Identificación de Gestos para la Interacción con Sistemas de Realidad Aumentada mediante Seguimiento de Mano

Por
Isauro López Cortegano

Máster en Ingeniería Informática
Facultad de Informática

Dirigido por
Alejandro Romero Hernández
Borja Manero Iglesias

Madrid, 2021–2022

Convocatoria de Julio, Calificación: 10

Dedicado a mis padres,
por su paciente esfuerzo y por el amor y el apoyo que siempre me ofrecen.

Agradecimientos a,
Raquel Carranza Arias por ser mi cielo estos seis últimos años
y a mis queridos hermanos:
Eugenio
Carlos
Jacinto
Mª del Pilar

Gracias de corazón.

Abstract

In an increasingly digitized world, one of the great technological challenges is to adapt and optimize human-machine interactions. This is especially true in the field of Extended Reality (XR), a technology that is currently one of the focal points of innovation and where many teams and companies are experimenting and researching new ways of interacting with the environment and digital artifacts.

Therefore, it is particularly important to design intuitive interfaces and usable interaction systems for users [1]. In this work we present a model that allows to define gestures digitally in order to speed up the gesture identification process from sequences of images captured by a camera. To evaluate the effectiveness and usability of the model, tools have been developed: one for computers and another for cell phones, which have been tested with users.

Volunteers who have participated in the experiments claim that the learning curve of the designed interaction system is low. The experiments conducted conclude that the designed system is reliable and that the applications that implement it are usable.

Keywords: *Mediapipe*, Augmented Reality, *Moverio BT35E* Smart Glasses, *Hand Tracking*, Gesture Modelling.

Resumen

En un mundo cada vez más digitalizado, uno de los grandes retos tecnológicos es adaptar y optimizar la interacción entre persona y máquina. Especialmente en el campo de la Realidad Extendida (XR), una tecnología que en la actualidad es uno de los focos de la innovación y, donde muchos equipos y compañías experimentan e investigan nuevas formas de interacción con el entorno y los artefactos digitales.

Por ello, es de especial importancia diseñar interfaces intuitivas y sistemas de interacción usables para los usuarios [1]. En este trabajo se presenta un modelo que permite definir gestos de manera digital con el que se busca agilizar el proceso de identificación de los mismos a partir de secuencias de imágenes capturadas por una cámara. Para evaluar la eficacia y usabilidad del modelo, se han desarrollado herramientas: una para ordenador y otra móvil de Realidad Aumentada (AR) con las que posteriormente se ha experimentado con usuarios.

Los voluntarios que han participado en los experimentos afirman que la curva de aprendizaje del sistema de interacción diseñado es baja. Los experimentos realizados concluyen que el sistema diseñado es fiable y que las aplicaciones que lo implementan son usables.

Palabras clave: *Mediapipe*, Realidad Aumentada, Gafas Inteligentes *Moverio BT35E*, Seguimiento de manos, Modelado de Gestos.

Índice general

Acrónimos

AR Realidad Aumentada. 1, 3–5, 7, 14, 34, 35, 51, 58, 61, 63, 64, 66

DP Alt Mode Display Port Alternative Mode. 38, 45, 58, 66

FPS Fotogramas Por Segundo. 52, 56

ML Machine Learning. 1, 15, 24, 58, 59, 61, 66, 67

MR Realidad Mixta. 5, 7

SDK Kit de Desarrollo Software. 37, 39, 46

SM Gafas Inteligentes. 4, 8, 9, 14, 36, 38, 45, 46, 54, 57, 58, 64, 66

VE Entorno Virtual. 5, 9, 15

VR Realidad Virtual. 5–9, 15

WSL Windows Subsystem para Linux. 47

XR Realidad Extendida. 4–6, 8, 37, 64

Índice de figuras

Índice de tablas

Lista de algoritmos

Capítulo 1

Introducción

1.1. Motivación

El continuo desarrollo de la tecnología y los avances científicos permiten, cada vez más, materializar ideas y modelos que tiempo atrás parecían imposibles. Este trabajo nace de la ilusión por experimentar con la tecnología de AR y del deseo de desarrollar uno de los sistemas presentes en el universo de la ciencia ficción: un sistema de interacción con programas a través de gestos.

1.2. Objeto de la investigación

En este trabajo se propone un modelo determinista que se basa en la solución de Machine Learning (ML), *Hands*, propuesta por *Google*. El objetivo del trabajo es agilizar el proceso de identificación de gestos haciendo uso del modelo propuesto. Gracias a este, se busca también agilizar el proceso de diseño de los sistemas de interacción con los programas. Por ello, se han desarrollado aplicaciones: una para escritorio y otra para dispositivos móviles, con el objetivo de evaluar la efectividad del modelo propuesto así como la experiencia de los usuarios.

1.3. Plan de trabajo

Este Trabajo de Fin de Máster ha sido realizado en colaboración con el Departamento de Ingeniería de Software e Inteligencia Artificial de la *UCM*. En esta sección se presenta la metodología de trabajo seguida y se documenta brevemente las tareas realizadas.

1.3.1. Metodología de trabajo

Para desarrollar este trabajo se ha seguido una metodología circular basada en el método científico, ilustrada en la Figura1.1. Esta metodología se ha escogido por la flexibilidad y agilidad que ofrece en caso de errores o tomar malas decisiones y por ser la más sencilla y eficiente de implementar en el marco de este trabajo.

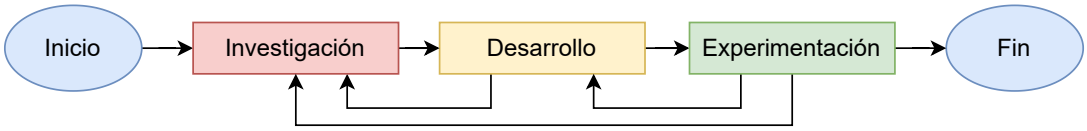


Figura 1.1: Metodología de trabajo

A continuación se describen brevemente los procesos ilustrados en la Figura 1.1.

- **Investigación:** Esta fase comprende aquellas tareas asociadas con el planteamiento, discusión de hipótesis, el estudio y búsqueda de artículos científicos y tecnología.
- **Desarrollo:** En la fase de desarrollo agrupamos todas las tareas relacionadas con el diseño, la programación y pruebas unitarias de los programas que se elaboran.
- **Experimentación:** La fase de experimentación comprende labores de testeo, experimentación con usuarios y la recogida y análisis de datos.
- **Documentación:** Si bien no está considerada como una fase de la metodología presentada, se hace especial mención a la misma por ser una labor de suma importancia que perdura y es constante a lo largo de todo el trabajo.

1.3.2. Repositorio de Trabajo

Para el control de versiones de este trabajo se ha empleado un repositorio público de *Git*. El enlace de acceso al repositorio es el siguiente: `https://gitlab.com/iSauron/tfm`.

Respecto a la *URL* del repositorio, Internet está vivo y en constante cambio. Si el enlace no se encuentra disponible y desea obtener una copia del contenido y la dirección web no se encuentra disponible, contacte conmigo y le remitiré los contenidos del trabajo.

1.3.3. Tareas y planificación

La Tabla 1.1 muestra las tareas primarias del trabajo categorizadas por proceso. Las numerosas sub-tareas y etapas en las que se descomponen no se representan por motivos de claridad y comprehensión.

ID	Nombre
1. Investigación	
1.a	Estudio del estado del arte
1.b	Investigación sobre tecnología de *handtracking*
1.c	Selección y adquisición del material de AR
2. Desarrollo	
2.a	Prototipos para la detección de gestos
2.b	Aplicación de AR
3. Experimentación	
3.a	Experimentación y análisis de gestos con los prototipos
3.b	Experimentación del aplicativo de AR con usuarios

Tabla 1.1: Tareas Realizadas

Planificación

La Figura 1.3.3 esquematiza el periodo de tiempo consumido para la realización de este trabajo. En esta figura, la segunda barra de título hace referencia a la semana del año. Dicho de otro modo, el trabajo se inició la semana del 14/06/2021 y finalizó la semana del 13/06/2022.

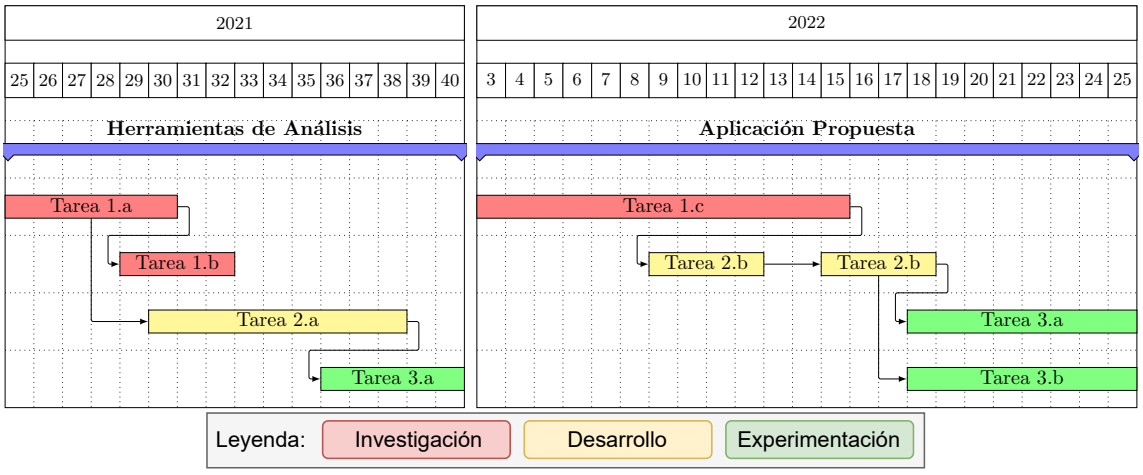


Figura 1.2: Diagrama *Gantt*

1.4. Estructura del trabajo

El resto del trabajo está organizado en 9 capítulos con la estructura que se describe a continuación.

El capítulo 2 contextualiza e introduce brevemente los conceptos, las bases teóricas y las tecnologías requeridas para comprender el marco de trabajo. Concretamente, presenta en primer lugar la tecnología de XR y sus variantes, así como las características y aplicaciones generales de las mismas. A continuación, se profundiza sobre los dispositivos de Gafas Inteligentes (SM), se presentan diversos modelos y se analizan las características que suelen tener así como los usos que se les da. Finalmente, se presenta el concepto de "*Hand Tracking*", del inglés, «seguimiento de la mano» y las técnicas que se emplean para identificar manos en contenido digital.

En el capítulo 3 presenta el estado del arte. En este, se presentan artículos, tecnologías y otros trabajos que comparten similitudes con el proyecto que se quiere realizar y se analizan las técnicas empleadas en el desarrollo de los mismos para su realización.

El capítulo 4 presenta la contribución de este trabajo, un modelo que posibilita la definición digital de gestos para su posterior procesamiento.

Los capítulos 5 y 6 presentan las herramientas y programas que se han desarrollado a lo largo de este trabajo para probar el modelo diseñado en el capítulo 4. En el capítulo 5 se presentan las herramientas elaboradas para automatizar el proceso de definición de gestos. El capítulo 6 presenta el proceso de diseño y desarrollo seguido para elaborar el aplicativo móvil de AR.

En el capítulo 7 se describen los experimentos realizados para evaluar la eficacia del modelo propuesto en el capítulo 4 y la experiencia de usuario del aplicativo presentado en el capítulo 6. Se presenta también el análisis y estudio realizado de los datos recogidos durante las sesiones de prueba.

El capítulo 8 concluye este trabajo: resumiendo brevemente la experiencia del proyecto y analizando los resultados de los experimentos realizados. Se presenta también el trabajo futuro y las posibles líneas de investigación.

Los capítulos 9 y 10 son traducciones al inglés de los capítulos 1 y 8, respectivamente.

Capítulo 2

Marco Teórico

2.1. Realidad extendida

El origen del término XR se remonta a la década de 1960 pero no fue hasta los 2000 que *Coleman*, *Landay* y *Paradiso* introdujeron el concepto "*Cross Reality*" [2]. La XR encapsula a todas aquellas tecnologías basadas en la interacción persona-máquina capaces de superponer texto, gráficos y animaciones en un Entorno Virtual (VE) o en el mundo real.

El concepto de XR agrupa bajo un mismo nombre, tecnologías como:

- La VR, que sumerge al usuario en un mundo virtual con el objetivo de que éste se sienta inmerso en el VE (ver figura 2.1).
- La MR, que sobrepone artefactos y objetos generados por ordenador en el entorno real con los que el usuario puede interaccionar. Los textos y gráficos sobrepuestos se «anclan» al entorno real, dicho de otro modo, los componentes virtuales aparecen en puntos concretos del espacio, se ven afectados por nuestras acciones, se pueden trasladan si se encuentran sobre un objeto en movimiento o si son golpeados por un objeto del mudo real (ver figura 2.2).
- La AR opera igual que la MR salvo que, a diferencia de esta, los objetos virtuales se anclan exclusivamente sobre la imagen que se captura y no sobre el mundo real. Pudiendo así, ver los artefactos virtuales independientemente del entorno y las condiciones en las que se encuentre el usuario (ver figura 2.3).

Cada una de estas tecnologías ofrece al usuario una forma única de interacción con el VE, en la tabla 2.1[3] se presenta un esquema comparativo que sintetiza de forma generalizada el volumen de artefactos generados por ordenador y la cantidad de elementos reales que

están presentes en una aplicación de XR así como el grado de interacción que ésta ofrece a los usuarios.

	Realidad Virtual (*VR*)	Realidad Aumentada (*AR*)	Realidad Mixta (*MR*)
Contenido virtual	Alto	Escaso	Intermedio
Contenido real	Escaso	Alto	Alto
Interacción	Escasa	Intermedia	Alta

Tabla 2.1: Comparativa de tecnologías XR

La XR se emplea en una amplia variedad de campos y sectores con propósitos y objetivos muy diversos. A continuación, enumeramos algunos de sus usos más extendidos como lo son:

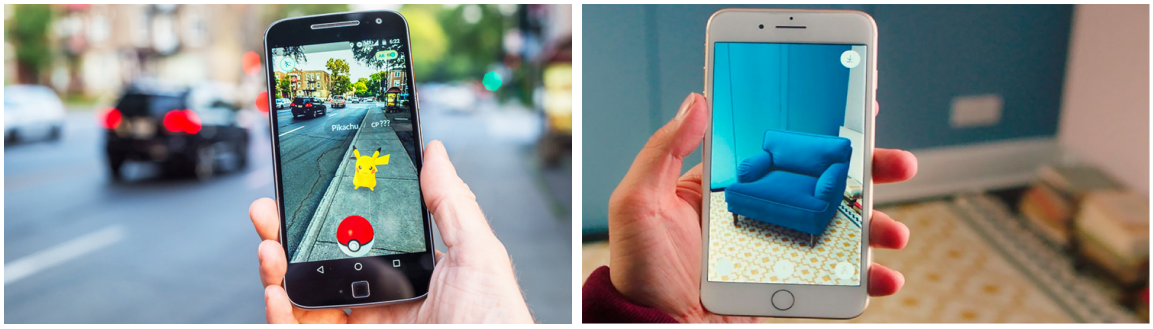
- El entretenimiento, como por ejemplo los videojuegos.
- Las actividades de ocio y turismo: desde tiendas de ropa y muebles virtuales, dónde los usuarios pueden probarse ropa o colocar muebles en su casa, a experiencias de turismo virtual.
- Las actividades profesionales: como pueden ser simuladores de vuelo, experiencias de marketing virtual para los clientes, diseño y modelado de edificios, etc.
- La asistencia remota: programas que posibilitan a especialistas, asesorar a técnicos de forma remota. En estos programas normalmente el especialista puede interactuar con la imagen que está viendo el técnico y dibujar sobre esta.
- La formación de personal profesional, como ocurre por ejemplo en el sector sanitario o en centrales nucleares. Se recrean entornos virtuales fieles a la realidad con el objetivo de que el personal entrene en condiciones seguras sin poner en riesgo y compromiso las instalaciones y la vida de las personas.

(a) Cascos de VR (b) Visión de la VR

Figura 2.1: Tecnología VR

(a) Gafas Inteligentes (b) Visión de la MR

Figura 2.2: Tecnología MR



(a) *Pokemon GO* (b) Tienda virtual

Figura 2.3: Tecnología AR

Las figuras 2.1[1], 2.2[2] y 2.3[3] ilustran los dispositivos físicos que se emplean respectivamente en los entornos de la VR, MR y la AR. En este trabajo, nos centraremos en desarrollar una aplicación de AR, por lo que profundizaremos en la siguiente sección sobre los dispositivos comerciales más extendidos que se emplean para el desarrollo de aplicaciones de AR.

2.2. Dispositivos de realidad aumentada

Un gran número de aplicaciones de AR, más de 6000 millones[4], utilizan un dispositivo móvil como soporte físico, de las cuales, el 77 % requiere el uso de algún dispositivo de AR externo.

Los dispositivos de AR, por lo general, se conectan al teléfono a través de un cable *USB* tipo *C*. Este cable proporciona la energía y los datos que necesita el dispositivo para

[1]https://virtual-reality.co.za/wp-content/uploads/2016/07/PlayStation-VR-South-Africa.jpg
[2]https://www.pcmag.com/news/google-glass-everything-you-need-to-know
[3]https://latam.ign.com/pokemon-go/45221/preview/pokemon-go-presenta-modo-de-realidad-aumentada-exclusivo-para-apple

operar. En algunos casos se suelen emplear baterías externas para extender el tiempo de uso del teléfono. A continuación, presentamos brevemente los dispositivos más extendidos así como sus principales características.

2.2.1. Cascos de realidad virtual

Los cascos de VR, también conocidos como gafas de VR o visor de VR, son dispositivos que reproducen imágenes sobre una pantalla colocada muy cerca de los ojos, o bien, proyectan la imagen directamente sobre la retina de estos.

Los primeros modelos comerciales surgieron en el año 1991 y siguientes, no obstante, fracasaron debido a las limitaciones tecnológicas de la época. En el año 2012 surgió una campaña de *kickstarter*, una campaña web de micromecenazgo para proyectos creativos, para unos cascos de VR conocidos como *Oculus Rift*. Este modelo marcó un antes y después, es un hito en la industria de la tecnología de la XR.

Como se observa en la figura 2.1(a), estos dispositivos suelen contar con un arnés que mantiene el casco fijo en la cabeza, de esta forma el usuario no pierde contacto con la imagen. Gracias a los giroscopios que lleva integrados, el casco permite a la aplicación de VR conocer en todo momento la dirección hacia la que mira el usuario y el programa, re-orienta la cámara del mundo virtual para dar la sensación al usuario de que está sumergido en un mundo virtual.

Dispositivos más modernos, como las gafas *Oculus Quest*, integran cámaras y multitud de sensores con las que se captura, entre otra información, la imagen del entorno y puede emplearse para el desarrollo de aplicaciones de XR.

2.2.2. Gafas inteligentes

Las SM, como la ilustrada en la figura 2.2(a), son dispositivos que incorporan una tecnología en forma de pantalla que puede proyectarse o reflejarse en las lentes. Estos dispositivos superponen información en el campo de visión del usuario sin provocar grandes distracciones.

Los primeros trabajos con gafas inteligentes datan del año 1997 y se inspiran de las tecnologías integradas en aviones militares de los años 70 para la asistencia al pilotaje. No obstante, no fue hasta el año 2012 que empresas como *Oakley* o *Google* anunciaron que estaban trabajando en el desarrollo de gafas inteligentes. El modelo de gafa inteligente de *Google*, las *Google Glass*, estuvo disponible para su venta al público en el año 2014

y generó una gran polémica en torno a que el uso de estos dispositivos podría violar las leyes de privacidad existentes.

Estos dispositivos están en continuo desarrollo y están cada vez más presentes en los sectores industriales, ayudando al personal a ser cada vez más eficiente. Hacemos mención a continuación de usos frecuentes de éste tipo de dispositivos en los sectores de:

- El ocio; la captura de fotos y publicación de historias en redes sociales.
- La seguridad; la búsqueda e identificación de personas, alertar de obstáculos a las personas con visibilidad reducida, etc.
- El deporte; muchas centradas en el ciclismo, ofrecen información de utilidad al usuario como su velocidad, la aceleración o la dirección del viento por ejemplo.

Los modelos comerciales de cascos de VR y SM en la actualidad integran cámaras y numerosos sensores, gracias a los cuales pueden elaborarse programas más complejos y precisos. Estudiaremos en la siguiente sección qué es la tecnología de "*Hand tracking*" y presentaremos algunos productos y soluciones que existen para integrarlo en los programas.

2.3. *Hand tracking*

"*Hand tracking*" o, seguimiento de la mano, es un término que se asocia con todas aquellas tecnologías que permiten adquirir datos en vivo sobre el movimiento de una mano y sus dedos. Las técnicas y estrategias empleadas para su puesta en marcha pueden agruparse en dos grandes tendencias:

- El uso de sensores.
- El análisis de imágenes con redes neuronales.

Los sensores se emplean para construir dispositivos hápticos que recogen datos en vivo con los que se puede alimentar a un sistema informático que procesa la información.

2.3.1. Guantes hápticos

Estos dispositivos, como su nombre indica, son guantes que permiten a los usuarios interactuar con el VE en los programas de VR con sus manos. Estos guantes están repletos de sensores que capturan el estado de la mano y la representan en el mundo virtual.

Este comportamiento se ilustra en la figura 2.1(b). Los dispositivos que se muestran en la figura 2.4[4][5] son ejemplos de guantes hápticos disponibles en el mercado.

(a) *HaptX DK2* (b) Manus VR

Figura 2.4: Guantes hápticos

Por el contrario, los sistemas basados en la captura de imágenes y su análisis con redes neuronales suelen hacer uso de varias cámaras, dispuestas en puntos estratégicos con el propósito de mejorar la precisión del análisis.

2.3.2. Mediapipe

MediaPipe es un repositorio de *Google* que contiene y ofrece algoritmos de *Machine Learning*(***ML***). El repositorio es accesible a través del siguiente enlace: `https://google.github.io/mediapipe/`.

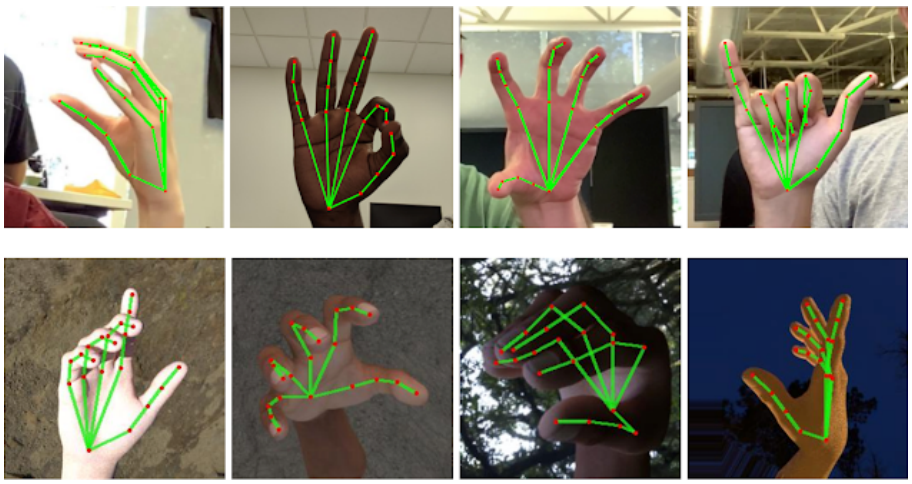


Figura 2.5: *MediaPipe*: manos

[4]https://gagadget.com/es/89852-una-startup-con-sede-en-seattle-dice-que-el-prototipo-de-guante-meta-vr-es-identico-a-su-propia-tecnologia-patentada/

[5]https://clipset.com/manus-vr-el-guante-que-integra-manos-y-dedos-en-la-realidad-virtual/

De entre las múltiples soluciones que ofrece, se encuentra *hands*. *Hands* es una solución de *ML* que analiza imágenes e identifica manos, como puede observarse en la figura 2.5[6].

Hands tiene soporte para operar en entornos donde analiza imágenes aisladas o secuencias de vídeo. Permite también cambiar y ajustar parámetros con los que opera el modelo de *ML* subyacente. Permitiendo así, adaptar la solución al caso práctico.

Cuando analiza una imagen, *Hands*, extrae un modelo tridimensional (**3D**) de la mano por cada mano que identifica. El modelo de una mano está compuesto por veinte vértices, que representan la posición del espacio que ocupa el punto de la mano correspondiente en un espacio 3D normalizado con el ancho y alto de la imagen que analiza. En la figura 2.6 se ilustran los vértices de los que se compone el modelo 3D.

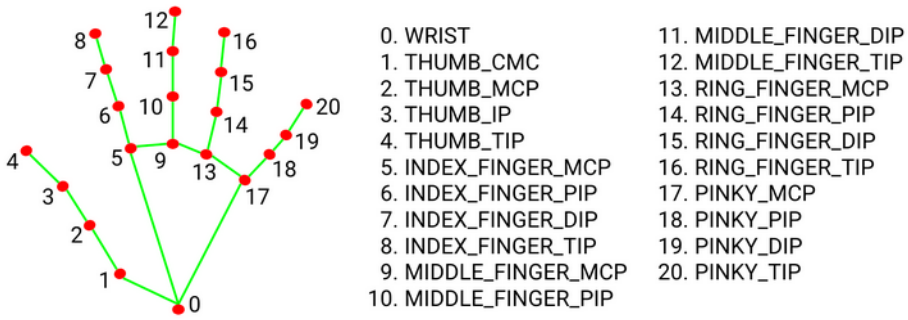


Figura 2.6: *Hands*: modelo de la mano

[6]https://google.github.io/mediapipe/solutions/hands.html

Capítulo 3

Estado del arte

Al campo de estudio que se dedica a la investigación y búsqueda de formas de interacción persona-máquina se le denomina **HCI**, del inglés "*Human Computer Interaction*". Su propósito es buscar nuevas y mejores formas de interactuar con los programas, diseñar interfaces de usuario intuitivas y útiles, con el objetivo de mejorar la experiencia de los usuarios o la eficiencia de los sistemas.

Los equipos profesionales de *HCI* suelen ser multi-disciplinares y están compuestos por expertos de una gran variedad de áreas. En la figura 3.1 se presentan las tres grandes disciplinas científicas que intervienen en estos estudios.

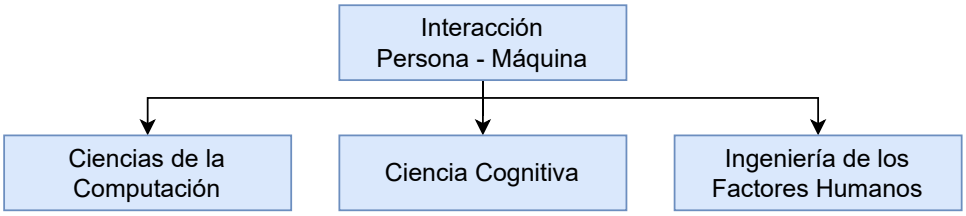


Figura 3.1: Disciplinas Científicas de la *HCI*

Participan expertos del comportamiento y la psicología humana, que asisten a los diseñadores en la elaboración de un producto y en la definición de los mecanismos de interacción con el mismo. Participan también ingenieros, científicos y personal técnico en el diseño y construcción de los programas y dispositivos. En nuestro trabajo, nos centramos en definir un modelo computacional que busca agilizar y facilitar la identificación de gestos. El modelo a proponer basa su funcionalidad en el bucle de interacción[5] ilustrado en la figura 3.2.

Este modelo se inspira del comportamiento subconsciente humano y comprende la interacción con las máquinas de la misma forma que las personas interactuamos con el entorno.

Por ejemplo, cuando estiramos el brazo para coger una taza, nuestro cerebro, utilizando la vista, corrige constantemente el movimiento del brazo para alcanzar el asa de la taza. Del mismo modo, las acciones que realizamos en un programa emiten estímulos visuales o sonoros para ayudarnos a realizar la siguiente acción. En el caso de nuestro modelo, cada entrada emite una señal de salida con la que, según el caso práctico, se determina qué acción ejecuta el programa.

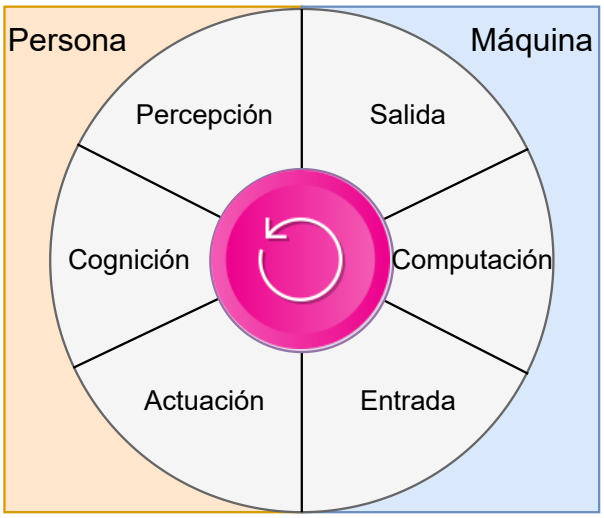


Figura 3.2: Bucle de Interacción

En esta sección del trabajo presentamos el estado del arte, exponemos a continuación un subconjunto de los artículos leídos y mencionaremos las tecnologías y técnicas empleadas que se mencionan.

Introduciremos primeramente algunas tecnologías empleadas en la detección de gestos e interacción en el aire basadas en sensores. Seguidamente, presentaremos trabajos que basan la interacción en el análisis de imágenes. Por último, analizamos los artículos leídos y contrastamos la tecnología empleada y sus escenarios de uso.

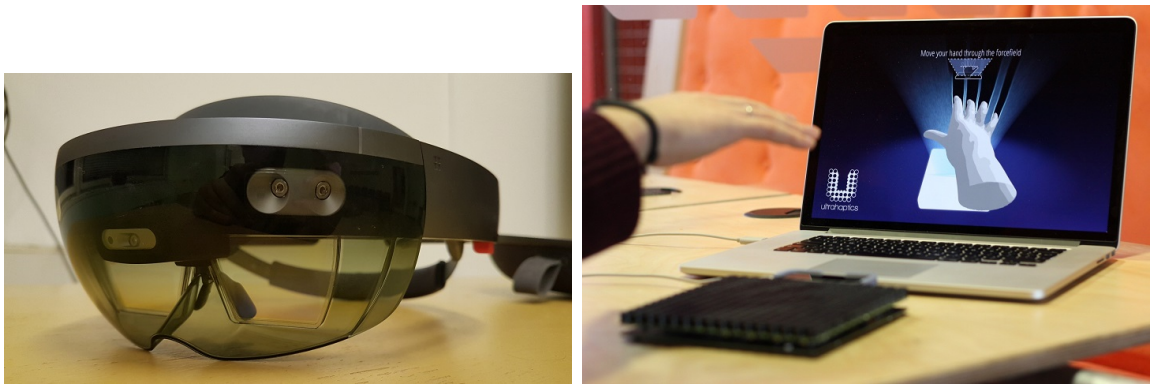
3.1. Interacción basada en sensores

3.1.1. Con transductores

Los transductores son dispositivos que pueden transformar una energía en otro tipo de energía, por ejemplo, un transformador electromecánico cambia la energía eléctrica a mecánica, un sensor piroeléctrico genera señales eléctricas ante oscilaciones térmicas, etc.

En el artículo "*Touch hologram in mid-air*"[6] se presenta un sistema con el que es

posible interactuar con pantallas holográficas. Para lograrlo utilizan, el dispositivo háptico *Ultrahaptics*, una matriz de transductores ultrasónicos. Gracias al cual, el sistema detecta objetos y perturbaciones ultrasónicas en un volumen de espacio controlado. Haciendo uso de este dispositivo, y proyectando hologramas sobre las SM *HoloLens* de *Microsoft*, detectan la mano de usuario y sus movimientos. Dando la sensación al usuario de que está interactuando con los objetos proyectados en la gafa inteligente. Los dispositivos anteriormente presentados se ilustran en la figura 3.3[1][2].



(a) *HoloLens* (b) *Ultrahaptics*

Figura 3.3: Interacción con Sensores Ultrasónicos

3.1.2. Con controladores *leap motion*

Figura 3.4: Interacción con Módulos Ópticos

El módulo óptico *leap motion* es un controlador desarrollado por la empresa americana *Leap Motion Inc.* para el seguimiento de manos. Este dispositivo captura los movimientos de ambas manos y digitaliza una representación de ésta y sus vértices. El dispositivo en cuestión se ilustra en la figura 3.4[3].

En el estudio "hand gesture-based interactions for manipulating 3D virtual objects in mobile AR"[7] se desarrolla una aplicación de AR para manipular objetos con las manos. En este estudio se propone el uso de un dispositivo *leap motion* que puede colocarse en la parte frontal o en el reverso del dispositivo móvil.

[1]https://xboxer.sk/wp-content/uploads/2017/02/HoloLens-1-1.jpg
[2]https://www.techspark.co/wp-content/uploads/2015/09/Ultrahaptics_T003_Mid_air_touch_technology_in_action.jpg
[3]https://www.ultraleap.com/product/leap-motion-controller/

Si bien no es el foco del artículo, en esta investigación se analiza, entre otros factores, la efectividad del dispositivo en distintos escenarios. Demuestran con sus experimentos la gran precisión y fiabilidad que tienen estos controladores.

3.2. Interacción basada en cámaras

Los artículos que se presentan a continuación basan sus sistemas de seguimiento de la mano en el análisis de imágenes procedentes de cámaras. Los módulos encargados del análisis de imágenes hacen uso de algoritmos de ML, en numerosas ocasiones, se alimentan de fotogramas procedentes de varias cámaras. Esto es así porque se busca minimizar los puntos ciegos y maximizar la precisión de las predicciones que emiten los módulos de identificación de manos.

3.2.1. Con realidad virtual

La tecnología de VR incorpora numerosas cámaras y sistemas de reconocimiento y seguimiento de gestos. Con esta tecnología se han desarrollado multitud de experiencias virtuales, sobre todo videojuegos.

En el artículo "*A brief analysis of gesture recognition in VR*"[8] se menciona que en la actualidad existe una gran tendencia a querer interactuar con el VE mediante el uso de mandos o de guantes hápticos y que el uso de estos dispositivos es complejo. También se afirma que una interacción de gestos basada en el uso de las manos desnudas ofrece mayor libertad de uso.

El casco de VR *Oculus Quest 2* es un modelo conocido por integrar un sistema de reconocimiento de gestos. Gracias al cual, los usuarios pueden interactuar con los objetos de la escena virtual mediante el uso de gestos sencillos, este tipo de tecnología está cada vez más presente en la industria. [8]

3.2.2. Con *Mediapipe*

Como se ha descrito en el Capítulo anterior, *mediapipe* es una solución de ML para identificar manos en imágenes.

En el estudio "*Applying Hand Gesture Recognition for User Guide Application Using MediaPipe*"[9] entrenan una red neuronal que se alimenta del modelo que proporciona *mediapipe*, ver figura 2.6. Haciendo uso del modelo neural que proponen, identifican los

gestos que hacen los usuarios para posibilitar la interacción con la herramienta sin necesidad de interactuar con el ratón.

Definieron un total de 10 perfiles de mano reconocibles. Los datos con los que entrenaron el modelo neuronal fueron obtenidos del procesamiento de 900 fotogramas. La precisión del modelo obtenido alcanza el 95 % de éxito.

3.3. Comparación del estado actual

Artículo	Utiliza	Estrategia	Tecnología	Sistema	Coste €
3.1.1	Sensores	*Ultrahaptics*	*Unreal* y *Unity*	Estático	Alto
3.1.2	Sensores	*Leap Motion*	*Unreal* y *Unity*	Estático	Medio
3.2.1	Cámaras	Cascos de *VR*	*Unreal* y *Unity*	Dinámico	Alto
3.2.2	Cámaras	*Mediapipe* y *ML*	Gran variedad	Dinámico	Bajo

Tabla 3.1: Tabla Comparativa de Artículos Presentados

La tabla 3.1 resume los artículos presentados en este Capítulo de forma sintética. En esta tabla se sintetiza de forma generalizada el escenario y los sistemas que se desarrollan en base a la tecnología indicada. La columna de coste representa el valor relativo del equipo y tecnología necesaria respecto a otras estrategias. Los sistemas basados en cámaras son más versátiles y cuentan con un volumen de interacción mayor, ofrecen más libertad de movimiento al usuario y no dependen de instalaciones fijas como sucede con los sensores.

Como se observa en la tabla, existen multitud de estrategias a la hora de capturar información sobre las manos, sin embargo, no existe un modelo unificado a la hora de procesar la información y determinar qué gesto realiza un usuario. Por esta razón, se presenta en el Capítulo siguiente, un modelo determinista con el que se busca unificar el proceso de identificación de gestos.

Capítulo 4

Descripción de la técnica propuesta

4.1. Objetivos

Basándonos en el modelo tridimensional de vértices que proporciona *Hands* (ver figura 2.6) elaboraremos un modelo que se alimente de fotogramas y permita asociar una secuencia de gestos con la ejecución de ordenes.

Nuestro objetivo es proporcionar una definición formal de un modelo con el que poder especificar «perfiles», gracias a los cuales, poder diseñar flujos y secuencias de interacción con los programas y herramientas.

4.2. Identificación *Vs* interpretación de perfiles

En el marco de nuestra investigación, definimos un perfil como el conjunto de configuraciones en la que puede hallarse una mano en el espacio y agruparse bajo un mismo concepto. En la figura 4.1 se ilustra el concepto anteriormente definido; los gestos ilustrados, tienen variaciones pero pueden considerarse «el mismo».

Considerando esto, suponemos que:

- La **identificación** de un perfil no depende de la orientación de la mano.
- La **interpretación** de una secuencia de perfiles puede requerir información del contexto, por ejemplo: lo orientación y dirección en la que se mueve la mano, objetos de la escena, conversaciones, emociones, etc.

Figura 4.1: Perfil «mano abierta»

En este trabajo nos enfocamos exclusivamente en lo que a identificación de perfiles respecta, por lo que no se profundiza en heurísticas o algoritmos para interpretarlos.

4.3. Digitalización de un perfil

Para representar una entidad en el mundo digital primero se debe concretar qué cualidades la caracterizan. En nuestro caso, tomando como punto de partida el modelo que proporciona *Hands*, definiremos cualidades y características sobre el conjunto de veintiún vértices que representan una mano. Obteniendo así, un modelo de alto nivel sencillo e intuitivo.

Puesto que asumimos que la orientación de la mano no debe afectar a la hora de identificar perfiles, las características extraídas deben respetar esta decisión de diseño.

Suposiciones

- La orientación de la mano no debe afectar a la hora de identificar perfiles.
- Los vértices de la palma de la mano se encuentran en el mismo plano (coplanares).
- Los dedos meñique, anular, corazón e índice son paralelos.

Considerando las premisas anteriores, podemos afirmar que:

- La palma de la mano representa un volumen que está siempre presente y apenas varía.
- La disposición de los dedos es lo que nos permite identificar y diferenciar unos gestos de otros.

4.3.1. Estado de los dedos

Representaremos las distintas configuraciones que pueden adoptar los dedos de la mano usando valores discretos, para ello haremos uso del conjunto F, representado en la definición 4.1.

$$F = \{Extended, Relaxed, Closed\} \tag{4.1}$$

Los elementos de F determinan los estados en el que puede hallarse un dedo. Para determinar el estado de los distintos dedos, tomaremos como puntos de referencia los vértices ilustrados en la figura 4.2.

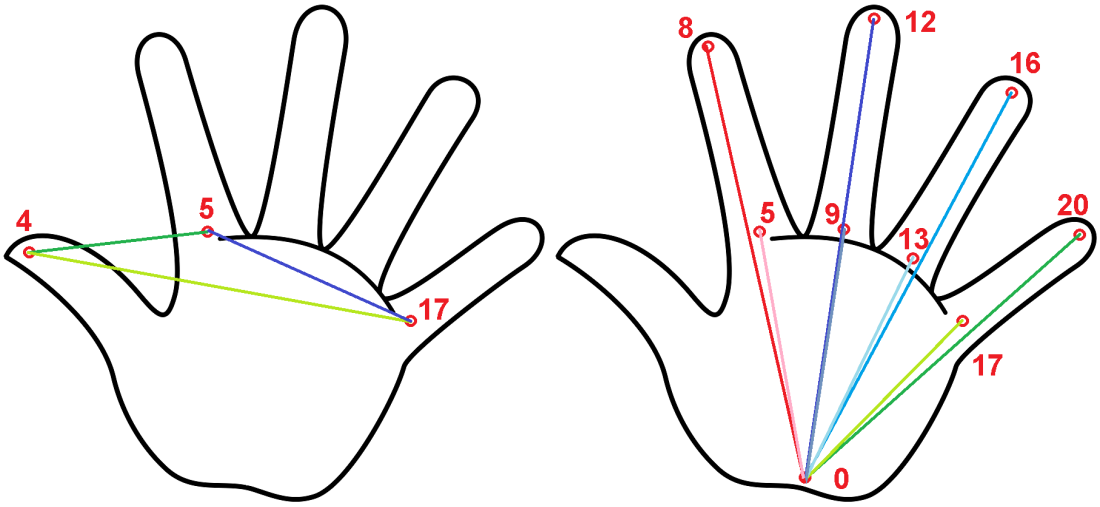


Figura 4.2: Puntos de referencia

En las figuras anteriores aparecen números y aristas. Los números corresponden con el identificador del vértice del modelo tridimensional de *Hands*, ver figura 2.6, mientras que las aristas representan la distancia entre dichos vértices. Estas distancias son empleadas por los algoritmos 1 y 2 para determinar el estado de los distintos dedos de la mano.

El estado de los dedos: meñique, anular, corazón e índice se determina haciendo uso del algoritmo 1. Como se observa en el algoritmo, se considera que un dedo está «estirado»

Algorithm 1: Obtener estado de un dedo

```
procedure getFingerState(handModel-> h, finger-> f)
    tip_distance ← distance(h.vertice[0], h.vertice[f.tip])
    mcp_distance ← distance(h.vertice[0], h.vertice[f.mcp])
    if tip_distance > mcp_distance then
        return F.EXTENDED
    else
        return F.CLOSED
end procedure
```

cuando la posición de la falange distal de un dedo está más alejado de la muñeca de lo que está el nudillo correspondiente, en el caso contrario se considera que el dedo está «contraído».

Algorithm 2: Obtener estado del pulgar

```
procedure getThumbState(handModel-> h, thumb-> t)
    pinky_distance ← distance(h.vertice[t.tip], h.vertice[t.pinky_mcp])
    hand_width ← distance(h.vertice[t.index_mcp], h.vertice[t.pinky_mcp])
    if pinky_distance > hand_width then
        index_distance ← distance(h.vertice[t.tip], h.vertice[t.index_mcp])
        if index_distance > hand_width then
            return F.EXTENDED
        else
            return F.RELAXED
    else
        return F.CLOSED
end procedure
```

El dedo pulgar emplea una estrategia similar pero considera puntos de referencia distintos y puede encontrarse en el estado *Relaxed*, contrariamente a los otros dedos. El pulgar se considera que está «contraído» cuando la falange distal se encuentra superpuesta con la palma, «estirado» si la distancia entre la falange distal y el nudillo del dedo índice es superior a la distancia entre los nudillos del dedo índice y meñique, en caso contrario se considera que el pulgar está «relajado».

4.3.2. Proximidad entre dedos

Representaremos la cercanía de un dedo con sus adyacentes haciendo uso de ángulos. Los ángulos considerados aparecen ilustrados en la figura 4.3 y son los siguientes:

- El ángulo i_a: es el ángulo que forma el dedo índice y corazón tomando como punto de origen la muñeca.
- El ángulo r_a: está formado el dedo anular y corazón.
- Y finalmente, el ángulo p_a: formado por el dedo meñique y el dedo anular.

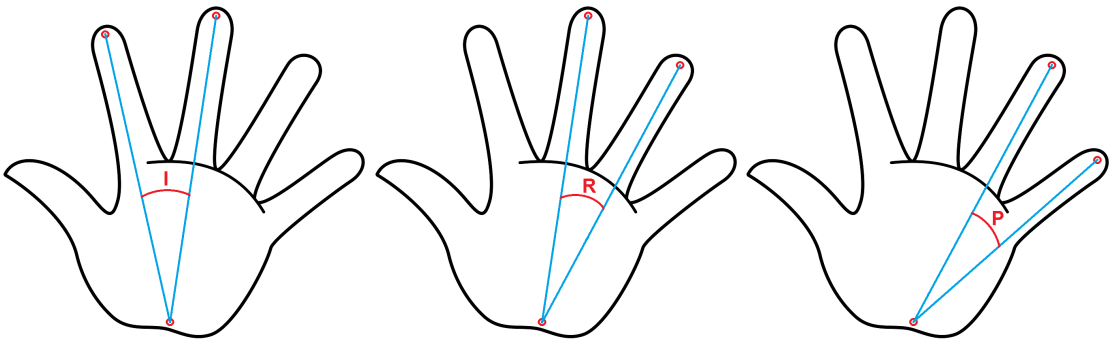


Figura 4.3: Ángulos de referencia

4.3.3. Transformación del modelo de *Hands*

Haciendo uso de las características anteriores, podemos transformar instancias del modelo de *Hands* en instancias de un modelo más abstracto, al que denominaremos "*Hand Profile*".

Sea *HandProfile(t, i, m, r, p, i_a, r_a, p_a)* el constructor del modelo dónde: t, i, m, r y p denotan $thumb, index, middle, ring$ y $pinky$, respectivamente y i_a, r_a y p_a son los ángulos que abstraen la orientación de los dedos.

Para cualquier instancia arbitraria H de *Hand Profile* se cumple que el dominio de las variables satisface la expresión 4.2.

$$t \in F \bigwedge i \in F \bigwedge m \in F \bigwedge r \in F \bigwedge p \in F \bigwedge i_a \in \mathbb{R} \bigwedge r_a \in \mathbb{R} \bigwedge p_a \in \mathbb{R} \tag{4.2}$$

Con el modelo *Hand Profile* podemos abstraer el modelo de *Hands* y recuperar instancias como las siguientes:

H_1(EXT, EXT, EXT, EXT, EXT, 2.38, 10.56, 4.10);
H_2(RLX, EXT, CLS, CLS, CLS, 3.66, 2.05, 1.98);

Donde *EXT*, *RLX* y *CLS* son abreviaturas de los estados de F.

Por cada fotograma en el que se identifique correctamente una mano obtendremos una instancia de *Hand Profile* semejante a las anteriores. Estas instancias son útiles como instantáneas pero no se pueden emplear para definir perfiles.

Supongamos que definimos un perfil denominado «mano abierta» usando la instantánea de la instancia H_1, sería prácticamente imposible de identificar porque estamos exigiendo que los ángulos i_a, r_a y p_a sean exactamente los de la instancia H_1.

4.3.4. Representación e identificación de un perfil

Para digitalizar un perfil, ampliaremos el poder expresivo del modelo *Hand Profile* definido en la sección anterior. La definición de perfiles debe ser lo más abstracta y configurable posible para ajustarse a las necesidades concretas del caso práctico. Por ello se propone:

- Definir un conjunto no vacío de valores que pertenezcan a F para cada una de las propiedades t, i, m, r y p.
- Definir un rango de valores para las propiedades i_a, r_a y p_a que permita delimitar el instante en el qué un gesto deja de identificarse con el perfil.

Sea *Profile(T, I, M, R, P, I_a, R_a, P_a)* el constructor del modelo de un perfil. Dónde T, I, M, R y P son subconjuntos no vacíos de F y I_a, R_a y P_a representan rangos de valores $[\alpha, \beta]$ definidos en $\mathbb{R}$.

Con este modelo podemos generar instancias de perfiles como la que se representa a continuación:

P_1({RLX, EXT}, {EXT}, {EXT}, {EXT}, {EXT}, [4.0, 10], [4, 8], [4, 8]);
P_2({CLS}, {EXT}, {CLS}, {CLS}, {CLS}, (-∞, +∞), (-∞, +∞), (-∞, +∞));

Gracias a estas instancias un programa podrá, a partir de un fotograma, identificar perfiles. Por cada fotograma, se recupera una instancia del modelo *Hand Profile*, sea H_x una instancia arbitraria de este modelo. Sea también P_x una instancia arbitraria del conjunto de perfiles definidos en un programa. Para determinar si H_x expresa el perfil especificado por P_x se hará uso del algoritmo 3.

Algorithm 3: Ajuste: mano - perfil

procedure HAND_MATCHES_PROFILE(H_x, P_x)

t_match $\leftarrow H_x.t \in P_x.T$

i_match $\leftarrow H_x.i \in P_x.I$

m_match $\leftarrow H_x.m \in P_x.M$

r_match $\leftarrow H_x.r \in P_x.R$

p_match $\leftarrow H_x.p \in P_x.P$

state_matches $\leftarrow t\_match \bigwedge i\_match \bigwedge m\_match \bigwedge r\_match \bigwedge p\_match$

i_a_match $\leftarrow P_x.I_a.\alpha \leq H_x.i_a \leq P_x.I_a.\beta$

r_a_match $\leftarrow P_x.R_a.\alpha \leq H_x.r_a \leq P_x.R_a.\beta$

m_a_match $\leftarrow P_x.P_a.\alpha \leq H_x.p_a \leq P_x.P_a.\beta$

orientaion_matches $\leftarrow i_a\_match \bigwedge r_a\_match \bigwedge m_a\_match$

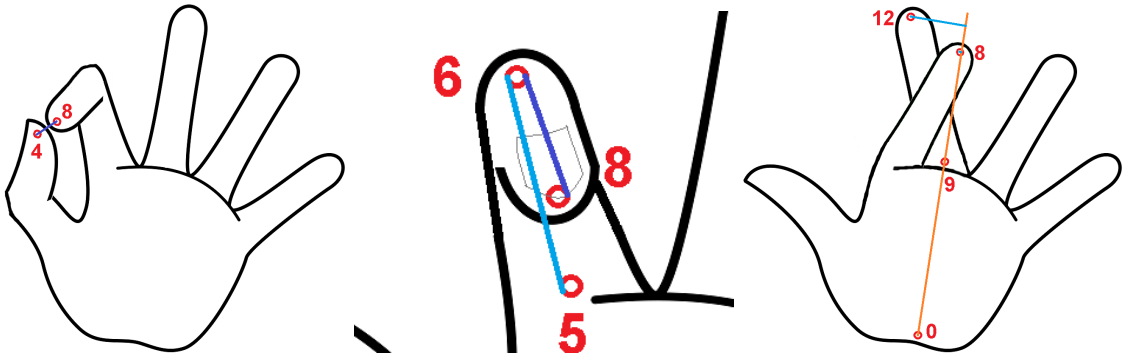
return state_matches $\bigwedge$ orientaion_matches

end procedure

4.4. Limitaciones del modelo propuesto

El modelo propuesto se ha elaborado con fines académicos y presenta limitaciones. Existen 48 configuraciones únicas de dedos para el modelo *Hand Profile* si consideramos exclusivamente el estado en el que pueden hallarse los dedos. Si bien el modelo puede ser enriquecido y extendido para mejorar su precisión y su poder expresivo, en esta sección del trabajo nos centramos en exponer las limitaciones que presenta.

El modelo actual impide representar y reconocer situaciones como las ilustradas a continuación:



(a) Puntas juntas (b) Dedo «encogido» (c) Dedos «superpuestos»

Figura 4.4: Limitaciones: configuraciones indistinguibles

Mejora del Poder Expresivo

Para extender el modelo y poder discriminar perfiles más variados se puede:

- Definir nuevos estados para el conjunto F.
- Definir nuevas estructuras de datos.
- Extraer métricas adicionales.

En las figuras anteriores: 4.4(a), 4.4(b) y 4.4(c) se representan las aristas que podrían emplearse para extraer características con las que identificar la configuración correspondiente.

4.4.1. Limitaciones de adaptabilidad

En lo referente a la adaptabilidad, el modelo propuesto hereda todas las limitaciones del algoritmo de ML y de la representación de ofrece *Hands*.

Esta forma de interacción no es adaptable a cualquier perfil de usuario. Se desconoce el comportamiento de ésta solución de ML en situaciones en las que se quiere identificar una mano y hacer un seguimiento de la misma cuando ésta presenta falta de dedos u otro tipo de lesiones y deformidades.

Para ofrecer un modelo que permita a este perfil de usuarios hacer uso de esta tecnología es necesario llevar a cabo estudios y experimentos que determinen cómo se comporta *Hands* en este tipo de escenarios. Y con ello, realizar las modificaciones que se consideren oportunas sobre los modelos definidos en éste Capítulo con el fin de ofrecer las opciones de adaptabilidad que se consideren correctas. En aquellos casos donde la acumulación de falta de dedos y/o deformidades presentes pone en compromiso la forma característica de una mano es altamente probable que los procesadores de imagen no la reconozcan.

4.4.2. Limitaciones de rendimiento

El propósito de este modelo es emplearse en escenarios dónde se capturan imágenes en vivo con el propósito de desencadenar acciones, por ello, a la hora de capturar fotogramas e identificar manos también intervienen los siguientes factores:

Factor *Hands*

Como se ha mencionado en la sección anterior, nuestro modelo hereda las limitaciones de *Hands*. La interacción con *Hands* presenta la siguientes limitaciones:

- La identificación de una mano ofrece resultados precisos cuando la palma de la mano mira a la cámara, en otros casos *Hands* puede proporcionar predicciones menos precisas o incluso no detectar nada.
- Si hay varias manos presentes en un fotograma puede no identificarse la mano deseada.

Factor Cámara

La cámara juega un papel fundamental puesto que es la que captura los fotogramas que serán analizados por *Hands*. Una baja resolución de imagen o imágenes comprimidas pueden dificultar la tarea de identificación y seguimiento de la mano.

Factor Entorno

En fotogramas donde la imagen que se captura contiene: muchos objetos, o en la que los colores capturados contrastan poco con el tono de piel de la mano, o bien por que no está lo suficientemente iluminado el entorno, pueden provocar que la predicción de *Hands* falle y se identifique una mano en una región errónea.

Factor Humano

Nuestro modelo opera en un escenario que exige a las personas interactuar, al menos, con una cámara. El sistema está expuesto, y es inevitable, a tratar con lecturas erróneas de la mano y a perderse durante el seguimiento de la misma.

El usuario puede sentirse inmerso en el programa con el que interactúa, puede distraerse y como consecuencia, olvida que en está interactuando con una cámara. Puede también gesticular fuera del campo de visión de la cámara, provocando que el sistema no capture sus ordenes. Con la tecnología actual, se exige además, que los movimientos de la mano sean limpios, suaves y lentos. El procesador de imágenes de *Hands* no ofrece resultados fiables si se gesticula a grandes velocidades.

Capítulo 5

Herramientas de análisis

Las herramientas que se presentan a continuación han sido desarrolladas con el lenguaje de programación *Python* v3.9.5. Se ha elegido esta tecnología por la agilidad y velocidad de prototipado que ofrece.

5.1. Objetivo

Con el fin de estudiar la viabilidad del modelo propuesto en el Capítulo anterior se han desarrollado herramientas auxiliares que automatizan los siguientes procesos:

- La recogida de muestras.
- El análisis de las muestras capturadas.
- La evaluación del modelo.

Estas herramientas tienen por objetivo agilizar el proceso de ajuste de parámetros de los distintos perfiles que queremos definir.

5.2. Generador de datos

El generador de datos es un programa destinado a la recolección de muestras. El programa, al igual que las otras utilidades que presentamos a continuación, debe ejecutarse por línea de comandos. El programa tiene por objetivo crear ficheros *csv* con la siguiente estructura:

- Cada fila representa una muestra.
- Cada fotograma que captura la cámara genera una muestra.

- Una muestra es una colección de datos de interés que se extraen de un fotograma.
- Todas las muestras del fichero se ajustan a un mismo perfil de mano.

La herramienta sigue el proceso ilustrado en el diagrama de flujo "*Data Generator*" de la figura 5.5. Se genera un fichero *csv* para cada uno de los perfiles especificados en la configuración. Previo a la recolección de muestras, la herramienta alerta al usuario para que se prepare para la sesión de muestreo. Posteriormente, la cámara empieza a capturar imágenes de las que se extraen las características del modelo propuesto. La figura 5.1 ilustra el comportamiento de la herramienta en la que se ha realizado el muestro de dos perfiles.

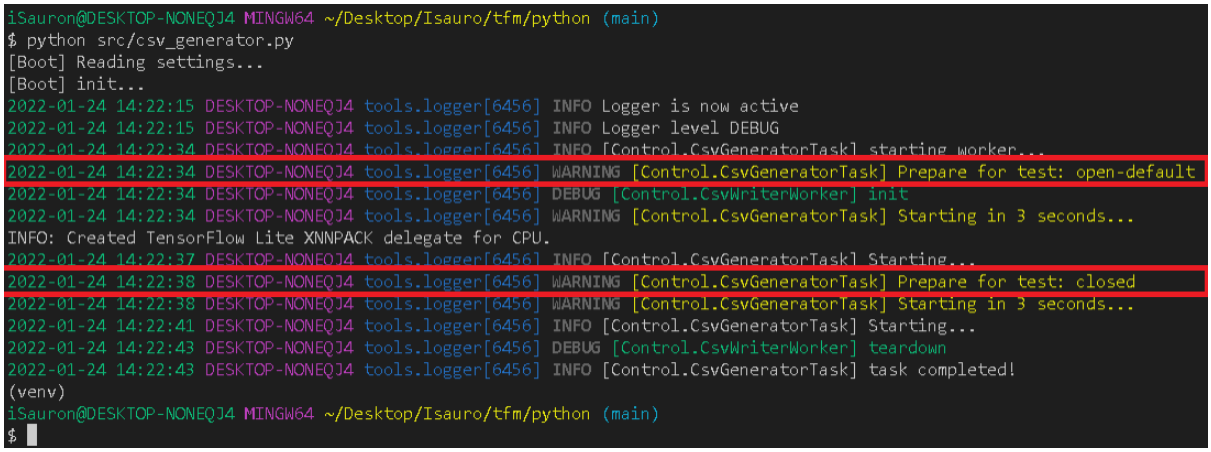


Figura 5.1: Herramientas *Mediapipe*: generador de datos

Es necesario que el usuario realice correctamente el gesto que especifica la herramienta. La corrección de los datos generados depende del usuario que participa en dicha sesión, puesto que los datos grabados en el fichero provienen de imágenes en vivo.

Los ficheros resultantes se almacenan en el directorio *out/generated* y se organizan por sub-carpetas que contienen los *csv*s asociados un mismo perfil.

5.3. Analizador de gestos

Esta herramienta tiene por objetivo analizar los ficheros de datos generados para cada uno de los perfiles y generar un informe en formato *PDF*. El diagrama de flujo "*Gesture Summary*" de la figura 5.5 ilustra el comportamiento de la herramienta.

Los informes generados se componen de tres secciones:

- Síntesis del muestreo.
- Distribución de los estados identificados para cada uno de los dedos.
- Informe estadístico de los ángulos considerados.

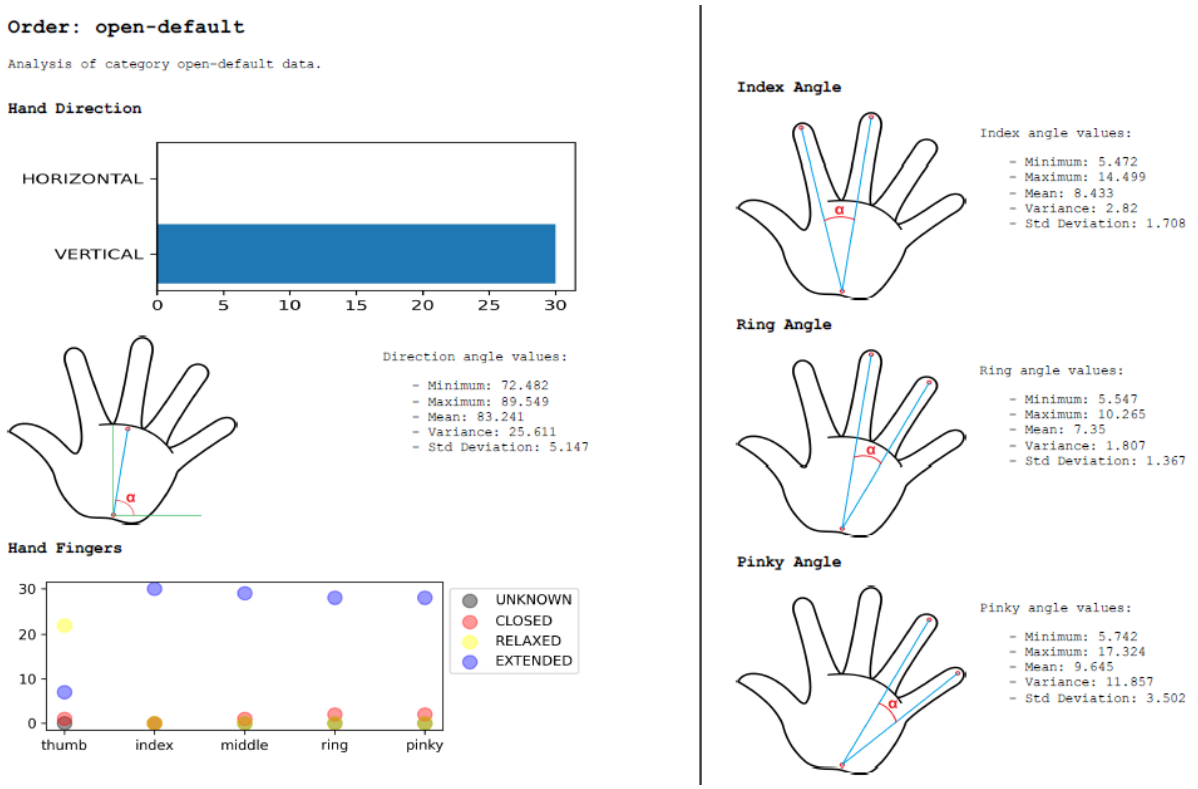


Figura 5.2: Herramientas *Mediapipe*: informe de Perfil

En la figura 5.2 se ilustra uno de los informes que se han generado para el perfil «mano abierta».

En la parte izquierda de la imagen se localizan las dos primeras secciones del documento: "*Hand Direction*" y "*Hand Fingers*". La primera sección cuantifica y perfila las muestras de las que se compone el informe, el propósito de recoger esta información es detectar sesgos en el muestreo. A continuación se muestra una gráfica que dibuja la distribución de los estado identificados en el conjunto de muestras. Como puede observarse, para el gesto «mano abierta» los dedos se encontraban mayoritariamente extendidos a excepción del dedo pulgar. Finalmente, en la parte derecha del documento se encuentra la última sección, que sintetiza y ofrece información estadística sobre los ángulos considerados en el modelo.

Es importante recordar que los datos del documento provienen de pruebas en vivo y, por lo tanto, están sujetas a múltiples fuentes de ruido y error como las descritas en la

sección 4.4. Por lo tanto, estos documentos deben ser interpretados con sentido común y espíritu crítico, son documentos que agilizan el proceso de análisis y la toma de decisiones. No por ello debe asumirse y considerar que la información que contienen es verídica y absoluta.

A continuación se ilustra en la figura 5.3 el comportamiento de la herramienta en ejecución. Gracias a estos informes se pueden modelar los perfiles muestreados y obtener una representación digital fiable del mismo. Por ejemplo, para el perfil «mano abierta» expuesto podríamos proponer la siguiente representación digital.

```
HandProfile {
    id: String
    restrictions: List[Restrictions]
}

OpenHand is a HandProfile {
    id: 'Open Hand'
    restrictions: [
        OneOf('thumb', [EXTENDED, RELAXED]),
        Equals('index', EXTENDED),
        ...
        GreaterThan('indexAngle', 4.0),
        Between('pinkyAngle', 4.0, 10.0),
        ...
    ]
}
```

```
iSauron@DESKTOP-NONEQJ4 MINGW64 ~/Desktop/Isauro/tfm/python/src (main)
$ python csv_analyzer.py
[Boot] Reading settings...
[Boot] init...
2022-01-24 15:11:34 DESKTOP-NONEQJ4 tools.logger[9496] INFO Logger is now active
2022-01-24 15:11:34 DESKTOP-NONEQJ4 tools.logger[9496] INFO Logger level DEBUG
2022-01-24 15:11:35 DESKTOP-NONEQJ4 tools.logger[9496] INFO [Control.CsvAnalyzeTask] starting worker...
2022-01-24 15:11:46 DESKTOP-NONEQJ4 tools.logger[9496] INFO [Pdf] generated for order "open-default" at: C:\Users\iSauron\Desktop\
Isauro\tfm\python\out\analysis\open-default.pdf
2022-01-24 15:11:49 DESKTOP-NONEQJ4 tools.logger[9496] INFO [Pdf] generated for order "closed" at: C:\Users\iSauron\Desktop\Isauro
\tfm\python\out\analysis\closed.pdf
2022-01-24 15:11:49 DESKTOP-NONEQJ4 tools.logger[9496] INFO [Control.CsvAnalyzeTask] task completed!
(venv)
iSauron@DESKTOP-NONEQJ4 MINGW64 ~/Desktop/Isauro/tfm/python/src (main)
$
```

Figura 5.3: Herramientas *Mediapipe*: analizador de datos

5.4. Identificador de gestos

Esta herramienta tiene por objetivo poner a prueba el modelo diseñado. Se alimenta de perfiles, como los definidos en la sección anterior, y nos permite observar, evaluar y validar el comportamiento del modelo en un entorno real. El diagrama de flujo "*Gesture Profiler*" de la figura 5.5 ilustra el comportamiento de esta herramienta.

El programa abre la cámara y graba indefinidamente hasta que sea interrumpido por el usuario. Por cada fotograma capturado, el proceso "*Match Profile*" hace uso del algoritmo 3 y comprueba por cada gesto definido si encuentra alguna definición para la cual el identificador devuelve un acierto.

A continuación, la herramienta muestra por terminal un mensaje con la información del gesto que se ha identificado, en caso de no devolverse un acierto en el proceso de identificación, se identifica el gesto como "*unknown*", del inglés, desconocido.

La figura 5.4 muestra la salida por terminal que proporciona la utilidad en ejecución. Como puede observarse, el programa muestra información sobre el gesto identificado (bordeado en rojo) y en qué dirección se está desplazando la mano (bordeado en amarillo).

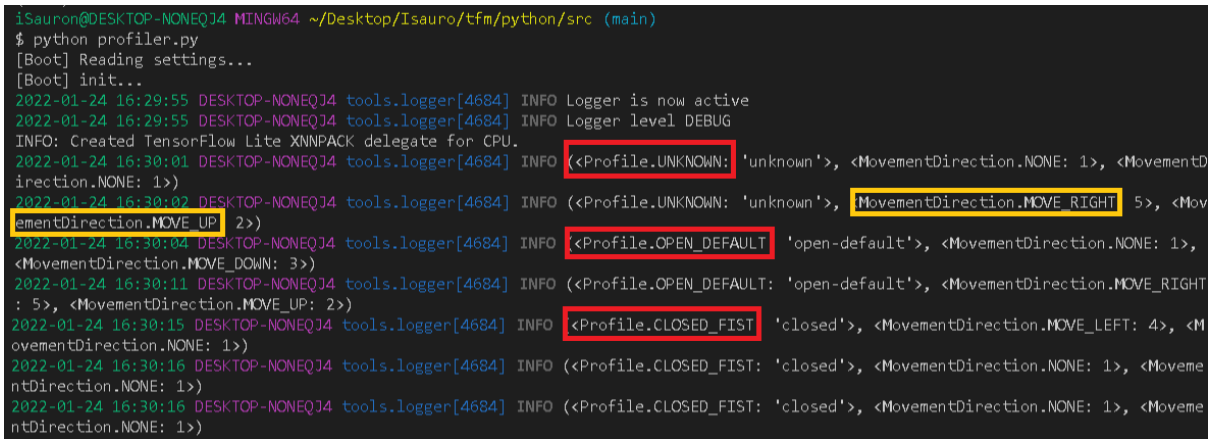


Figura 5.4: Herramientas *Mediapipe*: identificador de gestos

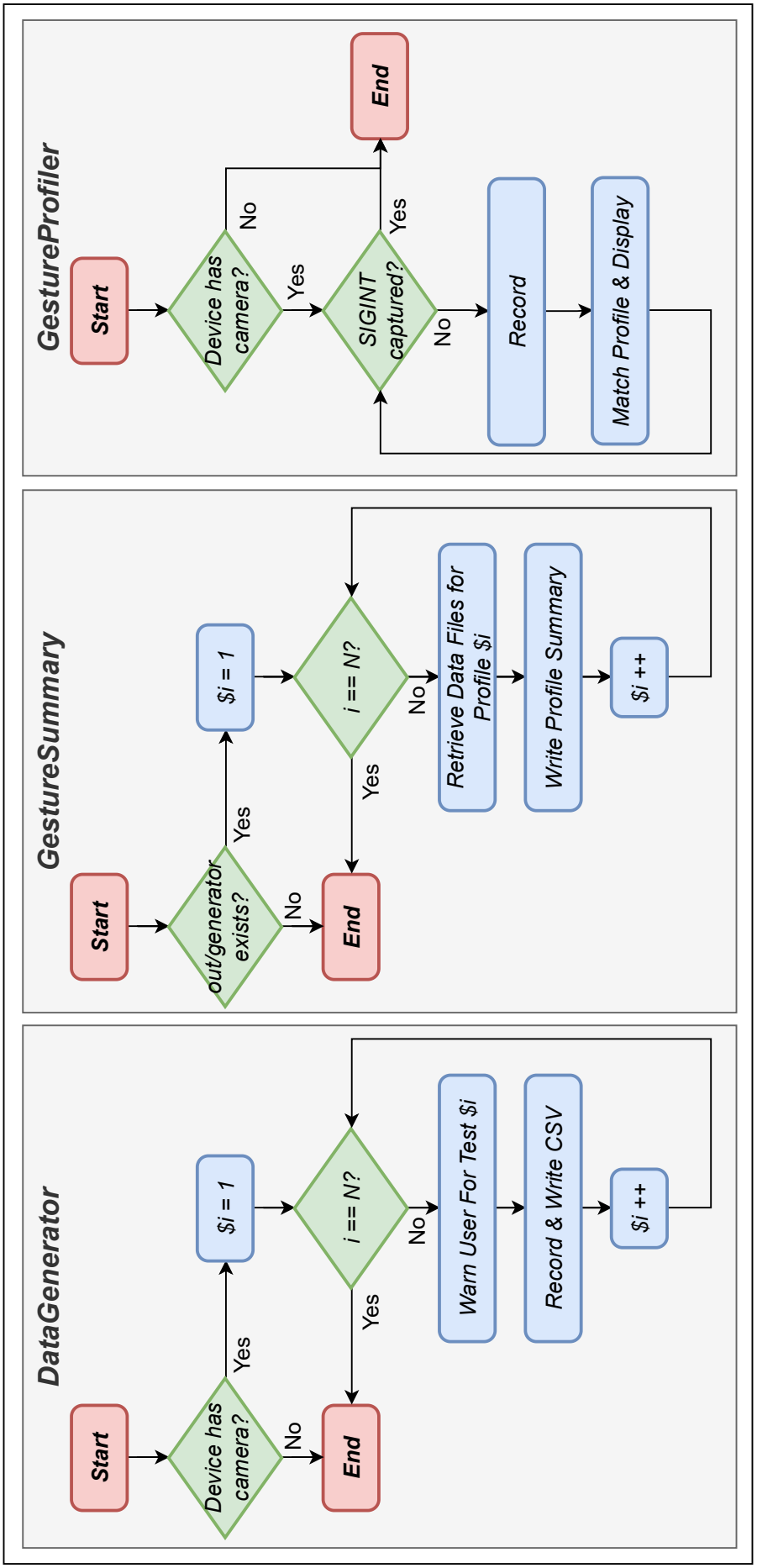


Figura 5.5: Herramientas *Mediapipe*: diagramas de flujo

Capítulo 6

Aplicación propuesta

6.1. Objetivos

Con el objetivo de probar y validar la eficacia del modelo presentado en un contexto real, se desarrollará una aplicación para evaluar su usabilidad con usuarios.

Inicialmente, describiremos los distintos escenarios y tecnologías involucradas que se han estudiado para elaborar el programa y justificaremos el diseño propuesto. A continuación, se describe con brevedad el proceso de adquisición del material necesario para la implementación de la herramienta. Posteriormente, se define formalmente el programa: los casos de uso y la arquitectura del mismo. Finalmente, se documenta el proceso de desarrollo, los problemas que se han encontrado durante el mismo y las correcciones realizadas.

6.2. Escenarios y tecnologías

En esta sección se presentan los diseños del sistema estudiados y se valoran distintos factores como son:

- La viabilidad del desarrollo
- El material necesario
- El entorno de uso

Al final se presenta una tabla comparativa que sintetiza las ventajas, desventajas y limitaciones identificadas en los diseños expuestos y se justifica el escenario que desarrollaremos.

6.2.1. Ordenador con cámara

El primer escenario planteado para elaborar el programa es el mismo que se ha utilizado para desarrollar las herramientas *Python* presentadas en el Capítulo anterior. En este escenario, ilustrado en la figura 6.1, el usuario se encuentra frente a un computador que dispone de cámara. El usuario interactúa con la herramienta gesticulando dentro del campo de visión de la cámara del ordenador.

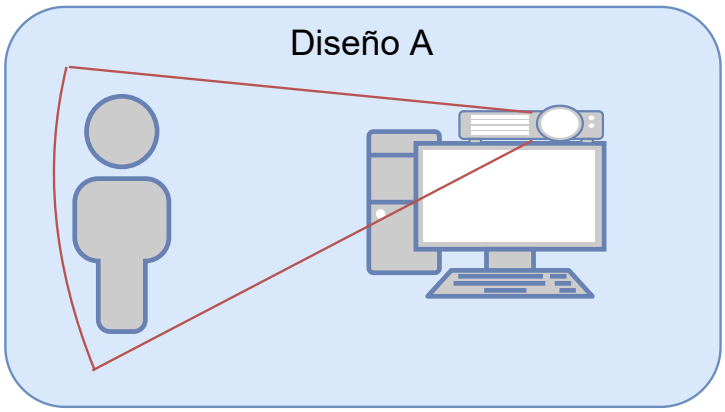


Figura 6.1: Diseño A: terminal con cámara

Este escenario destaca por su flexibilidad y versatilidad, pudiendo realizar desarrollos en multitud de lenguajes de programación compatibles con *Mediapipe*: *C++*, *Java*, *Python*, *Coral*, *JavaScript* y otros.

6.2.2. Proyección con estación de cámaras

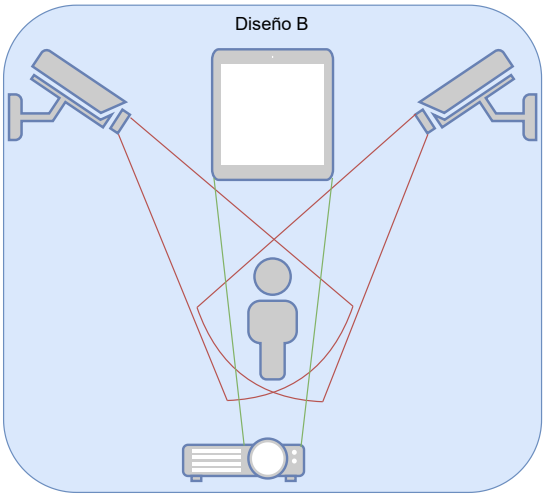


Figura 6.2: Diseño B: proyección

En el escenario ilustrado en la figura 6.2 el usuario se encuentra en un entorno observado por una o más cámaras. Éste interactúa con una aplicación que está siendo proyectada en una pantalla. Al igual que el escenario anterior, la aplicación está instalada en un computador que proyecta su pantalla por lo que a nivel tecnológico ofrece la misma flexibilidad.

El desarrollo se vuelve notablemente más complejo. El número de fotogramas incrementa proporcionalmente respecto al número

de cámaras. Los fotogramas que recogen las cámaras deben estar sincronizados y hay que analizar las predicciones todos ellos y determinar qué gesto se está realizando.

Este escenario, pese a comprender un coste y complejidad superior respecto al diseño anterior, ofrece al usuario mayor libertad de movimiento y mayor precisión a la hora de interactuar con la aplicación.

6.2.3. Proyección en gafas inteligentes

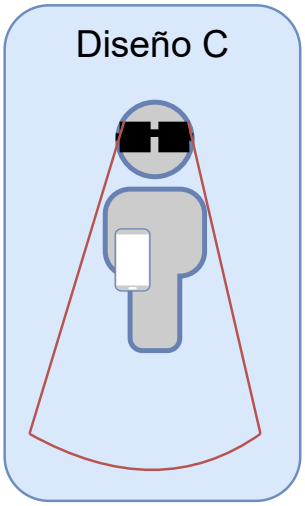


Figura 6.3: Diseño C: aplicación de AR

En este escenario, el usuario dispone de un teléfono móvil y unas gafas inteligentes que cuentan con una cámara integrada. El usuario puede interactuar con el aplicativo dentro del campo de visión de la cámara integrada de la gafa. Como se ilustra en la figura 6.3, el usuario debe llevar consigo el teléfono y las gafas en todo momento, contrariamente a los escenarios anteriores, este diseño ofrece una libertad de movimiento total para el usuario a costa de reducir considerablemente el espacio de interacción.

En lo que respecta al desarrollo móvil, *Mediapipe* ofrece soporte tanto para *iOS* como *Android*.

6.2.4. Selección y justificación

En la tabla 6.1 se sintetizan los principales factores de los escenarios anteriormente descritos.

El diseño A no se ha seleccionado por ser un escenario que «ancla» al usuario a un puesto fijo y por ser un trabajo repetitivo. Es un desarrollo similar a los ya realizados para elaborar las utilidades de *Mediapipe* presentadas en el Capítulo anterior. El diseño B se descarta por su alta complejidad, la estimación de tiempo de desarrollo y prueba supera el tiempo del que se dispone para realizar este trabajo.

	Volúmen de Interacción	Movilidad	Soporte Tecnológico	Coste Material	Coste Desarrollo
Diseño A	medio	bajo	alto	bajo	bajo
Diseño B	alto	media	alto	alto	alto
Diseño C	**bajo**	**alta**	**limitado**	**alto**	**medio**

Tabla 6.1: Comparativa de diseños planteados

Se desarrollará el diseño C por la ilusión personal de querer aprender, trabajar y experimentar con tecnología AR y por el desafío tecnológico que representa. El diseño C es el **escenario más limitante** de los propuestos:

- El volumen de espacio dónde la interacción es posible es reducido, ver figura 6.4.
- Los recursos *hardware* y el poder de cómputo de los dispositivos móviles es más limitado que los de un ordenador.
- El usuario cuenta con mucha movilidad, esto conlleva a que las imágenes que captura la cámara de la gafa provendrán de entornos muy variados, con mayor o menor iluminación y el sistema debe ser capaz de identificar manos y gestos.
- Es necesario investigar qué gafas inteligentes van a emplearse y definir qué tecnología va a usarse para el desarrollo.

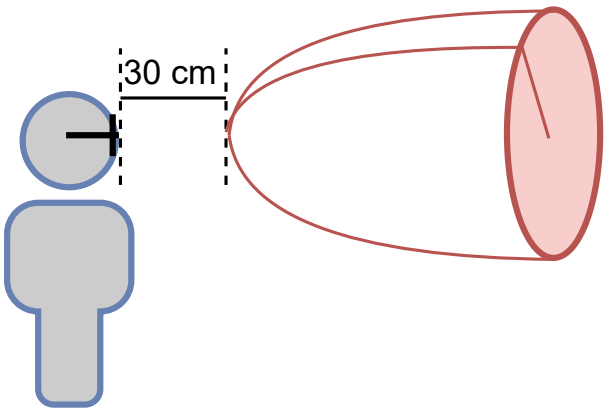


Figura 6.4: Volúmen de interacción

6.3. Adquisición del material

Previo al diseño, debe concretarse qué material va a emplearse, es capital conocer sus prestaciones y qué tecnologías están al alcance a la hora de desarrollar. En primer lugar

se hizo un estudio de mercado en el que se han recopilado dispositivos de SM compatibles con dispositivos móviles, con cámara integrada y que permitan visualizar gráficos.

Los modelos estudiados son:

- *Glass Enterprise Edition* de *Google*.
- *X2* de *ThirdEye Gen*.
- *Moverio BT-35e* de *Epson*.

Las gafas seleccionadas para realizar el desarrollo son las *Moverio BT-35E* por ser uno de los modelos más asequibles del mercado y por la extensa y detallada documentación técnica que ofrece *Epson* en su página web `https://tech.moverio.epson.com/en/bt-35e/`.

6.3.1. *Moverio BT-35e*

En las figuras 6.5(a) y 6.5(b) se muestra la SM *Moverio BT-35e* y su unidad de control.

(a) Gafas BT-35e (b) Interfaz de la gafa BT-35e

Figura 6.5: Gafas Inteligentes *Moverio BT-35e*

Siguiendo las indicaciones del fabricante, para emplear la SM con dispositivos móviles necesitamos hacer uso de los interfaces 1 y 2, resaltados en recuadros rojos en la figura 6.5(b). El interfaz 1 corresponde con el cable de la SM el cual comunica la gafa con la unidad de control. El interfaz 2 es un cable tipo *USB C*, el cual sirve como fuente de energía y canal bi-direccional de envío de datos, ver figura 6.6.

Las tablas 6.2 y 6.3 sintetizan la información de interés que proporciona el fabricante en su documentación oficial `https://tech.moverio.epson.com/en/bt-35e/developers_guide/introduction.html`.

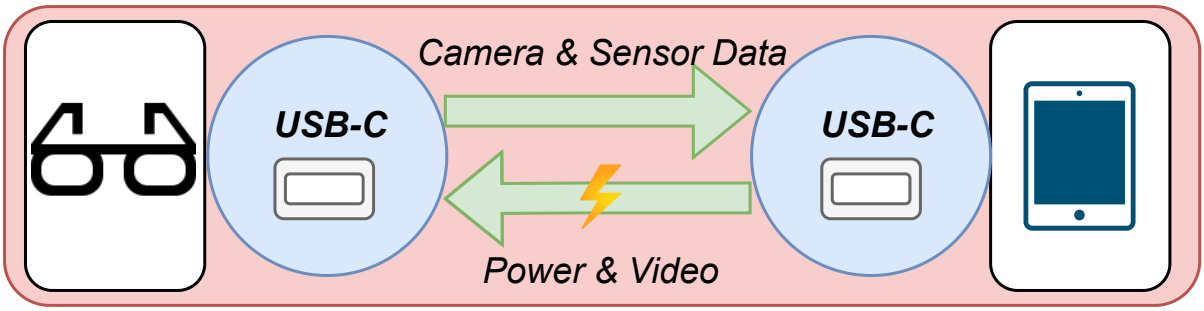


Figura 6.6: Puerto USB-C de Moverio BT35E

	BT-200	BT-300	BT-350	**BT-30C** **BT-35E**
Android SDK	✓	✓	✓	
Moverio SDK				✓

Tabla 6.2: Compatibilidad de modelos *Moverio*

Las gafas *BT-35e* no ofrecen retro-compatibilidad con modelos *Moverio* anteriores (descatalogados). Esto implica que no podemos hacer uso del *Android SDK* para programar con la gafa, éste Kit de Desarrollo Software (SDK) se emplea en el desarrollo de aplicaciones de XR en el entorno *Unity*, un motor de desarrollo de videojuegos.

	BT-30C	**BT-35E**
Acelerómetro	✓	✓
Campo magnético	✓	✓
Giroscopio	✓	✓
Luz ambiente		✓
Gravedad	✓	✓
Acelerómetro Lineal	✓	✓
Rotación	✓	✓
Captura de Imágenes		✓
Captura de Vídeo		✓

Tabla 6.3: Dispositivos de modelos *Moverio*

Al escoger las gafas *Moverio BT-35e*, heredamos todas las limitaciones y restricciones que implemente el SDK de *Moverio*. Este SDK está disponible hasta la fecha exclusivamente

para las plataformas *Android* y *Windows*.

Cable Usb-C a Usb-C con Soporte para Vídeo

El fabricante especifica que el cable *USB-C* que se conecta al interfaz debe tener el soporte Display Port Alternative Mode (DP Alt Mode) si se quiere proyectar imagen sobre la gafa o capturar imagen o vídeo de la gafa.

Dicho de otro modo, si usamos un cable *USB-C* a *USB-C* que **no** ofrezca este soporte, solamente podremos leer datos de los sensores integrados en la gafa (acelerómetro, giroscopio, ...). El cable no puede transmitir vídeo y por lo tanto no tendremos acceso a la cámara ni al visor de la gafa.

Teléfono con Puerto *Usb-C* y Soporte para DP Alt Mode

Puesto que no se disponía de un teléfono para desarrollar que contase con este soporte, se buscaron dispositivos móviles *Android* que tuvieran éste soporte y se compró el modelo más barato, un *Huawei P20*.

6.3.2. Restricciones y limitaciones del sistema

Haciendo uso de este modelo de SM y considerando el escenario seleccionado, nos atenemos a las siguientes restricciones:

- Plataforma de destino: dispositivos *Android* con versión de *API* de la plataforma: *Android* v28 e inferiores.
- Sólo para dispositivos móviles con soporte DP Alt Mode.

6.4. Entorno de desarrollo

6.4.1. Tecnología a emplear

Para el desarrollo de este aplicativo se utilizará el programa *Android Studio*, concretamente, la versión ***android-studio-arctic-fox-202031-stable*** que hace uso del gestor ***Gradle:3.1.1***.

Este gestor ofrece herramientas y utilidades que automatizan y agilizan el proceso de implementación, construcción e instalación de aplicaciones móviles en simuladores y teléfonos móviles. Se hará uso del lenguaje de programación ***Java 1.8*** y del **SDK versión 27 de *Android***.

6.4.2. Dependencias software

Para desarrollar este aplicativo haremos uso de las siguientes utilidades:

- **Moverio SDK**: esta dependencia se utiliza para interactuar con el *firmware* de la gafa inteligente *Moverio BT-35E*.
- **mediapipe**: esta librería nos ofrece las soluciones propuestas por *Google* con la que podremos identificar manos.

Para incluir las dependencias anteriormente listadas en el aplicativo debemos de indicarle al gestor *Gradle* dónde puede encontrarlas, para que las descargue y las integre automáticamente, para ello es necesario agregar la siguiente configuración:

```
dependencies {
...

// Moverio BT-35E
implementation files('libs/MoverioSDK_1.0.1.aar')

// Mediapipe
implementation 'com.google.mediapipe:solution-core:latest.release'
implementation 'com.google.mediapipe:hands:latest.release'
}
```

La descarga del fichero *MoverioSDK_1.0.1.arr* está disponible en la página web del fabricante: `https://tech.moverio.epson.com/en/bt-35e/download.html`.

6.5. Casos de uso

En este desarrollo, nuestro foco de atención será la interacción persona/máquina. La figura 6.5 ilustra los casos de uso más relevantes de nuestro sistema. Se desarrollará una aplicación que permita realizar los casos de uso anteriores. Todos ellos comparten

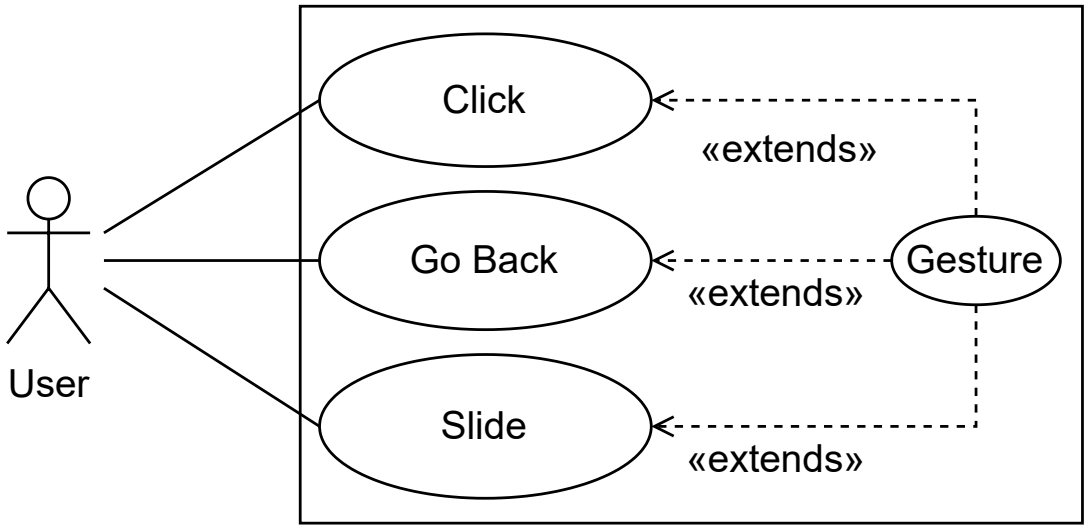


Figura 6.7: Casos de Uso

que son gestos, es decir, acciones que tiene el usuario a su disposición para interactuar con el aplicativo. A continuación, definimos con más detalle la función de cada uno de ellos:

- *Click*: selecciona e interactúa con elementos de la interfaz gráfica de un aplicativo.
- *Go Back*: se emplea para cerrar menús, abandonar la pantalla actual y regresar a la pantalla anterior.
- *Slide*: está acción sólo está disponible en aquellas pantallas que muestran listas de elementos. Permite deslizar la pantalla horizontalmente para visualizar los distintos elementos de una colección.

6.6. Diseño

Con los casos de uso identificados y definidos, debemos a continuación diseñar el aplicativo a desarrollar. En esta sección se presenta en primer lugar el diseño de la interacción persona/máquina y, posteriormente, una maqueta prototipo del programa. Posteriormente, se justifica el diseño propuesto así como la realización de los casos de uso definidos en la sección anterior.

6.6.1. Interacción

Para que la interacción persona/máquina sea realizable debemos identificar un conjunto de «perfiles» con los que definir «flujos de interacción».

En el marco de nuestro trabajo, definimos un perfil como la unidad mínima de interacción persona/máquina. Y un flujo de interacción, como una secuencia de perfiles que deben realizarse en cadena para ejecutar una orden. En la figura 6.8 se ilustra el conjunto de perfiles que identificará el programa, siendo el último de ellos el **perfil desconocido** (situado en la esquina inferior derecha).

Este perfil, como su nombre indica, actúa a modo de comodín y representa todos los perfiles posibles. Dicho de otro modo, la herramienta identificará éste perfil cuando el usuario no sitúa alguna de sus manos dentro del campo de visión de la cámara, cuando la mano que es capturada no se ajusta con ninguno de los perfiles definidos y cuando un perfil interrumpa un flujo de interacción.

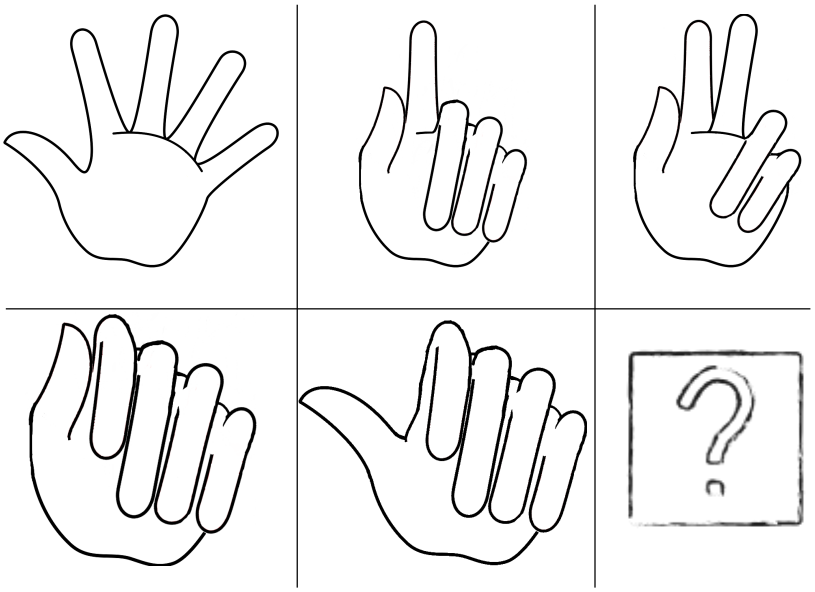


Figura 6.8: Perfiles de mano reconocibles

Una vez definidos los perfiles, diseñamos los flujos de interacción. Estos se muestran en la figura 6.9 dónde adicionalmente, se han asociado con la orden que emite dicho flujo de interacción. Una vez diseñado el aplicativo, estos flujos de interacción nos permitirán justificar la realización de los casos de uso.

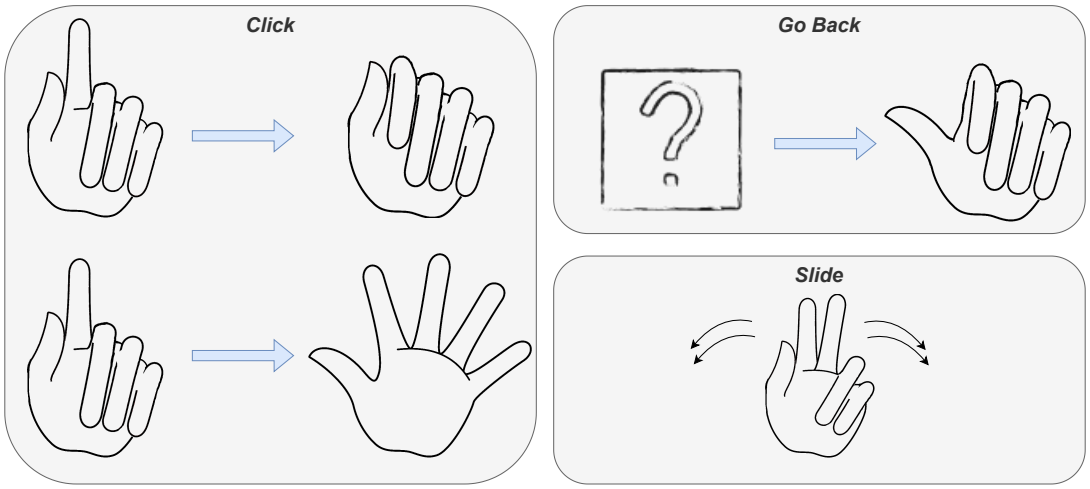


Figura 6.9: Flujos de interacción definidos

El programa identificará un perfil por cada fotograma capturado, por lo que es necesario definir un algoritmo para gestione el continuo flujo de fotogramas y perfiles que se identifican. Por este motivo, se ha diseñado la máquina de estados que se ilustra en la figura 6.10.

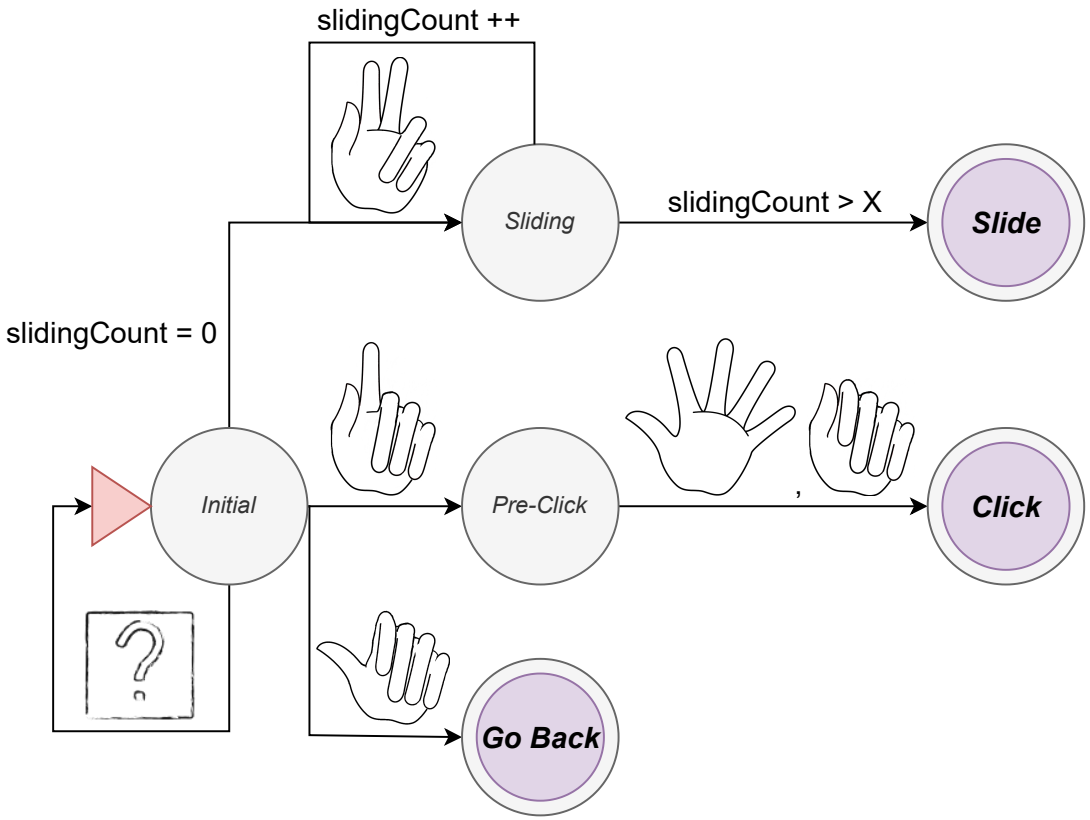


Figura 6.10: Máquina de estados

La máquina de estados ilustrada anteriormente es finita y determinista, se han eliminado transiciones de estados que dificultan la comprensión y claridad del diagrama. Todas aquellas transiciones que no aparecen ilustradas en el diagrama de estados retornan al estado inicial. Es decir, para alcanzar un estado final debe realizarse exactamente la secuencia de perfiles diseñada, en cualquier otro caso, la máquina de estados regresa a su estado inicial.

Para alcanzar el estado final *Slide*, la máquina de estados deberá identificar de forma ininterrumpida el perfil ilustrado durante X fotogramas. Esto permite que la máquina de estados recoja una cantidad de muestras con la que determinar en qué dirección (izquierda o derecha) debe deslizar la pantalla.

6.6.2. Aplicativo

En esta sección se propone el desarrollo de la aplicación ilustrada en la figura 6.6.2. Esta aplicación contará con:

- Un menú de opciones desde el que poder ajustar parámetros de configuración del visor así como el volumen de salida del puerto de audio integrado de la gafa.
- Una galería de elementos tematizada con ilustraciones y narraciones acerca de la cultura de la comunidad indígena *Uaurá*.

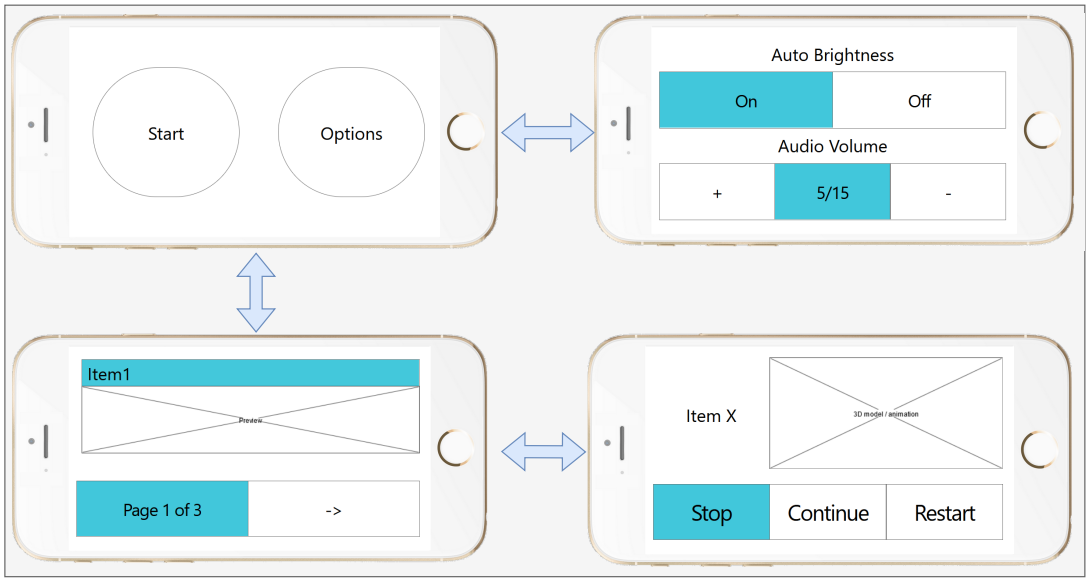


Figura 6.11: Aplicación Propuesta

Las pantallas no cuentan con botón de retroceso, puesto que ya existe un flujo de interacción que cubre dicha acción. Los *wireframes* de la parte inferior corresponden con las pantallas de la galería y, aunque dispongan de botones, se permite el uso de la acción

Slide para cambiar el elemento que se visualiza.

Se ha optado por diseñar un interfaz de usuario minimalista con botones grandes y espaciados para facilitar la interacción con estos. En la figura 6.6.2 se muestra el diseño propuesto y el sistema de interacción en juego, justificando así que el diseño propuesto cubre todos los casos de uso especificados. Nuestro foco a continuación es materializar estos diseños desarrollando una aplicación móvil con la que poder evaluar, entre otras, las decisiones de diseño que hemos tomado.

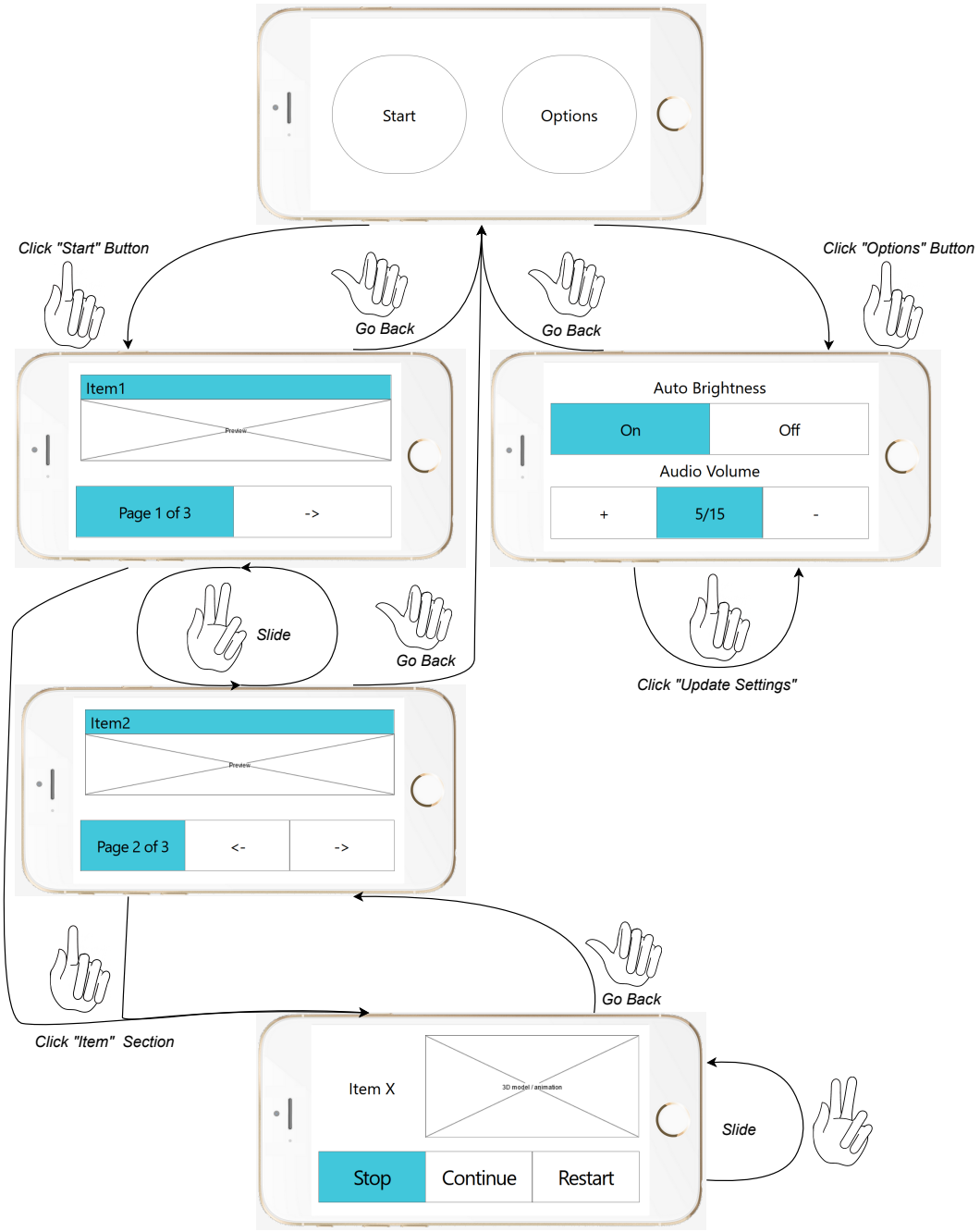


Figura 6.12: Realización de casos de uso

6.7. Arquitectura

En esta sección del trabajo se describen una o más configuraciones físicas sobre las cuales se puede realizar el despliegue del software y dónde puede ejecutarse, así como la infraestructura necesaria para la instalación del mismo.

En la figura 6.13 se ilustra el escenario general de distribución esperada para el aplicativo a desarrollar. Las características de los nodos y la comunicación entre los mismos se describe en la sección de adquisición del material 6.3.2.

Figura 6.13: Nodos del Sistema

Los nodos de nuestro sistema son:

- Las SM *Moverio BT-35e*.
- Un dispositivo móvil *Android* con soporte DP Alt Mode, en nuestro caso, un *Huawei P20*.

La vista lógica de nuestro sistema se ilustra en la figura 6.14, la aplicación resultante del desarrollo se instala en el nodo *Huawei P20*. Para que el sistema opere con normalidad se debe asegurar la conexión con el interfaz del nodo *Moverio BT35e*.

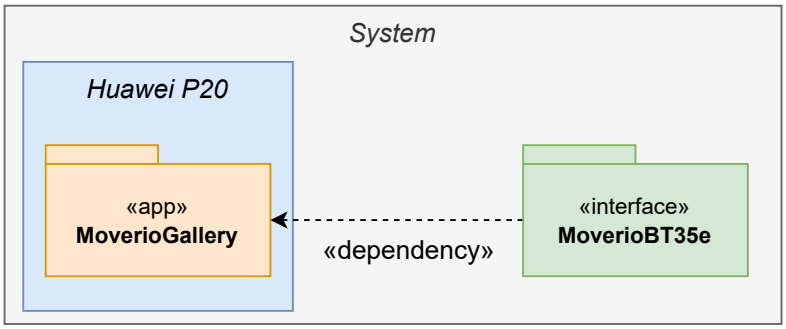


Figura 6.14: Arquitectura del Sistema

6.8. Desarrollo

En esta sección del trabajo abarcamos el desarrollo de la aplicación propuesta. Se describe brevemente cómo fue el proceso de desarrollo, las dificultades que se han afrontado y las decisiones tomadas para solventarlas.

6.8.1. Desarrollo del prototipo

En un primer lugar se desarrollaron dos aplicaciones móviles independientes a modo de prototipo:

- La primera accede a la cámara del teléfono, analiza los fotogramas e implementa la máquina de estados presentada en la figura 6.10.
- La segunda aplicación se conectaba a la interfaz de *Moverio* y se utilizó para aprender cómo integrar en *Android* el interfaz de *Moverio*.

Esto se hizo así porque cuando empezó el desarrollo no disponíamos de las SM *Moverio BT35-e*. Una vez tuvimos todo el material necesario para montar el sistema, se comenzó a desarrollar el segundo prototipo descrito anteriormente. Tras verificar el correcto funcionamiento del interfaz, se inició el desarrollo de la aplicación.

Para desarrollar la aplicación se reutilizaron los módulos desarrollados en los prototipos iniciales, no obstante el desarrollo no pudo completarse.

6.8.2. Fallos de diseño detectados

A la hora de integrar ambos prototipos aparecieron errores asociados con las versiones del *API* de *Android*. Tras investigar estos errores, se observó que existe una incompatibilidad entre las dependencias de *Moverio SDK* y *Mediapipe*. Ocurre que *Moverio SDK* opera con la versión 28 e inferiores del *API* de *Android*, no obstante *Mediapipe* opera con la versión 30 y superiores.

Se experimentaron distintas configuraciones de proyecto. Se observó que un proyecto configurado para generar un ejecutable usando el *API* 32 de *Android* compila bien ambas dependencias, no obstante el interfaz de *Moverio* no responde adecuadamente cuando se ejecuta el programa. Se experimentaron distintas configuraciones a lo largo de dos semanas sin éxito.

Por estas circunstancias y por falta de tiempo, se abandonó la idea de desarrollar una única aplicación. Se estudiaron alternativas para solventar la dificultad de integrar ambas

dependencias. Las alternativas propuestas fueron:

- **Alternativa A**: encapsular las funciones del módulo de análisis de gestos en una librería compilada con el *API* 32 y embeberla en un proyecto compilado con *API* 27 que se integre con *Moverio*.
- **Alternativa B**: descargarse el código fuente de la librería *Mediapipe*, eliminar las dependencias conflictivas para que pueda ser compilada con el *API* 27.
- **Alternativa C**: desarrollar dos aplicaciones que se comuniquen utilizando una base de datos común. Una de ellas opera con el *API* 27 y se conecta con el interfaz de *Moverio* y la otra aplicación opera con el *API* 32 y se encargaría del procesamiento de imágenes.

Alternativa A

Fue rápidamente descartada por dar los mismos problemas que el desarrollo inicial.

Alternativa B

Fue necesario instalar Windows Subsystem para Linux (WSL) porque la dependencia de *Mediapipe* es fácilmente compilable desde un sistema *Linux*. Adicionalmente, se instaló *Bazel*, un software que automatiza la creación y testeo de aplicaciones. Este software es imprescindible para compilar *Mediapipe* puesto que para el desarrollo de *Mediapipe* se hace uso de esta herramienta.

Tras estudiar el código fuente de *Mediapipe* se comenzó a borrar las dependencias de *androidx*, características por ser incompatibles con el *core* del *API* 27. Tras esto, se regeneró la librería haciendo uso de los siguientes comandos:

```
bazel build -c opt --strip=ALWAYS
    --host_crosstool_top=@bazel_tools//tools/cpp:toolchain
    --fat_apk_cpu=arm64-v8a,armeabi-v7a
    --legacy_whole_archive=0
    --features=-legacy_whole_archive
    --copt=-fvisibility=hidden
    --copt=-ffunction-sections
    --copt=-fdata-sections
    --copt=-fstack-protector
    --copt=-Oz
```

```
    --copt=-fomit-frame-pointer
    --copt=-DABSL_MIN_LOG_LEVEL=2
    --linkopt=-Wl, --gc-sections, --strip-all
    //mediapipe/examples/android/src/java/com/google/mediapipe/
       apps/aar_example:mediapipe_hand_detection.aar

bazel build -c opt
    mediapipe/modules/hand_landmark:
       hand_landmark_tracking_gpu_image
    --copt -DMESA_EGL_NO_X11_HEADERS
    --copt -DEGL_NO_X11
```

Se consiguió obtener una librería de *Mediapipe* compatible con *API* 27, no obstante el esfuerzo fue en vano por manifestarse errores en tiempo de ejecución que no dejaban trazas de error.

La falta de tiempo para investigar y dar con una solución precipitó el desarrollo de la alternativa C.

6.8.3. Corrección de la arquitectura

Previo a la experimentación de las alternativas anteriores, se conocía que la alternativa C daría solución al problema. Se quería evitar un desarrollo que involucrase dos aplicaciones por los problemas que se exponen más adelante.

Este cambio de diseño se refleja en la figura 6.15. La vista lógica de nuevo sistema involucra un nuevo aplicativo llamado *Mediapipe Backend*, este aplicativo es el encargado de procesar las imágenes que captura el aplicativo *MoverioGallery* y proporciona a esta las acciones a realizar. Esta interacción, inicialmente unificada en un único programa, se ha fragmentado en dos. El flujo de esta interactuación puede observarse en los diagramas de flujo 6.16, 6.17 y 6.18.

6.8.4. Impacto en el rendimiento

Los cambios realizados sobre la arquitectura del sistema tienen las siguientes consecuencias:

- Mayor consumo de batería: al haber dos aplicaciones en ejecución que se intercomunican, se consumen más recursos que con el diseño inicial.

- Mayor retardo en la ejecución de acciones.
- Caída del servicio de procesamiento de imágenes. Al haber un servicio que ejecuta en segundo plano, el aplicativo puede ser pausado o interrumpido por el sistema operativo *Android*, está sujeto a las políticas y configuraciones del teléfono.
- Sincronización entre el envío de fotogramas y la recepción de respuestas.

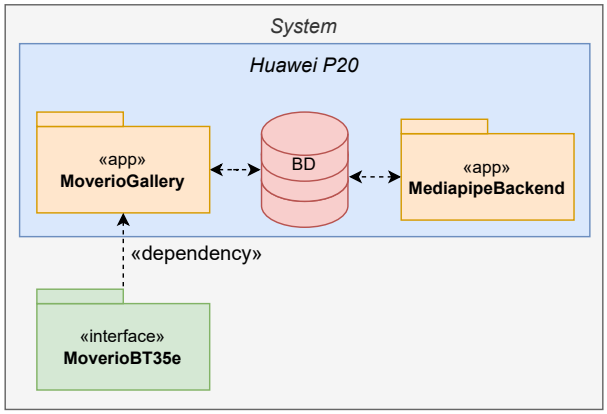


Figura 6.15: Nueva Arquitectura del Sistema

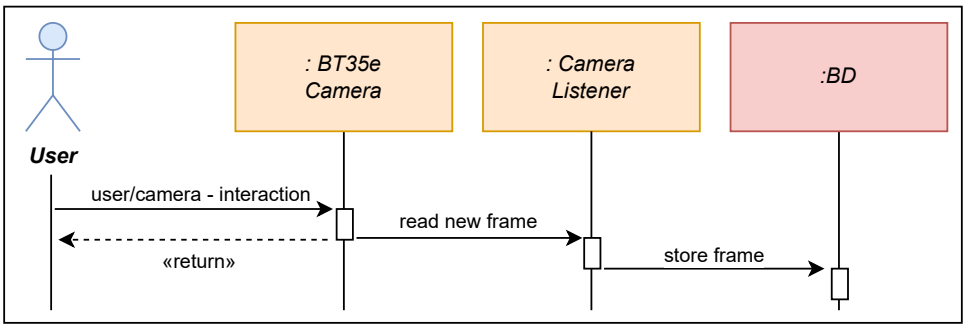


Figura 6.16: Almacenamiento de fotogramas

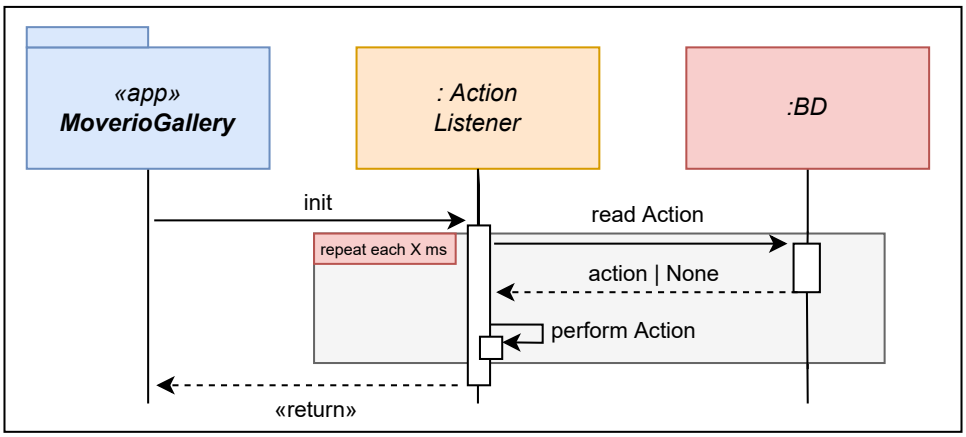


Figura 6.17: Ejecución de acciones

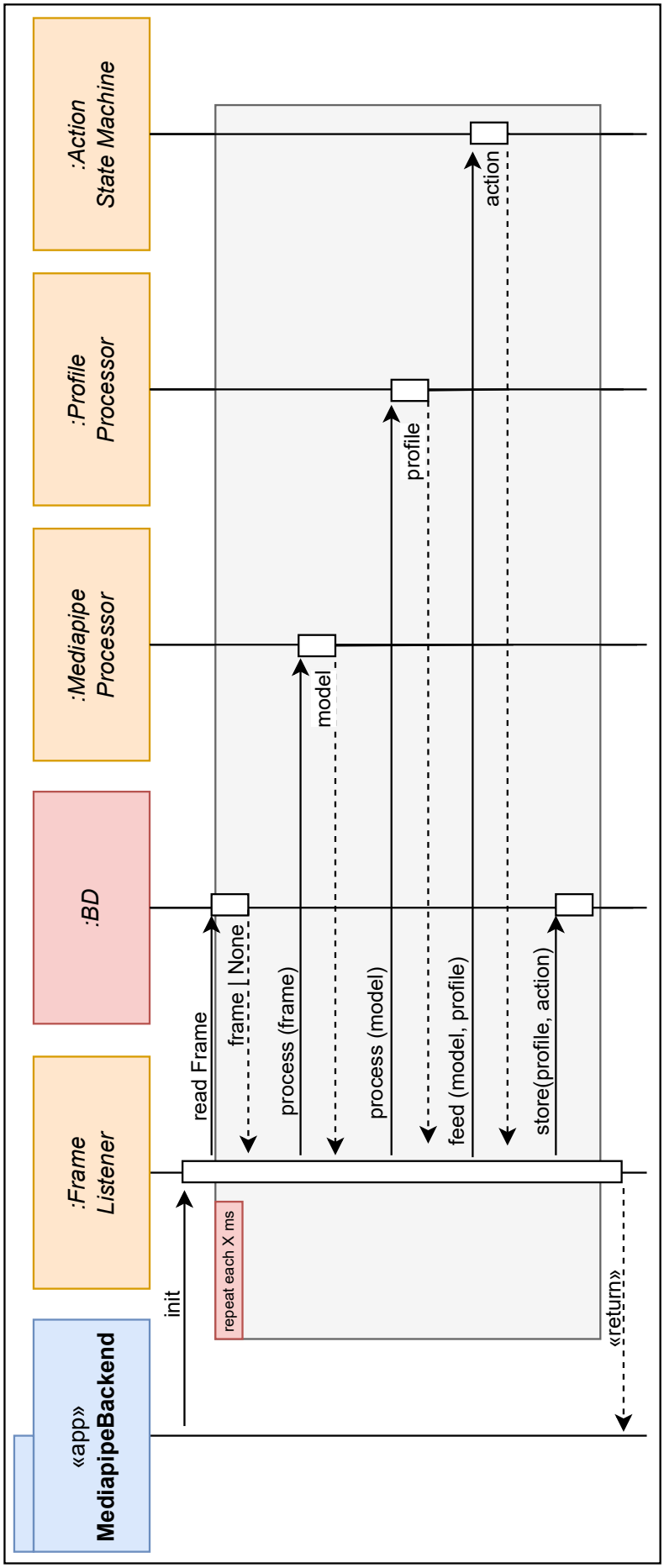


Figura 6.18: Procesamiento de fotogramas

Capítulo 7

Experimentos y resultados

En esta sección del trabajo presentamos los experimentos realizados. Por cada experimento realizado se describe en primer lugar, el objetivo y la dinámica del mismo. Posteriormente se presentan los resultados en forma de gráficas, métricas y observaciones. Para concluir, se hace un análisis crítico de los experimentos y el sistema construido en base a las observaciones y resultados obtenidos.

Los usuarios que se presentan voluntarios para experimentar nuestro trabajo realizan los experimentos en el orden en el que se presentan a continuación. De esta forma, garantizamos que los usuarios no prueban el aplicativo móvil de AR sin antes familiarizarse con el sistema de interacción diseñado, los gestos reconocibles y las acciones disponibles.

7.1. Identificador de perfiles

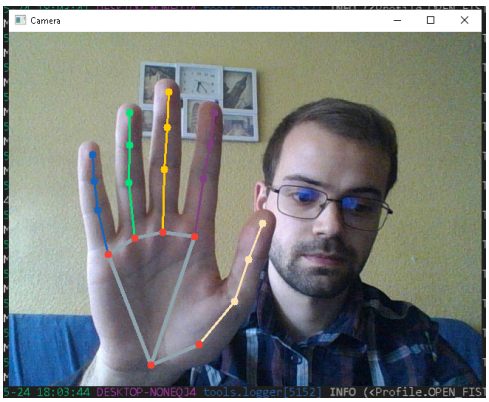


Figura 7.1: Fotograma procesado

Este primer experimento tiene por objetivo medir la efectividad del modelo a través de la herramienta presentada en la sección 5.4.

Se realiza en las condiciones descritas para el escenario 6.2.1. El programa abre una ventana en el escritorio desde la que pueden verse las imágenes procesadas, ver figura 7.1. El usuario interactúa con la herramienta situando su mano dentro del campo de visión de la cámara.

Se ha configurado la herramienta para que analice 10 Fotogramas Por Segundo (FPS), ésta toma un total de 150 muestras por perfil, por lo que la sesión de muestreo para cada perfil dura aproximadamente 15 segundos. En este periodo de tiempo el usuario traslada y rota su mano sin salir del campo de visión de la cámara. Si la mano abandona el espacio de interacción, la herramienta pausa el experimento hasta que regrese.

El experimento se ha llevado a cabo a una velocidad de 10 FPS con el objetivo de simular el rendimiento de máquinas con menor capacidad de computo.

La herramienta se emplea inicialmente para medir la tasa de aciertos para cada uno de los perfiles definidos, ver figura 6.8. Alternativamente, la herramienta cuenta con un modo tutorial, este se rige por un guión que exige al usuario realizar una secuencia de pasos que se detallan en la tabla 7.1, inicialmente el tutorial exige realizar correctamente cada uno de los gestos diseñados y, posteriormente, realizar las acciones interpretables por el sistema, ver figura 6.9.

Fase	Tarea	Descripción
1	Preparación	Cuenta atrás antes del inicio
2.a	Perfiles	Puño abierto
2.b		Índice extendido
2.c		Índice y corazón extendidos
2.d		Puño cerrado
2.e		Pulgar extendido
3.a	Acciones	Click
3.b		Deslizar
3.c		Retroceder
4	Resultados	Resumen de aciertos y fallos

Tabla 7.1: Guión del Experimento 1

La herramienta evalúa el rendimiento del usuario contabilizando la tasa de aciertos y de fallos. Este experimento sirve como entrenamiento y toma de contacto con el intérprete de gestos para la fase de experimentación con el aplicativo móvil, presentada en la sección siguiente.

Permitimos que los usuarios que participan en los experimentos repitan este experimento tantas veces como lo consideren necesario. Obtendremos así una o más muestras por cada persona que ha participado, no obstante, a la hora de extraer conclusiones y calcular métricas sólo consideraremos una muestra por persona. Analizaremos el rendimiento obtenido en la primera interacción con la herramienta y, de aquellas personas que escogieron repetir el experimento, el rendimiento de su última interacción.

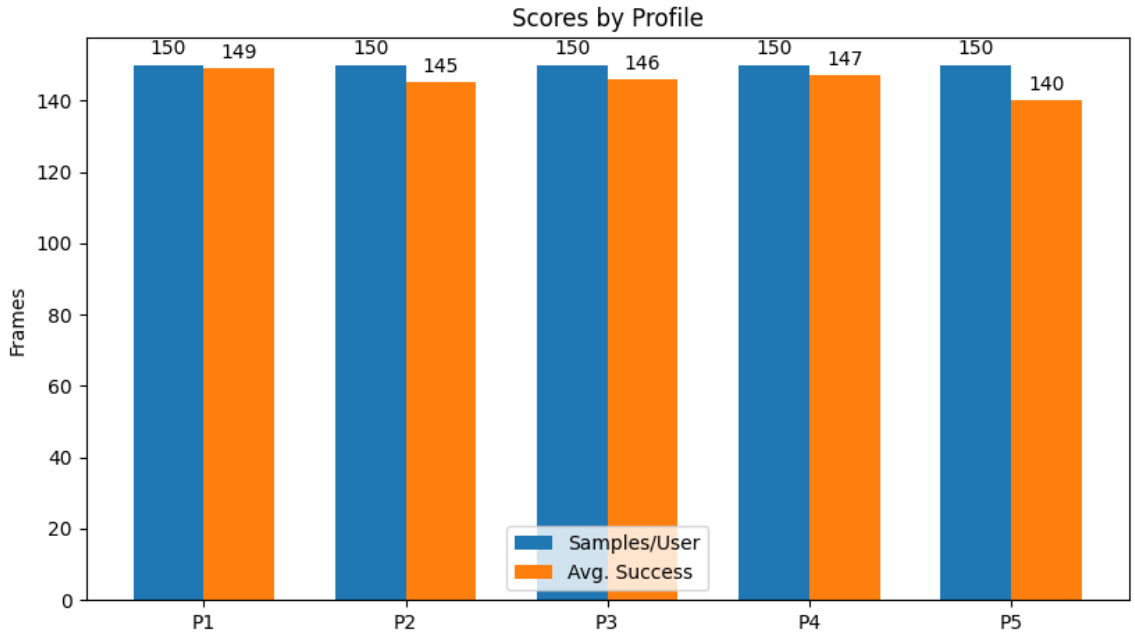


Figura 7.2: Tasa de Acierto de los Perfiles

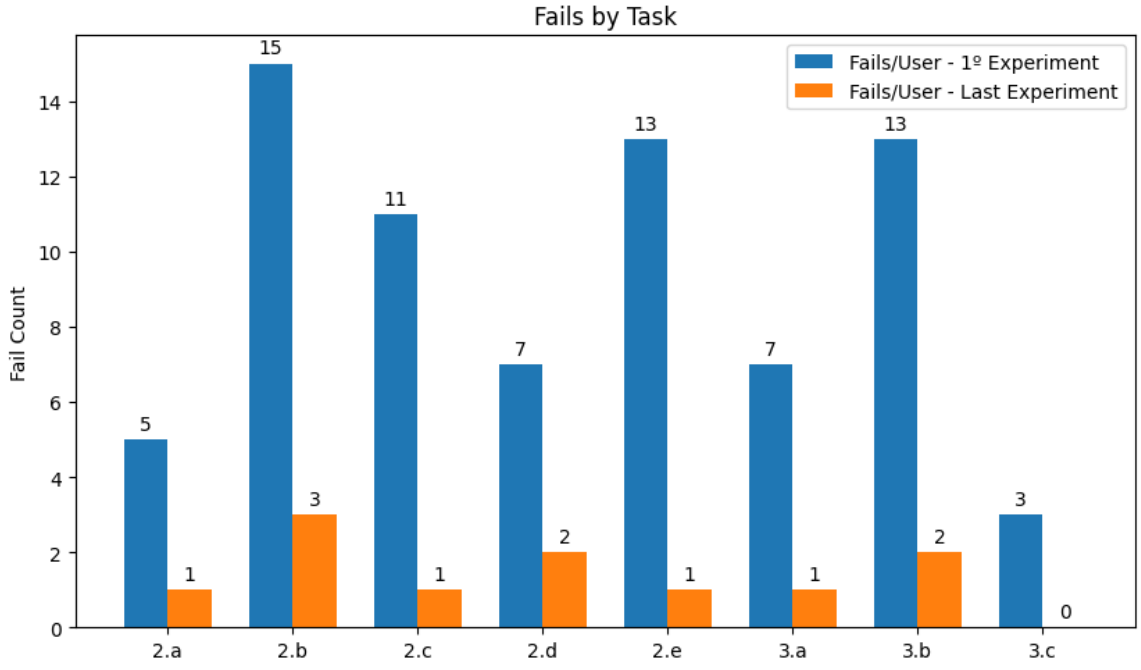


Figura 7.3: Fallos de los Usuarios

En este experimento se ha contado con la participación de 12 voluntarios. Las figuras 7.2 y 7.3 presentan los resultados de este primer experimento. La gráfica 7.2 pone en manifiesto la alta fiabilidad del sistema de identificación de gestos que, salvo en casos en los que se pierde el seguimiento de la mano, identifica con una tasa de acierto del 97 % el perfil especificado. Para la gráfica 7.3, el modo tutorial cuantifica los errores que cometen

los usuarios en cada uno de sus pasos. No obstante, la noción de «error» depende de la fase en la que se encuentre la herramienta.

- Fase 2: un error por cada 200ms en los que el usuario no realiza el perfil especificado.
- Fase 3: se contabiliza un fallo por cada acción incorrecta o por cada segundo en el que no se emite una acción. Para esto, la herramienta hace uso de la máquina de estados presentada en la sección 6.6.2.

Las barras azules ilustran los fallos cometidos por los usuarios en su primera interacción con la herramienta, la barra naranja contabiliza los fallos de los 3 participantes que quisieron repetir el experimento. De esta gráfica, llama la atención la gran tasa de errores. Esto se justifica, exceptuando las políticas empleadas para contar fallos, por la falta de experiencia de los participantes.

Se ha observado que una gran mayoría de los voluntarios olvidan las instrucciones de uso una vez iniciado el experimento. Sin embargo, esta gráfica es muy relevante porque también muestra que aquellas personas que han repetido el experimento han interiorizado las instrucciones y han aprendido a interactuar con el sistema diseñado.

7.2. Galería *Moverio BT35-e*

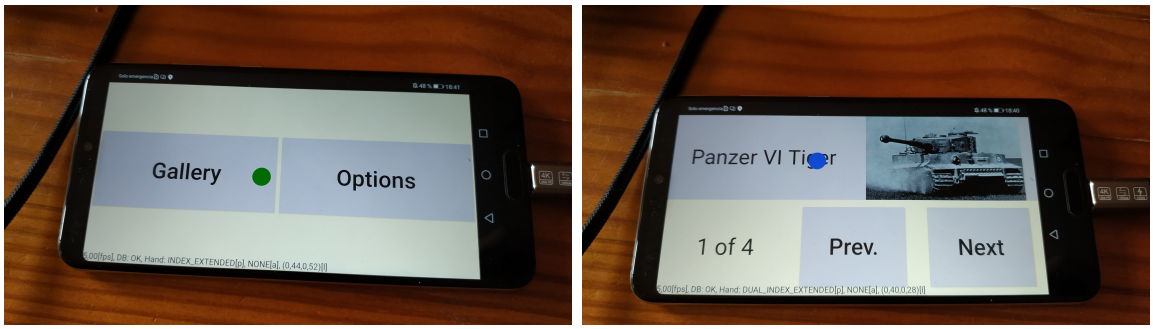
(a) Aplicación (b) Interacción

Figura 7.4: Galería *Moverio BT-35e*

En este experimento se ha hecho uso de la aplicación móvil desarrollada. Los usuarios equipan las SM *Moverio BT35-e* y disponen de tres a cinco minutos de prueba libre. Nuestro objetivo es estudiar la viabilidad del sistema de interacción propuesto para este escenario, evaluar la experiencia de los usuarios que han participado y analizar el rendimiento del sistema. Para ello hemos contado con la participación de 8 voluntarios.

Los datos recogidos en este experimento provienen de las siguientes fuentes:

- Las herramientas desarrolladas.
- Formularios con los que se ha recogido la opinión de los participantes.
- Apuntes realizados durante las sesiones de experimentos. En estos se contabilizan métricas como: fallos del sistema, intenciones del usuario, aciertos del usuario, etc. así como observaciones y comentarios del mismo.



(a) Menú (b) Galería

Figura 7.5: Aplicativo *Moverio BT-35e*

La figura 7.4 y 7.5 ilustra el sistema, el modo de interacción y el aplicativo desarrollado. La aplicación cuenta con un registro que captura todas las acciones y eventos que se producen cuando se hace uso del mismo. Esta información se contrasta, una vez finalizada la sesión de experimentos, con las observaciones recogidas durante el mismo. Se anota, entre otros datos, las acciones que el usuario pretende realizar y si tiene o no éxito.

Nº	Pregunta
1	Los gestos que identifica la herramienta son sencillos de aprender.
2	La herramienta identifica correctamente el gesto que indico.
3	Los punteros de colores me ayudan a asociar el perfil detectado y a realizar las acciones que quiero ejecutar.
4	La elección de colores del puntero me parecen adecuados.
5	La acción "*click*" es sencilla de realizar.
6	La acción "*slide*" es sencilla de realizar.
7	La acción "*go back*" es sencilla de realizar.
8	Se ejecutan acciones no deseadas cuando intento realizar cualquier acción. Por ejemplo, se ejecuta la acción "*go back*" cuando intento hacer "*click*" en un botón.
9	Noto cansancio en brazos u hombros tras realizar la prueba.
10	Noto cansancio ocular y/o mareos tras realizar la prueba.

Tabla 7.2: Encuesta del Experimento 2

La tabla 7.2 muestra la encuesta que se solicita a los participantes que rellenen una vez concluido el experimento. Los voluntarios responden a las preguntas dando una valoración del 1 al 5: dónde 1 representa «En desacuerdo» y 5 «De acuerdo».

Con motivo de los cambios de arquitectura presentados en la sección 6.8.3, el sistema opera a una velocidad de 5 FPS. A lo largo de las 7 sesiones de experimentos el sistema ha sido reiniciado un total de 3 veces.

La causa exacta del fallo se desconoce, se cree que el sistema operativo interrumpe el servicio en segundo plano. De ser así, esto puede guardar relación con el nivel de batería del dispositivo, al agotarse la batería el sistema operativo podría estar pausando o interrumpiendo servicios con el objetivo de alargar el tiempo de vida y uso del teléfono móvil.

Figura 7.6: Acciones Realizadas por los Usuarios

La gráfica 7.6 muestra el balance de las 136 acciones ejecutadas por el aplicativo a lo largo de las distintas sesiones de experimentos. Los «aciertos» agrupan a todas aquellas acciones ejecutadas por el sistema que figuran en los apuntes del anfitrión en el momento de la realización del experimento. Por lo tanto, un «fallo» es toda acción ejecutada por el sistema de la cual no se tiene registro: bien por distracción del anfitrión o bien por ser un falso positivo del módulo de análisis de gestos.

Las otras 21 acciones restantes corresponden a gestos que el aplicativo no ha registrado, pero sí figuran en las notas del anfitrión. Las causas de que estas interacciones no sean aciertos son: la perdida del seguimiento de la mano o que la maniobra tuvo lugar fuera del campo de interacción.

Las respuestas de las encuestas que rellenaron los voluntarios tras realizar los experimentos se ilustran en la gráfica 7.7. A continuación se discuten los resultados obtenidos de manera ordenada, para cada una de las preguntas de la tabla 7.2.

En primer lugar, una mayoría significativa considera que los perfiles y flujos de interacción diseñados son sencillos de aprender. No obstante, califican como negativa su experiencia con la herramienta. Los datos y las observaciones apuntan a que la causa es la frustración – las más de veinte interacciones que no han sido detectadas por el sistema sustentan esta hipótesis –. Por construcción, el campo de interacción de la gafa se encuentra alejado del individuo y es de reducido tamaño, exige mantener la mano estirada o la cabeza reclinada

para que se capture la mano. La mayoría ha experimentado algún tipo de cansancio en brazos u hombros tras la sesión.

Seguidamente, los usuarios han apoyado positivamente la inclusión de un puntero de colores que determina el perfil que está siendo identificado y valoran que los flujos de interacción son en su mayoría sencillos de realizar. Sin embargo, muchos usuarios han mostrado dificultades para realizar correctamente la acción «slide». Acorde con sus valoraciones, el flujo de interacción se confunde fácilmente con el de la acción «click». Este resultado es coherente con los datos extraídos del primer experimento en el paso 3.b, ver figura 7.3.

Finalmente, si bien con la gráfica anterior concluíamos que, aproximadamente, un 28 % de las acciones ejecutadas corresponden con fallos. La opinión de los voluntarios revela que una gran mayoría está en desacuerdo con la cuestión 8, lo que sugiere que, exceptuando las ocasiones en las que la gafa perdía el seguimiento de la mano, sentían control a la hora de interactuar con el aplicativo.

Ninguno de los voluntarios tiene experiencia previa con las SM *Moverio BT-35e* o modelos similares. Algunos de ellos han tenido dificultades para enfocar la imagen y han requerido de más tiempo para acomodar la gafa a sus necesidades.

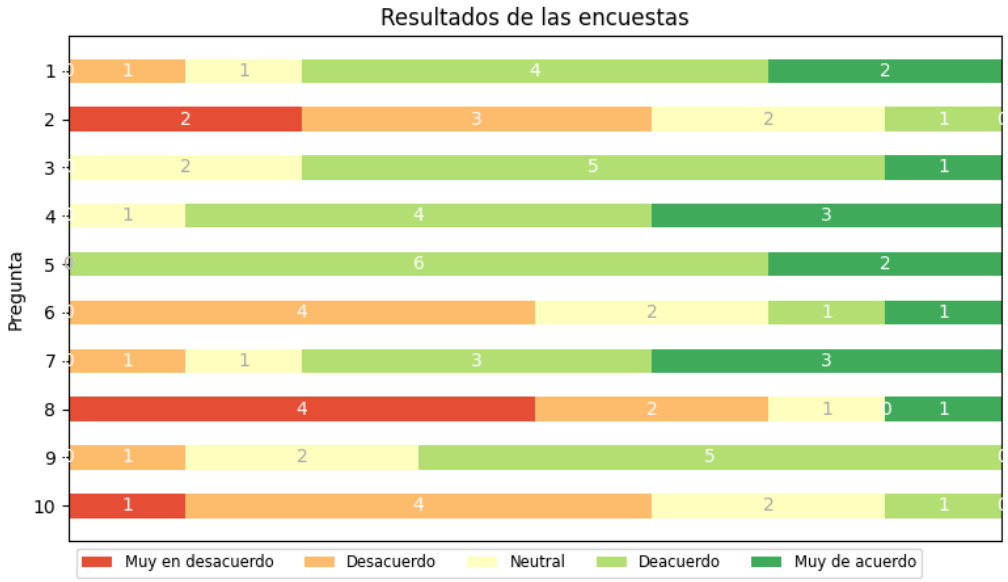


Figura 7.7: Resultados de las encuestas

Capítulo 8

Conclusiones y trabajo futuro

Tras estudiar los distintos artículos recopilados a lo largo de la fase de investigación, observamos que existe una gran variedad de técnicas y estrategias para interactuar mediante gestos con programas informáticos. El seguimiento de la mano puede capturarse haciendo uso de dispositivos especializados o a través del análisis de imágenes utilizando algoritmos del ML. Se optó investigar con la tecnología *Mediapipe*, fundamentalmente, por la amplia comunidad de desarrolladores y por su detallada documentación.

De entre todas las soluciones del ML que nos ofrece *Mediapipe*, se encuentra *Hands*. *Hands* proporciona un modelo 3D de la mano por cada mano que detecte en un fotograma. Basándonos en esto, se presentan las decisiones de diseño con las que se construye un modelo determinista con el que poder representar gestos de manera digital. El propósito del modelo es agilizar y simplificar la labor de diseño, abstrayendo el modelo 3D subyacente con características de alto nivel.

Especificado el modelo, se desarrollaron una serie de utilidades que buscan automatizar la labor de ajuste de parámetros de los perfiles de mano que se busca identificar. Nuestra motivación se centró en elaborar una aplicación de AR con el objetivo de evaluar el rendimiento del modelo con usuarios reales. Se han consideraron escenarios alternativos, hemos optado por la aplicación móvil de AR principalmente por: motivación personal y por ser también uno de los escenarios más limitantes a nivel tecnológico.

Para el desarrollo del aplicativo de AR se hizo un estudio de mercado en el que se contrastaron múltiples modelos de SM. Seleccionamos finalmente el modelo de gafa *Moverio BT-35e* por ser el más asequible del mercado. Los requisitos de esta gafa motivó la compra adicional de un *Huawei P20* y de un cable *usb-c / usb-c* con DP Alt Mode.

Se ha elegido desarrollar en uno de los escenario más limitantes a nivel tecnológico con el objetivo de demostrar la fiabilidad del sistema construido y su potencial portabilidad a

otros escenarios. Pese a las limitaciones tecnológicas y la multitud de problemas a los que nos hemos afrontado a lo largo de este trabajo, hemos logrado: definir un modelo sencillo para identificar gestos, diseñar un sistema de interacción basado en *mediapipe*, desarrollar utilidades y herramientas con las que poner a prueba nuestras hipótesis y modelos para finalmente, experimentar con usuarios.

Los voluntarios que han participado en los experimentos afirman que la curva de aprendizaje de los flujos de interacción diseñados es baja. Los datos y observaciones recogidas demuestran que, exceptuando los falsos positivos y las maniobras realizadas fuera del campo de interacción, los usuarios han manifestado sensación de control a la hora de interactuar con el aplicativo móvil.

El modelo diseñado es determinista y por lo tanto, no comete errores a la hora de identificar gestos. La evaluación del modelo por usuarios revela que la tasa de gestos correctamente identificados a la hora de interactuar es del 97 %. Este dato, en términos absolutos, supone una mejora del 2 % respecto al modelo de ML propuesto por *I.M. Harris y A.S. Agoes*[9]. No obstante, el resultado es considerablemente más significativo puesto que proviene de usuarios inexpertos.

8.1. Conclusiones

El modelo 3D que proporciona *Hands* ofrece un sinfín de posibilidades. La contribución de este trabajo presenta un modelo con el que se pueden definir los perfiles de manos que debe identificar una herramienta con una representación de alto nivel.

Contrariamente a lo que sucede con las alternativas que operan con algoritmos de ML en los que por cada perfil a identificar es necesario: elaborar o recurrir a un conjunto de datos, con lo que posteriormente entrenar al algoritmo de ML y finalmente, evaluar la calidad de sus predicciones.

Haciendo uso del modelo propuesto pueden definirse perfiles nuevos de forma ágil a través de un objeto de configuración. El modelo presenta limitaciones y no puede detectar: superposición de dedos y proximidad entre puntas de dedos entre otras. No obstante, esta estrategia agiliza enormemente el proceso de desarrollo, la prueba de aplicaciones y ofrece la posibilidad de personalizar dicha configuración de forma dinámica para ajustarse a las necesidades de usuarios particulares.

Los experimentos realizados y las valoraciones de los usuarios demuestran que el sistema de interacción diseñado es fiable y que los aplicativos desarrollados son herramientas usables.

8.2. Trabajo futuro

De cara al futuro, nuestro objetivo es mejorar el modelo y, con ello, construir una librería de gestos de código libre que pueda ser empleada para agilizar los procesos de diseño y desarrollo de aplicaciones basadas en la interacción con manos.

Como líneas futuras de investigación, podemos destacar las siguientes:

- Extender el modelo a fin de mejorar su poder expresivo y poder identificar, por ejemplo, las configuraciones presentadas en la sección 4.4.
- El modelo actual asume que sólo una mano va a estar presente en el campo de interacción. Se puede extender el modelo a fin reconocer configuraciones que exijan la intervención de dos o más manos.
- Basándonos en lo anterior, se puede adicionalmente, decorar el modelo con propiedades características que contengan información sobre las distintas manos que hay presentes, sus distancias, proximidad, etc. Eso sería de utilidad para definir flujos de interacción más elaborados.
- Se puede definir un modelo semántico que tenga por objetivo interpretar los gestos haciendo uso del modelo propuesto e información adicional del entorno. Esto podría ser de utilidad a la hora de interactuar con objetos 3D, ampliar una imagen, etc.
- Investigar qué políticas y estrategias deben emplearse en escenarios con más de una cámara para integrar el sistema de interacción.

Capítulo 9

Introduction

9.1. Motivation

The continuous development of technology and scientific advances allows, more and more, to materialize ideas and models that some time ago seemed impossible. This work was born from the illusion of experimenting with the technology of AR and the desire to develop one of the systems present in the universe of science fiction, a system of interaction with programs through gestures.

9.2. Context

This Master's Thesis has been carried out in collaboration with the Department of Software Engineering and Artificial Intelligence of the UCM.

9.3. Research purpose

In this paper we propose a deterministic model based on the solution of ML, *Hands*, proposed by *Google*. The objective of the work is to speed up the gesture identification process by using the proposed model. Thanks to this, it also seeks to speed up the design process of interaction systems with the programs. Therefore, applications have been developed: one for desktop and one for mobile devices, in order to evaluate the effectiveness of the proposed model as well as the user experience.

9.4. Work schedule

This section presents the work methodology followed and briefly documents the tasks performed.

9.4.1. Work methodology

A circular methodology based on the scientific method, illustrated in Figrue 9.1, has been used to develop this work. This methodology has been chosen for the flexibility and agility it offers in case of errors or bad decisions and for being the simplest and most efficient to implement in the framework of this work.

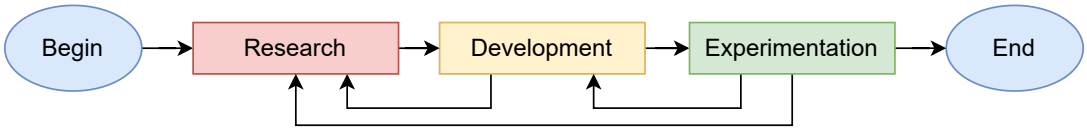


Figura 9.1: Work Methodology

The following is a brief description of the processes illustrated in Figure 9.1.

- **Research:** This phase comprises those tasks associated with the approach, discussion of hypotheses, the study and search of scientific articles and technology.
- **Development:** In the development phase we group all the tasks related to the design, programming and unit testing of the programs that are developed.
- **Experimentation:** The experimentation phase includes testing, experimentation with users and data collection and analysis.
- **Documentation:** Although it is not considered as a phase of the methodology presented, special mention is made of it because it is an extremely important task that lasts and is constant throughout the work.

9.4.2. Work repository

A remote *Git* repository has been used for version control of this work. The access link to the repository is as follows: `https://gitlab.com/iSauron/tfm`.

Regarding the *URL* of the repository, the Internet is alive and stuff moves. If you would like to obtain a copy of the contents and the web address is not available, please contact me and I will forward the contents of the paper to you.

9.4.3. Tasks and planning

Tasks completed

Table 9.1 shows the primary tasks of the job categorized by process. The numerous subtasks and stages into which they are broken down are not represented for the sake of clarity and comprehension.

ID	Name
1. Research	
1.a	State of the art study
1.b	Technology research on hand tracking technology
1.c	Selection and procurement of AR material
2. Development	
2.a	Prototypes for gesture detection
2.b	Application of AR
3. Experimentation	
3.a	Experimentation and gesture analysis with the prototypes
3.b	Experimentation with the application of AR with users

Tabla 9.1: Tasks Performed

Planning

Figure 9.4.3 outlines the time period consumed for the completion of this work. In this figure, the second title bar refers to the week of the year, i.e., the work started the week of 6/14/2021 and ended the week of 6/13/2022, with a period of inactivity between 9/27/2021 and 1/10/2022.

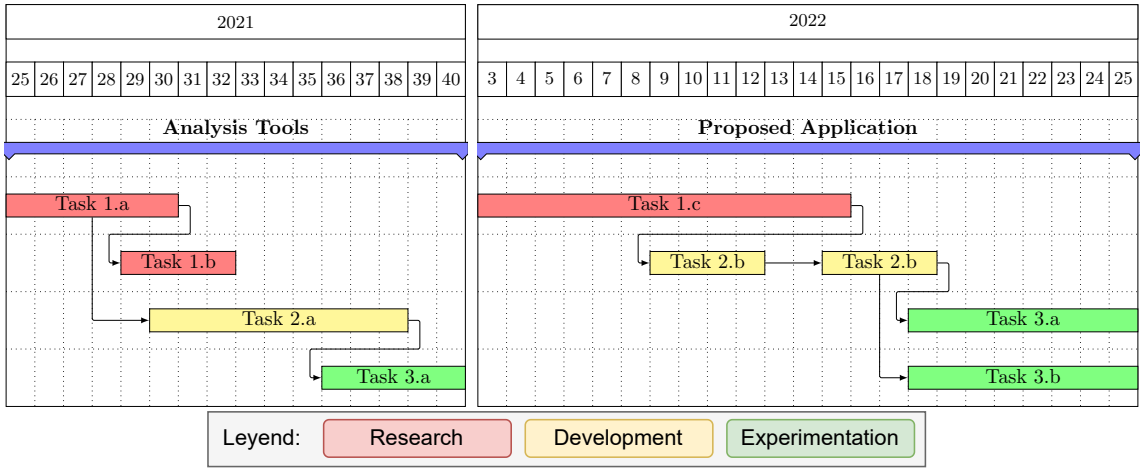


Figura 9.2: Gantt Diagram

9.5. Work structure

The remainder of the paper is organized into 9 chapters with the structure described below.

Chapter 2 briefly contextualizes and introduces the concepts, theoretical bases and technologies required to understand the framework. Specifically, it first introduces the XR technology and its variants, as well as their general characteristics and applications. Next, it elaborates on the SM devices, presents various models and discusses the characteristics they usually have as well as the uses to which they are put. Finally, the concept of "hand tracking" and the techniques used to identify hands in digital content are presented.

Chapter 3 presents the state of the art. In this chapter, articles, technologies and other works that share similarities with the project to be carried out are presented and the techniques used in their development for their realization are analyzed.

Chapter 4 presents the contribution of this work, a model that enables the digital definition of gestures for further processing.

Chapters 5 and 6 present the tools and programs that have been developed throughout this work to test the model designed in Chapter 4. Chapter 5 presents the tools developed to automate the gesture definition process. Chapter 6 presents the design and development process followed to elaborate the AR mobile application.

Chapter 7 describes the experiments carried out to evaluate the effectiveness of the model proposed in Chapter 4 and the user experience of the application presented in Chapter 6. The analysis and study of the data collected during the test sessions is also presented.

Chapter 8 concludes this work by briefly summarizing the project experience and analy-

zing the results of the experiments performed. Future work and possible lines of research are also presented.

Capítulo 10

Conclusions and future work

After studying the various articles collected during the research phase, we found that there is a wide variety of techniques and strategies for interacting with gestures with computer programs. Hand tracking can be captured using specialized devices or through image analysis using algorithms from ML. Research with the *Mediapipe* technology was chosen primarily because of the large developer community and detailed documentation.

Among all the solutions offered by *Mediapipe*, there is *Hands*. *Hands* provides a 3D model of the hand for each hand it detects in a frame. Based on this, design decisions are presented with which to build a deterministic model with which to represent gestures digitally. The purpose of the model is to speed up and simplify the design work by abstracting the underlying 3D model with high-level features.

Once the model was specified, a series of utilities were developed to automate the task of adjusting the parameters of the hand profiles to be identified. Our motivation focused on developing an application of AR with the objective of evaluating the performance of the model with real users. Alternative scenarios have been considered, we have opted for the AR mobile application mainly because of: personal motivation and because it is also one of the most technologically limiting scenarios.

For the development of the AR application, a market study was carried out in which multiple models of SM were contrasted. Finally, we selected the model of goggles *Moverio BT-35e* for being the most affordable in the market. The requirements of this goggle prompted the additional purchase of a *Huawei P20* and a *usb-c / usb-c* cable with DP Alt Mode.

It has been chosen to be developed in one of the most technologically limiting scenarios in order to demonstrate the reliability of the built system and its potential portability to other scenarios. Despite the technological limitations and the multitude of problems we

have faced throughout this work, we have managed to: define a simple model to identify gestures, design an interaction system based on *mediapipe*, develop utilities and tools with which to test our hypotheses and models and finally, experiment with users.

Volunteers who have participated in the experiments claim that the learning curve of the designed interaction flows is low. The data and observations collected show that, except for false positives and maneuvers outside the interaction field, users have shown a sense of control when interacting with the mobile application.

The model designed is deterministic and therefore does not make errors when identifying gestures. The evaluation of the model by users reveals that the rate of correctly identified gestures when interacting is 97 %. In absolute terms, is an improvement of 2 % over the ML model proposed by *I.M. Harris* and *A.S. Agoes*[9]. However, the result is considerably more significant since it comes from inexperienced users.

10.1. Conclusions

The 3D model provided by *Hands* offers endless possibilities. The contribution of this paper presents a model with which one can define the hand profiles to be identified by a tool with a high-level representation.

Contrary to what happens with the alternatives that operate with ML algorithms. In which, for each profile to be identified, it is necessary to: elaborate or resort to a set of data, with which to subsequently train the ML algorithm and finally, evaluate the quality of its predictions.

Using the proposed model, new profiles can be defined in an agile way through a configuration object. The model has limitations and cannot detect: overlapping fingers and proximity between fingertips among others. However, this strategy greatly speeds up the development process, the testing of applications and offers the possibility of dynamically customizing the configuration to suit the needs of particular users.

Experiments and user feedback show that the designed interaction system is reliable and that the developed applications are usable tools.

10.2. Future work

Looking ahead, our goal is to improve the proposed model and, with it, to build an open source gesture library that can be used to streamline the design and development processes of hand interaction based applications.

As future lines of research, the following can be highlighted:

- Extend the proposed model in order to improve its expressive power and to be able to recognize, for example, the configurations presented in section 4.4.
- The current model assumes that only one hand will be present in the interaction field. The model can be extended to recognize configurations that require the intervention of two or more hands.
- Based on the above, one can additionally decorate the model with characteristic properties containing information about the different hands that are present, their distances, proximity, etc. This would be useful for defining more elaborate interaction flows.
- A semantic model can be defined that aims to interpret gestures using the proposed model and additional information from the environment. This could be useful when interacting with 3D objects, zooming in on an image, etc.
- Investigate what policies and strategies should be employed in scenarios with more than one camera to integrate the proposed interaction system.

Bibliografía

[1] J. B. Sampson, "Intuitive engineering, human factors, and the design of future interfaces," in *Proceedings of the Cockpit and Future Displays for Defense and Security*, Proc. SPIE 5801, 2005.

[2] B. Coleman, "Using sensor inputs to affect virtual and real environments," *IEEE Pervasive Computing*, vol. 8, pp. 16 – 23, July 2009.

[3] J. H. S. Alizadehsalehi, A. Hadavi, "From bim to extended reality in aec industry," *Automation in Construction*, vol. 116, Aug. 2020.

[4] Statista, "Consumer mobile AR applications worldwide 2016-2022."

[5] C. G. Jesper Kjeldskov, "Human-computer interaction with mobile devices and services: A review of mobile hci research methods," *International Conference on Mobile Human-Computer Interaction*, p. 317–335, 2003.

[6] D. G. C. Kervegant, F. Raymond, "Touch hologram in mid-air," *Digital Library*, 2017.

[7] J. Y. L. Minseok Kim, "Touch and hand gesture-based interactions for directly manipulating 3d virtual objects in mobile augmented reality," *Springer Link*, 2016.

[8] L. W. Lin Jiang, Xiaoyang Yu, "A brief analysis of gesture recognition in VR," *Technical Papers: Touch and Interactive Displays*, 2020.

[9] A. S. A. Indriani, Moh. Harris, "Applying hand gesture recognition for user guide application using mediapipe," *Proceedings of the 2nd International Seminar of Science and Applied Technology*, 2021.

[10] G. LLC, "Hands - mediapipe."

[11] "Documentation | desarrolladores de android."

[12] Epson, "Developer's guide - documentation - BT-35e - technical information."

[13] S. Irshad and D. R. A. Rambli, "User experience evaluation of mobile AR services," in *Proceedings of the 12th International Conference on Advances in Mobile Computing*

and Multimedia, MoMM '14, pp. 119–126, Association for Computing Machinery, 2014.

[14] N. Bevan, "usability and user experience evaluation methods?."

[15] A. P. O. S. Vermeeren, E. L.-C. Law, V. Roto, M. Obrist, J. Hoonhout, and K. Väänänen-Vainio-Mattila, "User experience evaluation methods: current state and development needs," in *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, NordiCHI '10, pp. 521–530, Association for Computing Machinery, 2010.

[16] I. Reykjavik, *Proceedings of the international workshop on meaningful measures: valid useful user experience measurement*. Irit, 2008. OCLC: 496943454.

[17] N. I. Adhani and D. R. A. Rambli, "A survey of mobile augmented reality applications," p. 8, 2012.

[18] M. Kurze and A. Roselius, "Smart glasses: An open environment for AR apps," in *2010 IEEE International Symposium on Mixed and Augmented Reality*, pp. 313–313, 2010.

[19] S. S. Rautaray, "A real time hand tracking system for interactive applications."

[20] S. Melax, L. Keselman, and S. Orsten, "Dynamics based 3d skeletal hand tracking," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '13, p. 184, Association for Computing Machinery, 2013.

[21] J.-N. Voigt-Antons, T. Kojic, D. Ali, and S. Möller, "Influence of hand tracking as a way of interaction in virtual reality on user experience," in *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–4, 2020. ISSN: 2472-7814.

[22] Z. Lu, "Real-time hand tracking virtual reality interface," *2020 Summer Research Poster Session*, 2020.

[23] D. M. Andrew Clark, "A system for a hand gesture-manipulated virtual reality environment," *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, 2016.