaría diferencias significativas utilizando otras formas.

5.3. Cálculo de características de movimiento

Una vez que se tienen las regiones en las cuales se divide la cara se extraen características de movimiento asociadas a cada una de ellas. En este trabajo se presentan dos formas de calcular características asociadas al movimiento generado en un rostro. Por otro lado consideramos que el cálculo de DMHI (*Directional Motion History Images*) presentado en el capítulo 4 representa en sí una forma de extraer características del movimiento implícita ya que determina la dirección del mismo. Sin embargo no es incluída dentro de esta sección.

5.3.1. Proporción de Movimiento

El primer método de extracción se utiliza para detectar cuánto movimiento se genera dentro de una región. Una manera de determinar esta medida es mediante la utilización de un promedio sobre el valor de los pixeles dentro de la región a evaluar. Formalmente:

Def. 5.1. Dada una imagen I y una región r definimos la proporción de movimiento $C_r(I)$ de r en la imagen I como:

$$C_r(I) = \frac{\sum_{x,y \in r} I(x,y)}{size(r)} \tag{5.1}$$

donde $size(r) = w * h$ con w y h el ancho y el alto de r respectivamente.

Extrayendo esta característica de las MHI-direccionales estaremos considerando el movimiento que se produce en cada dirección.

En [10] además de la proporción de movimiento, se considera la proporción de movimiento horizontal y vertical a lo largo del tiempo. Estas medidas son usadas para poder diferenciar dos gestos que realizan los mismos movimientos pero varían en su duración. Este problema no se presenta en nuestro contexto por lo que se prescindió de usar estas medidas.

Finalmente se normaliza la proporción de movimiento de todas las regiones entre 0 y 1. El objetivo es tener una medida de comparación para cualquier rostro que indique cuánto mayor es el movimiento generado en una región con respecto a otra. Formalmente:

Def. 5.2. Dada una región $r_i \in R$, la imagen I y la proporción $c_i = C_{r_i}(I)$, en donde R son las regiones obtenidas luego de la segmentación, definimos $Norm(c_i)$ como la normalización de la proporción de movimiento de la región r_i de la siguiente manera:

$$Norm(c_i) = \begin{cases} 0 & \text{si } |maxCant - minCant| = 0 \\ \frac{c_i - minCant}{|maxCant - minCant|} & \text{sino} \end{cases} \tag{5.2}$$

donde:

$$\begin{aligned} maxCant &= max\{C_{r_i}(I) : \forall r_i \in R\} \\ mixCant &= min\{C_{r_i}(I) : \forall r_i \in R\} \end{aligned} \tag{5.3}$$

5.3.2. Momentos Geométricos

Otro método para la extracción de características es la utilización de momentos geométricos [31]. En esta sección definimos la noción de estos momentos y sus variantes para posteriormente explicar la forma de utilización en nuestro trabajo. En el capítulo 8 se analiza el comportamiento de este método y se detallan las razones de su mal funcionamiento en el contexto de esta tesis lo que lo hace inutilizable.

Descripción del método

Los momentos son propiedades numéricas que se pueden obtener de una determinada imagen. Son un mecanismo útil para la determinar las formas de los objetos. No sólo tienen en cuenta los bordes de una imagen sino todo su contenido. Existen tres tipos de momentos:

- Momentos Simples
- Momentos Centrales
- Momentos Centrales Normalizados

Los *momentos simples* de orden $(p+q)$ para una función $f(x,y)$ discreta se definen como:

$$M(p,q) = \sum_x \sum_y x^p y^q f(x,y) \tag{5.4}$$

En el caso de una imagen la función $f(x,y)$ se corresponde con el valor de la posición (x,y) en la misma. Un momento simple interesante es el momento de orden 0 definido como:

$$M(0,0) = \sum_x \sum_y f(x,y) \tag{5.5}$$

Este momento nos permite obtener el área de una imagen binaria y la superficie de una imagen en escala de grises. Otros dos momentos interesantes son los momentos de orden 1 ((0,1)(1,0)) definidos como:

$$M_{10} = \sum_x \sum_y x f(x,y) \quad M_{01} = \sum_x \sum_y y f(x,y) \tag{5.6}$$

Estos momentos son utilizados para calcular el centro de masa (x_c, y_c) de una determinada imagen de la siguiente manera:

$$x_c = \frac{M_{1,0}}{M_{0,0}} \quad y_c = \frac{M_{0,1}}{M_{0,0}} \tag{5.7}$$

Los *momentos centrales* se utilizan para reconocer figuras en una imagen independientemente de su posición. El momento central de orden $(p+q)$ se define como:

$$\mu_{p,q} = \sum_x \sum_y (x - x_c)(y - y_c) f(x, y) \tag{5.8}$$

Los *momentos centrales normalizados* por su parte permiten reconocer figuras independiente de su tamaño. El momento central normalizado de orden $(p+q)$ se define como:

$$\eta_{p,q} = \frac{\mu_{p,q}}{M(0,0)^{\frac{p+q}{2}+1}} \tag{5.9}$$

Por definición los momentos centrales y los normalizados de orden 1 cumplen:

$$\mu_{1,0} = \mu_{0,1} = \eta_{1,0} = \eta_{0,1} = 0 \tag{5.10}$$

Los momento centrales de orden 2 $(mu_{2,0}, mu_{0,2}, mu_{1,1})$ se utilizan entre otras cosas para extraer características sobre la componente horizontal, la vertical, y la simetría de la figura entre otras.

En [11] se propone usar *Momentos de Hu* [32] para describir la forma que presenta el movimiento en cada región. Los *Momentos de Hu* son capaces de caracterizar la forma de una figura de manera invariante a su escala y posición pero incluyendo la rotación como una nueva invarianza. Este método presenta la ventaja de no requerir demasiado tiempo de cómputo. Los 7 momentos de Hu están definidos a partir de los momentos centrales normalizados de orden 2 y 3 de la siguiente manera:

$$I_1 = \eta_{20} + \eta_{02} \tag{5.11}$$
$$I_2 = (\eta_{20} - \eta_{02})^2 + (2\eta_{11})^2 \tag{5.12}$$
$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{5.13}$$
$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{5.14}$$
$$I_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \tag{5.15}$$
$$(3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \tag{5.16}$$
$$I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \tag{5.17}$$
$$I_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - \tag{5.18}$$
$$(\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]. \tag{5.19}$$

En el caso de esta tesis no utilizaremos *Momentos de Hu* dado que no nos interesa obtener descriptores invariantes a rotación. De este modo emplearemos como método de extracción de características los momentos centrales y los normalizados.

Los momentos geométricos suelen aplicarse a figuras de las cuales se quiere conocer su forma. Para ello la segmentación previa debe permitir identificar de manera clara las figuras de las cuales se desea extraer características. Es por ello que en la sección 5.2.4 decidimos dividir el rostro en regiones que nos permitiesen identificar las zonas de movimiento en el mismo lo suficientemente generales como para poder analizar la forma de dicho movimiento de manera completa.

5.4. Relación con los métodos de clasificación

Las características extraídas de los MHI-direccionales según cada segmentación son combinadas en un único registro que concatena los datos del MHI, el MEI y los MHI-direccionales. Esto nos permite agrupar la información de cómo y dónde se produce el movimiento en un único registro. Formalmente si tenemos un región $R \in \mathbb{R}^n$ en donde n son la cantidad de subregiones de las cuales se extraen características, la combinación de todas ellas conforma un único registro $T \in \mathbb{R}^{6n}$(un MHI, un MEI y los cuatro DMHI). Este registro es utilizado luego para clasificar (ver capítulo 6) ya sea como un elemento de entrenamiento, como para la predicción de una seña a partir de una captura.

5.5. Conclusiones del capítulo

En este capítulo definimos el módulo encargado de la extracción de características a partir del movimiento generado en la región del rostro. Para ello se descompuso el proceso en dos partes: segmentación del área de la cara y cálculo de características a partir de las regiones segmentadas. Realizamos esta descompocisión con el objeto de poder estudiar cada proceso por separado a fin de encontrar la combinación entre ambos métodos que mejor se adapte a la solución deseada. Se estudiaron varias maneras de segmentar un rostro, variando la complejidad del algoritmo de segmentación. Se definieron métodos dependientes e independientes de la seña que se realiza con el objetivo de determinar si la dependencia es un factor importante en la segmentación de un rostro. Para extraer características se propusieron dos cálculos, uno basado en proporción de movimiento y otro en la forma que el mismo produce en el rostro cuando se genera un gesto. Si bien en los trabajos previos se utilizan nociones similares en cuanto a segmentación y extracción o se combinan ambos procesos en uno único, nos pareció interesante combinar dichas implementaciones en pos de analizar las ventajas y desventajas de cada método en el contexto de nuestro trabajo incluso realizando variaciones para adaptarlo al mismo. En el capítulo 8 analizaremos los resultados entre las posibles combinaciones en los métodos de extracción y segmentación. Se podrá observar también si a medida que aumenta la complejidad de los algoritmos mejoran o no los resultados.

Capítulo 6

Clasificación de Gestos

En el capítulo 5 analizamos diferentes formas de extracción de características de nuestras imágenes de representación de movimiento (*Motion History Image*, *Motion Energy Image* y *Directional Motion History Images*). En el presente capítulo, vamos a estudiar los métodos que permiten predecir, a partir de un conjunto de características, qué gesto se está realizando.

El capítulo comienza con la descripción del concepto de clasificación en el marco del Aprendizaje Automático en la sección 6.1. En la sección 6.2 se describen los dos algoritmos de clasificación utilizados en la tesis: Máquinas de Vectores de Soporte o *Support Vector Machines* (SVM) y K-Vecinos más Cercanos (K-NN). Luego, en la sección 6.3 se explica la forma de aplicación de los algoritmos en el contexto de nuestra tesis. La sección 6.4 introduce diferentes formas de rechazar aquellos gestos que no son considerados señas. Finalmente, comentamos las conclusiones del presente capítulo en la sección 6.5.

6.1. Clasificadores

El Aprendizaje Automático es la rama de la Inteligencia Artificial que se ocupa del proceso de inducción del conocimiento en un sistema informático. En esencia, se trata de crear sistemas capaces de generalizar conocimiento a partir de información suministrada en forma de ejemplos. La función de estos sistemas es inferir una clase etiquetada a partir de un conjunto de características tomadas como entrada.

Los sistemas de inferencia pueden ser generados mediante aprendizaje automático en forma supervisada o no supervisada. La diferencia entre ambas formas es el conocimiento, a priori, que se tiene acerca de la información suministrada a los mismos. En el Aprendizaje Supervisado se conocen las clases para cada uno de los ejemplos, por lo tanto la información se proporciona en forma de ejemplos etiquetados. Comúnmente esta forma de aprendizaje se utiliza en los problemas de clasificación. En este tipo de problemas se desea clasificar ejemplos asociándolos a distintas etiquetas o clases conocidas, con lo cual los sistemas de inferencia son denominados *clasificadores*. Por su parte, en el Aprendizaje No Supervisado los ejemplos no están etiquetados y se busca, entre otras cosas, agruparlos de acuerdo a sus características.

En nuestra tesis utilizamos sistemas generados mediante Aprendizaje Supervisado. Por lo tanto, la información proporcionada al sistema o clasificador en el proceso de aprendizaje tiene dos componentes fundamentales:

- Registros: son los datos caracterizados que se le suministran al sistema.
- Etiquetas de clase: son las clases que se corresponden con los registros suministrados.

Los registros y las etiquetas pueden representarse como un conjunto:

$$D = \{(x_i, y_i) | \forall i : 1 \leq i \leq n, x_i \in \mathbb{R}^m, y_i \in \mathbb{Z}\} \tag{6.1}$$

El cardinal de D ($\#D = n$) es la cantidad de ejemplos que se le proporcionarán al sistema y m es la cantidad de características relevadas para dichos ejemplos. Cada uno de los $y_i \in \mathbb{Z}$ representan las etiquetas de clase asociadas a cada elemento de entrenamiento o registro x_i.

A continuación presentaremos un ejemplo sencillo que muestra la forma de estructurar los datos descripta en el párrafo anterior. Supongamos que deseamos clasificar a las personas en hombres y mujeres. Para ello vamos a

Altura(cm)	Peso(kg)	Etiqueta
170	70	Hombre
160	50	Mujer
180	80	Hombre
185	80	Hombre
165	60	Mujer
170	60	Mujer
200	100	Hombre
155	50	Mujer
160	80	Hombre
175	70	Hombre

Tabla 6.1: Ejemplos de entrada para un sistema de clasificación de personas en hombres y mujeres.

definir 2 propiedades que nos permitan caracterizarlas: altura (cm) y peso (kg). Luego debemos determinar un conjunto de datos de entrenamiento que nos ayuden a inducir el conocimiento adecuado al sistema. Supongamos que nuestro conjunto de datos consta de 10 ejemplos que se especifican en la tabla 6.1. En este caso en particular, $\#D = 10$ y cada uno de los $x_i \in \mathbb{R}^2$ está dado por las primeras dos columnas de la tabla. La última columna conforma cada uno de los y_i. Transladando las clases a enteros podemos denotar a la clase Hombre con el entero $y_i = 1$ y a la clase Mujer con el entero $y_i = -1$.

El clasificador generado a partir de los datos de la tabla 6.1, será capaz de deducir, dado un nuevo registro en forma de (altura, peso), si se corresponde con un hombre o con una mujer.

Cabe destacar que un clasificador no es un sistema completamente fiable, dado que puede llegar a cometer errores en la clasificación. Estos errores pueden producirse por diferentes causas, por ejemplo: los datos de entradas no caracterizan correctamente a las clases del problema, existen errores en la obtención de datos de entrada o no hay suficientes datos de entrenamiento. Luego para generar un clasificador fiable debemos contar con:

- Una base suficiente de ejemplos diversos que contenga datos correctos.
- Un conjunto de características que describa correctamente las clases del problema.
- Un clasificador adaptado al contexto particular del problema de forma tal que se maximice la tasa de predicciones correctas.

6.1.1. Método de entrenamiento de un sistema de clasificación supervisado

A continuación definimos la secuencia de pasos que seguimos para generar un sistema de clasificación supervisado.

1. Definir las clases de clasificación de nuestro problema y las etiquetas asociadas a las mismas. En nuestro caso tenemos 7 clases (una para cada gesto de Truco) y sus etiquetas son: Ancho de Espada, Ancho de Basto, Siete de Espada, Siete de Oro, Tres, Dos y Ancho Falso.
2. Seleccionar un conjunto de muestras para entrenar al sistema. Las muestras, en nuestro problema, se corresponden con las imágenes de representación de movimiento generadas a partir de diferentes capturas de videos en las que se ejecuta un gesto (agrupadas por los gestos que se están realizando).
3. Definir la forma de obtener los registros que van a conformar nuestros datos de entrada. En nuestro caso esto se determina a partir de la selección de un método de extracción de características según lo definido en el capítulo 5.
4. Extraer las características de los ejemplos proporcionados y generar los registros x_i. Luego asociar los registros de entrada con su etiqueta correspondiente y_i a fin de determinar los datos de entrada.
5. Escoger un algoritmo de clasificación y generar un clasificador a partir de los datos de entrada. Los algoritmos de clasificación utilizados se explican en la sección 6.2.

6. Evaluar la confiabilidad del clasificador mediante experimentos. En este paso se definen ciertas métricas y se ajustan los parámetros de los clasificadores con el fin de determinar aquellos que producen mejores resultados. Los resultados de los experimentos realizados pueden verse en el capítulo 8.

7. Seleccionar el clasificador que produzca predicciones más certeras, según los experimentos y las métricas definidas.

6.2. Algoritmos de Aprendizaje Supervisado

Dentro del conjunto de métodos de aprendizaje supervisados existen diferentes familias de algoritmos que definen cómo generar y entrenar un clasificador. En nuestra tesis nos enfocamos particularmente en dos de estos subconjuntos de algoritmos: K - Vecinos más Cercanos (*K-Nearest Neighbors, K-NN*)[33] y Máquinas de Vectores de Soporte (*Support Vector Machines, SVM*)[34]. En las siguientes secciones describimos estos algoritmos y justificamos el porqué de la elección de los mismos teniendo en cuenta nuestro contexto particular.

6.2.1. K-Vecinos más Cercanos (K-NN)

El algoritmo intenta predecir la probabilidad de que un conjunto de datos pertenezca a una clase determinada. El mismo se caracteriza por no realizar ningún procesamiento durante la fase de aprendizaje, es decir, a partir de la información de entrenamiento, no se construye un modelo, simplemente se guardan todas las instancias. Posteriormente, en la etapa de clasificación, es en donde se realiza la mayor parte del procesamiento. La nueva instancia se clasifica en función a las distancias más cercanas con respecto a las instancias almacenadas. Dicha distancia se calcula a partir del valor de las propiedades de cada una de ellas.

La Figura 6.1 muestra para el ejemplo de Hombres y Mujeres, la elección de 4 vecinos, obteniendo como resultado de la clasificación para la nueva instancia de un Hombre.

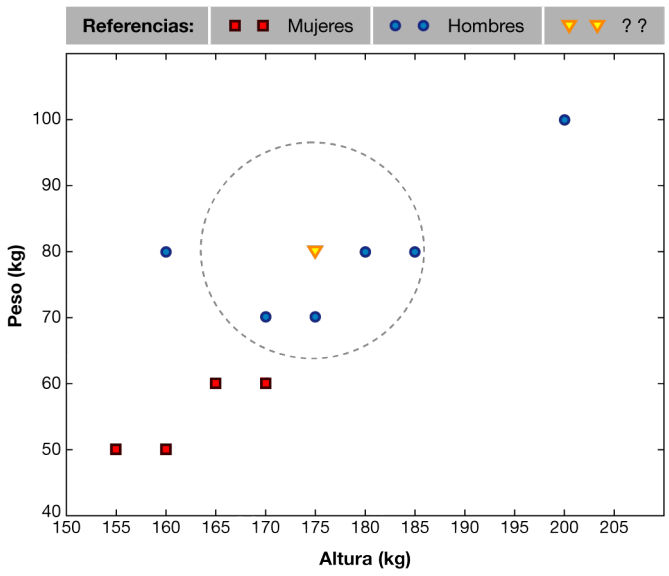

Figura 6.1: Resultados de la clasificación de una nueva instancia para el ejemplo de Hombres y Mujeres. La nueva instancia es clasificada como Hombre dada la mayoría de instancias de la clase Hombre entre los 4 vecinos más cercanos.

Descripción del método de Vecinos más Cercanos

Podemos considerar a cada una de las instancias de entrenamiento de un método de aprendizaje supervisado como vectores en un espacio multidimensional. Luego el método de Vecinos más Cercanos particiona este espacio en regiones asociadas a las etiquetas de los ejemplos de entrenamiento. Una instancia nueva en el espacio es

asignado a una determinada clase si la misma es más frecuente entre las k instancias de entrenamiento más cercanas.

Generalmente como función de distancia se utiliza la *Distancia Euclidiana*, pero pueden implementarse otras. En particular en esta tesis no se implementaron funciones de distancia adicionales.

Si consideramos una instancia de entrenamiento $x = (x_1, \ldots, x_n)$ y una instancia a clasificar $x' = (x'_1, \ldots, x'_n)$, la distancia d entre ambas se corresponde con:

$$d(x, x') = \sqrt{\sum_{i=1}^{n} (x_i - x'_i)^2} \tag{6.2}$$

Una vez que se calcula la distancia entre los vectores almacenados y el nuevo vector, se seleccionan las k instancias más cercanas. El nuevo ejemplo es clasificado con la clase que más se repite entre las clases de los vectores seleccionados.

Este clasificador tiene entre otras, las siguientes ventajas:

- Es simple de entender.
- La fase de entrenamiento no requiere cálculos.
- Es robusto con respecto a ruido incorporado en la información de entrenamiento, ya que por más que existan algunos elementos que no estén correctamente etiquetados dentro de dicha información la probabilidad de que influyan en la elección posterior es baja ya que el método se basa en elegir sobre los k vecinos más cercanos y no en sólo uno.

Además en [33] Cover y Hart demuestran que si la cantidad de elementos de entrenamiento tiene a infinito la probabilidad de error del vecino más cercano ($k = 1$) está acotado por el doble de la probabilidad de mínimo error de *Bayes* independientemente de la función de distancia utilizada. A partir de las ventajas mencionadas decidimos utilizar este método de clasificación para este trabajo.

Se debe tener especial cuidado en la elección de atributos o propiedades relevantes en un problema particular. No tiene sentido incluir ruido o incrementar el tiempo de cómputo incluyendo atributos irrelevantes. Muchas veces sucede que si bien no hay atributos irrelevantes, hay algunos que tienen mayor incidencia sobre la clasificación que otros. Este problema puede solucionarse asignando pesos a dichos atributos en el cálculo de la función de distancia. En el caso de ésta tesis particular vamos a suponer que todos los atributos tienen pesos equitativos.

La elección del valor k (la cantidad de vecinos a tener en cuenta), depende mucho de la cantidad de atributos del problema, de la relevancia de los mismos y de la cantidad de muestras de entrenamiento. Si tenemos poca cantidad de ejemplos de entrenamiento y atributos irrelevantes, el resultado de la clasificación puede ser el incorrecto debido a que dentro de los k vecinos se encuentren varias clases equitativamente distribuidas. Es una buena práctica variar experimentalmente los valores de k de acuerdo al problema que se quiere modelar. Esta tarea suele realizarse mediante validación cruzada. Por otro lado, es conveniente utilizar un conjunto de entrenamiento cuyas clases estén distribuidas de manera uniforme. En nuestro caso obtuvimos mejores resultados seleccionando 3 o 5 vecinos más cercanos.

6.2.2. Máquinas de Vectores de Soporte (SVM)

Los algoritmos de Máquinas de Vectores de Soporte (*Support Vector Machines, SVM*) son una familia de métodos, utilizados entre otras cosas, para construir clasificadores a partir de un conjunto de muestras de entrenamiento etiquetadas. Estos métodos, a diferencia de los métodos de vecinos más cercanos, sí construyen un modelo con la información de entrenamiento. En esta tesis utilizamos el algoritmo SVC (*Suppor Vector Classification*) [35, 34] como mecanismo de clasificación dentro de la familia SVM.

El modelo que generan separa, dentro de un espacio multidimensional, cada una de las clases por una distancia (margen) lo más amplia posible. Esta separación se realiza mediante hiperplanos de dimensiones altas o infinitas a modo de lograr una óptima separación entre cada una de las clases. Los hiperplanos se determinan de forma tal que mantengan la máxima distancia (margen funcional) con los puntos pertenecientes a las distintas clases más cercanos al mismo. Cuanto mayor sea el margen funcional menor va a ser el error de generalización del clasificador.

Se denomina vectores de soporte a los vectores que conforman los dos hiperplanos paralelos al hiperplano buscado, siendo la distancia entre ellos la mayor posible, siempre y cuando las clases sean separables. En caso

de que no lo sean también se agregan aquellos ejemplos mal clasificados. En la figura 6.2 figuran los vectores de soporte de dos clases separables distintas.

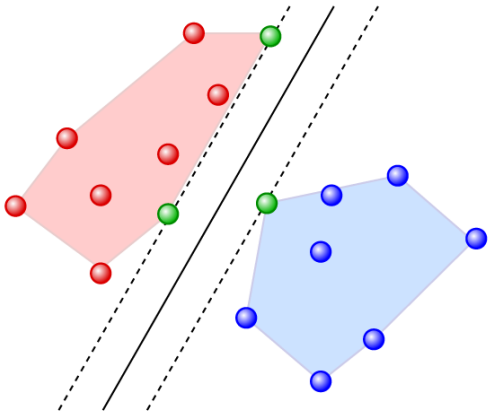

Figura 6.2: Vectores de soporte de un hiperplano coloreados en verde.

Supongamos que tenemos dos clases en el espacio, en la figura 6.3 se analizan dos hiperplanos que dividen ambas clases, sin embargo la imagen de la derecha muestra al hiperplano tal que la distancia a los puntos más cercanos de cada clase es máxima.

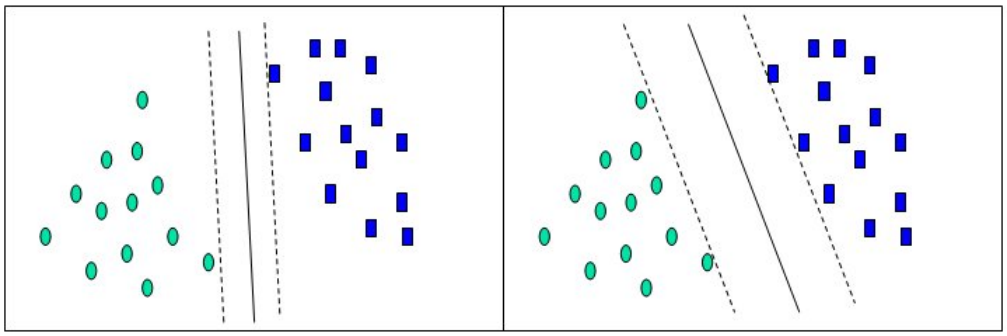

Figura 6.3: Dos formas distintas de definir un hiperplano. En el hiperplano de la imagen de la derecha se logra el margen funcional máximo.

Cabe aclarar que los métodos de Máquina de Vectores de Soporte no suelen aplicarse a problemas pequeños como los que se muestran en las figuras 6.2 y 6.3. Estos métodos resultan más aptos para problemas donde se tenga una gran cantidad de datos.

Definición del método

Teóricamente, el conjunto de n puntos en el espacio (datos de entrenamiento) podemos representarlo como:

$$D = \{(x_i, y_i) | \forall i : 1 \leq i \leq n, x_i \in \mathbb{R}^m, y_i \in \{-1, 1\}\} \tag{6.3}$$

donde cada $y_i \in \{-1, 1\}$ representa una etiqueta en un conjunto de dos etiquetas posibles y m es la cantidad de atributos definida para cada instacia de entrenamiento x_i.

Queremos entonces encontrar un hiperplano que divida a todos los puntos que tienen etiqueta y_i = 1 de los que tienen etiqueta y_i = -1. Cualquier hiperplano puede ser descrito como el conjunto de puntos x que satisfacen:

$$w^T \cdot x - b = 0 \tag{6.4}$$

donde $\cdot$ denota el producto matricial y w es el vector normal al hiperplano. El parámetro $\frac{b}{\|w\|}$ determina el desplazamiento desde el origen a lo largo del vector normal w.

Se quiere escoger w y b a modo de maximizar la distancia con los dos hiperplanos paralelos al hiperplano buscado que aún actúen cómo límite para las dos clases. Dichos hiperplanos pueden ser descritos como:

$$\begin{aligned} w^T \cdot x - b &= 1 \\ w^T \cdot x - b &= -1 \end{aligned} \tag{6.5}$$

Podemos observar estos conceptos en la figura 6.4

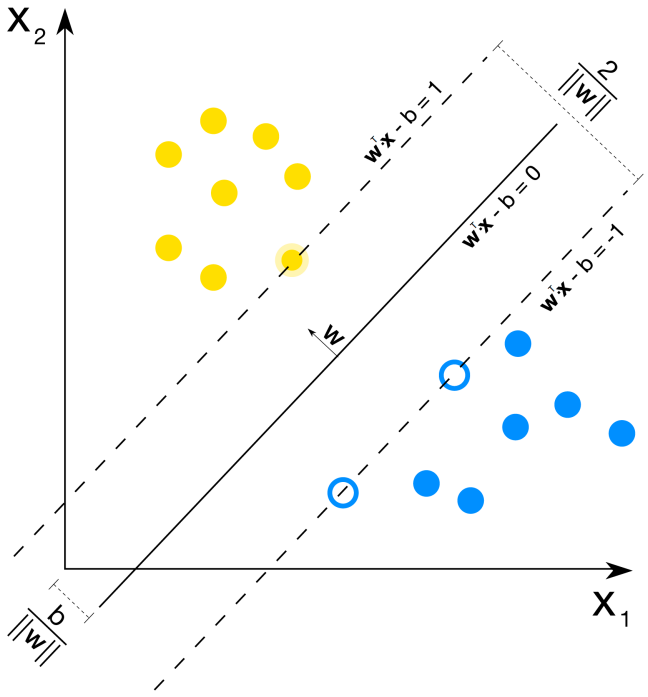

Figura 6.4: Hiperplano delimitador de dos clases.

Si los datos son linealmente separables la distancia entre los dos hiperplanos delimitadores puede ser calculada como $\frac{2}{\|w\|}$ con lo cual maximizar dicha distancia implicaría minimizar $\|w\|$. Además no se quiere que existan puntos contenidos en dicha distancia. Es decir, debe valer:

$$w^T \cdot x_i - b \geq 1 \tag{6.6}$$

para los x_i pertenecientes a la primera clase, y

$$w^T \cdot x_i - b \leq -1 \tag{6.7}$$

para los x_i pertenecientes a la segunda clase. Estas dos ecuaciones pueden quedar escritas como:

$$y_i(w^T \cdot x_i - b) \geq 1 \quad \forall i, 1 \leq i \leq n \tag{6.8}$$

Con lo cual se busca minimizar

$$\|w\| \quad \text{sujeto a } (\forall i, 1 \leq i \leq n)\ y_i(w^T \cdot x_i - b) \geq 1 \tag{6.9}$$

En la forma **primaria** de resolución del problema se reemplaza $||w||$ con $\frac{1}{2}||w||^2$ ya que al ser monótona creciente es lo mismo minimizar una que la otra y se evita lidiar con raíces cuadradas, manteniendo la misma solución. De este modo el problema cambia a minimizar (en w, b)

$$\frac{1}{2}||w||^2 \quad \text{sujeto a } (\forall i, 1 \leq i \leq n)\ y_i(w^T \cdot x_i - b) \geq 1 \tag{6.10}$$

Para la resolución del mismo se utilizan los multiplicadores de Lagrange, definiendo la función de Lagrange como:

$$J(w,b,\alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i[y_i(w^T \cdot x_i - b) - 1] \tag{6.11}$$

Las variables auxiliares no negativas α_i son denominadas multiplicadores de Lagrange. La solución es encontrada mediante el punto de silla en la función minimizado con respecto a w y b, y maximizado con respecto a α. Si la restricción de la ecuación 6.10 no se satisfaciera, es decir $(y_i(w^T \cdot x_i - b)) - 1 < 0$, J se incrementaría a medida que se incrementan los α_i. Al mismo tiempo w y b deberían cambiar de tal forma de decrementar J. Para evitar que $\alpha_i((y_i(w^T \cdot x_i - b)) - 1)$ se convierta en un número arbitrariamente grande y negativo el cambio en w y b debe garantizar que se satisfaga la restricción. Del mismo modo, se puede entender que para todas aquellas restricciones que no sean necesariamente igualdades, es decir para aquellas en que $(y_i(w^T \cdot x_i - b)) - 1 > 0$, los correspondientes α_i deben ser 0 y este valor de α_i es el que maximiza J. Esta última consideración proviene de las condiciones adicionales de optimalidad Karush-Kuhn-Tucker(KTT)[36]. Estas mismas condiciones son las que implican que anulando las derivadas parciales de J respecto a w y b:

$$\frac{\mathrm{d}J(w,b,\alpha)}{\mathrm{d}w} = 0 \tag{6.12}$$

$$\frac{\mathrm{d}J(w,b,\alpha)}{\mathrm{d}b} = 0 \tag{6.13}$$

puede alcanzarce:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \qquad y \qquad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{6.14}$$

Sólo algunos de los α_i serán mayores a 0. Los x_i asociados serán los vectores de soporte, que satisfacen $y_i(w^T.x_i - b) = 1$. De este modo los vectores de soporte cumplen:

$$w^T \cdot x_i - b = 1/y_i = y_i \Leftrightarrow b = w^T.x_i - y_i \tag{6.15}$$

que permite determinar el desplazamiento b.

Se pueden eliminar las variables b y w alcanzando lo que se denomina el **problema de optimización dual**. Este es el problema que se resuelve en la práctica. En este caso, encontrar el hiperplano de máximo margen y la posterior tarea de clasificación puede realizarse en función a los vectores de soporte. Para ello se expande la ecuación 6.11 como:

$$J(w,b,\alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i y_i w^T.x_i - b\sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i \tag{6.16}$$

y aplicando las ecuaciones 6.14 llegamos a:

$$Q(\alpha) = J(w,b,\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j x_i^T x_j \tag{6.17}$$

Luego se debe maximizar (α_i) sujeto a:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \qquad y \qquad \alpha_i \geq 0, \qquad \forall i, 1 \leq i \leq n \tag{6.18}$$

Función Kernel

Supongamos que tenemos un espacio bidimensional pero que no existe un hiperplano (en este caso una recta) que nos permita hacer una separación factible entre las clases como muestra la figura 6.5. En estos casos se utiliza una función Kernel $K(x_i, x_j)$ que nos permite proyectar los puntos de muestra a un espacio superior en donde la división por un hiperplano sea factible. El hiperplano encontrado luego es mapeado al espacio original como resultado final.

Para realizar la proyección es necesario una función $\phi(x)$ que mapee los vectores de entrada a vectores en un espacio de dimensiones superiores Una función Kernel es una función que permite mapear un producto matricial de un espacio(*input space*) a otro (*feature space*). Formalmente

$$K(x_i, x_j) = \phi(x_i).\phi(x_j) \tag{6.19}$$

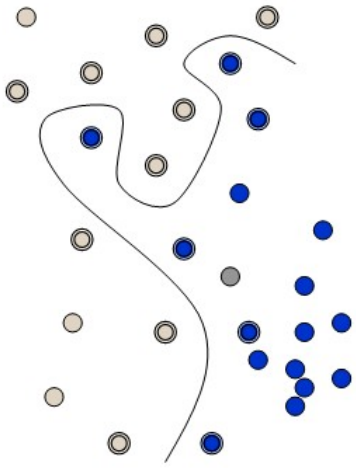

Figura 6.5: Espacio bidimensional en el que no existe una recta que permite dividir ambas clases.

Si analizamos la ecuación 6.17 la función Kernel está definida como $K(x_i, x_j) = x^T.x_j$. Es decir, dicha ecuación puede derivar en la forma:

$$Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{6.20}$$

Este tipo de función Kernel se denomina lineal. Sin embargo muchas otras funciones que permiten realizar el mapeo del producto matricial de un espacio a otro. Algunas que se utilizan habitualmente son:

- Polinomial (homogéneo):$K(x_i, x_j) = (x_i.x_j)^d$
- Polinomial (no homogéneo): $K(x_i, x_j) = (x_i.x_j - 1)^d$
- Gaussian Radial Basis o RBF(Radial Basis Function): $K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2)$ para $\gamma > 0$. Ver figura 6.6
- Tangente Hiperbólica: $K(x_i, x_j) = \tanh(kx_i.x_j + c)$ para algún $k > 0$ $c < 0$

Por ejemplo en el caso de la función Kernel $K(x_i, x_j) = (x_i.x_j)^d$ el mapeo del producto matricial queda definido de la siguiente manera:

$$\begin{aligned} K(x_i, x_j) &= (x_i.x_j)^d \\ K(x_i, x_j) &= (x_i)^d.(x_j)^d \\ K(x_i, x_j) &= \phi(x_i).\phi(x_j) \end{aligned} \tag{6.21}$$

donde $\phi(x) = x^d$.

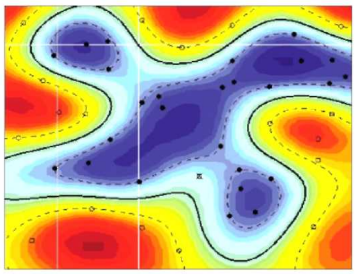

Figura 6.6: Clasificación utilizando Radial Basis Function.

Penalización

El parámetro de penalización C se utiliza en un método denominado *Soft Margin* cuando tampoco es posible separar linealmente los puntos en el espacio. La idea entonces es encontrar el hiperplano de máximo margen, teniendo en cuenta que pueden existir puntos de una clase particular contenidos en otra pero minimizando dicha cantidad. El método incluye el parámetro ξ_i que representa el grado de clasificación errónea del dato x_i. De éste modo el problema pasa a ser el siguiente: Minimizar (en w, b, ξ)

$$\begin{array}{c} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{sujeto a} \quad (\forall i, 1 \leq i \leq n) y_i(w^T.x_i - b) \geq 1 - \xi_i \end{array} \tag{6.22}$$

El parámetro C actúa como penalización y representa en su elección un *trade off* entre la distancia o margen y la cantidad de elementos mal clasificados.

SVM Multiclase

En los casos en los que se desea clasificar un conjunto de datos en más de dos clases suelen utilizarse múltiples clasificadores binarios combinándolos mediante algunas de las siguientes formas:

- **Uno contra el resto (o uno contra todos):** dado un conjunto de datos que tiene k etiquetas de clases, se generan k clasificadores binarios, uno por cada clase. Cada clasificador divide los datos de su etiqueta de los restantes datos. Este método es el empleado en esta tesis.
- **Uno contra uno:** en este caso se crea un clasificador para cada par de clases. Luego si tenemos k etiquetas de clases, se generan $k*(k-1)/2$ clasificadores. Al momento de predecir, la clase que recibe más votos es la seleccionada.

En la sección 8.2 se evalúa la performance de ambos tipos de clasificadores(*SVM y K-NN*) y se determinan los parámetros de los mismos que maximizan la tasa de reconocimiento.

Consideraciones adicionales

Si bien este clasificador nos permite clasificar una nueva instancia x mediante la función:

$$f(x) = sgn(w^T \cdot x - b) \tag{6.23}$$

en donde

$$sgn(y) = \begin{cases} 1 & \text{si } y > 0 \\ 0 & \text{si } y = 0 \\ -1 & \text{si } y < 0 \end{cases} \tag{6.24}$$

nuestra implementación utiliza la librería LIBSVM [37] que permite adicionalmente obtener como ouput de la clasificación las probabilidades de que una instancia pertenezca a cada una de las clases. Esta implementación utiliza validación cruzada a partir de la información de entrenamiento para el cálculo de dichas probabilidades. El método completo para obtener las mismas puede observarse en [38].

Alguna de las ventajas que nos llevaron a utilizar este método son:

- Tiene una poca cantidad de parámetros a ajustar.
- Tiene una gran variedad de funciones Kernel que permiten ajustar el método de acuerdo al desarrollo que se implemente, lo cual le da una gran capacidad de generalización.
- La complejidad del clasificador, mediante las distinas funciones Kernel, y el error, mediante el parámetro de penalización, pueden ser controlados explícitamente.

6.3. Aplicación de los algoritmos de clasificación

Las diferentes formas de segmentación de un rostro y la posterior extracción de características de las mismas (ver sección 8.2) van a generar variantes en la forma de aplicar los algoritmos de clasificación. Para la segmentación por cuadrícula, la de tres y cuatro regiones especiales, y la segmentación facial, se crea un

clasificador multiclase (uno contra todos), que permite determinar a partir de un conjunto de características cuál de los gestos se está desarrollando (ver Figura 6.7). Es decir, dado un vector de características $x \in \mathbb{R}^n$ y un algoritmo de clasificación $f : \mathbb{R}^n \to \mathbb{N}$ el output será una seña $y \in \mathbb{N}$ (considerando a cada seña como un número natural).

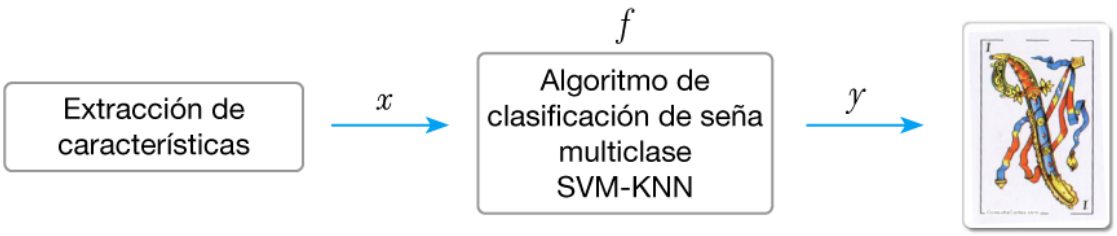

Figura 6.7: Esquema de clasificación multiclase.

En el caso de la segmentación *quadtree*, dado que para cada gesto se determina una segmentación distinta, es necesario crear un clasificador por gesto. Para combinar los resultados producidos por cada clasificador se utiliza un algoritmo de votación (ver figura 6.8). Es decir, dados siete vectores de características $x_1, \ldots, x_7$ extraídos de una misma imagen de representación de movimiento pero tomando en cuenta las diferentes segmentaciones, y siete algoritmos binarios de clasificación $f_1, \ldots, f_7$, se deduce un resultado a partir una función de votación $g : \mathbb{R}^7 \to \mathbb{N}$ que toma en cuenta la probabilidad de los resultados de cada uno de los clasificadores. En nuestro caso la función g se define como la máxima probabilidad entre todas las obtenidas.

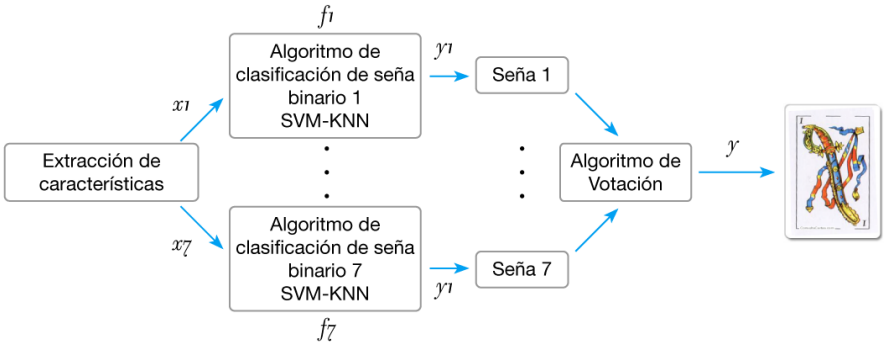

Figura 6.8: Esquema de clasificación binaria múltiple para segmentación quadtree.

6.4. Algoritmos de rechazo

En esta sección estudiamos diferentes métodos ideados para poder rechazar, a partir de la clasificación, aquellos gestos faciales que no se correspondan con ninguna de las señas de Truco. Los algoritmos de rechazo se utilizan como un filtro para descartar aquellos gestos que no son considerados señas de Truco, definiendo así dos posibles resultados de clasificación: seña o no seña. Si se determina que se produjo una seña, posteriormente se la clasifica utilizando los clasificadores mencionados en secciones anteriores, en alguna de las siete posibles señas que se quieren reconocer en esta tesis (ver figura 6.9).

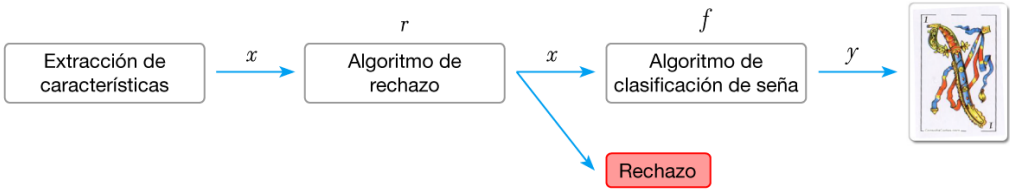

Figura 6.9: Esquema de clasificación utilizando rechazo.

Como lo muestra la imagen 6.9 nuestra tarea se centra en construir un clasificador binario que nos indique si el gesto que estamos realizando es una seña o no, de acuerdo a sus características extraídas. Para esta tarea utilizamos algunas variantes del algoritmo de clasificación SVM, para separar al conjunto de todos los gestos faciales posibles en señas y no señas de Truco. Los métodos que ideamos se aplican sobre el clasificador SVM por ser éste el que mejores resultados alcanza.

6.4.1. SVM de una clase *(One Class SVM)*

Los clasificadores SVM de una clase [39] se utilizan para detectar outliers a partir de la información de entrenamiento proporcionada al clasificador. La hipótesis principal en la que nos basamos es que una no seña debería ser considerada un outlier dentro del espacio de señas generado por el clasificador. En este caso se puede entrenar un clasificador SVM de una clase con una base de datos de señas para posteriormente predecir si una nueva muestra es considerada o no parte de dicha clase.

6.4.2. Umbralización de probabilidades

La implementación de clasificadores SVM con los que trabajamos permite obtener las probabilidades que tiene una muestra de pertenecer a cada una de las clases con las cuales el sistema fue entrenado. Teniendo en cuenta esto, podemos combinar las probabilidades de cada clase (denotadas como P_i para la clase i) en una única probabilidad ($P_{seña}$) que determina cuán factible es que se esté realizando una seña de Truco. El umbral para definir cuál es la probabilidad mínima a partir de la cuál consideraremos un gesto como seña de Truco se define experimentalmente en la sección 8.3.

Definimos dos forma de combinar las probabilidades de cada clase:

- **Probabilidad máxima:** Tomamos como probabilidad a la máxima de las probabilidades de cada clase. Formalmente definimos:

$$P_{seña} = max_i(P_i) \tag{6.25}$$

- **Cociente de las dos mayores probabilidades:** Sólo consideramos las dos mayores probabilidades y las combinamos mediante la siguiente ecuación:

$$P_{seña} = 1 - \frac{P_{max2}}{P_{max1}} \tag{6.26}$$

  donde P_{max1} es la máxima probabilidad y P_{max2} es la segunda mayor.

6.4.3. SVM binario

Como alternativa final decidimos utilizar un clasificador binario entrenando a un clasificador SVM con una base de datos integrada por señas y un subconjunto de todas las no señas y testeando contra otra base de datos de similares características. La composición de cada base de datos está definida en el capítulo 8. En este caso se consideraron a todas los señas como una única clase, y a las no señas como la otra.

6.5. Conclusiones del capítulo

En el presente capítulo analizamos el proceso de clasificación de gestos. Este proceso se encarga de determinar, a partir de un conjunto de características extraídas de una imagen de representación de movimiento, el gesto que se está realizando. Se estudiaron los algoritmo de clasificación Máquinas de Vectores de Soporte y k-Vecinos más Cercanos, con sus características y ventajas. Se analizaron además los parámetros que que tienen cada uno de los métodos para adaptarlos al contexto particular de nuestro problema. Por último se consideraron alternativas para poder rechazar aquellos gestos faciales que no son considerados señas de Truco.

En el capítulo 8 se pueden observar los resultados obtenidos para cada algoritmo, variando los parámetros seleccionados en cada uno de ellos.

Capítulo 7

Detección de Intervalos Temporales de Movimiento

En los capítulos 4, 5 y 6 analizamos a partir de una secuencia de imágenes, en la cual sabemos que existe un gesto, cómo extraer características de los movimientos que en ella se producen para determinar qué gesto se está realizando. Una captura de video puede tener varias de estas secuencias. El objetivo de este capítulo es describir un algoritmo para determinar los intervalos temporales en donde se producen los gestos.

Durante el desarrollo de un juego de Truco los compañeros de equipo pueden hablar y hacer gestos faciales que no son necesariamente señas de Truco. Por lo tanto, es importante distinguir entre dichas señas y otros tipos de gestos. Este es un requisito importante para que los jugadores pueden disfrutar del juego sin ningún tipo de restricciones. En otras palabras, en este paso, el objetivo es distinguir gestos pertinentes de otro tipo de movimientos. El resultado de este proceso es la ubicación temporal de los gestos, es decir, el primer y último *frame* en el que se desarrolla un gesto de Truco.

Esta es una tarea extremadamente difícil, porque hay una gran cantidad de gestos de la cara que tienen un patrón similar de movimiento y se producen en la misma región facial. Por otra parte, no tenemos toda la secuencia de imágenes con antelación, es decir cada *frame* es capturado de manera secuencial por la cámara o video. Por lo tanto, el método aquí propuesto debe encontrar el comienzo y el final de un gesto utilizando sólo la información previa al *frame* actual.

Un problema similar se estudió en el diseño de interfaces humano-computadora utilizadas para juegos de video, empleando técnicas de visión por computadora [40, 41]. La principal diferencia con este trabajo es que los métodos propuestos por los autores son en su mayoría para el reconocimiento de gestos de la parte superior del cuerpo, mientras que a nosotros nos preocupan los gestos faciales. Es interesante notar que este problema no ha recibido mucha atención en el área de gestos faciales, probablemente porque el trabajo previo en el área no se centra en interfaces hombre-computadora utilizando este tipo de gestos.

Teniendo en cuenta que estamos interesados en la detección de señas de Truco y que estas sólo se envían al compañero de equipo cuando ambos jugadores se miran unos a otros, el primer paso en nuestro método de reconocimiento de intervalos es considerar únicamente las imágenes que tienen una cara frontal. Posteriormente se buscan los movimientos faciales que son un potencial gesto de Truco. Una vez más, vamos a utilizar el conocimiento específico que se tiene sobre los mismos.

El capítulo se organiza de la siguiente manera. En la sección 7.1 se presenta el algoritmo utilizado para la detección de intervalos. En 7.2 se explica de qué forma se rechazan aquellos intervalos que no son considerados potenciales señas de Truco. En 7.3 se comenta cómo este algoritmo se integra con el algoritmo de reconocimiento de gestos presentado en los capítulos precedentes. Finalmente en 7.4 se extraen las conclusiones finales sobre el capítulo.

7.1. Detección de intervalos

El algoritmo que implementamos procesa la captura *frame* a *frame* indicando cuando se inicia y cuando finaliza un intervalo. Para poder detectar esto, se utiliza una medida que determina la proporción de movimiento en ciertas zonas de la cara para cada *frame*. Observando cómo evoluciona esta medida a lo largo de la captura podemos determinar que:

- El inicio de un intervalo está marcado por un crecimiento persistente en la proporción de movimiento.
- Luego se producen una serie de variaciones no muy marcadas en la proporción de movimiento.
- Finalmente se produce un decrecimiento continuo en la proporción de movimiento lo que indica el final del intervalo.

En la figura 7.1 se puede ver una gráfica representando la evolución en la proporción de movimiento en una captura de video que contiene varios gestos.

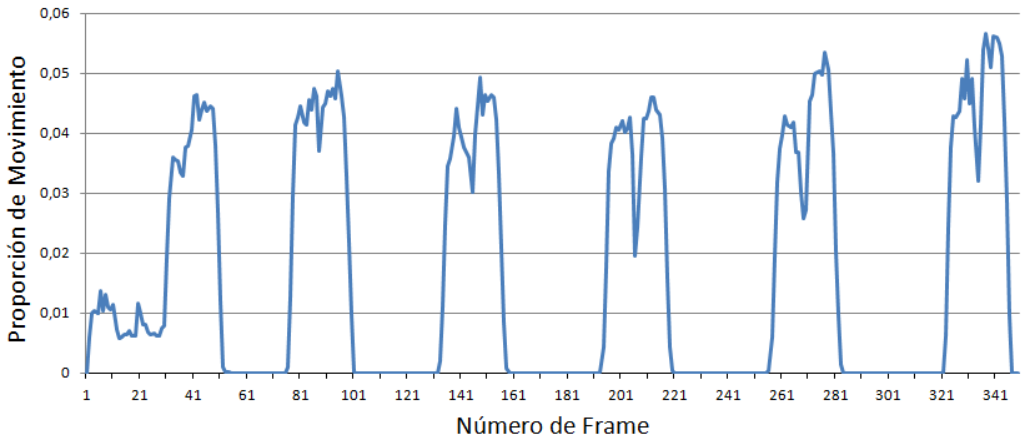

Figura 7.1: Evolución de la proporción de movimiento para una captura de video en las que se repite 6 veces el gesto del ancho de basto.

7.1.1. Proporción de movimiento

Analizamos diferentes variantes para calcular la proporción de movimiento. La que mejores resultados produjo, en relación a la detección de intervalos, está dada por la suma de movimiento a partir de la imagen de energía (MEI), tomando sólo en cuenta las regiones faciales definidas en 5.2.2. La definición 7.1 formaliza este concepto.

Def. 7.1. Sea E_k^α el MEI asociado al *frame* actual I_k, definimos la proporción de movimiento MV_k de dicho *frame* como:

$$MV_k = C_R(E_k^\alpha)$$

donde C_R es la función definida en **Def. 5.1**, R son las regiones de la segmentación facial definidas en 5.2.2 y $E_k^\alpha(x, y)$ equivale a 1 si el valor de la posición (x, y) en E_k^α es *true* y 0 en caso contrario.

Para lograr un decrecimiento marcado al finalizar la ejecución del gesto, tomamos la ventana temporal α con valor 10 sobre una MHI y MEI adicionales a los utilizados para clasificar. Valores mayores para este parámetro producen intervalos no tan marcados.

Otras variantes analizadas para calcular la proporción de movimiento tienen en cuenta la imagen de historia de movimiento o la suma de movimientos en toda la región de la cara. Si consideramos el movimiento generado en toda la cara, estaremos tomando en cuenta movimientos que no nos interesan, como el borde de la misma o la región del pelo, con lo cual se descartó esta opción. Por otro lado utilizamos la imagen de energía o MEI en lugar de la MHI, porque nos interesa considerar la proporción de movimiento existente en un determinado momento y no la evolución del mismo.

7.1.2. Algoritmo

Definida la forma de calcular la proporción de movimiento en un *frame*, pasamos a describir los algoritmos implementados para detectar los intervalos.

El algoritmo 4 se encarga de determinar cómo va evolucionando la proporción de movimiento a lo largo de la secuencia de *frames*. Este algoritmo nos indica, dada una secuencia de *frames*, si la proporción de movimiento está creciendo, decreciendo o no varía en forma pronunciada. En esencia, especifica cómo se mantiene la variable

pendienteDeMovimiento que indica la cantidad de *frames* en los que se viene detectando un movimiento continuo de decrecimiento o crecimiento. Si $pendienteDeMovimiento > 0$ se está produciendo un crecimiento en la proporción de movimiento, en el caso contrario, se está produciendo un decrecimiento.

A partir de la información provista por la variable *pendienteDeMovimiento*, un intervalo se detecta cuando la misma alcanza el valor de crecimiento $VC = 5$ y a continuación se produce un decrecimiento alcanzando el valor $VD = -5$. Estas constantes fueron definidos de manera empírica, variando sus posibles valores entre 4 y 10. Luego se detecta un intervalo cuando existen 5 *frames* en los que la proporción de movimiento crece y luego otros 5 *frames* donde la proporción de movimiento decrece.

Algoritmo 4 Algoritmo de detección de pendiente de movimiento

```
1: if Hay suficiente diferencia entre el movimiento de un frame MV_k y el anterior MV_{k-1} then
2:   diferenciaFrameAnterior = MV_k − MV_{k-1}
3:
4:   # Si la diferencia con el frame anterior es positiva
5:   # nos fijamos si estábamos creciendo (pendienteDeMovimiento positivo) o no
6:   if Min(diferenciaFrameAnterior, 0) == 0 then
7:     if Min(pendienteDeMovimiento, 0) == 0 then
8:       # Si crecíamos lo seguimos haciendo incrementamos en 1 la variable
9:       pendienteDeMovimiento = pendienteDeMovimiento + 1
10:    else
11:      # Si no crecíamos comenzamos a hacerlo poniendo 1 en la variable
12:      pendienteDeMovimiento = 1
13:    end if
14:  end if
15:
16:  # Si la diferencia con el frame anterior es negativa
17:  # nos fijamos si estábamos decreciendo (pendienteDeMovimiento negativo) o no
18:  if Max(diferenciaFrameAnterior, 0) == 0 then
19:    # Si decrecíamos lo seguimos haciendo restamos en 1 la variable
20:    if Max(pendienteDeMovimiento, 0) == 0 then
21:      pendienteDeMovimiento = pendienteDeMovimiento − 1
22:    else
23:      # Si no decrecíamos comenzamos a hacerlo poniendo -1 en la variable
24:      pendienteDeMovimiento = −1
25:    end if
26:  end if
27: else
28:  # Si no hubo suficiente diferencia con el frame anterior
29:  # continuamos creciendo o decreciendo de acuerdo al estado actual pendienteMovimiento
30:  if Min(pendienteDeMovimiento, 0) == 0 then
31:    pendienteDeMovimiento = pendienteDeMovimiento + 1
32:  end if
33:  if Max(pendienteDeMovimiento, 0) == 0 then
34:    pendienteDeMovimiento = pendienteDeMovimiento − 1
35:  end if
36: end if
37: MV_{k-1} = MV_k
38: return pendienteDeMovimiento
```

7.2. Rechazo de intervalos

Una vez que se detecta un intervalo, a partir de los crecimientos y decrecimientos en la cantidad de movimiento, se analizan ciertas medidas para considerar si dicho intervalo corresponde con una potencial seña de Truco:

- **Umbrales de submovimiento y sobremovimiento:** Con estas cotas determinamos si la cantidad máxima de movimiento detectada en el intervalo se encuentra en cierto rango. Utilizando estos umbrales podemos descartar intervalos donde no haya suficiente proporción de movimiento y aquellos en los que se produce demasiado movimiento (rotaciones y traslaciones marcadas del rostro, ver imagen 7.2).

- **Cantidad de *frames* en el intervalo:** definimos en este caso una cota mínima y una cota máxima para la cantidad de *frames* que puede tener un intervalo. Esto nos permite descartar por un lado movimientos que se ejecutan muy rápido y por otro lado aquellos movimientos que se mantienen continuamente (por ejemplo una persona hablando).

Por otro lado, el método de cálculo de intervalos en conjunto con las reglas de rechazo imponen una restricción al sistema: los gestos deben ejecutarse secuencialmente y dejando un lapso de tiempo entre cada par de gestos consecutivos. Si no se sucede este lapso de tiempo entre un par de gestos sucesivos, no se provocará el decrecimiento esperado para la finalización del intervalo y ambos gestos quedarán unidos en un solo intervalo que se descartará por exceder la cantidad de *frames*. Un ejemplo de esta situación puede verse en el primer gesto que se efectúa en la figura 7.1. Si bien estas reglas pueden llevar a que algunos gestos no sean detectados, nuestro algoritmo de clasificación se beneficia porque evitamos clasificar imágenes de movimiento con desplazamientos correspondientes a dos gestos distintos.

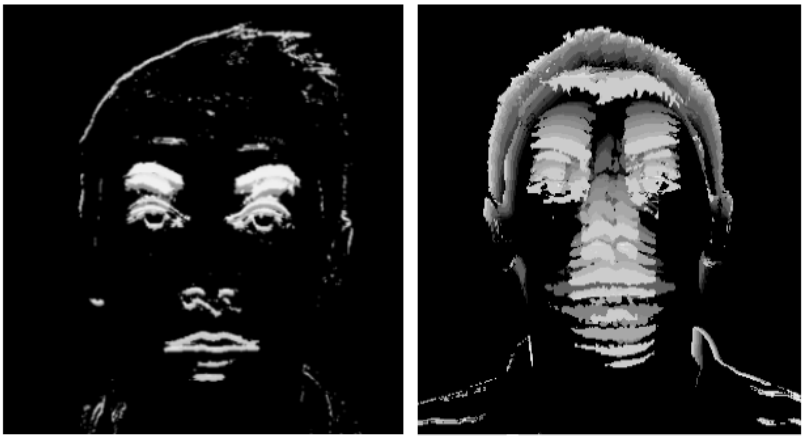

Figura 7.2: Movimiento generado por la seña del Ancho de Espada (izquierda) contra sobremovimiento generado por levantar la cabeza hacia arriba (derecha).

7.3. Integración del sistema

En las secciones anteriores presentamos un algoritmo para detectar el inicio y el fin de un intervalo de tiempo donde se está desarrollando un gesto. Contando con esta información, podemos determinar los momentos en los que se debe clasificar un conjunto de imágenes de movimiento, para determinar que gesto se está produciendo en las mismas.

Al momento de detectar el fin de un intervalo ya no podemos contar con la información de movimiento actualizada. Esto se debe a que, dado que analizamos decrecimientos para cerrar un intervalo, la cantidad de movimiento del *frame* procesado tiende a 0. Con lo cual enviar a clasificar las imágenes de movimiento de dicho *frame* no es adecuado. Para solucionar esto debemos ir manteniendo la información de alguna manera. En nuestro caso decidimos mantener las clasificaciones de las imágenes de movimiento que se van produciendo mientras un intervalo de tiempo se encuentra iniciado. Al momento que este finaliza tendremos una secuencia de clasificaciones para cada instante en que el intervalo se mantuvo abierto.

Con esta información se puede inferir el gesto que se está realizando de diferentes maneras. La primera estrategia elegida fue seleccionar la última clasificación que se realizó en el intervalo. Debido a que en algunos casos el final exacto del intervalo puede variar en algunos *frames*, esta estrategia producía algunos errores. En consecuencia se eligió tomar en cuenta las últimas clasificaciones realizadas y de las mismas seleccionar el gesto que más se repite. Esta estrategia produce mayor certeza sobre la clasificación y nos permite descartar aquellas clasificaciones que no sean tan confiables.

7.4. Conclusiones del capítulo

En este capítulo se explicó la forma de detectar los intervalos temporales que contienen gestos, durante el transcurso de una captura de video que puede contener varios intervalos. Para ello se especificó el concepto de proporción de movimiento a tener en cuenta para poder determinar aquellos intervalos que generen movimiento suficiente como para suponer que dentro del mismo se produjo un gesto. Finalmente se explicó como se produce la integración de este método con el resto del proceso definido en los capítulos anteriores.

Cabe destacar que para realizar este proceso se utilizaron los algoritmos para calcular imágenes de movimiento definidos en el capítulo 7. Particularmente se usaron imágenes de energía de movimiento (MEI) con una ventana de movimiento $\alpha = 10$ (a diferencia de las utilizadas en el capítulo 7 que tenía una valor de 25). Esto se hizo para lograr decrecimientos más rápidos en la proporción de movimiento al finalizar la realización de un gesto. Este parámetro debería ser adaptado si se desea utilizar el algoritmo en otros contextos.

Por otra parte, los umbrales VD y VC, también son sensibles a los diferentes contextos donde se aplique el algoritmo. Los mismos deberían ser adaptados teniendo en cuenta la separación temporal entre la ejecución de dos gestos consecutivos y la rapidez con que los mismo se realizan.

Capítulo 8

Experimentos

En este capítulo describimos los experimentos realizados y los resultados obtenidos sobre los algoritmos que componen nuestro método de detección y reconocimiento de gestos, con los objetivos de:

- Determinar los parámetros de los algoritmos para lograr la mejor performance.
- Evaluar la performance de cada uno de los algoritmos.
- Validar el funcionamiento del sistema en tiempo real.

8.1. Experimentos sobre parámetros de Motion History Image

Por lo definido en el capítulo 4 se deduce que el cálculo de las imágenes MHI y MEI está determinado por los parámetros α, β y μ. A continuación analizaremos el comportamiento de estos parámetros.

8.1.1. Parámetro α

Como lo definimos en la sección 4.3 el parámetro α indica cuántos de los últimos *frames* son considerados como recientes y tenidos en cuenta para calcular los *Templates Temporales*. Asumiendo que la duración de un gesto es menor a un 1 segundo podemos seleccionar un valor de α igual a la cantidad de *frames* por segundo que puede obtener nuestro dispositivo de captura. Dado que una cámara web estándar es capaz de capturar 25 o 30 *frames* por segundo el valor $\alpha = 30$ nos permitirá captar la ejecución de un gesto en forma completa. Genéricamente, el parámetro α queda definido como $\alpha = v$ donde v es la velocidad con que el dispositivo de captura obtiene las imágenes.

8.1.2. Parámetro β

Por lo explicado en la sección 4.3.1 este parámetro indica contra qué *frame* anterior se deberá comparar el *frame* actual, para determinar si se produjo movimiento. Este parámetro está relacionado con la rapidez con la que se realiza la ejecución del gesto. Un gesto que se efectúe rápidamente, producirá un cambio más brusco entre un *frame* y el siguiente. Por otro lado un gesto que se realice con más lentitud hará que los cambios entre un *frame* y los siguientes sean menos marcados.

En la figura 8.1 mostramos los resultados que se obtienen al ejecutar el cálculo de la MHI sobre una misma secuencia de imágenes, variando el valor del parámetro β. Podemos ver que para valores de β superiores se detecta una mayor proporción de movimiento.

Teóricamente el valor de β puede variar entre 1 y α. Las observaciones que se realizaron sobre una serie de casos, tomando $\alpha = 30$ permitieron determinar que la ventana de tiempo $\beta = 3$ es adecuada, para las capturas con que trabajamos. Para que el parámetro β se adapte a los diferentes dispositivos de captura y a la velocidad con que los mismos obtienen los frames, definimos $\beta = 0{,}1 * v$.

Figura 8.1: De izquierda a derecha se observan los resultados obtenidos del cálculo de la MHI sobre una secuencia de 25 *frames* para $\beta = 2, 3, 4$.

8.1.3. Parámetro μ

Como fue explicado en la sección 4.3.1, este parámetro actúa como umbral de filtrado. Intuitivamente si utilizamos valores muy bajos para dicho umbral estaríamos considerando mucho movimiento insignificante que podría agregar ruido al procesamiento total. Por otro lado, utilizar un umbral muy alto puede descartar información de movimiento que sea necesaria para su posterior procesamiento.

En la figura 8.2 mostramos los resultados que se obtuvieron en el cálculo de la MHI, variando los valores del parámetro de μ. Como puede observarse, con $\mu = 20$ observamos un grado de umbralización suficiente para poder trabajar con el movimiento de las zonas relevantes. Valores mucho mayores a 20 producen una perdida de información necesaria. Valores menores generan una excesiva cantidad de ruido. Pudimos observar que estos umbrales son sensibles a factores como la luminosidad, el contraste y la calidad de definición de las capturas, por lo que deberán ser adaptados al contexto en que sean utilizados.

Figura 8.2: De izquierda a derecha se observan los resultados obtenidos del calculo de la MHI sobre una secuencia de 25 *frames* para $\mu = 10, 15, 20, 25$.

8.2. Experimentos sobre reconocimiento de gestos

En este apartado evaluamos las diferentes combinaciones de métodos para representar movimiento, extraer características de las imágenes de movimientos y clasificarlas posteriormente para determinar qué gesto se está realizando. Se hicieron dos experimentos:

- **Experimento intra-base:** Se utilizó una estrategia de *cross-validation* para verificar como se comportan los algoritmos sobre una misma base de videos.
- **Experimento inter-base:** Se probaron las diferentes combinaciones de métodos, entrenando con una base de videos y testeando sobre la otra base.

El primero de los experimentos nos dio una idea de cuán bien se comporta el algoritmo si el individuo que realiza las señas es el mismo con el que se entrenó al sistema para que realice el reconocimiento de gestos. Además como los gestos los realiza una misma persona su ejecución es similar en cada uno de los ejemplos.

El segundo experimento nos permitió ver la eficacia del sistema cuando el individuo que realiza las señas no es el mismo que entrenó al sistema. Este experimento nos dio una idea de la robustez del sistema frente a las diferencias en la fisonomía del rostro del individuo que realiza las señas, los cambios de contexto, las variaciones en la ejecución del gesto, etc.

En ambos experimentos se analizaron todas las posibles combinaciones de métodos de representación de movimiento, extracción de características y clasificación de gestos que se presentan a continuación.

Métodos de representación de movimiento

Se utilizaron dos variantes para capturar y representar movimiento: imágenes de movimiento sin dirección e imágenes de movimiento direccionales. En la primera no se tiene en cuenta las direcciones de movimiento y sólo se genera una MHI obteniendo de la misma la MEI. En la segunda opción además de contar con la MHI y la MEI se generan las DMHIs correspondientes a las direcciones arriba, abajo, izquierda y derecha.

Métodos de extracción de características

En los experimentos se testearon las cinco formas definidas para segmentar el rostro de una persona: cuadrícula, facial, *quadtree* y las dos segmentaciones adicionales. La segmentación en cuadrícula se efectuó tomando los parámetros $filas = 5$ y $columnas = 4$. La propiedad extraída los primeros tres tipos de segmentación fue la proporción de movimiento mientras que para las segmentaciones adicionales se utilizó momentos geométricos.

Métodos de clasificación

Los métodos de clasificación utilizados fueron:

- Vecinos más Cercanos (KNN): se utilizaron las variantes de 3 y 5 vecinos.
- Máquinas de Vectores de Soporte (SVM): se utilizó la función kernel *Radial Basis Function (RBF)*. Asimismo se variaron los parámetros:
  - C en el intervalo $[2^{-5}, 2^{-3}, 2^{-1}, 2, 2^3, 2^5, 2^7, 2^9, 2^{11}, 2^{13}, 2^{15}]$
  - γ en el intervalo $[2^{-15}, 2^{-13}, 2^{-11}, 2^{-9}, 2^{-7}, 2^{-5}, 2^{-3}, 2^{-1}, 2]$

  Mediante validación cruzada (método *Leave One Out*) se obtuvo que los valores $C = 2^{11}$ y $\gamma = 2^{-7}$ eran los óptimos por presentar los mejores resultados.

8.2.1. Bases de Videos utilizadas

Para realizar estos experimentos se crearon dos bases de videos, cada una con una persona diferente (Gonzalo y Santiago), con las siguientes características:

	Base Gonzalo	Base Santiago
Ancho Basto	31	5
Ancho Espada	23	6
Ancho Falso	23	5
Dos	28	5
Siete Espada	30	4
Siete Oro	23	8
Tres	39	4
Total	197	32

Tabla 8.1: Comparación entre las bases de videos de señas creadas en este trabajo.

Cada una de las bases satisface las siguientes características:

- Cada video contiene sólo un gesto.
- Las capturas tienen un tamaño de 640 x 480 pixeles.
- Las vídeos tienen una frecuencia de captura de 25 *Frames Per Second(FPS)*.
- Los rostros dentro de las capturas tiene un tamaño aproximado de 100 x 100 pixeles.

8.2.2. Experimento intra-base

En este experimento se utiliza la estrategia de *Cross-Validation Leave One Out* para testear el comportamiento del sistema. Se realizó este experimento para ambas bases de videos (Gonzalo y Santiago). Los porcentajes de aciertos obtenidos pueden verse en las tablas 8.2 y 8.3.

Santiago		3-NN	5-NN	SVM
Motions sin Dirección	Cuadrícula	0.97	0.97	0.97
	Facial	0.97	0.92	0.92
	Quadtree	1.00	1.00	1.00
Motions con Dirección	Cuadrícula	1.00	1.00	1.00
	Facial	0.97	0.92	0.95
	Quadtree	1.00	1.00	1.00

Gonzalo		3-NN	5-NN	SVM
Motions sin Dirección	Cuadrícula	0.98	0.98	0.98
	Facial	1.00	0.99	0.99
	Quadtree	1.00	1.00	1.00
Motions con Dirección	Cuadrícula	0.99	0.99	0.99
	Facial	0.99	0.99	0.99
	Quadtree	1.00	1.00	1.00

Tabla 8.2: Resultados del test intra-base, mostrando el porcentaje de gestos correctamente clasificados. A la izquierda entrenando y testeando con la base Santiago. A la derecha entrenando y testeando con la base Gonzalo. El método de extracción se corresponde con Proporción de Movimiento (ver Sección 5.3.1).

Santiago		3-NN	5-NN	SVM
Motions sin Dirección	3-Regiones	0.70	0.73	0.78
	4-Regiones	0.76	0.78	0.86
Motions con Dirección	3-Regiones	0.76	0.68	0.76
	4-Regiones	0.78	0.81	0.89

Gonzalo		3-NN	5-NN	SVM
Motions sin Dirección	3-Regiones	0.91	0.92	0.96
	4-Regiones	0.91	0.91	0.95
Motions con Dirección	3-Regiones	0.95	0.95	0.99
	4-Regiones	0.90	0.93	0.94

Tabla 8.3: Resultados del test intra-base, mostrando el porcentaje de gestos correctamente clasificados. A la izquierda entrenando y testeando con la base Santiago. A la derecha entrenando y testeando con la base Gonzalo. El método de extracción se corresponde con Momentos Geométricos (ver Sección 5.3.2).

Este experimento deja en claro que la clasificación de gestos, utilizando como método de extracción la proporción de movimiento, es altamente robusta cuando el individuo con el que se testea el sistema es el mismo que realizó el entrenamiento. Con la utilización de momentos geométricos el porcentaje de aciertos disminuye en muchos casos de manera considerable, lo cual no lo hace aplicable a cualquier base de entrenamiento.

8.2.3. Experimento inter-base

En este caso entrenamos al sistema con una de las bases de videos y testeamos con la otra. Luego se invirtieron las bases de entrenamiento y testeo y se realizaron nuevamente los experimentos. En las tablas 8.4 y 8.5 se pueden observar los resultados obtenidos.

Santiago vs Gonzalo		3-NN	5-NN	SVM
Motions sin Dirección	Cuadrícula	0.85	0.84	0.85
	Facial	0.84	0.83	0.84
	Quadtree	0.83	0.84	0.91
Motions con Dirección	Cuadrícula	0.91	0.88	0.92
	Facial	0.92	0.94	0.97
	Quadtree	0.91	0.92	0.95

Gonzalo vs Santiago		3-NN	5-NN	SVM
Motions sin Dirección	Cuadrícula	0.89	0.89	0.89
	Facial	0.91	0.91	0.91
	Quadtree	0.78	0.86	0.76
Motions con Dirección	Cuadrícula	0.91	0.91	0.89
	Facial	0.95	0.95	0.94
	Quadtree	0.81	0.84	0.84

Tabla 8.4: Resultados del test inter-base, mostrando el porcentaje de gestos correctamente clasificados. A la izquierda entrenando con la base Santiago y testeando con la base Gonzalo, a la derecha entrenando con la base Gonzalo y testeando sobre la base Santiago. El método de extracción se corresponde con Cantidad de Movimiento 5.3.1.

Santiago vs Gonzalo		3-NN	5-NN	SVM
Motions sin Dirección	3-Regiones	0.52	0.51	0.52
	4-Regiones	0.56	0.53	0.57
Motions con Dirección	3-Regiones	0.45	0.51	0.66
	4-Regiones	0.58	0.56	0.59

Gonzalo vs Santiago		3-NN	5-NN	SVM
Motions sin Dirección	3-Regiones	0.73	0.73	0.75
	4-Regiones	0.65	0.64	0.73
Motions con Dirección	3-Regiones	0.62	0.62	0.78
	4-Regiones	0.70	0.73	0.67

Tabla 8.5: Resultados del test inter-base, mostrando el porcentaje de gestos correctamente clasificados. A la izquierda entrenando con la base Santiago y testeando con la base Gonzalo, a la derecha entrenando con la base Gonzalo y testeando sobre la base Santiago. El método de extracción se corresponde con Momentos Geométricos 5.3.2.

Como se observa, la tasa de aciertos disminuye en relación con el experimento intra-base. Analizando las capturas y las imágenes de movimiento que las mismas generan pudimos determinar que la principal causa de esta baja en los aciertos se debe a la variación en la forma de realización de los gestos. También se puede obtener indicios de que la fisonomía de la cara afecta escasamente a la generación de estas imágenes.

Por otro lado, comparando los métodos de extracción mediante proporción de movimiento en relación a la extracción por momentos geométricos, se observa que los resultados de los primeros son muy superiores a los últimos. Dado esto, podemos inferir que el método basado en momentos no define buenos descriptores de movimiento. Esto se debe a que la forma que generan los movimientos en un rostro difiere de acuerdo al individuo que produce el gesto e incluso en gestos iguales producidos por un mismo individuo. El movimiento facial no genera formas fácilmente comparables lo que hace inútil utilizar un mecanismo basado en formas como método de extracción de características. En [11] se utiliza una forma de extracción basada en momentos para el reconocimiento de movimientos corporales humanos que suelen ser mucho más definidos y comparables entre sí. Dados estos resultados descartamos para posteriores análisis la extracción de características basada en momentos.

8.2.4. Selección de métodos

Para seleccionar la mejor combinación de métodos promediamos los resultados que se obtuvieron en cada uno de los experimentos y seleccionamos el que mayor tasa de aciertos tiene en promedio (ver tabla 8.6). Por lo explicado anteriormente solo tomamos en cuenta la extracción de características por proporción de movimiento.

Promedios		3-NN	5-NN	SVM
Motions sin Dirección	Cuadrícula	0.9225	0.0.92	0.9225
	Facial	0.93	0.9125	0.915
	Quadtree	0.9025	0.925	0.9175
Motions con Dirección	Cuadrícula	0.9525	0.945	0.95
	Facial	0.9575	0.95	0.9625
	Quadtree	0.93	0.94	0.9475

Tabla 8.6: Promedio de la tasa de aciertos teniendo en cuenta los resultados de ambos experimentos (intra-base e inter-base).

De ambos experimentos se dedujo que la combinación de métodos que produce una mejor tasa de reconocimiento está dada por:

- Representación de movimiento: *Motions* Direccionales.
- Extracción de características: Segmentación Facial extrayendo la proporción de movimiento de cada región.
- Algoritmo de clasificación: Máquina de Vectores de Soporte (SVM).

La matriz de confusión que se obtuvo de los experimentos inter base para la combinación de métodos óptima, puede verse en las tablas 8.7 y 8.8.

	Ancho Espada	Siete Oro	Ancho Falso	Dos	Siete Espada	Ancho Basto	Tres
Ancho Espada	23	0	0	0	0	0	0
Siete Oro	0	20	0	1	0	0	2
Ancho Falso	0	0	21	2	0	0	0
Dos	0	0	0	28	0	0	0
Siete Espada	0	0	0	0	30	0	0
Ancho Basto	0	0	0	0	0	31	0
Tres	0	0	0	0	0	0	39

Tabla 8.7: Matriz de confusión de los resultados de clasificación, utilizando la combinación óptima de métodos, entrenando con la base Santiago y testeando con la base Gonzalo.

	Ancho Espada	Siete Oro	Ancho Falso	Dos	Siete Espada	Ancho Basto	Tres
Ancho Espada	6	0	0	0	0	0	0
Siete Oro	0	8	0	0	0	0	0
Ancho Falso	0	0	5	0	0	0	0
Dos	0	0	1	4	0	0	0
Siete Espada	0	0	0	0	4	0	0
Ancho Basto	0	0	0	0	0	5	0
Tres	0	0	1	0	0	0	3

Tabla 8.8: Matriz de confusión de los resultados de clasificación, utilizando la combinación óptima de métodos, entrenando con la base Gonzalo y testeando con la base Santiago.

8.3. Comparación de algoritmos de rechazo

En esta sección vamos a analizar el comportamiento de los algoritmos de rechazo propuestos en la sección 6.4. Para realizar este análisis vamos a utilizar curvas de ROC (*Receiver Operating Characteristic*, [42]) para observar el comportamiento de los clasificadores binarios a medida que se varía el umbral de discriminación (valor a partir del cual decidimos que un caso es una seña).

Una curva ROC es la representación gráfica de la tasa de verdaderos positivos o sensibilidad (probabilidad de detectar correctamente una seña cuando dicha seña está efectivamente presente y una no seña cuando la misma no esté presente) frente a la tasa de falsos positivos o 1-especificidad (probabilidad de detectar una seña cuando ésta no está presente y una no seña cuando una seña está presente) según se varía el umbral de discriminación. Un ejemplo de esto puede verse en la figura 8.3.

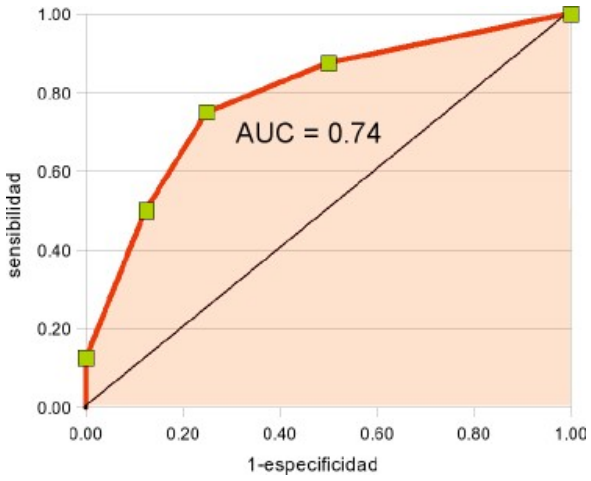

Figura 8.3: Ejemplo de una curva de ROC, indicando el área bajo la curva (AUC).

Cada resultado de predicción de un clasificador binario, para un determinado umbral, representa un punto en el espacio ROC. El mejor método posible de predicción se situaría en un punto en la esquina superior izquierda, o coordenada $(0, 1)$ del espacio ROC, representando un 100 % de sensibilidad (ningún falso negativo) y un 100 % de especificidad (ningún falso positivo). A este punto $(0, 1)$ también se le llama una clasificación perfecta. Por el contrario, una clasificación totalmente aleatoria (o adivinación aleatoria) daría un punto a lo largo de la línea diagonal, que se llama también línea de no-discriminación. Los puntos por encima de la diagonal representan los buenos resultados de clasificación (mejor que el azar), puntos por debajo de la línea de los resultados pobres (peor que al azar).

Para comparar dos clasificadores debemos reducir la información contenida en la curva de ROC a un valor escalar que represente la performance esperada. Para realizar esto comúnmente se utiliza el área bajo la curva (AUC, *Area Under Curve*, [43, 44]). Dado que el área bajo la curva es una porción del área del cuadrado de lado 1, su valor varia entre 0 y 1. Una representación gráfica del área bajo la curva también puede verse en la figura 8.3.

A continuación se muestran las curvas de ROC y el área bajo la curva para las diferentes formas de generar el clasificador binario de rechazo. Cada uno de los métodos fue probado sobre el conjunto de señas y no señas de las base Santiago y Gonzalo. Para las bases de señas se utilizaron las mismas definidas en la tabla 8.1 y adicionalmente se incorporaron bases de datos de no señas como sacar la lengua, inflar cada uno de los cachetes, etc. Los datos de las bases son los que se muestran a continuación:

	Base Gonzalo	Base Santiago
Señas	197	32
No Señas	30	42
Total	227	74

Tabla 8.9: Comparación entre las bases de videos de señas y no señas creadas en este trabajo.

Para representar movimiento se utilizaron MHI y DMHI de cuatro direcciones.

SVM de una clase *(One Class SVM)*

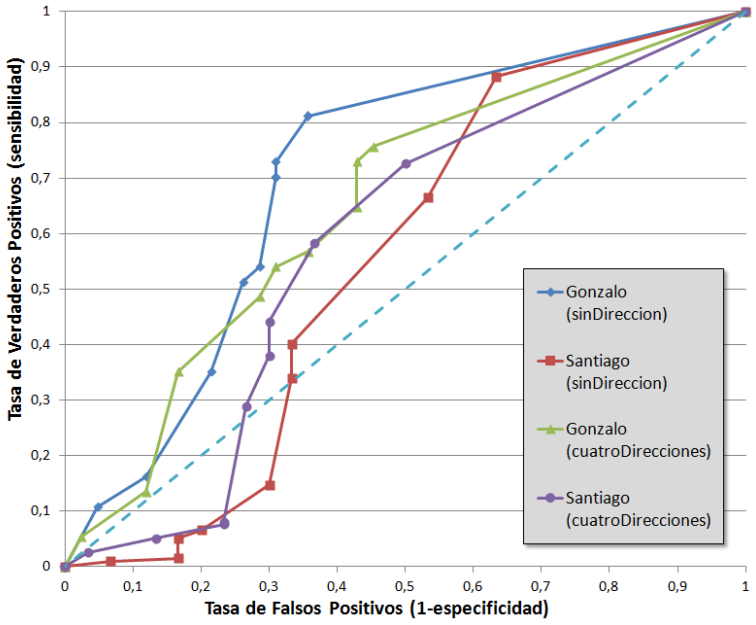

Figura 8.4: Curva de ROC para el método de SVM de una clase.

Área Bajo la Curva	Gonzalo	Santiago
Motions sin Dirección 5	0.703346126	0.55160771
Motions con 4 Direcciones	0.651866118	0.580879644

Probabilidad máxima

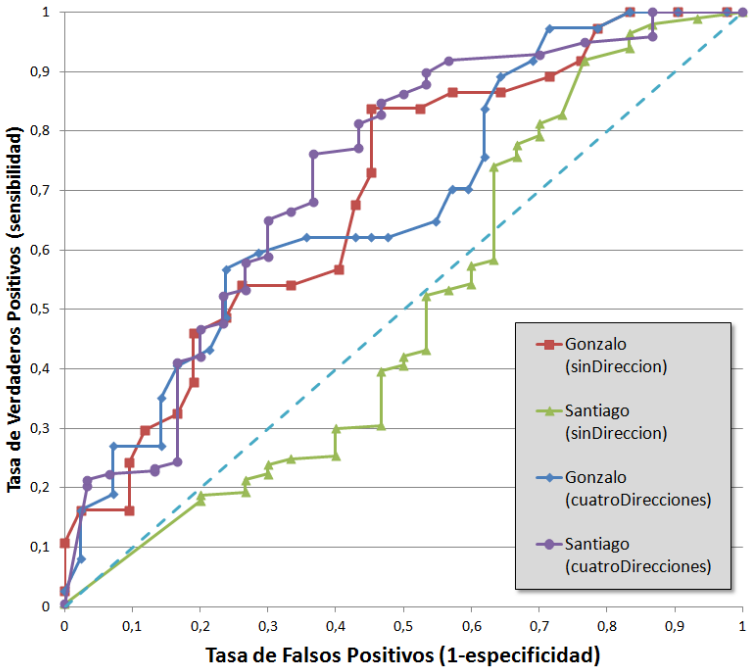

Figura 8.5: Curva de ROC para el método de umbralización por probabilidad máxima.

Área Bajo la Curva	Gonzalo	Santiago
Motions sin Dirección	0.669240669	0.499492386
Motions con 4 Direcciones	0.583655084	0.725549915

Cociente de las dos mayores probabilidades

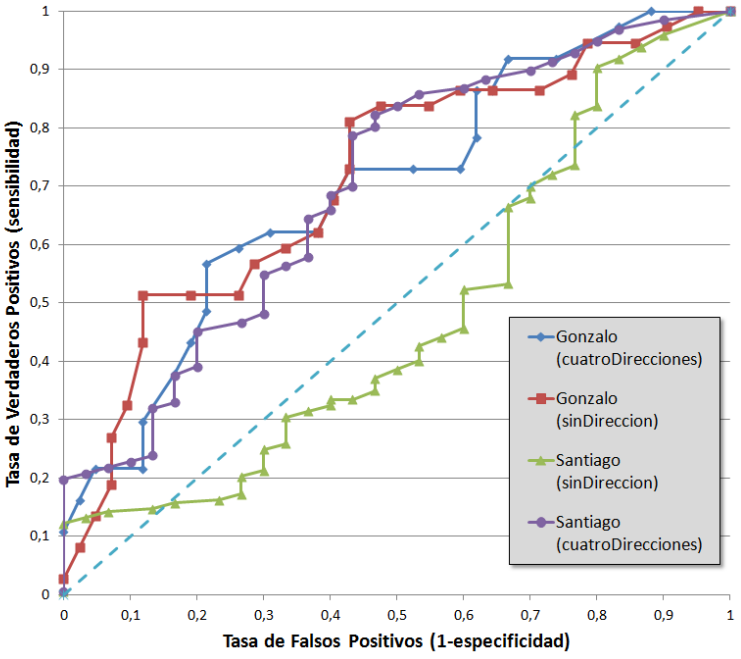

Figura 8.6: Curva de ROC para el método de umbralización por cociente de las dos mayores probabilidades.

Área Bajo la Curva	Gonzalo	Santiago
Motions sin Dirección	0.712998713	0.478087986
Motions con 4 Direcciones	0.696267696	0.698646362

SVM binario

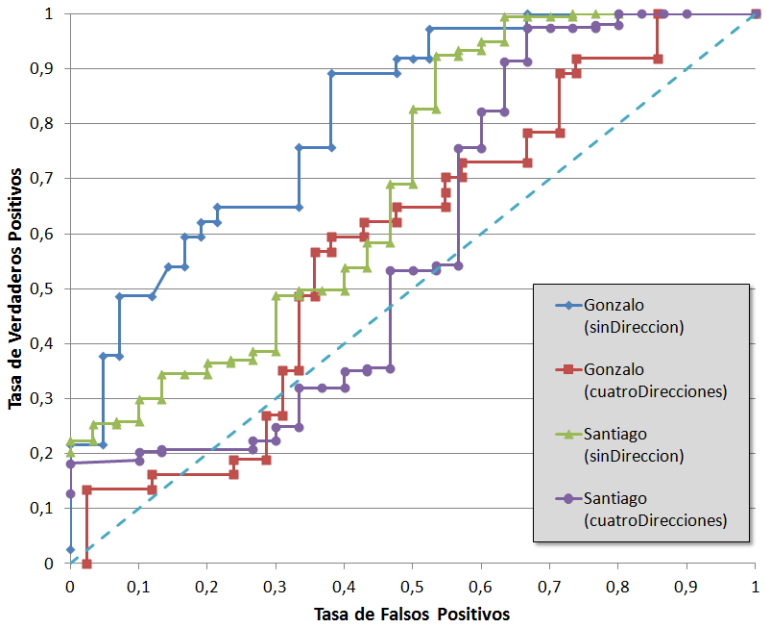

Figura 8.7: Curva de ROC para el método de SVM binario.

Área Bajo la Curva	Gonzalo	Santiago
Motions sin Dirección	0.810810811	0.692047377
Motions con 4 Direcciones	0.582368082	0.580456853

Como se puede observar en las gráficas de las curvas de ROC y en las mediciones del área bajo la curva, ninguno de los métodos implementados logra alcanzar un desempeño aceptable. Esto implica que el método de rechazo no sólo no logra comportarse acordemente en su tarea de rechazar gestos sino que influye de manera negativa en la detección de las señas de Truco disminuyendo la tasa de aciertos. De aquí en más se pueden considerar dos alternativas. Por un lado se puede incluir el algoritmo de rechazo generando un decrecimiento en la tasa de acierto de señas de Truco correctas. Por otro lado puede considerarse la idea de no incluir el algoritmo de rechazo en la solución final y sólo utilizar el rechazo de intervalos del algoritmo de detección de segmentos temporales presentado en 7.2. De esta manera, aquellos gestos no considerados señas del Truco pero cuyo movimiento facial se asemeje, serán considerados como válidos para la clasificación.

8.4. Performance

Para los experimentos de performance se realizaron 7 capturas de video, una para cada gesto, donde un individuo realizaba el gesto correspondiente una cierta cantidad de veces. Todas las capturas tienen un tamaño de 640x480 pixeles y fueron obtenidas a una velocidad de 30 *frames* por segundo. Las demás características de las capturas pueden verse en la tabla 8.10.

Los experimentos fueron realizados en una computadora de escritorio con las siguientes características: Procesador AMD Phenom II X4 64bits de 4 núcleos a 3.20Ghz, Memoria RAM de 4,00 GB.

Gesto	Cantidad de *Frames*	Cantidad de gestos
Ancho Basto	582	10
Ancho Espada	474	8
Ancho Falso	663	12
Dos	581	9
Siete Espada	528	8
Siete Oro	555	10
Tres	456	8

Tabla 8.10: Carácteristicas de las capturas utilizadas para los test de performance

8.4.1. Tiempos variando el tamaño de los *frames*

El tamaño de un *frame* tiene una incidencia directa en los tiempos del algoritmo, dado que muchas de las operaciones que se realizan se aplican a cada uno de los pixeles del mismo. Teniendo en cuenta esto, nos preocupamos por ver cómo evolucionan los tiempos variando el tamaño del *frame* de la captura. Tomamos mediciones de tiempos para el sistema teniendo en cuenta una representación de movimiento sin dirección y con cuatro direcciones.

La figura 8.8 muestra la cantidad de *frames* que puede procesar nuestro sistema en función del tamaño de los mismos. Para realizar este experimento utilizamos el sistema en su configuración óptima (ver sección 8.2.4). Se puede observar que si reducimos el tamaño de los *frames* podemos procesar una mayor cantidad en igual cantidad de tiempo. Para el caso de imágenes de movimiento sin dirección el sistema es capaz de procesar más de 30 frames a cualquier escala, lo que permite procesar todos los frames que obtiene un dispositivo de captura convencional. En el caso de tener en cuenta 4 direcciones, a partir de un tamaño 480x360 pixeles (75 % del tamaño default) el sistema es capaz de procesar 30 *frames* por segundo. Se observa que en ambos casos y en cualquier escala el algoritmo logra procesar más de 15 *frames* por segundo lo que permite lograr el funcionamiento en tiempo real.

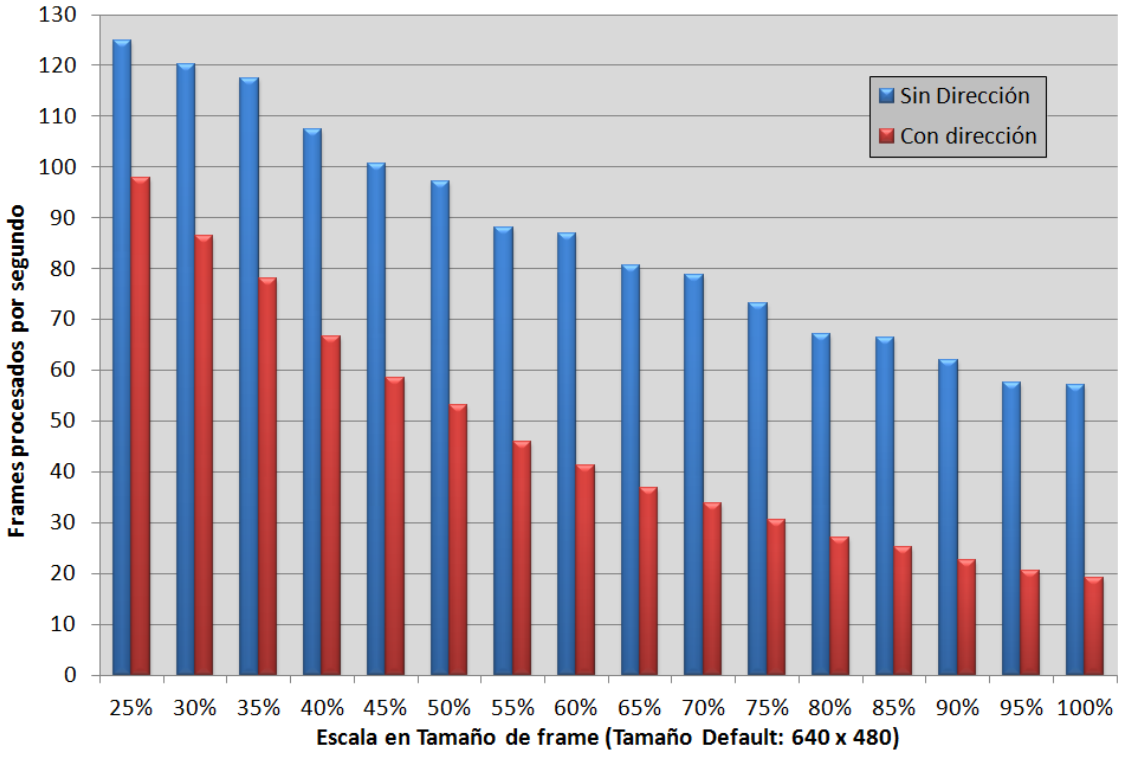

Figura 8.8: Cantidad de *frames* procesados por segundo en función de la escala en el tamaño del *frame*.

8.4.2. Tiempos del algoritmo

El segundo test de performance realizado consiste en medir los tiempos de ejecución de cada método del algoritmo completo, utilizando representación de movimiento con direcciones y sin direcciones. Los métodos fueron divididos en las siguientes categorías:

- Capturar *Frame*: Agrupa los métodos para obtener un *frame* de una captura de video.
- Trackeo de Rostro: Métodos para la detección y seguimiento de un rostro en un *frame*.
- Calcular Imágenes de Movimiento: Contiene los métodos para actualizar tanto el MHI como los DMHI (si corresponde), incluyendo el cálculo de los pixeles en los que se detecta movimiento y las direcciones de movimiento.
- Actualizar Detector de Intervalos: Incluye los métodos para determinar el inicio y el fin de un intervalo en el que se está produciendo un gesto.
- Clasificar: Reúne los métodos para segmentar y extraer características de una imagen de historia de movimiento, así como la clasificación de estas características y el rechazo de posibles no señas.

Para cada uno de los videos de ejemplo se calcularon los tiempos. Luego obtuvimos el porcentaje de incidencia de cada método sobre la duración total de la ejecución del algoritmo de detección y reconocimiento de gestos. Estos porcentajes se promediaron para evitar posibles errores de medición y con esto definimos la incidencia media de cada método sobre el algoritmo completo. Los resultados alcanzados utilizando representación de movimiento en 4 direcciones pueden verse en la figura 8.9.

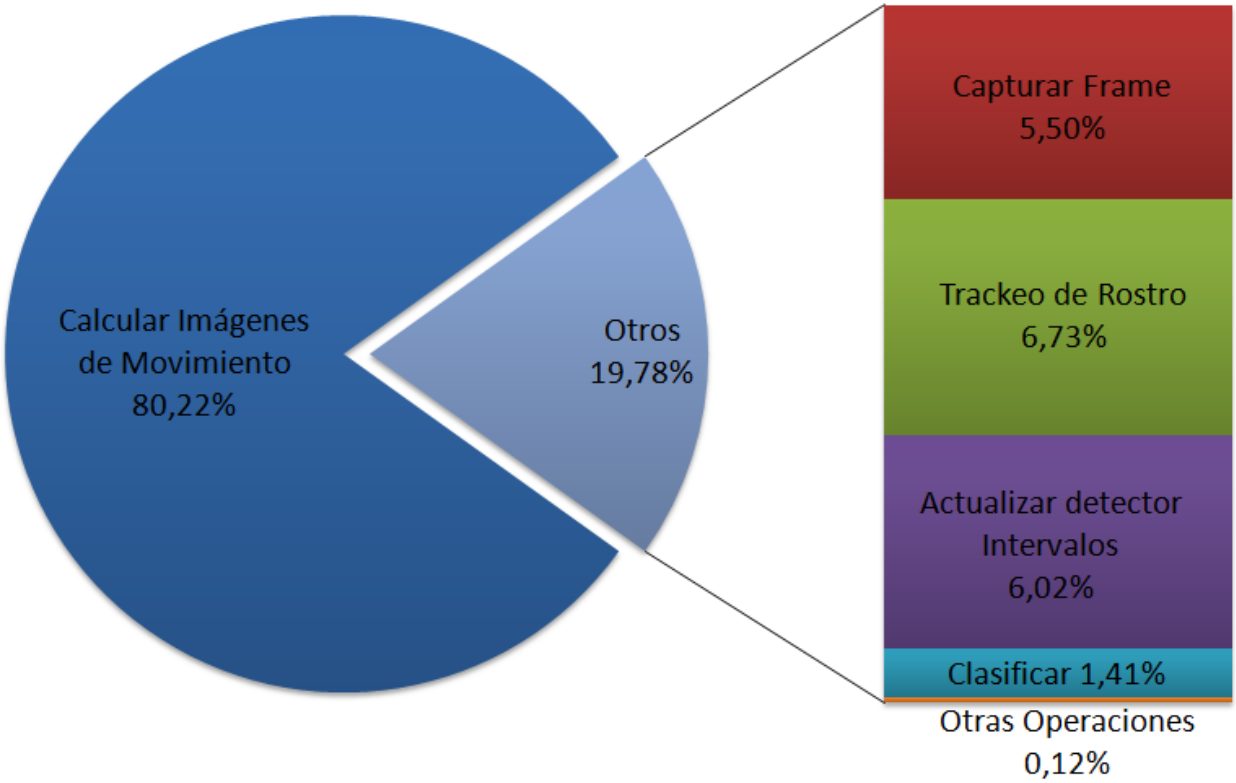

Figura 8.9: Incidencia de cada método sobre los tiempos del algoritmo completo.

Como se ve en la figura 8.9, la actualización de las imágenes de historia de movimiento es el método que mayor incidencia tiene sobre el sistema. Un análisis de cómo este método distribuye sus tiempos puede verse en la figura 8.10.

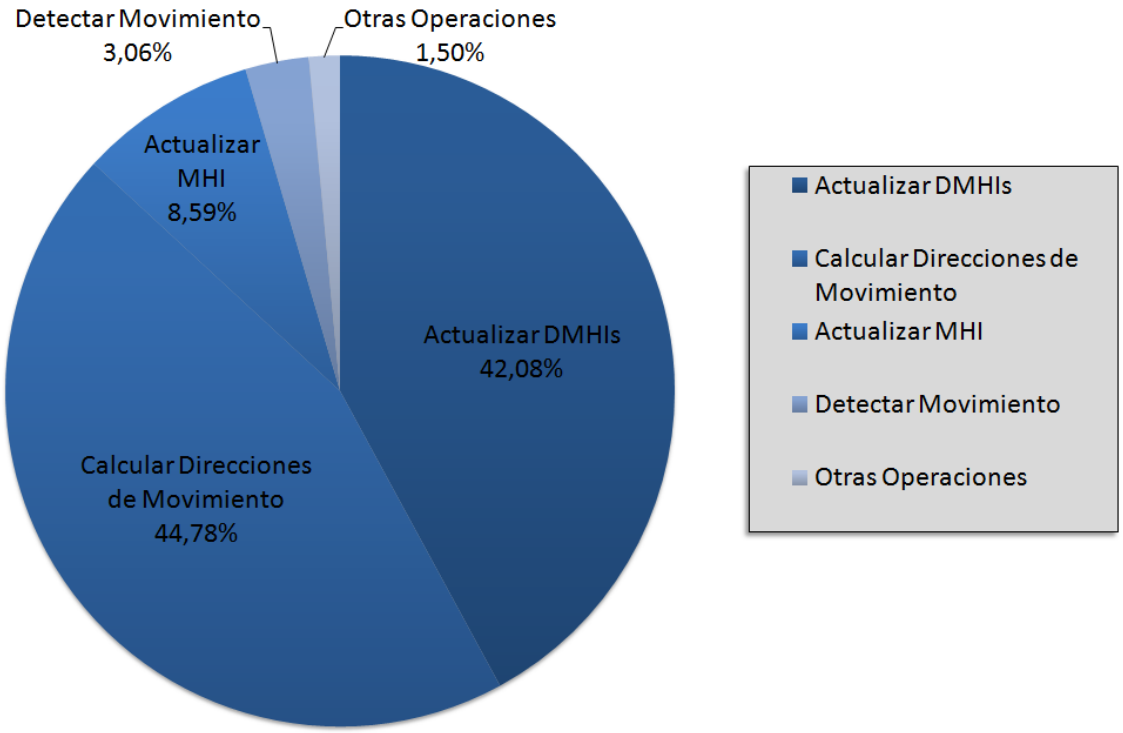

Figura 8.10: Distribución de los tiempos de incidencia en el cálculo de imágenes de historia de movimiento.

El mismo experimento se realizó para el sistema funcionando con imágenes de movimiento sin dirección. Con esto queríamos evaluar cómo cambian los tiempos al variar la forma de representación de movimiento. Los porcentajes de incidencia calculados pueden verse en la figura 8.11. Como se observa en el gráfico la incidencia de la tarea de cálculo de imágenes de movimiento es mucho menor en esto caso, lo que nos permite reducir sustancialmente el tiempo de proceso de un *frame*.

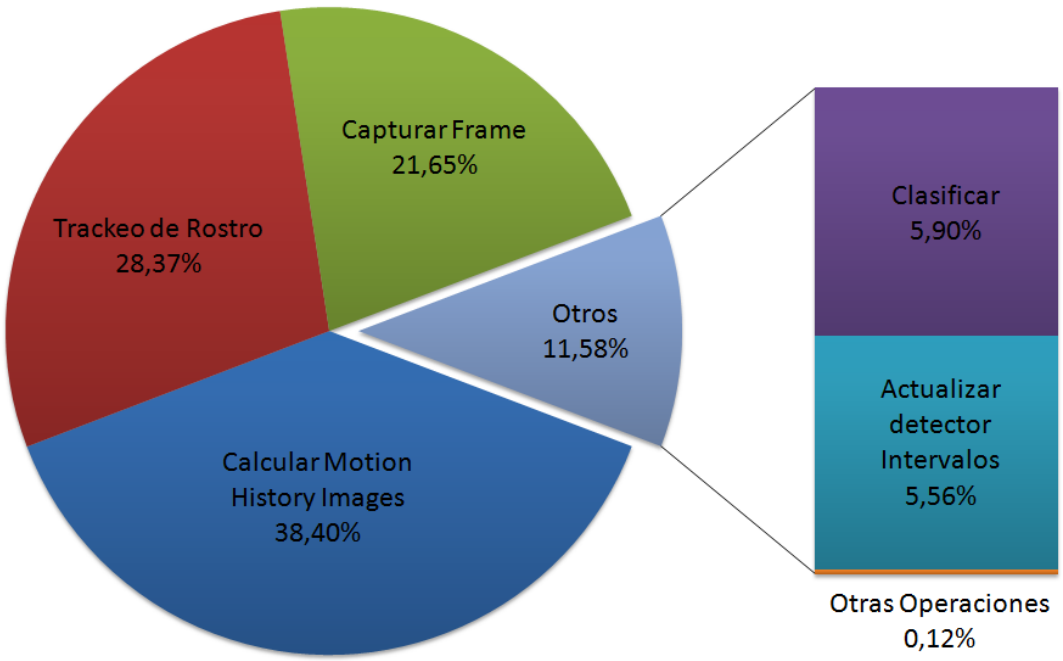

Figura 8.11: Incidencia de cada método sobre los tiempos del algoritmo completo, utilizando imágenes de movimiento sin dirección.

Capítulo 9

Conclusiones Finales

En este capítulo se comentan una serie de conclusiones finales de todo el trabajo realizado. En la sección 9.1 se exponen las limitaciones que se encontraron en el sistema implementado. En la sección 9.2 se resumen los objetivos alcanzados. Finalmente, en 9.3 se proponen algunas ideas consideradas como desarrollo posterior al sistema implementado, tanto para la mejora como para la extensión del mismo, en pos de su integración con otros sistemas.

9.1. Limitaciones del sistema desarrollado

Durante el desarrollo de esta tesis nos enfrentamos a varios problemas y limitaciones. Fuimos capaces de encontrar soluciones para mitigar la mayor parte de ellas, sin embargo, algunas de las mismas no pudieron ser solucionadas. Creemos convenientes describirlas en este apartado con el objetivo de que sean consideradas, mitigadas o solucionadas en trabajos futuros.

9.1.1. Reconocimiento de gestos mediante *Templates Temporales*

Trabajar con la técnica de representación de movimiento basado en *Templates Temporales* nos permitió poder caracterizar correctamente las señas del juego del Truco. Sin embargo este método tiene limitaciones propias que nos impidieron satisfacer otros objetivos.

Cuando se utiliza una máscara de movimiento (MHI, MEI), solamente se está considerando dónde el mismo ocurre y cómo el mismo evoluciona en una sola imagen en dos dimensiones. Esto acarrea problemas al querer diferencia aquellos gestos que se producen en zonas similares dentro del rostro. En nuestro caso inicialmente tuvimos dificultades para poder diferenciar correctamente las señas del Dos, el Tres o el Ancho Falso ya que las mismas ocurren en la zona de la boca. Para ello debimos encontrar formas de segmentar y extraer características que nos permitieran clasificar correctamente estos gestos. Una mejora introducida para reducir los efectos de este problema fue la utilización de MHI direccionales, lo que permitió obtener una mejor diferenciación de los mismos.

Sin embargo esta técnica no pudo solucionar el problema de rechazo de gestos que no son considerados señas del Truco. Existen una gran cantidad de gestos cuya imagen de movimiento es similar a la de alguna de las señas del Truco. Esto se debe a que dicho movimiento ocurre en las mismas zonas en la que ocurren las señas con lo cual en ciertos casos hace imposible diferenciar dos MHI's, consideradas las principales entradas de los algoritmos en esta tesis. Con una imagen de movimiento de tres dimensiones es más factible encontrar esas diferencias. Sin embargo, en este trabajo no se hizo uso de ese tipo de técnicas y la MHI traduce toda la información del movimiento en una imagen bidimencional. Por otro lado el rechazo de gestos no es un tema comúnmente mencionado en los trabajos anteriores relacionados a esta tesis, con lo cual los algoritmos propuestos, si bien resultaron novedosos, no fueron suficientes para lograr buenos resultados, es decir, no se logró una tasa de rechazo aceptable de gestos que no son señas de truco (alrededor del 80 %). Sin embargo, concluimos en que esto no sucede necesariamente por los algoritmos de rechazo mencionados sino por la técnica implementada para reconocer movimiento.

9.1.2. Algorimo de detección de intervalos temporales

Al trabajar con imágenes de movimiento sólo pudimos considerar esta información para poder determinar los intervalos en los que ocurre un movimiento significativo considerado una potencial seña del Truco.

Uno de los inconvenientes encontrados se produce debido a que las imágenes de movimiento consideran lo ocurrido en una cantidad de *frames* precedentes lo que dificulta en un determinado instante tener la información certera de si se está o no produciendo movimiento. En un ejemplo sencillo, se considera la ausencia de movimiento cuando la cantidad de pixeles en una MHI o MEI tienden a ser negros en su mayoría. Sin embargo como se comentó anteriormente para que ello suceda deben haber pasado una cantidad de *frames* en los que no ocurrió movimiento significativo. En particular nos interesa conocer el primero de esos *frames*, lo que indicaría que en ese instante dejó de haber movimiento. Mediante el algoritmo de detección implementado logramos estimar adecuadamente el principio y el final de intervalos de movimiento. Sin embargo existe la limitación de que dos gestos deban generarse de manera espaciada en el tiempo, de otro modo el intervalo de movimiento consideraría ambos gestos como uno único.

Por otro lado gestos involuntarios o situaciones de la vida real como parpadear o hablar también pueden agregar ruido al algoritmo de detección considerando este tipo de gestos como potenciales señas del Truco.

9.1.3. Lenguaje utilizado

Para el desarrollo del sistema utilizamos *Python* porque lo consideramos acorde a un trabajo basado fuertemente en la experimentación y el cambio. *Python* es un lenguaje simple, que permite realizar cambios en el código de manera muy cómoda y veloz lo que lo hace fuertemente útil en el desarrollo de una tesis experimental en donde el cambio y las pruebas son constantes. Sin embargo la gran desventaja es que es un lenguaje cuya *performance* computacional no está al nivel de lenguajes altamente optimizados (C++, C). Esto atenta con el requerimiento de funcionar en tiempo real del sistema. Sin embargo trabajar con *Python* nos permitió detectar cuáles son las etapas de nuestro algoritmo que consumen mayor tiempo de cómputo, haciendo factible que puedan ser reimplementadas e integradas fácilmente en nuestro sistema, incluyendo la posibilidad de implementarlas en otros lenguajes.

9.1.4. Sensibilidad en los parámetros seleccionados

Diversos parámetros fueron utilizados en nuestro sistema con el fin de adaptar los algoritmos a nuestro contexto particular. Entre ellos se incluyen:

- el parámetro de *proporción de aumento* de la ventana de búsqueda para la detección de rostros;
- los parámetros α, β y μ para la generación de imágenes de movimiento;
- la cantidad de vecinos y los parámetros C y γ para los algoritmos de clasificación (K-NN y SVM, respectivamente)

Los valores de estos parámetros en nuestro sistema fueron estudiados y definidos mediante experimentos o pruebas. Debido a esto, puede que los mismos tengan que ser adaptados si se los utiliza en otros contextos, por ejemplo, otras bases de datos, otras cámaras, etc.

9.1.5. Restricciones definidas

Como se comentó en el capítulo 1 propusimos una serie de restricciones para que el sistema funcione de manera adecuada. En principio, es necesario una cámara con una buena resolución, requisito que no toda cámara web suele cumplir. Por otro lado, los requerimientos de *hardware* si bien no son estrictos, tampoco son comunes en muchos hogares. La iluminación, el buen contraste, así como el tamaño del rostro en la captura considerando poca rotación del mismo, también se convierten en limitaciones. Si bien consideramos movimientos de la vida real en el algoritmo de detección de intervalos los mismos pueden introducir muchos de los errores descriptos en 9.1.2.

9.2. Objetivos Alcanzados

En el presente trabajo se implementó un sistema para detectar las señas del juego del Truco a partir de capturas de video en tiempo real. Para ello se descompuso el sistema en módulos independientes, lo que permitió evaluar distintas alternativas en cada uno de los mismos, a fin de lograr una tasa de aciertos satisfactoria en el reconocimiento de las señas.

Mediante la experimentación pudimos combinar las variantes de cada módulo presentando los resultados del proceso general y seleccionando la combinación que maximizó los mismos. Además se destacan los siguientes logros adicionales:

- El algoritmo funciona en tiempo real en capturas de video de tamaño considerable (640x480 pixeles), con requerimientos aceptables para una computadora de escritorio.
- Puede intuirse que los algoritmos son lo suficientemente independientes de la persona que ejecuta una seña, lo que permite la utilización del sistema sin la necesidad de entrenarlo previamente con bases de datos de dicha persona. Sin embargo esta premisa sólo está basada en las bases de datos de dos personas distintas lo cual nos impide generalizarla.
- La cantidad de elementos de la base de datos con la que se entrena el sistema puede no ser un determinante. Se pudo apreciar que con dos bases de datos con cantidades de videos de entrenamiento restringidas, los resultados fueron satisfactorios y similares. Nuevamente no podemos generalizar este comportamiento a partir de los experimentos realizados.

Por último, es importante destacar que esta implementación es la primera que integra la detección de expresiones en tiempo real al juego del Truco. Esto permite de aquí en más proponer mejoras y adaptaciones para poder utilizarlo con diversas integraciones.

9.3. Trabajo Futuro

El primer objetivo considerado para un trabajo futuro consiste mitigar las limitaciones presentadas en 9.1. En particular tomamos en cuenta los siguientes items como disparadores para investigaciones futuras:

- *Detección de movimiento en tres dimensiones:* es de esperar que técnicas que permitan detectar movimiento en tres dimensiones mejoren los resultados referidos al reconocimiento de los gestos, al agregar una nueva componente de diferenciación: la profundidad. Una manera de lograr esto consiste en la utilización de dos cámaras.
- *Cálculo de orientación de movimiento:* en nuestro trabajo propusimos una forma de calcular la orientación del movimiento para realizar el cálculo de las MHI's direccionales. Sin embargo existen otras que mejoran los resultados pero atentan contra el funcionamiento en tiempo real del sistema. Nuestro objetivo entonces es encontrar un punto medio que permita mejorar este cálculo minimizando el costo computacional.
- *Introducción de sonido:* mediante el reconocimiento de sonido evitaríamos las confusiones que genera una persona al hablar entre otras cuestiones. Por ejemplo descartando intervalos en donde el sonido fuese significativo.
- *Lenguaje:* como comentamos anteriormente *Python* es un lenguaje que no se caracteriza por ser extremadamente eficiente en términos computacionales. Sin embargo las librerías utilizadas también están diseñadas para integrarse con código C++, con lo cual optimizar determinadas partes del sistema no resulta una tarea complicada. Esto nos permitiría aprovechar todas las ventajas en cuanto a velocidad de cálculo que tiene un lenguaje como C++, para experimentar varias alternativas en los algoritmos utilizados.

Finalmente pretendemos integrar nuestro sistema a un software de juego de Truco, de forma tal que permita a una computadora jugar como un participante más. El trabajo presentado le permitiría a la computadora conocer las señas de su compañero mediante una cámara para posteriormente evaluar alternativas de juego. Existen muchas implementaciones del juego del Truco, con lo cual la integración con nuestro sistema introduciría la novedad de reemplazar completamente a un jugador por una computadora. No sería necesario interactuar de otra manera que no sea mediante los gestos y la utilización de la voz (como se mencionó en el tercer ítem) para coordinar los movimientos, las estrategias y el orden de juego.

Apéndices

Ejecución del cálculo de la MHI

Figura 1: Extracción del área de la cara detectada en cada uno de los *frames*.

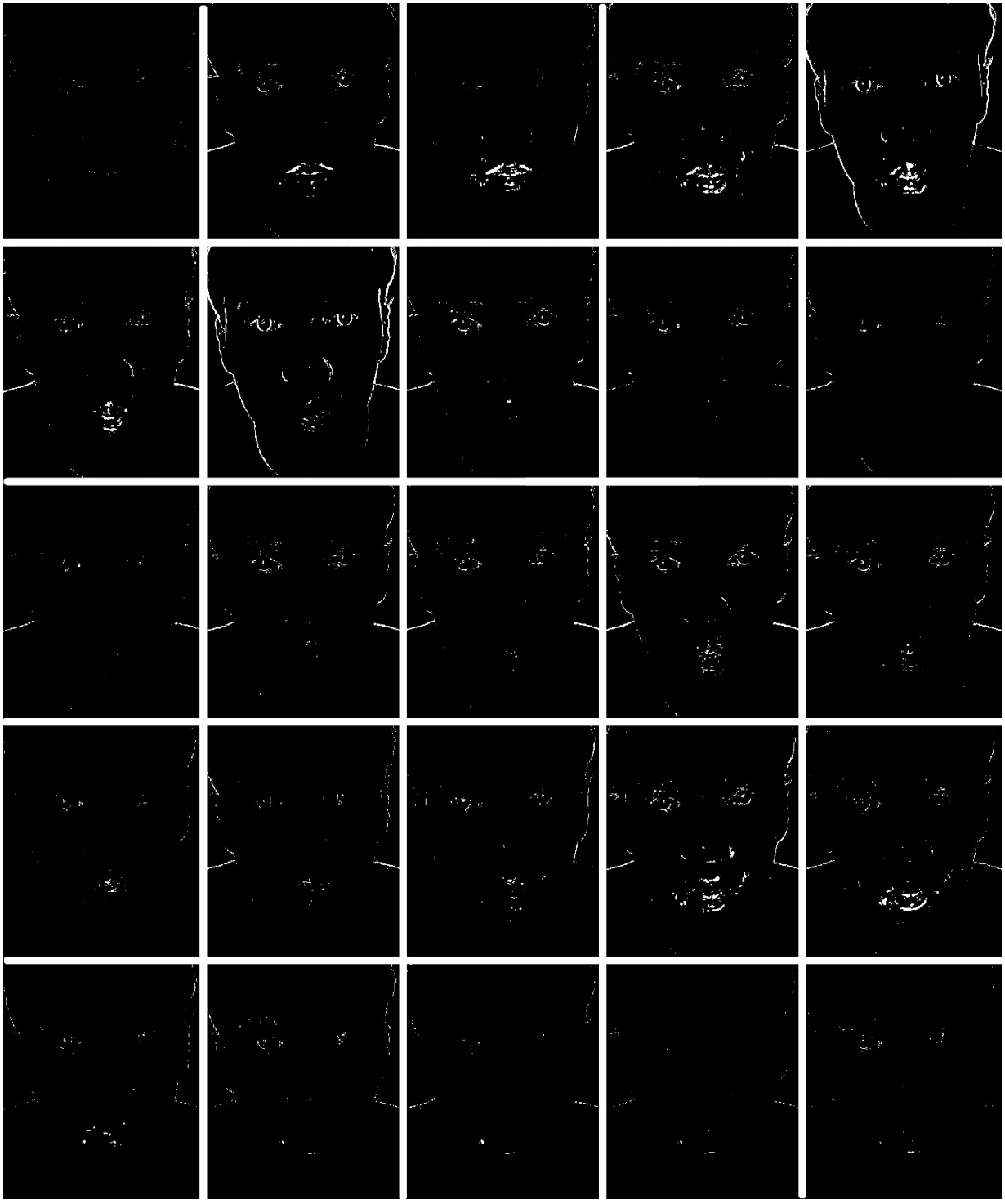

Figura 2: Movimientos detectados en cada uno de los *frames*.

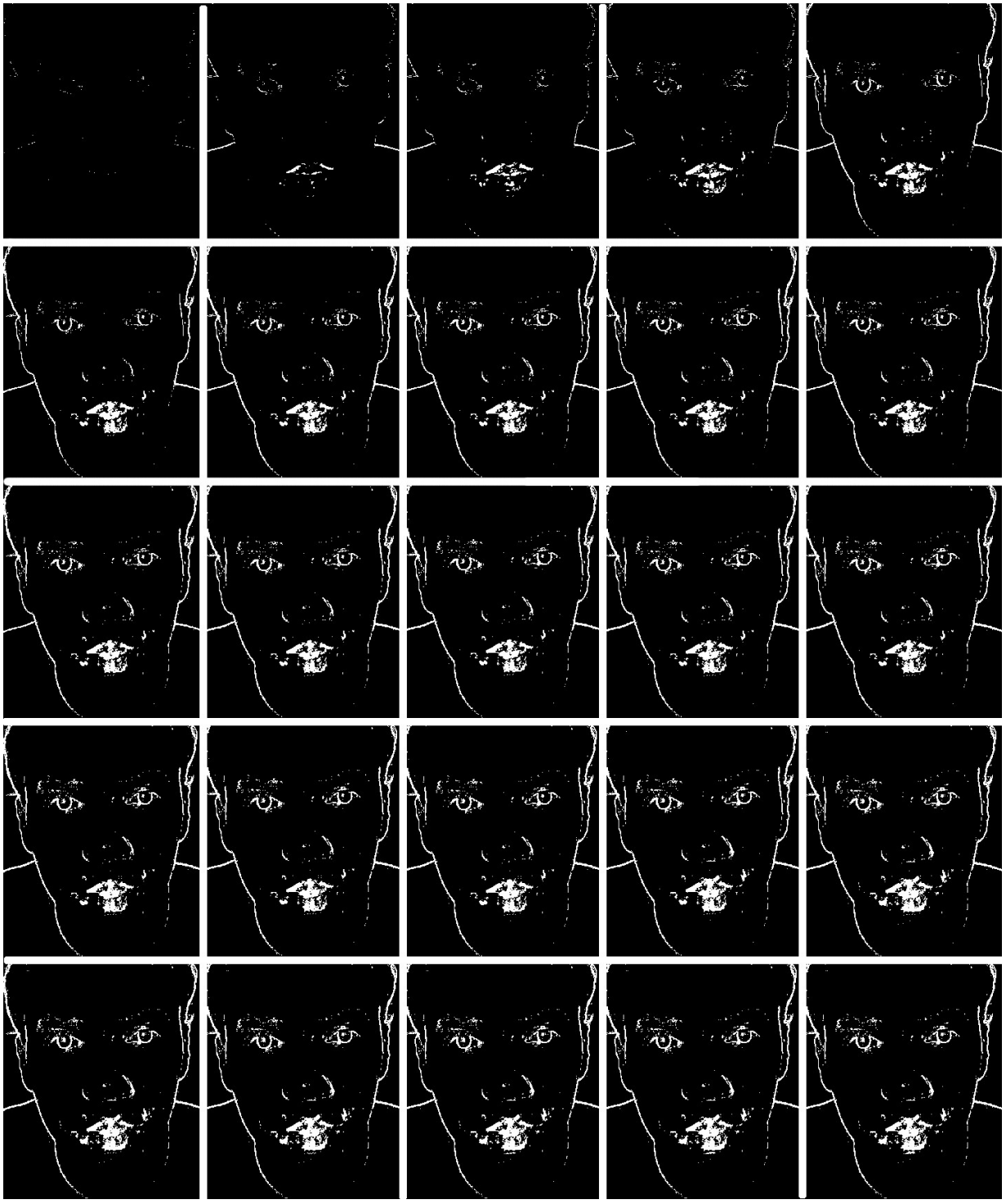

Figura 3: Formación del MEI a partir de los movimientos detectados.

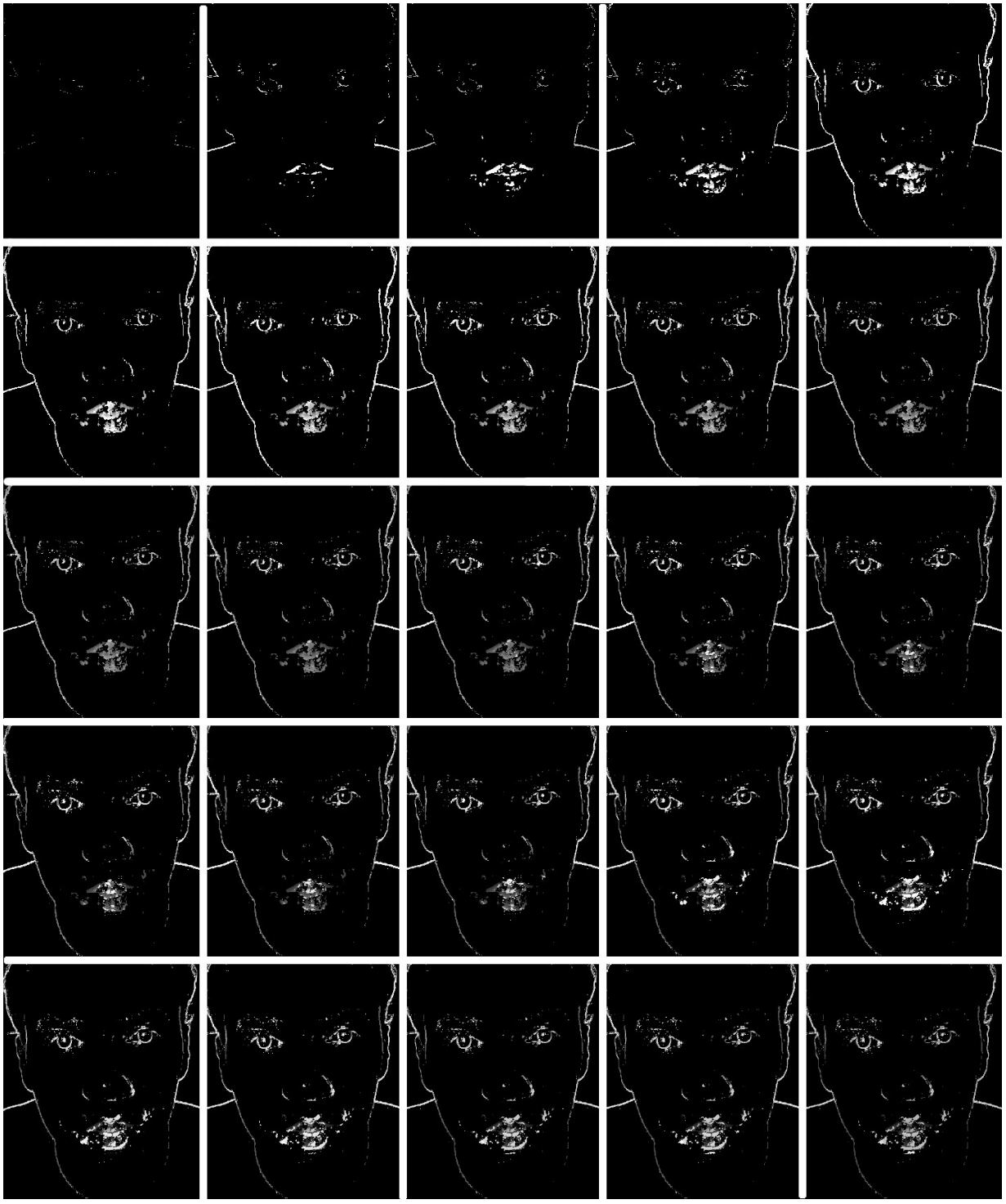

Figura 4: Formación del MHI a partir de los movimientos detectados.

Bibliografía

[1] P. Ekman and W. Friesen, *Facial Action Coding System: A Technique for the Measurement of Facial Movement.* Palo Alto: Consulting Psychologists Press, 1978.

[2] P. Ekman, W. V. Friesen, and J. C. Hager, *Facial Action Coding System (FACS): Manual*, vol. 1. Salt Lake City (USA): Research Nexus division of Network Information Research Company, 2002.

[3] A. Zelinsky, E. Zelinsky, and J. Heinzmann, "Real-time visual recognition of facial gestures for human-computer interaction," in *Proceedings of the Int. Conf. on Automatic Face and Gesture Recognition*, pp. 351–356, IEEE Computer Society Press, 1996.

[4] M. E. Algorri and A. Escobar, "Facial gesture recognition for interactive applications," in *Proceedings of the Fifth Mexican International Conference in Computer Science*, (Washington, DC, USA), pp. 188–195, IEEE Computer Society, 2004.

[5] A. R. Naghsh-Nilchi and et al., "An efficient algorithm for motion detection based facial expression recognition using optical flow," 2006.

[6] M. La Cascia, L. Valenti, and S. Sclaroff, "Fully automatic, real-time detection of facial gestures from generic video," *IEEE 6th Workshop on Multimedia Signal Processing 2004*, pp. 175–178, 2004.

[7] W.-K. Liao and I. Cohen, "Classifying facial gestures in presence of head motion," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops - Volume 03*, (Washington, DC, USA), pp. 77–, IEEE Computer Society, 2005.

[8] M. F. Valstar, I. Patras, and M. Pantic, "Facial action unit recognition using temporal templates," in *Proceedings of IEEE Int'l Workshop on Robot-Human Interaction (RO-MAN'04)*, (Kurashiki, Japan), pp. 253–258, September 2004.

[9] M. F. Valstar and M. Pantic, "Fully automatic facial action unit detection and temporal analysis," in *Proceedings of IEEE Int'l Conf. Computer Vision and Pattern Recognition (CVPR'06)*, vol. 3, (New York, USA), p. 149, June 2006.

[10] S. Koelstra, M. Pantic, and I. Patras, "A dynamic texture based approach to recognition of facial actions and their temporal models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 1940–1954, November 2010.

[11] J. W. Davis and A. F. Bobick, "The Representation and Recognition of Human Movement Using Temporal Templates," in *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, IEEE Computer Society, 1997.

[12] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS - PART C*, vol. 37, no. 3, pp. 311–324, 2007.

[13] M. A. R. Ahad, T. Ogata, J. K. Tan, H. Kim, and S. Ishikawa, "Motion recognition approach to solve overwriting in complex actions.," in *FG*, pp. 1–6, IEEE, 2008.

[14] R. E. W. Rafael C.Gonzalez, *Digital Image Processing.* Addison-Wesley, 1992.

[15] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (Hawaii), 2001.

[16] R. Lienhart, E. Kuranov, and V. Pisarevsky, "Empirical analysis of detection cascades of boosted classifiers for rapid object detection," in *In DAGM 25th Pattern Recognition Symposium*, pp. 297–304, 2003.

[17] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98, pp. 555–, IEEE, 1998.

[18] Y. Freund and R. Schapire, "A short introduction to boosting," *J. Japan. Soc. for Artif. Intel.*, no. 5, pp. 771–780, 1999.

[19] A. Bradski, *Learning OpenCV, [Computer Vision with OpenCV Library ; software that sees].* O'Reilly Media, 2008.

[20] M. Castrillón, O. Déniz, C. Guerra, and M. Hernández, "Encara2: Real-time detection of multiple faces at different resolutions in video streams," *J. Vis. Comun. Image Represent.*, vol. 18, pp. 130–140, April 2007.

[21] G. R. Bradski, "Computer Vision Face Tracking For Use in a Perceptual User Interface," *Journal of Intel Technology Q2 1998*, 1998.

[22] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *Journal of IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 603–619, 2002.

[23] C.-c. Hsieh, D.-h. Liou, Y.-m. Cheng, and F.-c. Cheng, "Robust visual mouse by motion history image," *Current*, pp. 161–166, 2010.

[24] M. Valstar, M. Pantic, and I. Patras, "Motion history for facial action detection from face video," in *Proceedings of IEEE Int'l Conf. Systems, Man and Cybernetics (SMC'04)*, (The Hague, Netherlands), pp. 635–640, October 2004.

[25] M. Ahad, J. Tan, H. Kim, and S. Ishikawa, "Solutions to motion self-occlusion problem in human activity analysis," in *Computer and Information Technology, 2008. ICCIT 2008. 11th International Conference on*, pp. 201 –206, December 2008.

[26] M. A. R. Ahad, J. K. Tan, H. Kim, and S. Ishikawa, "Motion history image: its variants and applications," *Machine Vision and Applications*, pp. 1–27, 2010.

[27] M. Ahad, J. Tan, H. Kim, and S. Ishikawa, "Human activity analysis: Concentrating on motion history image and its variants," in *ICCAS-SICE, 2009*, pp. 5401 –5406, August 2009.

[28] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (ijcai)," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, pp. 674–679, April 1981.

[29] M. Pantic and L. Rothkrantz, "Automatic recognition of facial expressions and human emotions," in *Proceedings of Annual Conf. of Advanced School for Computing and Imaging (ASCI'97)*, (Heijen, The Netherlands), pp. 196–202, June 1997.

[30] S. Milborrow, J. Morkel, and F. Nicolls, "The MUCT Landmarked Face Database," *Pattern Recognition Association of South Africa*, 2010. `http://www.milbo.org/muct`.

[31] C.-H. Teh and R. T. Chin, "On image analysis by the methods of moments," *Journal of IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 496–513, 1988.

[32] M.-K. Hu, "Visual pattern recognition by moment invariants," *Information Theory, IRE Transactions on*, vol. 8, pp. 179–187, February 1962.

[33] T. Cover and P. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21–27, 1967.

[34] C. Cortes and V. Vapnik, "Support-Vector Networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[35] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pp. 144–152, ACM Press, 1992.

[36] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability* (J. Neyman, ed.), pp. 481–492, University of California Press, Berkeley, CA, USA, 1950.

[37] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27:1–27:27, May 2011.

[38] H.-T. Lin, C.-J. Lin, and R. C. Weng, "A note on platt's probabilistic outputs for support vector machines," *Mach. Learn.*, vol. 68, no. 3, pp. 267–276, 2007.

[39] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, 2001.

[40] H. Kang, C. W. Lee, and K. Jung, "Recognition-based gesture spotting in video games," *Pattern Recognition Letters*, vol. 25, no. 15, pp. 1701–1714, 2004.

[41] W. T. Freeman, D. B. Anderson, P. A. Beardsley, C. N. Dodge, M. Roth, C. D. Weissman, W. S. Yerazunis, H. Kage, K. Kyuma, Y. Miyake, and K. ichi Tanaka, "Computer vision for interactive computer graphics," *IEEE Computer Graphics and Applications*, vol. 18, pp. 42–53, 1998.

[42] K. A. Spackman, "Signal detection theory: Valuable tools for evaluating inductive learning," in *Proceedings of the Sixth International Workshop on Machine Learning*, (San Mateo, CA), pp. 160–163, Morgan Kaufman, 1989.

[43] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, pp. 1145–1159, 1997.

[44] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating (roc) curvel characteristic," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.