ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE MÁLAGA

TRABAJO FIN DE MÁSTER

RECONOCIMIENTO DE PERSONAS PARA SEGUIMIENTO MEDIANTE ROBOTS MÓVILES

MÁSTER EN INGENIERÍA MECATRÓNICA

Málaga, 2022

Alberto Aguayo García

UNIVERSIDAD DE MÁLAGA

Universidad de Málaga
Escuela de Ingenierías Industriales

Don Ricardo Vázquez Martin y Don Jesús Fernández Lozano, tutor y cotutor del Trabajo Fin de Máster titulado: Reconocimiento de personas para seguimiento mediante robots móviles, que presenta Alberto Aguayo García, autorizan su presentación para defensa y evaluación en la Escuela de Ingenierías Industriales de Málaga.

Málaga, noviembre de 2022

El alumno: Los tutores:

Alberto Aguayo García Ricardo Vázquez Martin

Jesús Fernández Lozano

RESUMEN

Durante la ejecución de labores de rescate, la efectividad y rápidez juegan un papel crucial, ya que el tiempo que transcurre hasta que los heridos pueden ser atendidos en este tipo de maniobras puede ayudar a disminuir la gravedad del estado de los pacientes. En este trabajo se desarrolla un sistema de reconocimiento de personas para seguimiento mediante robots móviles. El objetivo es reconocer a una persona y localizar su distancia y orientación respecto al robot, de manera que éste pueda seguirlo. El desarrollo busca mejorar las capacidades de los sistemas actuales, cuyas limitaciones en misiones de rescate, identificadas durante ejercicios realistas, sirven como punto de partida para el desarrollo de este trabajo.

ABSTRACT

During rescue work, effectiveness and quickness are critical, as the time elapsed until wounded are attended in this type of manoevre can help to decrease the state gravity of the patient. In this work, a human recognition system is developed for mobile robot tracking. The goal is to recognize a person and locate their distance and orientation from the robot so that the robot can follow him. The development seeks to improve the capabilities of current systems, whose limitations in rescue missions, identified during realistic exercises, are used as a starting point for the development of this work.

Con todo mi cariño y dedicación,

A mis compañeros.

A mi familia.

A mis profesores.

Reconocimiento de personas para seguimiento mediante robots móviles.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN.

En el primer capítulo de la memoria de este Trabajo Fin de Máster se introducen las razones que motivaron a realizar dicho Trabajo, los objetivos del mismo y el estado del arte de sistemas de seguimiento de personas dentro de labores de rescate.

1.1. Motivación.

En la actualidad, estamos en continua interacción con máquinas que efectúan todo tipo de trabajos y servicios. Dichas máquinas van mejorando y desarrollando mecanismos que les otorguen mayor autonomía, gracias a grandes investigaciones llevadas a cabo por los humanos. Los campos de aplicación van ampliándose según mejora la tecnología, y así, herramientas como *Machine Learning* se emplean no solo en el ámbito ingenieril, sino también en otros sectores, como el sector de la salud, educativo y en el mundo de las finanzas. Tanto es así que el 69% de los estadounidenses prefieren comunicarse con los distribuidores a través de los conocidos *chatbots* por sus rápidas respuestas (Sweezey, 2019).

Centrándonos en el tema de este Trabajo Fin de Máster, la detección y seguimiento de personas es una aplicación frecuente dentro del *Machine Learning*. Hoy en día, la detección de personas se ha convertido en una herramienta que está presente en muchos sistemas de vídeo, siendo una de las soluciones más importantes de la visión por computador en la actualidad. Encontramos muchas aplicaciones útiles, como por ejemplo la detección de peatones en la conducción autónoma de automóviles (Zhenjiang, Kunfeng, Li, & Fei-Yue, 2006).

El uso de la visión por computador está ampliamente utilizado en diversos sectores económicos. La Figura 1, extraída del artículo (Chung, Butterfield, & Murphy, 2021) muestra que, a pesar de que el sector dominante sea el de investigación, se emplea en otros ámbitos como el industrial, eléctrico o sector público.

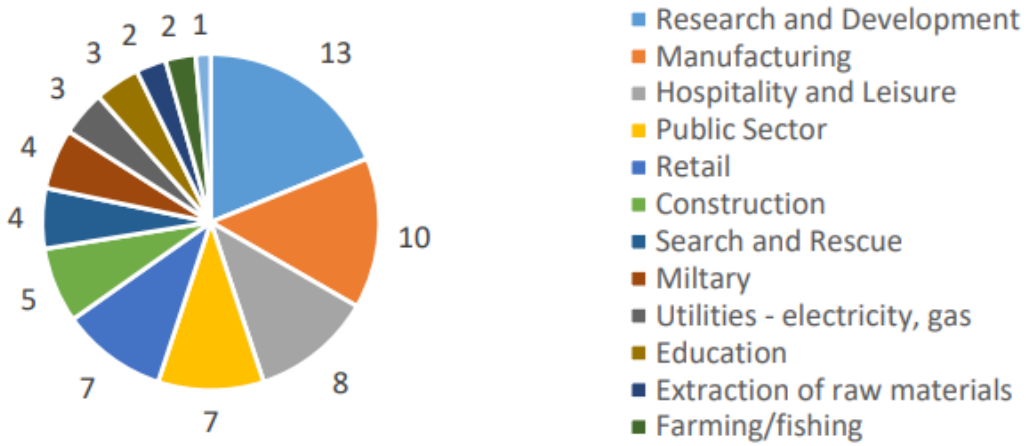


Figura 1: Aplicaciones que utilizan técnicas de visión por computador (Babenko, B., Yang, M.-H., & Belongie).

El ámbito de la seguridad y las emergencias constituye también un importante campo de aplicación para diversos tipos de tecnología. Acercándonos al tema de este Trabajo, la robótica cuenta con un gran potencial en este campo. Pueden citarse diversas aplicaciones, pero en los últimos tiempos destaca la evacuación de heridos. Esta tarea es peligrosa en sí misma, y requiere un número elevado de personal para llevar una camilla con seguridad (ver Figura 2).

Figura 2: Equipo de la UME realizando labores de rescate (Ministerio de Defensa, 2011).

Asimismo, la velocidad de evacuación aumenta las posibilidades de curación de los heridos. En este problema, puede usarse un robot para trasladar a los heridos, reduciendo la cantidad de personal necesaria. Para guiar el robot existen diversos métodos, como la teleoperación a la vista o remota (Mandow, Serón, Pastor, & García-Cerezo, 2020). Pero también es posible usar un sistema de reconocimiento para que el robot siga a una persona, que actúa como guía. Existen algunos sistemas comerciales que permiten lograr esto, pero cuentan con diversas deficiencias, que el Grupo de Robótica y Mecatrónica ha encontrado en el transcurso de diversos ejercicios realistas de evacuación de heridos (Bravo-Arrabal, Toscano-Moreno, Fernandez-Lozano, Mandow, Gomwz-Ruiz, & García-Cerezo, 2021).

Sobre esta base, este Trabajo persigue desarrollar un sistema de reconocimiento y seguimiento de personas que mejore algunas características respecto a los comerciales.

1.2. Objetivos.

El objetivo principal de este Trabajo Fin de Máster es desarrollar un sistema de reconocimiento de personas para seguimiento mediante robots móviles, en el contexto de misiones de rescate. El objetivo es reconocer a una persona y localizar su distancia y orientación respecto al robot, de manera que éste pueda seguirlo. En lugar de sensores activos, como el LIDAR, se emplearán sensores pasivos como las cámaras. El desarrollo busca mejorar las capacidades de los sistemas actuales, cuyas limitaciones en misiones de rescate, identificadas durante ejercicios realistas, sirven como punto de partida para el desarrollo de este Trabajo. Por último, se buscará la solución más eficiente a nivel computacional, con tal de no utilizar muchos recursos de computación a bordo del vehículo, ya que otras tareas más costosas necesitan de más recursos.

1.3. Seguimiento de personas en vehículos.

En este apartado se va a estudiar el estado del arte en sistemas de seguimiento de personas dentro de labores de rescate, así como el contexto en el cual se ubica este Trabajo Fin de Máster.

- **Estado del arte.**

Existen varios sistemas relativos al seguimiento, ya sea de personas u objetos, dentro del campo militar y de maniobra.

- ***Multi Follow Me* Convoy.**

El primero de ellos es la implementación de un convoy de vehículos autónomos utilizando la técnica *multi-Follow Me* (VEDECOM, 2021). De esta manera, existe un vehículo líder al cual siguen el resto de vehículos (ver Figura 3). Cabe destacar que el seguimiento del convoy no se realiza de forma tradicional, donde todos los vehículos replican la misma trayectoria del vehículo líder a través de un seguimiento GPS, por ejemplo. En este caso, el seguimiento emplea técnicas de visión para el sistema *Follow Me*, de tal forma que cada vehículo adapta su trayectoria y velocidad propia para formar el convoy.

Figura 3*: Multi follow-me* en maniobras militares (VEDECOM, 2021).

La peculiaridad de este sistema es que cada vehículo adapta su trayectoria y velocidad en función del resto, para formar así el convoy.

- **Rheinmetall Mission Master (UGV).**

Actualmente existen vehículos terrestres no tripulados, en inglés denominados *Unmanned Ground Vechicles* (UGV), utilizados en diversas aplicaciones militares cuando es peligroso o inconveniente la presencia humana. Refiriéndonos a las labores de rescate, existen UGV para operaciones de mantenimiento de la paz, los cuales permiten facilitar dichas labores. Fueron utilizados por primera vez tras los atentados en Nueva York del 11 de septiembre de 2001 (Carlson & Murphy, 2005).

Existe un vehículo denominado *Rheinmetall Mission Master* (Rheinmetall), capaz de evacuar a personas heridas yendo al punto de evacuación y llevando los cuerpos al puesto de atención de forma autónoma (ver Figura 4). Cuenta con un sistema *follow-me* basado en LIDAR. El *Mission Master* es una versión adaptada del *Argo Rover J8*, que es un UGV que ha utilizado el Grupo de Robótica y Mecatrónica en diversos ejercicios realistas en los últimos años.

Figura 4: Rheinmetall Mission Master UGV (Rheinmetall).

- **Contexto**.

El grupo de investigación de Robótica y Mecatrónica de la Universidad de Málaga, integrado dentro del Departamento de Ingeniería de Sistemas y Automática (ISA) lleva trabajando desde el año 1989 en el desarrollo tecnológico de robots para diversas aplicaciones, entre las que se incluyen las operaciones de emergencia y rescate. Todos estos avances son presentados en las Jornadas Internacionales sobre Seguridad, Emergencias y Catástrofes, organizadas por la Cátedra de Seguridad, Emergencias y Catástrofes de la Universidad de Málaga, donde se llevan a cabo diversos ejercicios en entornos realistas.

Figura 5: Jornadas de emergencia UMA (Morales, Vázquez-Martín, & Mandow, 2021).

Una de las operaciones que se realizan y en la que este Trabajo se centra es en el transporte de personas en un vehículo no tripulado, llamado Rover J8 (ver Figura 5), del cual se entrará más en detalle en el siguiente capítulo. Este robot móvil tiene la capacidad de seguir a la persona encargada del rescate. De esta manera, el transporte se realiza de una manera más segura y rápida. Además, se elimina el problema de tener que emplear un gran esfuerzo físico por parte del equipo de rescate, cuya misión principal ahora será la de llegar a un puesto seguro donde pueda ser atendido.

La Figura 5 y otras imágenes mostradas en esta memoria han sido obtenidas por el Grupo de Robótica y Mecatrónica de la Universidad de Málaga, incluyendo además el *dataset* UMA-SAR (Morales, Vázquez-Martín, & Mandow, 2021). Este *dataset* ha sido obtenido durante las jornadas de emergencias celebradas en la UMA durante varios años y contiene una colección de datos multimodales en entornos de búsqueda y rescate, en inglés *Search And Rescue* (*SAR*), a través de vehículos terrestres tripulados.

Las pruebas realizadas relativas a la misión de rescate de este año (Toscano, y otros, 2022) se llevaron a cabo asemejándose lo máximo posible a unas condiciones reales de rescate. Este ejercicio se realizó en conjunto por parte del equipo de investigadores y el Pelotón Sanitario del Tercio Alejandro Farnesio 4 de la Legión (Ejército de Tierra) junto con el Ala 78 (Ejército de Aire y del Espacio). La misión comienza en un centro de control, *forward control center* (FCC) cerca de la Escuela de Ingenierías Industriales de la Universidad de Málaga (ver Figura 6), donde será el punto de partida del vehículo UGV J8. En otra ubicación cercana, existe una zona caliente donde se ha recreado un acto terrorista. El vehículo se mantiene a la espera, hasta que en el centro de control se recibe una petición de evacuación. La arquitectura del sistema permite que la petición incluya las coordenadas GPS del lugar donde se necesita el robot. El robot puede acudir a ese punto teleoperado, autónomamente, o utilizando el modo follow-me. En cualquier caso, una vez ha cargado los heridos, el retorno hacia el puesto de atención sanitaria se realiza preferentemente con el modo follow-me (ver Figura 7). No obstante, existen otras circunstancias en las que se emplea este modo. Por ejemplo, cuando el terreno es desconocido o difícil.

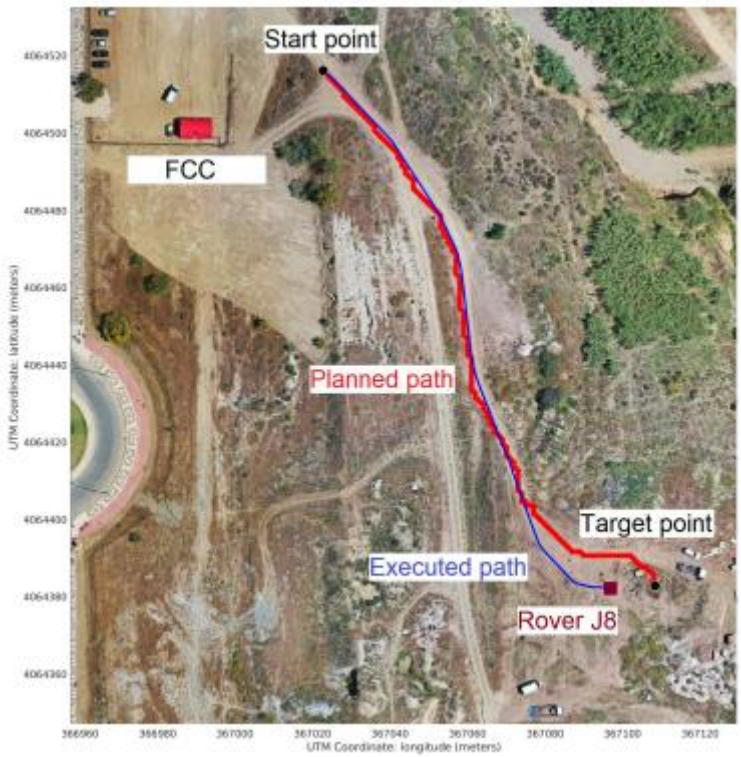


Figura 6: Trayectoria descrita por el vehículo Rover J8 (Toscano, y otros, 2022).

Además del modo *follow-me*, se implementaron otras características que se probaron en esta jornada, como por ejemplo una aplicación móvil para llamar al vehículo y que acuda al punto de rescate, diversos modos de teleoperación utilizando redes de comunicaciones 5G, o la integración mediante ROS de diversos medios heterogéneos. Pero puesto que no son relevantes para este Trabajo, no se entrará en sus detalles.

Figura 7: Secuencia de acciones en el ejercicio de evacuación y rescate (Toscano, y otros, 2022).

- **Problemas.**

Al término de la jornada, la valoración general fue bastante positiva. Sin embargo, hubo ciertos aspectos a mejorar, de los cuales algunos de ellos conciernen a este Trabajo Fin de Máster.

Tal y como se explicó anteriormente, el equipo de rescate empleó el modo *follow-me* para mover a los heridos desde el punto de rescate al puesto de socorro. Un defecto del sistema implementado es que no se desactivaba cuando el equipo de rescate cargaba a las personas heridas en el vehículo, haciendo que el Rover J8 hiciera movimientos inesperados, pudiendo poner en riesgo al equipo y a las víctimas.

Otro de los problemas, y del cual ha derivado principalmente este Trabajo, es que había momentos en los que el vehículo perdía la referencia de la persona que debía seguir, desviándose de la trayectoria. Esto se debe a que el modelo de seguimiento que presentaba hasta ahora este vehículo se basaba en la información proporcionada por un LiDAR.

De forma resumida, el LiDAR emplea la tecnología láser, enviando fotones que rebotan en objetos cercanos y que posteriormente son recibidos por el sensor. Gracias al registro de idea y vuelta de cada fotón se puede estimar la distancia de cada objeto a las proximidades del vehículo (ver Figura 8). Así es como se realizaba el seguimiento, determinando la distancia y posición entre el vehículo y la persona a seguir.

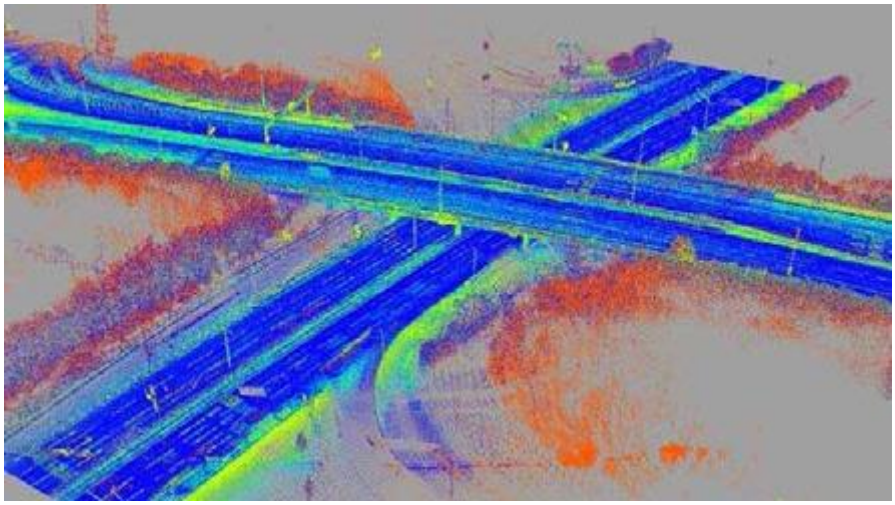


Figura 8: Información captada por un LiDAR en un vehículo (NOAA Coastal Services Center, 2012).

Pese a que este modelo presenta ciertas ventajas, entre las cuales destaca que se tiene información tridimensional del entorno, además de no depender de la iluminación del ambiente, posee otros inconvenientes. El más importante es que únicamente se tiene información sobre la distancia a los objetos, y no de la forma del objeto en sí, por lo que resulta más difícil que reconozca a una persona en concreto. En otras pruebas se pudo

observar este defecto cuando había personas moviéndose cerca de la persona encargada del rescate, lo que hacía que a veces cambiara de objetivo y no siguiera a la persona correcta.

- **Retos.**

Como hemos visto, el uso del dispositivo LiDAR hace mucho más complicada la tarea de poder discriminar entre la persona a seguir y otro tipo de objetos que se encuentren en el camino. Por tanto, el principal reto que en este Trabajo Fin de Máster se pretende resolver es el de solventar esta carencia, implementando un sistema que permita seguir a la persona en todo el trayecto que describa, otorgándole la máxima robustez para que el movimiento de la persona afecte lo menos posible al modelo de seguimiento. Además, el sistema debe ser capaz de discriminar a la persona a seguir. Dicho reto es de especial interés ya que en este tipo de situaciones se trabaja en entornos no estructurados, donde diferentes objetos y personas se encuentran a lo largo del trayecto del vehículo. Por último, el LiDAR es un dispositivo activo. Es decir, emite energía. Esta característica lo hace más detectable, lo que en determinados ámbitos de aplicación, como por ejemplo el caso de rescate en un escenario de enfrentamiento, no es deseable. Por tanto, se plantea utilizar sensores pasivos, como cámaras.

1.4. Organización de la memoria.

Para concluir este apartado introductorio, se va a explicar el esquema propuesto para la redacción de la memoria del Trabajo Fin de Máster. La memoria tiene la siguiente estructura:

- **Capítulo 1: Introducción.** Se ha explicado la motivación del Trabajo, así como los objetivos de éste y el estado del arte de seguimiento de personas en labores de rescate.
- **Capítulo 2: Equipo y software utilizado.** Se procederá a explicar los componentes hardware y herramientas software empleadas para la consecución de este Trabajo.
- **Capítulo 3: Seguimiento en el espacio de la imagen.** Se detallan los diferentes métodos estudiados para realizar el seguimiento de la persona dentro del espacio de la imagen en dos dimensiones, así como el método de seguimiento final escogido justificado.

- **Capítulo 4: Seguimiento en el espacio tridimensional.** Se ha explicado cómo se ha realizado el seguimiento de la persona en el espacio tridimensional, así como la localización de dicha persona y el nodo implementado para ello.
- **Capítulo 5: Experimentación y resultados.** Explicación de los experimentos realizados para comprobar el funcionamiento del sistema, así como los resultados obtenidos de dicha experimentación.
- **Capítulo 6: Conclusiones y trabajo futuro.** Conclusiones y valoraciones de todo el Trabajo Fin de Máster y trabajo futuro para la mejora del modelo de seguimiento.

Este trabajo está disponible en *github* a través del siguiente enlace: https://github.com/ricardovmartin/TFM-StereoVisionFollowMe. En él, se encuentra el código desarrollado para el sistema de reconocimiento y seguimiento y la propia documentación.

2. EQUIPO Y SOFTWARE UTILIZADO.

En este capítulo se va a explicar el equipo completo utilizado para la implementación del modelo de seguimiento de personas, así como las herramientas software básicas utilizadas para desarrollar la aplicación.

2.1. Software.

En primer lugar, se detallan los recursos software necesarios para la implementación del sistema. A modo introductorio, el programa completo está desarrollado dentro del sistema operativo **GNU/Linux**. En él, se ha utilizado el entorno integrado de desarrollo (IDE) **Eclipse** para desarrollar el seguimiento en el espacio de la imagen. Por otro lado, ha sido necesario el uso de la librería **OpenCV**, la cual incluye todos los paquetes necesarios de los algoritmos implementados en el seguimiento. Por último, para poder realizar el seguimiento en el espacio tridimensional y comunicar los datos obtenidos del seguimiento, se emplea **ROS** (Robot Operating System). A continuación, se procede a explicar en detalle cada uno de los recursos mencionados.

- **GNU/Linux.**

La aplicación de este Trabajo se ha desarrollado dentro de GNU/Linux. GNU/Linux es un software libre de código abierto. Esta característica hace que este sistema operativo tenga como objetivo la colaboración abierta por parte de usuarios especializados en informática, de tal forma que su código fuente pueda ser estudiado y modificado, aportando dichas mejoras a toda la comunidad de usuarios.

No es posible comenzar con la historia de Linux sin antes mencionar el sistema operativo **Unix**. Este software es un sistema operativo portable, multitarea y multiusuario desarrollado en 1969 por un equipo de desarrolladores del laboratorio *Bell*. La finalidad de dicho software era la de encontrar un software común para todos los ordenadores. A pesar de que en los años 80 varias grandes empresas utilizaran este software, únicamente se utilizaba en el sector industrial, pero no en ordenadores personales para el público general.

En 1991, Linus Torvalds, en aquel momento estudiante de la universidad de Helsinki, tuvo la idea de crear una versión académica libre basada en Unix. Dicho proyecto finalmente supuso el desarrollo del kernel Linux, como evolución del SO Unix para ordenadores personales. Hoy en día, numerosos procesadores instalados en cualquier dispositivo electrónico utilizan el software Linux.

- **Eclipse**.

El entorno integrado de desarrollo, en inglés *Integrated Development Environment* (IDE), en el que se ha desarrollado la primera parte de la aplicación, consistente en la localización y seguimiento dentro del espacio de la imagen, es **Eclipse**. Un IDE es un sistema de software utilizado para el desarrollo de aplicaciones, el cual mediante la combinación de diferentes herramientas en una sola interfaz de usuario gráfica (GUI) permite que la tarea del programador sea más sencilla. En general, un IDE cuenta con las siguientes herramientas:

- **Editor de código:** Editor de texto para desarrollar el código fuente en el lenguaje de programación en cuestión.
- **Compilador:** Interpreta las instrucciones del código fuente desarrolladas con el editor a código máquina para que sea entendible por el ordenador.
- **Depurador:** Permite analizar el programa de forma más minuciosa y encontrar errores en él.
- **Enlazador (linker):** Permite la conversión de archivos de código fuente distintos en un único fichero ejecutable.

El IDE Eclipse está compuesto por un conjunto de herramientas de programación de código abierto. Una de las características por las que se decidió desarrollar el código en esta plataforma es que presenta una interfaz sencilla de interpretar y un editor de textos con analizador sintáctico. Además, la compilación es en tiempo real, lo que permite agilizar mucho dicho proceso.

Originalmente, el IDE de Eclipse estaba pensado para desarrollar código en Java, pero permite la utilización de otros lenguajes de programación como C o C++ con la instalación de un *plugin*.

- **OpenCV.**

OpenCV (*Open Source Computer Vision Library*) es una librería de visión por computador y *Machine Learning*. Esta librería fue creada para ofrecer una infraestructura común para aplicaciones relacionadas con la visión por computador. OpenCV contiene más de 2500 algoritmos optimizados, los cuales incluyen técnicas clásicas y actuales de *Machine Learning*. La mayor parte del programa implementado se ha desarrollado haciendo uso de los módulos ofrecidos por OpenCV.

La librería OpenCV está desarrollada en su totalidad en C++. No obstante, incluye conectores para poder ser usada en otros lenguajes como *Python*, *Java*, *Matlab*, *Octave* y *Javascript*.

Varias características por las cuales esta librería es tan conocida y usada es que permite que sea usada libremente por cualquier usuario. Además, puede utilizarse en los sistemas operativos más conocidos (GNU/Linux, Mac OS X, Windows o Android). Por último, la documentación es muy extensa y detallada, además de estar en constante actualización.

- **ROS**.

Sistema Operativo Robótico, o en inglés *Robot Operating System* (ROS), es un sistema operativo de código abierto para el desarrollo de aplicaciones relacionadas con robots.

Los primeros comienzos de este sistema operativo tuvieron lugar a finales de 2006. El grupo de investigación de robótica de la universidad de Stanford buscaba estandarizar las aplicaciones robóticas de alguna manera. Finalmente, consiguieron desarrollar este software y el grupo de investigación evolucionó a la fundación *Open Source Robotics Foundation* desde 2014 hasta la actualidad. Hoy en día, proporcionan soporte a gran cantidad de empresas y equipos de investigación por todo el mundo.

Aunque en capítulos siguientes se entrará más en detalle sobre la implementación de ROS dentro del Trabajo, se procede a explicar brevemente cómo funciona este sistema operativo. ROS utiliza una red de pares en la que existe un nodo maestro denominado *roscore*, el cual da servicio a los nodos conectados. Los diferentes nodos constituyen procesos los cuales pueden enviarse mensajes entre sí a través de *topics*, mediante la publicación y suscripción a éstos. Un ejemplo de nodo podría ser un radar, cuyo sensor está publicando los datos capturados por un *topic* para otros nodos que requieran de dicho dato. La configuración de la comunicación entre estos procesos se establece por el nodo maestro. Esta es la principal tarea del *roscore*, por lo que los mensajes no pasan por él. Por tanto, la arquitectura es descentralizada, muy conveniente para aplicaciones robóticas.

2.2. Hardware.

En segundo lugar, se explica el equipo completo que compone el sistema de seguimiento.

- **Rover J8.**

El vehículo que transporta a los heridos y permite el seguimiento de los rescatadores es el Rover J8 (ver Figura 9).

Figura 9: Rover J8 (Grupo de Investigación en Robótica y Mecatrónica, UMA).

Las características mecánicas que más destacan es que es un vehículo 8x8 con capacidad anfibia, lo que le permite circular tanto por tierra como por terrenos acuáticos. Además, soporta 570 kg de carga útil y 680 kg de carga de arrastre. Puede alcanzar una velocidad máxima de 25 km/h, y cuenta con dos motores eléctricos, el principal y el de dirección.

El vehículo Rover J8 puede ser controlado mediante teleoperación y en modo *follow-me* que lleva implementado el propio vehículo. Además de estos dos controles, el grupo de robótica y mecatrónica de la UMA ha añadido ciertas características de movimiento autónomo, como son la teleoperación a través de redes 5G y la planificación de trayectorias en entornos no estructurados desde centros de control avanzado y lejano, a través de 5G.

- **NVIDIA Jetson AGX Xavier.**

Para la implementación del modelo de seguimiento, se ha utilizado el kit de desarrollo NVIDIA Jetson AGX Xavier (ver Figura 10). Se trata del equipo de a bordo del vehículo que permite realizar el procesamiento necesario para realizar las tareas de seguimiento. En él, se ejecutará el nodo de seguimiento y publicará los mensajes relacionados con la localización de la persona a seguir.

Figura 10: NVIDIA Jetson AGX Xavier [nvidia].

Este kit cuenta con una GPU Volta de 512 núcleos. Además, la arquitectura de la CPU es ARM con 8 núcleos de 64-Bit y cuenta con una memoria RAM de 32 GB. El ordenador Jetson AGX Xavier ofrece unas altas prestaciones en cuanto a eficiencia energética, densidad de computación y capacidades de inferencia de Inteligencia Artificial. Una de las características por la que hace que sea ideal para robots es su tamaño. Sus dimensiones son 100x87 mm, por lo que ofrece el mismo rendimiento que una *Workstation* pero con un tamaño diez veces menor. Por tanto, es una computadora diseñada para trabajar con máquinas autónomas específicamente.

- **ZED 2.**

Para poder localizar a la persona en el espacio tridimensional, es necesario conocer la distancia a la que se encuentra de la cámara y la desviación con respecto al centro de ésta, para enviar dicha información al vehículo y que actúe en consecuencia. Dicha información puede ser obtenida utilizando imágenes estereoscópicas, mediante el uso de dos cámaras en conjunto (Mrovlje & Vrancic, 2008).

En nuestro caso, tenemos a nuestra disposición el dispositivo ZED 2 (ver Figura 11) que integra ambas cámaras mediante software y hardware, haciendo que el proceso de obtención de la información requerida sea mucho más sencillo.

Figura 11: Cámara ZED2 [stereolabs].

Respecto a las especificaciones generales de esta cámara, cuenta con varias resoluciones de salida. En nuestro caso, hemos escogido una resolución de 1280x720 a 60fps. Para poder realizar los cálculos pertinentes, cuenta con giroscopios, acelerómetros y magnetómetros, además de barómetros y sensores de temperatura. El rango de medida de la profundidad va desde los 0,3 metros hasta los 20 metros, con una precisión entre 1% y 5%, en función de la distancia (menor precisión a mayor distancia).

La cámara ZED2 ofrece una nube de puntos e imágenes de confianza de las capturas (ver Figura 12) tomadas que nos serán realmente útiles para el desarrollo de nuestro programa como en capítulos venideros se explicará. Además, cuenta con un software propio que implementa la detección de objetos y seguimiento de personas, entre otras aplicaciones. No obstante, no se ha hecho uso de dichas herramientas.

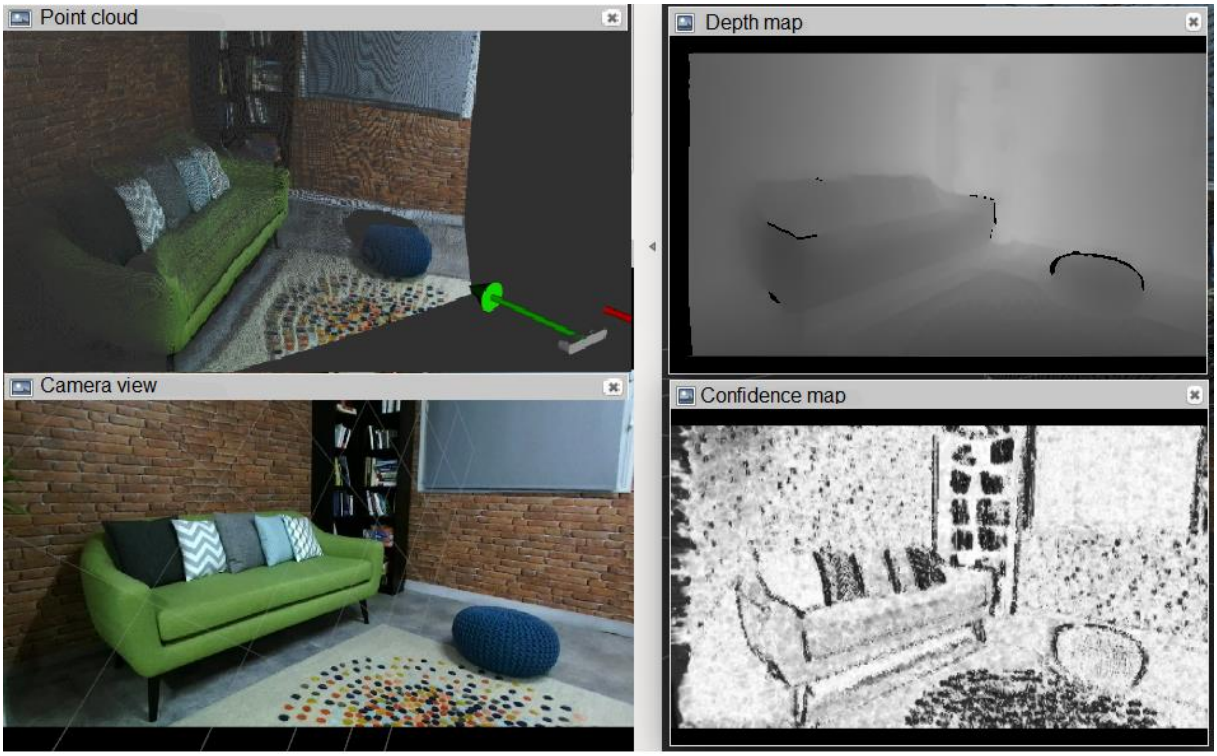


Figura 12: Mapas extraídos de la cámara ZED2 [stereolabs].

En el conjunto de imágenes (ver Figura 12) se muestran los mapas que la cámara ZED2 es capaz de extraer y que nos servirán para implementar nuestro modelo de seguimiento. El mapa de confianza, aunque se explicará más adelante, nos sirve para saber si las medidas son precisas o no, siendo los píxeles más claros más confiables que los más oscuros.

Por último, permite su integración con ROS, por lo que tendremos información de los datos comentados anteriormente mediante la suscripción a los *topics* correspondientes. Junto a esta integración y el Kit de Desarrollo Software (SDK) de ZED, podemos trabajar con la cámara desde la computadora Jetson.

3. SEGUIMIENTO EN EL ESPACIO DE LA IMAGEN.

En el tercer apartado de este Trabajo se van a explicar varias técnicas llevadas a cabo para el seguimiento de la persona en el espacio bidimensional de la imagen donde se proyecta la escena, así como la elegida definitivamente para nuestro modelo.

En todo el desarrollo del modelo de seguimiento se han utilizado imágenes que forman parte del *dataset* UMA-SAR (Morales, Vázquez-Martín, & Mandow, 2021) tal y como comentamos en el anterior apartado. Este conjunto de imágenes ha sido realmente útil para obtener el sistema de seguimiento, ya que ha sido tomado durante las jornadas de emergencia que tienen lugar en la UMA. Por tanto, tenemos un *dataset* que se asemeja mucho a los entornos a los que este sistema se encontrará en la realidad.

3.1. Filtro RGB y Máquinas vectores de soporte.

La primera versión por la que se decidió comenzar para el reconocimiento y seguimiento de la persona fue a través de varios filtros RGB, como se explicará más adelante, para eliminar el resto de objetos dentro de la imagen y quedarnos únicamente con la persona de interés. Además, para discriminar entre más personas que haya alrededor, se utilizarán varios descriptores y se entrenará una máquina de vectores de soporte (SVM) para poder seguir a la persona. Pese a que esta primera versión quedó descartada por los motivos que al final de este apartado se mencionarán.

- **Filtro RGB.**

Los colores primarios son aquellos que tienen características completamente diferentes entre sí, es decir no se pueden obtener mediante la adición de otros colores. Los colores primarios son el rojo, verde y azul (ver Figura 13). El resto de colores se obtienen a través de síntesis aditiva, por lo que sumando los colores primarios se puede representar cualquier color.

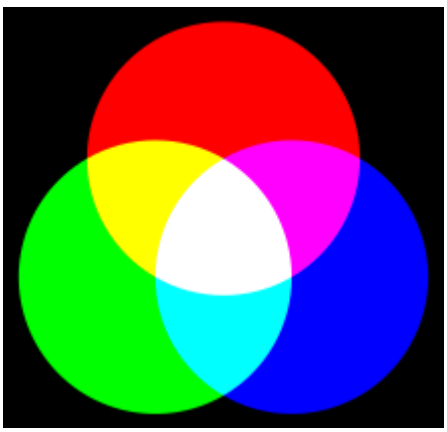


Figura 13: Colores primarios (hernescreatives.com).

La versión primera para la detección y seguimiento de personas consistía en eliminar el fondo, es decir partes que no fueran de nuestro interés. El camino descrito por la persona transcurre por un terreno de campo, por lo que interesará eliminar cualquier parte de matorrales, tierra y cielo que haya en la imagen. Para ello, se han aplicado diferentes filtros RGB, de tal forma que se van eliminando partes de la imagen que no nos interesa y están dentro de un rango RGB que permitan ser suprimidos.

La batería de imágenes que se han utilizado para el procesamiento describe la trayectoria realizada por diferentes personas (ver Figura 14).

Figura 14: Imagen referencia filtro RGB (Morales, Vázquez-Martín, & Mandow, 2021)

Aplicados los diferentes filtros, se obtiene la imagen binaria que se muestra en la Figura 15.

Figura 15: Imagen procesada filtro RGB.

Los píxeles blancos marcan las regiones de interés, mientras que los negros no se utilizan. Como vemos, se elimina gran parte del suelo, hierba y cielo de la imagen, quedando como regiones de interés principalmente personas y otras partes que no se han podido eliminar. Como vemos, hay zonas que no pertenecen a personas que se deberían de haber eliminado, pero no ha sido posible, ya que al aplicar simplemente diferentes filtros RGB no se distingue entre qué objetos eliminar y cuáles no. Esto es un gran problema, ya que si el color de la vestimenta de la persona es similar al del fondo, no podrá ser diferenciada. Desafortunadamente, esto es bastante común, ya que las labores de rescate pueden ser llevadas a cabo por militares que llevan ropa de camuflaje para mimetizarse con el entorno.

3.1.1. Máquinas de vectores de soporte (SVM).

Una máquina de vectores de soporte, en inglés *Support Vector Machine* (SVM) es un modelo de *Machine Learning*, capaz de trabajar tanto con clasificaciones lineales como no lineales o modelos de regresión.

El modelo de clasificación lineal de una máquina de vectores de soporte consiste en separar dos clases distintas por una frontera. Obviamente, estas dos clases han de ser linealmente separables, en caso contrario el clasificador no podría separarlas.

Uno de los parámetros más importantes del SVM a controlar es el hiperparámetro C. Este parámetro permite controlar la sensibilidad del margen que separa ambas clases.

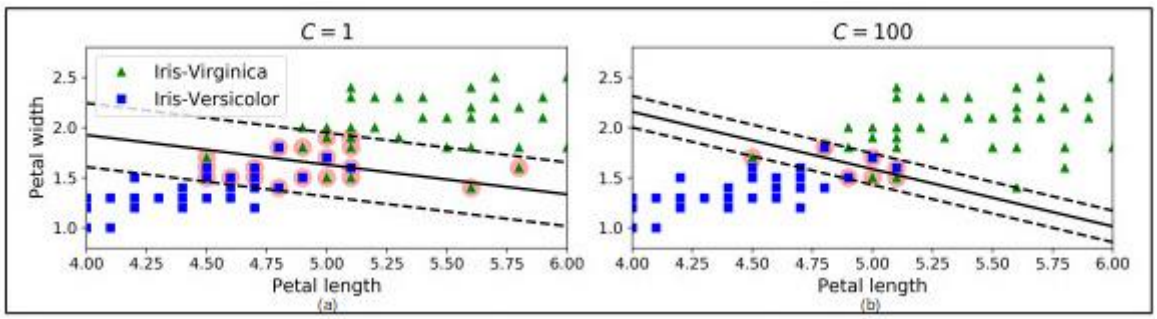


Figura 16: Hiperparámetro C en SVM (Facultad de Ingeniería, Universidad de la República, 2018).

Si se establece un valor pequeño de C, el margen será más amplio y generalizará mejor, pero habrá más violaciones de los márgenes (ver Figura 16, a). Por otro lado, aumentando C tendremos un modelo menos flexible pero con menor error, ya que menos predicciones serán incorrectas o entrarán dentro del margen (ver Figura 16, b).

3.1.2. Implementación.

En esta etapa se entrena el modelo de SVM capaz de identificar a las personas (en caso de haberlas) presentes en la imagen. Para ello, debemos partir de un conjunto de

imágenes y extraer las características deseadas de las personas para entrenar el modelo. Es por ello por lo que se procesó la imagen realizando un filtro RGB y así poder extraer dichas características. Por tanto, nuestro *dataset* consistirá en la batería de imágenes tomadas por la cámara en la que dos personas distintas realizaban un trayecto (Morales, Vázquez-Martín, & Mandow, 2021).

Figura 17: Dataset personas (Morales, Vázquez-Martín, & Mandow, 2021).

3.1.2.1. Extracción de características.

Existen muchos descriptores para clasificar objetos (en nuestro caso personas). En esta primera versión, se utilizaron dos descriptores basados en la geometría de la persona. El primero de ellos es muy sencillo y simplemente es el área del contorno que define la persona. El segundo es la circularidad, definida como el cociente del perímetro del contorno de la persona al cuadrado y el área correspondiente:

$$\text{Circularidad} = \frac{\text{Perímetro}^2}{\text{Área}}$$

Una vez tenemos el contorno de la persona de interés, hallamos estos dos descriptores y se guardan en un archivo para que posteriormente todo el conjunto sea entrenado. Por tanto, el aprendizaje es supervisado.

Antes de seguir con la clasificación, veamos cómo se obtuvieron los contornos para poder extraer las características.

3.1.2.2. Detección de contornos.

Encontrar el contorno de un objeto consiste en trazar la curva que une todos los puntos a lo largo del borde que tienen valores de color o intensidad iguales. Por tanto, es importante tener una imagen en blanco y negro como la que obtuvimos con el filtrado RGB para poder aplicar esta operación. A partir de dicha imagen, aplicamos la función *findContours* de **OpenCV** (OpenCV). Para no ocupar mucho espacio en la memoria guardando todas las posiciones de todos los contornos, únicamente se guardan coordenadas específicas de los contornos, es decir, solo toma la coordenada del punto final del segmento, y no todo el segmento (ver Figura 18).

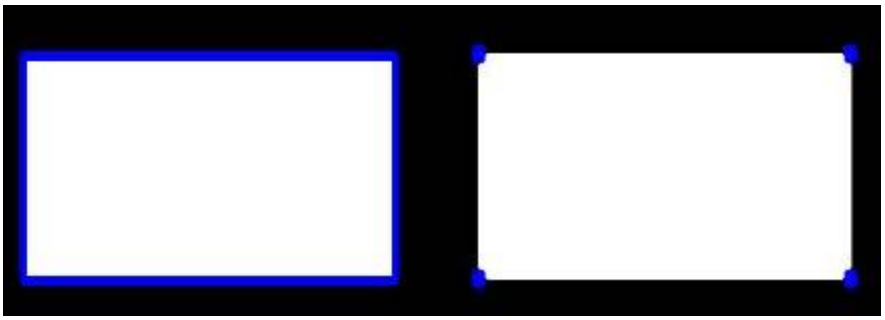


Figura 18: Obtención contornos (OpenCV).

Con estos contornos, obtenemos el área y la circularidad de la persona de interés en cada imagen para almacenarlos en un fichero, el cual nos servirá para entrenar a la máquina de vectores de soporte.

Después de entrenar el clasificador, éste es capaz de identificar a las personas correctamente, pero también detecta algunos objetos que tenían una forma similar a la de una persona. Por tanto, fue necesario incluir un nuevo descriptor, pero sin usar otro clasificador, tal y como se detalla en el apartado 3.1.2.3.

3.1.2.3. Histograma de Gradientes Orientados (HOG).

En primer lugar, se va explicar en qué consiste el Histograma de Gradientes Orientados, para después ver cómo se implementó en nuestro primer programa.

- **Concepto teórico.**

El Histograma de Gradiente Orientados (*Histogram of Oriented Gradients HOG)* es un descriptor de características ampliamente utilizado en visión por computador y procesamiento de imágenes.

Primeramente, vamos a analizar brevemente cómo se obtienen los gradientes en una imagen. En la Figura 19 tenemos un círculo dibujado en blanco y el fondo en negro. Si recorremos la imagen horizontal o verticalmente, habrá un cambio de intensidad cuando traspasamos el círculo. De esta forma simple, obtenemos la dirección del gradiente.

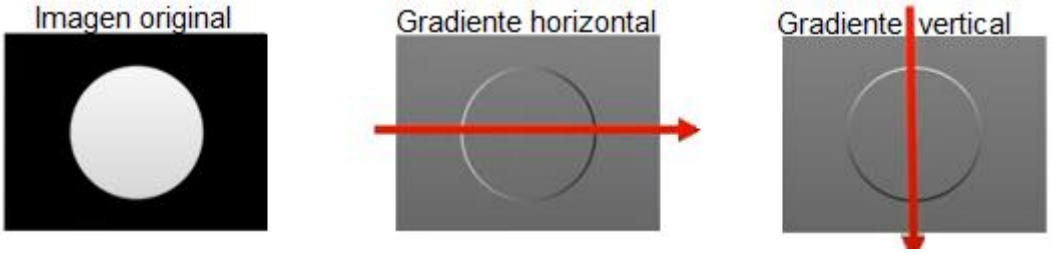


Figura 19: Dirección gradiente.

Para una imagen, el valor del gradiente en cada píxel se obtiene a partir de la intensidad de los píxeles vecinos. Denotando 'x_0' como la posición del píxel en cuestión y 'x_1', 'x_2','y_1','y_2' a los píxeles de alrededor, la magnitud y dirección del gradiente será:

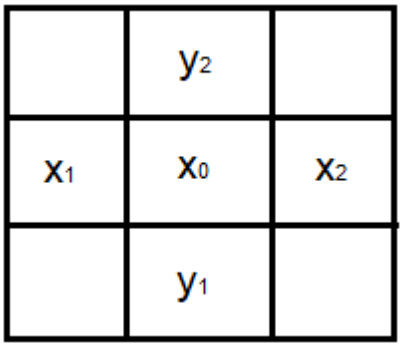


Figura 20: Cálculo gradiente.

$$\text{Magnitud} = \sqrt{(x_2^2 - x_1^2) + (y_2^2 - y_1^2)}$$

$$\text{Dirección} = \tan^{-1}\frac{y_2 - y_1}{x_2 - x_1}$$

De esta manera, en cada píxel de la imagen tenemos un vector representativo, cuya magnitud y dirección dependerá de los píxeles vecinos.

No obstante, no se utiliza la información del gradiente de un único píxel de forma separada, ya que puede contener ruido. Para solucionar esto, se agrupa un conjunto de píxeles en bloques y se calcula el histograma sobre ellos.

- **Cálculo histograma.**

El histograma contiene una serie de casillas donde se guarda la suma de las magnitudes que contengan un ángulo de gradiente específico (ver Figura 21).

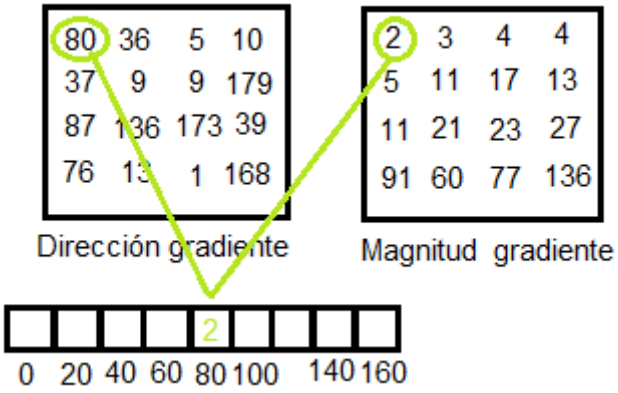


Figura 21: Cálculo histograma.

- **Implementación.**

La inclusión del descriptor HOG en la primera versión consistió en adquirir una primera imagen y obtener el descriptor de la persona en concreto, sabiendo su posición de antemano. Por tanto, al principio del proceso antes de iniciar el movimiento se toma dicho descriptor de la persona en estático. Al comenzar el movimiento, tendremos varios candidatos a ser la persona a seguir en cada uno de los fotogramas siguientes (ver Figura 22).

Figura 22: Personas detectadas primera versión.

A continuación, extraemos el descriptor HOG de cada contorno candidato. Cada uno de ellos se compara con el descriptor de la imagen original, y el que sea más parecido

significará que es la misma persona y por consecuente la persona a seguir. Dicho descriptor viene expresado como un vector, como explicamos anteriormente, por lo que la comparación se hará hallando la distancia euclídea con todos los puntos de los dos vectores.

$$\text{distancia} = \sqrt{\sum_i \left(v_{HOG,1}(i) - v_{HOG,1}(i)\right)^2}$$

De esta manera, comparando cada contorno reconocido por el clasificador como persona con el modelo de persona, obtenemos la posición de la nueva persona a seguir (ver Figura 23). Este proceso se realiza de forma continua, actualizando la nueva posición y descriptor de referencia en cada imagen.

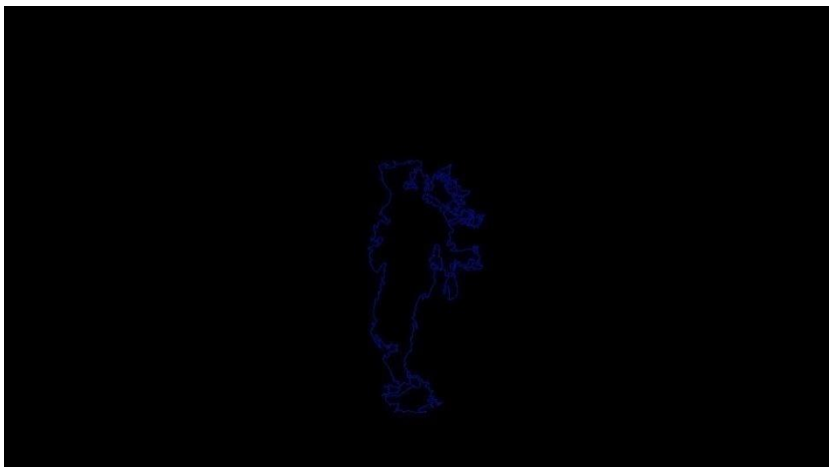


Figura 23: Resultado final primera versión.

Como dijimos al comienzo de este apartado, este método no es robusto, ya que es muy sensible a la iluminación y el fondo donde se encuentre el objetivo. De hecho, hubo que aplicar hasta tres filtros RGB distintos a lo largo del recorrido para obtener buenos resultados. Por tanto, el siguiente estudio debe consistir en encontrar una solución que no dependa de los colores del fondo, ya que es un problema que va a estar siempre presente en misiones de rescate.

3.2. Sustracción de fondo.

El segundo método que se analizó fue la sustracción de fondos. Esta técnica es sencilla de entender ya que, como su propio nombre indica, calcula el fondo de la imagen sustrayendo el fotograma actual del anterior y obtiene así únicamente el objeto en movimiento (ver Figura 24).

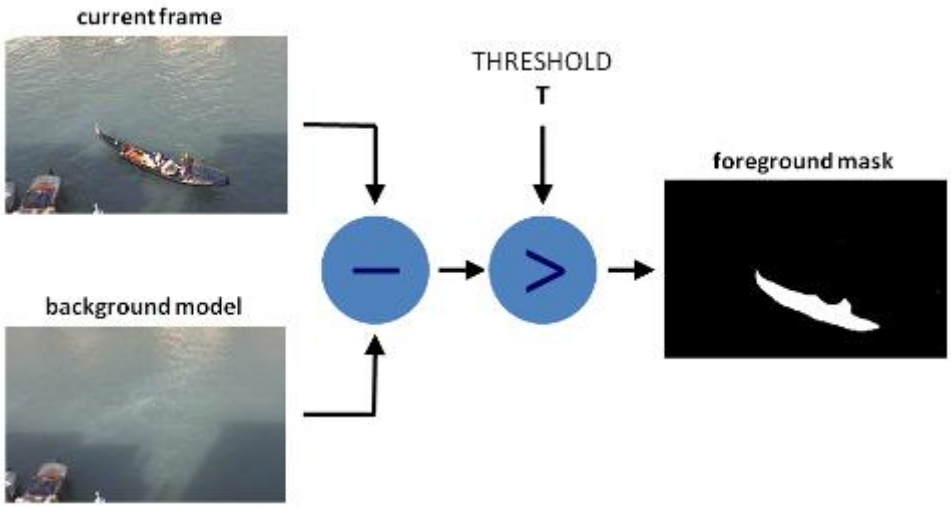


Figura 24: Sustracción de fondo (OpenCV).

El problema que presenta este método es que solo es útil para imágenes estáticas, es decir donde en toda la imagen el único movimiento, o al menos el principal, sea el objeto a identificar, además de que la cámara no ha de estar en movimiento, por supuesto. En nuestro caso, ninguna de estas condiciones se cumple, ya que la cámara está en movimiento al estar colocada en el vehículo y el fondo está cambiando constantemente. Por tanto, fue una técnica que se tuvo en cuenta pero fue descartada al analizar las condiciones de nuestros ensayos. No obstante, cabe destacar que es un método bastante efectivo y sencillo para la detección de objetos en imágenes estáticas. Esta técnica es ampliamente utilizada en videollamadas para detectar el movimiento de la persona, ya que por lo general el fondo es estático. Además, también se utiliza en compresión de vídeo, para reducir el tamaño del vídeo.

3.3. Flujo óptico.

El siguiente método que se decidió implementar fue el uso del flujo óptico. Esta técnica consiste en determinar el movimiento de los objetos entre dos fotogramas consecutivos, causado por el movimiento relativo entre el objeto y la cámara. De esta manera, se podría hacer que el sistema fuera independiente de las condiciones de luminosidad y colores presentes en la imagen, además de que tanto la cámara como el objeto pueden estar en movimiento.

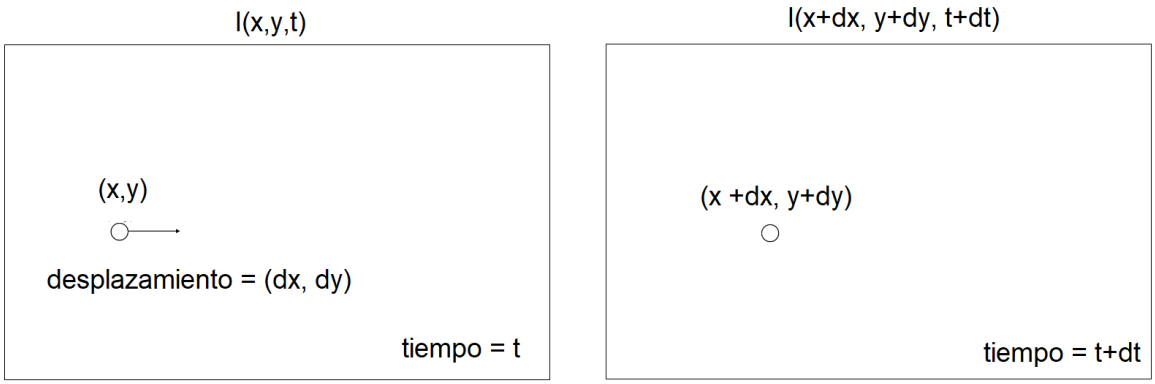


Figura 25: Problema del flujo óptico.

Tal y como se muestra en la Figura 25, la intensidad del píxel representado como un círculo es una función de la posición y el tiempo, de tal forma que el pixel se mueve una distancia (dx, dy) en el instante de tiempo dt, es decir el tiempo hasta el siguiente fotograma. Existen diferentes métodos para el cálculo de la velocidad debido al movimiento de cada píxel. Aunque en esta memoria no se entrará detalle, los más conocidos son el flujo óptico disperso (*Sparse Optical Flow*) y flujo óptico denso (*Dense Optical Flow*).

Varios estudios, (Huang, Zou, Zhu, & Zhu, 2019) o (Yun, Lim, & Young, 2016), entre otros, se han basado en el flujo óptico denso para obtener el movimiento de personas. La principal razón por la que no han utilizado flujo óptico disperso es porque presenta menor exactitud que el flujo denso, aunque su coste computacional es menor. Brevemente, el flujo óptico disperso obtiene los vectores de flujo de solo ciertas características, como pueden ser esquinas o contornos de un objeto, mientras que el flujo óptico denso calcula el flujo de toda la imagen.

De forma resumida, los métodos para el seguimiento de personas que se basan en flujo óptico consisten en extraer el movimiento global del fondo, para así extraer la persona de interés puesto que tendría un movimiento diferente. Una manera de estimar el movimiento del fondo (Yun, Lim, & Young, 2016) es asumir que este es mucho más grande que el objeto de interés. Sabiendo esto, dicho movimiento se representa con una matriz de transformación que representa el movimiento completo del fotograma anterior en t-1 y el actual, según:

$$H_{t:t-1} = \frac{\left[X_1^{(t)}, X_2^{(t)}, \dots\right]}{\left[X_1^{(t-1)}, X_2^{(t-1)}, \dots\right]}$$

Donde:

$$X_i^{(t-1)} = (x_i, y_i, 1)^T, \qquad X_i^{(t)} = (x_i + u_i, y_i + v_i, 1)^T$$

Las variables u_i e y_i se refieren al desplazamiento de los píxeles correspondientes.

Las regiones que pertenezcan a las personas que están moviendo no satisfacen la ecuación anterior, por lo que se obtiene la velocidad del movimiento de las personas, siendo posible por tanto identificarlas.

Figura 26: Estimación del movimiento de las personas (Huang, Zou, Zhu, & Zhu, 2019).

Al implementar dicha técnica en nuestro programa, no se obtuvieron buenos resultados. Analizando la razón por la cual no se conseguía separar la persona en movimiento del resto de la imagen, se vio que no había una clara diferencia entre el movimiento de la persona y el vehículo, por lo que no era posible separarlos. Puesto que el objetivo del vehículo es seguir el movimiento de la persona y dado que dicho movimiento respecto al vehículo no es muy significativo, resulta difícil separar ambos movimientos.

3.4. Mecanismos de seguimiento en imagen.

La librería OpenCV ofrece varios algoritmos de seguimiento de objetos en imágenes dinámicas. En este apartado se van a explicar los más interesantes para nuestro sistema, así como el método definitivo elegido.

De forma resumida, un seguidor o *tracker* necesita que se detecte el objeto en el primer fotograma. Una vez detectado simplemente se irá actualizando con diversa información a lo largo de la secuencia de imágenes. No obstante, puede ocurrir que se pierda el seguimiento, por lo que cada cierto tiempo será necesario realizar una nueva detección del objeto para evitar que éste se pierda. Esto suele ocurrir cuando el objeto, o en este caso la persona, es ocluido por otro objeto por un periodo significativo de tiempo o se mueve muy rápido.

A continuación, vamos a analizar la metodología y funcionamiento de varios seguidores.

3.4.1. Median Flow.

El primer *tracker* que vamos a estudiar surge como un método para detectar cuándo un seguidor ha fallado en el seguimiento y permitir así que otros tipos se seguidores puedan evaluar por ellos mismos su fiabilidad. El error obtenido en el *tracking* será utilizado para implementar el seguidor Median Flow (Kalal, Mikolajczyk, & Matas, 2010).

La detección del fallo se realiza analizando el movimiento de un píxel en el tiempo. Después, se analiza dicho punto desde el último fotograma hacia atrás, es decir en sentido contrario. Por último, se comparan las dos trayectorias y si difieren significativamente, el seguimiento se considera incorrecto.

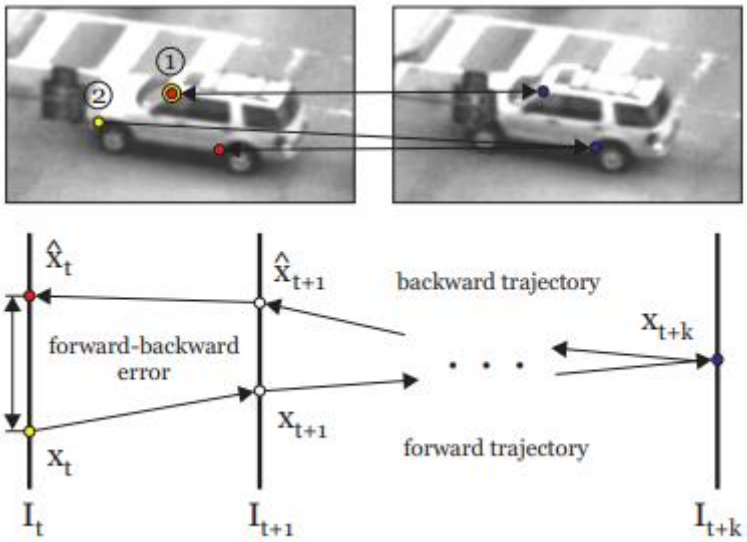


Figura 27: Estimación del error en Median Flow (Kalal, Mikolajczyk, & Matas, 2010).

La Figura 27 muestra un ejemplo de una detección fallida. En este caso, el punto 1 es visible en todo el recorrido, por lo que el seguimiento sería correcto. Sin embargo, el punto 2 está oculto en el último fotograma, por lo que si se realiza el análisis de delante hacia atrás, el seguidor detectaría un punto distinto.

El seguimiento del objeto se realiza a través del seguidor Lucas-Kanade basado en flujo óptico. Con las predicciones que se realizan en cada fotograma, se calcula el error que antes se mencionaba, y se filtran aquellos peores. Aquellos puntos con buenos resultados se utilizan para estimar el desplazamiento del recuadro que conforma el objeto a seguir, y se va actualizando en cada fotograma.

El método para calcular el error permite detectar fallos cuando el movimiento del objeto es lento. Sin embargo, no es posible detectar fallos cuando ocurren oclusiones o

movimientos rápidos. Esta fue la principal razón por la que se desestimó utilizar este seguidor, ya que necesitábamos de un seguidor más robusto frente a cambios en la trayectoria.

3.4.2. Tracking-Learning-Detection.

Como su propio nombre indica, el seguidor *Tracking-Learning-Detection* (TLD) combina estas tres técnicas para el seguimiento de objetos.

En primer lugar, el seguidor sigue al objeto imagen a imagen. Es común que el seguidor acumule error durante la trayectoria y falle, especialmente si el objeto desaparece del campo de visión de la cámara. Es aquí donde actúa el detector, el cual analiza las apariciones del objeto en todos los fotogramas y corrige al seguidor en caso de ser necesario. Sin embargo, requiere de un entrenamiento previo, por lo que no es aplicable a objetos desconocidos. Por último, existe un aprendizaje para estimar el error del detector y actualizarlo para prevenir errores futuros.

Entrando un poco más en profundidad, el seguidor que se utiliza aquí está basado en formas geométricas y su movimiento se estima entre fotogramas consecutivos. En el caso del detector, no utiliza el típico procedimiento de extracción y reconocimiento de características, ya que para ello es necesario conocer la geometría del objeto previamente. Como alternativa, emplea el método propuesto en el artículo (Viola & Jones, 2001), el cual se basa en analizar la imagen dentro de una ventana de varios tamaños y con esa ventana decide si se encuentra el objeto de interés o no.

Respecto a la parte de entrenamiento, se tiene en cuenta que solo se posee una única muestra primera etiquetada para realizar el seguimiento, puesto que no se dispone de un *dataset* previo para realizar el entrenamiento. Por tanto, el entrenamiento deberá ser semisupervisado. En cada fotograma, se evalúa el detector y se identifican sus errores, actualizando el detector para evitar dichos errores en el futuro. El entrenador se dividirá en dos estimadores diferentes: uno para identificar los falsos positivos y otro para los falsos negativos.

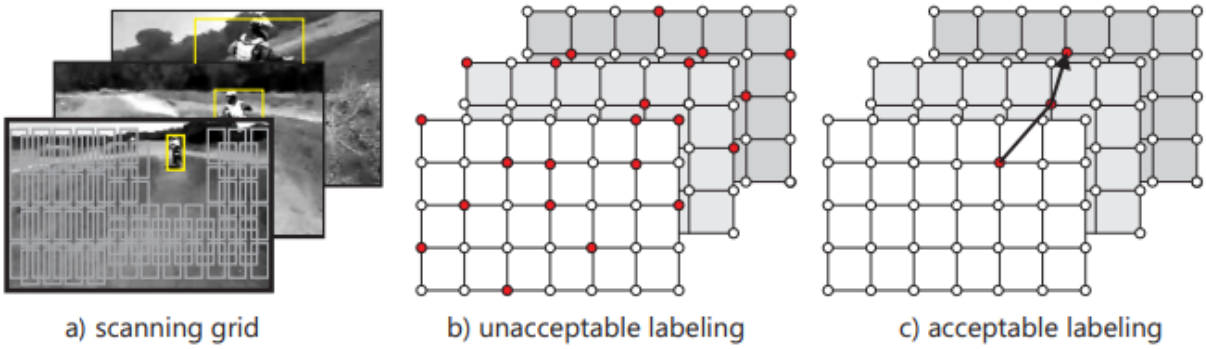


Figura 28: Secuencia aprendizaje TLD (Viola & Jones, 2001).

La Figura 28 muestra una secuencia de imágenes para entender mejor cómo funciona el entrenador. Como dijimos anteriormente, la imagen se divide en ventanas más pequeñas, las cuales son independientes para el detector. Los puntos rojos de la imagen (b) indican que el objeto aparece en dichas ubicaciones, por lo tanto es poco probable que la detección sea correcta. Por el contrario, en la imagen (c) se puede observar como el punto donde está el objeto describe una trayectoria a lo largo del tiempo. La idea por tanto del diseño de los estimadores es analizar esta estructura para detectar los errores.

El estimador que detecta los falsos negativos aprovecha el movimiento del objeto a lo largo del tiempo. Este estimador memoriza la ubicación del objeto en el fotograma anterior y estima su ubicación en el fotograma actual utilizando el seguidor que antes mencionábamos. En caso de que el detector considere dicha ubicación como negativo, y por tanto esté considerando un falso negativo, el estimador considera dicha ubicación como positiva en su lugar.

Por otro lado, el estimador que encuentra los falsos positivos asume que el objeto solo puede aparecer en una única ubicación en la imagen. El estimador analiza todas las respuestas del detector en el fotograma actual y selecciona la que sea más confiable.

Una de las grandes ventajas del seguidor TLD es que trabaja muy bien cuando existen oclusiones de la persona a seguir en varios fotogramas. Sin embargo, se obtienen muchos falsos positivos que hacen que sea un seguidor poco útil. Por tanto, no es aconsejable implementar este seguidor en nuestro sistema.

3.4.3. Multiple Instance Learning.

El seguidor MIL (*Multiple Instance Learning*) está diseñado para tener buenos resultados en escenarios adaptativos, donde el modelo a seguir va evolucionando a lo largo

del tiempo. Los fundamentos teóricos de este seguidor están desarrollados en el artículo (Babenko, Yang, & Belongie, 2009).

En relación al seguimiento, en lugar de considerar únicamente la ubicación del objeto actual, se toma un área pequeña alrededor del objeto para generar muestras positivas potenciales, denominada 'mochila'. Si una mochila se considera positiva, implica que habrá alguna muestra positiva dentro de ella, y por tanto existirá la persona a seguir dentro de dicha área. Esto puede dar lugar a que existan varias muestras positivas dentro de una misma mochila. Esta ambigüedad debe ser solventada por el clasificador, que debe ser capaz de adivinar qué muestra es la mejor, cuya tarea recae en el algoritmo MIL.

El entrenamiento MIL consiste en considerar una 'mochila' de posibles muestras positivas, como comentábamos anteriormente. La obtención de una muestra positiva está basada en términos de probabilidad, de tal forma que si una de las instancias tiene una probabilidad alta de ser el objeto a seguir, la mochila también tendrá una probabilidad alta (Zhang, Platt, & Viola, 2005).

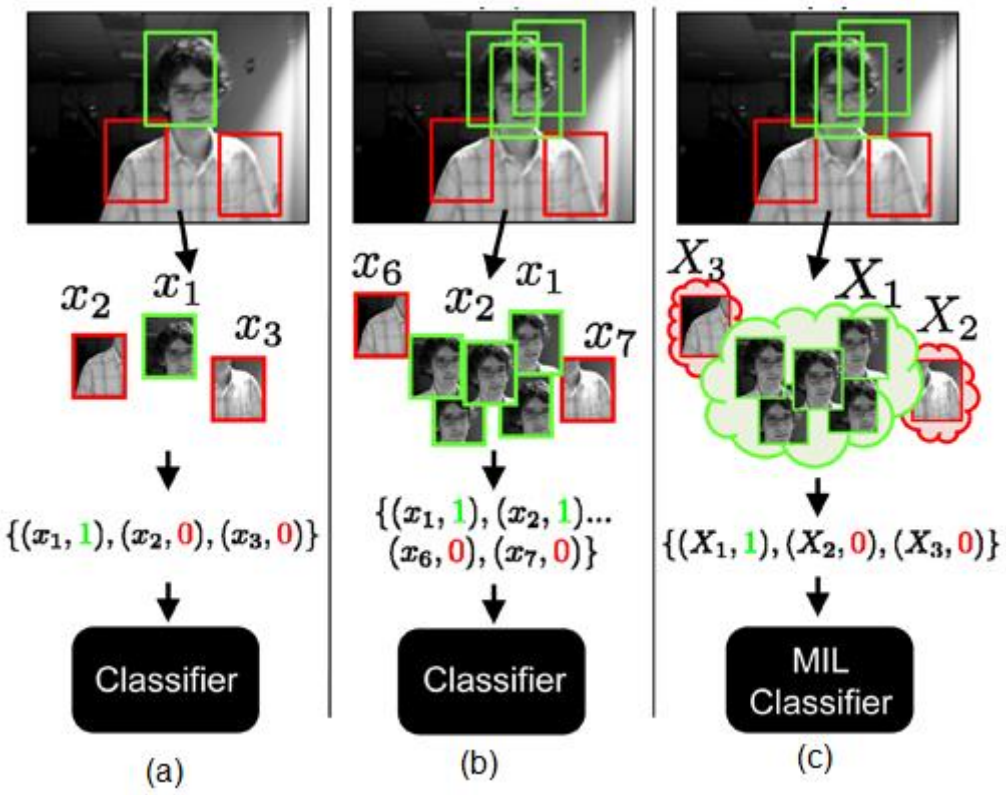


Figura 29: Comparativa actualización modelo y seguidor MIL (Babenko, Yang, & Belongie, 2009).

En la Figura 29 podemos ver una comparativa de diferentes clasificadores para entender mejor cómo funciona el clasificador MIL. En la imagen (a) se utiliza una única imagen para actualizar el clasificador mediante un método tradicional. En la imagen (b) se

tratan varias áreas de la imagen positivas de forma independiente para el clasificador, pudiendo obtener malos resultados de él. Por último, en la imagen (c) se encuentra el clasificador MIL, el cual utiliza solamente una mochila con varias áreas de la imagen.

De los seguidores presentados hasta ahora, este es el que mejores resultados puede ofrecer para nuestro sistema. No obstante, solo se recomienda si la versión de OpenCV que se está utilizando es menor a la 3.0. En caso contrario, es más recomendable utilizar otro seguidor al ofrecer aún mejores resultados y del cual hablaremos a continuación.

3.4.4. Kernelized Correlation Filters.

El seguidor empleado en nuestro sistema y del cual vamos a entrar en profundidad se denomina ***Kernelized Correlation Filters*** **(KCF)** (Henriques, Caseiro, Martins, & Batista, 2014). Este *tracker* utiliza el Histograma de Gradientes Orientados como descriptor, precisamente el descriptor que se va a utilizar para la descripción de la persona a seguir, y su modelo de regresión es el *Kernel Ridge Regression*. En primer lugar, vamos a estudiar el modelo de regresión empleado, para después ver cómo se implementa en el seguidor.

- **Regresión lineal.**

El modelo de regresión lineal (ver Figura 30) hace una predicción del resultado estimado simplemente realizando la suma ponderada de las características de entrada, más una constante llamada término bias (θ_0) (Géron, 2019).

$$\hat{y} = \theta_0 + \theta_1 x_1 \theta_2 x_2 + \cdots + \theta_n x_n$$

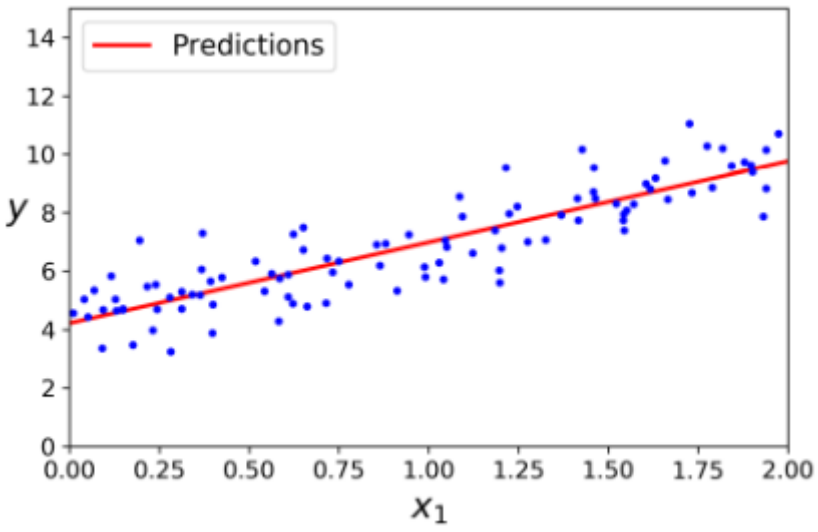


Figura 30: Modelo de predicción de la regresión lineal (Géron, 2019).

- **Regresión Ridge.**

El sobreajuste en los modelos de entrenamiento es un problema muy común. Para reducir este sobreajuste se restringe el modelo, puesto que cuantos menos grados de libertad tenga, más difícil será sobreajustar los datos. En un modelo lineal, típicamente se restringen los pesos del modelo. Así es como este tipo de regresión funciona.

La regresión ***Ridge*** es una versión regularizada (restringida) de la regresión lineal. Añadiendo un término de regularización permite no solo ajustar los datos, sino también mantener los pesos del modelo con los valores más pequeños posibles.

La función de coste representa el error cuadrático medio del número de muestras de los datos, para todos los datos de entrenamiento. Para este modelo, la función de coste es:

$$J(\theta) = MSE(\theta) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

El hiperparámetro que controla cuánto se restringe el modelo es α (ver Figura 31). Conforme su valor aumenta, los pesos tienden a cero y la recta se va haciendo cada vez más horizontal, mientras que si α tiende a cero, se asemeja más a un modelo lineal.

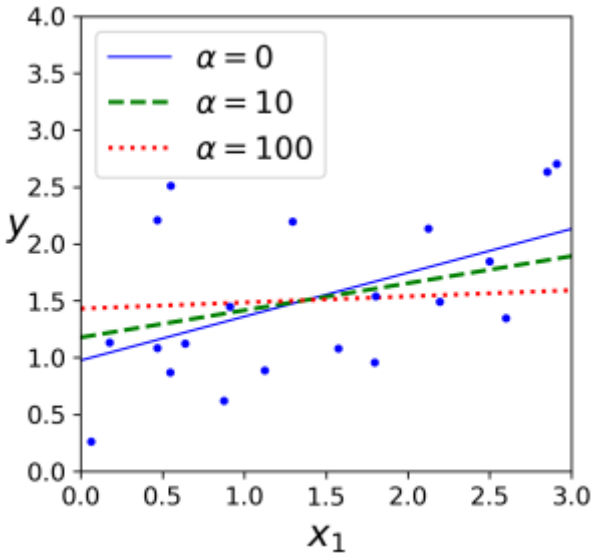


Figura 31: Modelo de predicción de la regresión ridge (Géron, 2019).

- **Regresión kernel ridge.**

Este modelo combina la regresión *Ridge* antes mencionada con el truco del kernel. El truco de kernel se emplea cuando no es posible hallar un hiperplano que pueda separar dos clases distintas ya que no son linealmente separables, es decir sufren de multicolinealidad. De forma resumida, consiste en desarrollar una nueva dimensión que nos permita separar las clases (ver Figura 32).

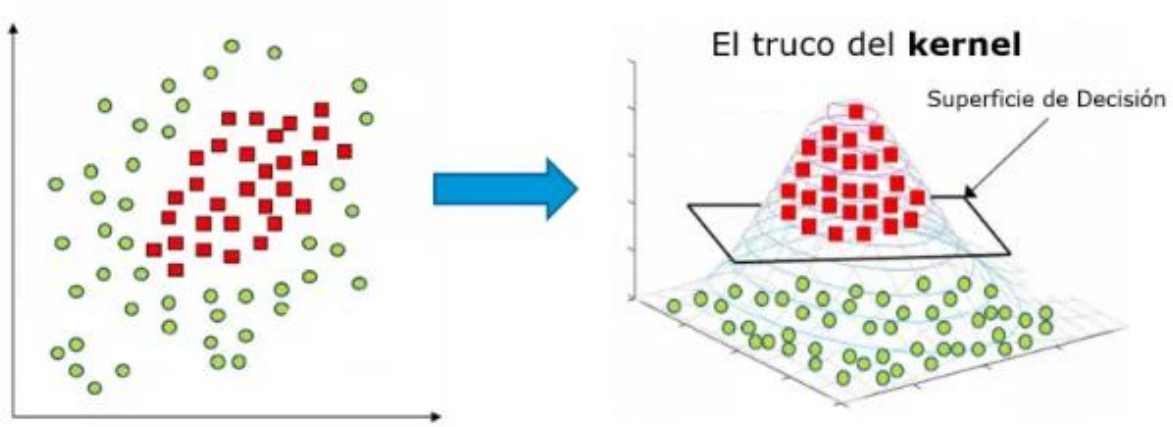


Figura 32: Truco del kernel (Vaerenberh, S. V., & Santamaria, I. 2018).

- **Implementación.**

El modo de funcionamiento de este seguidor es similar al seguidor MIL, por lo que también considera ventanas cercanas a la posición de la persona. Sin embargo, elimina el problema de solapamiento que ocurría en el MIL al poder existir múltiples muestras positivas, gracias a las herramientas matemáticas explicadas anteriormente.

La secuencia de implementación del seguidor KCF consiste básicamente en dos pasos. En primer lugar, se inicia el seguidor con el primer fotograma y las coordenadas de la persona a seguir, delimitadas por un recuadro. En este momento ya se ha creado un descriptor HOG con los datos de la persona a seguir. A continuación, se va actualizando dicho descriptor y la ubicación del recuadro en los siguientes fotogramas, siempre y cuando el modelo de regresión determine que existe la persona objetivo con la característica extraída del HOG. En caso contrario, el seguidor no actualiza la nueva ubicación de la persona hasta que vuelva a encontrarse. En el siguiente capítulo se darán más detalles del seguidor implementado.

En relación a las ventajas respecto a otros seguidores, se obtiene mayor precisión y velocidad, además de detectar mejor los fallos. Estas fueron las principales razones por las que se decidió utilizar este *tracker* para realizar el seguimiento de la persona en nuestro Trabajo. No obstante, el seguidor no es capaz de recuperar a la persona si esta se ocluye totalmente.

La principal limitación que se ha encontrado durante la implementación de este seguidor es que no actualiza el tamaño del recuadro de la persona a seguir, a pesar de que sí actualice la posición de este recuadro. Esto puede ser un problema cuando la distancia de la persona a la cámara difiera bastante de la inicial ya que, si la persona se aleja mucho por

ejemplo, el recuadro que delimita a la persona seguirá siendo igual de grande, pese a que debería ser más pequeño. Esto hace que el descriptor sea cada vez menos fiable, puesto que extraerá características de objetos que no pertenecen a la persona y por tanto no será capaz de reconocerla, pudiendo ocasionar un fallo en el seguimiento. Esta desventaja se presentará como trabajo futuro en el último capítulo de la memoria de este Trabajo.

4. SEGUIMIENTO EN EL ESPACIO TRIDIMENSIONAL.

En el cuarto capítulo de esta memoria se describe el procedimiento llevado a cabo para realizar el seguimiento dentro del espacio tridimensional de la escena frente al vehículo, consistente en el método de discriminación de la persona, la localización del objeto móvil en el espacio, así como la implementación del nodo de seguimiento.

4.1. Descriptor de personas.

En este apartado se detalla el método implementado para discriminar a la persona a seguir del resto de objetos. Existen dos etapas de discriminación: una primera etapa inicial para detectar a la persona que será el objetivo del seguimiento y una segunda que constituye el seguimiento en sí.

4.1.1. Reconocimiento inicial.

Esta primera etapa consiste en detectar y reconocer a la persona a la que el vehículo seguirá. Para ello, se emplea el método de sustracción de fondo cuyo fundamento teórico se explicó anteriormente en esta memoria. En primer lugar, la cámara adquirirá una imagen de la persona quieta, colocada delante de la cámara. Una vez tomada la imagen, la persona se moverá ligeramente para que se detecte un cambio en la imagen, que corresponderá precisamente con la ubicación de la persona (ver Figura 33).

Figura 33: Movimiento inicial de la persona para la detección (ISA-UMA).

Esto nos servirá para obtener el recuadro que engloba a la persona sustrayendo el fondo y por tanto su ubicación de partida, además del descriptor HOG de la persona. La manera en la que se detecta a la persona se realiza simplemente mediante el movimiento de la persona a seguir. Es importante destacar que el movimiento principal en el plano de la cámara ha de ser el de la persona, además de que se sitúe más o menos centrada en dicho plano. De esta forma, no interferirán otros objetos móviles en la detección. Como se puede observar en la Figura 34, existen otros movimientos pequeños cuando se aplica la máscara de sustracción de fondo, pero al encontrar los contornos y no ser de un tamaño significativo

no afectan en la detección. Tal y como se comentó en el apartado 1.2 de objetivos, el sistema de reconocimiento y seguimiento ha de ser ligero a nivel de computación, puesto que existen otras tareas en el vehículo bastante costosas y por tanto nuestro sistema debe consumir los menos recursos posibles. Por tanto, no se ha recurrido a técnicas de aprendizaje profundo para la detección porque implicaría un mayor consumo de recursos.

Figura 34: Sustracción del fondo para detección de la persona.

Tal y como se comentó previamente, el descriptor que emplea el seguidor KCF es el Histograma de Gradientes Orientados, por lo que una vez esté delimitado el recuadro perteneciente a la persona, se inicializa el seguidor con la información de dicho recuadro, momento en el que extrae la información para formar el primer HOG de la persona.

En el sistema de seguimiento se van a utilizar dos descriptores HOG. Uno de ellos describe a la persona de frente y el otro de espaldas, por lo que habrá dos descriptores para el seguimiento. Por tanto, se utilizan dos seguidores basados en ambos descriptores para que el sistema de visión sea capaz de detectar y reconocer a la persona a seguir tanto si está de cara al vehículo como de espaldas. Esto último es lo más habitual, ya que la persona irá avanzando de cara en el sentido de la marcha. Por tanto, al inicio será necesario tomar imágenes de la persona tanto de frente como de espaldas. En la Figura 35 se muestra el recuadro que engloba a la persona en la inicialización, una vez sustraído el fondo. Cabe destacar que dicha imagen no corresponde a las cámaras estéreo que se utilizan para el seguimiento. No obstante, el método de identificación de personas y su seguimiento en la imagen es independiente de la cámara que se utilice.

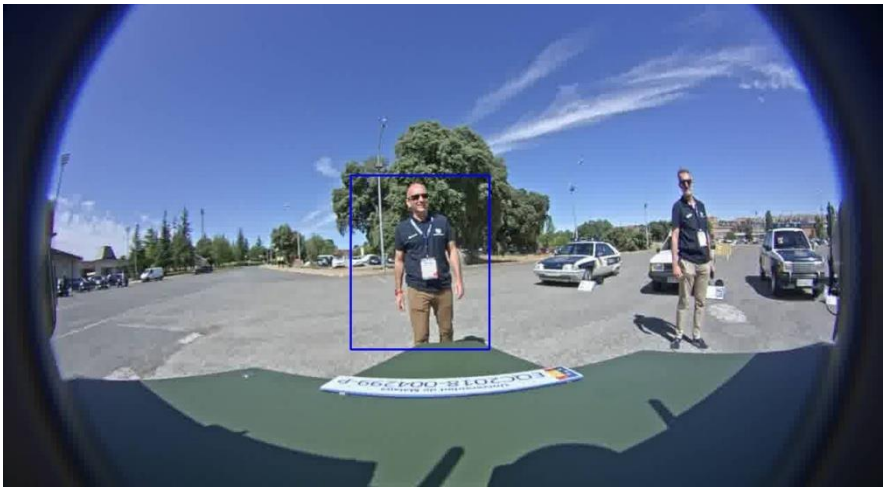


Figura 35: Reconocimiento inicial persona (ISA-UMA).

4.1.2. Seguimiento.

Una vez inicializados ambos seguidores con los descriptores de frente y de espaldas, se puede iniciar el movimiento. En este momento, el programa encargado del seguimiento entra en un bucle, actualizando la ubicación y descriptor de la persona en cada fotograma de forma recursiva.

A continuación se muestra una secuencia de imágenes donde el programa sigue a la persona correctamente (ver Figura 36), a partir del *dataset* suministrado por el grupo de investigación de robótica y mecatrónica de la UMA.

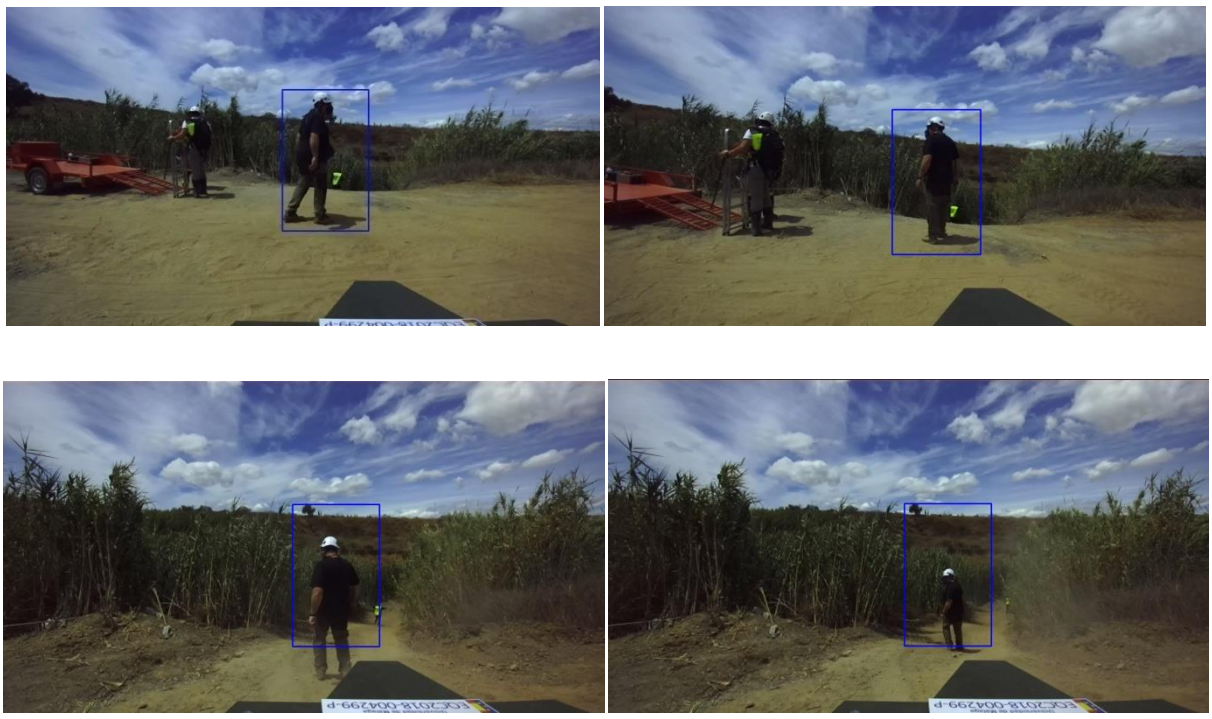


Figura 36: Secuencia seguimiento.

Tal y como se comentó anteriormente, en caso de que el seguimiento falle, la persona tendrá que detenerse y darse la vuelta y el programa cambiará de seguidor para volver a reconocer a la persona y continuar con la maniobra de rescate.

En la Figura 37 se muestra el flujograma que describe el proceso específico de reconocimiento y seguimiento llevado a cabo.

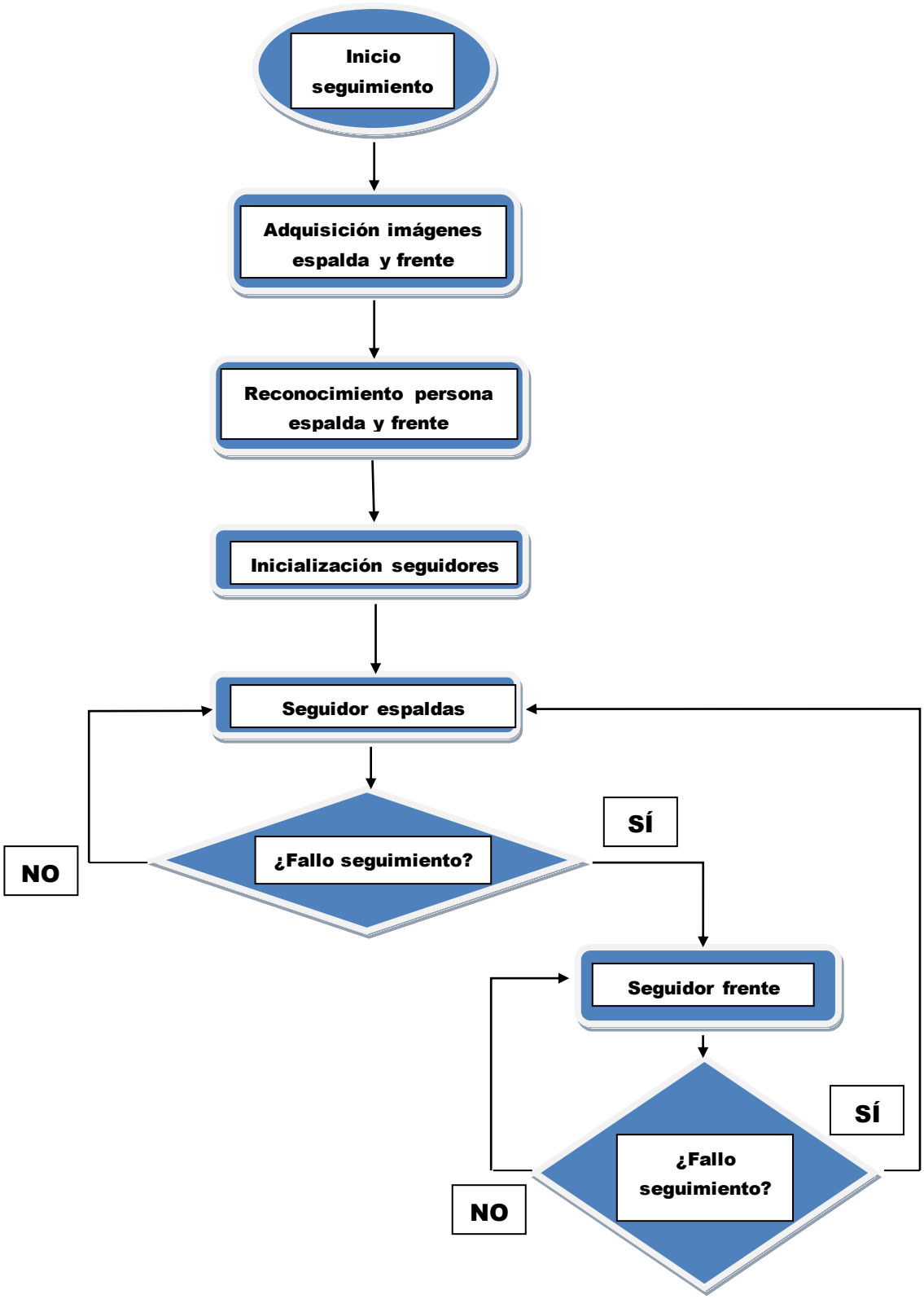


Figura 37: Flujograma específico reconocimiento y seguimiento.

En el capítulo 5 de la memoria se explicarán las pruebas realizadas y los resultados obtenidos en el seguimiento.

4.2. Localización del objeto móvil en el espacio.

Anteriormente hemos visto cómo se realizaba el seguimiento y se ubicaba a la persona dentro del espacio de la imagen. Con esta información, se ha de obtener la distancia a la que está la persona de la cámara y el grado de desviación respecto a ella. La profundidad nos servirá para saber cuánto de rápido se ha de mover el vehículo para intentar mantener una distancia segura más o menos constante respecto a la persona que está siguiendo. Por otro lado, la inclinación sirve para que el vehículo gire de tal forma que la persona siempre esté ubicada en el centro del vehículo (ver Figura 38).

Figura 38: Profundidad e inclinación de la persona.

4.2.1. Acceso a la información de la cámara.

En primer lugar, vamos a ver cómo se consigue encontrar la correspondencia de los puntos en una cámara estéreo. Una vez visto este fundamento teórico, se explicará la obtención de la información espacial de nuestra cámara estéreo.

- **Geometría epipolar.**

Como se ha comentado antes, para encontrar la correspondencia de los puntos en una cámara estéreo se emplea la geometría epipolar (Hartley & Zisserman, 2000).

Una cámara estéreo se compone de dos cámaras, donde cada una de ellas proyecta un plano. Si tenemos un punto *X* en el espacio tridimensional, éste se reflejará en las dos vistas, siendo *x* para la primera y *x'* para la segunda. Estos puntos y el centro de las cámaras comparten el mismo plano (ver Figura 39a). Dicho plano se denota con la letra π y se denomina plano epipolar.

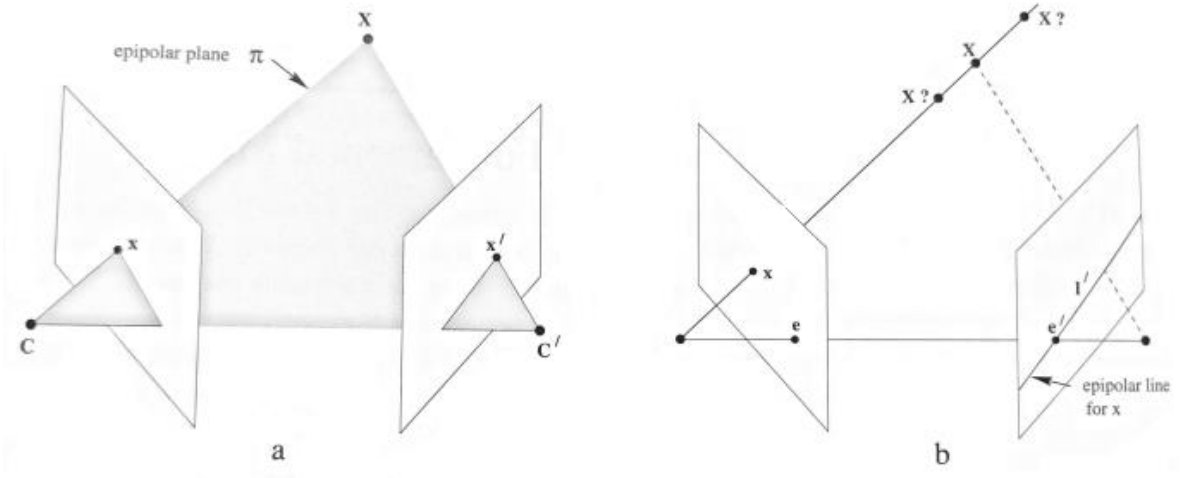


Figura 39: Correspondencia del punto en el espacio (Hartley & Zisserman, 2000).

Suponiendo que solo se conoce *x*, se pretende saber dónde se ubicaría el punto correspondiente *x'*. Por un lado, se sabe que el punto *x'* está ubicado en el plano epipolar π (ver Figura 39a). Por otro lado, dicho punto estará ubicado en la intersección de la línea *1'* y el plano π (ver Figura 39b). Dicha línea es la línea epipolar y se obtiene al unir el centro de la cámara izquierda *C* con el punto *x*, de tal forma que desde la cámara derecha se ve como una línea (*1'*) en el plano de su imagen. Sabiendo que la proyección de *X* (*x'*) ha de estar contenida en la línea epipolar, podemos saber si ambos puntos corresponden al mismo punto *X* en el espacio tridimensional gracias a esta restricción. La forma de saber si dos puntos corresponden al mismo punto tridimensional es la siguiente: si las dos líneas de proyección de ambos puntos intersectan con precisión en *X*, se puede calcular la posición de *X* a partir de las coordenadas de los puntos de cada imagen, a través de un proceso denominado triangulación.

Por tanto, para saber la posición en el espacio de un punto, se realiza un proceso iterativo como el mencionado anteriormente para todos los puntos que estén situados en la misma línea epipolar. En las imágenes estereoscópicas (ver Figura 40) habrá tantas líneas epipolares como filas de píxeles haya. De esta manera, se conoce la correspondencia entre los dos puntos de cada cámara que corresponderá a un único punto en el espacio. En cada cámara, las líneas epipolares son paralelas entre sí, y la dirección de dichas líneas

dependerá de la posición de una cámara respecto a otra. En nuestro caso, dado que las cámaras están dispuestas en el mismo plano, las líneas epipolares no tendrían ninguna inclinación, es decir son paralelas al eje horizontal de la cámara.

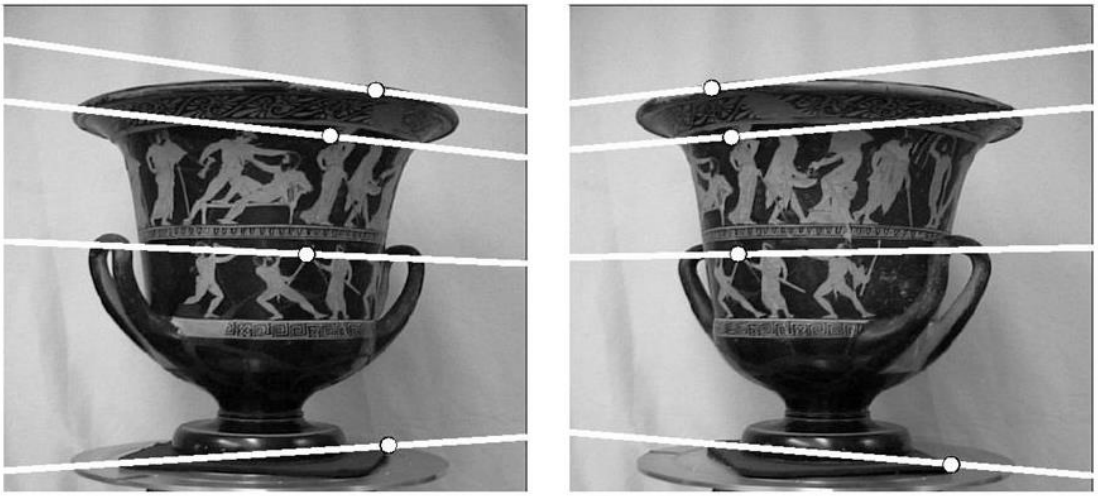


Figura 40: Cámaras convergentes (Hartley & Zisserman, 2000).

La información espacial se va obtener de la cámara estéreo ZED2. Se hará uso de la librería denominada *ZED ROS wrapper* (Inc, 2021). Esta librería contiene varios paquetes que nos permitirán obtener la información de profundidad y un mapa de nube de puntos a través de la suscripción a ciertos nodos mediante ROS. En nuestro caso, nuestro nodo se va a suscribir a tres nodos diferentes que se explican a continuación.

4.2.2. Imagen rectificada.

En primer lugar, se accede a la imagen rectificada de la lente derecha para adquirir la imagen que nos servirá para realizar todo el proceso de reconocimiento y seguimiento de la persona en dos dimensiones. La rectificación de la imagen es un proceso por el cual las imágenes proyectadas se convierten en un plano común de la imagen (ver Figura 41). Esta técnica es muy utilizada en visión estereoscópica, como es nuestro caso.

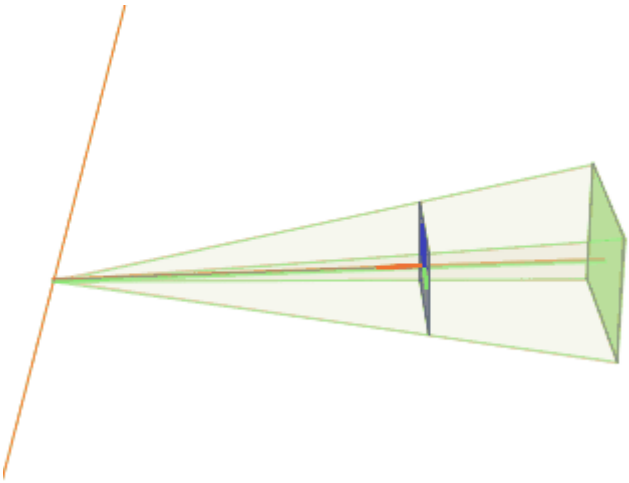


Figura 41: Rectificación imagen (Wikipedia).

4.2.3. Imagen de profundidad.

Con la imagen rectificada RGB previa obteníamos la ubicación de la persona dentro del espacio de la imagen. Dicha información la utilizamos para obtener la profundidad a la que está la persona, suscribiéndonos al mapa de profundidad registrado en la lente izquierda.

La información está almacenada en datos de coma flotante de 32-bit, indicando la distancia en metros. La forma de acceder a estos datos no se realiza de igual forma que cuando se accede a un píxel de una imagen, ya que el acceso a memoria de la estructura de la clase correspondiente a una imagen se realiza como si fuese una matriz. Sin embargo, en los datos de profundidad se acceden como un vector. Los datos de profundidad se reparten en dicho vector fila a fila con respecto a la correspondiente matriz de la imagen, tal y como se ilustra en la Figura 42.

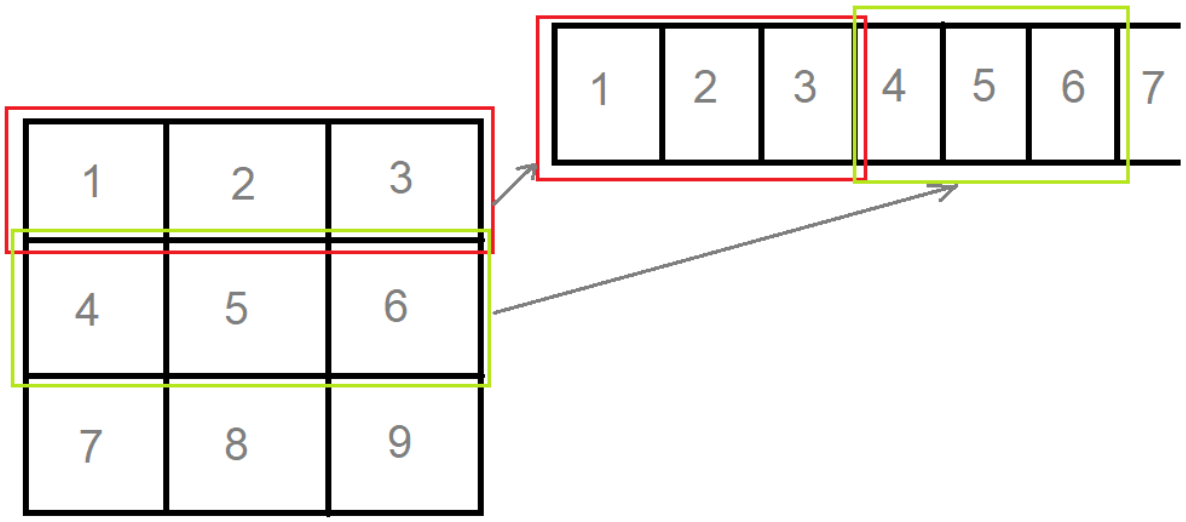


Figura 42: Correspondencia vector de profundidad con matriz imagen.

Por tanto, para hallar los valores de profundidad de interés, se aplicará la anterior correspondencia.

4.2.4. Nube de puntos.

Una nube de puntos es una representación en coordenadas *X*, *Y* y *Z* de la superficie externa de los objetos que existan en el campo de visión de la cámara. Por tanto, con la nube de puntos tenemos información tridimensional que nos permitirá obtener la profundidad e inclinación de la persona con respecto a la cámara accediendo a las distintas coordenadas (ver Figura 43).

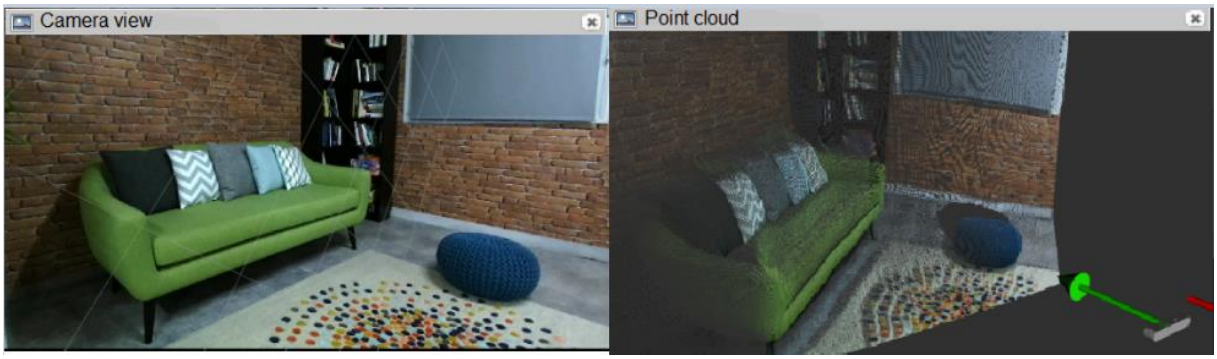


Figura 43: Nube de puntos [stereolabs].

En ROS existen dos tipos de datos a la hora de representar la nube de puntos: variable tipo *PointCloud* y *PointCloud2*. Este último tipo de variable es el que se obtiene cuando accedemos al nodo de la nube de puntos. El problema que presenta es que su tamaño varía en función del contenido de la imagen, luego no es posible acceder a un píxel en concreto fácilmente. Por el contrario, convirtiendo dicha variable a *PointCloud*, se tiene un vector de puntos de tamaño fijo, en el que la forma de acceder a un píxel en concreto y ver la posición en el espacio es idéntica a la comentada con los anteriores mapas.

4.2.5. Nodo de localización.

Conocidos cómo se presentan los datos de los diferentes nodos que suministra la cámara, vamos a ver el nodo de localización desarrollado, el cual tratará dichos datos y obtendrá de la manera más veraz posible la profundidad e inclinación de la persona.

Hemos visto que aplicando el seguidor KCF la localización de la persona se describe a través de un recuadro. Dicho recuadro no contiene únicamente a la persona, sino que dentro de él habrá otros objetos cercanos no pertenecientes a la persona que formarán parte del fondo de la imagen o serán otros objetos a diferentes distancias a la cámara. Por tanto, si accediéramos a todos los píxeles del recuadro para obtener la profundidad y orientación, se introduciría mucho error en la medida. El objetivo ahora será el de extraer la región de la

imagen donde se proyecta la persona en el plano de la imagen para localizarla correctamente.

4.2.6. Segmentación de la imagen.

La segmentación en una imagen consiste en dividir la imagen en regiones que mantienen una relación de similitud, como por ejemplo el nivel de intensidad, color, profundidad o movimiento.

El método empleado para la segmentación de la imagen es el *clustering*, el cual consiste en agrupar los píxeles que tienen características similares en grupos. La forma más simple para dividir los grupos (clústeres) es buscar que los criterios de *clustering* estén lo más optimizados posibles. El criterio más usado es el *clustering error*, el cual consiste en calcular para cada punto la distancia cuadrática al centro de cada clúster, y a continuación sumar todas estas distancias para todos los puntos de la imagen (Likas, Vlassis, & Verbeek, 2002).

Un método conocido que minimiza este error es el *clustering k-means* que ofrece *OpenCV*, el cual utilizaremos para nuestro sistema. Este método divide los datos en celdas Voronoi. De forma resumida, dichas celdas son el resultado de particionar el plano según un número especificado de objetos o semillas (Aurenhammer, 1991). Cada celda agrupa una región que está más cerca de la semilla de la celda que de cualquier otra semilla. Por tanto, las celdas que pertenezcan a una semilla tendrán características similares entre ellas.

A continuación, vamos a entrar en una definición más detallada del algoritmo de *k-means*. Supongamos que tenemos un conjunto de datos $X = \{x_1, \dots, x_N\}, x_n \in R^d$. El objetivo del *clustering* es el de dividir este conjunto de datos en *M* subconjuntos diferentes $C_1, \dots, C_M$, denominados clústeres. Como se comentó anteriormente, se pretende minimizar el *clustering error*, el cual se define como la suma de la distancia euclídea cuadrática entre cada punto x_i y el centroide m_k de cada clúster C_k que contiene x_i. Este criterio depende por tanto del centro de los clústeres, y se define mediante la siguiente expresión:

$$E(m_1, \dots, m_M) = \sum_{i=1}^{N} \sum_{k=1}^{M} I(x_i \in C_k) ||x_i - m_k||^2$$

Donde $I(X) = 1$ si x_i pertenece a C_K y 0 si no pertenece.

El algoritmo encuentra mediante un proceso iterativo soluciones óptimas locales con respecto al *clustering error* que hemos comentado. El método comienza ubicando el centro

de los clústeres en posiciones arbitrarias y en cada iteración se van desplazando para minimizar dicho error. La inicialización arbitraria de los centros da lugar a una desventaja que se debe a la sensibilidad de la posición inicial de los centros. Es por tanto que son necesarias varias iteraciones para alcanzar resultados fiables.

A modo resumen, en la Figura 44 se presenta el diagrama de flujo correspondiente al algoritmo de *k-means*.

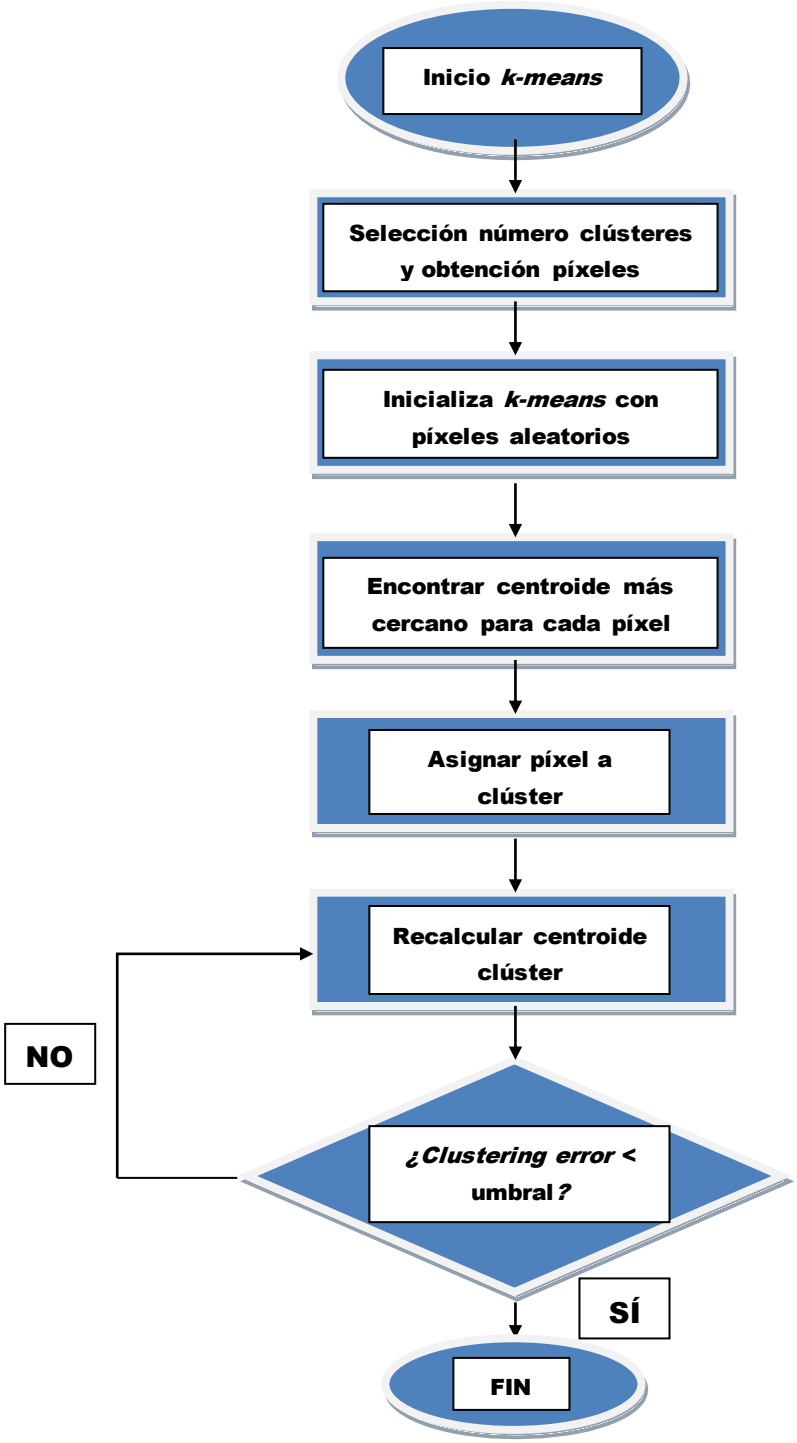


Figura 44: Flujograma k-means.

- **Implementación.**

Una vez explicados los fundamentos de cómo funciona el algoritmo *k-means*, vamos a ver la forma en la que se ha implementado en nuestro Trabajo.

Según el problema que se trata de resolver, segmentar a la persona en una imagen, se ha utilizado como característica básica para la segmentación la profundidad, debido a que la persona estará a una misma distancia a la cámara y se pretende eliminar el resto de la imagen que no pertenezca a la persona a seguir. Dicha característica básica se obtiene del mapa de profundidad que accedemos suscribiéndonos al nodo correspondiente. Es necesario normalizar los valores de dicho mapa de tal forma que tengamos una imagen en escala de grises, donde el nivel de intensidad esté relacionado con la profundidad. Dado que la distancia máxima que puede medir la cámara ZED2 es de 30 metros, utilizaremos este valor para normalizar los datos del mapa. Además, hay que tener en cuenta que no aplicaremos la segmentación a toda la imagen, sino únicamente al recuadro obtenido del seguimiento. De esta manera, los resultados serán mucho más fiables y rápidos computacionalmente, puesto que dentro de ese recuadro estamos seguros de que se encuentra la persona y solamente necesitamos eliminar aquellas partes del recuadro que no pertenezcan a ella.

Una vez tratada la imagen, se configura el algoritmo de *k-means*. En primer lugar, definimos el número de grupos en el que vamos a dividir la imagen. En nuestro caso, únicamente vamos a particionar la imagen en tres grupos. No se ha elegido un número elevado de clústeres para que no haya posibilidad de que se segmenten los propios píxeles pertenecientes a la persona en más de un grupo. Por otro lado, no es conveniente elegir un número pequeño de grupos porque se podrían incluir muchos píxeles que no pertenecieran a la persona. Según los resultados obtenidos en la segmentación, el número de grupos elegido es suficiente para segmentar la imagen correctamente.

En segundo lugar, definimos el criterio de finalización del algoritmo. En nuestro caso, hemos utilizado dos a la vez. El primero de ellos ya se ha mencionado con anterioridad y es el relativo al *clustering error*, con un valor de 0,0001. Si definimos un error demasiado pequeño puede ocurrir que el algoritmo realice muchas iteraciones y haga que nuestro programa se ralentice, es por ello que hemos incluido un segundo criterio. Este criterio limita el número de iteraciones máximo que puede realizar el algoritmo. Para nuestro Trabajo, el número de iteraciones se ha limitado a 10000. En la Figura 45 podemos ver en la imagen (a) el recuadro que contiene a la persona y sobre el cual se aplicará la segmentación. En la imagen (b) vemos el resultado de la partición después de haber aplicado la segmentación.

En ella podemos apreciar los 3 grupos diferenciados por color, donde la silueta de la persona ocupa la mayor parte del recuadro.

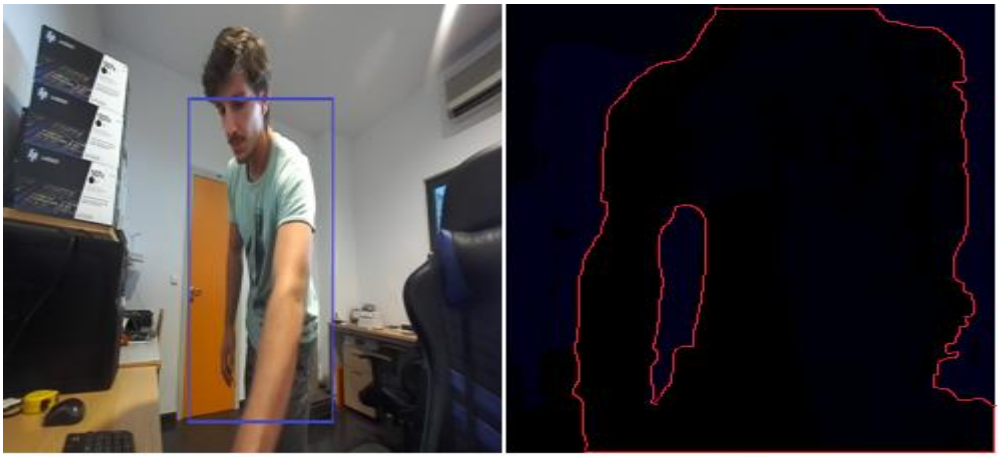


Figura 45: Imagen segmentada.

Una vez tenemos la imagen segmentada, debemos escoger el grupo de píxeles que pertenezca a la imagen. Como comentábamos anteriormente, una de las ventajas de aplicar la segmentación sobre el recuadro obtenido del seguimiento es que la mayoría de píxeles pertenecen a la persona. Aprovechándonos de esto, el clúster que describe la silueta de la persona será aquel que contenga mayor número de píxeles. Sabiendo esto, podemos aplicar dicha operación simple y obtener la silueta de la persona, tal y como se muestra en la Figura 46.

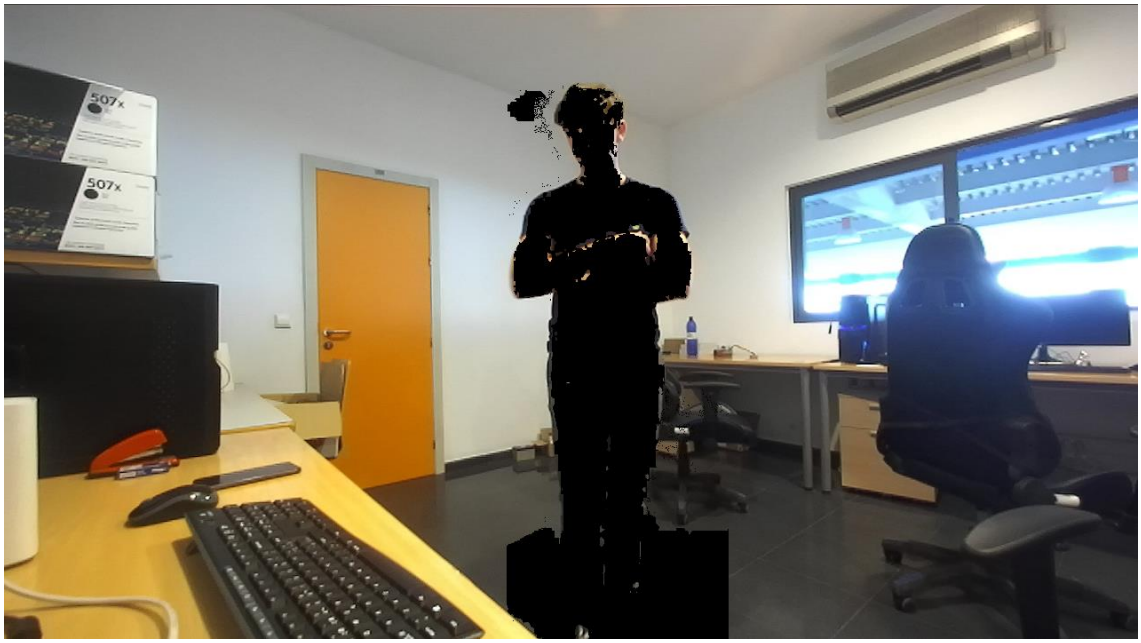


Figura 46: Resultado segmentación entorno laboratorio.

4.2.7. Acceso a la nube de puntos.

A continuación, se va a proceder a explicar el acceso a la nube de puntos para conocer la localización de la persona. Una vez hemos segmentado la imagen a través de *clustering*, sabemos qué píxeles pertenecen a la persona. Esto nos servirá para saber a qué posiciones de la nube de puntos acceder, pero antes debemos conocer qué información nos proporciona el nodo y cómo interpretarla.

Como comentamos en el apartado 4.2.4, vamos a trabajar con la variable tipo *PointCloud* que presenta un array de tamaño fijo. Cada posición del array contiene a su vez un array de puntos con 32 bits de precisión que representa la posición de dicho punto en el espacio libre. Accediendo a este punto, obtenemos las coordenadas *X*, *Y* y *Z* como flotantes. Dichas coordenadas están referenciadas con respecto al centro de la lente izquierda. La convención de signos y métrica de las coordenadas se puede encontrar en la documentación de ROS (ROS documentation) y es la siguiente:

- **X:** Positiva hacia delante en metros.
- **Y:** Positiva hacia la izquierda en metros.
- **Z:** Positiva hacia arriba en metros.

De forma gráfica, el sentido de los ejes se muestra en la Figura 47.

Figura 47: Convención ejes acceso nube de puntos.

Conocida la convención de ejes, estamos en disposición de obtener la localización de la persona. Para obtener la profundidad, guardamos todas las medidas relativas a la coordenada *X* que se obtienen de los píxeles de la segmentación y realizamos una simple media sobre ella. De forma análoga, obtenemos los valores de la coordenada *Y* para hallar la orientación de la persona, aunque hay que tratar dicho dato para saber el ángulo. Si formamos un triángulo rectángulo con las coordenadas, la coordenada *X* sería el cateto

contiguo y la coordenada *Y* el cateto opuesto del triángulo, tal y como se describe en la Figura 48.

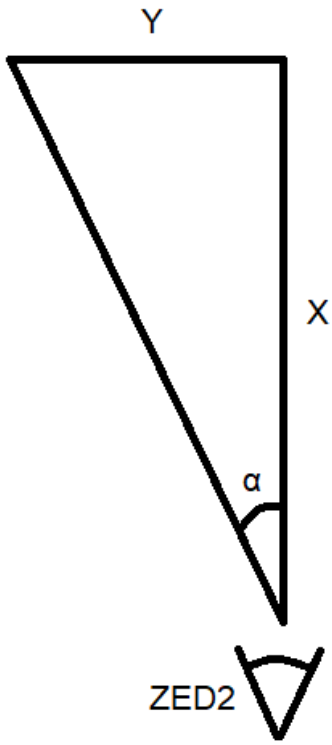


Figura 48: Relación coordenadas nube de puntos y orientación.

Por tanto, para obtener la orientación de la persona con respecto a la cámara, aplicamos simple trigonometría a través de la tangente:

$$\tan \alpha = \frac{Y}{X}$$

Existen varios métodos para obtener el dato global de la ubicación de la persona una vez tenemos todos los píxeles que la constituyen. En este caso, tal y como se ha comentado anteriormente, se ha realizado una simple media de todas las medidas. A continuación, se ha realizado una prueba en la que se toman todos los datos de profundidad y orientación de todos los píxeles que conforman la región de la imagen que corresponde a la persona para un instante de tiempo. La persona estaba situada a un metro de distancia de la cámara y 0º con respecto a ésta. Los histogramas de profundidad y orientación se muestran en la Figura 49, así como la media y varianza de cada magnitud, recogida en la Tabla 1.

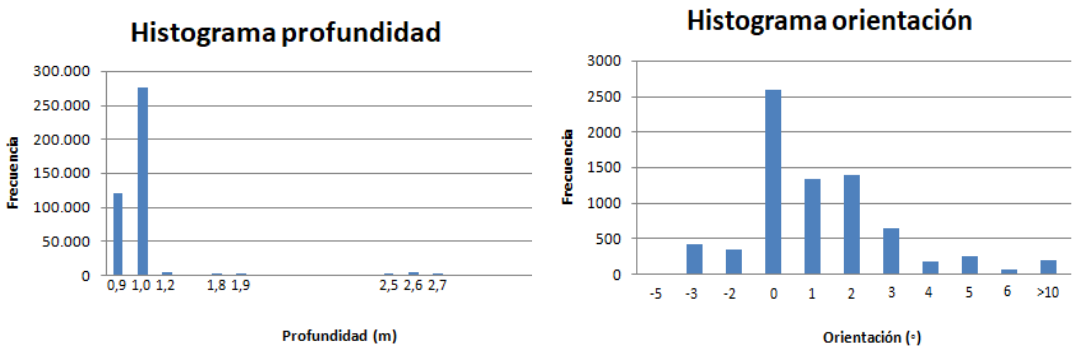


Figura 49: Histogramas profundidad y orientación en un instante de tiempo.

	Media	Varianza
Profundidad (m)	1,0166	0,0216
Orientación (º)	0,4912	1,2396

Tabla 1: Media y varianza de la profundidad y orientación en un instante de tiempo.

Viendo los resultados, no hay excesiva dispersión entre los datos y existe poca cantidad de *outliers* si analizamos los respectivos histogramas. También se probó a tomar la mediana de los datos en lugar de la media. Sin embargo, no fue adecuado su uso ya que el error (alrededor de 50 cm) que se obtenía respecto a la medida real era mayor que el obtenido con la media (17 mm).

Por tanto, la elección de la media es un buen método para obtener la localización de la persona, ya que no hay una dispersión significativa de los datos.

4.2.8. Caracterización de las medidas.

Con el algoritmo implementado de segmentación y acceso a la nube de puntos se han realizado pruebas dentro del laboratorio. Dichas pruebas consisten en, con la persona en estático y conocida la localización de la persona a priori, medir la profundidad y orientación de la persona de nuestro algoritmo y cuantificarla. En cada una de ellas se ha medido la media, varianza, intervalo de confianza al 95 % e histograma para la profundidad y orientación. Además, se hará una representación de la distribución de los puntos tomando como ejes de coordenadas la orientación, eje *Y*, y la profundidad, eje *X*. La caracterización de estas medidas se ha realizado a través del sistema de cómputo numérico *MATLAB R2021b*.

En la primera prueba, la persona se ha situado a un metro de distancia y a 0º respecto a la cámara. El histograma de la profundidad, orientación y la distribución de puntos se muestra en la Figura 50. Se ha obtenido una varianza para la profundidad de 0,018 m y 0,405º para la orientación. De todos los datos recogidos, la máxima desviación de una medida puntual es de 0,23 metros y 3,8º. No obstante, la dispersión de los puntos no es

significativa según observamos en la gráfica de dispersión de puntos, luego la mayoría de puntos se encuentran próximos a la media de todos los valores medidos. Además, la varianza de la profundidad y orientación son de un bajo valor.

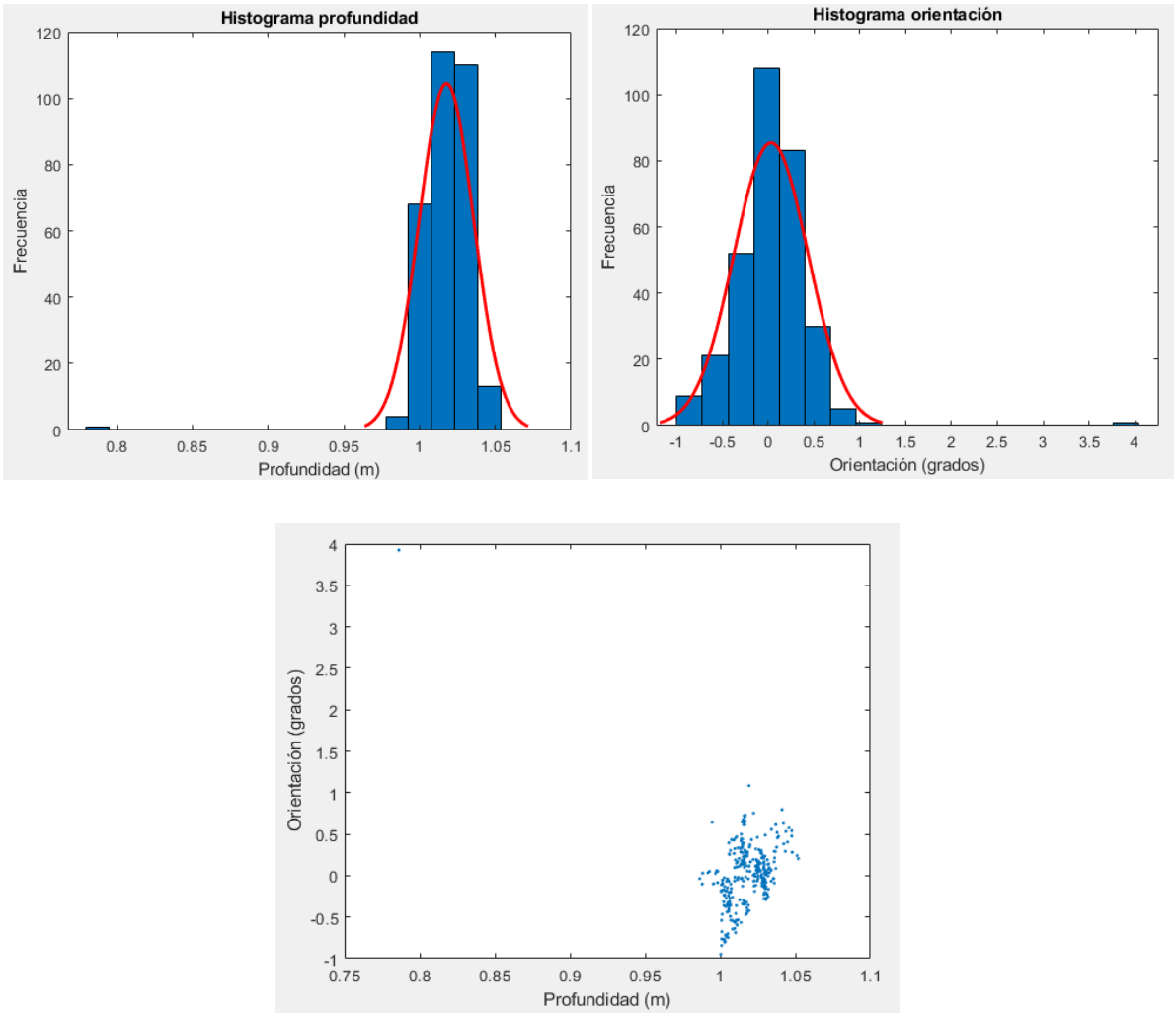


Figura 50: Gráficas primera prueba a 1 m y 0º.

A continuación, realizamos la misma prueba, pero en este caso con la persona situada a 1 metro de la cámara y a 10º de orientación. Los resultados se muestran en la Figura 51.

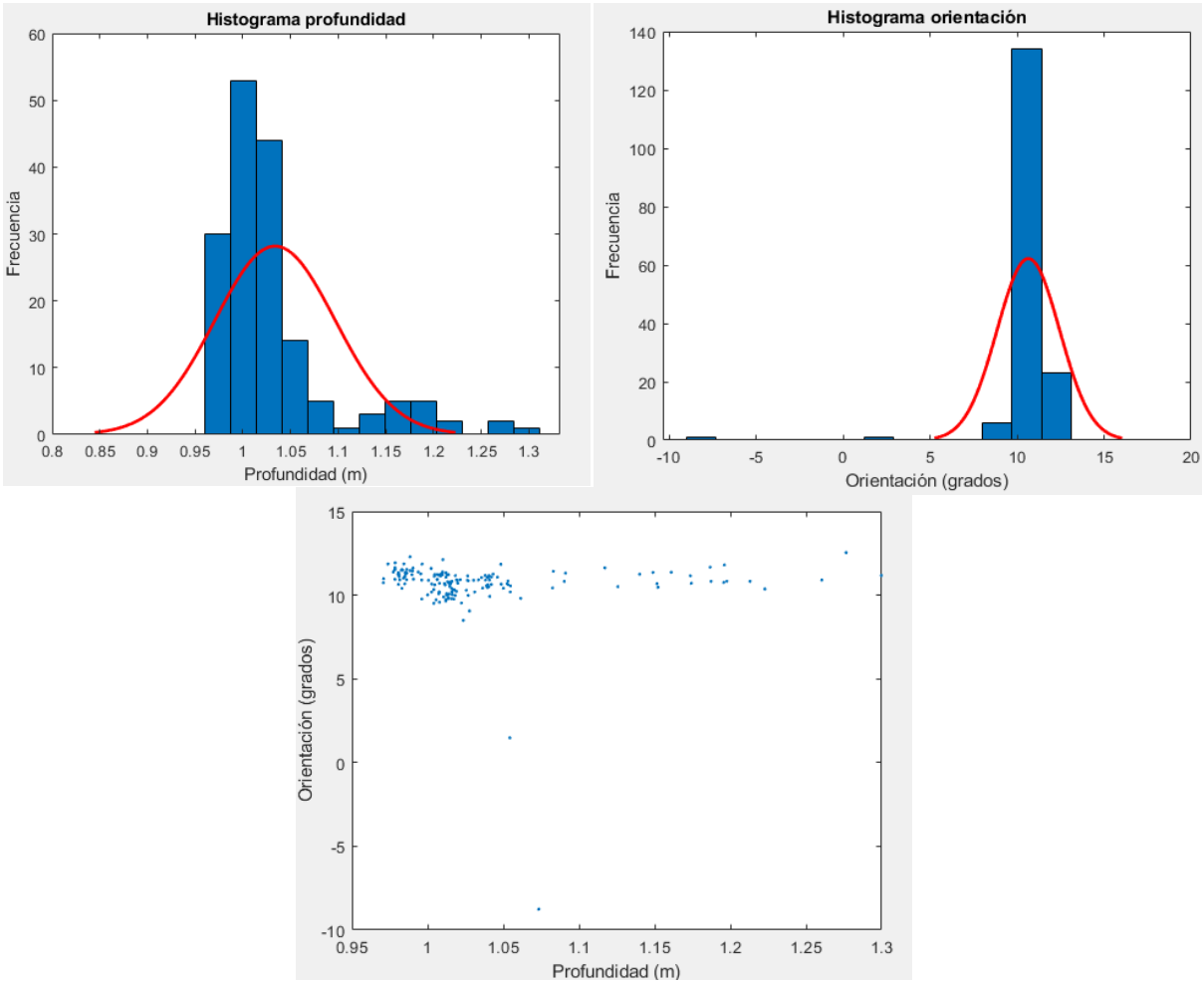


Figura 51: Gráficas primera prueba a 1 m y 10º.

En tercer lugar, la persona se situó a 2 metros de distancia respecto a la cámara y en frente de ella, es decir con una orientación de 0º. Los resultados se reflejan en la Figura 52.

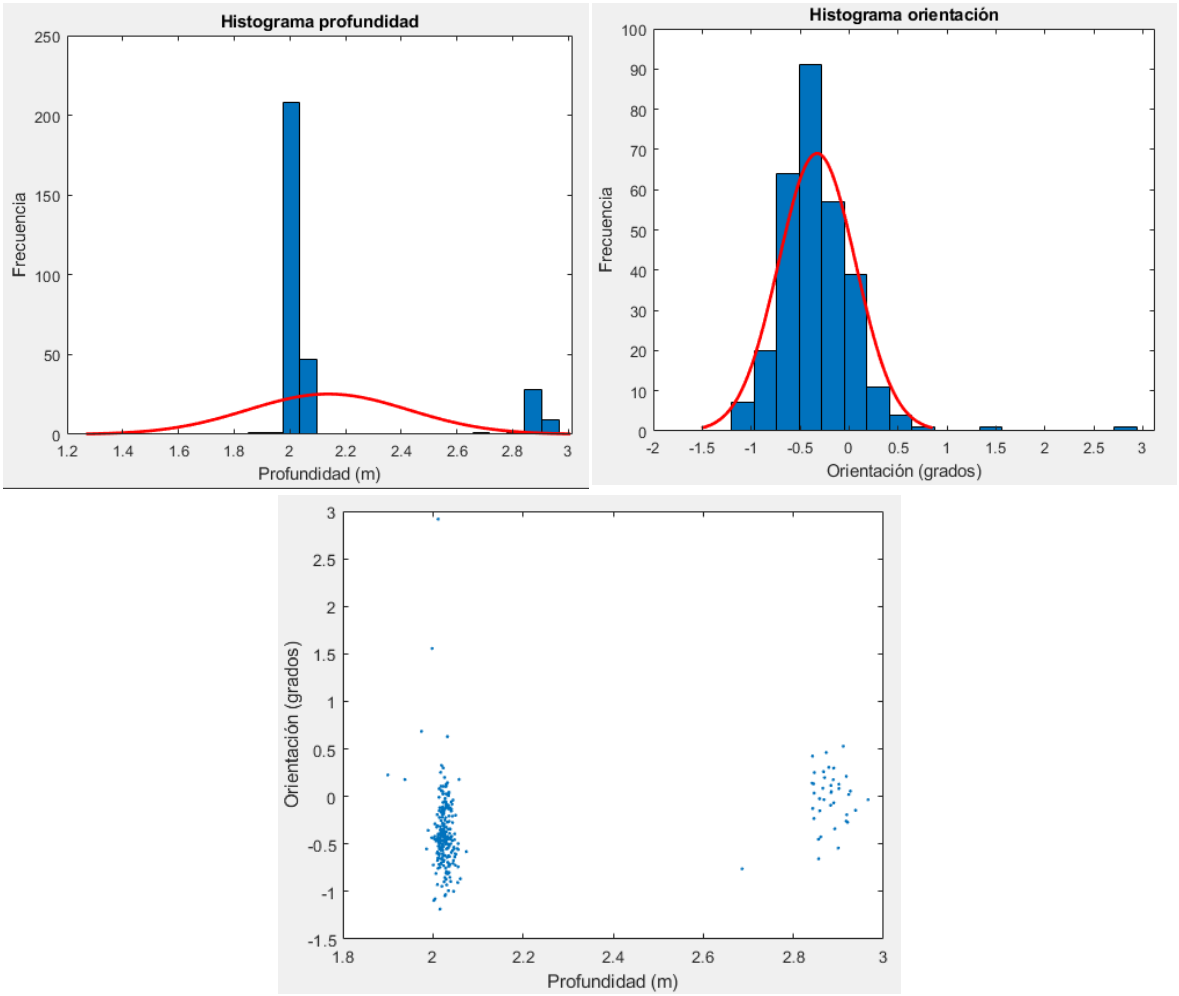


Figura 52: Gráficas primera prueba a 2 m y 0º.

En último lugar, la persona está situada a 1 metro de distancia y -20º de orientación. En esta última prueba ha sido donde peores resultados se han obtenido, los cuales se muestran en la Figura 53. Simplemente atendiendo a la gráfica de dispersión se puede observar la disparidad de las medidas obtenidas. Concretamente, le varianza de la profundidad es de 0,361 metros y la de la orientación de 12,843º. Se observa una mayor varianza de los datos en la orientación, donde la media de los datos obtenidos es de -10,978º cuando realmente la persona está a -20º.

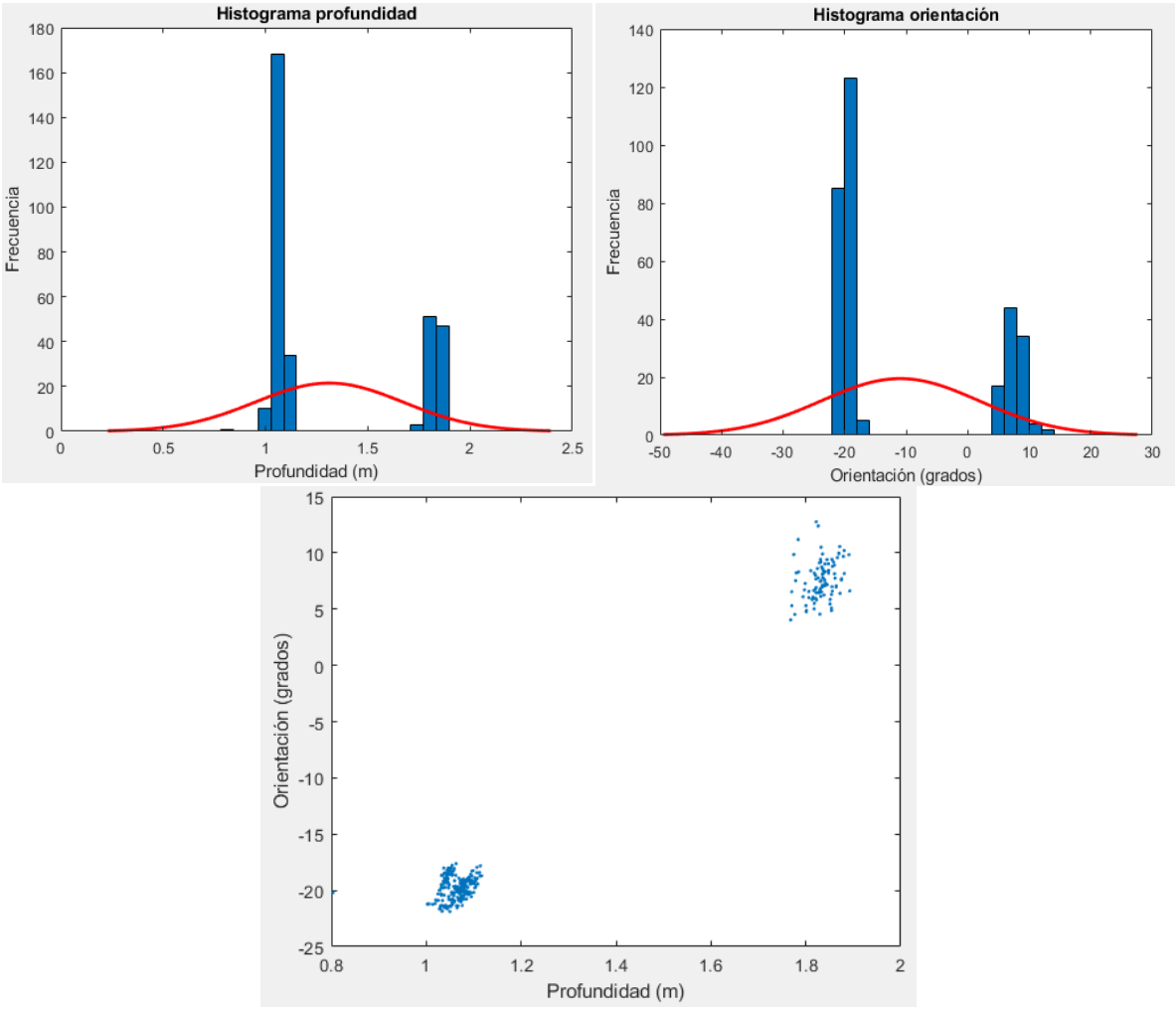


Figura 53: Gráficas primera prueba a 1 m y -20º.

Profundidad (m)	Orientación (º)	$\mu_{profundidad}$ (m)	$\sigma_{profundidad}$(m)	$\mu_{orientación}$ (º)	$\sigma_{orientación}$ (º)
1	**0**	1,018	0,018	0,030	0,405
1	**10**	1,034	0,063	10,647	1,799
2	**0**	2,138	0,289	-0,327	0,393
1	**-20**	1.311	0,361	-10,977	12,843

Tabla 2: Resultados primera prueba.

En la Tabla 2 se muestran los resultados obtenidos de las pruebas a modo de resumen. Analizando estos resultados podemos ver que cuando la persona se encuentra alrededor del centro del plano de la cámara, se obtienen unos resultados fiables, donde la varianza de los datos no es significativa. Sin embargo, conforme la persona se aleja de dicho centro y por tanto la orientación aumenta, la dispersión de los datos es mayor, especialmente en la medida de orientación. Es por ello por lo que se ha de implementar un mecanismo que disminuya esta variación de los datos para una misma posición y se obtengan por tanto unos datos más fiables.

4.2.4. Estimación de la localización mediante el Filtro de Kalman.

Para mejorar los resultados obtenidos con la segmentación, se va a aplicar el filtro de Kalman (Bar-Shalom, Li, & Kirubarajan, 2001) a los datos de posición. Este método también es usado para problemas de seguimiento, pero en este caso lo vamos a utilizar para mejorar la localización de la persona. Dicho algoritmo consta de dos procesos que se realizan de forma recursiva: predicción y corrección. A continuación, se estudia más en detalle cómo funciona el filtro de Kalman.

4.2.4.1. Estudio teórico del Filtro de Kalman.

- **Estimación de la dinámica.**

La idea básica del filtro de Kalman es la de estimar el estado de un sistema lineal dinámico en un tiempo *k*. Considerando el sistema discreto, se asume que el estado del sistema en un tiempo *k+1* evoluciona desde un estado *k* según la siguiente ecuación dinámica:

$$x(k+1) = F(k)x(k) + G(k)u(k) + v(k) \qquad k = 0,1, \dots$$

Donde *F(k)* es la matriz de transición que relaciona el estado anterior con el actual, *x(k)* es el estado del sistema, también denominado vector, *u(k)* define las entradas conocidas del sistema, *G(k)* es la matriz de transformación que se aplica a *u(k)* y *v(k)* es el ruido blanco gaussiano con covarianza *Q(k)*. Dicho ruido es aleatorio y se caracteriza por no existir relación entre sí a lo largo del tiempo. La función de densidad corresponde con una distribución normal, de ahí el nombre gaussiano.

La ecuación anterior va emparejada con el modelo de medida que se define a continuación.

$$z(k) = H(k)x(k) + \omega(k) \qquad k = 0,1, \dots$$

Donde *H(k)* es la matriz de transformación que convierte el estado en el dominio de medida y *ω(k)* representa la secuencia del ruido blanco gaussiano de medida con covarianza *R(k)*.

Se toman varias suposiciones que constituyen la suposición lineal gaussiana. Las matrices *F*, *G*, *H, Q* y *R* se toman como conocidas y posiblemente cambiantes en el tiempo. Por tanto, se considera que el sistema cambia con el tiempo y los ruidos son no estacionarios. El estado inicial *x(0)* se considera generalmente desconocido y se modela

como una variable aleatoria con distribución gaussiana. Por último, tanto las secuencias de ruido como el estado inicial se toman como mutuamente independientes.

4.2.4.2. Ecuaciones de Kalman.

Las ecuaciones de Kalman se dividen en dos grupos: las ecuaciones de predicción y de corrección.

- **Predicción.**

Las ecuaciones de predicción se encargan de proyectar el estado actual y la estimación del error de covarianza para estimar el próximo estado.

En primer lugar, la estimación del siguiente estado $\hat{x}(k+1|k)$ se calcula aplicando en la ecuación de estado, enunciada al principio de este apartado, el operador de expectación condicionado en Z^k. Operando, se obtiene el estado estimado.

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k) + G(k)u(k)$$

A continuación, la predicción del estado de la covarianza se define mediante la siguiente ecuación:

$$P(k+1|k) = F(k)P(k|k)F(k)' + Q(k)$$

Donde *P(k|k)* es la matriz de covarianza del estado previo y *Q* el ruido de la covarianza del proceso.

- **Corrección.**

Por otro lado, las ecuaciones de corrección mejoran el estado siguiente predicho anteriormente incorporando una nueva medida en dicho estado.

La ganancia del filtro se define como:

$$W(k+1) = P(k+1|k)H(k+1)'S(k+1)^{-1}$$

Para corregir el estado estimado $\hat{x}(k+1|k+1)$ se mide la medida residual, también llamada innovación *v(k+1)*. La innovación es la diferencia entre la medida real *z(k+1)* y la medida estimada previa $\hat{z}(k+1|k)$. Finalmente, la estimación del estado corregido se obtiene sumando la estimación del estado proyectado $\hat{x}(k+1|k)$ y el producto de la ganancia del filtro y la medida residual.

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + W(k+1)v(k+1)$$

Por último, la covarianza actualizada en el estado *k+1* es:

$$P(k+1|k+1) = [I - W(k+1)H(k+1)]P(k+1|k)$$

- **Resumen.**

A modo de resumen, el filtro de Kalman necesita asumir un estado inicial y una covarianza, para saber la estimación inicial y la precisión de dicha estimación, respectivamente. A continuación, se proyecta la evolución del sistema en función de una entrada conocida y una perturbación aleatoria. Por último, usando la medida actual, se corrige la predicción del estado siguiente.

4.2.4.3. Implementación Filtro de Kalman.

La función del filtro de Kalman dentro de nuestro sistema será el de minimizar los datos erróneos de localización. Dentro del espacio de ROS, existe un paquete denominado *robot_localization* que implementa el filtro de Kalman en uno de sus nodos. Este nodo utiliza el filtro para proyectar el estado de la localización en el tiempo a través de un modelo de movimiento y corrige dicha estimación con los datos suministrados. En nuestro caso, dichos datos serán la profundidad (coordenada *X* de la cámara estereoscópica) y desplazamiento horizontal (coordenada *Y*). El proceso de instalación del nodo se detalla en el apartado 7.1.1 de los anexos. A continuación, vamos a ver qué datos se han de configurar para el filtro de Kalman, así como del dato que envía del sensor, en nuestro caso nuestro nodo de seguimiento.

- **Ruido de la covarianza del proceso.**

Esta matriz representa el ruido que añadimos al error total después de realizar la predicción. Cuanto mejor se aproxime nuestro modelo de movimiento al sistema, la matriz puede tener valores menores. No obstante, configurar estos valores correctamente no es sencillo. La documentación del paquete en cuestión de ROS (ROS documentation) aconseja aumentar el valor de las posiciones de la matriz correspondientes a la variable que estamos midiendo. Dado que únicamente enviamos las coordenadas *X*, *Y* y *Z*, solo aumentaremos el valor de estas posiciones en la matriz. De esta manera, el error predicho del filtro será mayor y confiará más en la medida entrante durante la corrección y por tanto convergerá antes al valor de la medida.

- **Sistemas de referencias.**

ROS define cuatro sistemas de referencias para plataformas móviles: *base_link*, *odom*, *map*, *earth*. En primer lugar, *base_link* está rígidamente fijado a la base del robot móvil. Los sistemas *map* y *odom* están fijados al mundo y alineados con la posición de inicio del robot. Por último, *earth* se utiliza como referencia para otros sistemas. En esta aplicación, este sistema no es relevante.

En caso de utilizar únicamente el filtro para la estimación de la posición, como es nuestro caso, tanto *world_frame* como *odom_frame* tendrán el mismo valor. Esto permitirá que el nodo de estimación funcione en su configuración por defecto.

- **Tratamiento de los datos del sensor.**

A continuación, vamos a ver cómo enviar el dato de las medidas tomadas por nuestro nodo de seguimiento al nodo del filtro de Kalman. Dicho nodo ha de recibir en un mensaje las posiciones de la persona localizada a través de odometría. El sistema de odometría contiene una estimación local precisa de la posición y velocidad del robot. En nuestro caso solo nos interesa la posición, por lo que la velocidad será obviada.

Principalmente, solo es necesario que el mensaje de odometría contenga dos parámetros: posición y covarianza. La posición que se envía al nodo del filtro contiene las coordenadas *X*, *Y* y *Z* de la posición de la persona. Realmente, solo nos interesa conocer las posiciones en *X* e *Y* para calcular la profundidad y orientación de la persona, pero para que la estimación del movimiento se realice correctamente es necesario que se envíen las tres coordenadas. Por otro lado, la covarianza nos permitirá limitar el error máximo que asumimos que tiene nuestro sistema de medida. De esta forma, eliminaremos cualquier medida estimada respecto a la anterior que supere dicho error.

Una vez el filtro de Kalman estima la posición de la persona, nuestro sistema de seguimiento obtiene dicha estimación suscribiéndose al nodo correspondiente. Con los nuevos datos, se obtiene la profundidad y orientación tal y como se explicó en el apartado 4.2.7.

4.2.4.4. Caracterización de las medidas.

De nuevo, se han realizado una serie de pruebas en estático dentro del laboratorio similares a las del apartado 4.2.8 para observar si la introducción del filtro de Kalman ha mejorado nuestro sistema. En primer lugar, se sitúa la persona a 1 metro de distancia de la

cámara y a 10º respecto a ésta. Las gráficas obtenidas se muestran en la Figura 54. Para esta situación, se ha obtenido una varianza en la profundidad de 0,028 m y 0,797º para la orientación. Además, observando los histogramas de la Figura 54, los datos están bastante agrupados, obteniendo pocos valores fuera de rango.

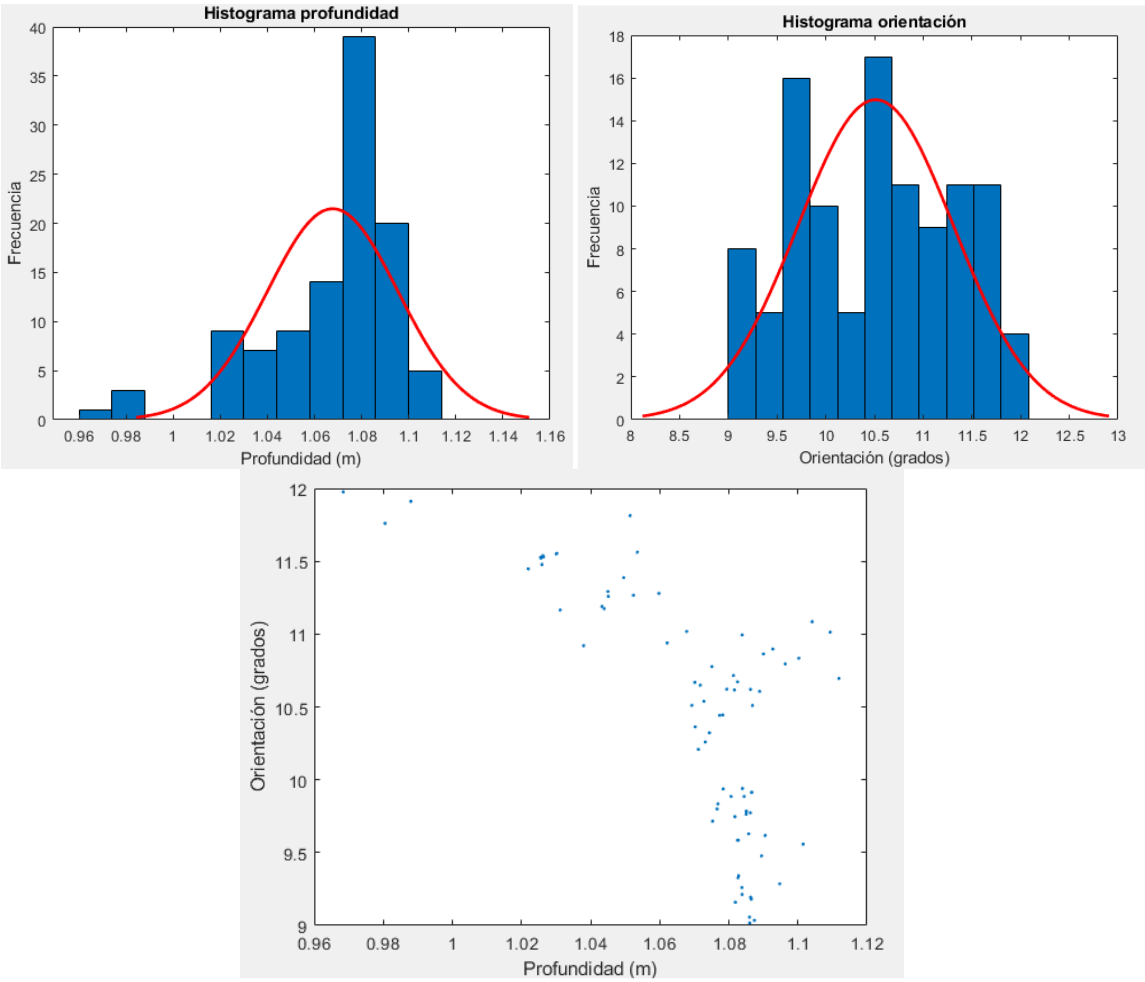
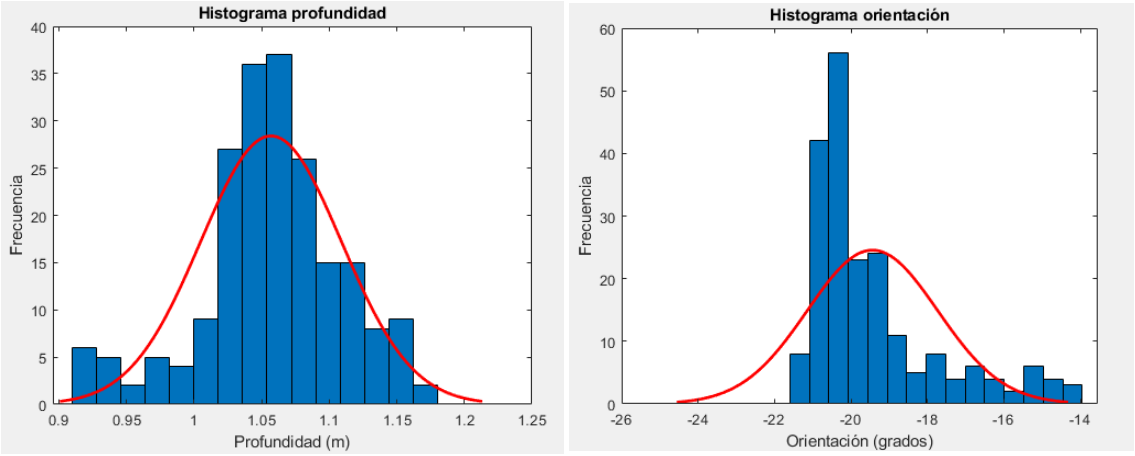


Figura 54: Gráficas segunda prueba a 1 m y 10º.

En segundo y último lugar, se quiso replicar la prueba de la que se obtuvo peores sin filtro de Kalman para verificar si había habido alguna mejora. Es decir, la persona se situó a 1 metro de distancia y a -20º respecto a la cámara. Observando los resultados obtenidos en la Figura 55, se obtiene una varianza de 0,052 m para la profundidad y 1,706º en la orientación. Además, la media de la orientación de los medidas realizadas es de -19,437º, por lo que comparado con la media obtenida sin filtro (-10,978º) la mejora es notable. A modo resumen, la Tabla 3: Resultados segunda prueba.Tabla 3 muestra los resultados obtenidos para esta segunda aplicando el filtro de Kalman.



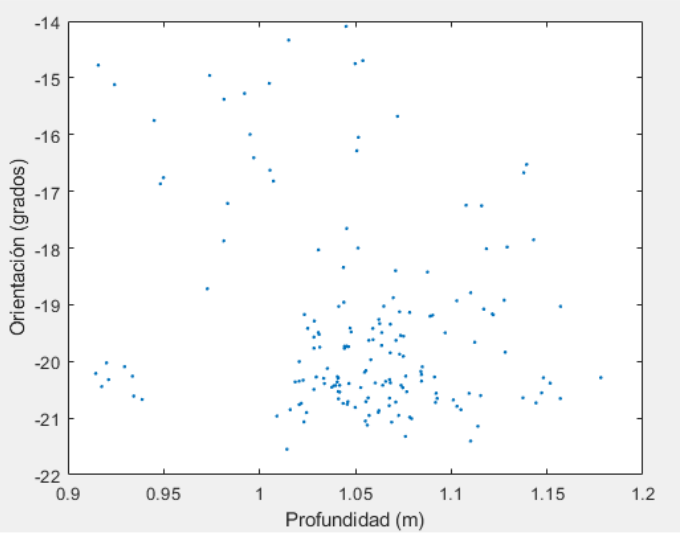


Figura 55: Gráficas segunda prueba a 1 m y -20º.

Profundidad (m)	Orientación (º)	$\mu_{profundidad}$ (m)	$\sigma_{profundidad}$(m)	$\mu_{orientación}$ (º)	$\sigma_{orientación}$ (º)
1	**10**	1,067	0,028	10,513	0,798
1	**-20**	1,057	0,052	-19,437	1,706

Tabla 3: Resultados segunda prueba.

De los resultados mostrados podemos ver que la varianza de los datos ha disminuido considerablemente, por lo que la dispersión de los datos es mucho menor. Además, el error de las medidas respecto al valor real se ha reducido significativamente, especialmente cuando la orientación es mayor. No obstante, al igual que ocurría sin el filtro de Kalman, cuando la persona está más alejada del centro de la cámara hay más datos fuera de rango, pero con el filtro utilizado hay mucha menos proporción de ellos gracias a la estimación del movimiento. Por tanto, podemos confirmar que la inclusión del filtro de Kalman es favorable.

A modo de resumen, en la Figura 56 se muestra el diagrama de bloques de los diferentes nodos relacionados en nuestro sistema de seguimiento.

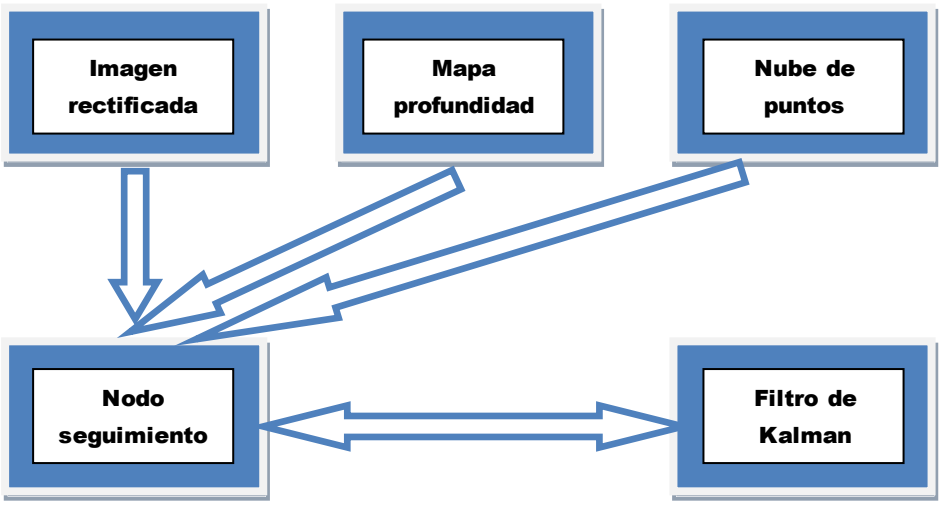


Figura 56: Diagrama de bloques suscripción y publicación nodos.

5. EXPERIMENTACIÓN Y RESULTADOS.

En este apartado se detalla el montaje llevado a cabo para realizar los experimentos del sistema de seguimiento, así como los resultados obtenidos de dichas pruebas.

Tenemos que tener en cuenta que el sistema de seguimiento desarrollado no se ha implementado dentro del sistema de conducción del vehículo. El movimiento del vehículo en los experimentos ha sido mediante el modo de seguimiento del vehículo basado en el LiDAR. De esta forma, se trata de evaluar las situaciones que se pretenden resolver mediante el seguimiento basado en cámaras. Además, para hacer una implementación real se debería dotar a nuestro sistema de mecanismos de seguridad y realizar muchas más pruebas para asegurar así su correcto funcionamiento, al estar en un entorno con seres humanos.

5.1. Montaje.

En primer lugar, se montaron los dispositivos necesarios para el seguimiento mencionados en el apartado 2.2 en el vehículo J8. Dicho montaje se llevó a cabo en el taller del departamento de Ingeniería de Sistemas y Automática. La cámara estereoscópica ZED2 se montó en la parte delantera del vehículo (ver Figura 57) junto a los dispositivos LiDAR que el vehículo ya contenía. Además, se colocó el router 5G que nos servirá para poder conectarnos a distancia a la Jetson Xavier a través del protocolo SSH.

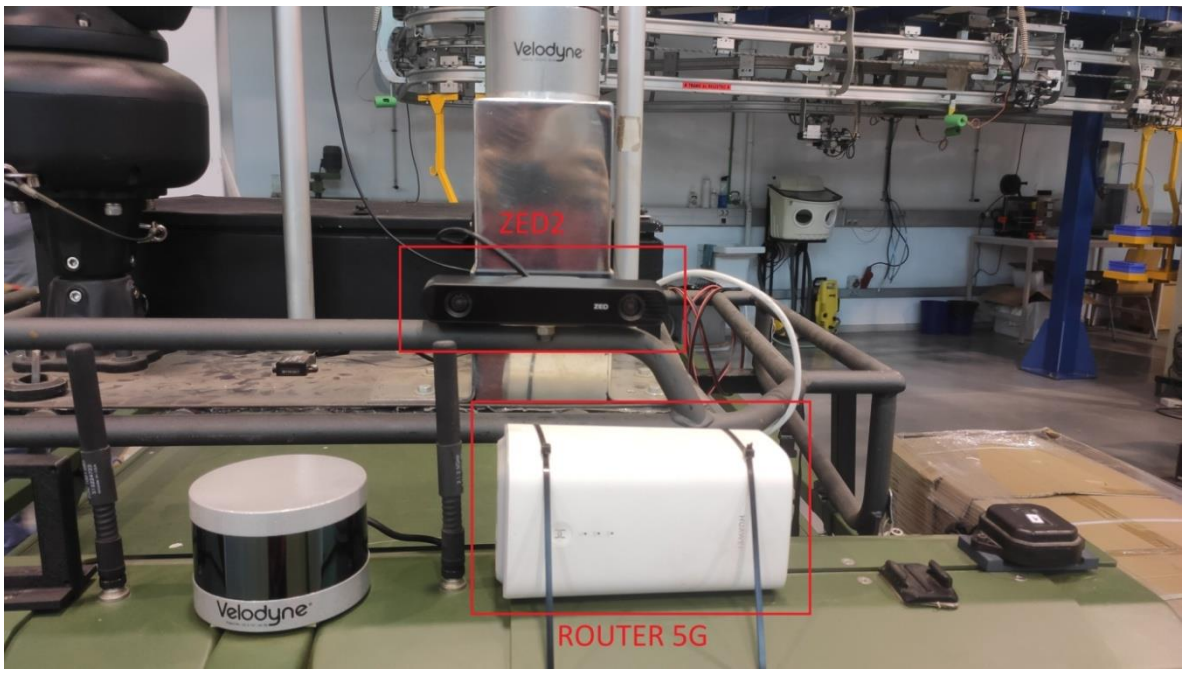


Figura 57: Ubicación cámara ZED2 y router 5G.

Por otro lado, colocamos la Jetson Xavier en la plataforma de sensores del vehículo como se muestra en la Figura 58. Dicha plataforma posee una tapa que nos permitirá proteger a nuestra CPU del entorno en las pruebas que se realicen. Conectaremos la cámara ZED2 mediante USB a la Jetson y el router 5G mediante un cable de red. Por último, alimentaremos tanto la CPU como la Jetson a través de un generador de continua.

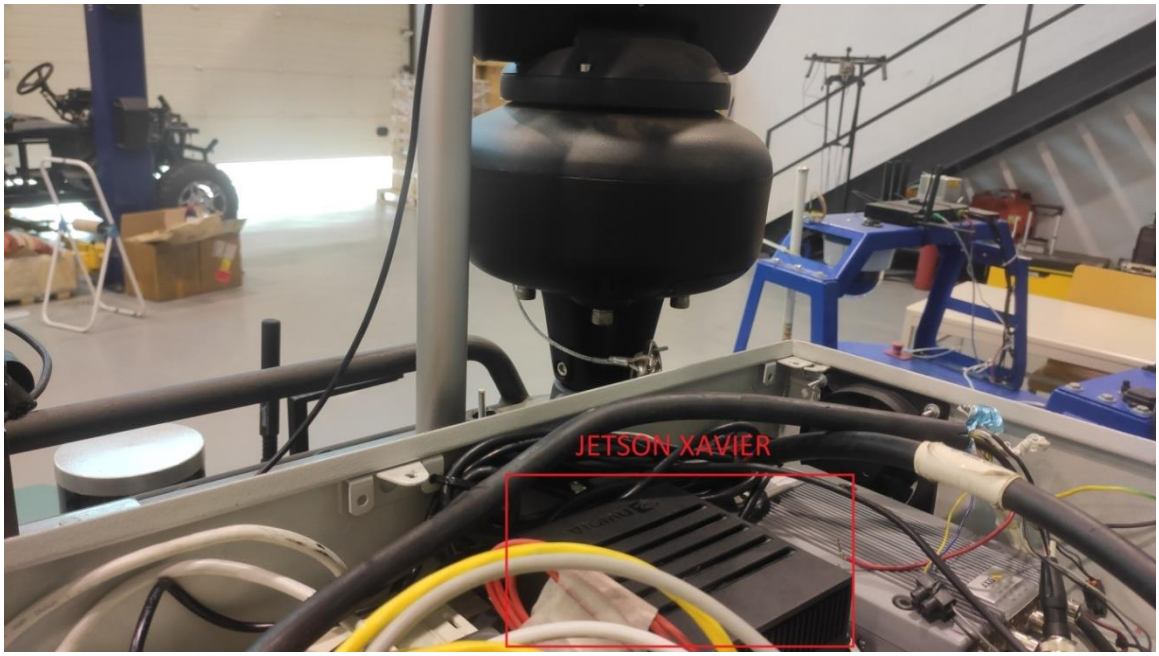


Figura 58: Montaje NVIDIA Jetson AGX Xavier.

A continuación, en la Figura 59 se muestran todos los dispositivos montados en el vehículo J8 en su conjunto.

Figura 59: Visión global J8.

5.2. Pruebas de funcionamiento.

A continuación, se explicarán las pruebas realizadas para observar el funcionamiento del sistema de seguimiento. Tal y como se explicó al comienzo de este apartado, la información de localización de la persona no se envía al vehículo para que éste se mueva. El vehículo J8 se moverá gracias al sistema *follow-me* que trae consigo implementado el propio vehículo. Por tanto, el estudio consistirá en analizar los mensajes de localización que nuestro nodo de seguimiento obtiene mientras que el vehículo se está moviendo, simulando lo que ocurriría si nuestro nodo se integrara dentro del vehículo. Para todas las pruebas se realiza un reconocimiento inicial de la persona en estático para establecer el objetivo del seguimiento.

Los experimentos realizados se han tenido en cuenta para verificar las cuestiones que se pretenden mejorar con este nuevo sistema de seguimiento con cámaras con respecto al que existía anteriormente basado en LIDAR, tal y como se detalló en el apartado 1.3. Por tanto, en las pruebas analizaremos si el sistema es capaz de seguir a la persona cuando ésta realice giros bruscos o en presencia de obstáculos. Por último, verificaremos si el nodo de seguimiento es capaz de discriminar entre la persona objetivo y otras que se encuentren dentro del campo de visión de la cámara.

Se han llevado a cabo dos pruebas para verificar los requerimientos anteriores. En la primera de ellas hay una única persona en todo el trayecto, mientras que en la segunda hay dos personas y el sistema deberá ser capaz de discriminar entre la persona objetivo y la otra. Dichas pruebas se detallan a continuación.

5.2.1. Prueba con una persona.

En esta primera prueba solo va a haber una persona en el trayecto a la que nuestro nodo de seguimiento deberá de seguir. En esta prueba, nuestro sistema fue capaz de seguir a la persona durante el tramo recto (ver Figura 60, a), pero comenzó a dar datos erróneos de localización cuando la persona realizó un giro de 90º, como se muestra en la Figura 60, b.

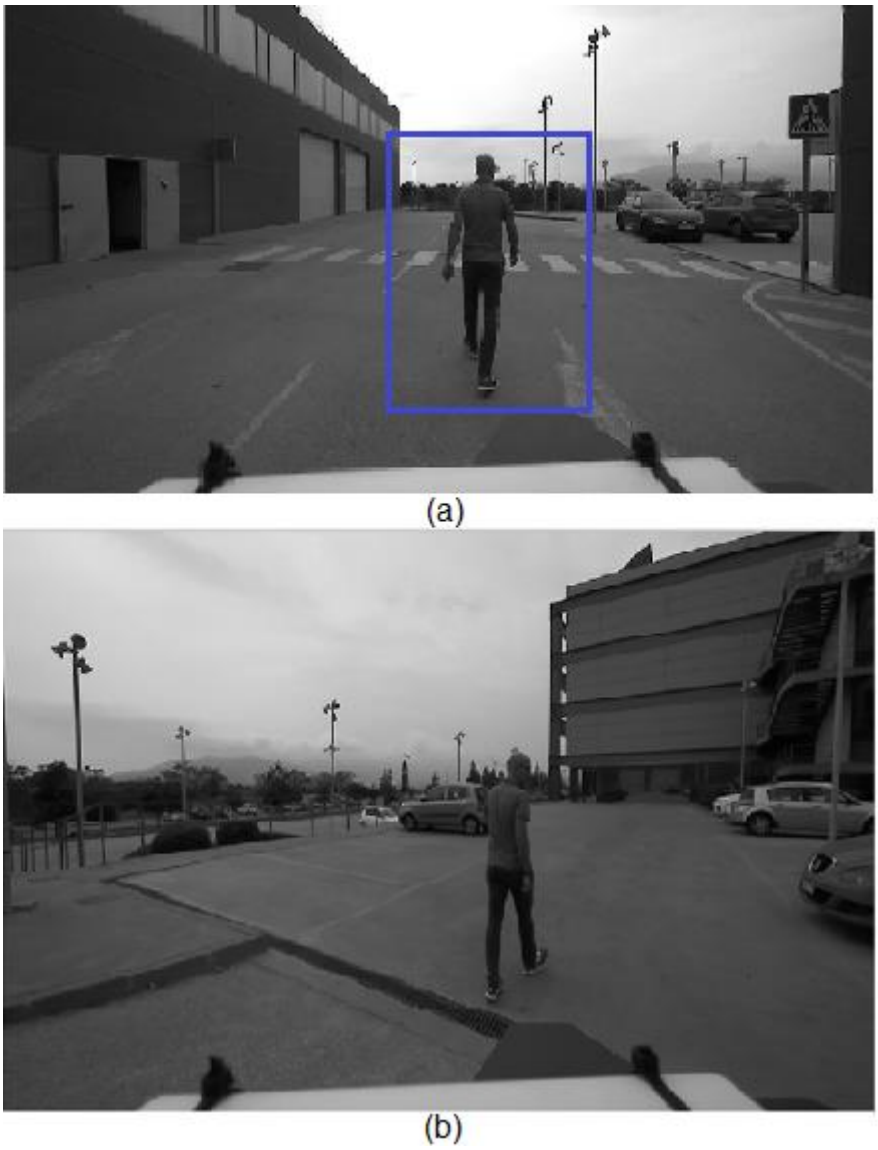


Figura 60: Fotogramas primera prueba.

En primer lugar, vamos a analizar los datos obtenidos cuando el sistema sigue a la persona correctamente. Las gráficas obtenidas se recogen en la Figura 61. Analizando dichas gráficas, podemos asegurar que la medida de profundidad y orientación son correctas. Existe una variación en la profundidad que es debido principalmente al propio movimiento del vehículo, ya que éste no está a la misma distancia en todo momento. Además, observando los datos de orientación vemos cómo al final del recorrido la persona se desplaza a la derecha, tal y como veíamos en la Figura 60, b. A partir de ese momento, el seguidor comienza a dar datos erróneos de localización por lo que no se han mostrado en las respectivas gráficas.

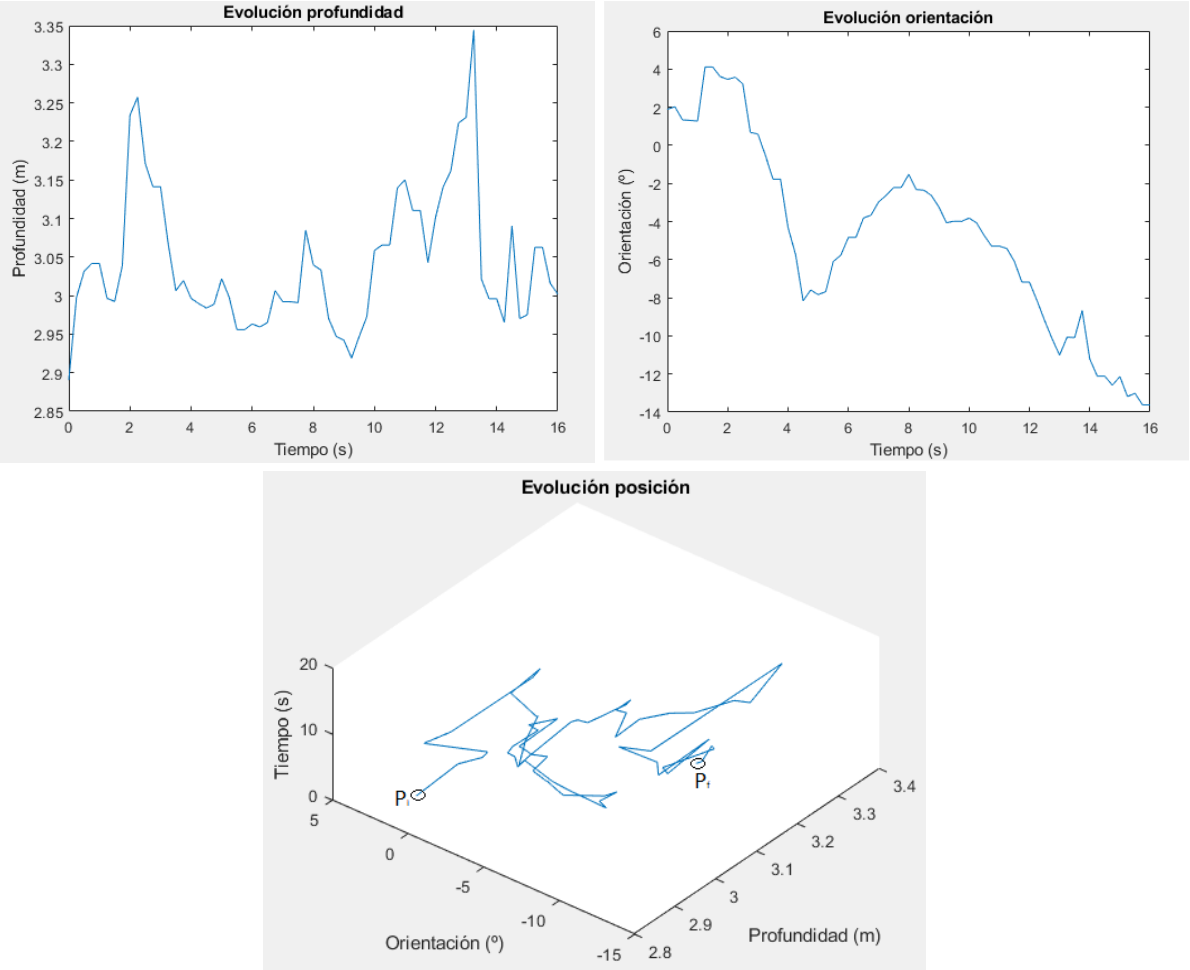


Figura 61: Evolución localización prueba con una persona.

Centrándonos en el fallo del seguidor, sabemos que éste ya no está siguiendo a la persona porque la medida que envía de profundidad es en torno a 8 metros. Cabe destacar que el sistema no ha detectado que haya perdido el objetivo, ya que seguía suministrando datos. Esto se debe posiblemente a que cuando se realizó el reconocimiento inicial, el recuadro que delimita a la persona fuera significativamente más grande, por lo que el descriptor HOG tendría más información que no pertenece a la persona. Esto provoca que si en algún momento la persona se sale algo del recuadro y el seguidor sigue actualizándose con aún más información que no es de la persona se pierda la referencia completamente, como ocurrió en esta prueba.

5.2.2. Prueba con dos personas.

A continuación, se realizó una prueba en la que durante el trayecto una persona externa (ver Figura 62, b) se interpondría en la trayectoria de la persona a seguir (ver Figura 62, a) para verificar que el sistema es capaz de discriminar a la persona.

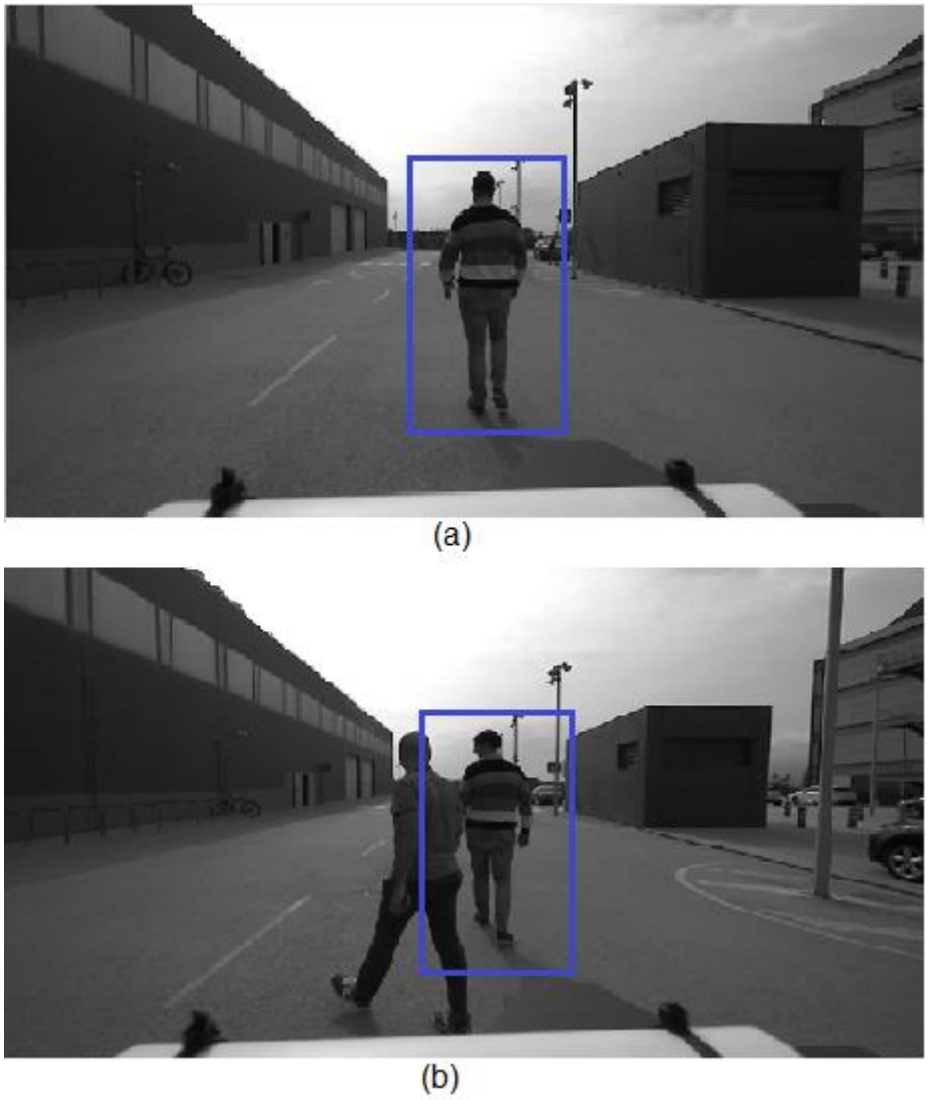


Figura 62: Fotogramas segunda prueba.

Tal y como podemos observar en las gráficas de la Figura 63, el sistema sigue al objetivo aunque otra persona se cruce en su camino, por lo que hemos cumplido la condición de discriminación.

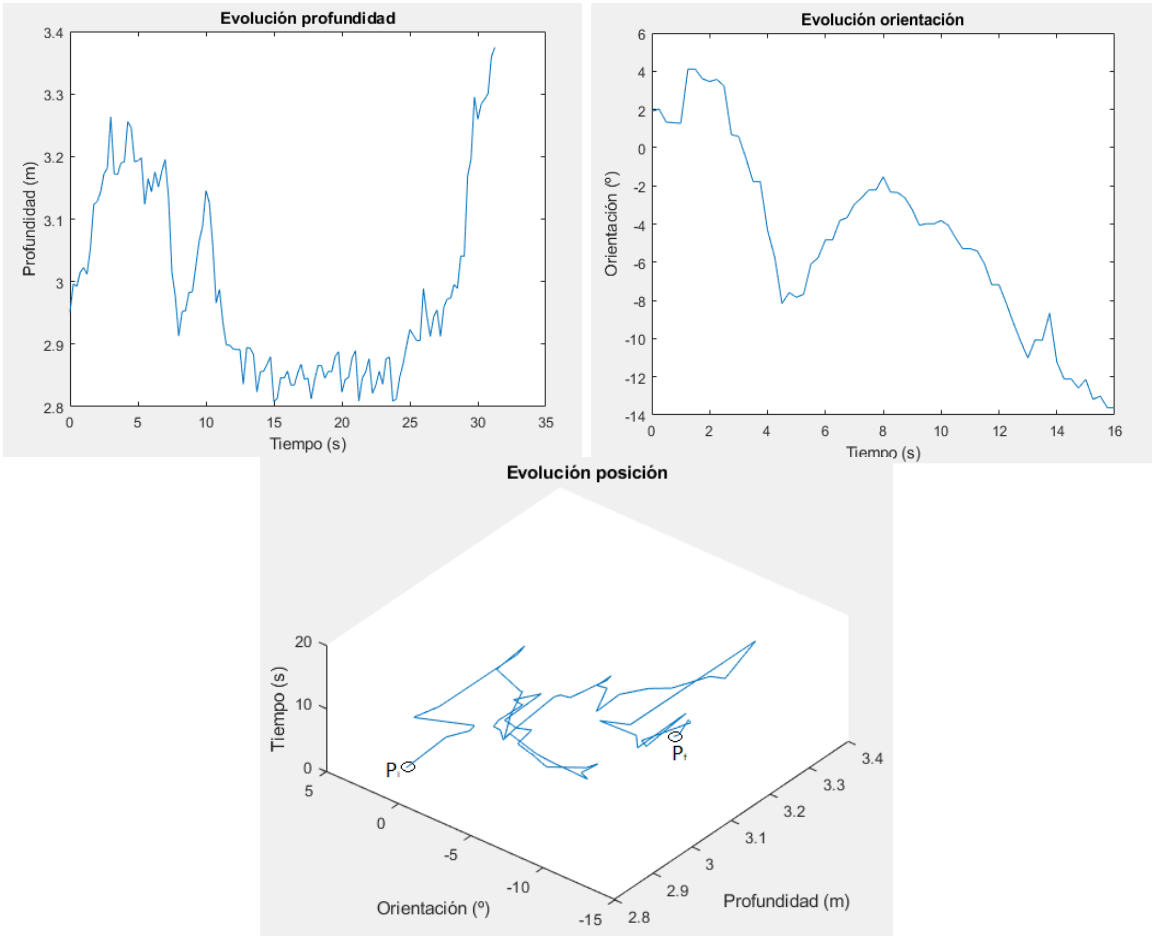


Figura 63: Evolución localización prueba con dos personas.

No obstante, el sistema vuelve a fallar cuando se realiza un giro. Podemos observar que el giro efectuado es de sentido contrario al de la prueba anterior porque el signo de la orientación es diferente. Dicho giro se representa en la Figura 64. No obstante, en este caso el seguidor sí detecta que ha perdido la referencia porque no está enviando ningún dato de localización, pero no es capaz de volver a recuperarla.

Figura 64: Fotograma giro en la segunda prueba.

6. CONCLUSIONES Y TRABAJO FUTURO.

De acuerdo con los resultados obtenidos en los ensayos, se deducen las funcionalidades del sistema de reconocimiento de personas para seguimiento desarrollado, así como trabajos futuros para la mejora de este sistema.

A lo largo de este Trabajo Fin de Máster se ha desarrollado un sistema capaz de reconocer a una persona y seguirla durante una trayectoria. Los objetivos marcados al principio de este documento se han ido cumpliendo, comenzando por un estudio previo de las metodologías y técnicas para reconocer y seguir a una persona en concreto y poder distinguirla del resto de objetos presentes en el campo de visión, para después seleccionar el método más correcto para nuestra aplicación. A continuación, se desarrolló un nodo capaz de obtener la localización tridimensional de la persona. Dicha localización está implementada en un nodo de ROS para que pueda incluirse en un vehículo autónomo. Este trabajo está disponible en *github* a través del siguiente enlace: https://github.com/ricardovmartin/TFM-StereoVisionFollowMe. En él, se encuentra el código del nodo de ROS y la documentación.

De las pruebas de funcionamiento realizadas en los apartados 5.2 y 4.2.4.3 de este trabajo cabe destacar que el sistema es capaz de distinguir a la persona a seguir del resto de personas que se crucen en su trayectoria y la exactitud de los datos cuando la persona está en estático, con una varianza máxima de 0,05 metros para la profundidad y 1.7º para la orientación. No obstante, se ha de mejorar el sistema cuando la persona a seguir realiza giros, ya que el sistema pierde el seguimiento.

A continuación, se presentan propuestas de mejora para este sistema de reconocimiento y seguimiento de personas que se plantean como trabajo futuro.

6.1. Seguimiento en el espacio de la imagen a través del filtro de kalman.

Una de las limitaciones que tiene nuestro sistema es que el seguidor KCF empleado no actualiza la dimensión del recuadro que delimita a la persona. El seguidor es capaz de modificar la ubicación bidimensional de dicho recuadro pero no varía su tamaño. Este problema radica en que si la persona está a mayor distancia de la cámara con respecto a la posición inicial cuando se hizo el reconocimiento, el descriptor HOG del seguidor va a tener mucha más información que no pertenece a la persona y por tanto dará lugar a error, como ocurría en las pruebas realizadas.

La solución que se propone a esta invariación del tamaño del recuadro es emplear un filtro de Kalman en el seguimiento, de tal forma que se vaya modificando su tamaño a lo largo del tiempo en función de lo lejos o cerca que esté la persona de la cámara. La idea general sería reconocer inicialmente a la persona como se ha hecho en este Trabajo y tomar el descriptor HOG del recuadro. A continuación, se aplicaría el filtro de Kalman al recuadro para modificar su tamaño basándose en los datos del descriptor en cada instante de tiempo. Añadiendo otra ventaja, el filtro sería capaz realizar predicciones no solo del tamaño del objeto seguido, sino también de movimiento, en caso de oclusiones.

7. ANEXOS.

7.1. Instalación y configuración de paquetes ROS.

En este apartado se detalla el proceso de instalación y configuración de los paquetes utilizados de ROS, así como de los nodos necesitados en nuestro sistema.

7.1.1. ZED ROS wrapper.

El paquete *ZED ROS wrapper* permite utilizar la cámara ZED2 con ROS. En primer lugar, deberemos preparar un espacio de trabajo *catkin* para ello. Una vez configurado, ejecutamos las siguientes instrucciones en un terminal:

```
$ cd ~/catkin_ws/src
$ git clone –recursive https://github.com/stereolabs/zed-ros-wrapper.git
$ cd ..
$ rosdep install –from-paths src –ignore-src –r –y
$ catkin_make –DCMAKE_BUILD_TYPE=Release
$ source ./devel/setup.bash
```

Para poder acceder a los datos de la cámara ZED2, debemos lanzar su nodo. Para ello, ejecutar el siguiente comando:

```
roslaunch zed_wrapper zed2.launch
```

7.1.2. Filtro de Kalman.

El filtro de Kalman está implementado dentro del paquete *robot_localization*. Para instalarlo, debemos estar dentro de la carpeta *catkin_ws* donde se encuentren el resto de paquetes y nodos de ROS y ejecutar la siguiente instrucción en el terminal:

```
git clone -b kinetic-devel https://github.com/cra-ros-pkg/robot_localization.git
```

A continuación, lanzamos el nodo:

```
roslaunch robot_localization ekf_template.launch
```

Por último, para ver los datos de localización después de haber aplicado el filtro de Kalman, el nodo anterior publica dichos datos en otro nodo denominado */odometry/filtered/*. Para acceder a él, ejecutar el siguiente comando:

```
rostopic echo /odometry/filtered
```

Bibliografía

Alahi, A., Ortiz, R., & Vandergheynst, P. (2012). FREAK: Fast Retina Keypoint. *2012 IEEE Conference on Computer Vision and Pattern Recognition.* Lausanne.

Antonio, J. A., & Romero, M. (2018). Pedestrians' detection methods in video images: A literature review. *International Conference on Computational Science and Computational Intelligence*, (págs. 354-360).

Aurenhammer, F. (1991). *Voronoi Diagrams - A Survey of a Fundamental Geometric.* Schießstattgasse: ACM Computing Surveys.

Babenko, B., Yang, M.-H., & Belongie, S. (2009). Visual Tracking with Online Multiple Instance Learning. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*

Bar-Shalom, Y., Li, R., & Kirubarajan, T. (2001). *Estimation with Applications to Tracking and Navigation.* John Wiley & Sons.

Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2006). Speedep-Up Robust Features (SURF). En *Computer Vision and Image Understanding* (págs. 246-359).

Bravo-Arrabal, J., Toscano-Moreno, M., Fernandez-Lozano, J., Mandow, A., Gomwz-Ruiz, J., & García-Cerezo, A. (2021). The Internet of Cooperative Agents Architecture (X-IoCA) for Robots, Hybrid Sensor Networks, and MEC Centers in Complex Environments: A Search and Rescue Case Study. *Sensors.*

Carlson, J., & Murphy, R. (2005). How UGVs physically fail in the field. *IEEE Transactions on Robotics*, (págs. 423-237).

Chung, S., Butterfield, & Murphy, A. (2021). Current State of Art in the Object Detection for Autonomous Systems. *37th International Manufacturing Conference.* Athlone.

Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow.* O'REILLY.

Hartley, R., & Zisserman, A. (2000). *Multiple View Geometry.* Canberra: Cambridge.

Henriques, J. F., Caseiro, R., Martins, P., & Batista, J. (2014). High-Speed Tracking with Kernelized Correlation Filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (págs. 583-596).

Huang, J., Zou, W., Zhu, Z., & Zhu, J. (2019). An Efficient Optical Flow Based Motion Detection Method for Non-stationary Scenes. *Chinese Control And Decision Conference.* Pekín.

Ian, W., Frank, E., Hall, M., & Pal, C. (2017). Data Mining. Chapter 9.

Inc, S. (2021). *Getting Started with ROS and ZED*. Recuperado el 1 de 11 de 2022, de https://www.stereolabs.com/docs/ros/

Kalal, Z., Mikolajczyk, K., & Matas, J. (2010). Forward-Backward Error: Automatic Detection of Tracking Failures. *International Conference on Pattern Recognition.* Estambul.

Likas, A., Vlassis, N., & Verbeek, J. (2002). *The global k-means clustering algorithm.* Ioannina y Amsterdam.

Lowe, D. (1999). *Object recognition from local scale-invariant features.* Kerkyra.

Mandow, A., Serón, J., Pastor, F., & García-Cerezo, A. (2020). Experimental Validation of a Robotic Stretcher for Casualty Evacuation in a Man-Made Disaster Exercise. *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, (págs. 241-245).

Morales, J., Vázquez-Martín, R., & Mandow, A. (2021). *The UMA-SAR Dataset: Multimodal data collection from a ground vehicle during outdoor disaster response training exercises. https://www.uma.es/robotics-and-mechatronics/cms/menu/robotica-y-mecatronica/datasets/.*

Mrovlje, J., & Vrancic, D. (2008). *Distance measuring based on stereoscopic pictures.* Izola.

OpenCV. (s.f.). *OpenCV documentation*. Recuperado el 1 de Noviembre de 2022, de https://docs.opencv.org/

Peer-to-Peer Service.https://www.investopedia.com/terms/p/peertopeer-p2p-service.asp#:~:text=A%20peer%2Dto%2Dpeer%20(P2P)%20service%20is%20a,intermediation%20by%20a%20third%20party.

Reynolds. *Gaussian Mixture Models.* 2009: Encyclopedia of Biometrics.

Rheinmetall. (s.f.). *The Rheinmetall Mission Master family.* Recuperado el 1 de Noviembre de 2022, de https://www.rheinmetall-defence.com/en/rheinmetall_defence/systems_and_products/unbemannte_fahrzeuge/mission_master/index.php

ROS documentation. (s.f.). Recuperado el 1 de Noviembre de 2022, de http://wiki.ros.org/

ROS documentation. (s.f.). Recuperado el 14 de Octubre de 2022, de robot_localization: http://docs.ros.org/en/melodic/api/robot_localization/html/index.html

Rother, C., Kolmogorov, V., & Blake, A. *"GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts.*

Sánchez, M. (2018). *Detección de personas mediante técnicas de aprendizaje automático: SVM y CNN.* Madrid.

Stereolabs. (s.f.). *Getting started with ROS and ZED.* Recuperado el 1 de Noviembre de 2022, de https://www.stereolabs.com/docs/ros/

Sweezey, M. (2019). *key Chatbot Statistics to Know in 2019.* Recuperado el 1 de Noviembre de 2022, de https://www.salesforce.com/blog/chatbot-statistics/

Toscano, M., Bravo, J., Sánchez, M., Serón, J., Vázquez, R., Fernandez, J., y otros. (2022). *Integrating ROS and Android for Rescuers in a Cloud Robotics Architecture: Application to a Casualty Evacuation Exercise. Submited to IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR).* University of Málaga.

Vaerenberh, S. V., & Santamaria, I. (2018). *Métodos kernel para clasificación*. Recuperado el 1 de Noviembre de 2022, de Universidad de Cantabria: https://gtas.unican.es/files/docencia/APS/apuntes/07_svm_kernel.pdf

VEDECOM, I. (8 de Febrero de 2021). *Autonomous vehicle: first follow-up demonstration in an interoperable military convoy.* Recuperado el 1 de Noviembre de 2022, de https://www.vedecom.fr/vehicule-autonome-premiere-demonstration-de-suivi-en-convoi-militaire-interoperable/?lang=en

Viola, P., & Jones, M. (2001). Rapid object detection using a bossted cascade of sumple features. *Conference on Computer Vision and Pattern Recognition.*

Xiaodong, R., Sanping, D., & Yi, Z. (2017). Parallel RCNN: A deep learning method for people detection using RGB-D images. *International Congress on Image and Signal Processing, BioMedical Engineering and Informatics.* Shanghai, China.

Yun, K., Lim, J., & Young, J. (2016). *Scene conditional background update for moving object detection in a moving camera.* Seúl: Department of Electrical and Computer Engineering, ASRI, Seoul National University.

Zdimalova, M. (2015). *Graph cuts in image processing.* Bratislava.

Zhang, C., Platt, J., & Viola, P. (2005). Multiple Instance Boosting for Object Detection. En *Advances in Neural Information Processing Systems.* University of British Columbia NIPS 2005.

Zhang, J., Xiao, J., Zhou, C., & Peng, C. (2018). A multi-class pedestrian detection network for distorted pedestrians. *IEEE Conference on Industrial Electronics and Applications (ICIEA)*, (págs. 1079-1083).

Zhenjiang, L., Kunfeng, K., Li, L., & Fei-Yue, W. (2006). A Review on Vision-Based Pedestrian Detection for Intelligent Vehicles. *2006 IEEE International Conference on Vehicular Electronics and Safety.* Shanghai.