UNIVERSIDAD POLITÉCNICA DE VALENCIA

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

Máster Universitario en Automática e Informática Industrial

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

Implementación de una aplicación de tracking, con el fin de conocer la posición espacial de una persona y comparación entre diferentes cámaras y técnicas de visión artificial.

TRABAJO FINAL DE MÁSTER

Autor/a:

Eric Beaucamps Santofimia

Tutor/a:

Eugenio Ivorra Martínez

Resumen

En el siguiente proyecto se pretende implementar y analizar diferentes técnicas de seguimiento de uno o varios usuarios en un espacio conocido devolviendo, de esta forma, datos 3D acerca de la ubicación y desplazamiento de éstos. A su vez, se desarrollará un método de comunicación a través de la red, entre aplicaciones y dispositivos para hacer uso de dichos datos de desplazamiento.

Con el fin de comparar diferentes metodologías, se van a implementar tres aplicaciones de tracking haciendo uso de diferente hardware y técnicas de visión artificial. Por un lado, se hará uso de un equipo semi profesional que constará de una cámara 3D, la cual cuenta con diferentes sensores que le permiten construir un mapeado de profundidad y de esta manera obtener las coordenadas de los usuarios de la sala de manera precisa.

Por otro lado, se empleará un modelo mucho más cotidiano que empleará webcams ordinarias para realizar el seguimiento del usuario. De esta manera, se implementarán dos aplicaciones haciendo uso de webcams:

- Una aplicación con dos cámaras, la cual, mediante técnicas de reconocimiento facial, una correcta calibración y triangulación sea capaz de obtener resultados sobre la ubicación del usuario de forma similar que con la cámara 3D
- Otra aplicación que utiliza una sola cámara, y de forma similar a la anterior mediante técnicas de reconocimiento facial realice el seguimiento del usuario, pero esta vez para calcular las coordenadas de éste se emplearán diferente metodología, la cual se basará en el conocimiento de ciertos parámetros iniciales. Para obtener dichos parámetros se implementará un algoritmo que a partir de la interactuación con el usurario y en complemento con ciertas nociones acerca de proporciones anatómicas se calculen de forma precisa.

El objetivo del proyecto no es solo la implementación de las diferentes aplicaciones de tracking, sino además se compararán los resultados obtenidos y las ventajas y desventajas de cada modelo. Para ello se analizarán parámetros como la precisión del modelo, el tiempo de procesado entre frames o el precio.

Palabras clave

Tracking, clasificador, *stream*, cámara 3D, webcam, red neuronal, descriptores, *frames* por segundo, precisión, angular, coordenadas, tiempo de vuelo, resolución, visión artificial.

Resum

En el següent projecte es pretén implementar i analitzar diferents tècniques de seguiment d'un o diversos usuaris en un espai conegut retornant, d'aquesta manera, dades 3D sobre la ubicació i desplaçament d'aquests. Al seu torn, es desenvoluparà un mètode de comunicació a través de la xarxa, entre aplicacions i dispositius per a fer ús d'aquestes dades de desplaçament.

Amb la finalitat de comparar diferents metodologies, s'implementaran tres aplicacions de tracking fent ús de diferent maquinari i tècniques de visió artificial. D'una banda, es farà ús d'un equip semi professional que constarà d'una càmera 3D, la qual compta amb diferents sensors que li permeten construir un mapatge de profunditat i d'aquesta manera obtindre les coordenades dels usuaris de la sala de manera precisa.

D'altra banda, s'emprarà un model molt més quotidià que emprarà webcams ordinàries per a realitzar el seguiment de l'usuari. D'aquesta manera, s'implementaran dues aplicacions fent ús de webcams:

- Una aplicació amb dues cambres, la qual, mitjançant tècniques de reconeixement facial, un correcte calibratge i triangulació siga capaç d'obtindre resultats sobre la ubicació de l'usuari de manera similar que amb la càmera 3D
- Una altra aplicació que utilitza una sola cambra, i de manera similar a l'anterior mitjançant tècniques de reconeixement facial realitze el seguiment de l'usuari, però aquesta vegada per a calcular les coordenades d'aquest s'empraran diferent metodologia, la qual es basarà en el coneixement d'uns certs paràmetres inicials. Per a obtindre aquests paràmetres s'implementarà un algorisme que a partir de la interactuació amb l'usurari i en complement amb unes certes nocions sobre proporcions anatòmiques es calculen de manera precisa.

L'objectiu del projecte no és només la implementació de les diferents aplicacions de tracking, sinó a més es compararan els resultats obtinguts i els avantatges i desavantatges de cada model. Per a això s'analitzaran paràmetres com la precisió del model, el temps de processament entre frames o el preu.

Paraules clau

Tracking, classificador, stream, càmera 3D, webcam, xarxa neuronal, descriptors, frames per segon, precisió, angular, coordenades, temps de vol, resolució, visió artificial.

Abstract

The following project aims to implement and analyze different monitoring techniques for one or more users in a known space, thus returning 3D data about their location and movement. In turn, a communication method will be developed through the network, between applications and devices to make use of said movement data.

In order to compare different methodologies, three tracking applications will be implemented using different hardware and artificial vision techniques. On the one hand, a semi-professional equipment will be used that will consist of a 3D camera, which has different sensors that allow it to build a depth mapping and in this way obtain the coordinates of the users of the room in a precise way.

On the other hand, a much more everyday model will be used that will use ordinary webcams to track the user. In this way, two applications will be implemented using webcams:

- An application with two cameras, which, through facial recognition techniques, correct calibration and triangulation, is able to obtain results on the user's location in a similar way as with the 3D camera.
- Another application that uses a single camera, and in a similar way to the previous one, through facial recognition techniques, tracks the user, but this time to calculate the coordinates of the user, different methodology will be used, which will be based on the knowledge of certain parameters initials. To obtain these parameters, an algorithm will be implemented that, based on the interaction with the user and in addition to certain notions about anatomical proportions, are calculated precisely.

The objective of the project is not only the implementation of the different tracking applications, but also the results obtained and the advantages and disadvantages of each model will be compared. For this, parameters such as the precision of the model, the processing time between frames or the price will be analyzed.

Key words

Tracking, classifier, stream, 3D camera, webcam, neural network, descriptors, frames per second, precision, angular, coordinates, time of flight, resolution, artificial vision.

INDICE

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

1 Introducción

El uso de la visión artificial se incorpora en un sinfín de aplicaciones debido a las ventajas que ofrece, sobre todo a la hora de automatizar procesos. Pero uno de los usos más útiles que nos ofrece el uso de estas técnicas, es el reconocimiento de objetos y el conocimiento a tiempo real del comportamiento de éstos mismos.

En la actualidad trabajo como técnico investigador en la UPV, concretamente en una de las ramas del Instituto de Telecomunicaciones y Aplicaciones Multimedia (iTEAM), donde nos encargamos de investigar y desarrollar diferentes soluciones relacionadas con el audio y las telecomunicaciones.

Unas de las limitaciones con las que se encontraban mis compañeros, era la falta de datos sobre la posición espacial de los usuarios, para de esta forma poder hacer sus aplicaciones dinámicas y no depender del conocimiento previo sobre una ubicación fija del usuario.

En el siguiente proyecto, se va a implementar y comparar diferentes aplicaciones de tracking, con el fin de conocer a tiempo real la ubicación de uno o varios usuarios en un lugar determinado mediante el uso de diferentes cámaras y técnicas de visión artificial. Por otro lado, se analizará cómo integrar lo anteriormente mencionado con otras aplicaciones, comunicando los datos de desplazamiento del usuario entre sí.

Uno de los principales problemas a la hora de implementar este tipo de aplicaciones, es la necesidad de conocer la distancia del usuario o usuarios a la cámara, por eso en los siguientes apartados abordaremos también las diferentes técnicas que se pueden emplear para conseguir ese fin. Ya sea utilizando una cámara con infrarrojos como una webcam.

2 Objetivos del proyecto

El objetivo principal de proyecto es implementar una aplicación de tracking, que tiene como fin realizar el seguimiento de diferentes usuarios en una sala, para de esta manera de volver sus coordenadas 3D en el mundo real. Con el fin de obtener de realizar un análisis en profundidad de las diferentes técnicas de tracking se implementarán tres aplicaciones haciendo uso de diferentes cámaras para su comparación. Para conseguir el objetivo principal primero habrá que ir cumpliendo los siguientes subobjetivos:

- Recopilación de datos acerca de las principales cámaras 3D del mercado, para de esta manera realizar una comparación de las prestaciones y precios de cada una.
- Implementación y comparación entre diferentes métodos para la detección de rostros en una imagen mediante técnicas de visión artificial.
- Implementación de tres aplicaciones de tracking, haciendo uso de diferentes cámaras:
    - Cámara 3D
    - Una sola webcam
    - Dos webcams
- Comunicación entre aplicaciones a través de la red usando el protocolo de comunicación UDP.
- Comparación entre las diferentes aplicaciones para comprobar las ventajas y desventajas de cada una de ellas.

Cabe decir que, para llevar a cabo todos los objetivos, ha sido necesaria una labor previa de investigación para determinar las técnicas, entorno de trabajo y lenguajes de programación para la correcta implementación de las diferentes aplicaciones.

3 Estado del arte

La problemática derivada de la implementación de sistema de seguimiento de usuarios ha sido investigada en múltiples ocasiones. Una de las maneras más eficientes de conseguirlo, es haciendo usa cámara 3D. Como afirma Wenjun Zeng [1] en su investigación acerca de las funcionalidades de Kinect, "Microsoft Kinect Sensor and Its Effect" (2019), los sensores que posee las cámaras 3D permite obtener información de profundidad realmente útil y precisa a la hora de calcular la posición de usuarios en una sala.

Existen diferentes cámaras 3D en el mercado las cuales ofrecen diferentes prestaciones, en el artículo "Comparison of the performance of 3D camera systems" escrito por Y. Ehara [2] en 2019 se realiza una comparación exhaustiva de las principales cámaras y sistemas de visón 3D del mercado, analizando las prestaciones que ofrecen cada una de ellas. Después de analizar el paper anterior, se puede concluir en que Kinect y Orbbec Astra han sido las principales cámaras 3D del mercado. Debido a este hecho se decidió profundizar más en la utilización de estos dispositivos. Como cabía esperar, se puede encontrar mucha documentación referente a éstos dos dispositivos, por ejemplo en los articulos "Effects of camera viewing angles on tracking kinematic gait patterns using Azure Kinect, Kinect v2 and Orbbec Astra Pro v2" de Ling-Fung Yeung [3] y "Interchangeability of Kinect and Orbbec Sensors for Gesture Recognition" por Alina Delia Calin [4], se llevan a cabo una disertación acerca de los algoritmos de detección empleados en ambas cámaras y diferentes prestaciones de cada una de ellas muy interesante. Otra de las cámaras que la actualidad está empezando a competir junto con Kinect y Orbbec con las InterRealsens D400 Series, debido a que disponen de unas prestaciones óptimas y un precio competente, en el artículo de Silvio Giancoila "Metrological Qualification of the Intel D400™ Active Stereoscopy Cameras" [5] se realiza un análisis de esta serie donde se analizan sus ventajas y desventajas.

Por otro lado, existen diferentes técnicas que permiten obtener resultados óptimos haciendo uso de un hardware mucho más rudimentario. En el simposio de Múnich de 2008 acerca de patrones de reconocimiento para la detección de formas humanas, se habla del uso de clasificadores en cascada para la clasificación de descriptores basados en cambios de intensidad y en el gradiente, como son Haar o HOG. De esta manera se utilizan algoritmos como el de Viola-Jones, debido el bajo coste computacional que su pone su implementación.

En la actualidad se ha puesto de moda el uso de técnicas basadas Deep Learning debido a su buen funcionamiento y baja tasa de errores. En el paper de "Joint Face detection and Facial Expression Recognition with MTCNN" por Xiang y Zhu [6] hablan del excelente comportamiento de la red MTCNN, que utiliza una arquitectura en cascada para detección facial. En cuanto, al reconocimiento de cuerpo completo uno de los algoritmos más estandarizadas es YOLO, debido a que es de codigo abierto y presenta muy pocos errores de detección. Además, según el reciente paper, publicado por Kamel Boudjit [8], "Human detection based on deep learning YOLO-v2 for real-time UAV applications", El algoritmo base corre a 45 frames por segundo (FPS) sin procesamiento de lote haciendo uso de una GPU Titan X.

A la hora de calcular la posición en coordenadas 3D del objeto a seguir, existen diferentes técnicas basadas en triangulación para obtener la distancia del objeto a la cámara utilizando dos o más cámaras. En el paper de 2020 "Development of a Camera Localization System for Three-Dimensional Digital Image Correlation Camera Triangulation" por Alessandro Sabato [8] se estudia cómo obtener

los datos de desplazamiento en coordenadas reales, haciendo uso de dos cámaras, triangulación lineal y conociendo la separación entre las cámaras. Por otro lado, para calcular dichas coordenadas haciendo uso de una sola cámara hay mucha menos información y los proyectos que se pueden encontrar utilizan objeto con un tamaño conocido como referencia.

Para implementar el cálculo de coordenadas con el uso de una sola cámara, se ha necesitado del conocimiento previo de los datos de ubicación inicial y tamaño del objeto a seguir. Para la determinación de dichos datos iniciales se han empleado técnicas basadas en estudios estadísticos sobre anatomía humana. Algunos de estos estudios han sido "Human body proportions explained on the basis of biomechanical principles" de H. Witte [7], donde se analizan las proporciones humanas basándose en tesis como la de Le Corbusier o el mismísimo Leonardo DaVinci. Otro lugar de consulta que ha sido de los más útil es la base de datos de BIOSIS de donde se pueden extraer datos estadísticos acerca del tamaño y proporción de las diferentes partes del cuerpo humano en la sociedad, dividendo estos datos en función del género y edad de la población.

Una vez recopilada toda esta información, se combinarán diferentes técnicas vistas para implementar las diferentes aplicaciones de tracking, sacando el máximo rendimiento posible al hardware disponible. De esta manera, se compara el rendimiento de diferentes métodos y aplicaciones para sacar conclusiones acerca del rendimiento de cada uno de ellos en función de la aplicación que se le vaya a dar.

4 Fases del proyecto

En este punto se resumirán las diferentes fases que se han seguido para llevar a cabo el desarrollo del proyecto y por tanto cumplimiento de los objetivos mencionados en el punto anterior.

Fase 1. La fase inicial consiste en la búsqueda de información acerca de las diferentes opciones para implementar las aplicaciones de tracking para así discernir cuales son aparentemente las óptimas.

Una vez descubierto el uso de las cámaras 3D y su aplicación en las aplicaciones de tracking, se buscaron nuevas técnicas para conseguir implantar la aplicación de seguimiento haciendo uso de un hardware menos dedicado y más cotidiano. En esta fase también se incluiría la recopilación de las cámaras y la información necesaria para hacerlas funcionar de manera adecuada.

Fase 2. Para empezar con el desarrollo de la aplicación haciendo uso de la cámara 3D, primero fue necesario instalar el entorno necesario para aprovechar al máximo las herramientas de desarrollo proporcionada por *Orbbec*. Entre estas tareas se encuentra la de selección de un IDE adecuado, donde Visual Studio fue el elegido por su versatilidad y fácil manejabilidad.

Una vez configurado el entorno, se realizaron diferentes pruebas para de esta manera aprender a adquirir imágenes a través de la cámara y conectarse a los diferentes *streams* que esta ofrece. También se dedicó tiempo a la familiarización con el SDK para aprovechar el máximo las diferentes clases y funciones de las que está compuesta.

Fase 3. Con todo preparado ya fue mucho más sencillo empezar con el desarrollo de la aplicación de tracking haciendo uso de una cámara 3D. La primera parte del desarrollo consistió en la adquisición de imagen de profundidad y detección de cuerpos, para a continuación, realizar el conteo y obtención de datos de desplazamiento y rotación de éstos.

A lo largo del desarrollo fueron saliendo diferentes problemas que solventar y funcionalidades que pulir para finalmente, obtener los datos de forma correcta y a continuación transmitirlos a través de la red. Para concluir con esta fase, se realizaron diferentes test de precisión y de tiempo de cómputo para comprobar el rendimiento de la aplicación

Fase 4. La siguiente aplicación por implementar es la que utiliza como hardware dos webcams. A pesar de que la estructura de la aplicación estaba clara gracias a la anterior fase de investigación, fue necesario llevar a cabo un proceso de implementación y comparación de diferentes técnicas de visión artificial para la detección de rostros.

Una vez implementadas las diferentes técnicas, se aplica un test de rendimiento para comprobar la precisión y tiempo de cómputo de cada una de ella.

Fase 5. Para la implementación de la aplicación de tracking haciendo uso de dos cámaras, se pasaron por diferentes fases que incluyen calibración, detección facial, triangulación y obtención de datos de desplazamiento. Finalmente se implementó la ya comentada comunicación entre aplicaciones a través de la red y haciendo uso de UDP.

De forma similar a la aplicación anterior, tras la implementación se realzo un test de rendimiento para de esta manera poder sacar conclusiones.

Fase 6. La última aplicación para implementar es igual que la anterior pero esta vez haciendo uso de una sola cámara. Tras una investigación para obtener la técnica óptima se inicia la implementación de la aplicación, hasta llegar a un punto bloqueante donde es necesario obtener ciertos parámetros previos al funcionamiento de esta.

Fase 7. Esta fase consiste es solucionar el problema bloqueante surgido en la fase anterior, el cual consiste en la obtención de ciertos parámetros iniciales. Tras una reflexión e investigación acerca de las proporciones de la anatomía humana, se llega a una conclusión que se implementa y pule hasta llegar a su versión óptima.

Tras resolver el error bloqueante, se pudo seguir con la implementación para finalmente pasar un test de rendimiento similar al de resto de aplicaciones.

Fase 8. Ya se han implementado y testeado todas las aplicaciones. Esta fase consistirá en el análisis y comparativa de los resultados obtenidos para de esta manera discernir cual es la mejor de las tres aplicaciones, o por lo menos cual es más recomendable en función de los requerimientos.

Fase 9. Por último, se llevó a cabo la redacción y maquetación de esta memoria, para plasmar los procedimientos y resultados obtenidos.

5 Aplicación de tracking mediante el uso de cámara 3D.

El desarrollo de sensores que permitan representar fielmente la percepción humana ha fascinado al ser humano a lo largo de la historia. De hecho, los primeros acercamientos para conseguir imágenes tridimensionales nacen casi con la fotografía misma de la mano Sir Charles Wheatstone, que en 1838 publicó los primeros artículos acerca de la imagen estereoscópica.

Los avances en la captación de imágenes han fomentado el desarrollo tecnológico y humano gracias a su aplicación en diferentes campos como la medicina, ingeniería molecular, industria o meramente para eventos relacionados con el ocio audiovisual. Este hecho ha desembocado en la aparición de cámaras mucho más modernas, las cuales mediante la incorporación de diferentes tipos de sensores y en combinación con las técnicas de visión artificial permiten obtener información de lo más útil para llevar a cabo un sinfín de tareas.

5.1 Comparación entre diferentes cámaras 3D del mercado

5.1.1 Kinect

Una de las empresas que más han invertido en el desarrollo de cámaras en las últimas décadas es el gigante Microsoft, que en 2011 lanza la conocida cámara Kinect (originalmente conocido por el nombre en clave «Project Natal») enfocada al mundo de los videojuegos y con la intención de sumir a sus usuarios en una experiencia mucho más inmersiva pero que finalmente se ha utilizado en muchos otros proyectos relacionados con la visión artificial.

Ilustración 1. Modelos de cámaras Kinect.

El dispositivo cuenta con una cámara RGB, un sensor de profundidad, un micrófono de múltiples matrices y un procesador personalizado que ejecuta el software patentado, que proporciona captura de movimiento de todo el cuerpo en 3D, reconocimiento facial y capacidades de reconocimiento de voz. El sensor de profundidad es un proyector de infrarrojos combinado con un sensor CMOS monocromo que permite a Kinect ver la habitación en 3D en cualquier condición de luz ambiental. Las cámaras que utilizan el sensor de infrarrojos de esta manera, se denomina cámaras de tiempo de vuelo y estiman las distancias a los objetos calculando el tiempo transcurrido entre la emisión y la recepción de un haz de luz infrarrojo. Cabe decir que, en la actualidad, el proyecto Kinect se ha visto parado, dado principalmente al fracaso de ventas de la Kinect v2 que salió junto a la consola “Xbox One”, de forma que Microsoft está centrando su esfuerzo a la incorporación de realidad virtual en su nueva familia de consolas.

A pesar del fracaso en ventas de la cámara Kinect, este tipo de cámaras pueden explotar su funcionalidad y aportar valor en tareas para muchos campos, entre los que encontraríamos:

- **Medicina**. Las cámaras 3D cada vez tienen cada vez más cabida en el campo de la salud, ya sea como herramienta quirúrgica, como soporte para la monitorización en la rehabilitación de un paciente y demás funcionalidades. Como, por ejemplo TedCas de la empresa Tedesys, una aplicación que, aprovechando el sensor de movimiento de Kinect, permite realizar diferentes tareas en el quirófano.
- **Gestión de ventas**. Las técnicas de visión artificial son usadas por muchas empresas para estudios de mercado gracias a ciertas aplicaciones como: reconocimiento facial, conteo de personas, monitoreo de inventario, escaparate interactivo y otros casos de uso de análisis de mercado minoristas.
- **Automatización.** Gracias al uso de cámaras e inteligencia artificial se transforma la forma en que las empresas agrícolas e industriales automatizan ciertos procesos, consiguiendo una mayor eficiencia y abaratar los costes en labores como la inspección, medición, recolección o pesaje nocturno con cámaras 3D.
- **Robótica**. La visión de robot en tiempo real es un aspecto clave en robótica que permite al robot llevar a cabo mapeos, localización, evasión de objetos e interacción en interiores.
- **Escaneo 3D**. Recopilación de datos de profundidad de objetos y espacios del mundo real para reconstruir modelos digitales tridimensionales.
- **Domótica**. Cree un sistema interconectado que brinde seguridad avanzada, conveniencia de vanguardia y mayor eficiencia para el hogar u oficina.

A parte de Kinect existen otras cámaras similares en el mercado con diferentes especificaciones y precios, que en función de nuestros intereses se adaptará mejor o peor a la tarea en cuestión. A continuación, se pasará a analizar y comparar los diferentes aspectos de las principales cámaras 3D del mercado.

En el caso de Kinect existen dos versiones de la misma Kinect v1 y Kinect v2 siendo la segunda una versión mucho más refinada y con ciertas diferencias con respecto a las características. En la siguiente tabla se resumen algunos de los aspectos principales de ambas versiones.

Tabla 1. Prestaciones de los diferentes modelos de Kinect

	Kinect V1	**Kinect V2**
Cámara RGB	640 x 480 @30 fps	1920x 1080@30 fps
Cámara de profundidad	320 x 240	512 x 424
Dist. Profundidad máx	4.5 m	4.5 m
Dist. Profundidad min	40 cm	50 cm
Campo visión horiz.	57°	60°
Campo visión verti.	43°	60°
Motor de inclinación	Sí	No
Articul. esqueleto	20	26
Nº máx esqueletos	2	6
USB Standart	2.0	3.0
Sistema operativo	Win 7, Win 8	Win 8

Para empezar, podemos apreciar que la segunda versión ofrece una mejor resolución y un mayor ángulo de visión tanto en vertical como en horizontal. La cámara Kinect ofrece varios tipos de vistas, la vista de profundidad generada haciendo uso del sensor infrarrojos y la vista de esqueletos donde se detectan los cuerpos en la imagen y se genera un esqueleto para cada uno siguiendo las articulaciones que lo conforman. En el caso de la vista de profundidad "Kinect v2" ofrece una mayor profundidad y en la vista de esqueletos permite detectar un mayor número de cuerpos y de articulaciones. En cuanto a las distancias máximas y mínimas de funcionamiento la primera versión muestra la única ventaja frente a la segunda, la distancia mínima es menor lo que para algunas aplicaciones puede ser bastante útil. Otra diferencia importante es que la segunda versión incluye un motor de inclinación que podría usarse para inclinar la cámara y aumenta el posible espacio de interacción de la cámara en +27 y -27 grados.

Ilustración 2. Ejemplo de vista de profundidad

Ilustración 3. Ejemplo de vista de esqueletos

En general la "Kinect v2" es una versión más refinada que ofrece, mejores especificaciones. En cuanto al precio, a pesar de que ya no se fabrican ninguna de las dos versiones es fácil y relativamente barato encontrarlas tanto de primera como segunda mano. Los precios más baratos que se puede encontrar rondan entre los 50 € para la v1 y 100€ para la v2.

5.1.2 Orbbec Astra

Existen empresas que han comercializado cámaras 3D similares a la Kinect de Microsoft, entre ellas se encuentra Orbbec, empresa fundada en 2013 y que ofrece cámaras y soluciones tecnológicas basada en visión 3D. Dentro de su catálogo de cámaras se encuentra una gran variedad enfocada a industria, pero dentro de la sección de cámaras para usuario encontramos la Orbbec Astra, que de forma similar a Kinect habilita varias funciones como reconocimiento facial, reconocimiento de gestos, seguimiento del cuerpo humano, medición tridimensional, percepción del entorno o reconstrucción de mapas tridimensionales. Cabe decir que en la actualidad Astra ha participado en el desarrollo de las cámaras 3D para telefonía que incluirán los nuevos móviles de OPPO. Además, la empresa ha lanzado recientemente Obbec Astra Persee, un nuevo modelo que además de las funcionalidades de la anterior Orbbec Astra, incorpora un procesador que la convierte en la primera computadora con cámara 3D del mercado. Nosotros nos centraremos en las diferentes versiones de la cámara 3D *Orbbec* Astra.

Ilustración 4. Cámara Orbbec Astra.

Dentro del modelo Orbbec Astra, encontramos varias versiones muy similares entre sí, pero que se diferencia ligeramente en las especificaciones que ofrecen, en la siguiente tabla veremos una breve comparativa entre los modelos Orbbec Astra, Orbbec Astra S y Orbbec Astra Pro.

Tabla 2. características de las cámaras de Orbbec.

	Orbbec Astra	**Orbbec Astra S**	**Orbbec Astra Pro**
Dimensiones	160 x 30 x 40 (mm)	160 x 30 x 40 (mm)	160 x 30 x 40 (mm)
Peso	300 g	300 g	300 g
Rango	0.4 a 8 m	0.4 a 2 m	0.4 a 8 m
Cámara profundidad	640*480 (VGA) 16bit @30 FPS	SXGA(1280 x 1024 @ 5 FPS Windows Only) VGA(640 x 480 @ 30 FPS) QVGA(320 x 240 @ 30 FPS)	1280 x 720 @30FPS
Cámara RGB	1280*960 @10FPS	HD(1280 x 960 @ 7 FPS) VGA(640 x 480 @ 30 FPS) QVGA(320 x 240 @ 30 FPS)	HD(1280 x 960 @ 7 FPS) VGA(640 x 480 @ 30 FPS) QVGA(320 x 240 @ 30 FPS)
Campo visión	60° horiz. x 49.5 ° vert. (73º diagonal)	60° horiz. x 49.5 ° vert. (73º diagonal)	60° horiz. x 49.5 ° vert. (73º diagonal)
Micrófonos	2	2	2
Sistema Operativo	Windows, Linux, Android	Windows, Linux, Android	Windows, Linux, Android
Temperatura	0 - 40°	0 - 40°	0 - 40°

Como se puede observar en la tabla, las principales diferencias entre los diferentes modelos residen en la resolución que pueden ofrecer las cámaras de profundidad y RGB. En el modelo Orbbec Astra S, a pesar de tener menor rango de funcionamiento, se pueden variar el ratio y resolución en ambas cámaras, pudiendo de esta manera elegir la relación resolución/fps en función de las necesidades del proyecto. A pesar de las leves diferencias entre los modelos el precio es el mismo para los tres y este rondará los 150€. Esto no incluye algunas funcionalidades como la detección de cuerpos, que requerirán del pago de una licencia aparte.

En comparación con Kinect vemos que mejora en los principales aspectos: resolución, campo de visión y rango de funcionamiento. Otra diferencia destacable es que el único SO soportado por Kinect será Windows mientras que Astra soporta los tres principales sistemas operativos, lo que la convierte en una cámara más cómoda a la hora de trabajar. Cabe decir que Astra ofrece un software propio que incluirá una serie de librerías y funcionalidades para el desarrollo de aplicaciones y a su vez es compatible con el framework gratuito OpennNi, que permite la gestión de los recursos de la cámara 3D, detección de movimiento y reconocimiento de formas. En un principio OpennNi fue desarrollado en una asociación con Asus para desarrollar un dispositivo compatible para PC similar a la Kinect, pero en la actualidad se ha generalizado entre la mayoría de los proyectos de código abierto.

5.1.3 Intel RealSense D400 Series

Una de las últimas compañías en lanzar su propia cámara 3D ha sido Intel con las RealSense D400, tres modelos diferentes que ofrecen unas prestaciones realmente interesantes. Los modelos cuentan con sensores similares a los casos previamente analizados, pero a su vez presenta unas prestaciones diferentes que analizaremos a continuación.

D415 D435 D455

Ilustración 5. Cámaras Intel RealSense D400 Series

Dentro de las D400 Series encontramos tres modelos principales, los cuales de forma similar a Astra se diferencian entre sí en la resolución y angular de sus cámaras y rango de funcionamiento. Las cámaras están alimentadas por USB y constan de un par de sensores de profundidad, un sensor RGB y un proyector de infrarrojos. La cámara de profundidad cuenta con un obturador de imagen global y un amplio campo de visión, de esta manera ofrece una percepción de profundidad precisa cuando el objeto se está moviendo o el dispositivo está en movimiento, cubriendo más área y minimizando los puntos ciegos. En la siguiente tabla aparecen reflejadas las especificaciones de los modelos D415, D435 y D455.

Tabla 3. Características de las Intel RealSense D400.

	D415	**D435**	**D455**
Dimensiones	100 x 47 x 38 mm	100 x 47 x 38 mm	132 x 47 x 41 mm
Cámara profundidad	1280 x 720 px	1280 x 720 px	1280 x 720 px
Cámara RGB	1920 x 1080 px	1920 x 1080	1280 x 800 px
Angular RGB	65° x 40°	87 ° x 58°	87° x 58°
Angular Profundidad	69° x 42°	69 ° x 42°	90° x 65°
Rango Func.	0.2 a 2 m	0.2 a 2 m	0.2 a 2 m
Velocidad streaming	30 fps	30 fps	30 fps
Sistema Operativo	Win,Lin,Android	Win,Lin,Android	Win,Lin,Android

Lo primero que se aprecia es que son cámaras de corto rango de funcionamiento lo que podría ser un inconveniente dependiendo de la aplicación que se le quiera dar. Las tres cuentan con una resolución muy parecida en las cámaras de profundidad y RGB, llegando a 1920 x 1080 px a una velocidad de streaming de 30 fps. En el modelo D455 cuenta con un gran campo de visión con respecto al resto de cámaras vistas previamente, llegando a los 90° horizontal y 65° en vertical. De la misma manera que

las cámaras de Orbbec, éstas son compatibles con los principales sistemas operativos y cuenta con su própio SDK para el desarrollo de aplicaciones, el Intel® RealSense ™ SDK 2.0. Los precios de las cámaras son 160€ para la D415 siendo la más barata, 180€ para la D435 y 200€ para la D455.

En comparación con los otros modelos, para empezar el precio es ligeramente más elevado debido entre otras cosas a la capacidad de resolución sin sacrificar los 30 fps en el *stream*. Además, las cámaras tienen un campo de visión más grande debido a lentes de gran angular y un sistema de obturador global para todas las cámaras. Ambas características tienen beneficios para los casos de uso de seguimiento de objetos o personas que tienen que cubrir un área grande o que se mueven rápidamente.

En conclusión, la cámara óptima dependerá de las necesidades del proyecto en el que estemos trabajando. Es cierto que las cámaras de Intel ofrecen una mejor resolución y velocidad de *streaming*, pero esto es a costa de un menor rango de funcionamiento y un precio más elevado. Las Kinect se pueden conseguir a un precio bastante asequible, pero a su vez se han quedado obsoletas en algunos aspectos con respecto a los modelos más modernos. Lo importante aprovechar al máximo los diferentes aspectos de la cámara que seleccionemos sin gastar más de lo que necesitemos.

5.2 Desarrollo de la aplicación

Recordando el objetivo del trabajo, la intención es llevar a cabo el seguimiento de diferentes usuarios en una sala para de esta forma pasar los datos de desplazamiento a otra aplicación a través de la red, vía UDP. Con el fin de cumplir el objetivo con una cámara 3D, se ha utilizado el modelo Orbbec Astra de Orbbec, ya que es con lo que disponía en el laboratorio y además su amplio campo de visión y rango de funcionamiento serán muy útiles teniendo en cuenta las dimensiones de la sala donde se realizará en el seguimiento. En el siguiente esquema se representa el objetivo de una forma más gráfica e intuitiva.

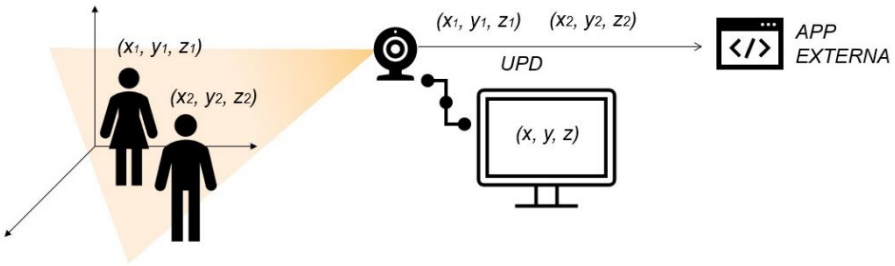

Ilustración 6. Esquema instalación

Hay que tener en cuenta que la aplicación de tracking deberá pasar los datos de desplazamiento en tiempo real, por lo que una de las prioridades será que el tiempo de procesado de la aplicación sea el mínimo posible para facilitar la coordinación entre la adquisición y transmisión de datos con la aplicación externa. Las herramientas que se han necesitado para el desarrollo y puesta en marcha son los siguientes:

- Ordenador portátil Asus:
  - Procesador: AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz
  - Ram 16GB
  - Windows 10
  - Gráfica Nvidea Geforce RTX 2060
- Cámara 3D Orbbec Astra
- Otro ordenador con la aplicación externa, conectado a la misma red que el ordenador principal.

5.2.1 Instalación del entorno de trabajo

Para llevar a cabo el desarrollo de la aplicación de tracking, se empleará el SDK patentado que nos ofrece Astra, el cuál veremos posteriormente como se estructura y como utilizarlo. Pero antes de entrar más a fondo en el desarrollo de la aplicación veremos una breve explicación sobre el entorno de trabajo utilizado, las herramientas y el software necesario.

Como ya se ha comentado anteriormente el software de Astra es compatible tanto con Windows, Android, Linux e incluso puede ser utilizado junto a Unity para labores relacionadas con los videojuegos o el desarrollo de animación 3D. A continuación, se analizarán los requerimientos mínimos para cada uno de los sistemas operativos.

- Windows
  - Windows 7,8,10 de 32 o 64 bits.
  - x86-based processor @ 1.8+ ghz
  - USB 2.0
  - 4 Gb de RAM
- Linux
  - Ubuntu (20.04, 18.04, 16.04, 14.04), x86_64, arm, aarch64
  - amd64-based processor @ 1.8+ ghz
  - USB 2.0
  - 1 Gb de RAM
- Android
  - Android OS 4.4.2 (KitKat) +
  - ARMv7a/ARM64v8a processor @ 1.5 ghz+
  - USB 2.0 host support (OTG-capable)
  - 512 Mb de RAM
- Unity
  - Windows 7, 8 and 10, 32-bit y 64-bit
  - Android 4.4.2 o superior

En este caso veremos como instalar los requerimientos necesarios en Windows 10, ya que es el sistema operativo utilizado para este trabajo. Lo primero necesario para el tener el entorno de trabajo correctamente configurado, será instalar los drivers de la cámara, los cuales encontraremos fácilmente en la página oficial de Astra (https://orbbec3d.com/develop) y donde tendremos que elegir la versión adecuada para el sistema operativo que utilicemos.

Una vez ejecutado el instalador y finalizado con éxito la instalación, accederemos a “Administrador de dispositivos donde nos aparecerá ahora la pestaña “orbbec” indicando que la instalación de los drivers ha finalizado exitosamente.

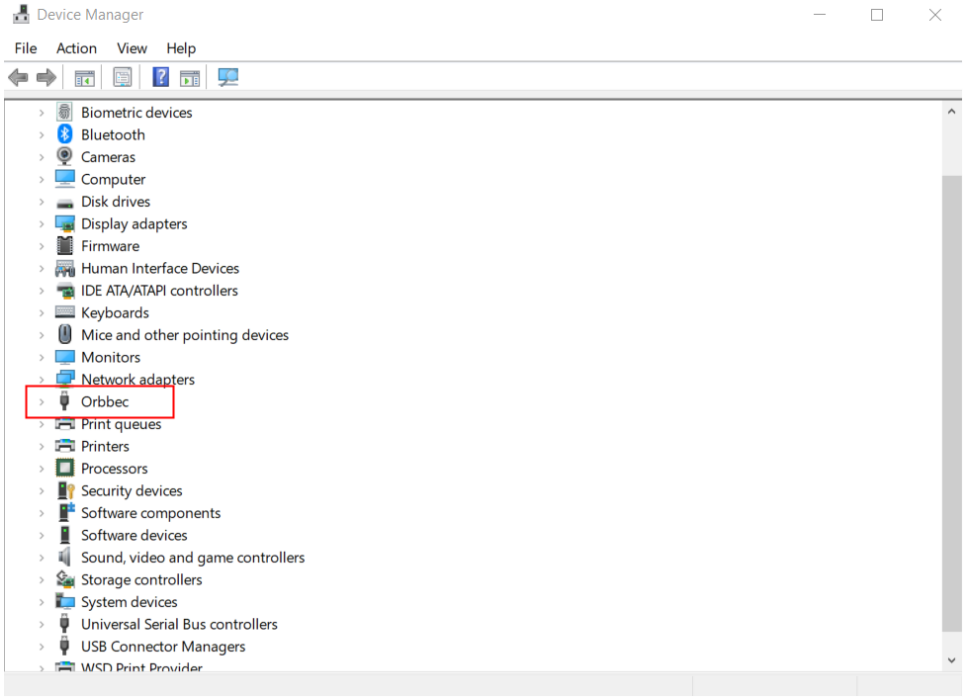

Ilustración 7. Comprobación de drivers instalados.

A continuación, será necesario elegir un IDE para escribir, ejecutar y depurar nuestro código cómodamente. Los propios desarrolladores de Astra recomiendan encarecidamente utilizar Visual Studio por lo que en los siguientes pasos veremos como configurarlo correctamente para empezar a trabajar.

Teniendo ya instalada alguna de versión de Visual Studio, recomendablemente Visual Studio 2015 o superior, creamos un nuevo proyecto y pasamos directamente a configurar las dependencias para poder hacer uso del SDK de Astra, el cual descargaremos desde la página de Astra seleccionando la versión correcta para el sistema operativo utilizado.

1. Lo primero que haremos será crea una nueva variable de entorno llamada $ASTRA_HOME donde se encontrará la ruta al SDK de Astra previamente descargado.
2. Tras comprobar que están las herramientas de compilación para C/C++ correctamente instaladas, pasamos a acceder a las propiedades del proyecto y desplegamos la pestaña C/C++ donde en la opción “General” podremos modificar la ruta de los directorios adicionales. En esta opción añadiremos la siguiente ruta "$(ASTRA_HOME)\include".

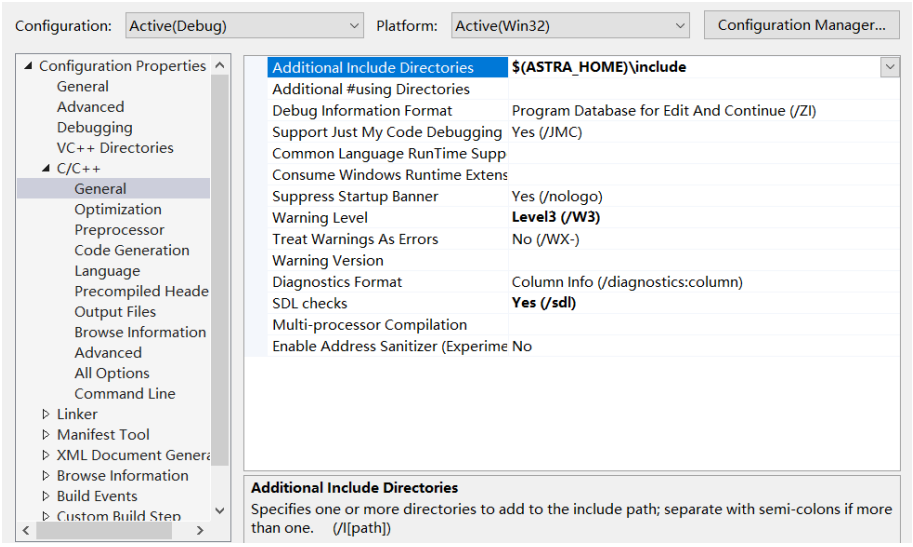

Ilustración 8. Añadir dll al proyecto.

3. Lo siguiente que haremos será desplegar la pestaña del “Vinculador”, acceder a “General” y de la misma manera que antes añadiremos dos rutas a la opción de librerías adicionales, las cuales serán "$(ASTRA_HOME)\lib" y "$(ASTRA_HOME)\lib\Plugins".

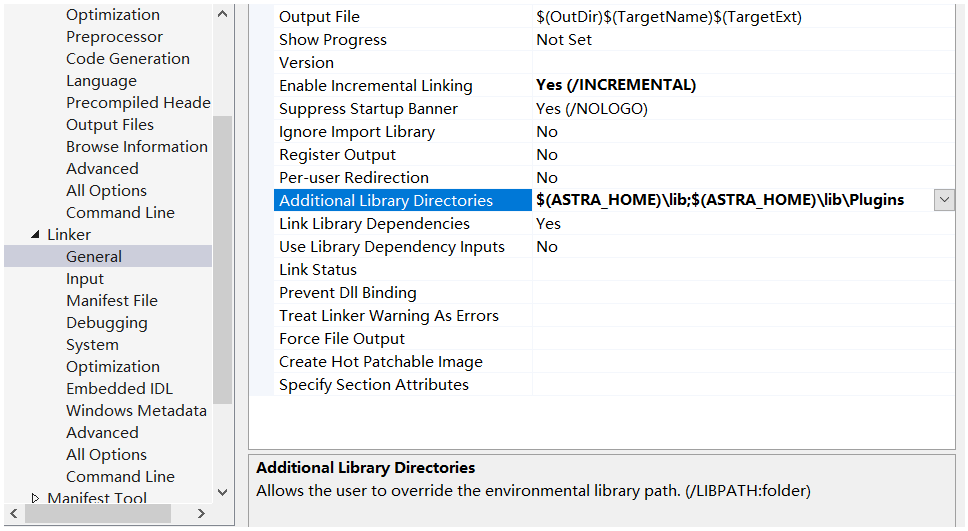

Ilustración 9. Añadir librerías al proyecto.

4. Por último y también en la sección “Vinculador” accederemos a la opción “Entrada” para modificar las entradas adicionales añadiendo las siguientes librerías: astra.lib, astra_core.lib,astra_core_api.lib.

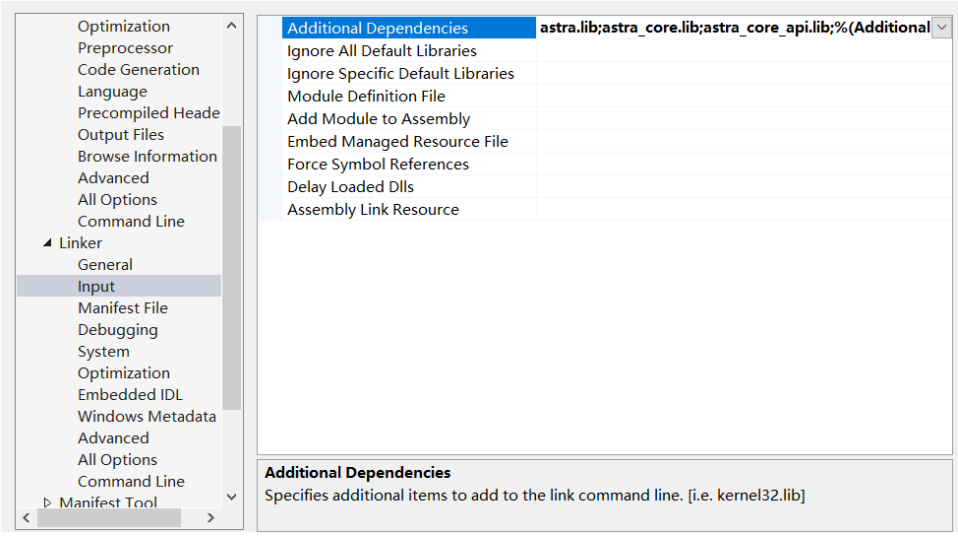

Ilustración 10. Añadir dependencias al proyecto.

5. Para comprobar que la configuración del proyecto se ha configurado correctamente, podemos descargar algunos de los ejemplos que encontraremos en la página de Astra como por ejemplo SimpleStreamViewer-SFML. Si todo ha ido correctamente podremos visualizar algo parecido a la siguiente imagen.

Ilustración 11. Ejemplo de visión de profundidad.

5.2.2 Definición de términos básicos

Como se ha mencionado antes, para llevar a cabo el desarrollo de la aplicación se procederá a utilizar el SDK que ofrece Astra, aunque también cabe la posibilidad de usar el SDK gratuito OpenNi para el cual existen una gran cantidad de proyectos de código abierto.

Antes de pasar a analizar el código de la aplicación, veremos algunos términos básicos necesarios para comprender el funcionamiento del SDK de Astra y para aprovechar al máximo los recursos disponibles.

- **Stream.** Un stream hace referencia al flujo de datos o tramas correspondiente a una fuente de datos en particular. En nuestro caso los datos proceden de los diferentes sensores disponibles en la cámara. Un ejemplo similar podría ser una película, donde ésta es el stream completo y cada fotograma sería una trama del stream. Los diferentes sensores de la cámara dan lugar a diferentes streams, como serían el de color o el de profundidad, pero a su vez estos streams pueden conformar otros sub streams a su vez como el de mano, cabeza o cuerpo.
- **Stream set.** Un stream set es un conjunto de streams relacionados entre sí. Siguiendo con el ejemplo de la película, el audio y la imagen serían dos streams relacionados que combinados entre ellos formarían un stream set. Si bien un stream set podría estar compuesto en su totalidad por streams provenientes de un sensor físico, también puede contener streams creados a través de las funcionalidades del SDK. Como ejemplo, la cámara Astraes capaz de generar un stream set que contiene un stream de profundidad, un stream de color, así como streams generados haciendo uso del SDK como los stream de mano, cabeza o cuerpo. Los stream sets están referenciados mediante un identificador único (URI) basado en cadenas que permite a los desarrolladores acceder a varios stream sets distintos a la vez.
- **Stream reader.** El stream reader será el encargado de obtener los fotogramas correspondientes a un stream set. Además, se sincroniza los diferentes streams de un stream set en la recepción como podría ser el audio y la imagen.

5.2.3 Análisis de código

Dentro de los lenguajes que se pueden utilizar para programar aplicaciones usando el SDK de Astra se encuentras: C, C++ y Java. En este caso se utilizará C++ por familiaridad con el lenguaje y por la capacidad de generar clases y objetos a diferencia de C. Como se ha comentado previamente la herramienta para programar, depurar y ejecutar el código será Visual Studio.

Para empezar, utilizare un *stream* set conformado por el *stream* de profundidad y de cuerpos el cual está conformado a partir de los datos de profundidad y haciendo uso de una red neuronal 3D entrenada con la intención de relacionar puntos clave dentro del espacio 3D, de forma que se obtiene información no solo del cuerpo en general sino además de 19 *joints* como cabeza o manos.

Ilustración 12. Articulaciones de esqueleto Astra.

Además, haciendo uso del *framework Nuitrack* se puede obtener una vista en la que se representa la relación entre los puntos detectados conformando un esqueleto.

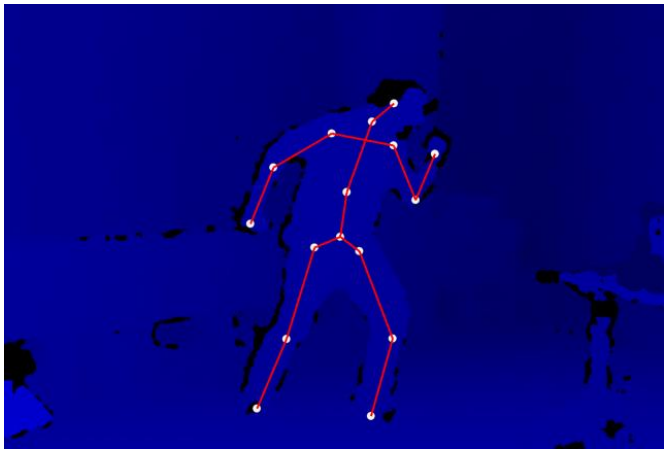

Ilustración 13. Ejemplo de visión de esqueleto.

Como se ha mencionado anteriormente para hacer uso del *body stream* es necesario pagar una licencia a parte de unos 70 $, de forma que una vez paguemos y nos envíen el código de activación deberemos introducirlo en el código para hacer uso de este *stream*. Por lo que el primer paso será inicializar el SDK de Astra e introducir el código de activación de la licencia. Para finalizar limpiamente el SDK será necesario hacer uso de la función *astra::terminate()* al final del script.

```
set_key_handler();
astra::initialize();
const char* licenseString = "código de activación";
orbbec_body_tracking_set_license(licenseString);
```

Ahora que hemos visto como inicializar Astra y como introducir la licencia es momento de conectarse al sensor de la cámara. Para ello, usamos la clase *StreamSet*, la cual eencapsular un grupo de fuentes de datos relacionadas entre sí y hace de portal con el sensor de la cámara.

```
// Se instancia un stream set
astra::StreamSet streamSet;
```

Es hora de hacer funcionar el objeto *StreamSet* para empezar a obtener información procedente de los sensores de la cámara. Para ello será necesario leer de los flujos de datos provenientes de los diferentes *streams* que nos interesen, de forma que los datos vendrán empaquetados en lo que llamaremos "frames". Para acceder a la transmisión de cada *stream* será necesario crear un objeto de tipo *StreamReader* que será el encargado de ir leyendo los *frames* adquiridos por la cámara.

```
// Se crea un StreamReader asociado al anterior StreamSet
astra::StreamReader reader = streamSet.create_reader();
```

En este caso los *streams* interesantes son el de profundidad y el de cuerpo, por lo que a través del *reader* arrancaremos ambos.

```
//´Se arrancan los streams de profuncidad y cuerpo
reader.stream<astra::DepthStream>().start();
reader.stream<astra::BodyStream>().start();
```

Con ambos *streams* arrancados, toca empezar a obtener *frames* de ambos flujos para eso utilizaremos el *reader* y la función *get_latest_frame().*

```
// se obtienen el último fotograma de amos streams
astra::Frame frame = reader.get_latest_frame();
const auto depthFrame = frame.get<astra::DepthFrame>();
const auto bodyFrame = frame.get<astra::BodyFrame>();
```

El *stream* de profundidad solo será necesario para obtener la vista de profundidad, por lo que nos centraremos el flujo de cuerpos del cual se tendrá que obtener la información de posición de los usuarios que aparecen en la imagen. Para ello se utilizará la función *astra_bodyframes_body_list* la cuál devuelve el objeto *astra_body_list_t* con los cuerpos encontrados, además de la variable de estatus que indicará si el tracking se ha realizado correctamente o en su defecto no se ha encontrado ningún cuerpo.

```
const astra_status_t rc = astra_bodyframe_body_list(bodyFrame, &bodyList);
if (rc != ASTRA_STATUS_SUCCESS)
{
    printf("Error %d in astra_bodyframe_body_list()\n", rc);
    return;
}
```

La clase *astra_body_list_t* cuenta con diferentes atributos bastante útiles como la cuenta total de cuerpos traqueados, pero lo más interesante será el array de objetos de la clase *astra_body_t* los cuales hacen referencia a cada uno de los cuerpos de la imagen. A partir de este objeto se podrán acceder a cada una de las articulaciones previamente mencionadas, siendo la cabeza la más interesante para esta aplicación.

```
astra_body_t* body = &bodyList.bodies[i];

const astra_joint_t* joint = &body->joints[ASTRA_JOINT_HEAD];
```

Una vez tenemos el objeto *astra_joint_t*, ya se podrán extraer los datos de desplazamiento requeridos ya que son atributo de la clase. El atributo que ofrece las coordenadas x,y,z se llama *worldPosition.*

Otro dato interesante es el que ofrece el atributo *orientation*, el cual devuelve la matriz de rotación de la articulación en cuestión.

```
//COORDENADAS DESPLAZAMIENTO
const astra_vector3f_t* worldPosition = &joint->worldPosition;
//ROTACION
const astra_matrix3x3_t* orientation = &joint->orientation;
const astra_vector3f_t* xAxis = &orientation->xAxis; // m00, m10, m20
const astra_vector3f_t* yAxis = &orientation->yAxis; // m01, m11, m21
const astra_vector3f_t* zAxis = &orientation->zAxis; // m02, m12, m22
```

A partir de los datos de desplazamiento se pintará sobre la imagen de profundidad puntos que indicarán la posición de la cabeza en la imagen y de esta forma se podrá realiza el seguimiento de los usuarios de una manera gráfica. En cuanto al atributo de orientación de vuelve la matriz de rotación como la que aparece a continuación a partir de la cual se podrán obtener los grados de torsión de la cabeza en x,y,z.

$$R = \begin{pmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_x u_z(1-\cos\theta) + u_y\sin\theta \\ u_y u_x(1-\cos\theta) + u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_y u_z(1-\cos\theta) - u_x\sin\theta \\ u_z u_x(1-\cos\theta) - u_y\sin\theta & u_z u_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{pmatrix}$$

Una vez gestionados los recursos de la cámara para obtener los datos deseados toca conectarse con la aplicación externa para de esta manera transmitir estos datos. Para ello he decidido conectar los dispositivos a través de la red haciendo uso del protocolo UDP, el cuál es ideal para aplicaciones en tiempo real. Esto es debido a que, aunque no se lleva un control de los paquetes enviados ni se realiza una conexión previa (por eso a diferencia de TCP, se dice que no es orientado a conexión), lo que, aunque puede significar la perdida de algo de información en algunos momentos sobre todo si la red está saturada, también permite una transmisión más rápida de los datos.

Con el fin de llevar a cabo la transmisión, la aplicación externa hará de servidor mientras que ésta hará de cliente enviando los datos a cada *frame* obtenido por el *stream*. Para implementar esta estructura utilizaremos las librerías de *winsock2* a través de las cuales podremos levantar un cliente y un servidor en las direcciones IP y puertos deseados y pertenecientes a la misma red**.**

5.3 Resultados

Una vez finalizado el script encargado de administrar los recursos de la cámara *Orbbec* Astra, ya es posible realizar el seguimiento de diferentes usuarios y comunicarse con un servidor UDP para pasarle tanto las coordenadas de los usuarios como la matriz de rotación con los datos de torsión de la cabeza.

Con el fin de recibir los datos de tracking con una aplicación externa, se ha utilizado Matlab a modo de simulación, de forma que se levantará un servidor UDP que recibirá los datos del script de tracking.

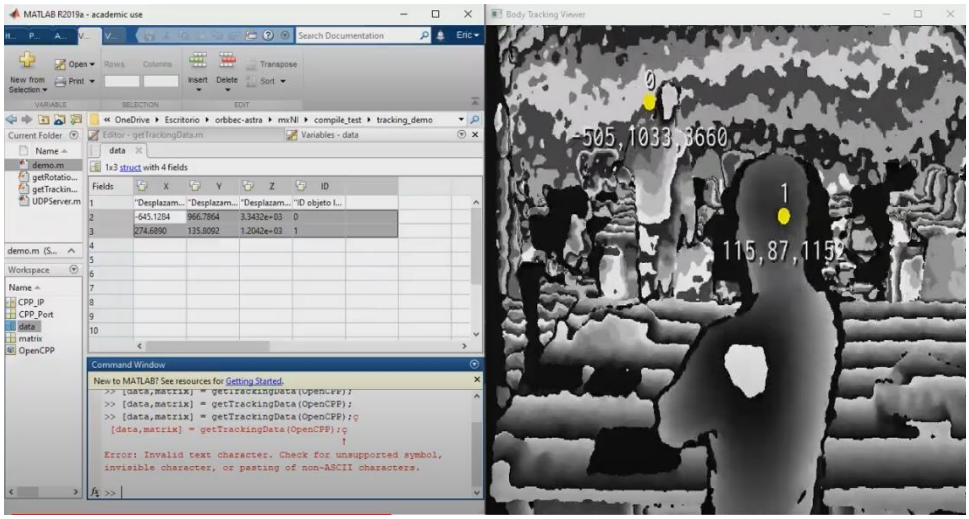

Ilustración 14. Ejemplo de funcionamiento de aplicación de tracking usando cámara 3D.

Para realizar un análisis completo de la aplicación y así discernir los puntos fuertes y débiles de esta, se probarán distintos aspectos que determinarán su precisión y robustez ante diferentes eventos.

- **Precisión**. Se medirá la precisión del modelo calculando la diferencia en cm entre los resultados obtenidos por la cámara y las distancias reales a las que se encuentra el usuario. Para conseguir determinar la distancia del usuario a la cámara, se empleará un metro laser de alta precisión. Además, se medirá la precisión del modelo a diferentes distancias, para de esta manera obtener la relación entre la distancia del usuario a la cámara y la precisión de las medidas.
- **Tiempo de ejecución.** A partir del cálculo del tiempo entre frame y frame, se podrá comprobar si la aplicación es capaz de correr a los frames por segundos indicados en las especificaciones. Para obtener una aproximación más cercana se sacará la media de tiempo tras el transcurso de 250 frames.
- **Errores de detección**. La robustez del sistema dependerá de varios factores, una de las pruebas realizadas será comprobar en que situaciones el modelo de clasificación basado en "deep learning", no es capaz de detectar los cuerpos que aparecen en la imagen. Las pruebas realizadas consistirán en comprobar cuanta parte del cuerpo es necesaria para una correcta detección y el rango de funcionamiento o, dicho con otras palabras, hasta que distancia es la cámara capaz de realizar las detecciones.
- **Iluminación**. La cantidad de luz en la escena suele ser un factor determinante en el funcionamiento de aplicaciones basadas en técnicas de visión artificial. Para comprobar la influencia de la iluminación, se empleará la aplicación de android "Lux Light Meter" para comprobar los niveles en luxes de luz en el ambiente.

Precisión

Para obtener la precisión del modelo a diferentes distancias, se van a tomar diferentes medidas cada medio metro, partiendo de 1 metro de distancia y llegando hasta los 4 metros, distancia a la que la cámara no puede realizar la detección del cuerpo correctamente. Los resultados serán representados a través de una gráfica para de ésta comprobar los resultados de una manera visual.

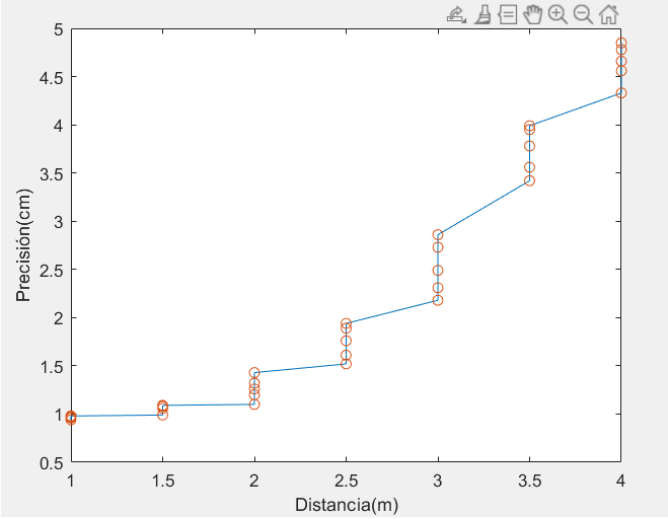

Ilustración 15. Gráfico de relación error - distancia

Como se puede observar en el gráfico, la distancia afecta directamente a la precisión de las medidas. En los primeros 3 metros el error de precisión solo llega hasta los 2 cm, pero a partir de los 3 metros el error asciende exponencialmente hasta casi los 5 cm. Por otro lado, dentro de los escalones de distancia se puede apreciar que cuanto mayor es la distancia mayor es la dispersión en las medidas de cada escalón, por lo que se hará cada vez más difícil intentar compensar en función de la distancia.

Tiempo de ejecución

Para calcular el tiempo de ejecución se hará un test durante 250 fotogramas donde aparecerán diferentes personas en la imagen, las cuales estarán en continuo movimiento. Despúes de calcular el tiempo transcurrido entre frames, se ha obtenido un resultado medio de 0.035 ms en cada transición. A partir de este dato podemos obtener los fotorgamas medios por segundo, dando como resultado una media de 28.5 FPS. De los resultados obtenidos se puede entender que cuando el software tiene que detectar a más de un usuario, se produce una pequeña caída de FPS debido al coste computacional que ello conlleva.

Errores de detección.

En este apartado, y como se ha comentado antes, se va a comprobar la robustez del modelo frente a diferentes situaciones.

- Rango de funcionamiento: Después de probar la distancia máxima a la que se producen las detecciones correctamente, en el 90% de los casos la detección falla a partir de los 4.5 m, siendo éste su rango máximo de funcionamiento.

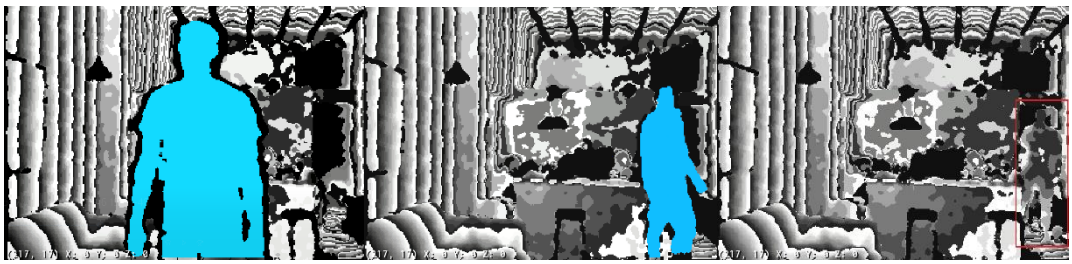

Ilustración 16. Prueba rango de funcionamiento

- Robustez de la detección. La otra prueba que se llevará a cabo será la comprobación de la robustez de las detecciones, para ello se ha utilizado un video donde el usuario va mostrando su cuerpo en pantalla paulatinamente, para de esta forma comprobar cuanta fracción de cuerpo es necesaria para la detección.

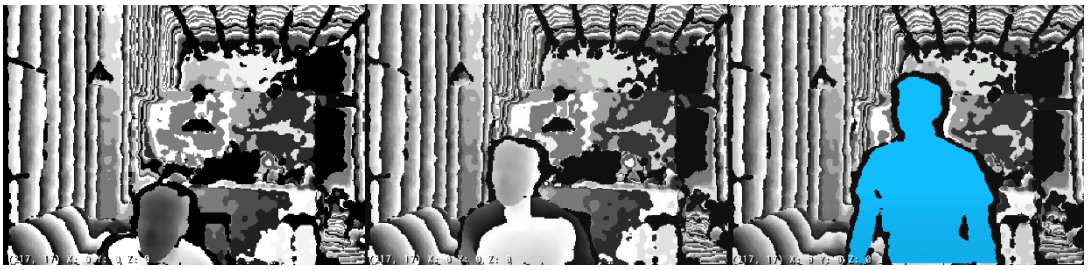

Ilustración 17. Prueba robustez de la detección I

Por otro lado, tras una serie de comprobaciones, se ha podido discernir que cuando se ha realizado la detección, se puede realizar la detección en el siguiente frame aunque la fracción de cuerpo sea muy pequeña. De esta observación se puede sacar la conclusión de que una vez se ha realizado la detección, se busca esa misma detección en los siguientes frames, haciendo el sistema mucho más robusto.

Ilustración 18. Prueba robustez de la detección II

Iluminación

La última prueba que se realiza será la comprobación del efecto producido por cambios de iluminación en la sala. Haciendo uso de un medidor de luminancia se ha comprobado el funcionamiento a diferentes niveles. Después de realizar la prueba, se ha comprobado que los cambios en la iluminación de la sala no producen ningún efecto en el sistema. Esto tiene sentido, ya que la imagen de profundidad utilizada para aplicación del detector de cuerpos se implementa a partir de los datos obtenidos a través del sensor de infrarrojos. Este hecho implica una gran ventaja a la hora de utilizar este tipo de cámaras.

6 Aplicación de *tracking* mediante el uso de cámaras webcam

Una vez vista la implementación de la aplicación de tracking haciendo uso de la cámara 3D, se pasará a realizar el montaje e implementación de un sistema que empleará dos cámaras estilo webcam sin ningún otro tipo de sensor para realizar la misma función.

Para llevar a cabo el montaje se emplearán dos cámaras de la marca “papalook” con las siguientes especificaciones:

- Full HD 1080P con enfoque manual.
- Ángulo de visión de 65 grados.
- Solo compatible con Windows.

Como software para implementar la aplicación se utilizará Matalb, que aparte de ser una potente herramienta que permite realizar la implementación, depuración y visualización de resultados de una manera cómoda, cuenta con diferentes *plugins* que permiten y facilitan diferentes labores. Éstos *plugins* son denominados *toolbox* y uno de los más determinantes en este proyecto ha sido *Computer Vision Toolbox* el cual cuenta con diferentes funciones y aplicaciones que permiten llevar a cabo diferentes técnicas de visión artificial fácilmente.

La intención es calibrar y obtener imágenes con ambas cámaras de forma sincronizada. De esta manera, se aplicarán diferentes técnicas de visión artificial para detectar el rostro del usuario frente a la cámara. Una vez determinado la posición en la imagen del usuario, gracias a la previa calibración y empleando una técnica basada en trigonometría denominada triangulación, determinar la distancia del usuario a las cámaras, las cuales se encontrarán y a una distancia alineadas entre sí conocida.

En resumen, los pasos seguido para la implementación de la aplicación serán:

- Calibración
- Detección de rostro del usuario.
- Cálculo de distancia mediante triangulación

A continuación, se explicarán de una forma desarrollada cada uno de los puntos mencionados, para concluir con el análisis de los resultados obtenidos.

6.1 Calibración

El primer paso será corregir la percepción del sensor de la cámara para que el sistema de referencia de la cámara coincida con la realidad y a su vez corregir las posibles distorsiones introducidas por la lente y perspectiva. La técnica para solucionar estos problemas se denomina calibración y permite obtener a partir patrones preestablecidos, como podrían ser las esquinas de un tablero de ajedrez, diferentes parámetros que definen el comportamiento y percepción de la cámara, estos parámetros se dividen en intrínsecos y extrínsecos. Además de para corregir perspectiva y posibles distorsiones, el proceso de calibración es un paso necesario para obtener medidas de la escena a partir de imágenes de esta. La exactitud de la calibración determinará posteriormente la precisión de las medidas que se realicen.

Los **parámetros intrínsecos** representan las propiedades ópticas de la lente de la cámara, estos son fijos y permiten proyectar puntos 3D del mundo real a puntos 2D en el plano de la imagen. Dentro de esta categoría los principales parámetros son:

- **Distancia focal**. Representa la distancia entre el a lente y el sensor de forma que al variar la distancia focal se conseguirá un mayor o menor acercamiento en la imagen, lo que entendemos como el zoom de la cámara. Además, éste dato es necesario a la hora de calcular las medidas de objetos reales en el plano 2D de la imagen.
- **Punto principal**. Es el punto a partir del cual, mirando hacia el punto focal, se define el eje óptico 3D, de forma que la lente de la cámara es simétrica con respecto a su punto principal

Los parámetros intrínsecos son representados en la denominada matriz de calibración que determinará las propiedades ópticas de la cámara, siendo (fx ,fy) el punto focal y (cx,cy) el punto principal.

$$M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

En el caso de los **parámetros extrínsecos** definen la posición espacial de la cámara con respecto al mundo real, de forma que se expresa la relación entre un punto real y su proyección en la imagen haciendo uso de una matriz de rotación y un vector de translación. En la siguiente ecuación se define la relación entre un punto real y otro proyectado en la imagen, siendo R la matriz de rotación, t el vector de translación y A los parámetros internos de la cámara.

$$s\begin{pmatrix} p' \\ 1 \end{pmatrix} = A\begin{pmatrix} & R & & t \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} p \\ 1 \end{pmatrix}$$

Para el cálculo de los parámetros internos de calibración tomaremos como nomenclatura para las coordenadas del objeto real tridimensional (Xm, Ym, Zm), mientras que para el sistema de coordenadas de la cámara usaremos (Xc, Yc, Zc). La variable R hará referencia a la matriz de rotación y T al vector de translación. Partiendo de esta premisa, y conociendo el valor de T y R se pueden

obtener las coordenadas reales a partir de las coordenadas de la cámara, multiplicando estas por la matriz de rotación y sumándole a su vez el vector de translación.

$$\begin{bmatrix} Xc \\ Yc \\ Zc \end{bmatrix} = R \begin{bmatrix} Xm \\ Ym \\ Zm \end{bmatrix} + T = \begin{bmatrix} r1 & r2 & r3 \\ r4 & r5 & r6 \\ r7 & r8 & r9 \end{bmatrix} \begin{bmatrix} Xm \\ Ym \\ Zm \end{bmatrix} + \begin{bmatrix} Tx \\ Ty \\ Tz \end{bmatrix}$$

Sin tener en cuenta la distorsión radial y tangencial, conociendo las coordenadas 2D en la imagen (Xi, Yi) obtenidas a través de la cámara y el factor de escalada referente al zoom de la cámara (k):

$$\begin{bmatrix} Xi \\ Yi \\ 1 \end{bmatrix} = k \begin{bmatrix} Fcx & 0 & Cx \\ r4 & Fcy & Cy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Xc \\ Yc \\ Zc \end{bmatrix}$$

Se pueden despejar las distancias focales Fcx y Fcy, además de los valores de las coordenadas del centro óptico de la cámara (Cx,Cy).

En este caso se van a calibrar dos cámaras alineadas y colocadas a una distancia conocida entre sí, de forma que aparte de calibrar cada una de ella individualmente, es interesante obtener la posición y orientación de la cámara 2, en relación con la cámara 1. Para llevar a cabo este proceso de una manera cómoda, se va a emplear la aplicación de Matlab perteneciente al *toolbox* de visión artificial *Stereo Camera Calibrator*, el cual permitirá obtener un objeto con los parámetros de calibración. Gracias a la aplicación seremos capaces de calcular las ubicaciones tridimensionales correspondientes a los puntos deseados de la imagen, pero además es útil para rectificar las distorsiones de la imagen y reconstrucción 3D de la escena.

Es importante llevar a cabo la calibración de una manera rigurosa ya que de ello dependerá la precisión del modelo que posteriormente utilizaremos para calcular las distancias a la cámara. Partiendo de esta premisa, será necesario seguir una serie de pasos que junto con las funcionalidades de la aplicación de calibración nos permitirá realizar la labor de una manera correcta.

- **Preparación de cámaras y patrón de calibración**. Lo primero será colocar las cámaras de manera alineada y a una distancia conocida entre sí, por ello se hará uso de un metro para comprobar la correcta alineación de las lentes y establecer una distancia entre las cámaras de 25 cm.

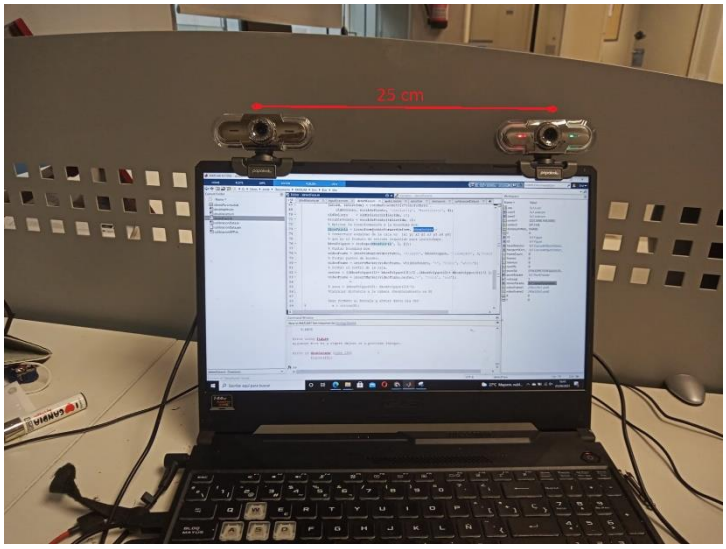

Ilustración 19. Montaje de cámaras.

Una vez colocadas las cámaras correctamente será necesario preparar el patrón de calibración que utilizaremos para extraer los puntos clave necesarios. Existen diferentes patrones, pero el más estandarizado es el tablero ajedrez cuyas esquinas internas conformarán los puntos de calibración. Recomiendo la impresión de un tablero de ajedrez para posteriormente pegar a una tabla que servirá para conseguir una perspectiva homogénea, de forma que el patrón del tablero deberá cumplir ciertos requisitos:

  - El tablero deberá ser rectangular, es decir en un lado contará con un número impar de cuadrados y en otro un número par.

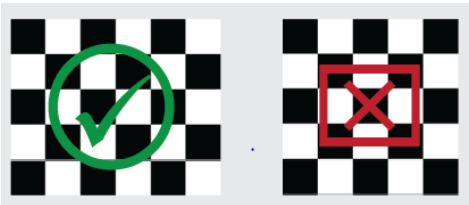

Ilustración 20. Ejemplo de correcto patrón de calibración.

  - Se deberá conocer el tamaño de los lados de los cuadrados que conforman el tablero
  - Para mejorar la velocidad de detección es recomendable evitar en mayor medida el desorden del fondo de la imagen.

- **Obtener pares de imágenes**. Una vez preparadas las cámaras y el patrón de calibración, será necesario realizar fotos del tablero de ajedrez de forma sincronizada con las dos cámaras, de forma que ambas imágenes se tomen en el mismo instante de tiempo, pero en las diferentes perspectivas de la cámara. Lo ideal es tomar entre 10 y 20 pares de imágenes para conseguir una calibración más robusta.

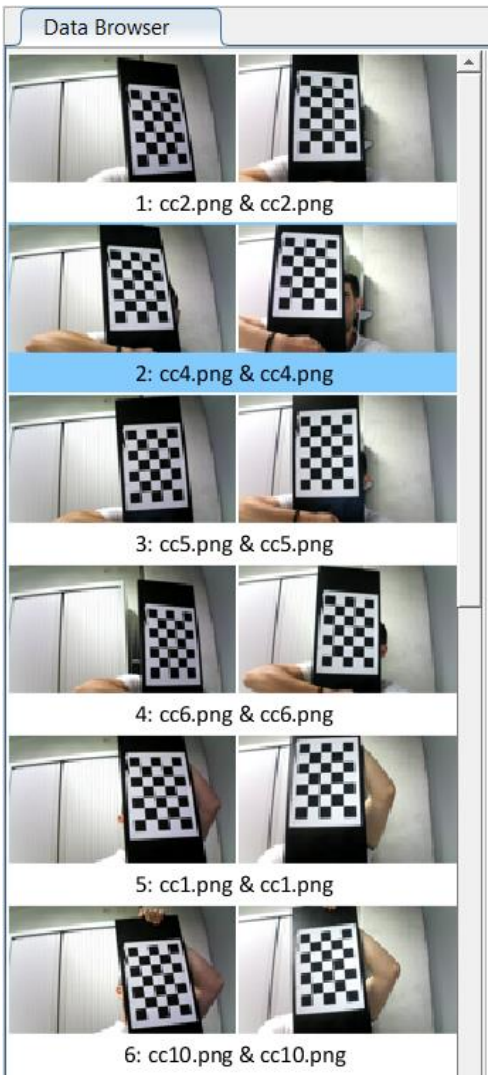

Ilustración 21. Pares de imágenes para calibración.

Una vez obtenidas las imágenes abriremos la aplicación de calibración estero, para de esta manera, accediendo a la opción "Agregar imágenes" en la sección "Archivo" de la pestaña "Calibración" cargar los pares de imágenes previamente realizados. Después de cargar las imágenes pertinentes tendremos la opción de introducir la longitud del lado los cuadrados del tablero.

Con las imágenes ya cargadas, existe la posibilidad de que la aplicación descarte los pares defectuosos. Como se puede apreciar en la siguiente imagen las esquinas del tablero se detectan automáticamente, pudiendo analiza runa a una las imágenes para asegurarse de que las esquinas han sido correctamente detectadas usando el zoom, lo cual es recomendable para conseguir la máxima precisión en la calibración.

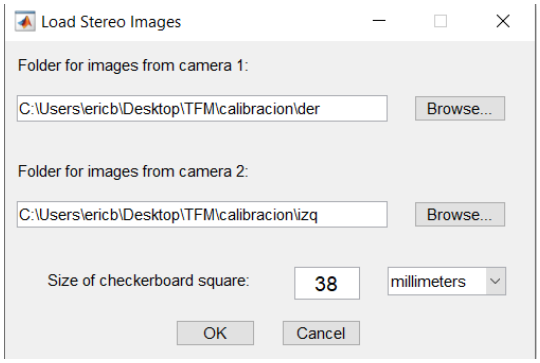

Ilustración 22. Cargar imágenes para calibración.

- **Calibrar**. Gracias a la aplicación el proceso de calibración será tan sencillo como pulsar la opción "Calibrar". A continuación, apreciaremos en la parte inferior de la pantalla la aparición de dos ventanas nuevas, una con los errores de reproyección y otra con el gráfico 3D que representa los parámetros extrínsecos de las cámaras.
    - Los **errores de retroproyección** representan las distancias, en píxeles, entre los puntos detectas y los reproyectados. La aplicación calcula los puntos de ajedrez en el mundo real, proyectados en la imagen, de forma que tras una comparación de ambos se obtiene el error mencionado en cada uno de los pares de imágenes. La idea es obtener un error menor a un *pixel*, pero obviamente cuanto menor sea el error más precisa será la aplicación. Para calcular el error de retroproyección se calculará el error cuadrático medio entre varios puntos del mundo real y sus proyecciones en la imagen, obteniendo de ésta manera el error entre los puntos en píxeles.

$$\mathrm{ErrorGeometrico}\,(F) = \sqrt{\frac{1}{n}\cdot\sum_{i=1}^{n}\mathrm{dist}\left(m, Fm'\right)^2}$$

En la siguiente gráfica se aprecia el error de retroproyección medio para diferentes pares de imágenes.

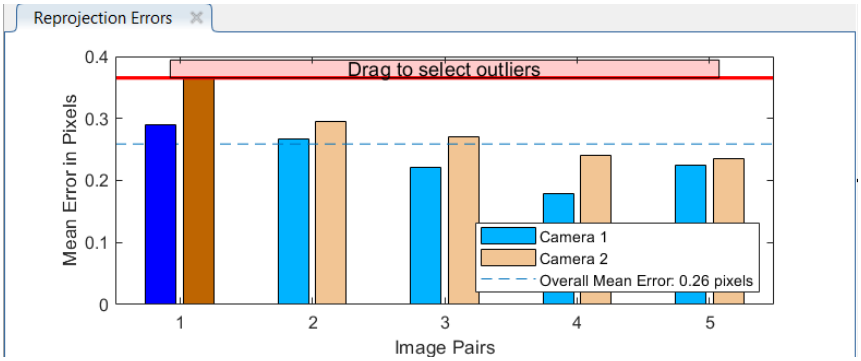

Ilustración 23. Errores de re-proyección.

- En el gráfico 3D de la derecha, podremos visualizar los parámetros extrínsecos tanto de las cámaras al tablero como viceversa, pudiendo cambiar de opción haciendo uso de las dos pestañas posibles. Examinando el gráfico podremos observar como la aplicación ha calculado la posición relativa de las cámaras y de los tableros pudiéndonos hacer una idea de si la calibración es correcta y pudiendo comprobar que la distancia entre las cámaras es la utilizada.

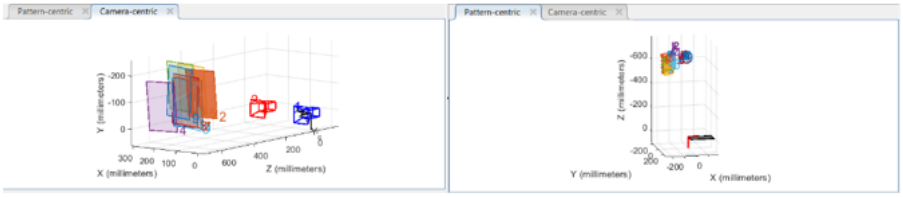

Ilustración 24. Gráfico 3D del resultado de la calibración.

- **Exportar parámetros.** Una vez satisfechos con el resultado de la calibración se podrán exportar los datos accediendo a la opción "Exportar parámetros de la cámara", obteniendo de esta manera un objeto con el nombre que deseemos, que tendrá como atributos los parámetros de calibración extraídos, entre ellos la matriz de rotación y el vector de transformación que serán muy útiles para la posterior triangulación.

Para acabar con la calibración, cabe mencionar que es posibles mejorar el resultado de ésta aplicando diferentes técnicas como quitar los pares de imágenes con mayor error o cambiar el número de coeficientes de distorsión.

6.2 Detección de rostros

Uno de los pasos clave a la hora de implementar la aplicación de tracking será detectar el objeto a seguir, en este caso el objetivo es seguir el rostro del usuario en el escenario. Para detectar rostros existen diferentes técnicas basadas en visión artificial que cumplen la labor de forma eficiente, en el siguiente punto se explicarán y compararán dos técnicas diferentes. Por un lado, se implementará el clásico algoritmo para detección facial de Viola-Jones en combinación con el algoritmo Kanade–Lucas–Tomasi (KLT) con la intención de hacer el modelo más robusto, por otro lado, se hará uso de técnicas más avanzadas de *Deep Learning* haciendo uso de la herramienta *MultiTask Cascaded Convolutional Neural Network* (MTCCN).

6.2.1 Viola-Jones

El algoritmo de Viola-Jones es uno de los algoritmos más antiguos y estandarizados a la hora de detectar rostros, fue implementado en 2001 por Paul Viola, de Mitsubishi Electric Research *Labs* y Michael Jones, de Compaq CRL. En la actualidad se sigue utilizando en diversas aplicaciones debido a su bajo coste computacional.

Viola-Jones se basa en la comparación entre las intensidades de diferentes regiones aplicando diferentes filtros para ello y obteniendo de esta forma las denominadas características Haar. Estas características son utilizadas por un conjunto de clasificadores pequeños que en su combinación forman uno mayor y más robusto, esta técnica de clasificación se conoce como clasificador en cascada y ésta basado en el algoritmo de aprendizaje AdaBoost.

6.2.2 Descriptores Haar

Los descriptores de tipo de Haar son unos de los más eficientes debido a su bajo coste computacional. Están basados en el cálculo de diferencia de intensidad entre pixeles, aplicando filtros de diferentes formas y escalas por toda la imagen, estos son denominados filtros de Haar.

Ilustración 25. Ejemplos de filtros Haar.

Como se puede apreciar en la ilustración los filtros tienen diferentes formas y éstos se aplican por toda la imagen en todas las posibles posiciones. El funcionamiento de los filtros es muy simple, los rectángulos negros aportarán una contribución negativa mientras que los negros una negativa, de forma que se sumarán las intensidades lumínicas de los píxeles de cada una de las regiones para a continuación restar el sumatorio de la región negra a la de la blanca.

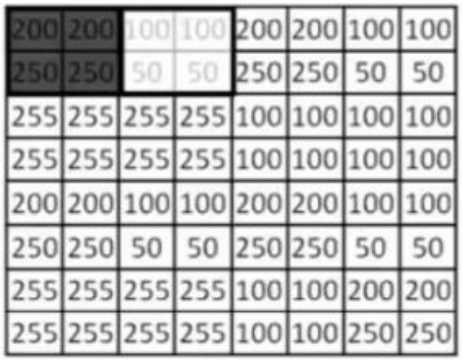

Ilustración 26. Ejemplo de aplicación de filtro en la imagen.

Siguiendo la lógica mencionada en el ejemplo se aplica un filtro en una de las posibles posiciones que puede adquirir dentro de la imagen donde se llevaría a cabo la suma de intensidades de cada región para luego ser restadas entre sí dando como resultado el cálculo de (200+200+250+250) – (100+100+50+50) = 600.

El resultado puede requerir de normalización por el tamaño de la ventana del filtro para conseguir que los valores sean invariantes a cambios en los tamaños de los objetos y también para que características aplicadas a diferentes escalas en toda la imagen tengan valores comparables. Igualmente, y de forma previa puede ser necesario normalizar la imagen para conseguir también invariancia a cambios de iluminación.

Después de aplicar diferentes filtros a diferentes escalas se dará lugar a la obtención de una elevada cantidad de cálculos que darán como resultado los descriptores de tipo Haar, los cuales se utilizarán posteriormente para clasificación haciendo uso de un clasificador en cascada.

6.2.3 Adaboost

Adaboost es un método de clasificación el cual nos permite tratar de forma muy eficiente con un número muy elevado de características durante el proceso de aprendizaje, lo cual nos vendrá estupendamente a la hora de trabajar con descriptores HAAR, ya que éstos exigen el procesamiento de una gran cantidad de características.

La base de este clasificador es la combinación de diferentes clasificadores muy simples, pero que sin embargo la todos combinados nos va a acabar dando como resultado final un clasificador global que minimizará el error de clasificación. Por otro lado, cada clasificador va a dar un peso relativo a cada ejemplo para aprender dependiendo del resultado de los pasos anteriores. Así para cada nuevo clasificador simple que se va a aprender, se va a incrementar el peso de aquellos ejemplos que se han clasificado mal en los pasos anteriores mientras que se va a disminuir el peso de aquellos ejemplos que están bien clasificados, de esta forma los clasificadores se irán centrando en aquellos ejemplos que no se han clasificado en paso anteriores. Veamos un ejemplo gráfico:

1. Tenemos un conjunto de entrenamiento con dos clases de características diferentes y vemos que es imposible trazar una línea que separe perfectamente a las dos

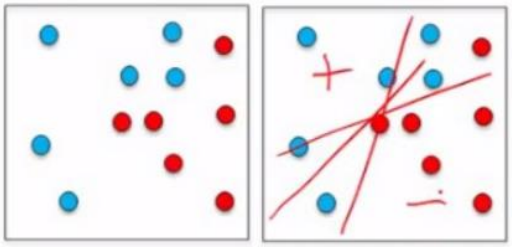

Ilustración 27. Conjunto de entrenamiento de dos clases características diferentes.

2. Aplicamos un clasificador simple que marcará un umbral, y todas las muestras que superen ese umbral se separarán de las que no lo hagan.

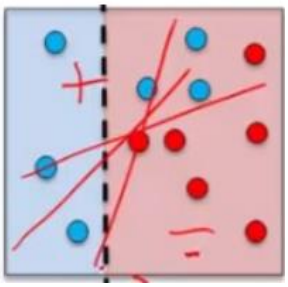

Ilustración 28. Clasificación de descriptores 1.

3. Cómo vemos hay tres muestras que han sido mal clasificadas, por lo que éstas recibirán un peso relativo mayor mientras que al resto un peso menor, de forma que la clasificación se ve condicionada por este hecho, como vemos en la imagen.

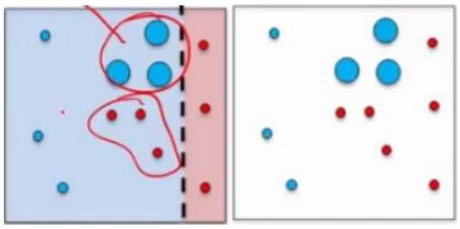

Ilustración 29. Añadir librerías al proyecto.

4. Después del último clasificador se han quedado tres muestras clasificadas, por lo que repetiremos el proceso anterior.

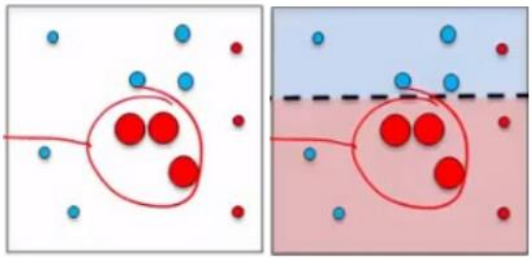

Ilustración 30. Clasificación de descriptores 2.

5. Finalmente, todas las clasificaciones se agrupan en un clasificador global, de forma que las regiones donde exista una mayor cantidad de muestras de las clases A serán clasificadas como regiones A y lo mismo con la B.

La cascada de clasificadores nos va a permitir alcanzar este objetivo mediante una combinación secuencial de clasificadores, de forma que una imagen solo será detectada como cara si realmente es reconocida de forma correcta como cara por todos los clasificadores de la cascada. De esta manera tras la combinación de diferentes clasificadores simples se obtiene uno mucho más robusto.

6.2.4 Algoritmo Kanade–Lucas–Tomasi

Tras realizar diferentes pruebas con el algoritmo de detección de Viola-Jones se puede apreciar que el modelo no es demasiado robusto a cambios de iluminación y sobre todo a cambios en la perspectiva de la cara, de manera que el rostro solo se es correctamente detectado si se encuentra rigurosamente de cara a la cámara.

El algoritmo KLT se basa en el rastreo de un conjunto de puntos característicos en la región de la imagen donde se encuentre el objeto a detectar, para de esta forma en los siguientes *frames* se buscarán estos puntos clave para determinar la ubicación del objeto a detectar en la imagen. Otra utilidad interesante que permite la aplicación de este algoritmo es el cálculo de la transformación geométrica de los puntos de un *frame* a otro, de forma que se podría calcular, en el ejemplo de detección de rostros, los grados de torsión de la cabeza en el eje vertical. Los pasos que se siguen son los siguientes.

1. Detección del rostro haciendo uso del algoritmo Viola-Jones de la manera previamente explicada.

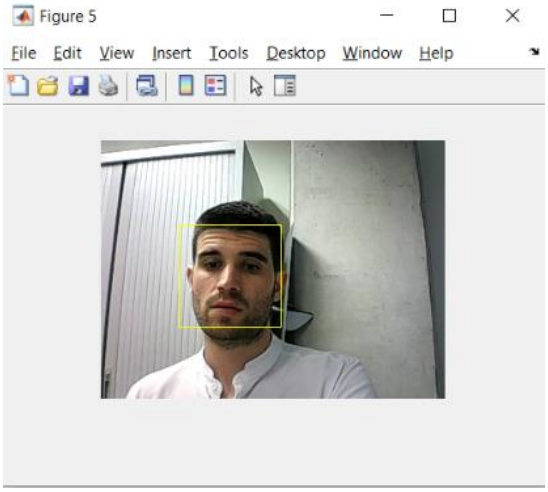

Ilustración 31. Ejemplo de detección usando el algoritmo de Viola-Jones.

2. Obtención de los puntos característicos dentro de la detección, siendo estos puntos pertenecientes a bordes o a transiciones de intensidad entre píxeles.
3. En el siguiente *frame*, buscar estos puntos previamente obtenidos para de esta forma determinar la nueva detección sin usar el algoritmo de Viola-Jones.

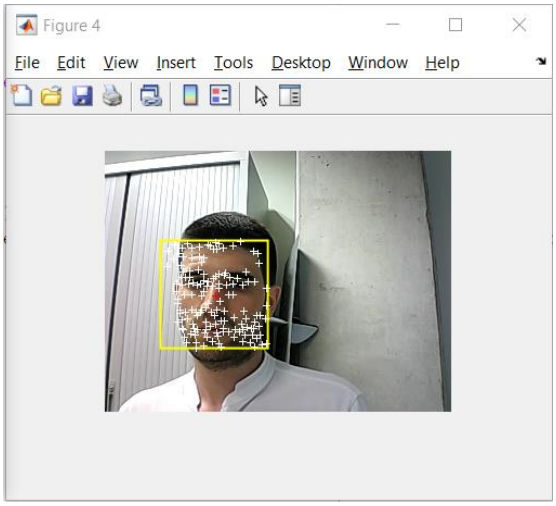

Ilustración 32. Ejemplo de detección usando el algoritmo de Viola-Jones y KLT.

4. Calcular la transformación geométrica de los puntos de un frame a otro, para así aplicar esta transformación al "bounding box" y de esta manera vera la transformación geométrica gráficamente, o simplemente calcular los grados de torsión.
5. Si los puntos detectados de un frame a otro son insuficientes para determinar la ubicación del objeto, se vuelve a aplicar Viola-Jone para de éste forma volver a empezar con el proceso.

Gracias a este método el modelo de clasificación se vuelve bastante más robusto a cambios de luminosidad o perspectiva, a costa de aumentar en cierta manera el coste computacional de éste.

6.2.5 MTCNN

Para el siguiente modelo de clasificación se va a emplear MTCNN una herramienta de visión artificial que salió a la luz en 2016 por Kaipeng Zhang basada en Deep Learning utilizada para detección de objetos. La red que conforma el modelo está estructurada en tres capas, siguiendo los siguientes pasos:

1. La imagen es reescalada a diferentes tamaños dando lugar a la llamada pirámide de imágenes

Ilustración 33. Ejemplo de rescaldado MTCNN.

2. **Proposal Network**. En la primera fase se analizan las imágenes reescaladas para empezar a realizar las primeras detecciones usado un umbral muy bajo, de manera que se obtienen como resultado una gran cantidad de falsos positivos que serán filtrados en las siguientes capas.

Ilustración 34. Ejemplo de P-Net MTCNN.

3. **Refine Network**. Como su nombre indica en esta segunda fase del modelo se refinarán los falsos positivos obtenidos previamente. Para ello, el algoritmo se encarga de buscar puntos clave relacionados con la fisionomía facial con la que se ha entrenado previamente el modelo, dando lugar a un menor número de candidatos mucho más precisos.

Ilustración 35. Ejemplo de R-Net MTCNN.

4. **Output Network.** En la última fase o fase de salida se realizará un filtrado similar a la de la fase anterior, pero esta vez se buscarán los 5 puntos característicos que conforma ojos, nariz y boca. Para acabar, a la salida de esta capa se obtendrá una detección de gran precisión del rostro buscado.

Ilustración 36. Ejemplo de O-Net MTCNN.

En conclusión, el método emplea red estructurada para conseguir unos resultados precisos y robustos en la detección de rostros. En los siguientes puntos se realizará una comparación con respecto al anterior modelo para ver hasta qué punto mejora la detección en tiempo real.

6.2.6 Comparación entre métodos de detección facial.

Una vez vista la base teórica sobre los dos métodos de detección facial propuestos, en el siguiente punto se compararán y analizarán los diferentes aspectos que nos ayudarán a evaluar los métodos de forma objetiva. Para la evaluación se utilizará el video que se encuentra en el *toolbox* de visión por computador por defecto para pruebas, en el cual el usuario realiza varios movimientos de cabeza y de desplazamiento en todas las direcciones para comprobar la robustez de la detección.

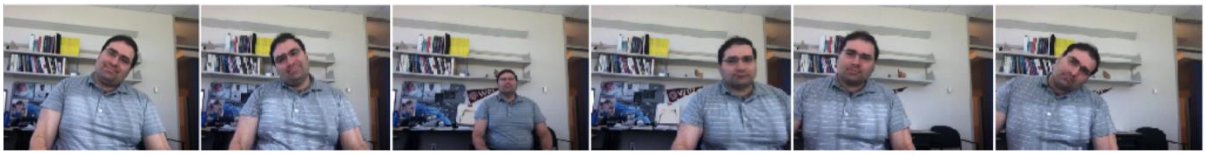

Ilustración 37. Frames extraidos del video de testeo.

Los aspectos que se evaluarán de esta forma son:

- Frames en los que no se detecta el rostro del usuario.
- Frames en los que se realiza una detección errónea.
- Tiempo de procesado.
- Precisión de la detección.

Para llevar a cabo la comparación se empleará el siguiente script en el cual se realizan un conteo de los errores de detección incrementando el contador de errores cuando el *bounding box* que marca la detección esté vacío. Además, haciendo uso de la herramienta *tic-toc* se calculará el tiempo de procesado de la detección y de esta manera obtener una estimación de los *frames* por segundos a los que podría funcionar el modelo.

```
clc;clear;
video = VideoReader('tilted_face.avi');

while  frames <  video.NumFrames
    frames = frames + 1;
    videoFrame  = readFrame(video);
    tic
    % Detección usando MTCNN
    [bbox,center,videoFrame] = detectFacesMTCNN(videoFrame);
    % Detección usando KLT
    [bbox,videoFrame,center] = detectFaceKLT(videoFrame);
    time = toc;
    tiempo_total = tiempo_total + time;
    tiempo_medio = tiempo_total / frames;
    if isempty(bbox)
        error = error + 1;
    end
end
```

Ilustración 38. Código de Matalb para testeo.

6.2.6.1 Test de rendimiento del método Adaboost con KLT

A continuación, se pasará a analizar el test de rendimiento, haciendo uso del método de detección basado en Adaboost y KLT. En las siguientes imágenes podemos ver ejemplos de las diferentes detecciones y los resultados obtenidos.

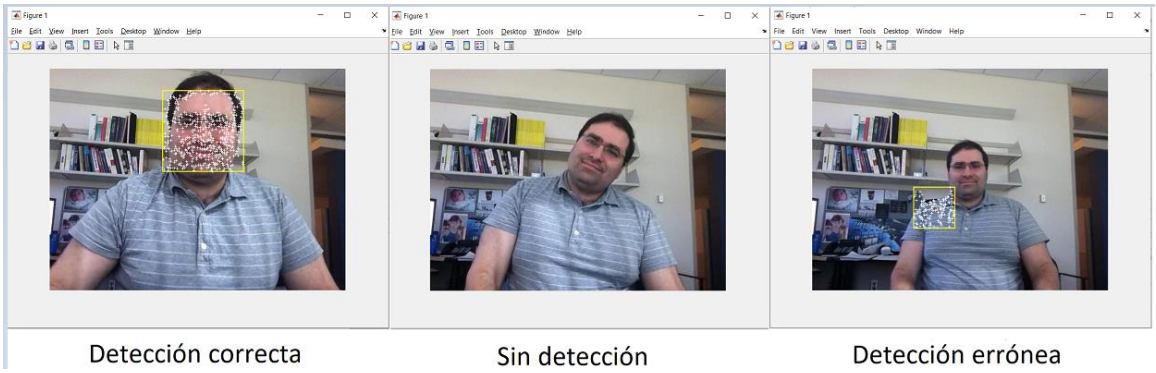

Ilustración 39. Ejemplos de detecciones con Adaboost y KLT.

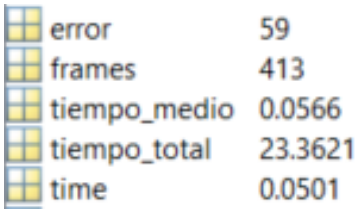

Ilustración 40. Variables resultantes del testeo con Adaboost y KLT.

- De los 413 *frames* que conforman el video de testeo, El tiempo medio de procesado por *frame* es de 0.0566s dando lugar un tiempo total de 23.36 s a lo largo de los 413 *frames* del video. Obviando el tiempo de procesado introducido por el script de testeo se puede afirmar que usando esta técnica se obtiene un *stream* que mantiene una ratio de prácticamente 20 *frames* por segundo.
- Por otro lado, analizando el *bounding box* obtenido tras la detección, se puede apreciar como los bordes de este no se acaban de ajustar del todo al rostro, dejando a ambos lados un margen de espacio. Una peor precisión en la detección conlleva la introducción de cierto error en el cálculo de la distancia, y más teniendo en cuenta que el margen de espacio varia *frame* a *frame*.

6.2.6.2 Test de rendimiento del método MTCNN

A continuación, se pasará a analizar el test de rendimiento, haciendo uso del método de detección usando MTCNN. En las siguientes imágenes podemos ver ejemplos de las diferentes detecciones y los resultados obtenidos.

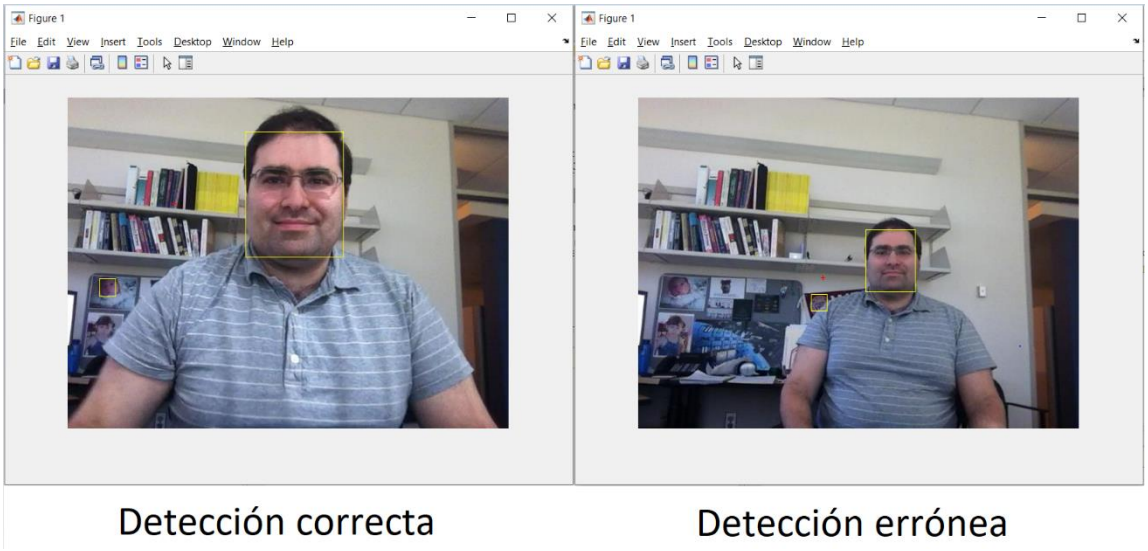

Ilustración 41. Ejemplos de detecciones con MTCNN.

error	0
frames	413
tiempo_medio	0.1101
tiempo_total	45.4756
time	0.1198

Ilustración 42. Variables resultantes del testeo con MTCNN.

- De los 413 *frames* que conforman el video de testeo, en ninguno de ellos no se ha realizado ninguna detección, en su contra algunos *frames* han presentado una segunda detección errónea, en concreto en 3 de ellos.
- Debido al coste computacional introducido por la red neuronal l tiempo medio de procesado por *frame* asciende a 0.1101s dando lugar un tiempo total de 45.47 s a lo largo de los 413 *frames* del video. Obviando el tiempo de procesado introducido por el script de testeo se

puede afirmar que usando esta técnica se obtiene un *stream* que mantiene una ratio de prácticamente 10 *frames* por segundo.

- Como se puede observar en la imagen, el *bounding box* resultante de la detección se ajusta casi a la perfección a los bordes del rostro y de manera uniforme, facilitando la labor del cálculo de distancias.

6.2.6.3 Conclusión de la comparación de métodos de detección facial.

Tras analizar los resultados obtenidos en el test de ambos métodos, mediante la comparación de éstos se pueden sacar conclusiones para discernir cuál de ellos es mejor para la labor de tracking. En la siguiente tabla se resumen los resultados obtenidos.

Tabla 4. Resultado del test de rendimiento entre métodos de detección facial.

	Adaboost + KLT	**MTCNN**
Frames por segundo	20 FPS	10 FPS
Errores de detección	59	0
Frames sin detección	9	3
Detección precisa	NO	SI

Como se puede ver en la tabla, la velocidad de reproducción a la que funciona el método de Adaboost es prácticamente del doble con respecto a MTCNN, esto es debido al alto coste computacional introducido por la red neuronal utilizada. Esta característica convierte al primer método en el mejor candidato si lo primordial es que la aplicación trabaje a tiempo real, a no ser que se utilice un equipo con mejores prestaciones y que pueda ofrecer una mayor velocidad de cómputo. Por otro lado, la precisión y robustez de Adaboost deja mucho que desear empeorando la precisión en el cálculo de las coordenadas del objeto. El método MTCNN presenta mucho mejores resultados en ésta aspecto, convirtiendo a este en la mejor opción si lo que apremia es la precisión. En conclusión, el mejor modelo dependerá de las necesidades del proyecto para el que se requiera la aplicación de tracking.

6.3 Cálculo de distancia mediante triangulación

Una vez obtenidos los parámetros de calibración y detectados los rostros en la imagen, será necesario obtener las coordenadas 3D del objeto a seguir a la cámara. Para ello y a partir de los parámetros de calibración y las dos perspectivas de las cámaras, se empleará el algoritmo de triangulación linear desarrollado por Hartley y Zisserman. En el siguiente punto se resumirá la base teórica sobre la que se cimenta el algoritmo de triangulación.

Para llevar a cabo el cálculo de la triangulación será necesario conocer los parámetros de calibración intrínsecos de cada cámara y la orientación de ambas cámaras, una respecto a la otra.

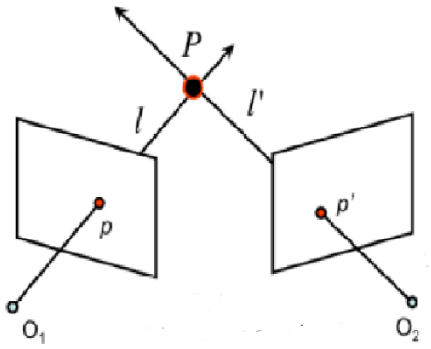

Ilustración 43. Boceto de triangulación.

Suponiendo que se conocen los puntos del objeto a seguir en ambas cámaras p y p', para conocer las coordenadas en 3D del punto real P se trazará las trayectorias l y l' entre los puntos centrales de la cámara O1 y O2 y los puntos de las imágenes p y p' para determinar la ubicación en la transición de ambas trayectorias. Para calcular las trayectorias con los centros de las cámaras existen diferentes métodos. A continuación, se explicará el algoritmo de triangulación lineal basado en la tesis de Hartley y Zisserman.

A partir del método de triangulación lineal se pretende calcula la intersección de las proyecciones de los puntos p y p'. Teniendo en cuenta que M es la matriz de proyección y que los puntos p y p' corresponden con el mismo desde diferentes perspectivas podemos decir que p = MP y que p'= M'P. Además, basándonos en el producto vectorial p X (MP) = 0. A partir de estas igualdades y la matriz de proyección se pueden declarar tres restricciones, donde Mi es la fila i de la matriz de proyección:

$$x(M_3P) - (M_1P) = 0$$
$$y(M_3P) - (M_2P) = 0$$
$$x(M_2P) - y(M_1P) = 0$$

Obteniendo las mismas restricciones para p' y teniendo en cuenta que AP = 0 podemos definir que:

$$A = \begin{bmatrix} xM_3 - M_1 \\ yM_3 - M_2 \\ x'M_3' - M_1' \\ y'M_3' - M_2' \end{bmatrix}$$

A partir de esta ecuación y utilizando la técnica de descomposición en valores singulares podremos estimar la ubicación del punto P en coordenadas 3D del mundo real. Cabe decir que un aspecto interesante de esta técnica es que se pueden utilizar infinitas perspectivas para de esta manera mejorar la precisión del cálculo.

6.4 Implementación y resultados

Finalmente, tras ver la base teórica sobre la que se fundamentará el desarrollo de la aplicación de tracking, se comentarán los pasos seguidos para la implementación de la aplicación haciendo uso de los dos métodos de detección. A continuación, se analizarán los resultados obtenidos y para de esta manera poder evaluar el comportamiento de la aplicación en comparación con el del resto de cámaras.

Como se ha comentado previamente, el primer paso será la calibración de las cámaras, para ello se utilizará el "toolbox" de visión por computador y la aplicación de calibración estero siguiendo los pasos vistos en puntos anteriores, para de esta forma obtener los parámetros de calibración necesarios para la posterior triangulación.

Para la triangulación se hará uso de la función del *toolbox* llamada *triangulate*, la cual a partir de las coordenadas del punto a seguir en ambas cámaras y los parámetros de calibración es capaz de obtener la ubicación tridimensional del punto común entre las cámaras. El punto a seguir en este caso será el centro del *bounding box* que se ha obtenido de la detección facial.

```
point3d = triangulate(center1, center2, stereoParams);
```

Por otro lado, la implementación de los métodos de clasificación se realizará a partir del uso de diferentes librerías para cada uno de ellos. En el caso de la implementación del método de Adaboost con KLT se hará uso de las funciones ofrecidas por el propio *Computer Visión Toolbox*, de forma que para la detección basada en descriptores HAAR se utilizará la función *CascadeObjectDetector* la cual a partir de un clasificador en cascada y los descriptores pasados como argumento, implementará el detector, si no se le pasa argumento alguno por defecto la función usará descriptores pertenecientes a una base de datos de rostros frontales.

```
faceDetector = vision.CascadeObjectDetector();
```

Para llevar a cabo el algoritmo de KLT, primero será necesario obtener los puntos pertenecientes a los bordes de la detección obtenida previamente a partir del clasificador en cascada, haciendo uso de la función *detectMinEigenFeatures*. Después, en el siguiente *frame*, usando un objeto de tipo *pointTracker* se rastrearán los puntos del anterior *frame* para determinar la nueva detección. Además, a partir de la función "estimateGeometricTransform2D" y pasándole los nuevos y antiguos puntos se obtendrá la transformación geométrica de los puntos de un *frame* a otro, esta transformación geométrica puede servir para obtener los grados de torsión de la cabeza o simplemente para aplicar dicha torsión al *bounding box* que marcará la detección.

```
points = detectMinEigenFeatures(videoFrameGray, 'ROI', bbox(1, :));
```

```
[xform, inlierIdx] = estimateGeometricTransform2D(...
    oldInliers, visiblePoints, 'similarity', 'MaxDistance', 4);
```

Mientras que para la implementación del método basado en MTCNN se utilizará el plugin dedicado *MTCNN Face Detection* cuyos requisitos mínimos son:

- MATLAB R2019 o superior
- Es necesaria la instalación de los siguientes *toolboxes*:
  - *Deep Learning Toolbox*
  - *Computer Vision Toolbox*
  - *Image Processing Toolbox*

Una vez cumplido con los requerimientos necesarios se podrá hacer uso de la función *detectFaces* poniendo en marcha la red neuronal explicada en puntos anteriores. Pasándole como argumento el *frame* con los rostros a detectar y devolverá las coordenadas de las detecciones en la imagen, los cinco puntos clave que conforman boca, nariz y ojos y las probabilidades obtenidas tras el uso de la red neuronal.

```
[bbox, scores, landmarks] = mtcnn.detectFaces(videoFrame);
```

Una vez implementado ambos métodos y conociendo las fortalezas y debilidades de cada uno de ellos, es hora de comprobar el resultado final y la precisión con la que se calculan las coordenadas del usuario en seguimiento. Para comprobar el funcionamiento correctamente, se emplearán ambos métodos y mediante el uso de un medidor láser se comprobará la precisión de los datos de distancia entre las cámaras y el usuario.

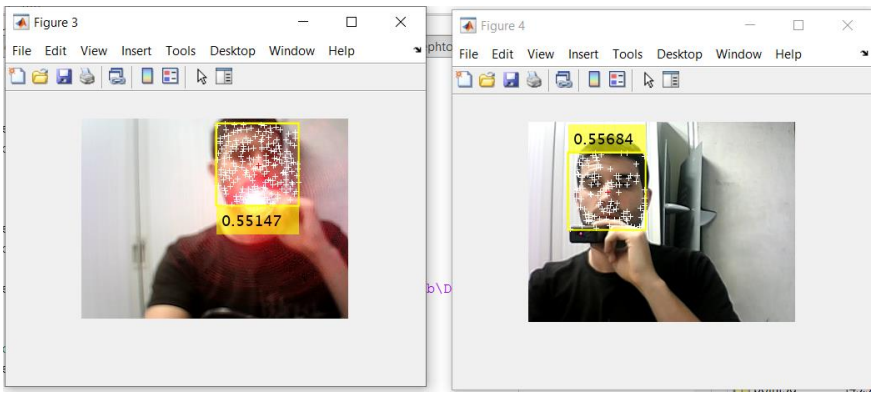

Ilustración 44. Frames de testeo de precisión de la aplicación usando Adaboost y KLT.

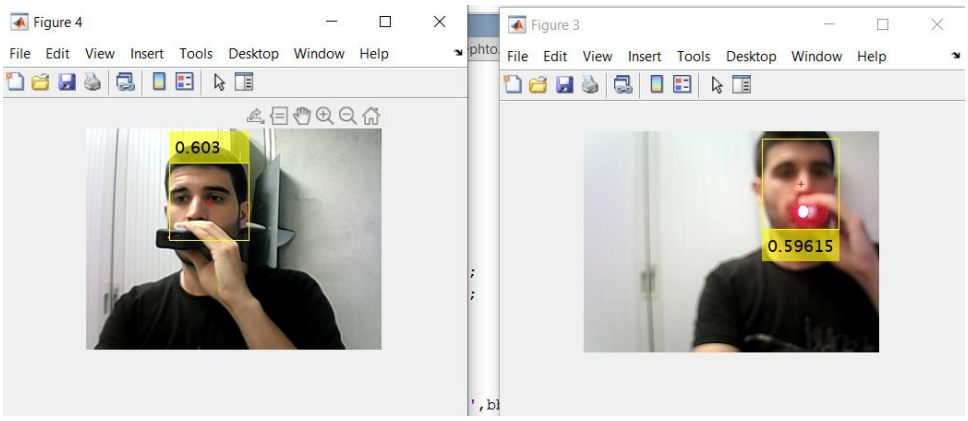

Ilustración 45. Frames de testeo de precisión de la aplicación usando MTCNN.

Tras realizar varias medidas con el medidor laser y comprobar los resultados en el cálculo de distancia, se puede afirmar que haciendo uso del método de clasificación de Adaboost y KLT la precisión es de ±4 cm, además en ésta caso en algunos momentos se realizan detecciones erróneas que falsean totalmente el resultado, haciendo la aplicación muy poco robusta. Mientras que usando el método de detección basado en MTCNN la precisión aumenta llegando a menos de ±2 cm y manteniendo una detección más robusta y fiable.

Los peores resultados obtenidos por Adaboost y KLT se deben principalmente a que la detección obtenida no es precisa y a que además es variante en todo momento, independientemente de que el usuario se encuentre quieto. Este hecho perjudica el cálculo de distancia entre usuario y cámaras obteniendo un resultado menos preciso. Por otro lado, cabe tener en cuenta que pese a la menor precisión los tiempos de procesado son mucho mejores que MTCNN, llegando a conseguir el doble de *frames* por segundo. En concreto la media de tiempo por *frame* de MTCNN es de 0.1550s y la de Adaboost de 0.085s.

7 Aplicación de *tracking* mediante el uso de una sola cámara webcam

En este último punto se va a describir y analizar la implementación y resultados de una aplicación de tracking que siga a los usuarios de la imagen y devuelva sus coordenadas 3D en el espacio real de forma similar que, en puntos anteriores, pero esta vez haciendo uso de una sola cámara webcam.

El principal problema es que solo se tiene una perspectiva, por lo que no se podrá llevar a cabo técnicas de triangulación. Para el cálculo de la distancia será necesario utilizar otro método que no requiera de una segunda perspectiva, por lo que el método elegido es el conocido como "similitud de triángulos" el cual será explicado posteriormente y se basa en el conocimiento del tamaño del objeto a seguir, así como su distancia inicial a la cámara.

Los pasos para la implementación serán los siguientes:

- Calibración de la cámara, pero esta vez no con la intención de obtener los parámetros extrínsecos e intrínsecos, si no para corregir las posibles distorsiones introducidas por la cámara.
- Detección de rostros, para lo que se volverán a utilizar y comparar los métodos de Adaboost con KLT y MTCNN.
- Determinación del tamaño del objeto a seguir y su distancia inicial a la cámara.
- Cálculo de las coordenadas 3D del usuario en el mundo real a partir de las coordenadas de la imagen.

Tras ver la implementación de la aplicación se analizarán los resultados obtenidos para de esta manera poder comparar posteriormente con los otros métodos vistos.

7.1 Calibración. Corrección de distorsiones.

Debido al efecto provocado por las lentes de las cámaras y su comportamiento con los rayos de luz, se producen distorsiones en la focalización y por tanto en la reproducción del mundo reflejada en las imágenes obtenidas. Dentro de las distorsiones posibles las dos más comunes son las de barril y de acerico, las cuales podemos ver representadas en la siguiente ilustración.

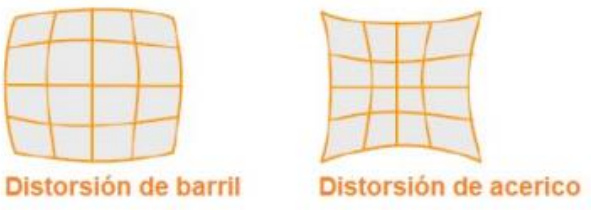

Ilustración 46. Ejemplos de distorsión.

Para llevar a cabo la calibración y corrección de la imagen será necesario hacer uso de un patrón de calibración igual que se vio en el punto anterior. Para no repetir el patrón de calibración en este apartado se utilizará una rejilla de 11x11 puntos perfectamente medidos y a una distancia de 3 cm entre sí. Una vez tomadas las imágenes del patrón se podrá apreciar como la separación y distribución de los puntos ha variado con respecto a la imagen originar, de manera que se podrá observar la distorsión introducida.

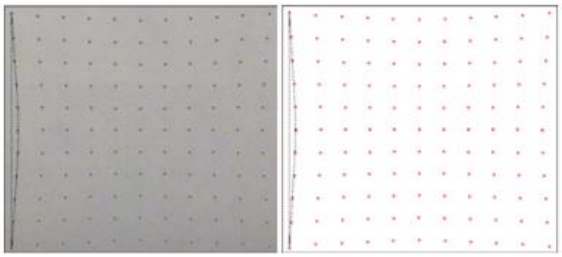

Ilustración 47. Resultado de la corrección de distorsiones.

Para llevar a cabo la calibración, la aplicación *Camera calibrator* que ofrece el *toolbox* de visión por computador en Matlab. Haciendo uso de la aplicación se cargarán las imágenes obtenidas, las cuales deben ser entre 10 o 20 para un correcto funcionamiento. A continuación, tras hacer uso de la opción de calibrado y comprobado el error de reproyección para asegurarse de un correcto calibrado, se obtendrán los parámetros intrínsecos y extrínsecos los cuales posteriormente se utilizarán para relazar la corrección de distorsiones.

La corrección de las distorsiones se hará aprovechando la función *undistortImage*, la cual pasando como argumento la imagen distorsionada y los parámetros de calibración devolverá la imagen corregida y libre de distorsiones.

```
[J,newOrigin] = undistortImage(I,cameraParams)
```

Éste no es un paso indispensable como podía suceder en la aplicación anterior, ya que se requerían los parámetros de calibración, pero mejorará el funcionamiento y la precisión del modelo.

7.2 Cálculo de coordenadas 3D

Con la cámara calibrada se puede empezar el desarrollo de la aplicación de tracking. Como en los puntos anteriores el primer paso será detectar los usuarios a seguir en la sala, para ello se emplearán las técnicas de reconocimiento facial que hemos visto previamente: Adaboost con KLT y MTCNN basado en una red neuronal. Después de ver la implementación de la aplicación, se analizarán los diferentes resultados obtenidos con ambos métodos.

Teniendo en cuenta que los usuarios ya han sido detectados en la imagen, se calculará las coordenadas 3D de éstos en la sala. En este caso al contar solo con una cámara, no será posible hacer uso de técnicas de triangulación para solventar este problema, por lo que a continuación se explicará el método seguido para solventar la falta de cámaras.

La técnica utilizada para el cálculo de distancia entre usuario y cámara ha sido la similitud de triángulos, según la cual, conociendo la anchura W del objeto a seguir y colocándolo a una distancia

inicial también conocida (D), se podrá calcular la distancia focal de la cámara (F). El último dato necesario para el calcula de la distancia focal es el tamaño en pixeles del objeto (P), el cual no es necesario conocer previamente ya que lo podremos ir calculando a tiempo real a partir del *bounding box* obtenido tras la detección.

$$F = \frac{P\,x\,D}{W}$$

La distancia focal es la distancia entre lente y sensor, en principio, esta se mantiene fija ya que al variarla estaríamos variando el zoom de la cámara. En definitiva, la distancia focal es el factor que determinará la escala y perspectiva con la que se verá en la imagen la representación de la escena en el dominio 3D del mundo real.

Después de calcular la distancia focal como F, y conociendo la distancia inicial y tamaño real del objeto, este se moverá con respecto a su posición inicial. Teniendo en cuenta la que la distancia focal se mantiene fija, se puede despejar de la ecuación anterior la nueva distancia del objeto a la cámara.

$$D = \frac{W\,x\,F}{P}$$

Para comprobar el funcionamiento del algoritmo, se ha implementado en Matlab un script que haciendo uso de los detectores faciales. Para implementar las fórmulas vistas es necesario conocer el tamaño del objeto y la distancia inicial, por lo tras medir mi cabeza (16 cm) y colocarme a una distancia de 70 cm al comenzar la aplicación se puede corroborar la precisión del modelo. Para comprobar la precisión se utilizará un medidor laser con el que se tomarán diferentes medidas que se contrastarán con los resultados obtenidos.

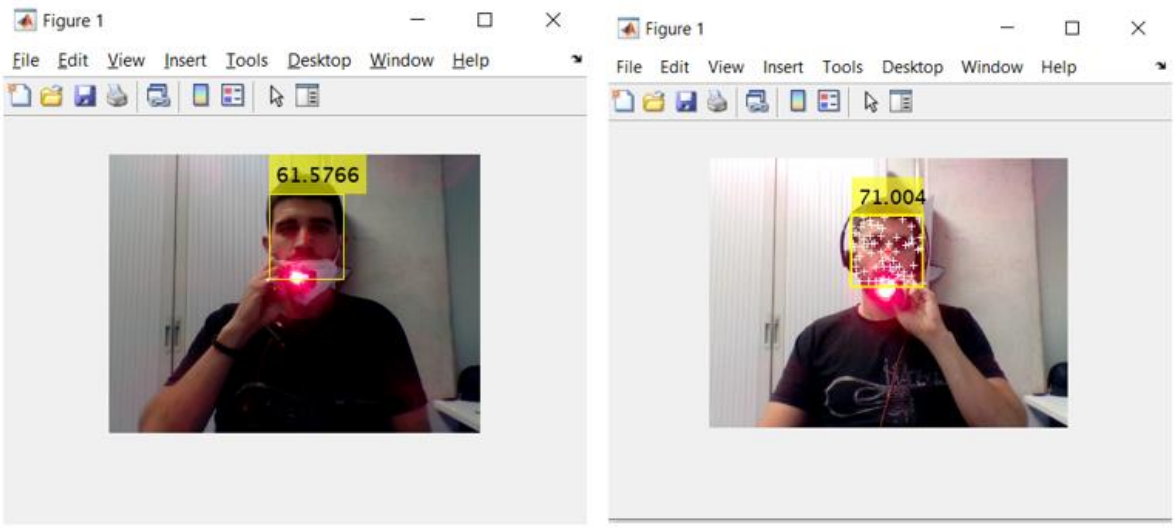

Ilustración 48. Resultado aplicación de tracking usando dos cámaras.

Tabla 5. Resultados de precisión con ambos amos métodos de detección.

	MTCNN	**Adaboos + KLT**
Medida 1	+ 2 cm	+ 4 cm
Medida 2	+ 3 cm	+ 6 cm
Medida 3	+ 3 cm	+ 4 cm
Medida 4	+ 4 cm	+ 6 cm
Medida 5	+ 5 cm	+ 6 cm
Medida 6	+ 3 cm	+ 5 cm
Medida 7	+ 5 cm	+ 5 cm
Medida 8	+ 4 cm	+ 7 cm
Medida 9	+ 3 cm	+ 6 cm
Medida 10	+ 4 cm	+ 5 cm

Tras realizar la prueba de precisión los resultados son los esperados. En el caso de utilizar el método de detección facial de MTCNN la precisión es más elevada dando como media + 3,6 cm con respecto a distancia real, mientras que el método de Adaboost con KLT ofrece un peor comportamiento, dando como resultado una media de + 5,6 cm. A parte de una peor precisión, en el segundo método la detección en algunos frames se pierde. Los resultados son los esperados ya que con Adaboost el "boundig box" no se ajusta del todo al rostro del usuario. En cuanto a los tiempos de ejecución, la aplicación se demora unos 0.024 s de media cuando se hace uso de Adaboost y 0.096s cuando se utiliza MTCNN como detector, teniendo un mayor tiempo de procesado el segundo método debido al coste computacional introducido con la red neuronal.

En conclusión, la aplicación de tracking con una sola cámara funciona correctamente, consiguiendo unos resultados muy similares a la versión con dos cámaras y con un tiempo de cómputo mucho más reducido, ya que no se tienen que gestionar los recursos de dos cámaras y de dos detectores. El problema que aparece en esta aplicación es que se deben conocer los parámetros de tamaño real del objeto y de distancia inicial, en el siguiente apartado se verá cómo solucionar este problema.

7.3 Obtención de parámetros iniciales para cálculo de distancias

Las proporciones en la anatomía humana fueron una de las principales ramas científicas siglos atrás y en la actualidad existen un sinfín estudios estadísticos para corroborar todo tipo de teorías acerca de cómo se proporciona el cuerpo humano. En el siglo XV Leonardo DaVinci ya cimentó algunas de las teorías acerca de la proporción humana con el famoso hombre de Vitruvio, un famoso estudio donde, entre otras cosas, relaciona las medidas de las diferentes partes del cuerpo entre sí y demuestra que son submúltiples de la dimensión del cuerpo entero o de una de sus partes principales. Una de estas proporciones y en la que se va a basar el siguiente apartado es la afirmación de que el ser humano forma una cruz perfecta, es decir que la longitud de la envergadura (distancia de la punta de los dedos de una a mano a otra, con los brazos extendidos) es igual a la altura.

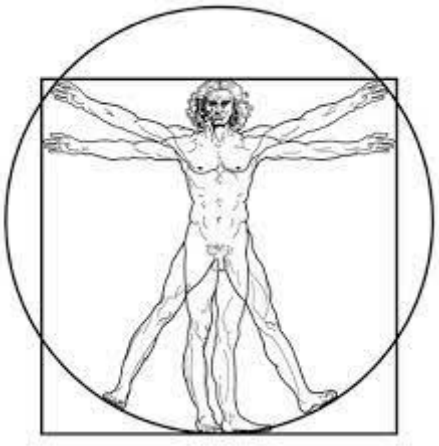

Ilustración 49. Hombre de Vitrubio.

En la actualidad existen diferentes estudios estadísticos que demuestran que este hecho es cierto en el 90 % de hombre adultos y en el caso de las mujeres en un 85 %. Basándonos en esta premisa y en la relación del pecho con respecto a la longitud de los brazos, se puede afirmar, en la mayoría de los casos, que la longitud del brazo de una persona es 0.38 veces la altura de ésta misma. Por otro lado, se ha profundizado en las dimensiones medias en la cabeza humana, en concreto de la anchura del rostro, entendiendo como anchura la distancia entre pómulos. Después de extraer información de diferentes fuentes, se ha podido establecer que la anchura facial en el caso de los hombres ronda una media de 158 mm mientras que en el caso de las mujeres la media es aproximadamente de 142 cm.

Existen técnicas que permiten calcular la distancia de un usuario a la cámara, una de ellas se implementa calibrando la cámara con un patrón de calibración para posteriormente usar el mismo patrón para comparar su tamaño con el del objeto y así obtener el tamaño de éste. En el apartado previo se ha analizado el proceso para obtener la distancia del objeto a la cámara de una forma más accesible para el usuario, teniendo la pega de la necesidad de conocer ciertos parámetros inicial previamente que serán el tamaño del objeto real y su posición inicial. El proceso de obtención de parámetros iniciales estará basado en las proporciones de la anatomía humana.

Con el objetivo de conocer la anchura del rostro del usuario y la distancia inicial a la que se encuentra de la cámara se han ideado una serie de preguntas iniciales que junto a los conocimientos de proporción anatómica vistos permitirán calcular dichos parámetros:

1. Se pregunta al usuario si es hombre o mujer, de esta manera se puede estimar la anchura del rostro basándose en las medias según género.
2. Se pregunta la estatura media del usuario para así mediante la regla de que el brazo es 0,38 veces la altura del usuario conocer la longitud del brazo.

3. Se pide al usuario que se coloque erguido y con el brazo estirado tocando la pantalla para asegurar que se encuentra a la distancia de su brazo.

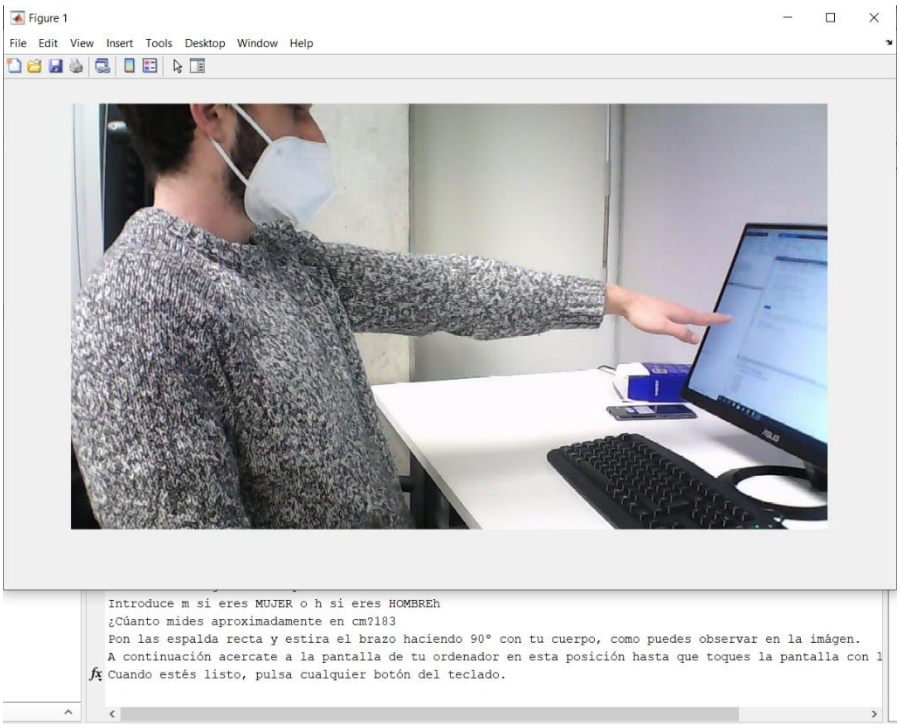

Ilustración 50. Ejemplo sistema de obtención de medidas del usuario.

Siguiendo estos pasos se podrán obtener los parámetros iniciales necesitados y de esta manera garantizar el funcionamiento de la aplicación sin necesidad de utilizar factores o herramientas externas. Las desventajas del empleo de esta técnica son obvias, para empezar los usuarios deberán ser adultos para que cumplan las reglas de proporcionalidad y anatomía vistas, por otro lado, la precisión de la aplicación variara en función de la proporcionalidad del usuario. En conclusión, a pesar del correcto funcionamiento de la aplicación, es poco robusta frente agentes externos.

8 Conclusiones

En este último punto se van a analizar las diferentes prestaciones de las tres aplicaciones de tracking. Tras analizar las ventajas y desventajas de cada método se podrá discernir cuál de ellos es mejor, aunque esto dependerá de la aplicación que se le vaya a dar. Los diferentes aspectos que se utilizarán para la comparación serán:

- Los *frames* por segundo a los que pueda funcionar la aplicación, lo cual dependerá de la capacidad y coste computacional de cada una.
- La precesión en cm de captar la posición en coordenadas 3D.
- Dificultad de implantación. Este aspecto refleja desde el punto de usuario la facilidad de montar la aplicación con el hardware necesario, para un uso cotidiano.
- Robustez a diferentes eventos, como la iluminación, movimiento, etc.. .
- Número máximo de usuarios capaces de captar.
- Precio del hardware necesario para le montaje.

Con la intención de resumir los datos de una forma más intuitiva se utilizará la siguiente tabla.

Tabla 6. Comparativa entre diferentes aplicaciones de tracking.

	Cámara 3D	**Una webcam**	**Dos webcams**
Frames por segundo	Entre 10 y 30 FPS	Ente 20 y 30 FPS	Entre 10 y 20 FPS
Precisión	± 1 cm	± 3 cm	± 5 cm
Dificultad implantación	Alta	Baja	Media
Robustez	Alta	Media	Baja
Nº máximo usuarios	8	3/4	3/4
Precio	150 €	Entre 20 € y 50 €	Entre 40 € y 100 €

Como se puede apreciar en la tabla hay que analizar los métodos desde diferentes puntos de vista. En cuanto a la velocidad de reproducción, cada cámara 3D tiene diferentes prestaciones, pero en concreto Orbbec Astra variará en función de los streams que utilices variando entre 10 y 30 FPS en función de la necesidad. Las webcams dependerán del coste computacional sobre todo del método de detección utilizada pudiendo oscilar entre los 20 y 30 fps usando una sola y 10 a 20 fps utilizando ambas con triangulación. La precisión es claramente más alta haciendo uso de la cámara 3D debido al sensor infrarrojo con el que cuenta.

Es difícil que un usuario cotidiano cuente con una cámara 3D, mientras que es mucho más fácil que cuente con un par de webcams para triangular y casi seguro que su pc portátil cuente con una webcam integrada, de ahí los resultados referentes a la dificultad de implantación.

Los cambios de iluminación, movimientos del usuario y demás factores afectan al rendimiento de la aplicación, en este caso la cámara 3D es la más robusta. Por otro lado, entre las aplicaciones que hacen uso de webcams, la que usa una sola se enfrenta además a la necesidad de que el usuario cumpla las reglas de proporcionalidad anatómica implementadas, por eso podemos decir que es la aplicación menos robusta.

El número máximo de usuarios será mayor en la cámara 3D debido al gran angular con el que cuenta y la precisión de las medidas, lo que puede ser un factor determinante en función de la necesidad. Por último, el precio de cada aplicación dependerá de las cámaras utilizadas siendo obviamente las más caras las 3D. Por otro lado, las webcams oscilan mucho de precio en función del modelo y prestaciones.

En conclusión, en este proyecto se resuelven la mayoría de las dudas existentes en cuanto a la cámara o métodos que utilizar para desarrollar una aplicación de tracking, además de servir como guía. En lo personal, me ha servido para aprender mucho sobre diferentes aspectos de software y hardware relacionados con la visión artificial.

9 Líneas futuras

La capacidad de realizar el seguimiento y obtener las coordenadas de usuarios en tiempo real se puede aplicar para el uso de diferentes labores. Uno de los sectores que más se puede beneficiar de este tipo de tecnología es el sector de la seguridad, donde se puede llevar un seguimiento de las personas que accedan a un establecimiento, casa o cualquier lugar privado que necesite de un control rutinario.

La tecnología de tracking no tiene que ir necesariamente ligada al seguimiento de personas, se podría tratar de entrenar los algoritmos de detección con la intención de detectar otro tipo de objetos con el fin de cubrir diferentes necesidades, algunas de ellas podrían ser:

- Detección de vehículos en el acceso a un parquin. En la actualidad existen parkings subterráneos que llevan un control de las plazas ocupadas, gracias a sistemas que hacen uso de sensores infrarrojos ubicados en cada una de las plazas disponibles. Mediante un sistema de tracking que realice el seguimiento de coches, se podría controlar la entrada y salida de coches en parkings exteriores, las plazas ocupadas, coches mal aparcados e incluso se podría almacenar la matrícula de los coches estacionados.
- Detección de animales para sector de la ganadería. Otra aplicación que se le podría dar a la tecnología de tracking sería la detección de diferentes tipos de animales para el control de la ganadería. De esta forma, mediante el uso de drones que cuenten con sistemas de visión y una capacidad de cómputo suficiente, se podría controlar la cantidad de animales en cada rebaño, la ubicación de éstos o incluso si sufren algún tipo de peligro.
- Detección de rostros sin mascarilla. En la actualidad y debido a la reciente pandemia, en la mayoría de los locales es obligatorio el uso de mascarilla por motivos sanitarios. Una aplicación de tracking podría realizar el seguimiento de los clientes que acceden a un establecimiento, pero además se podría implementar la opción de detectar si el cliente lleva puesto mascarilla. Debido a las necesidades introducidas por la pandemia, ha surgido la nueva labor de controlar el acceso de clientes a los establecimientos, para lo que se necesita a un trabajador exclusivamente dedicado a esta labor. Mediante el uso de la tecnología descrita se podría ahorrar esta labor, ayudando a optimizar las obligaciones de los trabajadores.

A parte de aplicaciones relacionadas con la seguridad, en el sector del entretenimiento se podría aplicar la tecnología para diseñar experiencias únicas y con una total interacción del consumidor. En concreto, el uso de cámaras 3D facilita la detección de gestos y movimientos de diferentes articulaciones que se podrían aplicar en el diseño de las experiencias. Algunas de las aplicaciones podrían ser:

- Desarrollo de videojuegos. Actualmente ya existen diferentes compañías relacionadas con los videojuegos que han implementado este tipo de tecnología en sus consolas. De hecho, Kinect salió a la luz relacionando su uso con el de la XBOX 360. A pesar de que ya existen videojuegos que detectan los movimientos del usuario, esta técnica se podría seguir desarrollando para hacer juegos cada vez más inmersivos.
- Experiencias culturales. De la mano del arte, surgen un sinfín de experiencias que pretenden sumir el consumidor en la mente del artista. Mediante el seguimiento de los usuarios se podrían implementar diferentes eventos relacionados con la obra, que surgiesen cuando el usuario se acercase a alguna zona o si éste realizase algún gesto en concreto.

- En otro sector donde se podría implementar la detección de usuarios y gestos podría ser en domótica. Algunas de las utilidades que se le podría dar al seguimiento de usuarios y detección de gestos podrían ser:
    - Seguir a los usuarios por la casa, para de esta forma apagar los sistemas de iluminación y ventilación en las habitaciones en las que no haga falta, ya que no hay ningún usuario. De esta forma se podría ahorrar de forma dinámica en el consumo eléctrico.
    - Control de diferentes utilidades mediante gestos. Por ejemplo, la subida y bajada de persianas, control de volumen de altavoces o apagado y encendido de diferentes dispositivos.

Al margen de las aplicaciones que se le puede dar al sistema de tracking, en un futuro sería interesante seguir trabajando en la optimización de los procesos de detección y cálculo de coordenadas, para así conseguir mejores tiempos de cómputo y sobre todo una mayor precisión en el caculo de la posición del usuario.

10 Presupuesto

PRESUPUESTO DEL PROYECTO					
Nº de orden	Unidad	Descripción de las unidades	Medición	Precio unidad	Importe
01		**ESTUDIO EXPERIMENTAL**			
01.01		**Licencias de software utilizados para el cálculo**			
01.01.01	%	MATLAB	0,26	300,00 €	78,00 €
01.01.02	%	MATLAB COMPUTER VISION TOOLBOX	0,26	500,00 €	130,00 €
01.01.03	%	ORBBEC HUMAN TRACKING	0,26	70,00 €	18,20 €
01.01.04	%	MICROSOFT OFFICE 365 PROFESSIONAL	0,26	59,00 €	15,34 €
01.01.05	%	VISUAL STUDIO	0,26	540,00 €	140,40 €
		total capítulo 01.01			**381,94 €**
01.02		**Materiales utilizados en la computación**			
01.02.02	%	ORDENADOR AMD RYZEN , 16 GB RAM, 2060 GTX	0,26	1.000,00 €	260,00 €
		total capítulo 01.02			**260,00 €**
		TOTAL CAPÍTULO 01			**641,94 €**
02		**MATERIAL EMPLEADO PARA EL ESTUDIO**			
02.01		**Hardware**			
01.02.02	%	AMORTIZACIÓN CÁMARA 3D ORBBEC ASTRA	0,26	150,00 €	39,00 €
01.02.03	%	AMORTIZACIÓN WEBCAM PAPALOOK HD	0,26	60,00 €	15,60 €
01.02.04	%	AMORTIZACIÓN WEBCAM PAPALOOK HD	0,26	60,00 €	15,60 €
01.02.05	%	AMORTIZACIÓN LÁSER	0,26	40,00 €	10,40 €
		total capítulo 02.01			**80,60 €**
02.02	u	**Material fungibles**			
02.02.01		PAPEL, FOTOCOPIAS, BOLÍGRAFOS, ETC.	1	20,00 €	20,00 €
		total capítulo 02.02			**20,00 €**
		TOTAL CAPÍTULO 02			**100,60 €**
03		**HONORARIOS**			
03.01	horas	INGENIERO DE TELECOMUNICACIÓNES	360	32,00 €	11.520,00 €
		TOTAL CAPÍTULO 03			**11.520,00 €**

RESUMEN DEL PRESUPUESTO		
Nº de orden	**Descripción de las unidades**	**Importe**
01	ESTUDIO EXPERIMENTAL	641,94 €
02	MATERIAL EMPLEADO PARA EL ESTUDIO	100,60 €
03	HONORARIOS	11.520,00 €

PRESUPUESTO DE EJECUCIÓN MATERIAL	12.262,54 €
12% Gastos Generales	1.594,13 €
6% Beneficio Industrial	735,75 €
PRESUPUESTO BRUTO	14.592,42 €
21% I.V.A	3.064,41 €
PRESUPUESTO LÍQUIDO	17.656,83 €

Suma el presente presupuesto la cantidad de:
DIECISIETE MIL SEISCIENTEOS CINCUENTA Y SEIS CON OCHENTA Y TRES

11 Bibliografía

[1]. Zhang, Z., 2021. Microsoft Kinect Sensor and Its Effect. Microsoft Research.

[2]. Ehara, Y., Fujimoto, H., Miyazaki, S., Tanaka, S. and Yamamoto, S., 2021. Comparison of the performance of 3D camera systems.

[3]. Yeung, L., Yang, Z., Cheng, K., Du, D. and Tong, R., 2021. Effects of camera viewing angles on tracking kinematic gait patterns using Azure Kinect, Kinect v2 and Orbbec Astra Pro v2.

[4]. Ieeexplore.ieee.org. 2021. Interchangeability of Kinect and Orbbec Sensors for Gesture Recognition.

[5]. Giancola, S., Valenti, M. and Sala, R., 2021. Metrological Qualification of the Intel D400™ Active Stereoscopy Cameras.

[6]. Ieeexplore.ieee.org. 2021. Joint Face Detection and Facial Expression Recognition with MTCNN.

[7]. H, W., H, P. and S, R., 2021. Human body proportions explained on the basis of biomechanical principles.

[8]. Boudjit, Kamel & Ramzan, Naeem, 2021. Human detection based on deep learning YOLO-v2 for real-time UAV applications. Journal of Experimental & Theoretical Artificial Intelligence. 1-18. 10.1080/0952813X.2021.1907793.

[9]. Sabato, Alessandro & Valente, Nicholas & Niezrecki, Christopher, 2020. Development of a Camera Localization System for Three-Dimensional Digital Image Correlation Camera Triangulation. IEEE Sensors Journal. PP. 1-1. 10.1109/JSEN.2020.2997774.

[10]. Joint Face detection and Facial Expression Recognition with MTCNN, Xiang y Zhu. 2017

[11]. Human detection based on deep learning YOLO-v2 for real-time UAV applications, Kamel Boudjit. 2021

[12]. Real-Time Human Pose Estimation on a Smart Walker using Convolutional Neural Networks, Manuel Palermo. 2018

[13].Técnica Flexible de Zhang para el cálculo de parámetros intrínsecos y extrínsecos en el ajuste de dietas en cultivos de Oreochromis Sp, Edgar Matallana. 2017

[14]. Auto-calibración de redes de cámaras, Carlos Fernandez Herrero. 2019

[15].A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In Proceedings of the 6th ACM international conference on Image and video retrieval, pages 401–408. ACM, 2007

[16].] D. Chen, S. Ren, Y. Wei, X. Cao, and J. Sun. Joint cascade face detection and alignment. In European Conference on Computer Vision (ECCV), 2014.

[17]. Z. Yu and C. Zhang, "Image based static facial expression recognition with multiple deep network learning," in ACM International Conference on Multimodal Interaction (MMI), 2015, pp. 435–442.

[18].C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on, 2015

[19].H.-M. Hsu, T.-W. Huang, G. Wang, J. Cai, Z. Lei, and J.-N. Hwang. Multi-camera tracking of vehicles based on deep features re-id and trajectory-based camera link models. In IEEE Conf. Comput. Vis. Pattern Recog. Worksh., pages 416–424, 2019.

[20].P. Khorramshahi, N. Peri, J.-c. Chen, and R. Chellappa. The devil is in the details: Self-supervised attention for vehicle re-identification. In Eur. Conf. Comput. Vis., pages 369–386, 2020.

[21].M. Liu, R. Wang, Z. Huang, S. Shan, and X. Chen. Partial least squares regression on grassmannian manifold for emotion recognition. In Proceedings of the 15th ACM on International conference on multimodal interaction, pages 525–530. ACM, 2013.

[22].A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[23]. R. T. Ionescu, M. Popescu, and C. Grozea. Local learning to improve bag of visual words model for facial expression recognition. In Workshop on Challenges in Representation Learning, ICML, 2013.

[24].J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research, 12:2121–2159, 2011.

[25]. A. Dhall, R. Goecke, J. Joshi, K. Sikka, and T. Gedeon. Emotion recognition in the wild challenge 2014: Baseline, data and protocol. In Proceedings of the 16th International Conference on Multimodal Interaction, pages 461–466. ACM, 2014.

[26].M. S. Bartlett, G. C. Littlewort, M. G. Frank, C. Lainscsek, I. R. Fasel, and J. R. Movellan. Automatic recognition of facial actions in spontaneous expressions. Journal of multimedia, 1(6):22–35, 2006

[27].Y. Qian, L. Yu, W. Liu, and A. G. Hauptmann. Electricity: An efficient multi-camera vehicle tracking system for intelligent city. In IEEE Conf. Comput. Vis. Pattern Recog. Worksh., pages 588–589, 2020.

[28].T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft coco: Com- ´ mon objects in context. In Eur. Conf. Comput. Vis., pages 740–755, 2014.

[29].X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. In Adv. Neural Inform. Process. Syst., pages 21002–21012, 2020.

[30].X. Pan, P. Luo, J. Shi, and X. Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In Eur. Conf. Comput. Vis., pages 464–479, 2018.

[31].B. Pang, Y. Li, Y. Zhang, M. Li, and C. Lu. Tubetk: Adopting tubes to track multi-object in a one-step training model. In IEEE Conf. Comput. Vis. Pattern Recog., pages 6308–6318, 2020.

[32]. Zhang, Kaipeng, et al. "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks." IEEE Signal Processing Letters 23.99(2016):1499-1503.

[33].E. Bochinski, T. Senst, and T. Sikora. Extending iou based multi-object tracking by visual information. In IEEE Int. Conf. Adv. Video Sign. Surv., 2018.

[34].S. He, H. Luo, W. Chen, M. Zhang, Y. Zhang, F. Wang, H. Li, and W. Jiang. Multi-domain learning and identity mining for vehicle re-identification. In IEEE Conf. Comput. Vis. Pattern Recog. Worksh., pages 582–583, 2020.

[35].E. Bochinski, V. Eiselein, and T. Sikora. High-speed tracking-by-detection without using image information. In IEEE Int. Conf. Adv. Video Sign. Surv., 2017.

[36]. Pramerdorfer, Christopher, and M. Kampel. "Facial Expression Recognition using Convolutional Neural Networks: State of the Art." (2016).

[37].P. Bergmann, T. Meinhardt, and L. Leal-Taixe. Tracking ´ without bells and whistles. In Int. Conf. Comput. Vis., pages 941–951, 2019.

[38].B. Pang, Y. Li, Y. Zhang, M. Li, and C. Lu. Tubetk: Adopting tubes to track multi-object in a one-step training model. In IEEE Conf. Comput. Vis. Pattern Recog., pages 6308–6318, 2020

[39]. Orbbec Astra Guide. https://orbbec3d.com/develop/

[40]. Intel RealSense D400 Series Catalog. https://www.framos.com/en/product-catalog/3d/3d-cameras/

[41]. MatlabComputerVisionToolbox Documentation. https://www.mathworks.com/help/vision/

[42]. Curso Fundamentos Ópticos, http://fotoigual.com/3-distancia-enfoque-tamano-la-imagen/