UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Desarrollo de una aplicación interactiva con Intel RealSense

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Sergio Molero Fernández-Calvillo

Tutor: Agustí Melchor, Manuel

Curso 2017/2018

Agradecimientos

Quiero agradecer el apoyo recibido a Débora durante todo el proyecto y la paciencia que ha tenido mientras estaba enfrascado en la programación. Sin su inestimable ayuda hubiera muerto de hambre, gracias por acercarme la comida al ordenador.

También, agradecer a Borja su crítica y opinión objetiva durante el desarrollo y diseño del proyecto.

Y por último y no menos importante, agradecer a mi tutor Manolo por su ayuda e implicación en el proyecto.

Resumen

Quién no ha deseado alguna vez pasar las diapositivas de una presentación con el simple hecho de mover la mano. En este TFG se ha desarrollado una aplicación que te permitirá hacer tus deseos realidad.

Una de las tendencias tecnológicas que más repercusión está teniendo en la sociedad es el procesado de imágenes tridimensionales, que se refiere a la disponibilidad de la información de color junto con la de distancia de cada punto a la cámara. A lo largo de mi vida laboral he adquirido conocimientos sobre distintos sistemas de procesado de imágenes 2D y datos para su posterior integración en entornos 3D. Uno de los grandes retos de trabajar en entornos virtuales en 3D es la adquisición, procesado y generación de información en tiempo real.

Esta información se puede obtener bien a través de técnicas de la visión estéreo o bien por la disponibilidad de obtener la información de coordenadas espaciales mediante el propio sensor, la cámara. Este es el caso de las cámaras *Intel RealSense*. Este trabajo trata sobre su desarrollo y sus aplicaciones a nivel tanto comercial como de uso doméstico.

Para ello se hará uso de la cámara *Intel RealSense SR300* y el SDK Intel RealSense. Hablaremos de las dependencias de estas tecnologías y de los modos de funcionamiento que ofrece para conseguir la interacción con el usuario. Después se realizarán una serie de experimentaciones para determinar su rango de uso y finalmente, se realizará una aplicación que integrará esas funcionalidades. La aplicación reconocerá gestos en tiempo real y los asociará a acciones en otras aplicaciones. A lo largo de este documento descubrirá los pasos que se han realizado para su desarrollo.

Palabras clave: cámara, profundidad, Intel, RealSense, SR300, gestos.

Resum

Qui no ha desitjat alguna vegada passar les diapositives d'una presentació amb el simple fet de moure la mà. En aquest TFG s'ha desenvolupat una aplicació que et permetrà fer els teus desitjos realitat.

Unes de les tendències tecnològiques que més repercussió està tenint en la societat és el processat d'imatges tridimensionals, que es refereix a la disponibilitat de la informació de color juntament amb la de distància de cada punt a la càmera. Al llarg de la meua vida laboral he adquirit coneixements sobre diferents sistemes de processament d'imatges 2D i dades per a la posterior integració en entorns 3D. Un dels grans reptes de treballar en entorns virtuals en 3D és l'adquisició, processament i generació d'informació en temps real.

Aquesta informació es pot obtenir bé a través de tècniques de la visió estèreo o bé per la disponibilitat d'obtenir la informació de coordenades espacials mitjançant el propi sensor, la càmera. Aquest és el cas de les càmeres Intel RealSense. Aquest treball tracta sobre el seu desenvolupament i les seues aplicacions a nivell tant comercial com d'ús domèstic.

Per això es farà ús de la càmera *Intel RealSense* SR300 i el SDK Intel RealSense. Parlarem de les dependències d'aquestes tecnologies i de les maneres de funcionament que ofereix per aconseguir la interacció amb l'usuari. Després es realitzaran una sèrie d'experimentacions per determinar el seu rang d'ús i finalment, es realitzarà una aplicació que integrarà aquestes funcionalitats. L'aplicació reconeixerà gestos en temps real i els associarà a accions en altres aplicacions. Al llarg d'aquest document descobrirà els passos que s'han realitzat per al seu desenvolupament.

Paraules clau: càmera, profunditat, Intel, RealSense, SR300, gestos.

Abstract

Who has never wished to pass the slides of a presentation just by moving the hand. In this TFG an application has been developed that will allow you to make your wishes come true.

One of the technological trends that is having the most impact on society is the processing of three-dimensional images, which refers to the availability of color information along with the distance of each point to the camera. Throughout my working life I have acquired knowledge of different 2D image processing systems and data for later integration into 3D environments. One of the great challenges of working in 3D virtual environments is the acquisition, processing and generation of information in real time.

This information can be obtained either through stereo viewing techniques or paper by the availability of spatial coordinate information through the sensor itself, the camera. This is the case with Intel RealSense cameras. This deals with its development and applications for both commercial and domestic use.

This will be done using the *Intel RealSense SR300* camera and the Intel RealSense SDK. We will talk about the dependencies of these technologies and how they work to achieve user interaction. Afterwards, a series of experiments will be carried out to determine its range of use and finally, the application will be developed that will integrate these functionalities. The application will recognize gestures in real time and associate them with actions in other applications. Throughout this document you will discover the steps that have been taken for its development.

Keywords : camera, depth, Intel, RealSense, SR300, gestures.

Índice

Índice de figuras

Acrónimos

API	*Application Programming Interface*
AR	Augmented Reality
FOV	Field of View
FPS	Frames Per Second
HMD	*Head Mounted Display*
IDE	*Integrated Development Environment*
PYMES	Pequeña Y Mediana Empresas
RGB	*Red, Green and Blue*
TFG	*Trabajo Final de Grado*
USB	*Universal Serial Bus*
UWP	*Universal Windows Platform*
VGA	*Video Graphics Array*

1. Introducción

1.1 Motivación

En el siglo XXI se ha producido una revolución digital que ha favorecido el acercamiento de la tecnología al gran público. Este acercamiento ha provocado que la sociedad deje de ser reacia a las innovaciones tecnológicas en su vida diaria y con ello se ha conseguido un abaratamiento de la tecnología. Hoy en día cualquier persona puede innovar y experimentar de un modo que hace años era inimaginable.

Una de las tendencias tecnológicas que más repercusión está teniendo en la sociedad es el procesado de imágenes con la información de la distancia de cada punto de la imagen a la cámara. Su uso es muy variado, desde actividades de ocio, aplicaciones industriales, reconocimiento facial, escaneado tridimensional para impresiones 3D a aplicaciones militares. En este TFG quiero trasmitir mis impresiones sobre la cámara *Intel RealSense SR300* sobre su desarrollo y aplicaciones a nivel tanto comercial como de uso doméstico.

A lo largo de mi vida laboral en el mundo del *broadcast* he adquirido conocimientos sobre distintos sistemas de procesado de imágenes 2D y datos para su posterior integración en entornos 3D. Uno de los grandes retos de trabajar en entornos virtuales en 3D es la adquisición, procesado y generación de información en tiempo real.

Cada vez que la tecnología avanza en el campo de la adquisición de datos, en este caso captura de imágenes y vídeo, la industria demanda mayores resoluciones. Hace unos años era compleja la grabación de vídeos de alta calidad HD (1920×1080 píxeles). Hoy en día, con la llegada de las gafas de realidad virtual como Oculus la industria está demandado resoluciones de 8K (8192×4320 píxeles) o superiores para poder generar entornos 3D realistas.

La visión por computador es el conjunto de técnicas de análisis de imagen que se utilizan sobre un computador y abarca infinidad de procesos, desde los más comunes (habitualmente denominado como procesado de imagen y que se refiere a operaciones como el recorte de imagen, sustracción de color o la corrección de color), hasta los más complejos (como la detección de objetos, análisis de la imagen para el cálculo de la posición de la cámara que obtuvo la imagen, etc.) y que conllevan un alto coste computacional.

La cámara *Intel RealSense* SR300 nos aporta la capacidad de obtención de imagen *RGB* de alta resolución junto con una matriz de puntos que generan un mapa 3D asociado a los puntos de la imagen RGB. Estas capacidades sumadas a la integración hardware con el procesador Intel del ordenador le dotan de la capacidad de trabajar en tiempo real. Esta simbiosis con el procesador es una gran ayuda para el reconocimiento de personas, manos, gestos, expresiones faciales en tiempo real, etc. En contrapartida, te obliga a usar procesadores Intel como mínimo de 6ª generación ya que la cámara usa instrucciones específicas del procesador y te limita la variedad de hardware con que esta cámara puede funcionar.

El mundo del reconocimiento de imágenes y su integración con mapas 3D solo está empezando. En los próximos años se esperan grandes avances con dispositivos más potentes, ligeros y asequibles listos para que le saquemos todo su potencial para nuestro día a día como interfaz persona computador.

1.2 Objetivos

El principal objetivo de este trabajo final de grado es el diseño y desarrollo de una aplicación interactiva con *Intel RealSense SR300*.

La aplicación será un *middleware* entre la cámara *Intel RealSense SR300* y una aplicación de terceros (ver figura 1.1). El *middleware* se encargará de procesar la información de la cámara para detectar gestos de manos o rostros y enviará mensajes simulando las pulsaciones de teclas a la aplicación de terceros elegida, por ejemplo a *Impress* de *Libre Office*. La aplicación de terceros controlada no será modificada para aceptar estos mensajes.

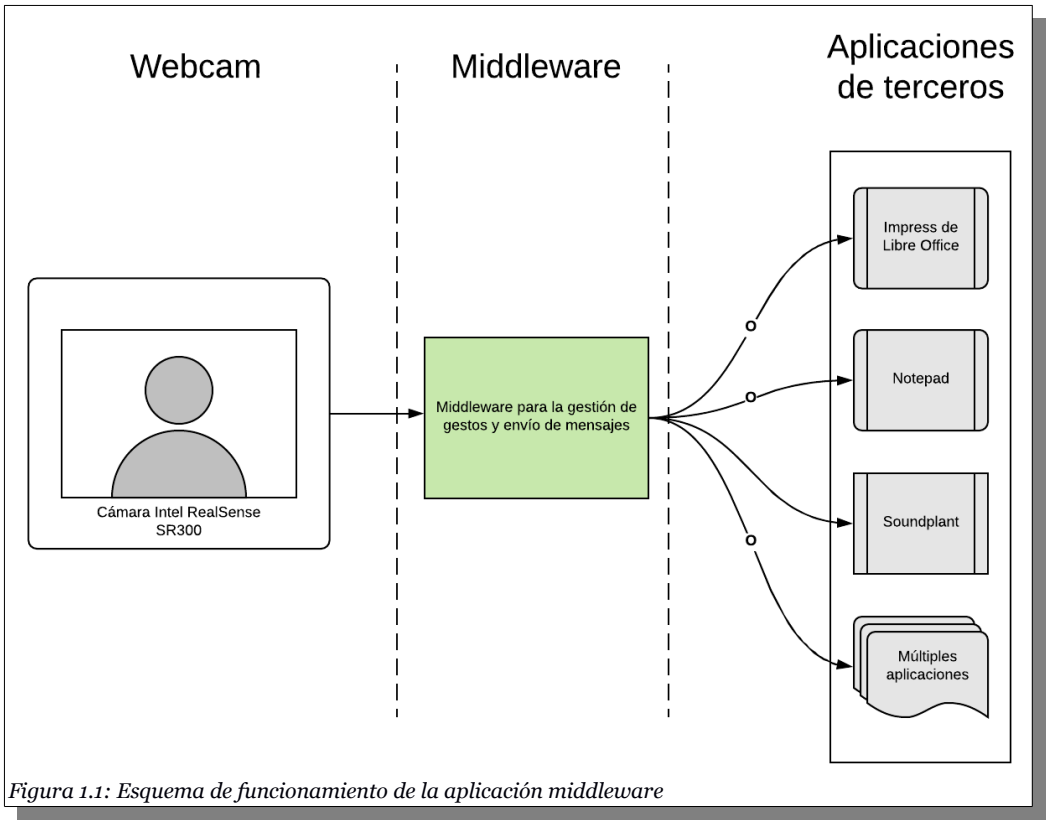

Figura 1.1: Esquema de funcionamiento de la aplicación middleware

La cámara *Intel RealSense* divide los modos de trabajo en módulos. Nuestra aplicación integrará los distintos módulos para que la cámara pueda trabajar según la necesidad, ya sea detectando gestos de las manos o rostros, haciendo uso de los sensores de imagen y profundidad para dichos propósitos.

Para poder llevar a cabo este objetivo será necesario alcanzar los siguientes subobjetivos:

1. Establecer requisitos mínimos de *hardware* y *software* para el desarrollo de la aplicación.
2. Integración del módulo *Cursor Mode* en la aplicación *middleware* para la detección de manos en modo cursor (ver apartado 3.3.3).
3. Integración del módulo *Hand Tracking e*n la aplicación *middleware* para el seguimiento de manos y poder tener acceso al esqueleto de las manos detectadas (ver apartado 3.3.2).
4. Integración del módulo *Face Tracking and Recognition e*n la aplicación *middleware* para tener acceso reconocimiento de rostros (ver apartado 3.3.4).
5. Implementación de método de comunicación mediante el paso de mensajes a través de la API de Microsoft con aplicaciones de terceros (ver apartado 3.3.5).
6. Implementación de sistema de mensajes a modo de realimentación (*feedback)* para el usuario. Este sistema no tiene que ser bloqueante para que se procese en paralelo mientras se accede a la cámara (ver apartado 3.3.6).

1.3 Estado del Arte

Desde finales de los 90 se usan distintos métodos para la interacción con el ordenador como el uso de teclados, ratones o comandos de voz; su *feedback* por excelencia ha sido la pantalla y los altavoces. Pero el uso del teclado y ratón es poco intuitivo e ineficiente, por lo que es necesario tener un entrenamiento para poder usarlo prescindiendo del sentido de la visión. Por otro lado, el uso de órdenes de voz es socialmente incómodo, lento y provoca cansancio al usuario. Esto presenta un nuevo reto para poder encontrar nuevas formas de comunicarse con el computador.

El camino de la búsqueda de un nuevo sistema de interacción comenzó con uno de los elementos más innovadores de uso doméstico como son las consolas. La consola Wii de Nintendo salió al mercado el 19 de noviembre de 2006 [18]. Esta añadió un emisor y unos receptores de infrarrojos a los típicos mandos de consola. La suma del sistema de infrarrojos y los acelerómetros incluidos en los mandos eran capaces de calcular la distancia relativa del usuario hasta la barra emisora de infrarrojos y detectar patrones de movimientos realizados con las manos del usuario. Este producto presentó un revolucionario interfaz de comunicación con un ordenador aunque adolece de la necesidad de usar dos mandos para que el sistema pueda interactuar con el usuario y detectar el movimiento de las dos manos de forma independiente.

El siguiente paso en la evolución en las interfaces persona computador lo dio Microsoft con su accesorio Kinect para XBOX (ver figura 1.2) que fue lanzado en Norteamérica el 4 de noviembre de 2010 [19]. Microsoft lo dotó de micrófonos, una cámara VGA (640x480 píxeles) y un proyector de infrarrojo con una cámara infrarroja de (640 x840 píxeles). Con este hardware era capaz de capturar un mapa 3D (que se suele denominar nube de puntos) y generar un esqueleto virtual del usuario 30 veces por segundo sin necesidad de mandos o marcadores. Por contra, su tamaño era lo suficientemente grande para que la integración con un HDM fuese aparatosa.

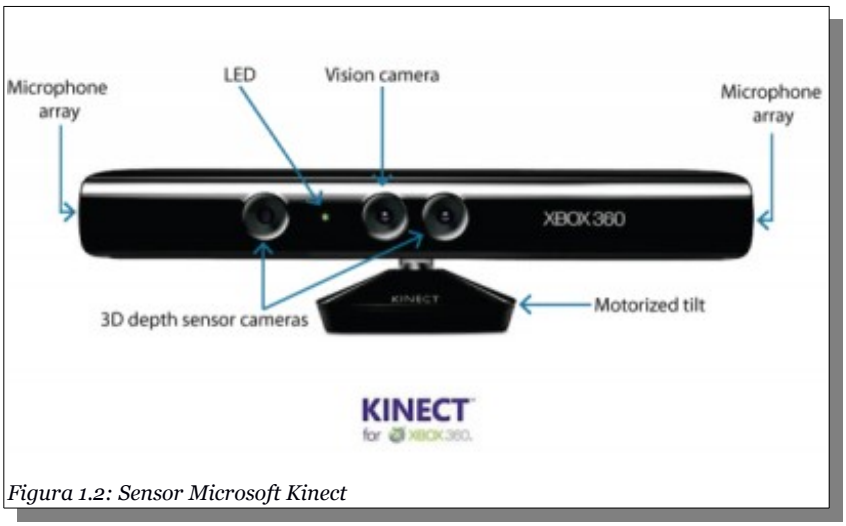

Figura 1.2: Sensor Microsoft Kinect

En el 2018 tenemos sistemas de reconocimiento de cuerpo completo de uso cotidiano como Kinect, pero también hay una necesidad de detección de zonas más focalizadas y con mayor precisión como por ejemplo las manos. Estos sistemas de reconocimiento de gestos son ideales para capturar movimientos de las manos con mayor precisión y con

menor margen de error, y por este motivo pueden ser usados en cascos de realidad aumentada o simuladores. Además, al tener un nivel de detalle mayor hace que pueda reconocer múltiples gestos. Un referente para la detección de gestos y seguimiento de manos, es *Leap Motion* (ver figura 1.3) que salió al mercado en 2013 [23] y ofrece una buena precisión en el seguimiento de manos.

Figura 1.3: Leap Motion sistema de seguimiento de manos

La cámara *Intel RealSense SR300* (ver figura 1.4) [20] es una de las opciones que hay actualmente en el mercado para la adquisición de imagen 2D y de un mapa 3D. El tamaño de la cámara se ha visto reducido notablemente y hace más sencilla la integración en dispositivos HDM o portátiles. La cámara es capaz de detectar 14 poses detectando el esqueleto de la mano y 5 gestos dinámicos detectando la mano. También reconoce caras, expresiones faciales y es capaz de hacer segmentación de fondo lo que la convierte en un dispositivo muy interesante.

Figura 1.4: Cámara Intel RealSense SR300

Hay otros sensores 3D en el mercado que le hacen competencia a la cámara *Intel RealSense SR300* como es el caso de *ASUS XtionPro LIVE* o *Structure Sensor* que pugnan en una carrera por la reducción de tamaño de los dispositivos y su portabilidad (ver figuras 1.5 y 1.6).

Figura 1.5: Cámara XtionPro

Figura 1.6: Cámara Structure Sensor

Analizando las especificaciones de las tres cámaras se desmarca ligeramente la *RealSense SR300* con un FOV más amplio, mayor resolución del sensor RGB y una mayor tasa de frecuencia del sensor de profundidad que llega hasta 60 fps donde las otras se quedan en 30 fps.

	ASUS XtionPro Live [35]	**Structure Sensor [36]**	**Intel RealSense SR300 [34]**
Alcance del sensor 3d	0.8 a 3.5 m	0.4 a 3.5 m	0.2 a 1.5m
3D Resolución	640x480 30 fps	640x480 30 fps	640x480 60 fps
RGB Resolución	1280x1024 30fps	640x480 30 fps	1920x1080 30 fps
FOV	58° H, 45° V, 70° D	58°H, 45° V	73°H, 59° V, 90° D
Tamaño	180x40x25 mm	119.2x28x29 mm	110x12.5x 3.75 mm
Conexión	USB 2.0	USB 2.0	USB 3.0

No todos los casos de uso de control de gestos de visión por ordenador requieren una pantalla montada en la cabeza, pero los HMDs son, sin duda, el foco actual de la industria. Un ejemplo de los caso prácticos de uso que podemos encontrar en el mercado es Edison, un software de Brainstorm Multimedia para educación [1] en el que se aprovechan las funcionalidades de segmentación de fondo y detección de gestos (ver figura 1.7)

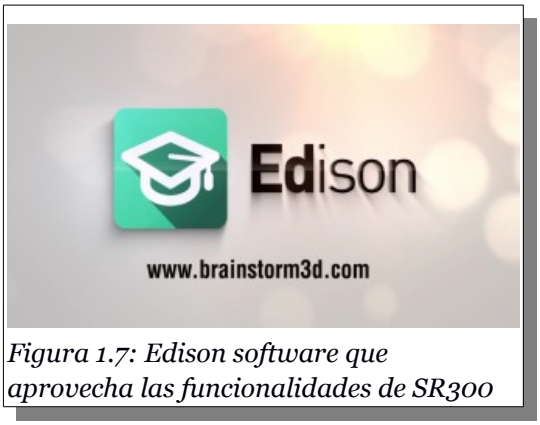

Figura 1.7: Edison software que aprovecha las funcionalidades de SR300

Hasta el momento hemos revisado el hardware como pieza importante en la adquisición de imágenes y nubes de puntos 3D, pero otra parte no menos importante es el trabajo que realiza el software para la interpretación de los datos. Cada fabricante de dispositivos de reconocimiento de imagen y gestos proveen de su propio SDK (kit de desarrollo de software) para la integración en soluciones personalizadas aunque existen otros SDK genéricos que son independientes del dispositivo.

Algunos de los SDK genéricos para el procesado de los distintos sensores 3D que hay disponibles hoy en día son los siguientes:

- **Nuitrack:** Es un SDK que soporta múltiples sensores 3D como Asus Xtion1/2, Kinect v1 o Intel RealSense D415/D435. Este SDK ofrece distintas funcionalidades como captura de nube de puntos, seguimiento de cuerpo completo generando un esqueleto y reconocimiento de gestos. Es multiplataforma con soporte Windows, Android, Linux x64 y Linux ARMv7 y es de carácter comercial[30].
- **Curvsurf:** Es un SDK destinado a la detección de superficies y objetos en tiempo real para AR, a la visión por computador y a los sistemas de anticolisiones que acepta como entrada diferentes tipos de sensores 3D como HoloLens o cámaras RealSense [31].
- **Gestoos:** Es un SDK de reconocimiento de gestos y un robusto sistema de seguimiento de manos. Es multiplataforma y soporta múltiples sensores 3D como entrada. Los modelos soportados son: Orbbec Astra Series, Occipital Structure, Asus Xtion y PMD Flexx and Monstar. Es de carácter comercial y es necesario solicitar acceso al SDK [33].

2 Metodología

Para el desarrollo de la aplicación definida en el apartado 1.2 objetivos, dividiremos el trabajo de investigación en 3 fases diferenciadas.

Primera fase:

La primera fase es la de determinar el software y el hardware requerido que está descrito en los apartados 3.1 y 3.2.

Segunda fase:

La segunda fase ha consistido en definir métodos de interacción con el usuario y un sistema de envío de teclas a otras aplicaciones.

Lo primero que se ha realizado en esta segunda fase es un estudio del SDK versión R2 y de la versión R3 para evaluar sus funcionalidades, testear las aplicaciones de ejemplo y las distintas herramientas que vienen disponibles con el SDK. Entre las distintas funcionalidades que ofrece la cámara *Intel RealSense SR300*, hemos decidido implementar en la aplicación los dos modos de reconocimiento y seguimiento de manos al comprobar a través de los diferentes ejemplos que vienen con el SDK, que son lo bastante precisos para poder ser utilizados como método de interacción eficiente entre el usuario y el computador. También se incorpora la funcionalidad de reconocimiento de rostros en la aplicación ya que en los ejemplos funciona de manera bastante eficiente y será una funcionalidad interesante para investigar en este TFG.

Una vez seleccionados los métodos de interacción persona computador, nos centramos en el estudio de envío de mensajes entre aplicaciones en entorno de Windows 10. Se analizará la clase de la API de Microsoft virtual keys [14] y sus distintas interacciones con diferentes programas, y su complejidad de implementación dependiendo de las tecnologías en las que se han desarrollado los programas a los que se quieran enviar los mensajes de teclas.

Tercera fase:

Se hará un estudio para elegir el método de representación gráfica para las interacciones del usuario a través de la cámara. Esta parte es crítica porque tiene que ser lo suficientemente fluida para que la aplicación se pueda procesar y representar la información en tiempo real.

3 Desarrollo

En este capítulo procedemos a exponer el desarrollo que se ha llevado a cabo en la realización del proyecto y las aplicaciones utilizadas. Inicialmente vamos a realizar una descripción del entorno de trabajo y sus instalaciones más importantes.

En el transcurso del desarrollo de la aplicación *middleware* se tomó como inicio el SDK R2 [3]. Cuando se completó la implementación del seguimiento de manos se migró al SDK R3 [4], al ser más reciente y incluir mejoras sobre el R2. Hay que indicar que la migración del SDK R2 al SDK R3 no es transparente y por tanto conlleva un coste de reimplementación. En el SDK R3 nos encontramos con una estructura más compacta y limpia. Las funciones existentes en el SDK R3 varían significativamente con respecto al SDK R2. Además, ya no se usan variables globales para controlar las funciones de la cámara. Estas variables globales ahora son propiedades de los objetos que gestiona la cámara. Otro cambio importante que encontramos en esta versión es que los diferentes modos de trabajo de la cámara han sido reorganizados en módulos separados lo que permite generar programas menos pesados al tener la opción de incluir en la aplicación solo los módulos requeridos en ella.

Por todo esto nos centraremos en el proceso de desarrollo sobre el SDK R3, ya que es el último publicado por Intel y está más depurado que el SDK R2.

3.1 Software

En este apartado vamos a enumerar y especificar el software utilizado para el análisis y programación de las funcionalidades de la *Intel RealSense SR300*.

El uso de un IDE de programación es importante para reducir el tiempo de desarrollo y depuración de la aplicación. Hemos decidido usar *Visual 2015 community* porque es el IDE utilizado por Intel para desarrollar los ejemplos y tutoriales y además, cuenta con la ventaja de ser gratuita para uso académico.

También se ha usado un control de versiones para gestionar la evolución del código a lo largo del proceso de programación, algo que debería ser de obligatorio uso para cualquier proyecto de desarrollo. En este caso se ha elegido Mercurial [21] con su interfaz gráfica TortoiseHG [22] por ser un entorno del que ya poseo amplios conocimientos.

Por último, señalar que en el inicio del proyecto se usó el SDK RealSense SDK 2016 R2 y posteriormente se migró el código al RealSense SDK 2016 R3. Con este cambio mejoramos el rendimiento y reducimos el tamaño de la aplicación.

3.1.1 Visual 2015 *community*

Es un entorno de desarrollo integrado para sistemas operativos Windows. Soporta múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django. Esta edición de 2015 tiene uno de sus puntos fuertes en la posibilidad de desarrollo multiplataforma (Android, IOS y Windows).

Visual Studio 2015 tiene tres versiones disponibles: *community*, profesional y *enterprise:*

- Visual Studio Community. Esta edición está dirigida a proyectos pequeños de *código abierto*. Esta versión es completamente gratuita para cualquier persona que quiera sumergirse en el mundo de la programación.
- Visual Studio Profesional. Con esta edición se permite desarrollar cualquier proyecto integrando herramientas de colaboración.
- Visual Studio Enterprise. Esta es la edición superior de Visual Studio y tiene herramientas colaborativas para el trabajo en grupo.

Términos de la licencia. ¿Quién puede utilizar Visual Studio?

- Si eres un desarrollador tanto para uso personal como para uso comercial, sin ninguna restricción.
- PYMES para uso comercial siempre y cuando sean 5 o menos usuarios desarrollando.
- Cualquier organización, empresa o persona que desarrolle un software de código abierto.
- Los docentes y formadores sin restricciones.
- Las personas que lo utilicen para la investigación académica.

La versión Community se equipara a la versión Profesional con la única diferencia de la licencia. Si no cumples los pasos anteriormente citados tienes que comprar una licencia Profesional con derecho a soporte MSDN.

3.1.2 TortoiseHG

TortoiseHG es una extensión del *shell* de Windows, con una serie de aplicaciones para el sistema de control distribuido de versiones Mercurial. Mercurial es un sistema de control de versiones distribuido libre, gratuito y viene incluido con la instalación de TortoiseHG (ver figura 3.1).

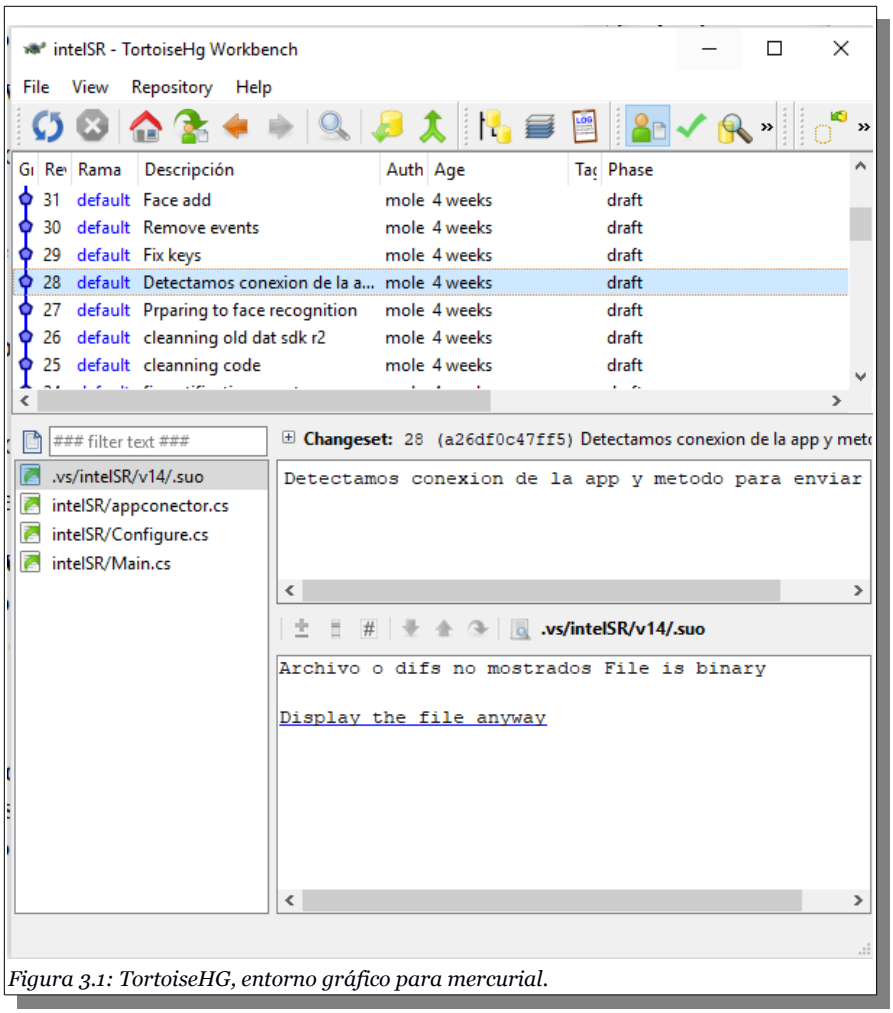

Figura 3.1: TortoiseHG, entorno gráfico para mercurial.

3.1.3 SDK Intel RealSense SDK v.11.0.27.1384 (2016 R3)

El SDK 2016 R3 está dividido en seis módulos, un módulo principal o core y 5 módulos que permiten acceder a las principales funcionalidades de la cámara. Esta estructura permite integrar solo las funcionalidades que se requieran y no estás obligado a integrar todas las funcionalidades en la aplicación. Esto supone hacer aplicaciones más ligeras.

Descripción de módulos:

- *Intel rs sdk mw core*: El núcleo ofrece capacidad de retransmisión básica, interfaces y herramientas como edición de clips y exploración de ejemplos.
- *Intel rs sdk mw cursor*: Para aplicaciones de interfaz de usuario que requieren una delicada estabilidad y precisión de mano, con gestos específicos que se pueden utilizar para varios propósitos de la interfaz.
- *Intel rs sdk mw face*: Identifica la presencia de un rostro en los límites del rango de la cámara o características faciales en un rostro individual. Admite 78 puntos de referencia para garantizar una mayor precisión y una auténtica detección de rostros tridimensional, así como la rotación, el ángulo de movimiento hacia arriba y hacia abajo, y el ángulo de movimiento de derecha a izquierda del rostro.
- *Intel rs sdk mw hand*: Para juegos y aplicaciones que requieren múltiples gestos de manos, sin necesidad de un seguimiento de manos preciso.
- *Intel rs sdk mw scan3d*: Reconstruye la forma y el aspecto de objetos estáticos a partir de una secuencia de imágenes tomadas desde diversos ángulos. Utilizando técnicas de segmentación y seguimiento, la secuencia de imágenes se convierte en una malla de triángulos 3D (por ej. PLY), a los fines de realizar una simulación, edición e impresión o análisis.
- *Intel rs sdk mw seg3d*: Genera una imagen segmentada por cuadro que se puede utilizar para eliminar o reemplazar partes de la imagen detrás de la cabeza y los hombros del usuario.

La capa de envoltura del SDK (ver figura 3.2) expone la interfaz del SDK a una variedad de lenguajes. Esto le da al usuario la flexibilidad de usar cualquier idioma de su elección.

El núcleo del SDK es responsable de organizar el *pipeline* de ejecución. También es la base de los componentes que gestionan los dos tipos de módulos que proporcionan funcionalidades SDK a su aplicación.

- Módulos de E/S: Captura los datos de entrada de su dispositivo y los envía a un dispositivo de salida o a los módulos de capacidad.
- Módulos de capacidad: Incluye varios algoritmos de RealSense, como los algoritmos de detección y reconocimiento de patrones, el reconocimiento y seguimiento de caras, el seguimiento de dedos, el reconocimiento de gestos y el reconocimiento y síntesis de voz.

Es posible tener múltiples módulos de capacidad contenidos dentro de la *pipeline* al mismo tiempo y utilizar más de una cámara u otro dispositivo de entrada en la aplicación.

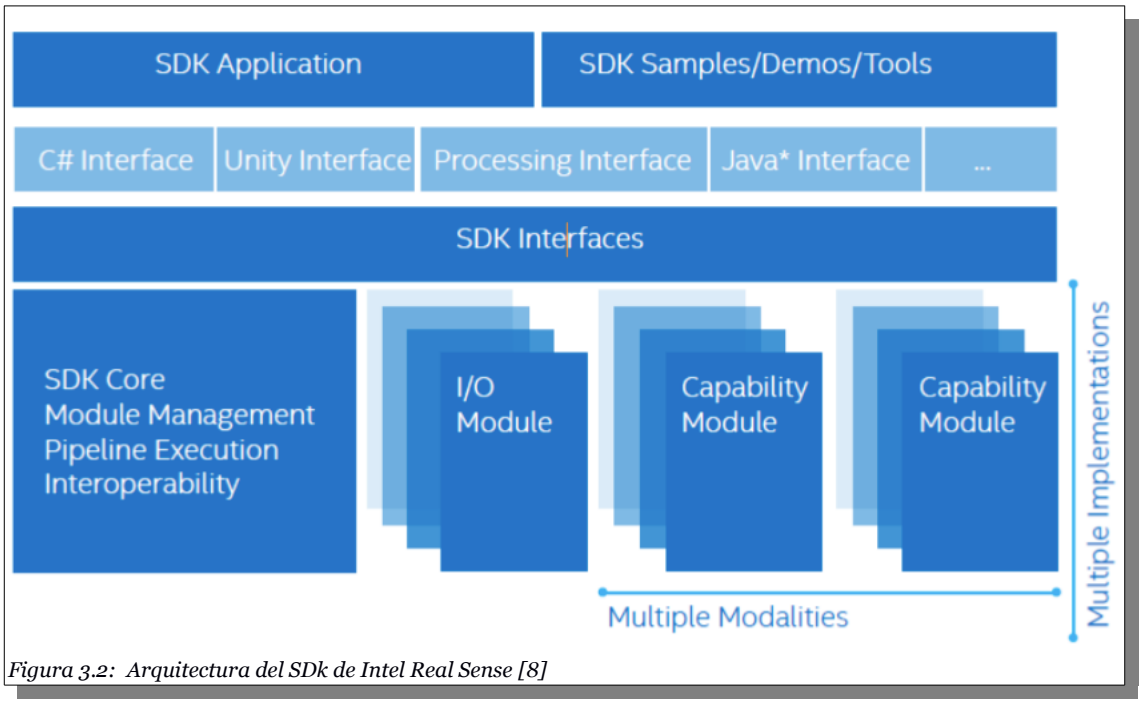

Figura 3.2: Arquitectura del SDk de Intel Real Sense [8]

Requerimientos *hardware*:

- 6th Generación Intel CoreTM Processor (Haswell), o posterior. Core i5/i7 recomendado.
- 8 GB libres de almacenamiento.
- *Intel RealSense Camera* soportadas:
  - F200 PRQ Camera
  - SR300 Camera
- Un puerto USB3 para soportar el ancho de banda de la *Intel RealSense*.

Requerimientos *software*:

- Sistemas operativos soportados para Intel SR300.
  - Microsoft* Windows* 10
- Microsoft Visual Studio* 2012-2015 con el último service pack.
- Microsoft Visual Studio* 2015 con "Universal Windows App development tools"->"Tools and Windows 10 SDK (10.0.10586)" para el desarrollo UWP.
- Microsoft .NET* 4.0 Framework para C#.
- Unity 5.2.3.p3 o posterior.

- Intel IrisTM and HD Graphics Driver for Windows* 10/8.1 64-bit.

3.1.4 Intel controlador DCM v.3.3.27.5718

En cuanto el controlador se ha usado la versión de controlador 3.3.27.5718 [37] [7]de la cámara *Intel RealSense SR300* compatible tanto para SDK R3 y el SDK R2 de Intel. Estos controladores son para cámaras de corto y largo alcance, están diseñados para exponer las interfaces a la transmisión de vídeo desde las cámaras Intel RealSense tanto en color como en profundidad.

3.2 Hardware

La plataforma elegida para el desarrollo de la aplicación ha sido un portátil DELL OMEN de alta gama que lleva la cámara *Intel RealSense SR300* integrada. Una ventaja de tener la cámara integrada es que evitas fallos de conexión del puerto USB y hace el sistema más portable.

Dell Omen características [2]:

- Procesador Intel Core i7-6700HQ @ 2.60 GHz.
- Pantalla WLED 4K eDP de 43,9 cm (17,3 pulg.) con tecnología NVIDIA G-SYNC.
- Memoria 16 GB DDR4-2400 SDRAM (2 x 8 GB).
- Disco duro SATA de 1 TB 7200 rpm + SSD de 256 GB PCIe NVMe M.2.
- Gráficos NVIDIA GeForce GTX 1060 (8 GB GDDR5 dedicado).

3.2.1 Cámara *Intel RealSense SR300*

Para poder usar todas las funcionalidades de la cámara *Intel RealSense SR300* se requiere de un Procesador Intel core de 6ª generación. Sí el ordenador no dispone de un procesador compatible solo podrá usar el sensor RGB, y sí se quiere usar otra funcionalidad como el sensor de profundidad provocará un error en la aplicación.

Intel. Intel RealSense Requisitos hardware. [5]

Procesadores	6ª generación procesador Intel Core
Sistemas operativos	Windows 10 de 64 bits
Almacenamiento de información	8 GB
USB	3.0
Memoria	4 GB

La cámara *Intel RealSense SR300* está dotada de múltiples sensores (ver Figura 3.3) para captar su entorno y es la segunda generación de cámaras de rango corto de Intel. El proyector de infrarrojos lanza un haz de rayos que es detectado por la cámara de infrarrojos para obtener una nube de puntos que aportan la información espacial que se combinará con la imagen captada por el sensor RGB dando forma a una imagen final realista en 3D.

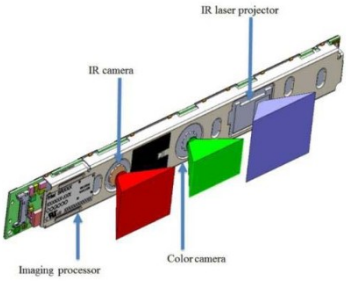

Figura 3.3: Situación de los componentes en la cámara.[17]

El sensor de infrarrojos dispone de una resolución VGA (640×480 píxeles) que reduce el tiempo de exposición y permite movimientos dinámicos de hasta 2 m/s, lo que permite obtener una imagen más limpia de ruido.

La cámara *Intel RealsSense SR300* es capaz de generar una serie de vídeos sincronizados de color, profundidad e infrarrojos. La cámara trabaja de manera óptima entre un rango de 0,2 metros a 1,5 metros en interiores. El rango para usar la función de escaneo 3D se sugiere que se mantenga dentro de los 70 cm, ya que en distancias mayores la precisión de la cámara disminuye drásticamente (ver figura 3.4).

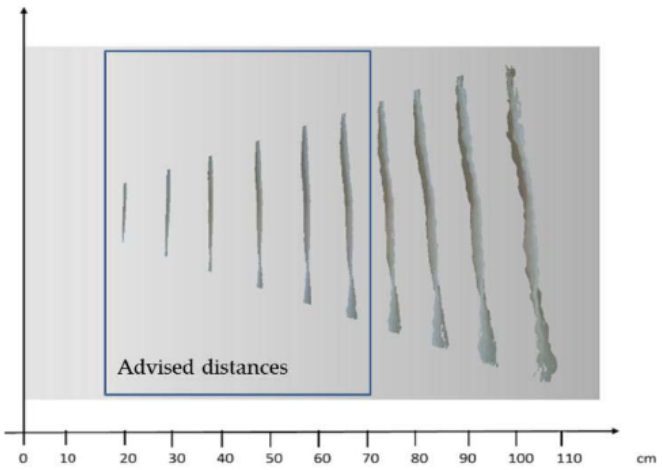

Figura 3.4: Capturas con el sensor 3D de objeto plano a distintas distancias.[17]

3.3 Entorno de trabajo

Para llevar a cabo el proyecto ha sido necesario realizar la configuración del entorno de trabajo en el cual hemos procedido a realizar la instalación de diversas herramientas y controladores.

En primer lugar, se ha preparado un ordenador con las siguientes características:

- Procesador Intel Core i7-6700HQ.
- Memoria 16 GB DDR4.
- Gráficos NVIDIA GeForce GTX 1060.
- Cámara *Intel RealSense SR300* integrada.

3.3.1 Instalación controladores y SDK

En segundo lugar, se ha procedido a la instalación del controlador Intel DCM v.3.3.27.5718 [7]. Este controlador está diseñado para exponer las interfaces de la transmisión de vídeo de las cámaras Intel RealSense, tanto de color como de profundidad. La versión del controlador v.3.3.27.5718 tiene una serie de errores conocidos y documentados en las notas de versión que Intel publica junto con el driver [37] (ver Anexo 7.3).

Una vez instalado el controlador instalaremos los módulos del SDK R3 que se pueden obtener de la web de Intel [20]. Se instalarán uno a uno ejecutando los instaladores de los módulos necesarios para el seguimiento de manos y reconocimiento de caras.

Los módulos necesarios para desarrollar la aplicación *middleware* son:

- Intel rs sdk mw core: Es el núcleo del SDK (imprescindible para el desarrollo).
- Intel rs sdk mw cursor: Módulo cursor de manos.
- Intel rs sdk mw face: Módulo de seguimiento de caras y reconocimiento.
- Intel rs sdk mw hand: Módulo de seguimiento de manos.

En tercer lugar, se ha procedido a la instalación y configuración del IDE, en nuestro caso el *Visual 2015 community*. Vamos a trabajar sobre un proyecto de C# de tipo formulario de windows al que tendremos que instalar un *wrapper* de ficheros JSON llamado *Newtonsoft.Json* versión 11.0.2. Esto lo podremos hacer desde el gestor de NuGet's que viene con Microsoft Visual 2015 (ver Anexo 7.4). Los pasos básicos de configuración del proyecto de C# están especificados en el Anexo 7.1.

3.3.2 Gestión de módulo *handshape*

Con el módulo *handshape (ver figura 3.5)* tenemos acceso a catorce tipos de gestos posibles que la cámara *Intel RealSense* puede detectar.

Gestos disponibles en modo handshape

Gesto	Icono	No activar a la vez con el gesto	Descripción del gesto
click		*two_fingers_pinch_open*	Abra la mano mirando hacia la cámara, mueva el dedo índice rápidamente hacia el centro de la palma de la mano.
fist		*full_pinch*	Todos los dedos doblados en un puño. El puño puede estar en diferentes orientaciones siempre y cuando la palma de la mano esté en la dirección general de la cámara.
full_pinch		*fist*	Todos los dedos extendidos y tocando el pulgar. Los dedos pellizcados pueden estar en cualquier lugar entre apuntando directamente a la pantalla o de perfil.
spreadfingers		*NA*	Abre la mano, mirando a la cámara.

swipe_down		*NA*	Mano con la palma hacia la cámara, se mueve hacia abajo e inmediatamente de vuelta a la posición inicial.
swipe_left		*wave*	Mano con la palma hacia la cámara, se mueve a la izquierda e inmediatamente de vuelta a la posición inicial.
swipe_right		*wave*	Mano con la palma hacia la cámara, se mueve a la derecha e inmediatamente de vuelta a la posición inicial.
swipe_up		*NA*	Mano con la palma hacia la cámara, se mueve hacia arriba e inmediatamente de vuelta a la posición inicial.
tap		*NA*	Una mano en una postura natural y relajada se mueve hacia adelante como si presionara un botón.
thumb_down		*NA*	Cerrando la mano con el pulgar hacia abajo.

thumb_up		*NA*	Cerrando la mano con el pulgar hacia arriba.
two_fingers_pinch_open		*click*	Abra la mano con el pulgar y el índice tocándose entre sí.
v_sign		*NA*	Cerrando la mano con el dedo índice y el dedo corazón apuntando hacia arriba.
wave		*swipe_left* *swipe_right*	Cerrando la mano con el dedo índice y el dedo corazón apuntando hacia arriba.

La cámara nos da la información de la posición 3D de las articulaciones y los dedos. A partir de esta información de la cámara somos capaces de representar el esqueleto de ambas manos. Si deseamos detectar gestos o posiciones de la mano distintas de las que nos ofrece el SDK R3, debemos generar nuestro propio algoritmo para detectarlos.

Podemos medir distancias con una precisión de ±0,5 cm de error a unos 70 cm de la cámara (ver apartado 4.1). Este cálculo es una estimación según las pruebas realizadas durante el desarrollo. Sí se desea conocer datos más exactos sobre la precisión de la cámara se puede consultar en la bibliografía [17].

Figura 3.5: Modo handshape, extraída de la aplicación del TFG

Estas son las condiciones óptimas para que el seguimiento de manos funcione mejor:

- Un ancho mínimo de la palma de la mano de 5,5 cm (esto suele ser adecuado para un niño de 5 años o mayor).
- La mano debe estar al menos a 20 cm de la cámara. La distancia máxima depende de la cámara; consulte la tabla anterior.
- Velocidad de la mano no puede debe ser superior a 2m/s.

La velocidad de fotogramas óptima para el flujo de profundidad es de 60 fps. La precisión de seguimiento de manos puede deteriorarse para frecuencias inferiores a 50 fps [4].

3.3.3 Gestión de módulo cursor mode

Con el módulo *cursor mode* (ver figura 3.6) tenemos acceso a cinco tipos de gestos posibles que la cámara *Intel RealSense* puede detectar.

Gestos disponibles en modo cursor mode

Gesto	Icono	Descripción
CURSOR_CLICK		Cierre y abra la mano en un movimiento suave.
CURSOR_CLOCKWISE_CIRCLE		Mueva la mano en círculo en el sentido de las agujas del reloj.
CURSOR_COUNTER_CLOCKWISE_CIRCLE		Mueva la mano en círculo en sentido contrario a las agujas del reloj.
CURSOR_HAND_CLOSING		Cierre la mano.
CURSOR_HAND_OPENING		Abra la mano, mirando a la cámara.

Es un modo pensado para usarlo como puntero porque te da la posición 3D de la palma de las manos diferenciando la derecha de la izquierda. La cámara es sensible al movimiento cuando la distancia entre la mano y la cámara es de aproximadamente 20 cm a 115 cm[4].

También se aplican las siguientes limitaciones de rango:

- Detección - La identificación inicial de las manos puede realizarse hasta a 100 cm de la cámara.
- Seguimiento - El seguimiento continúa hasta 115 cm.
- Gestos - Los gestos se identifican hasta 115 cm.

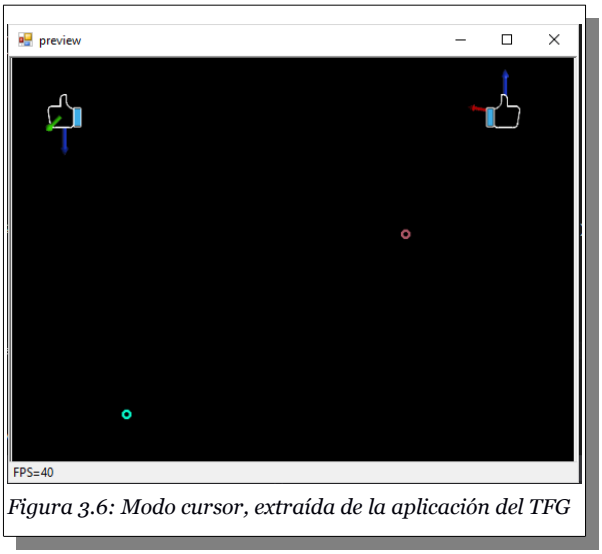

Figura 3.6: Modo cursor, extraída de la aplicación del TFG

3.3.4 Gestión de módulo face tracking

Con el módulo de *face tracking* (ver fig*ura 3.7)* tenemos acceso a las funcionalidades relacionadas con el reconocimiento y seguimiento de rostros. Estas funcionalidades son *Face Location Data, Face Landmark Data, Face Pose Data, Facial Expresion, Face Recognition, Pulse Estimation y Gaze Tracking*.

Con la funcionalidad *Face Location data* podemos acceder a los datos de ubicación de caras (ver figura 3.7). Puede haber múltiples caras en la imagen y cada cara tendrá un identificador único diferente. Los datos de ubicación de la cara incluyen la siguiente información:

- *Bounding box*: El cuadro delimitador es una región rectangular, en los píxeles de color, donde se encuentra la cara detectada. La caja delimitadora puede o no coincidir exactamente con el tamaño de la cara detectada.
- Profundidad media: El valor medio de profundidad indica la distancia a la que la cara detectada está de la cámara.

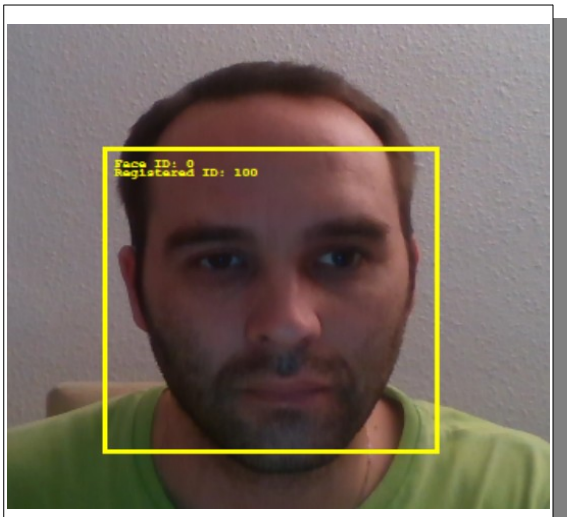

Figura 3.7: Reconocimiento de rostros, extraída de la aplicación del TFG

De forma similar a la detección de ubicación de caras (Face Location), se puede usar la detección de puntos de referencia (*Face Landmark Data)*.En este modo se ponen marcadores de los principales rasgos del rostro (ver figura 3.8). Esto se utiliza usualmente en el cine para la animación de caras en personajes 3D .

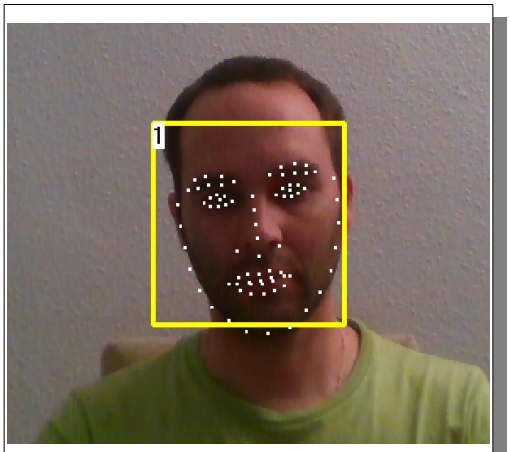

Figura 3.8: Reconocimiento de landmarks, extraída de Face Tracking ejemplo SDK

Usando la funcionalidad de detección de la posición de la cara *(Face Pose Data), se* obtienen los datos de la pose detectada, que incluye la siguiente información:

- Posición de la cabeza.
- Posición Quaternion que indica la posición de la cara en las coordenadas del mundo.
- Los ángulos de Euler indican la posición de la cara en términos de rotación a lo largo de los tres ejes (ver figura 3.9).
- Matriz de rotación que representa la posición de la cara.

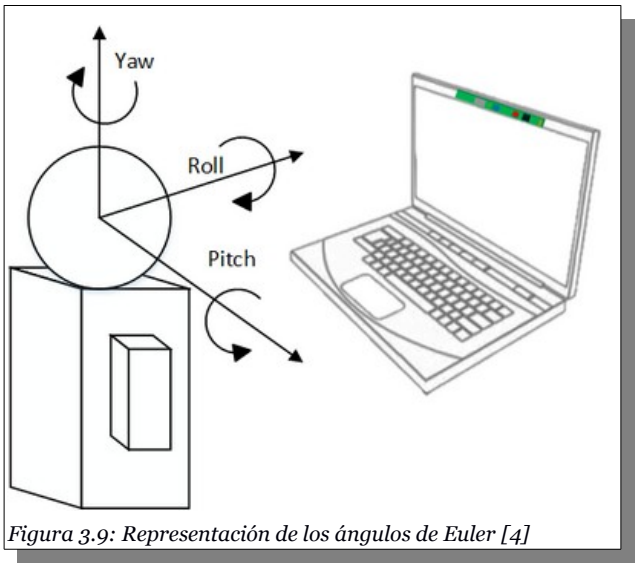

Figura 3.9: Representación de los ángulos de Euler [4]

La cámara nos ofrece otras opciones de reconocimiento como la detección de expresiones (*Facial Expresion)* en el rostro del usuario mostrando la probabilidad en forma de porcentaje sí el usuario está haciendo una expresión facial (ver figura 3.10). El algoritmo soporta la detección de los siguientes grupos de expresiones [38] :

- Movimiento de la cabeza (*Deprecated*): Por ejemplo, la cabeza está girando a la izquierda o inclinada a la derecha.
- Movimiento de cejas: Por ejemplo, la ceja izquierda/derecha se levanta o se baja.
- Movimiento de los ojos: Por ejemplo, los ojos están mirando a la izquierda o a la derecha.
- Movimiento de la boca: Por ejemplo, la boca está abierta, o haciendo una acción de beso.

Cada expresión tiene un parámetro de intensidad asociado que forma parte de la estructura *FaceExpressionResult*, para indicar la probabilidad/escala de la detección.

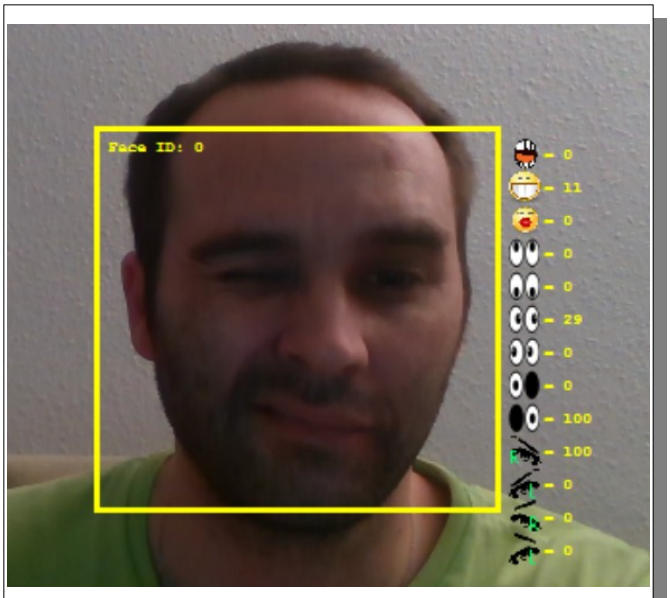

Figura 3.10: Reconocimiento de expresiones, extraída de Face Tracking ejemplo SDK

Al usar funcionalidad de reconocimiento facial (*Face Recognition)* medimos la similitud entre dos caras, una de la imagen actual y la otra de algunas referencias en una base de datos de reconocimiento. La medición se realiza en todas las referencias de la base de datos de caras. Un usuario se considera reconocido si la puntuación de la referencia más coincidente supera cierto umbral, y el resultado del reconocimiento es simplemente el identificador de usuario asociado a la referencia. Por lo tanto, para cualquier reconocimiento facial, primero de debe crear en la aplicación una base de datos con las caras a reconocer. El algoritmo crea un almacenamiento activo predeterminado si no se define ninguno.

El SDK nos provee de ciertas funciones para administrar los registros y la base de datos y son las siguientes:

- Cargar/guardar base de datos: Puede guardar la base de datos en el disco y cargarla más tarde. La interfaz *RecognitionModuleData* proporciona funciones para exportar la base de datos de reconocimiento, y la interfaz *RecognitionConfiguration* para importar una. Guarda el buffer de la base de datos en un archivo y lo carga desde un archivo.
- Registro/Desregistro de usuarios: Registra una cara de usuario en la base de datos utilizando la función *RegisterUser*. El usuario se convierte en una de las referencias de la base de datos. Se puede dar de baja a un rosto a través de la función *UnregisterUser*. Además del registro explícito, la interfaz *RecognitionConfiguration* también introduce un modo de registro continuo en el que el algoritmo de reconocimiento determina automáticamente si un usuario debe estar registrado en la base de datos o no.
- Reconocer la cara actual: Utiliza UserID para realizar el reconocimiento de la cara actual contra la base de datos de reconocimiento activa.

Usando la funcionalidad de detección de pulso (*Pulse Estimation*) podremos obtener el pulso de la persona analizando la imagen de la cara.

Por último, la cámara nos da la opción de hacer un seguimiento de la mirada (*Gaze Tracking)*. La calibración de la mirada es un paso obligatorio para el correcto seguimiento de la mirada. El proceso de calibración consiste en mostrar 6 estímulos, uno a la vez, durante 2 segundos cada uno: centro, centro inferior, centro izquierdo, centro superior, centro derecho y centro de nuevo, que el usuario debe observar (ver Figura 3.11). Al final de la calibración, el algoritmo reporta el estado de la calibración.

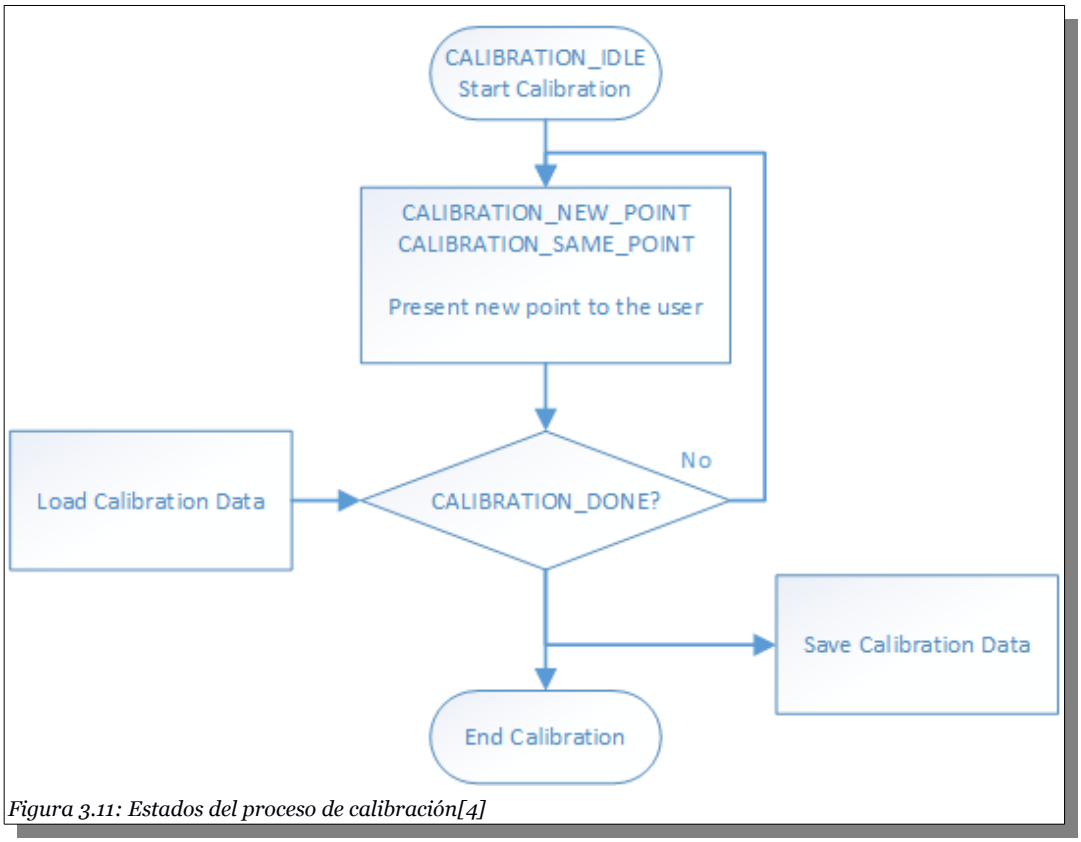

Figura 3.11: Estados del proceso de calibración[4]

En esta aplicación middleware usaremos el reconocimiento de rostros para detectar y identificar al usuario cuando esté presente ante la cámara emulado a un sistema de seguridad de control de acceso. [29].

3.3.5 Gestión envío de teclas virtuales

Para el desarrollo del *middleware* se ha diseñado un sistema que simula la pulsación de teclas en otras aplicaciones mediante mensajes a ventanas usando la API de Windows. Se ha implementado una clase *Appconector* que gestiona el arranque de la aplicación a ejecutar, localizando el identificador del proceso y el manejador de la ventana que va a recibir las pulsaciones (ver figura 3.12). También se encarga de recibir las peticiones de teclas a enviar en forma de mensaje a la aplicación de terceros y debe diferenciar si son teclas especiales como F1, flecha izquierda, etc. o son teclas alfanuméricas con representación gráfica. Está diferenciación es importante ya que el procedimiento para el envío varía en función del tipo de tecla a enviar.

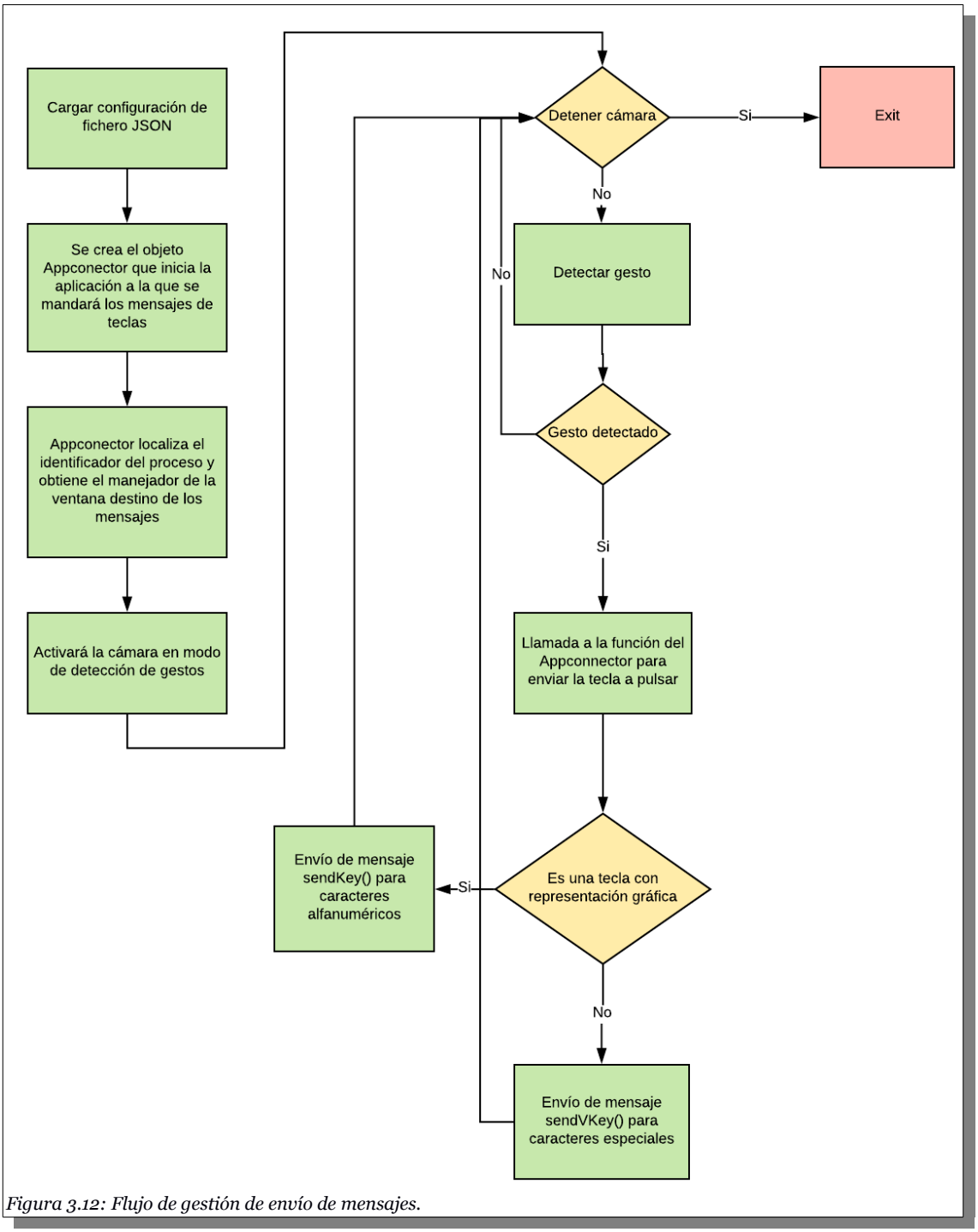

Figura 3.12: Flujo de gestión de envío de mensajes.

Este sistema lee un fichero de configuración JSON (ver Anexo 7.2) donde se especifica la información necesaria para la ejecución de la aplicación. En él se enumeran distintos perfiles que contienen cada uno la ruta de la aplicación que se va a arrancar a la que se le van a enviar los mensajes. De manera opcional, se puede indicar el nombre del proceso que va a recibir los mensajes de pulsación de la teclas que no tiene porqué ser el mismo que el proceso inicial de arranque citado anteriormente. La necesidad de especificar el nombre del proceso depende de la arquitectura de la aplicación que recibirá los mensajes. Averiguar el proceso/componente que tiene que recibir los mensajes no es algo trivial y es recomendable usar un analizador de

interfaces para localizar su nombre y así poder interceptar los mensajes que recibe el proceso/componente. En este proyecto se ha usado como analizador de interfaces la utilidad *Microsoft Spy++* que viene como herramienta de Microsoft Visual 2015. Además, en el fichero también se especifica en qué modo se va utilizar la cámara, en modo *shape* o modo cursor.

Por último, se enumeran todos los gestos que puede detectar la cámara indistintamente del modo de la cámara y a cada gesto se le asigna una tecla. Para indicar la tecla se usa la nomenclatura de la API de Windows para seguir un estándar conocido [15], *por ejemplo* VK_LEFT se utiliza para referirse a la tecla de "flecha del cursor a la izquierda"

3.3.6 Estructuras de hilos y comunicación con el interfaz.

Cualquier aplicación de tipo formulario de Windows corre en un hilo o *thread* principal. Este hilo tiene su propio contexto y en él se gestionan sus objetos y variables. Aparte de este hilo se han creado cuatro hilos independientes que se han diseñado para que su ejecución no bloquee el hilo principal y se pueda procesar otras operaciones en paralelo. Tres de estos hilos están destinados para acceder a la cámara *RealSense,* uno por cada módulo que queremos activar, y un cuarto hilo para el dibujado gráfico encargado de indicarle que se ha detectado una acción al usuario. Durante la ejecución de la aplicación se llegan a usar tres hilos de forma concurrente, el hilo principal, un hilo de acceso a la cámara *RealSense* y el hilo de dibujado gráfico para notificar al usuario los gestos detectados.

Los hilos de gestión de la *RealSense* deben modificar elementos gráficos de la aplicación como la previsualización o los textos para registrar los eventos de los módulos. Desde el hilo principal se modifica el hilo de dibujado gráfico para notificar al usuario la detección de gestos. Estas interacciones entre hilos son complejas ya que se trabaja con contextos distintos (cada hilo tiene su propio contexto). Por ejemplo, pongamos el caso de que el hilo que controla la cámara para poder detectar gestos en modo cursor tiene que modificar un elemento gráfico del hilo principal, como cambiar el color de un botón. Esta operación no puede hacerse directamente ya que producirá una excepción en tiempo de ejecución. Para evitar esta excepción se utiliza el método *delegate* [16], de esta forma podremos modificar elementos de otro hilo.

3.3.7 Problemas encontrados.

Se han detectado distintos problemas tanto de software como de hardware que han provocado retrasos en el desarrollo. No ha sido tarea fácil acotar el error y dar una solución. A continuación explicaremos los problemas más graves que hemos encontrado durante el desarrollo y como hemos podido solucionarlos o paliar sus efectos.

El primer problema que nos encontramos es que el hardware no es muy estable y en ocasiones deja de funcionar sin motivo aparente. Es necesario desconectar la cámara del puerto USB o en el caso que esté integrada en el sistema, apagar el sistema y volverlo a arrancar para que la cámara funcione correctamente.

Otro problema que detectamos con el uso de la cámara es que el sistema deja de reconocer el dispositivo. Esto fue debido a las actualizaciones automáticas del sistema operativo Windows 10, ya que una actualización puede llegar a desconfigurar la cámara

inutilizando el sensor de profundidad. Se puede llegar a solucionar pero el proceso no es válido para todas las configuraciones de equipos. No tenemos una solución oficial al problema por lo que los usuarios de la cámara no están muy satisfechos con el soporte de Intel.

Respecto a estos problemas de desconfiguración citados anteriormente provocados por las actualizaciones de Windows, el único modo de solucionarlo fue seguir una guía publicada por Intel en sus foros [12], pero no a todos los usuarios les ha funcionado correctamente. En mi opinión, pienso que está guía es un buen punto de partida para reinstalar la cámara y que el sistema la vuelva a reconocer.

Por último, en el apartado de software hay que señalar que la evolución entre el SDK R2 y el SDK R3 ha sido una mejora de organización que ayuda a hacer un código más limpio y legible. Por contra, se han modificado muchas funciones del SDK R2 que no son válidas en el SDK R3 con la obligación de reescribir el código sí se quiere migrar. El problema más notable encontrado en el SDK R3 ha sido que hay funciones en el SDK R3 que no están bien implementadas, como la funcionalidad de guardar la base de datos de las caras registradas a un fichero (ver apartado 4.3).

3.4 Diseño de la aplicación para interacción con la cámara Intel RealSense

A continuación procedemos en este apartado a dar una visión general del interfaz, el funcionamiento de la aplicación y de las funcionalidades de la cámara RealSense que han sido integradas en el middleware.

El interfaz se compone por cuatro formularios, un formulario principal llamado RealSense (ver figura 3.13), otro formulario de configuración, Configure, y otros dos para ver la cámara en modo reconocimiento de caras y detección de manos, Facepreview (ver figura 3.13, número 9) y preview (ver figura 3.13, número 10) respectivamente.

En el formulario principal tenemos una barra superior con tres menús. El menú *file* tiene la opción de cerrar la aplicación, el menú *device* muestra un lista de las cámaras detectadas y el menú *Configuration* abre el formulario de configuración (ver figura 3.13, número 1). Debajo de la barra de menú tenemos tres botones *start* (ver figura 3.13, número 2) que inicia la detección de gestos y arranca la aplicación a la que irán dirigidos los mensajes que simularán las pulsaciones de teclas.

El segundo botón, *stop,* detendrá el uso de la cámara en cualquiera de sus modos (ver figura 3.13, número 3), y por último, el botón de p*review* despliega los formularios que muestran una representación de lo que está viendo la cámara (ver figura 3.13, número 4). La zona intermedia del formulario principal (ver figura 3.13, número 5) es una zona donde se registrarán todos los eventos de la cámara.

En la parte inferior tenemos tres botones más, *Face scan* (ver figura 3.13, número 6), que activa la cámara en modo detección de rostros, el botón *face add* que registra una cara para ser detectada (ver figura 3.13, número 7) y se mostrará en su ventana de previo (ver figura 3.13, número 9) con la etiqueta *registered* y el número de identificación, pero si la cara no esta registrada aparecerá la etiqueta *unregistered*. Por último, tenemos el tercer botón *face remove* que elimina la cara registrada (ver figura 3.13, número 8).

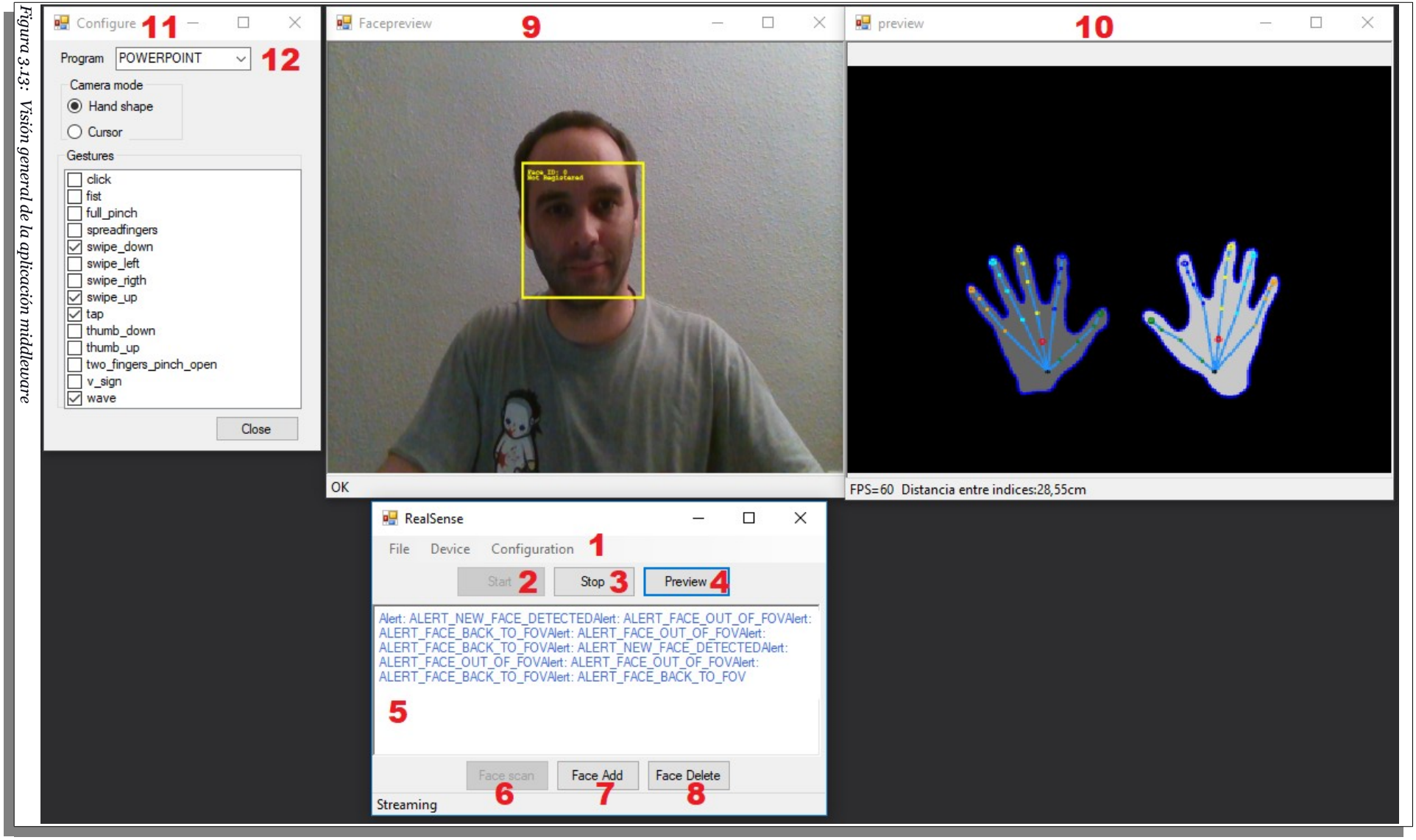

Figura 3.13: Visión general de la aplicación middleware

En el formulario de configuración tenemos un selector donde se seleccionará el programa que será controlado. Se puede elegir el modo que queremos que se use para la detección de gestos y ver un listado de gestos posibles. En el listado aparecerán con un tick los gestos que están habilitados para ser detectados. Dependiendo del modo de la cámara tendremos unos gestos para el modo *hand shape* (ver figura 3.15) y otros para el modo *cursor mode* (ver figura 3.14).

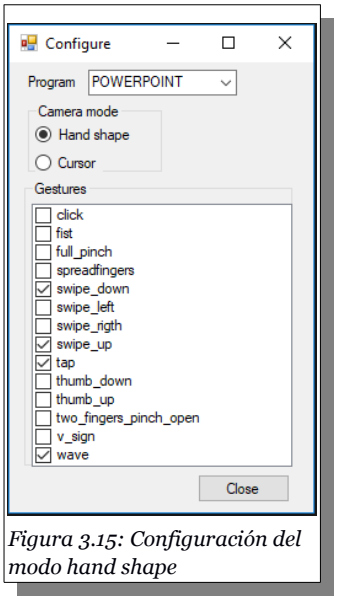

Figura 3.15: Configuración del modo hand shape

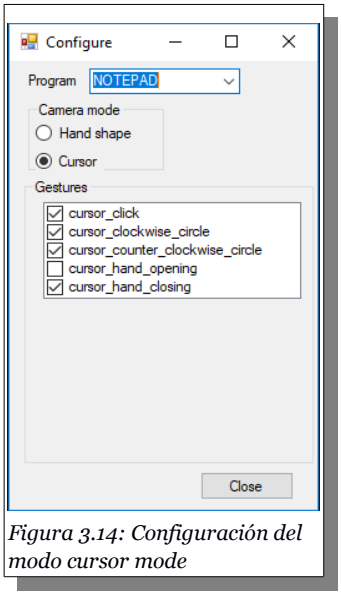

Figura 3.14: Configuración del modo cursor mode

Los formularios de previsualización muestran la información relevante detectada por la cámara *RealSense*. La previsualización en modo *face scan* recuadra hasta cuatro caras en la imagen e indica sí el rostro ha sido registrado mostrando su identificador único y la etiqueta *registered* o sí no está registrado mostrando la etiqueta *unregistered* (ver figura 3.16).

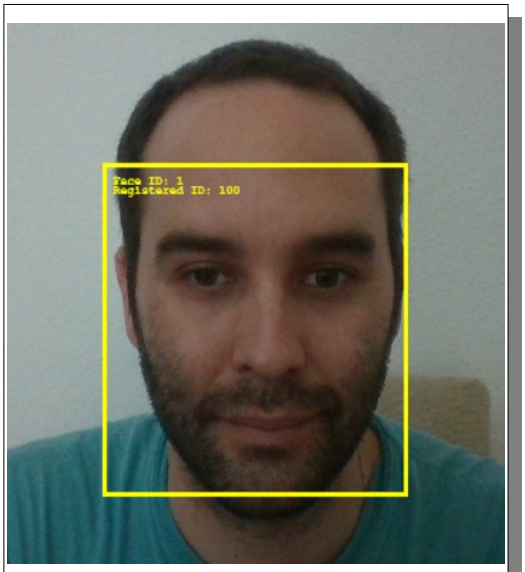

Figura 3.16: Formulario Facepreview mostrando una cara registrada en modo face scan

En modo *hand shape* el formulario *preview* (ver figura 3.17) muestra una representación de la silueta de las manos detectadas, con un máximo de 1 mano derecha y 1 mano izquierda ya que la cámara cataloga la mano con la parte del cuerpo a la que pertenece y no pueden repetirse. La cámara puede detectar los centros de masas de las manos dibujándolos con un punto rojo y además, detecta donde están las articulaciones pudiendo recrear un esqueleto de las manos. En la parte baja del formulario preview podemos ves los cuadros por segundos que se están dibujando. Sí se han detectado dos manos en la parte inferior mostrará la distancia que hay entre los dos dedos índices.

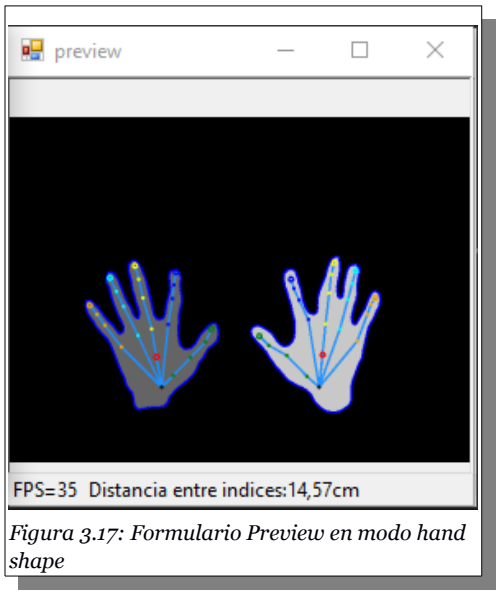

Figura 3.17: Formulario Preview en modo hand shape

En modo cursor mode el formulario *preview* (ver figura 3.18) muestra la posición del centro de masas de las manos con un filtrado para evitar el efecto de temblor. Este centro de masas varía su color según la distancia de la mano varíe con respecto a la cámara. En la parte superior izquierda aparecerá un icono de una mano cuando detecte una mano izquierda y en la parte superior derecha si es la mano derecha. Las flechas que se dibujan en las manos indican que las manos han salido de una zona del espacio 3D, solo tiene el fin de representar el desplazamiento en profundidad en una representación en 2D.

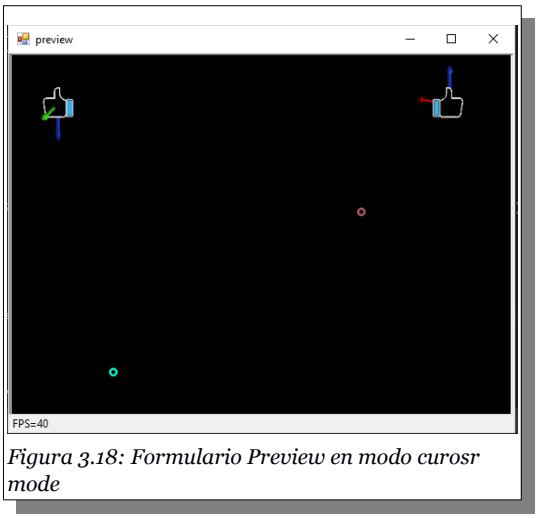

Figura 3.18: Formulario Preview en modo curosr mode

En la aplicación se han integrado tres modos de trabajo de la cámara, reconocimiento de gestos en modo cursor, en modo *hand shape* y reconocimiento de rostros. Se ha definido el siguiente flujo de trabajo para su implementación.

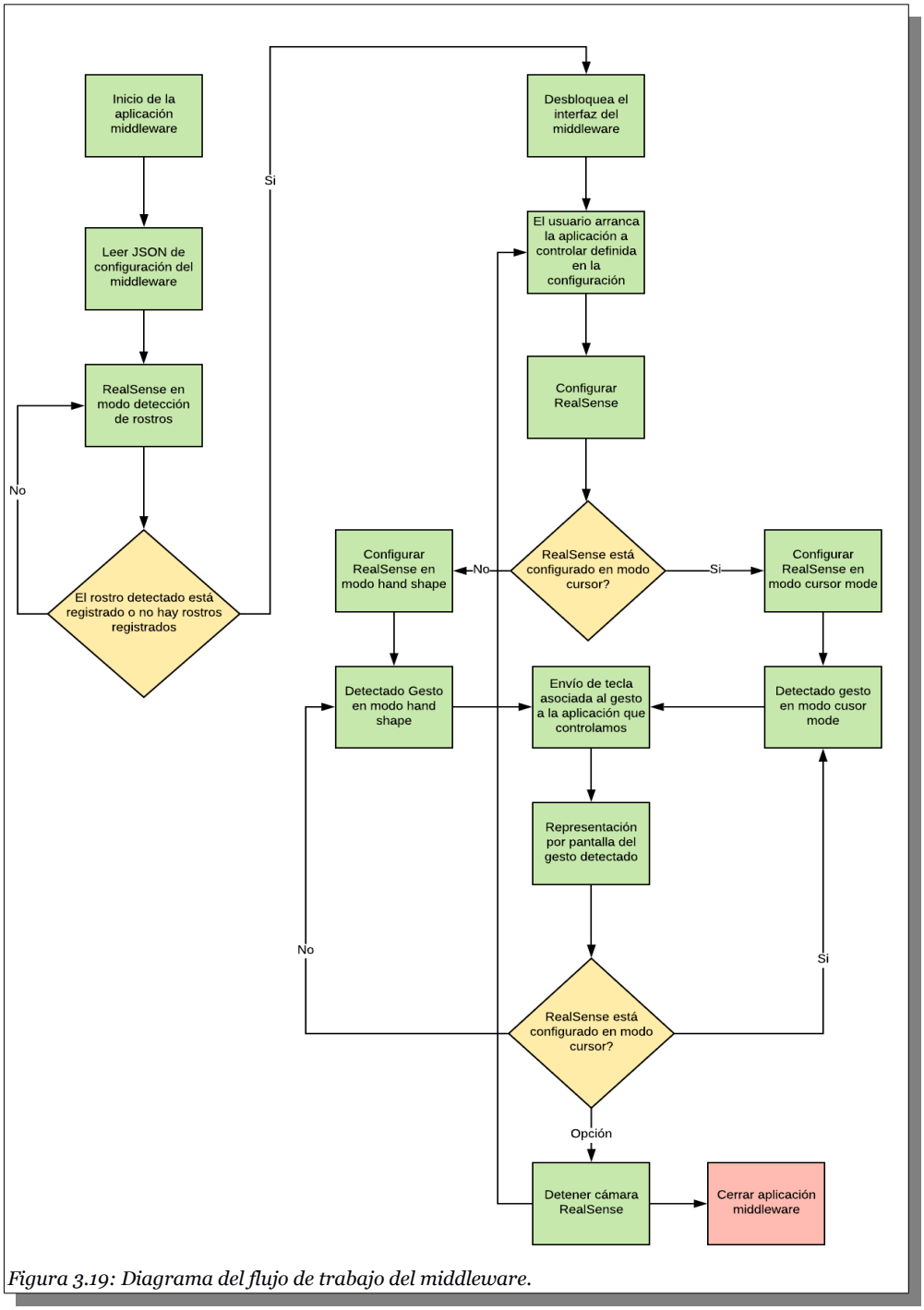

Figura 3.19: Diagrama del flujo de trabajo del middleware.

Tal y como se describe (ver figura 3.19) el flujo de trabajo se inicia arrancando la aplicación. Tras leer el fichero de configuración, el programa carga la configuración por defecto, que en este caso será el primer programa definido en la lista de programas (ver figura 3.13, número 11*)*. La aplicación permanece bloqueada hasta que el usuario autorizado es detectado por la cámara. Este usuario autorizado ha sido registrado y configurado con anterioridad para que la cámara pueda detectarlo, si el programa no

tiene registrado ningún rostro, este arrancará desbloqueado listo para usar. Una vez desbloqueado, el usuario arranca manualmente el proceso para iniciar el reconocimiento de gestos pulsando el botón *start* (ver figura 3.13, número 2*)*. Este proceso arrancará la aplicación seleccionada en configuración del middleware y activara la cámara *RealSense*.

Una vez activada la cámara, el middleware tendrá dos comportamientos diferentes que dependerán del modo en que esté configurado, puede ser en modo *cursor mode* o en modo *shape hand*. Independientemente del modo en el que el middleware haya sido configurado la cámara detectará los gestos definidos para su configuración y que a su vez estén activados en la configuración del middleware. Una vez detectado el gesto se enviará un mensaje con la pulsación de la tecla a la aplicación seleccionada para ser controlada, que estará arrancada y esperando órdenes. Después del envío del mensaje el middleware notificará al usuario el gesto detectado mediante un icono representativo de dicho gesto. El icono aparecerá en el lado derecho superior de la pantalla sí se ha detectado la mano derecha o en el izquierdo superior de la pantalla sí ha sido con la mano izquierda. El proceso seguirá detectando gestos hasta que el usuario desactive la cámara.

En el apartado 3.3.5 se muestra la gestión del envío de teclas que se ha integrado en nuestra aplicación middleware. El uso de una clase *Appconector* centraliza la gestión de envío de mensajes y se encarga de inicializar la aplicación definida por el usuario.

Proceso de inicialización del Appconector

```
public appconector(string app, string processName)
        {
            appProcess = new Process { StartInfo = { FileName = app } };
            appProcess.Start();
            appProcess.WaitForInputIdle(5000);

            Process[] processlist = Process.GetProcesses();

            foreach (Process process in processlist)
            {
                if (!String.IsNullOrEmpty(process.MainWindowTitle))
                {
                    Debug.WriteLine("Process: {0} ID: {1} Window title: {2}",
process.ProcessName, process.Id, process.MainWindowTitle);
                }
            }
            Process[] processes = Process.GetProcessesByName(processName);
                // minimize soffice
                foreach (Process p in processes)
                {
                    // get mainwindow handle
                    IntPtr pFoundWindow = p.MainWindowHandle;
                    if (processName == "notepad")
                        pFoundWindow = FindWindowEx(p.MainWindowHandle, new IntPtr(0),
"Edit", null);
                hWComponent = pFoundWindow;
                //Use Handle to minimize
                if (!pFoundWindow.Equals(IntPtr.Zero))
                {
                    ShowWindowAsync(pFoundWindow, (int)WSTATUS.SW_SHOWNORMAL);
                }
            }
}
```

Para poder comunicar al usuario que se ha detectado un gesto correctamente se crea un hilo para el dibujado del icono. Se generarán tantos hilos como gestos se detecten. El ciclo de vida del hilo es muy corto, cuando se crea el hilo recibe como parámetros el icono que tiene que mostrar en mapa de bits y en que lado de la pantalla se ha de dibujar (ver anexo 7.5). El hilo dibuja el icono del gesto y este se desplaza hacia la parte superior de la pantalla para desaparecer de forma gradual (ver figura 3.20). Una que el gesto ha sido representado el hilo creado se destruye.

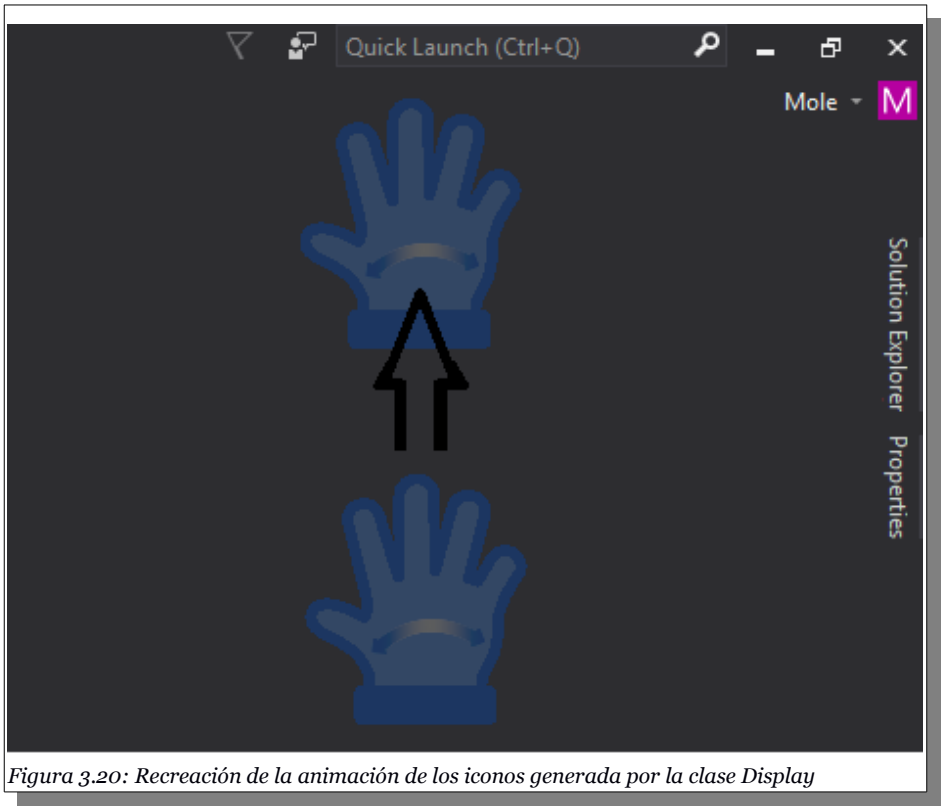

Figura 3.20: Recreación de la animación de los iconos generada por la clase Display

4 Resultados

En este apartado enumeraremos y evaluaremos los resultados de todos los tests que han sido realizados durante el desarrollo de la aplicación e indicaremos los resultados que hemos obtenido con la aplicación.

Como se ha mencionado anteriormente en el inicio del desarrollo de aplicación se utilizó el SDK R2. No todas las funcionalidades disponibles en el SDK R2 están disponibles en el SDK R3, ya que Intel las ha ido restringiendo o modificando. También Intel ha ido variando las herramientas que ofrece con cada SDK para procesar los datos obtenidos por la cámara. Por estos motivos los tests nos han sido útiles para seleccionar las funcionalidades que eran más interesantes incluir dentro de la aplicación y para saber qué nivel de precisión nos da la cámara.

Aunque la cámara se vende como un producto acabado al público, la verdad que parece más un producto para pruebas de laboratorio. Durante el desarrollo de la aplicación se ha comprobado la falta de estabilidad de la cámara y las pérdidas de comunicación con la misma durante su uso. Como estos fallos de comunicación son una constante nos centraremos en los resultados de los tests sin mencionar los fallos aleatorios que registramos en las pruebas.

4.1 Test de cálculo de distancias.

La cámara *RealSense* no dispone de un sistema integrado de cálculo de detección de objetos sencillos (esferas, cajas, superficies) o planos. Si quisiéramos implementar estas funcionalidades deberíamos acceder a la información RAW de los sensores y tomar esta información como un punto de partida para hacer los cálculos, pero este no es nuestro objetivo en el TFG. Nuestro objetivo es evaluar las funcionalidades que nos ofrece el SDK para la implementación de nuestra aplicación. Ese es el motivo por el que se ha usado el modo *hand shape* para poder usar la información del esqueleto de las manos (ver figura 4.1).

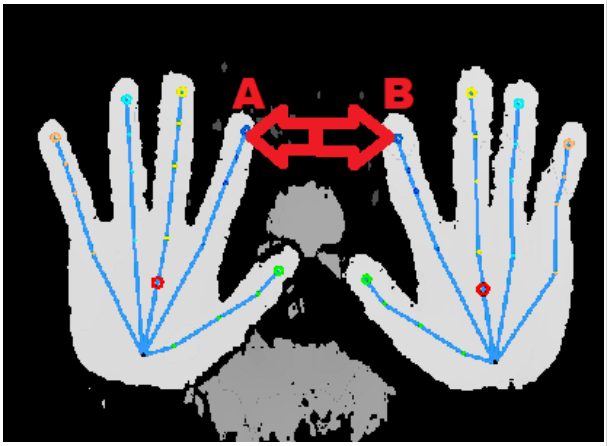

Figura 4.1: Cálculo de la distancia entre los dedos índices en 3D , extraída de la aplicación del TFG

Hemos tomado la información espacial de los dedos índices de las manos para calcular la distacia de separación entre ellos. Este es un método rudimentario para medir distancias. Se aplica la fórmula siguiente (ver figura 4.2) para encontrar la distancia entre A (x_1, y_1, z_1) y B (x_2, y_2, z_2).

$$AB = \sqrt{(x_2 - x_1) + (y_2 - y_1) + (z_2 - z_1)}$$

Figura 4.2: Fórmula de la distancia entre dos puntos en 3D

Con ayuda de una regla entre ambas manos y a una distancia de 50 cm de la cámara (ver figura 4.3 y 4.4), se obtienen los valores de la posición 3D de los dedos índices a través de sus marcadores. Sabiendo la posición 3D de los dedos índices y la distancia real entre ellos podemos establecer una correlación entre la distancia real y la distancia entre los puntos 3D de los dedos, aplicando una sencilla regla de tres. A la distancia obtenida se le ha de restar el grosor del dedo porque el marcador está posicionado en la mitad del dedo.

Código para el cálculo de distancias

```
double d = Math.Sqrt(Math.Pow(finguer1.x - finguer2.x, 2) + Math.Pow(finguer1.y -
finguer2.y, 2) + Math.Pow(finguer1.z - finguer2.z, 2));

double reald = (d * 17) / (0.207032777677396) ;
reald = reald - 0.5;
```

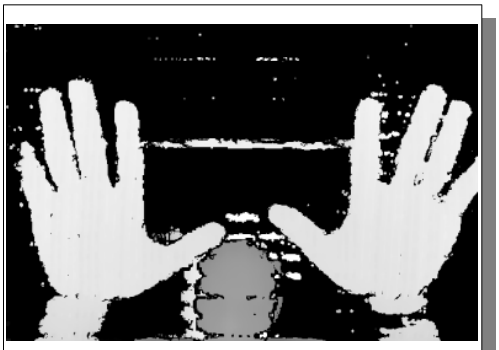

Figura 4.3: Calibración de la cámara, extraída del TFG

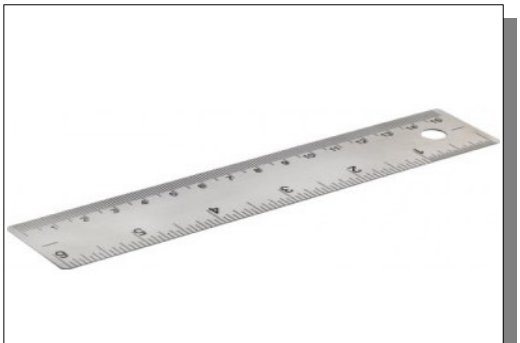

Figura 4.4: Regla usada para calibrar la cámara.

Se han realizado una serie de mediciones con los dedos índices separados entres sí a 16 cm y con las palmas de la mano paralelas al objetivo de la cámara (ver figura 4.3 y 4.4). El error en el cálculo crece conforme las manos se posicionan más alejadas de la cámara.

	Distancia de la cámara	**Distancia real entre índices**	**Distancia obtenida**
1.-	25 cm	16 cm	16,23 cm
2.-	50 cm	16 cm	16,50 cm
3.-	75 cm	16 cm	17,70 cm
4.-	100 cm	16 cm	No detecta las manos

Si se desea conocer más datos sobre la precisión de la cámara *Intel RealSense SR300* existe una publicación *On the Performance of the Intel SR300 Depth Camera [17]* donde se analiza la precisión y rendimiento de la cámara al detalle.

4.2 Test de captura de nube de puntos

En este test hemos querido comprobar la funcionalidad de la captura de nube de puntos para el posterior uso en aplicaciones 3D o para su uso en impresiones 3D. Se ha usado el ejemplo 3D scan para C# (ver figura 4.5) con la configuración para escaneo de cabeza con textura.

Figura 4.5: Ejemplo 3D scan (C#)

Una vez finalizado el escaneo se obtiene un fichero OBJ con la nube de puntos 3D y una textura obtenida del sensor RGB. Podemos ver los resultados por separado (ver figura 4.6 y 4.7).

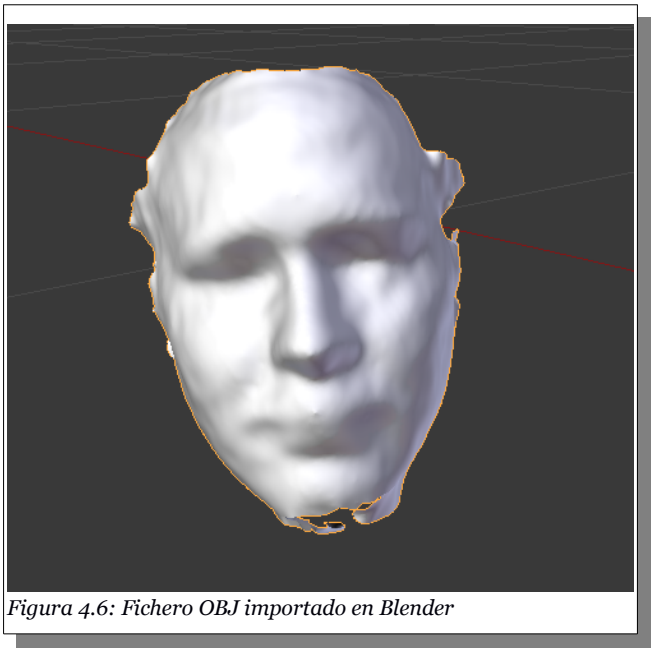

Figura 4.6: Fichero OBJ importado en Blender

Figura 4.7: Textura obtenida del escaneo

Mediante un programa de edición 3D podemos componer una escena con la nube de puntos y la textura. En este caso utilizamos Blender [27], herramienta open source, para visualizar los resultado en bruto sin retocar la escena. El resultado es aceptable (ver figura 4.9) teniendo en cuenta que el entorno de trabajo es casero con iluminación artificial que no evita las sombras en el modelo. En un entorno con una buena iluminación se pueden llegar a conseguir mejores resultados en cuanto a la calidad de la textura obtenida. Hay que destacar la calidad de la malla 3D ya que se ve bastante detallada y no presenta aberraciones o artefactos que deformen el objeto escaneado (ver figura 4.8).

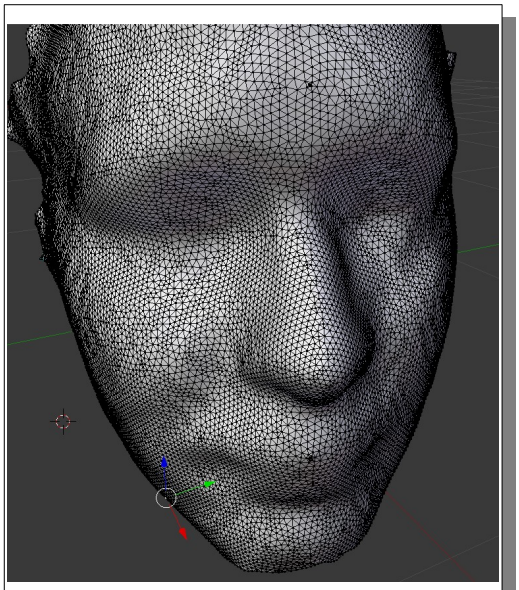

Figura 4.8: Representación nube de puntos con Blender

Figura 4.9: Combinación del OBJ y la textura con Blender

Hemos comprobado a través de una comparativa [26] de cámaras para escaneo 3D, que la cámara *RealSense SR300* ofrece una funcionalidad bastante avanzada y de calidad para los creadores de contenidos 3D que quieran escanear objetos pequeños. Hemos querido añadir una muestra del potencial de la cámara realizada en un estudio con una cuidada iluminación evitando sombras (ver figuras 4.10, 4.11 y 4.12).

Figura 4.10: Busto físico, extraída de 3Dscanexpert [26]

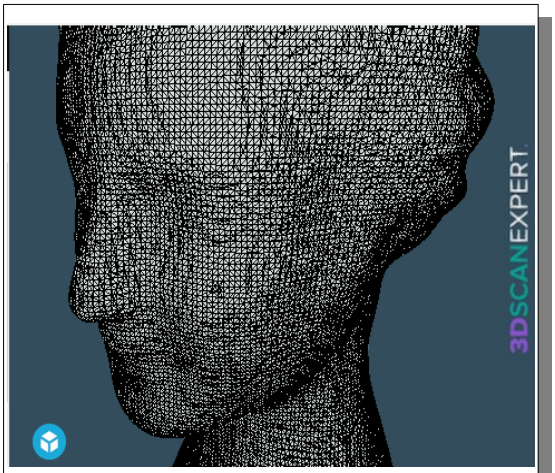

Figura 4.12: Malla 3d obtenida con SR300, extraída de 3Dscanexpert [26]

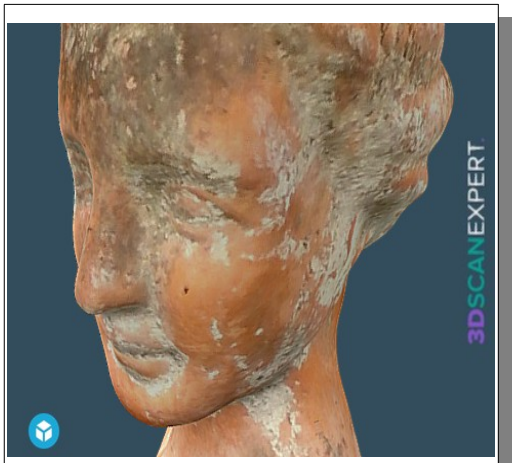

Figura 4.11: Render final, textura y malla obtenidas con SR300, extraída de 3Dscanexpert [26]

4.3 Test de detección de rostros

Hemos podido testear la opción de reconocimiento de rostros y se ha comprobado que la cámara tiene la capacidad de detectar hasta 4 caras de manera simultánea y permite registrar rostros. Estos rostros registrados se guardan en una base de datos interna de la aplicación para identificarlos durante el procesado de los siguientes cuadros (*frames)*. La cámara RealSense llega a reconocer a una persona registrada, llevando gafas o sin gafas perfectamente (ver figuras 4.13 y 4.14). Su uso es bastante sencillo y con una pocas líneas de código se puede identificar a una persona anteriormente registrada en tiempo real.

Hay una serie de reglas que se deben tener en consideración para que la detección y seguimiento 2D de rostros sea óptima[29]:

- Es importante que la iluminación sea buena, evitando las sombras, la iluminación a contraluz y la iluminación direccional fuerte, incluida la luz solar.
- Los movimientos tienen que ser lentos, y la distancia máxima, de 1,2 metros.
- No se debe inclinar la cabeza más de 30 grados (a lo largo de ningún eje) respecto de la pantalla.
- Hay que recordar que la cámara tiene un campo de visión limitado así que la mayor distancia será en el centro de la pantalla.

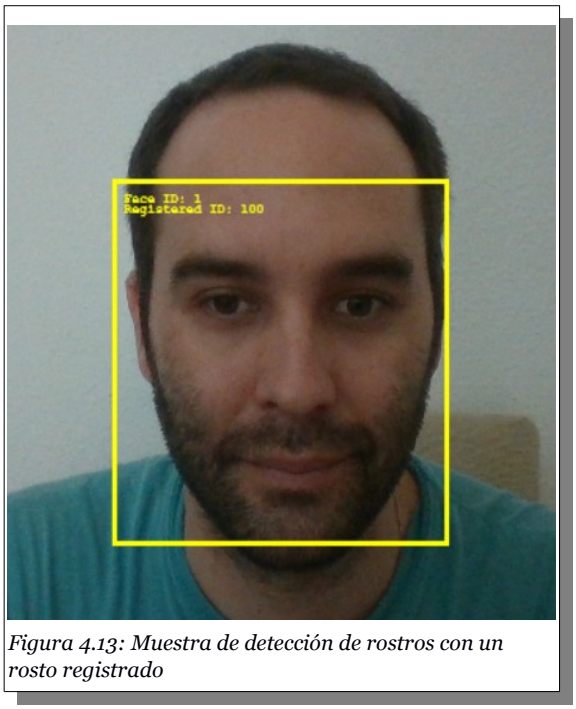

Figura 4.13: Muestra de detección de rostros con un rosto registrado

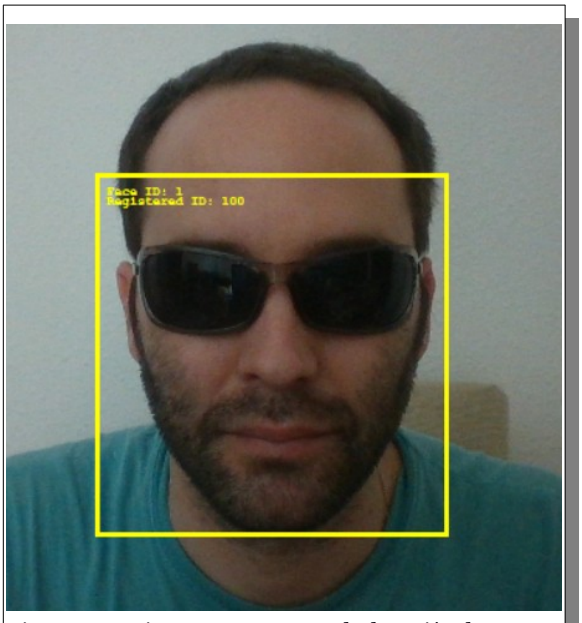

Figura 4.14: Figura 5.11: Muestra de detección de caras, rostro reconocido sin gafas ahora es identificado llevando gafas.

El único pero que se ha encontrado es que no se ha implementado correctamente el guardado de la base de datos interna a fichero [24] para reutilizarlo en posteriores ejecuciones de la aplicación, esto le resta bastante funcionalidad a la hora de integrarlo en la aplicación.

Aquí mostramos un fragmento de código extraído del SDK R3 [24], para ilustrar el problema detectado guardando la base de datos de las caras registradas a fichero.

Código para obtener la base de datos de rostros registrados

```
// rdata is a RecognitionData instance
// rcfg is a RecognitionConfiguration instance


// allocate the buffer to save the database

RecognitionModuleData rmd =rdata.RecognitionModule;

Int32 nbytes = rmd.DatabaseSize;

Byte[] buffer = new Byte[nbytes];


// retrieve the database buffer

buffer = rmd.DatabaseBuffer;
```

Para ilustrar de forma más detallada el error describiré los pasos para reproducirlo. Debemos acceder al módulo de reconocimiento, este es guardado sin ningún problema para acceder a sus objetos y propiedades mediante la instrucción *RecognitionModuleData rmd = rdata.RecognitionModule*. Posteriormente solicitamos que nos diga el tamaño que tiene la base de datos *Int32 nbytes = rmd.DatabaseSize*. Esta instrucción nos devuelve un valor no nulo si se ha registrado una cara con anterioridad. Podemos comprobar que este valor se ve incrementado a medida que vamos registrando más de una cara, por lo que en principio está devolviendo el tamaño correcto de la base de datos. El problema aparece cuando intentamos obtener el buffer donde guarda la base de datos, que está en memoria, mediante la instrucción *buffer = rmd.DatabaseBuffer*. Tras la ejecución del comando obtenemos un valor vacío *NULL*. Podemos comprobar en los foros de Intel la existencia de más usuarios del SDK R3 que han tenido el mismo problema [25].

4.4 Test de detección de gestos

El reconocimiento de gestos funciona con un alto grado de acierto tanto en el modo cursor como en modo *hand shape*. Para ese test se han usado los ejemplos del SDK *hands cursor viewer* (ver figura 4.16) y *hands viewer* (ver figura 4.15). La cámara *RealSense* informa sobre si ha detectado un gesto mediante alarmas, estas alarmas solo son disparadas si se habilitan en la configuración de la cámara. Hay que tener en cuenta que no es conveniente tener activados todo los gestos ya que para hacer algunos gestos tienes que pasar por otros gestos y puede ser un poco incontrolable. Por ejemplo, si quieres registrar cuando levantes el dedo gordo hacia arriba tienes que tener en cuenta que posiblemente en algún momento te detecte el puño cerrado. El rango de funcionamiento de la cámara es limitado ya que está pensado para que el usuario se encuentre sentado enfrente de la cámara.

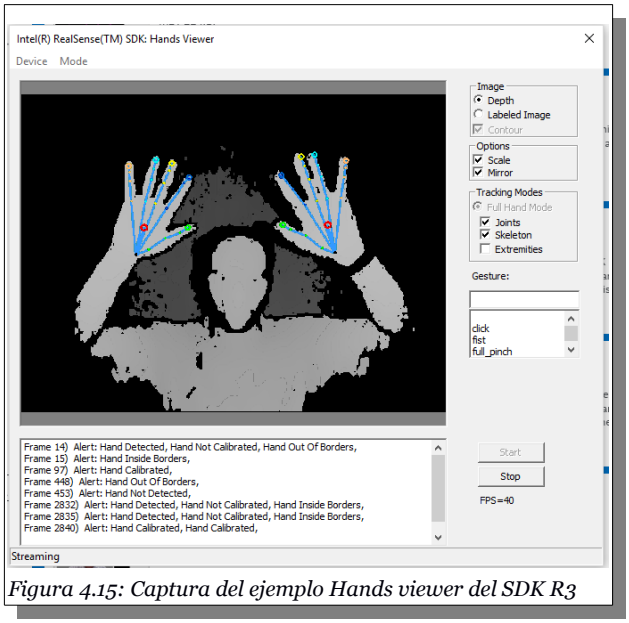

Figura 4.15: Captura del ejemplo Hands viewer del SDK R3

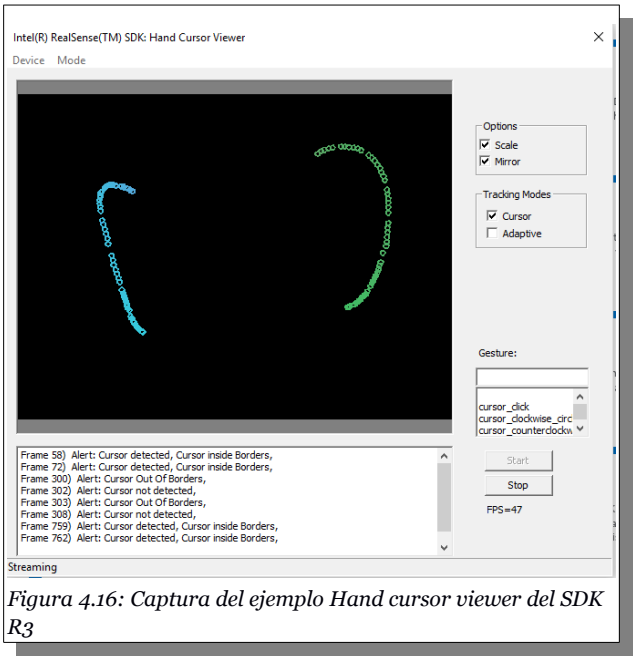

Figura 4.16: Captura del ejemplo Hand cursor viewer del SDK R3

4.5 Resultados obtenidos por la aplicación middleware

La funcionalidad que se ha querido implementar con el reconocimiento de caras es que la aplicación reconociera al usuario nada más arrancar. Este objetivo no ha podido ser implementado usando el SKD R3 [24] al no poder grabar la información del reconocimiento en disco para luego recuperarla en el arranque (ver apartado 4.3). Aunque el SDK tiene funciones diseñadas para extraer la información de las caras registradas y poder almacenarlas en disco, la función que debería devolver la información de la base de datos no funciona correctamente. Se ha implementado parte de la funcionalidad inicial y de esta manera podemos reconocer a la persona registrada entre múltiples caras pero hay que registrar el rostro al inicio de la aplicación de forma manual.

El reconocimiento de gestos funciona de forma fluida y con precisión, pero el usuario no puede estar muy alejado de la cámara para que sea detectado, con distancias superiores a 1m ya no son identificados los gestos. El sistema de representación de gestos trabaja en un hilo independiente a la aplicación principal lo que esto permite que si hay un alto número de gestos detectados en un corto periodo de tiempo no bloquee la aplicación. Los gestos detectados son mostrados al usuario mediante un icono representativo del gesto y se dibujan por encima del resto de aplicaciones desplazándose la parte superior del monitor y desaparecen con un fundido. Esta forma de representarlo hace que la representación de múltiples gestos consecutivos no se oculten unos a otros. Por ejemplo, si haces el gesto de click probablemente detecte también el gesto de puño (ver figura 4.17) y serán visibles los dos iconos de los dos gestos.

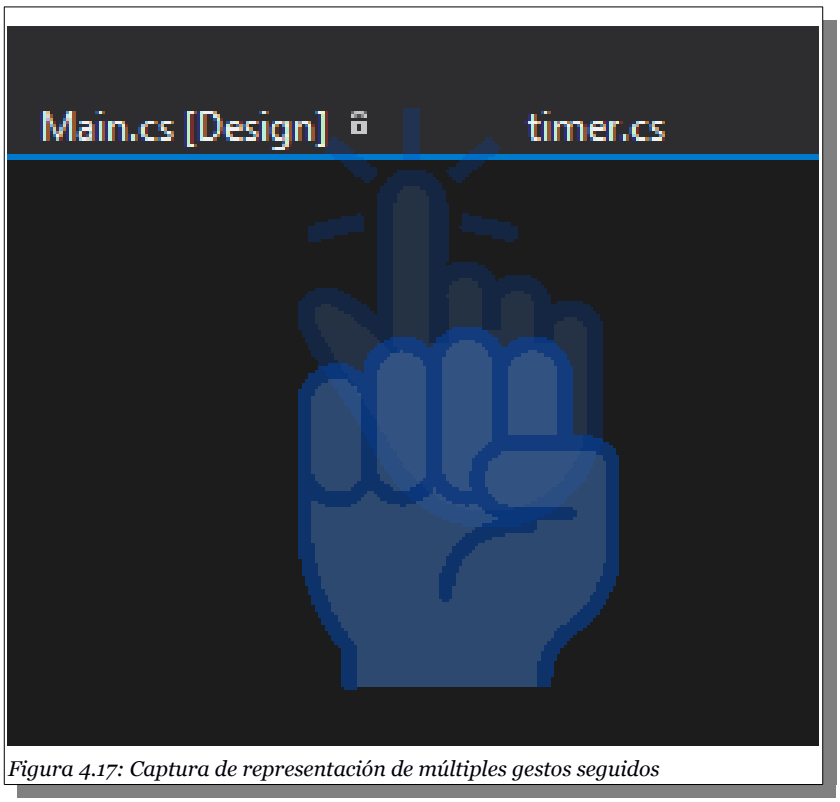

Figura 4.17: Captura de representación de múltiples gestos seguidos

El envío de mensajes simulando teclas a aplicaciones de terceros es más complejo de lo esperado. Mi anterior experiencia con el envío de mensajes para simular pulsaciones de teclas había sido con aplicaciones sencillas como el *notepad* de windows al que le mandas los mensajes al proceso que arranca la aplicación, y la aplicación recibe la pulsación comportándose como si pulsaras la tecla del teclado en la aplicación. Este método no funciona en aplicaciones más complejas como puede ser *Impress* de *Libre Office* [24], donde el proceso que arranca la aplicación va generando múltiples procesos que es la forma con la que lo han diseñado para gestionar múltiples documentos a la vez.

Simplificando el problema, cuando ejecutas una aplicación se crea un proceso, a ese proceso le mandas los mensajes de teclas, cuando el proceso los recibe los trata como pulsaciones reales de teclado. En el caso de *Impress* se invoca al ejecutable *simpress.exe*. El proceso muere pero antes genera otros dos procesos más *soffice.exe* y soffice.bin. De estos dos procesos el que debe de recibir los mensajes de pulsación de teclas es *soffice.bin*. ¿Cómo se puede averiguar a que proceso de los dos se ha de enviar los mensajes? Una solución es por prueba y error, otra solución es usando un analizador gráfico como *Microsoft Spy++*[25]. También puedes hacer tu propia utilidad para explorar los procesos activos y así sacar el nombre del proceso correcto.

Código para obtener información de los procesos activos

```
Process[] processlist = Process.GetProcesses();
foreach (Process process in processlist)
{
 if (!String.IsNullOrEmpty(process.MainWindowTitle))
 {
  Debug.WriteLine("Process: {0} ID: {1} Window title: {2}", process.ProcessName, process.Id, process.MainWindowTitle);
 }
}
```

Salida por consola de la información de los procesos activos

```
Process: devenv ID: 9180 Window title: intelSR (Running) - Microsoft Visual Studio
Process: SnippingTool ID: 11200 Window title: Herramienta Recortes
Process: WinStore.App ID: 8468 Window title: Microsoft Store
Process: SystemSettings ID: 7152 Window title: Configurac i n
Process: mspaint ID: 7792 Window title: Sin t tulo - Paint
Process: Taskmgr ID: 16120 Window title: Administrador de tareas
Process: mspaint ID: 17720 Window title: a.png - Paint
Process: firefox ID: 14372 Window title: WhatsApp - Mozilla Firefox
Process: soffice.bin ID: 14040 Window title: Proyecto.odt - LibreOffice Writer
Process: explorer ID: 7864 Window title: doc
Process: notepad ID: 13448 Window title: Sin t tulo: Bloc de notas
Process: mspaint ID: 13868 Window title: Sin t tulo - Paint
```

En gran medida la estructura de la aplicación de terceros que va a recibir los mensajes fija el nivel de dificultad para que un usuario no avanzado sea capaz de generar una configuración para que la aplicación middleware funcione correctamente, y la aplicación que controla reciba correctamente los mensajes de las pulsaciones de teclas (ver Anexo 7.2).

Aquí tenemos unas capturas de *Microsoft Spy++* analizando el árbol de procesos para poder ver sus componentes. Se puede observar que el *notepad* tiene tres componentes gráficos: la ventana del programa, la parte donde se escribe, Edit, y una barra de estado (ver figura 4.18).

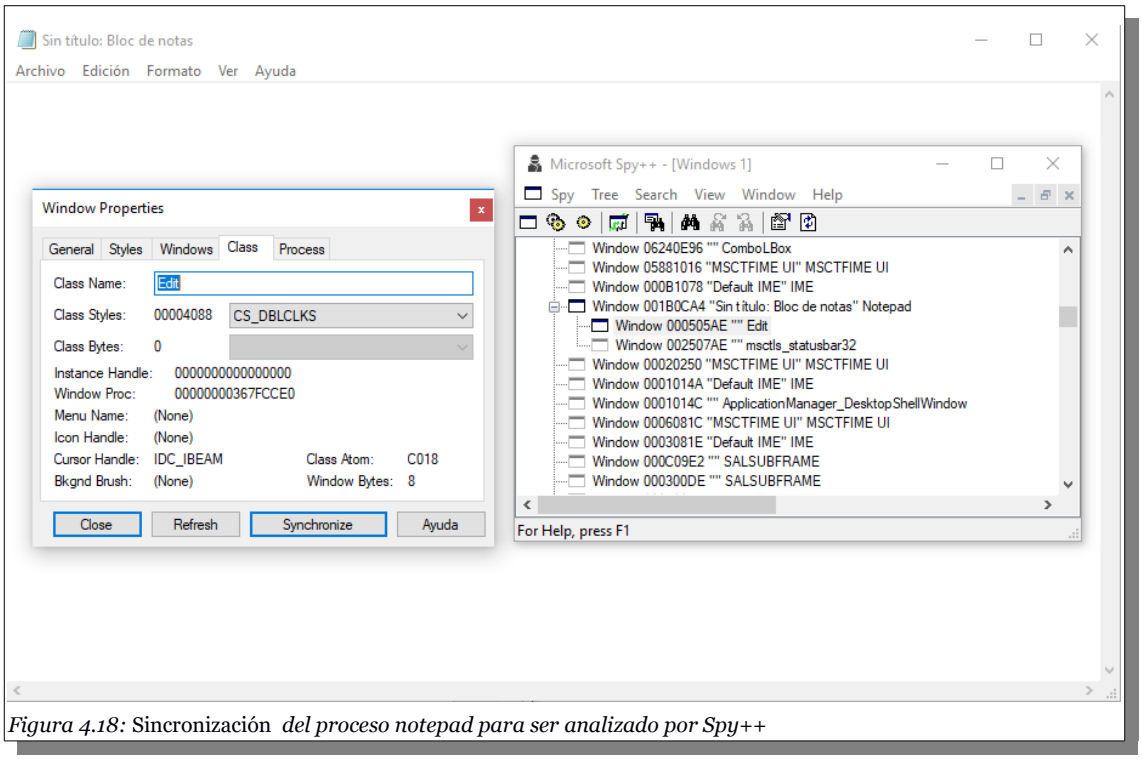

Figura 4.18: Sincronización *del proceso notepad para ser analizado por Spy++*

En esta otra captura de *Microsoft Spy++* se están interceptando los mensajes recibidos por la aplicación. En este caso son pulsaciones de teclas, como las que vamos a enviar desde nuestra aplicación. Se puede apreciar que se ha escrito la palabra “hola”, cada carácter pulsado genera 3 mensajes WM_KEYDONW, WM_CHAR y WM_KEYUP (ver figura 4.19).

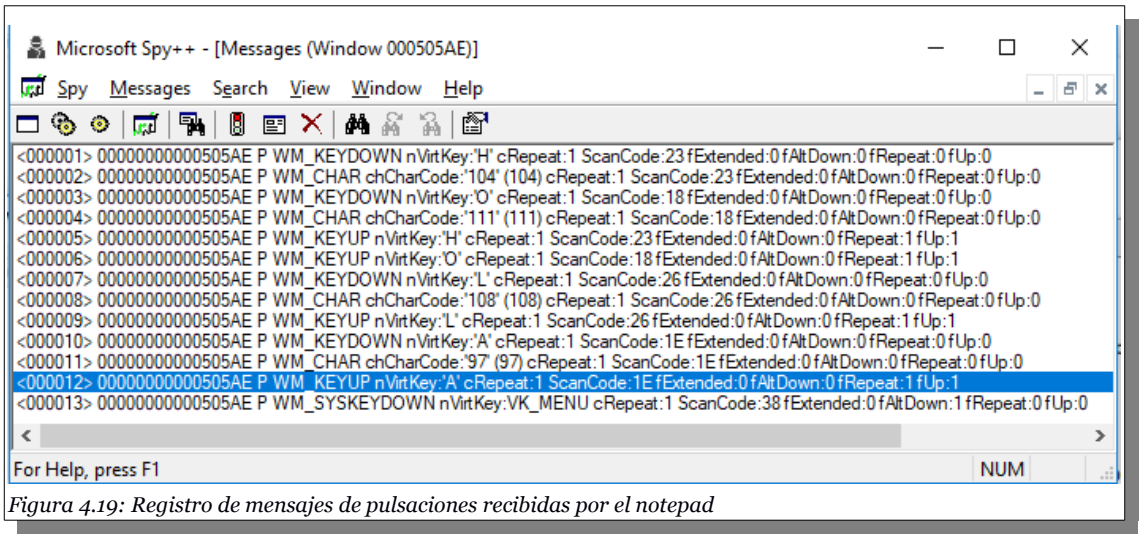

Figura 4.19: Registro de mensajes de pulsaciones recibidas por el notepad

5 Conclusiones

El desarrollo de la aplicación para RealSense SR300 (ver apartado 3.4) ha sido muy gratificante al poder comprobar cómo ha evolucionando la tecnología en la detección, reconocimiento y seguimiento en tiempo real de objetos y personas. La programación de la cámara no es muy compleja, pero hay que tener conocimientos de cómo gestionar hilos y su comunicación entre sí.

Se han conseguido los objetivos propuestos para el desarrollo de la aplicación interactiva con *Intel RealSense SR300* mencionados en el apartado 1.2. Hemos formulado el diseño y desarrollo de una aplicación interactiva con *Intel RealSense SR300*, que funcionará a modo de *middleware*, procesando la información de la cámara para interactuar con el usuario y enviando mensajes para interactuar con las aplicaciones configuradas. La interacción será posible a nivel de detección de posición y de gestos de la mano (incluyendo el esqueleto de las manos detectadas) y el reconocimiento de rostros.

5.1 Valoración personal

En mi opinión es complicado hacer una herramienta genérica que simule pulsaciones de teclas en aplicaciones de terceros y que a la vez sea sencilla para el usuario. Aunque la cámara ha respondido de forma fluida en todas las pruebas, tiene serios problemas de estabilidad. Respecto a estos problemas de estabilidad he llegado a la conclusión que son debidos a una mezcla entre el controlador de Intel de la cámara y las actualizaciones del sistema Windows 10. Lamentablemente, Intel ha bajado la prioridad del soporte del SDK de la cámara *RealSense SR300* para centrarse en nuevos modelos de cámaras con sensor de profundidad como los nuevos modelos de cámaras D400 series [13].

La cámara *Intel RealSense SR300* es un dispositivo válido para desarrolladores pero no es un producto maduro para el público en general. Es necesario pulir los controladores para que sean más estables y no provoquen quebraderos de cabeza a los usuarios domésticos.

5.2 Futuras ampliaciones y mejoras

Una vez finalizado el TFG y teniendo en consideración las dificultades y los retos de la programación del *middleware* que he encontrado, le daría un cambio de filosofía al programa.

Una opción sería que en lugar de hacerla genérica dando la posibilidad de definir las aplicaciones que quieres controlar y qué teclas mandar, le ofrecería al usuario una lista cerrada de aplicaciones que pudiera controlar. Por cada aplicación que se quisiera controlar habría definida una máquina de estados, en está máquina de estados se especificaría cuando los gestos estarían activos. De ese modo podrían dejar de registrarse gestos que no queremos que interfieran en ciertos momentos o cambiar la tecla de forma dinámica durante la ejecución. Por ejemplo, sí la cámara detecta un

puño enviaría una tecla que pondría la aplicación que controlamos a pantalla completa, sí a partir de este momento queremos que el gesto de puño envíe una tecla distinta deberemos definir dos estados. Un estado para cuando inicias la aplicación donde el gesto puño manda una tecla, y otro estado para el modo presentación donde el gesto puño tendría asociado otro valor de tecla distinto al inicial.

Una ampliación interesante sería integrar la aplicación con aplicaciones de terceros mediante sus API y mediante el paso de mensajes de Windows a ventanas. Durante el desarrollo del TFG se estudió la posibilidad de embeber dentro de la aplicación una instancia de *Google Earth* mediante su API, pero Google restringió la posibilidad de embeberlo dentro de la otra cuando sacaron la versión profesional [28]. Este hecho que desconocía, añadido a la limitación de tiempo para investigar la nueva API de Google Earth Pro, hizo que descartara esta opción en las primeras fases de desarrollo.

6 Bibliografía

[1]. *Brainstorm. Edison education software.* <http://education.brainstorm3d.com/products/edison/> [Consulta: 12 de Junio de 2018]

[2]. Dell. Características OMEN Laptop - *15t gaming especificaciones*. <http://www8.hp.com/us/en/campaigns/omenlaptop/overview.html?jumpid=in_r12135_us/en/psg/gaming_pcs_hp_omen_family/omen-laptops-slide-learn-more-button> [Consulta: 12 de Junio de 2018]

[3]. Intel. *Intel RealSense SDK 2016 R2 Documentation.* <https://software.intel.com/sites/landingpage/realsense/camera-sdk/v1.1/documentation/html/index.html?doc_devguide_introduction.html> [Consulta: 11 de Junio de 2018]

[4]. Intel. *Intel RealSense SDK 2016 R3 Documentation.* <https://software.intel.com/sites/landingpage/realsense/camera-sdk/v2016r3/documentation/html/index.html?doc_devguide_introduction.html> [Consulta: 11 de Junio de 2018]

[5]. Intel. Intel RealSense Requisitos hardware. <https://www.intel.la/content/www/xl/es/support/articles/000020964/emerging-technologies/intel-realsense-technology.html> [Consulta: 11 de Junio de 2018]

[6]. Intel. Intel RealSense SDK for Windows. <https://software.intel.com/en-us/realsense-sdk-windows-eol> [Consulta: 11 de Junio de 2018]

[7]. Intel. Intel RealSense Depth Camera Manager. <https://downloadcenter.intel.com/download/25044/Intel-RealSense-Depth-Camera-Manager?product=92329> [Consulta: 11 de Junio de 2018]

[8]. Intel. Intel RealSense SDK Architecture. <https://software.intel.com/en-us/articles/intel-realsense-sdk-architecture> [Consulta: 11 de Junio de 2018]

[9]. Intel. *Introducción a la cámara Intel RealSense SR300.* <https://software.intel.*com*/es-es/realsense/sr300/get-started> [Consulta: 11 de Junio de 2018]

[11]. Intel. *A Comparison of Intel RealSense Front-Facing Camera SR300 and F200* <https://software.intel.com/en-us/articles/a-comparison-of-intel-realsensetm-front-facing-camera-sr300-and-f200> [Consulta: 12 de Junio de 2018]

[12]. *Intel communities. How to get SR300 working on Windows 10 Build 1703* <https://communities.intel.com/thread/113973> [Consulta: 12 de Junio de 2018]

[13]. *Intel. Intel RealSense serie D400* <https://software.intel.com/es-es/realsense/d400> [Consulta: 12 de Junio de 2018]

[14]. *Microsoft. API Microsoft Windows, Clase SendKeys.* <https://msdn.microsoft.com/es-es/library/system.windows.forms.sendkeys(v=vs.110).aspx> [Consulta: 12 de Junio de 2018]

[15]. *Microsoft. API Microsoft Windows, Virtual-Key Codes.* <https://docs.microsoft.com/es-es/windows/desktop/inputdev/virtual-key-codes> [Consulta: 12 de Junio de 2018]

[16]. *Microsoft. Guía C# Delegados.* <https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/delegates/using-delegates> [Consulta: 12 de Junio de 2018]

[17]. Monica, C. et al (2017). On the Performance of the Intel SR300 Depth Camera*: Metrological and Critical Characterization* en IEEE Xplore Digital Library. <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7929364> [Consulta: 12 de Junio de 2018]

[18]. Wikipedia. Lanzamiento de la consola Wii. <https://es.wikipedia.org/wiki/Wii#Lanzamiento> [Consulta: 12 de Junio de 2018]

[19]. Wikipedia. Lanzamiento de Kinect. <https://es.wikipedia.org/wiki/Kinect#Lanzamiento > [Consulta: 12 de Junio de 2018]

[20]. Intel. Web oficial RealSense SR300. <*https://software.intel.com/es-es/realsense/sr300*> [Consulta: 12 de Junio de 2018]

[21]. Mercurial. Web oficial Mercurial. <https://www.mercurial-scm.org> [Consulta: 12 de Junio de 2018]

[22]. Intel. Web oficial TortoiseHG. <https://tortoisehg.bitbucket.io> [Consulta: 12 de Junio de 2018]

[23]. Xataka Blog. Leap motion sale a la venta. <https://www.xataka.com/otros/leap-motion-ya-esta-disponible> [Consulta: 12 de Junio de 2018]

[24]. Intel. SDK R3, *Face Recognition. Loading/Saving Recognition Database.* <https://www.xataka.com/otros/leap-motion-ya-esta-disponible> [Consulta: 12 de Junio de 2018]

[25]. *Intel community*. Foro: *QueryDatabaseBuffer(buffer) return null.* <*https://communities.intel.com/thread/111846*> [Consulta: 11 de Junio de 2018]

[24]. Libre Office Fundation. *Web* oficial *Impress*. <https://es.libreoffice.org/descubre/impress/> [Consulta: 12 de Junio de 2018]

[25]. Microsoft MSDN. Introducción a Spy++. <https://msdn.microsoft.com/es-es/library/dd460756.aspx> [Consulta: 12 de Junio de 2018]

[26]. 3Dscanexper Blog . Structure Sensor vs. *Intel RealSense SR300* vs. Kinect V2. <https://3dscanexpert.com/structure-sensor-realsense-sr300-kinect-v2-3d-scanning/> [Consulta: 12 de Junio de 2018]

[27]. Blender Software. Web oficial de Blender. <https://www.blender.org/> [Consulta: 12 de Junio de 2018]

[28]. Google Blog . *Announcing deprecation of the Google Earth API* . <https://mapsplatform.googleblog.com/2014/12/announcing-deprecation-of-google-earth.html> [Consulta: 12 de Junio de 2018]

[29]. Intel Blog. Seguimiento de rostro y cabeza con el SDK para Intel RealSense. <https://software.intel.com/es-es/blogs/2015/02/09/seguimiento-de-rostro-y-cabeza-con-el-sdk-para-intel-realsense> [Consulta: 12 de Junio de 2018]

[30]. Nuitrack. Web oficial nuitrack. <https://nuitrack.com> [Consulta: 12 de Junio de 2018]

[31]. Intel Blog. Web oficial curvsurf. <http://www.curvsurf.com/> [Consulta: 12 de Junio de 2018]

[33]. Gestoos. Web oficial gestoos. <http://gestoos.com> [Consulta: 12 de Junio de 2018]

[34]. Intel. *Datasheet* de la cámara Intel RealSense. <https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/realsense-sr300-datasheet1-0.pdf> [Consulta: 12 de Junio de 2018]

[35]. Asus.Web de Xtion Pro Live. <https://www.asus.com/es/3D-Sensor/Xtion_PRO_LIVE/> [Consulta: 12 de Junio de 2018]

[36]. Asus.Web de structure. <https://structure.io/support/what-are-the-structure-sensors-technical-specifications> [Consulta: 12 de Junio de 2018]

[37]. Intel . *Release Notes for Intel RealSense Depth Camera Manager (DCM) SR300 3.3*. <https://downloadmirror.intel.com/25044/eng/SR300_3_3_Release_Notes.pdf> [Consulta: 12 de Junio de 2018]

[38]. Intel . SDK R3 FaceExpression. <https://software.intel.com/sites/landingpage/realsense/camera-sdk/v2016r3/documentation/html/index.html?faceexpression_expressionsdata_pxcfacedata.html> [Consulta: 12 de Junio de 2018]

7 Apéndices, anexos

7.1 Estructura del proyecto con RealSense SR300 en C#

Vamos a indicar los pasos básicos de configuración de un proyecto que nos permita usar el *SDK Intel RealSense* y mostraremos la estructura genérica del bucle de procesado de un módulo de la cámara RealSense.

Lo primero será crear Proyecto, un proyecto nuevo de tipo *windows form application*. A este proyecto hay que añadir las referencias a los módulos que dan acceso a las funcionalidades que se van a usar en la aplicación (ver figura 7.1). Además, en las propiedades del proyecto hay que activar la opción *Allow unsafe mode*, para poder compilar (ver figura 7.2).

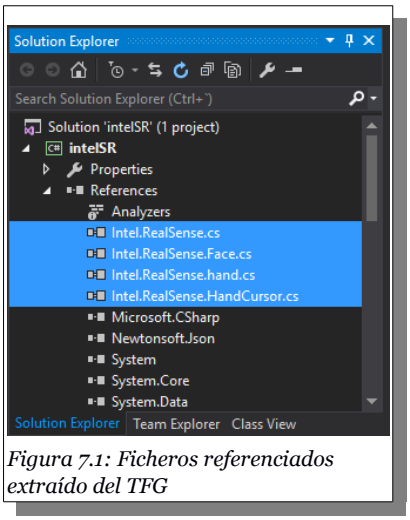

Figura 7.1: Ficheros referenciados extraído del TFG

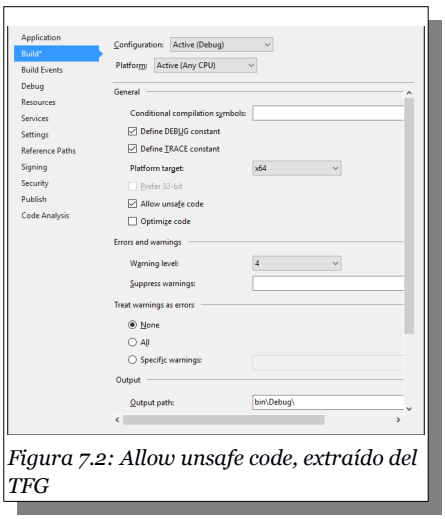

Figura 7.2: Allow unsafe code, extraído del TFG

En este caso vamos a adjuntar el módulo principal, seguimiento y reconocimiento de manos y caras a las referencias del proyecto. Mostramos aquí la ubicación de los ficheros necesarios para crear las referencias:

- Intel rs sdk mw core: Es el núcleo del SDK (imprescindible para el desarrollo).
  C:\Program Files (x86)\Intel\RSSDK\bin\x64\Intel.RealSense.cs.dll
  C:\Program Files (x86)\Intel\RSSDK\bin\x64\libpxcclr.cs.dll
- Intel rs sdk mw face: Módulo de seguimiento de caras y reconocimiento.
  C:\Program Files (x86)\Intel\RSSDK\bin\x64\Intel.RealSense.Face.cs.dll
- Intel rs sdk mw hand: Módulo de seguimiento de manos.
  C:\Program Files (x86)\Intel\RSSDK\bin\x64\Intel.RealSense.hand.cs.dll
- Intel rs sdk mw cursor: Módulo cursor de manos.
  C:\Program Files (x86)\Intel\RSSDK\bin\x64\Intel.RealSense.HandCursor.cs.dll

Para empezar a usar la cámara primero se ha de crear una sesión para al control de la cámara Real Sense SR300, puede ser como la del código adjunto.

```
Intel.RealSense.Session session = Intel.RealSense.Session.CreateInstance();
if (session != null)
{
  Application.Run(new MainForm(session));
  session.Dispose();
}
```

Una vez inicializado tiene que invocarse un hilo *thread* para que la aplicación pueda ser redibujada y no se bloquee mientras se está accediendo a la cámara.

```
var thread = new Thread(DoTracking);
   thread.Start();


private void DoTracking()
{
  var ft = new FaceTracking(this);
ft.SimplePipeline();
}
```

Dentro del *thread* se configuran las funciones de la cámara para el seguimiento y reconocimiento, en este caso de caras. Dentro del *thread* de control de la cámara se han de realizar todos los cálculos en un bucle hasta que la aplicación le informe que pare.

```
class FaceRecognition
{
  public void SimpleFacePipeline()
  {
    SenseManager pp = _form.session.CreateSenseManager();
   /// .......
    CaptureManager captureMgr = pp.CaptureManager;
  //Zona de configuración
   if (applyChangesStatus < Status.STATUS_NO_ERROR || pp.Init() <
Intel.RealSense.Status.STATUS_NO_ERROR)
    _form.UpdateStatus("Init Failed");
  }
  else
  {
    using (Intel.RealSense.Face.FaceData moduleOutput = faceModule.CreateOutput())
    {
          /// .......
      while (!_form.stop)
       {
            // bluce de procesado
        /// .......
         pp.ReleaseFrame();
       }
    }
 /// Zona de destrucción del modulo
   }
}
```

7.2 Ejemplo de fichero de configuración JSON

Especificación de la estructura del fichero "`programs.json`" donde se define la configuración de la aplicación. Este fichero de configuración debe estar en la misma ruta del ejecutable para que pueda ser leído por la aplicación.

```
[{
	"Name" : "PowerPoint",
	"Exe"  : "C:\\Program Files\\LibreOffice\\program\\simpress.exe",
	"ProcessName" : "soffice.bin",
	"DrawType"	: "Shape",
	"fist" : "",
	"click" : "",
	"full_pinch" : "",
	"spreadfingers" : "",
	"swipe_down" : "VK_F5",
	"swipe_left" : "",
	"swipe_rigth" : "",
	"swipe_up" : "VK_LEFT",
	"tap" : "VK_RIGHT",
	"thumb_down" : "",
	"thumb_up" : "",
	"two_fingers_pinch_open" : "",
	"v_sign" : "",
	"wave" : "VK_ESCAPE",
	"cursor_click" : "VK_RIGHT",
	"cursor_clockwise_circle" : "VK_LEFT",
	"cursor_hand_closing" : "VK_RIGHT",
	"cursor_counter_clockwise_circle" : "VK_ESCAPE",
	"cursor_hand_opening" : ""
}]
```

7.3 Limitaciones del controlador *RealSense Depth Camera Manager*

Algunos problemas conocidos del controlador *RealSense Depth Camera Manager (DCM) SR300* v3.3.27.5718 [37].

Issue	Recovery
IR Stop working after Improve recognition while select camera on skype that using Depth camera	Known limitation
Windows Hello might not work when a DCM of more than one camera type is installed	Windows limitation
Low FPS with multi SDK applications	If an application is not working properly, leave only the required application open
Depth device might disappear from Device Manager if shutting down the system during driver update	Known limitation
When SDK application updates a property the new value is not updated in Universal App	Close the app and open a new universal app
FW Upgrade can be done for up to two cameras that are connected	Known limitation
Replacing the camera while DCM is being installed is not supported	Uninstall DCM and remove the following key from registry: HKEY_LOCAL_MACHINE\SOFTWARE\Wow6 432Node\Intel\RSSDK\Components\dcmservice_sr300\IntegratedCamera

7.4 Instalación del wrapper Newtonsoft.Json

Newtonsoft.Json es un *wrapper* para facilitar la lectura y manipulación de ficheros JSON, para la instalación de este componente gratuito seguiremos unos sencillos pasos.

Iremos a la opción *manage NuGet package for solution* que se encuentra en el menú *Tools→NuGet package Manager*. Una vez abierto el gestor de NuGet iremos al apartado Browse y escribiremos el nombre del componente, Newtonsoft.Json. Aparecerá con su descripción y nos dará la opción de instalarlo (ver figura 7.3).

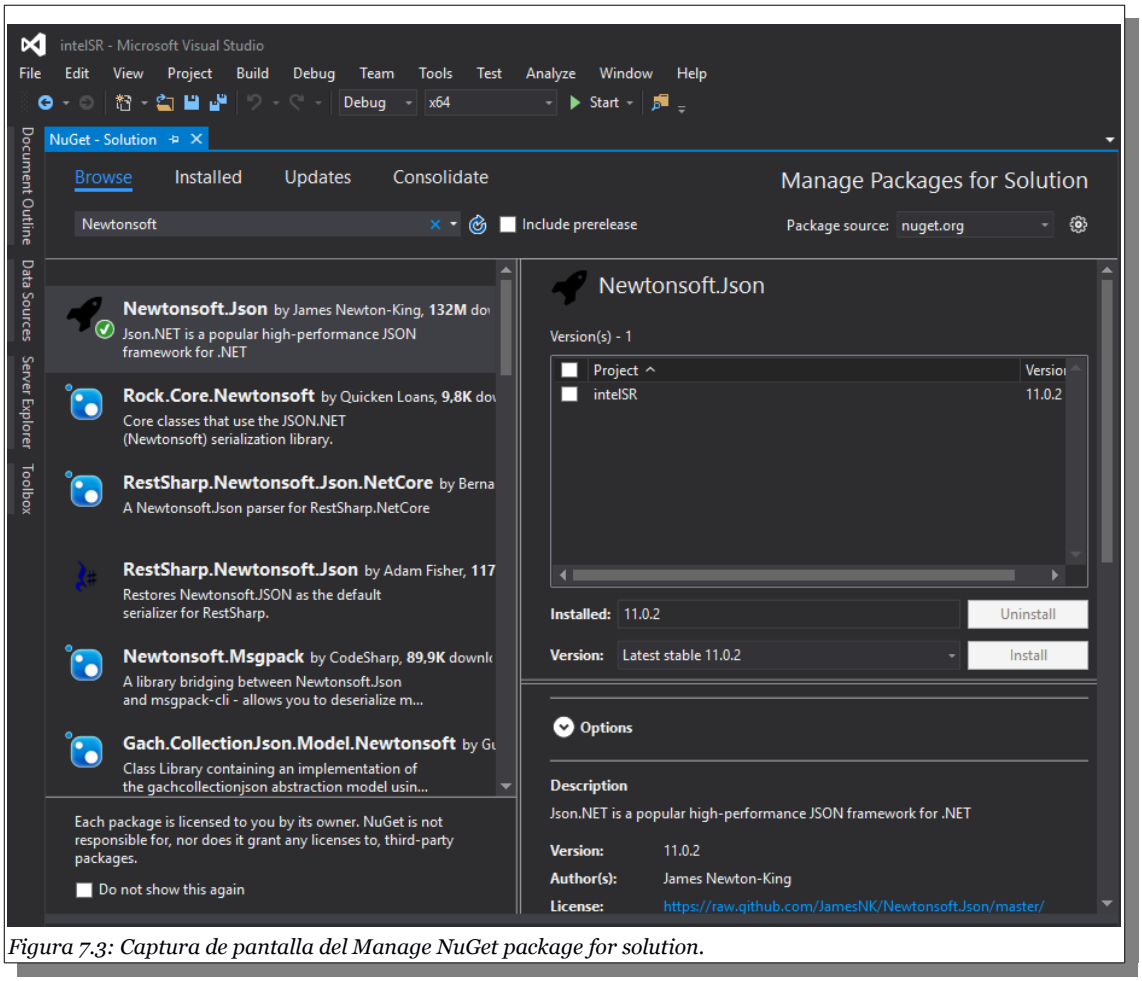

Figura 7.3: Captura de pantalla del Manage NuGet package for solution.

7.5 Implementación del hilo Display

El hilo display se encarga de comunicar al usuario los gestos detectados y tiene como finalidad evitar del bloqueo de la aplicación.

Ejemplo de invocación del hilo

```
private delegate void SendMsgDelegate(string gesture, string side);

public void SendMsg(string gesture,string side)
{
   this.Invoke(new SendMsgDelegate(delegate (string s, string p)
   {
     System.Drawing.Image backimage =
(System.Drawing.Image)Resources.ResourceManager.GetObject(gesture.ToLower());
     new Display(backimage ,side);

     if (app != null)
       app.pressKey(cf.getGestureKey(gesture));
   }), new object[] { gesture, side });
 }
```

Implementación del hilo Display

```
namespace intelSR
{
  public partial class Display : Form
  {
    private Timer timerShow = null;
    private Timer timerWaiting = null;
    private Timer timerHide = null;

    public ImageLayout Stretch { get; private set; }

    public Display()
    {
      InitializeComponent();
      Opacity = 0;      //first the opacity is 0
    }

    public Display(Image backimage, String side)
    {
      BackgroundImage = backimage;
      ConfigureForm(side);
      Opacity = 0;
      Show();
      showGesture();
    }

    private void ConfigureForm(String side)
    {
      BackColor = Color.White;
      TransparencyKey = Color.White;
      this.StartPosition = FormStartPosition.Manual;
      BackgroundImageLayout = Stretch;

      FormBorderStyle =  System.Windows.Forms.FormBorderStyle.None;

      TopMost = true;
      Size = new Size(128, 128);
      switch (side)
      {
```

```
            case "left":
                this.Location = new Point(1600, 200);

                break;
            case "right":
                this.Location = new Point(200, 200);
                break;
            default:
                break;
        }
    }

    private delegate void EmptyDelegate();

    private void showGesture()
    {
        if (timerShow == null)
        {
            timerShow = new Timer();
            timerShow.Tick += new EventHandler(fadeIn);  //this calls the function that changes opacity
        }

        timerShow.Interval = 10;  //we'll increase the opacity every 10ms
        timerShow.Start();
    }

    void fadeIn(object sender, EventArgs e)
    {
        if (Opacity >= 1)
        {
            timerShow.Stop();   //this stops the timer if the form is completely displayed
            waiting();
        }
        else
        {
            Opacity += 0.03;
            Top -= 1;
        }
    }
    private void waiting()
    {
        if (timerWaiting == null)
        {
            timerWaiting = new Timer();
            timerWaiting.Tick += new EventHandler(waitGesture);  //this calls the fade out function
        }
        timerWaiting.Interval = 500;
        timerWaiting.Start();
    }

    void waitGesture(object sender, EventArgs e)
    {
        timerWaiting.Stop();
        hidegesture();
    }

    private void hidegesture()
    {
        if (timerHide == null)
        {
            timerHide = new Timer();
            timerHide.Tick += new EventHandler(fadeOut);  //this calls the fade out function
        }
        timerHide.Interval = 10;
        timerHide.Start();
```

```
    }

    void fadeOut(object sender, EventArgs e)
    {
        if (Opacity <= 0)     //check if opacity is 0
        {
            timerHide.Stop();    //if it is, we stop the timer
            this.Dispose();   //and we try to close the form
        }
        else
        {
            Opacity -= 0.03;
            Top -= 2;
        }
    }
  }
}
```