UNIVERSIDAD
DE MÁLAGA

TESIS DOCTORAL

Sistema avanzado de detección de obstáculos y navegación autónoma para vigilancia y protección basado en flota de vehículos aéreos no tripulados

Departamento de Tecnología Electrónica

JESÚS GARCÍA MERINO

2016

AUTOR: Jesús García Merino

https://orcid.org/0000-0002-2934-8161

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga

DEDICATORIA

A mis padres

AGRADECIMIENTOS

A David, mi director de tesis doctoral

Dr. D. Francisco David Trujillo Aguilera, director de la presente investigación para aspirar al grado de Doctor por **D. Jesús García Merino**, hace constar:

Que la tesis titulada: "**SISTEMA AVANZADO DE DETECCIÓN DE OBSTÁCULOS Y NAVEGACIÓN AUTÓNOMA PARA VIGILANCIA Y PROTECCIÓN BASADO EN FLOTA DE VEHÍCULOS AÉREOS NO TRIPULADOS**", realizada por D. Jesús García Merino, reúne las condiciones científicas, académicas y requisitos formales legalmente establecidos para su presentación.

Y para que así conste y surta los efectos oportunos, expide y firma el presente documento en Málaga, a 10 de noviembre de 2015.

Fdo.: Dr. D. Francisco David Trujillo Aguilera

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

LISTADO DE ACRÓNIMOS Y ABREVIATURAS

3D: 3 Dimensiones
AHRS: Attitude Heading Reporting System firmware
AIE: Agencia Internacional de la Energía
API: Application Programmer Interface
ARM: Advanced RISC Machine
AUV: Autonomous Underwater Vehicle
BLOB: Binary Large Object
BM: Block Matching
BOW: Bag-of-Words
BRIEF: Binary Robust Independent Elementary Features
BSD: Berkeley Software Distribution
CDTI: Centro de Desarrollo Tecnológico Industrial
CE: Comisión Europea
CEPE: Comisión Económica para Europa
CMOS: Complementary Metal-Oxide Semiconductor
EKF: Extended Kalman Filter
EST: Expansion Search Tree
FAO: Organización de Naciones Unidas para la Alimentación y Agricultura
FAST: Features from Accelerated Segment Test
FEDER: Fondos Europeos de Desarrollo Regional
FLANN: Fast Library for Approximate Nearest Neighbors
FOV: Field of View
GPIO: General Purpose Input/Output
GPL: General Public License
GPS: Global Positioning System
HW: Hardware
I+D+I: Investigación, Desarrollo e Innovación
IDS: Imaging Development Systems
IMU: Intertial Messurement Unit
IR: Infrarrojo.
KLT: Kanade-Lucas-Tomasi
KPIECE: Kinodinamic Panning by Interior-Exterior Cell Exploration
LIDAR: Light Detection and Ranging o Laser Imaging Detection and Ranging
LSD: Large-Scale Direct Monocular
MAVLink: Micro Air Vehicle Communication Protocol
ONU: Organización de Naciones Unidas

OpenCV: Open Computer Vision Library
ORB: Oriented-FAST and Rotated BRIEF
PCL: Point Cloud Library
PIC: Protección de Infraestructuras Críticas
PRM: Probabilistic Roadmap
PWM: Pulse Width Modulation
SFM: Structure from Motion
SBL: Single-query Bidirectional Lazy collision checking
SW: Software
RANSAC: Random Sample Consensus
RGB: Red Green Blue
RGBD: Red Green Blue Depth
ROS: Robot Operating System
RRT: Rapidly-exploring Random Trees
RRT*: Optimal Rapidly-exploring Random Trees
RRG: Rapidly-exploring Random Graph
RTAB-Map: Real-Time Appearance-Based Mapping
SGBM: Semi Global Block Matching
SIFT: Scale-Invariant Feature Transform
SLAM: Simultaneous Localization and Mapping
SO: Sistema Operativo
SURF: Speeded Up Robust Features
TIC: Tecnologías de la Información y Comunicaciones
UE: Unión Europea
URL: Uniform Resource Locator
USB: Universal Serial Bus
V4L: Video for Linux
WVGA: Wide Video Graphics Array

CAPÍTULO 1: INTRODUCCIÓN

1.1 Antecedentes

El sector de los drones (o vehículos aéreos no tripulados) constituye hoy día un componente importante y dinámico de la aviación mundial. La Unión Europea calcula que en poco más de una década, el mercado superará los 15.000 millones de euros al año, llegando a superar el 10 % de la facturación total del sector aeronáutico. En los últimos años, las aplicaciones civiles de los drones se han multiplicado, abarcando cada vez más ámbitos de la sociedad. Estos datos no hacen más que confirmar que estamos solamente ante el principio de una nueva era en la que esta tecnología ha dejado de estar reservada exclusivamente al ámbito militar para llegar al ámbito civil de forma acelerada.

El gran potencial de los drones de uso civil es innegable. Sin embargo, al mismo tiempo son una tecnología emergente muy controvertida. Se trata de una tecnología barata, versátil y de fácil acceso a la población sin ningún tipo de control. Los riesgos y amenazas asociadas a su uso civil crecen a medida que lo hace el propio mercado. Es necesario comprender los peligros que conllevan estas aeronaves para establecer medidas que garanticen el uso de esta tecnología de forma segura y responsable. Si bien la aviación civil convencional (las grandes aeronaves que transportan pasajeros y mercancías) ha evolucionado considerablemente en términos de fiabilidad de sus componentes, las tecnologías asociadas a los drones no alcanzan todavía unos niveles adecuados que permitan abrir el espacio aéreo de forma segura. En otras palabras, la probabilidad de que un dron provoque un accidente es todavía alta debido a factores humanos (intencionados o no) y tecnológicos. Como consecuencia, la mayoría de las normativas actuales sobre el uso de drones a nivel mundial no contemplan el vuelo libre de estas aeronaves sobre entornos urbanos o industriales para cualquier usuario, quedando relegado a situaciones de riesgo, catástrofe o calamidad pública, y a su uso exclusivo en estos casos extremos para los cuerpos y fuerzas de seguridad así como los cuerpos de gestión de emergencias.

Los drones constituyen pequeñas aeronaves que pueden ser gobernadas remotamente mediante el control manual de un humano con la ayuda de sistemas de control por radiofrecuencia. Pero también pueden realizar vuelos automatizados a través de un conjunto de órdenes de vuelo pre-planificadas y el apoyo de sistemas de posicionamiento. En cualquiera de ambos casos, los drones de uso civil y pequeño tamaño tienen asignados el espacio aéreo más cercano al suelo, cuya altura máxima depende de la normativa del país aplicable, y cuyo alcance no supera nunca los cientos de metros (120 metros en el caso de España y con la normativa vigente).

Además, los espacios aéreos por los que circulan los drones son cada vez más complejos, sobre todo si nos referimos a ciudades o entornos industriales. En estos entornos, la presencia de estructuras representa distintos obstáculos para el dron, lo cual dificulta la navegación aérea segura. Además, la cercanía de estos vuelos a la cota de tierra hace que el vuelo del dron tenga que ser obligatoriamente compartido con la presencia de estructuras dentro de espacios diversos: urbanos, rurales, industriales o el propio medioambiente.

Por todo ello es de suponer que, si los drones van a adquirir un papel relevante en la prestación de servicios a la sociedad allí donde cohabita el ser humano, entonces los sistemas de reconocimiento del entorno embarcados supondrán una de las áreas tecnológicas de más relevancia para garantizar la seguridad dentro de la industria de los drones en el ámbito civil, en especial en las tareas de protección y mantenimiento de las infraestructuras críticas que soportan los servicios esenciales para la vida del ser humano en el planeta.

1.2 Metodología

El ámbito de conocimiento de esta tesis doctoral es el de la ingeniería aplicada. La metodología utilizada para el primer objetivo parcial será la recopilación y estudio de la documentación existente en bases de datos científicas sobre las diferentes técnicas de detección de obstáculos y cálculo de distancias. Tras este estudio teórico y de recopilación de documentación se realizarán implementaciones reales de las metodologías analizadas. Estas implementaciones se realizarán

combinando dispositivos electrónicos, sensores y software para tal efecto. Posteriormente se realizarán distintos experimentos y pruebas para validar la caracterización de las metodologías analizadas teóricamente.

Por último, y para dar cobertura al segundo objetivo parcial de la tesis, se realizará una implementación de un sistema de sincronización hardware de cámaras para visión artificial, mejorando los resultados en términos de eficiencia y tiempo de las metodologías de sincronización software.

1.3 Marco de la investigación

El ámbito de trabajo de las investigaciones realizadas en esta tesis se han desarrollado bajo el marco de un proyecto de I+D cofinanciado con Fondos Europeos de Desarrollo Regional (en adelante FEDER) en la empresa Spin Off de la Universidad de Málaga AEORUM y dentro de las actividades del proyecto "*SYSROBOTICS: Desarrollo de un Sistema Avanzado de Información para el Mantenimiento y Protección de Infraestructuras Críticas*". Proyecto que fue galardonado en 2012 con el primer premio Spin Off de la Universidad de Málaga en la modalidad de estudiantes y presentado en colaboración por el titular de esta tesis doctoral. El proyecto SYSROBOTICS fue además co-financiado por el Ministerio de Economía a través del Centro de Desarrollo Tecnológico Industrial (en adelante CDTI) en la modalidad de proyectos en consorcio mediante la convocatoria FEDER-INNTERCONECTA 2013 para la Comunidad Autónoma de Andalucía con un presupuesto total de 2.484.015,00€.

El desarrollo de esta tesis en el marco de un proyecto co-financiado por fondos europeos FEDER ha permitido la propia realización de las investigaciones y desarrollos. El elevado coste de los equipamientos, materiales y recursos humanos utilizados durante la elaboración de la tesis habrían constituido una barrera insalvable en términos de financiación para el titular de la tesis.

Los resultados aquí expuestos se corresponden exclusivamente al ámbito de trabajo de las tareas asignadas a la empresa AEORUM y dirigidas por el aspirante a doctor en el marco del proyecto

SYSROBOTICS. A lo largo de todo el documento de tesis nunca se revelan datos considerados como sensibles o confidenciales de la empresa AEORUM. Tampoco se incluyen en esta tesis ninguno de los resultados considerados por dicha empresa como no publicables.

Por último, para garantizar el cumplimiento de la ley de protección de datos personales, se han establecido medidas de protección de la intimidad (en las ilustraciones de esta tesis) de los integrantes del equipo de investigación que colaboraban con el proyecto SYSROBOTICS con objeto de salvaguardar su intimidad. Estas medidas garantizan el cumplimiento de las normativas de protección de datos personales existentes.

1.4 Necesidad del estudio: justificación socioeconómica

Nuestro mundo vive un nivel de desarrollo económico y de bienestar con el que apenas podíamos soñar hace unas décadas. El crecimiento y consolidación de las estructuras productivas, así como el mejor aprovechamiento de los recursos naturales han hecho posible este progreso. Las agresiones a las infraestructuras, tales como el cambio climático, la propia actividad humana, los actos de sabojate u otros fenómenos naturales (como los terremotos) suponen una grave amenaza para la estabilidad de los países e incluso para la propia vida en el planeta. Si queremos conservar nuestros niveles de bienestar, estamos obligados a preservar las infraestructuras y medios naturales que constituyen nuestra mayor riqueza, poniendo al servicio de éstos todo el conocimiento y los esfuerzos necesarios para garantizar su protección, y con ello garantizar los servicios esenciales.

La evolución de los sistemas de mantenimiento y protección de las infraestructuras que representan nuestro bienestar supone una necesidad para la sociedad en su conjunto, pues un buen control de nuestras infraestructuras garantiza la correcta prestación de los servicios esenciales para la vida en el planeta con unos mínimos de calidad. En este proceso continuo de evolución de los sistemas de mantenimiento y protección, los drones representan una oportunidad sin precedentes, pues permiten el acceso a localizaciones remotas, así como el análisis e inspección sin riesgos. Además, los drones ofrecen la posibilidad de automatizar las tareas encomendadas,

mejorando la calidad de los resultados y reduciendo los costes de supervisión industrial. Es por ello, que realizar todas estas actividades de forma segura, representa una prioridad para el normal desarrollo de la vida de los seres humanos y su mejora futura.

Los servicios esenciales, como el agua, la energía o los transportes, son aquellos cuya naturaleza no permite soluciones alternativas, y por tanto, cualquier perturbación de los mismos tendría un grave impacto sobre la actividad humana pues no hay sustitutivos para estos. Dada la importancia de los servicios esenciales, los nuevos retos para el siglo XXI pasan por garantizar las infraestructuras críticas , pues se entiende por infraestructuras críticas aquellas que dan cobertura y permiten el abastecimiento de los servicios esenciales.

No existe actividad humana que no se encuentre vinculada o dependa de alguno de los doce servicios esenciales contemplados por la normativa española [1], que son: Administración, agua, alimentación, energía, espacio, industria química, industria nuclear, instalaciones de investigación, salud, sistema financiero y tributario, tecnologías de la información y comunicaciones (en adelante TIC) y transporte. Según innumerables proyecciones realizadas por la Comisión Económica para Europa (en adelante CEPE), la Organización de Naciones Unidas (en adelante ONU) y la Organización de Naciones Unidas para la Alimentación y Agricultura (FAO), el cambio climático supone una de las mayores amenazas, no sólo para la vida en el planeta, sino también para la economía de los países desarrollados, pues tendrá como consecuencia una repercusión muy negativa en los servicios esenciales, especialmente en la generación de energía, en los recursos hídricos disponibles, así como en la degradación material y en la exposición a catástrofes naturales de las infraestructuras de transporte [2]. Y sin embargo, durante el siglo XXI la demanda de consumo de agua y energía, así como el uso de los transportes en el planeta continuará creciendo. Por todo ello, los Estados modernos se enfrentan actualmente a diferentes desafíos que confieren a la seguridad nacional un carácter cada vez más complejo, en especial, para los servicios esenciales con una previsión mayor de ser castigados por el cambio climático: la energía, el agua y los transportes.

En el ámbito de la energía, según estimaciones de la Agencia Internacional de la Energía (en adelante AIE) [3] la demanda mundial de energía crecerá más de un tercio hasta 2035. Se prevé que las necesidades de agua para la producción de energía crezcan dos veces más rápido que la demanda de energía y ello provocará un grave problema en la generación de energía. La crisis provocada por el tsunami en la central nuclear de Fukushima en 2010 puso en evidencia las carencias de los administradores de infraestructuras energéticas en la evaluación de anomalías en la estructura durante la gestión de crisis. En este caso, un rápido diagnóstico de los daños hubiera supuesto agilizar la toma de decisiones, además de haber evitado los riesgos personales por exposición a radiaciones o a explosiones de los evaluadores. La inundación de la sala de máquinas de la central hidroeléctrica de Sayano-Shushénskaya (Siberia) en 2010 provocó 10 muertos y 72 desaparecidos; además, la mayor fábrica de aluminio del mundo se paralizó a consecuencia del corte de suministro eléctrico. Según los medios de comunicación, la administración rusa informó de que no se observaron "*daños visibles en la presa*" [4], lo cual podría dar indicios de que los sistemas actuales de inspección material de las infraestructuras son insuficientes.

En el plano de los recursos hídricos, según estimaciones de la ONU, siete mil millones de seres humanos padecerán escasez de agua en 2050 debido a los efectos del cambio climático [5]. Hay que tener en cuenta que en 2050 la población mundial será de nueve mil millones de personas [6]. Actualmente, únicamente el 2,53% del total de agua existente en el planeta es dulce y el resto es salada. Además, las dos terceras partes del agua dulce se encuentran inmovilizadas en glaciares y nieves perpetuas. El agua dulce es esencial para la producción de energía: para la generación eléctrica; para la extracción, el transporte y el procesamiento de petróleo, gas y carbón; y, cada vez más, para el riego de los cereales empleados para producir biocombustibles. Además, la dependencia entre el agua y la generación de energía será cada vez mayor y el agua ganará importancia como criterio de evaluación de la viabilidad de los proyectos de energía [5].

Por último, en el plano de las infraestructuras de transporte y en relación a las consecuencias que el cambio climático tendrá sobre

estas, los efectos negativos se agravarán especialmente en la Europa meridional, donde se prevé un mayor aumento de las sequías, de las temperaturas y de las catástrofes medioambientales provocadas por las lluvias torrenciales [2].

Todo ello afectará de forma más acusada a la seguridad material de grandes infraestructuras y entornos urbanos, acelerando su degradación y aumentando el riesgo de graves accidentes por colapso. Por ello, la Comisión Europea (en adelante CE) alerta y exige ***[1]*** a los Estados miembros para que tomen medidas protectoras a través de nuevos mecanismos de conservación y previsión de catástrofes adaptados a las nuevas amenazas como el cambio climático o el terrorismo.

En un mundo globalizado económicamente e inestable, las amenazas a nuestra seguridad no entienden de fronteras, y nadie puede negar que las infraestructuras críticas de las que dependen nuestro bienestar, se encuentran expuestas a graves amenazas, que en caso de materializarse pueden provocar un colapso de nuestros sistemas. Estas amenazas para nuestras infraestructuras se pueden concentrar en dos focos principalmente: el cambio climático [7] [8] [9] [10] y el terrorismo [11] [12] [13] [14]. Puesto que el riesgo al que se encuentran expuestas las Infraestructuras críticas, es función de la probabilidad de materialización de las amenazas existentes y del valor del impacto, la pregunta es ¿cuál es la probabilidad de materialización de la amenaza? No es fácil responder a esta pregunta, pero los estudios y previsiones expuestos hacen estimar una probabilidad preocupante, y como el valor del impacto en caso de incidente es muy alto, el riesgo de que se presupone también es elevado.

Los Estados deben asumir la responsabilidad de garantizar la seguridad de las infraestructuras críticas, legislando, implantando políticas y procedimientos, y coordinando los esfuerzos de todas las organizaciones involucradas en el despliegue y gestión de las

[1] *En junio 2012, por 573 votos a favor, 90 en contra y 16 abstenciones, el Parlamento Europeo aprobó un informe presentado por el eurodiputado Ivailo Kalfin, en el que se pide a la CE que tome la iniciativa para exigir a los gobiernos nacionales que adopten las medidas necesarias, a fin de proteger las infraestructuras críticas.*

infraestructuras críticas, tal y como se establece en la Ley de Protección de Infraestructuras Críticas (en adelante PIC) [11].

En base a lo anterior, la necesidad detectada en los mercados de las tecnologías tratadas en esta tesis a corto, medio y largo plazo es amplia y heterogénea. El abanico de potenciales sectores comprende tres segmentos principales donde se cabría esperar un impacto de mercado más amplio: el segmento de las infraestructuras energéticas, las infraestructuras hídricas y las infraestructuras de transporte.

Independientemente de las prescripciones legales y de seguridad, existen infraestructuras críticas productivas donde la concienciación de los propietarios sobre la seguridad es tanta que no se requeriría siquiera de prescripciones legales para su implantación, por ejemplo: las plantas de tratamiento. O incluso en el ámbito civil de zonas urbanas con gran densidad de población: es fácil imaginar las devastadoras consecuencias de una presa colapsada frente a un entorno urbano o de una catástrofe en una central nuclear (como los casos de Chernobyl o Fukushima).

La conservación de grandes infraestructuras se va a convertir en uno de los segmentos de mercado de mayor crecimiento en el sector de la construcción. Cada vez se hace más necesario un adecuado mantenimiento de las infraestructuras públicas, no solo para garantizar una correcta prestación del servicio público, sino para mantener las fuertes inversiones asociadas a su construcción y mantenimiento. En este marco, y ante las dificultades de obtener el nivel de servicio requerido y el mantenimiento del patrimonio, la Administración está adoptando nuevas fórmulas de colaboración público-privada, dando paso a un mayor protagonismo de la iniciativa privada en la conservación y el mantenimiento de las grandes infraestructuras de transporte, energéticas e hidráulicas. Todos los actores vinculados al desarrollo de infraestructuras y servicios se encuentran hoy en día ante un nuevo reto, marcado por varias pautas, entre las que cabe destacar: la escasez de recursos, el desarrollo de nuevas técnicas de gestión cada vez más eficientes, incorporando tecnologías, y la necesidad de poner en práctica nuevos instrumentos financieros para garantizar su construcción, operación y mantenimiento

La Unión Europea [UE] y los Estados miembros han invertido esfuerzos en materia de prevención frente al cambio climático de forma importante; sin embargo, establece en su Libro Verde sobre adaptación al Cambio Climático que "*esos esfuerzos van a tener que intensificarse como consecuencia del cambio climático*" [8].

En general, toda medida adoptada en la protección de infraestructuras tiene por objeto reducir el peligro humano y de los daños ocasionados en caso de catástrofe. Independientemente de las prescripciones legales vigentes de diversa índole, cada vez más las grandes infraestructuras tienden a dicho fin, pues acogen en sus senos activos de gran valor económico, medioambiental y humano. Tal es el caso de las infraestructuras críticas como:

- **Infraestructuras energéticas**: Centrales hidroeléctricas, nucleares y térmicas, refinerías, líneas de alta tensión, buques petroleros y gaseros, aerogeneradores, gasoductos y oleoductos.
- **Infraestructuras hídricas**: presas, embalses, canales.
- **Infraestructuras de transporte**: líneas férreas, viaductos y túneles, autopistas, grandes buques.

Desde un punto de vista funcional, los resultados de esta tesis serían de utilidad para su uso en una red de sensores móviles (sobre drones) para ser utilizados como parte de un sistema para la detección de anomalías y actuación automática. Esto último reducirá considerablemente los costes de mantenimiento para los administradores de infraestructuras pues no habrá que instalar grandes sistemas cableados y de sensores. Dichas prestaciones lo convierten en una poderosa herramienta sin antecedentes en sector de la seguridad para grandes infraestructuras. Esta funcionalidad persigue:

- **1.** Reducir el número de personas afectadas en caso de catástrofe, valorado en función del número potencial de víctimas mortales o heridos con lesiones graves y las consecuencias para la salud pública.
- **2.** Reducir el impacto económico en función de la magnitud de las pérdidas económicas y el deterioro de productos y servicios en caso de catástrofe.

- **3.** Reducir el impacto medioambiental, degradación en el lugar y sus alrededores.
- **4.** Reducir el impacto público y social, por la incidencia en la confianza de la población en la capacidad de las Administraciones Públicas, el sufrimiento físico y la alteración de la vida cotidiana, incluida la pérdida y el grave deterioro de servicios esenciales.
- **5.** Reducir los costes de mantenimiento y protección de las infraestructuras.
- **6.** Mejorar las prestaciones con respecto a los sistemas de supervisión fijos y móviles actuales.

1.5 Estructura de la tesis

El núcleo principal de este documento de tesis se compone de cinco capítulos.

Los capítulos uno y dos son introductorios. Estos capítulos definen las bases de la investigación y desarrollo objeto de esta tesis, que se recoge en los capítulos tres y cuatro.

El capítulo tres está dedicado al análisis teórico y práctico del estado del arte, y se divide en dos bloques:

- BLOQUE A: dedicado a la detección de obstáculos y el cálculo de distancias.
- BLOQUE B: dedicado al mapeo robótico y al posicionamiento de los vehículos autónomos no tripulados.

El capítulo cuatro se centra en el desarrollo de un nuevo método de sincronización hardware que mejora las prestaciones de procesamiento de un sistema de visión artificial que utiliza estereoscopía y que utiliza métodos de sincronización software.

Por último, el capítulo cinco cierra este documento de tesis con las conclusiones de este estudio así como las líneas de desarrollo futuro.

Además de los cinco capítulos que constituyen el núcleo de la tesis, se aporta información complementaria a través de distintos anexos e índices. Estos son:

Previos a los cinco capítulos que conforman la tesis:

- Portada.
- Dedicatoria y agradecimientos.
- Índice general.
- Índice de ilustraciones.
- Índice de tablas.
- Listado de acrónimos y abreviaturas.

Posteriores a los cinco capítulos que conforman la tesis:

- Índice de referencias bibliográficas.

El criterio de identificación y numeración para las ilustraciones y las tablas referenciadas siguen un número de secuencia de números enteros comenzando desde 1 (es decir: Ilustración 1, Ilustración 2, Ilustración 3…., Ilustración N) a lo largo de todo el documento de tesis. Sin embargo, el listado de ecuaciones no está recogido en ningún índice, por ello, y con objeto de facilitar la localización referenciada, se ha seguido el siguiente criterio de numeración e identificación:

Ecuación *(Número de Capítulo).(Caracter del Bloque).(identificador secuencial de la ecuación)*

Es decir, una ecuación localizada en el capítulo 3, bloque B de esta tesis se identificaría de la siguiente forma:

Ecuación 3.B.x

Donde x es un número de secuencia que comienza con el valor "1" al inicio de cada capítulo y se incrementa con cada aparición de una nueva ecuación dentro del capítulo.

CAPÍTULO 2: OBJETIVO Y ALCANCE DE LA INVESTIGACIÓN

El objetivo general de esta tesis es la investigación es el desarrollo de un procedimiento que mejore las condiciones de navegabilidad de un dron en entorno complejo donde el dron deberá detectar, identificar y sortear distintos obstáculos desconocidos. Este objetivo general se compone de dos objetivos parciales:

- **OBJETIVO PARCIAL 1:** el estudio teórico, el análisis y la validación práctica de distintas técnicas y tecnologías que se utilizan actualmente para la detección de obstáculos en sistemas basados en drones y otros vehículos autónomos no tripulados en general (terrestres, aéreos y marítimos). Esta actividad ocupa la mayor parte de esta tesis (Capítulo 3). La finalidad de este objetivo es poder establecer un buen punto de partida sustentado en un análisis teórico y práctico del estado del arte. Este análisis previo constituye el mejor marco para poder desarrollar implementaciones futuras que aporten una mejora sustancial respecto al propio estado del arte.

- **OBJETIVO PARCIAL 2:** el desarrollo de un sistema de visión artificial con visión estereoscópica, que mejore las prestaciones de los sistemas actuales a través del proceso de adquisición de datos y procesamiento de los mismos. La finalidad de este objetivo es mejorar las prestaciones de navegabilidad de los sistemas aéreos no tripulados con respecto al estado del arte actual.

La consecución de estos dos objetivos parciales asegura el cumplimiento del objetivo general de esta tesis.

El alcance de esta tesis se ciñe al cumplimiento de los objetivos establecidos. Dicho alcance incluye la validación en laboratorio y pruebas en exterior de las distintas metodologías, configuraciones y técnicas conocidas de detección de obstáculos y cálculo de distancias. Dentro del alcance se contempla también la implementación de cada uno de los sistemas o técnicas de detección de obstáculos integrando

cada una de sus partes, configurándolas y realizando distintos experimentos para su entendimiento, así como su validación modificando los distintos parámetros que compongan cada sistema.

CAPITULO 3: ESTADO DEL ARTE Y SU VALIDACION

BLOQUE A: DETECCIÓN DE OBSTÁCULOS Y CÁLCULO DE DISTANCIAS

3.1 Técnicas de visión artificial

La visión artificial pretende simular la visión humana mediante el uso de sensores pasivos (cámaras) y una unidad de procesado suficientemente potente. El problema a la hora de usar cámaras, es la pérdida de una de las dimensiones espaciales, la profundidad. Esto ocurre al pasar de un escenario real a un soporte bidimensional (el sensor de la cámara).

Una gran gama de utilidades para los vehículos que se están desarrollando pasan por la navegación autónoma de los mismos. Para este fin se han desarrollado varias técnicas con las que se ha intentado detectar y evadir obstáculos. Sin embargo, por los resultados que se han obtenido, la visión artificial ha tomado un papel importante.

Existen distintas configuraciones de sistemas de visión artificial en función del número de cámaras usadas. En la presente tesis se desarrollan las pruebas realizadas con una y dos cámaras. Es decir, configuración monocular y estéreo.

3.1.1 Visión monocular

La primera de las configuraciones es el uso de una única cámara para obtener la profundidad. Sin embargo, para llegar a reconstruir la profundidad, se pasa por dos problemas: el modelado de la cámara a través de matrices y la calibración de la cámara. La calibración será un paso crítico para obtener buenos resultados.

3.1.2 Conceptos generales

El modelo básico para analizar es la cámara oscura (en inglés pinole) [15]. Consiste básicamente en una apertura muy pequeña sin lente.

Esto va a permitir una primera aproximación a la matemática que envuelve las cámaras y la necesidad de calibrar.

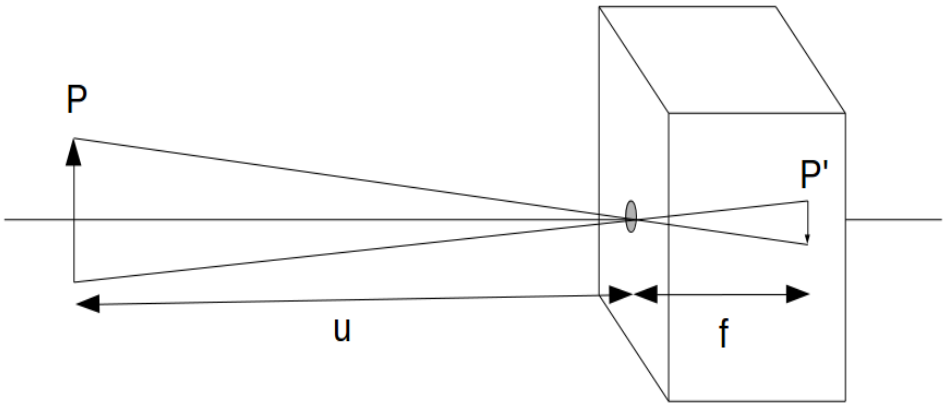

Figura 1: *Modelo de cámara oscura*

Suponiendo válida la aproximación de la teoría de rayos, se aplica sobre la imagen anterior. La distancia entre la apertura y el plano de la imagen se define como distancia focal (f). Este parámetro será importante para obtener la profundidad de manera fiable.

Aplicando simple triangulación en la Figura 1, se puede obtener la proyección del objeto real en el plano de la imagen:

$$-\frac{P'}{f} = \frac{P}{u} \rightarrow -P' = \frac{P}{v} f \qquad \textbf{Ecuación 3.A.1}$$

Sin embargo, interesa estudiar el plano de la imagen delante del plano de la apertura. Así que se va a dar la vuelta a los dos planos de la Figura 1. Así, la apertura se redefine como el centro de la proyección, que no será necesariamente el centro del sensor como ocurría en el modelo más simple:

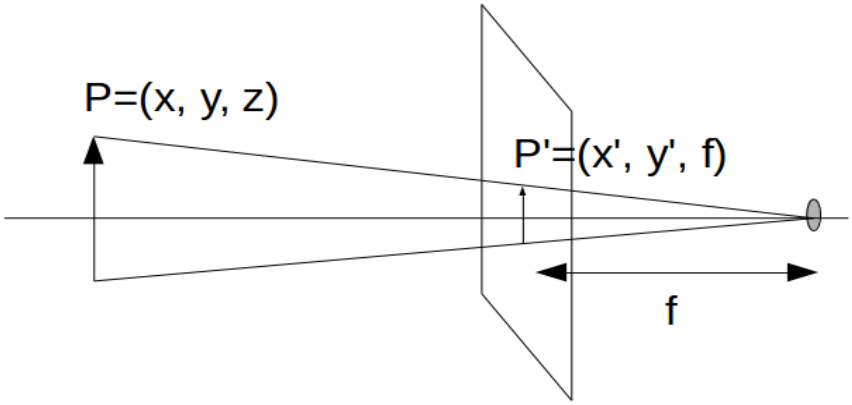

Figura 2: *Cambio de planos en cámara oscura*

Ahora, el rayo que sale del punto P, deja su proyección en el plano imagen como un punto P'. Así para todos los puntos que forman el objeto. Así se obtiene en el plano de la imagen la proyección del objeto completo.

Debido a que el plano de la imagen (interprétese plano de imagen como sensor de la cámara) no será cuadrado, se necesita introducir dos nuevos parámetros: c_x y c_y. Modelan el posible desplazamiento del centro del plano de la imagen y el centro de proyección. Estos parámetros también serán importantes a la hora de calibrar la cámara. En esta situación se tiene:

$$x' = f_x \frac{x}{z} + c_x \qquad y' = f_y \frac{y}{z} + c_y \qquad \textbf{Ecuación 3.A.2}$$

Se ha tenido que introducir dos distancias focales diferentes debido a que la dimensión en píxeles por milímetro de los sensores que se van a utilizar no son cuadrados. Serán, por lo general, rectangulares. Es una relación entre la distancia focal real de la lente y los píxeles por milímetro que tenemos en el sensor.

Para pasar de los puntos del mundo real a los puntos de la imagen hay que hacer uso de la geometría proyectiva. Se usarán transformaciones que requieren trabajar con coordenadas homogéneas [16]. Es decir, el punto del espacio (x, y, z) se convierte en (x, y, z, w).

Como el plano de proyección es de dos dimensiones, se van a tener puntos de tres coordenadas $P = (p_1, p_2, p_3)$. Los puntos con coordenadas proporcionales serán iguales, así que se pueden

recuperar dividiendo por p_3. Así se pueden relacionar los parámetros de la cámara con los puntos de la escena real y los conseguidos en el plano imagen. La relación matricial es:

$$q = MQ \rightarrow \begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \textbf{\textit{Ecuación 3.A.3}}$$

La matriz M que se obtiene es la matriz de parámetros intrínsecos [17]. Con ella se obtiene la proyección de los puntos del mundo real en el plano de la imagen. Mediante el modelado de la cámara oscura, se ha obtenido una relación matricial entre los puntos del espacio y su posición en la cámara.

Como el modelo no contempla uso de lentes y la apertura es muy pequeña, se necesitaría mucho tiempo de exposición para que el sensor escribiera la información. Por ello se hace necesario el uso de lentes, para formar imágenes de forma más rápida, pues tienen la capacidad de concentrar más luz incidente.

Sin embargo, las lentes no son ideales. Introducen distorsión [18] en mayor o menor medida, puesto que el proceso de fabricación no puede ser perfecto. Los tipos de distorsión más importantes son:

- **Distorsión radial**. Aleja el pixel de la posición que debería tomar en la imagen. Los dos tipos de distorsión radial más importantes son la de barril y la de cojín. La de barril se puede encontrar también como ojo de pez [19]. El origen de la distorsión radial es la forma de la lente, pues es mucho más sencillo fabricar una lente esférica que una lente cóncava como las ideales. En la figura 3 se puede ver el efecto de ambas:

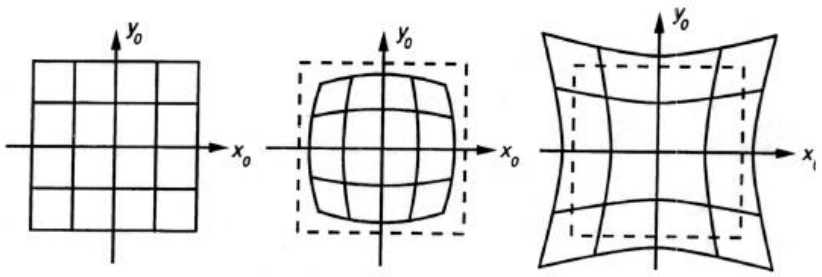

Figura 3: *Tipos de distorsiones radiales. De izquierda a derecha, imagen sin distorsión, distorsión de barril y distorsión de cojín.*

- **Distorsión tangencial**. Procede de la mala alineación de componentes en el proceso de ensamblaje de la cámara. Los elementos de la cámara de desalinean y pierden el paralelismo. En la figura 4 se muestra un posible error de ensamblaje de la cámara:

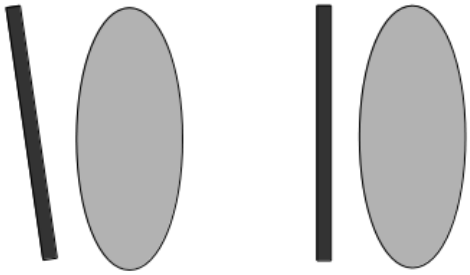

Figura 4: *Distorsión tangencial. De izquierda a derecha, derivas entre sensor y lente en el ensamblaje y libre de derivas.*

Las distorsiones descritas tienen que ser corregidas para un futuro tratamiento. Como se puede observar, la distorsión radial es menor en el centro óptico de la imagen, y va aumentando hacia la periferia. Por lo general, se puede corregir por un desarrollo en Serie de Taylor centrado en cero. La corrección de la posición del pixel en el sensor sigue la siguiente ecuación:

$$x_{\sin distorsion} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$y_{\sin distorsion} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Ecuación 3.A.4

Para corregir la distorsión tangencial hay que sumar dos nuevos parámetros. La forma de corregir este tipo de distorsión es:

$$x_{\sin distorsion} = x + (2p_1 y + p_2(r^2 + 2x^2))$$

$$y_{\sin distorsion} = y + (p_1(r^2 + 2y^2) + 2p_2 x)$$

Ecuación 3.A.5

Donde (x, y) es la coordenada del pixel distorsionado en el sensor. Se tienen entonces cinco parámetros para corregir la distorsión. Para cámaras sin mucha distorsión radial, se pueden tomar los dos primeros términos del desarrollo en serie de Taylor de la corrección radial. Sin embargo, para cámaras con gran distorsión radial (como las montadas con lentes de ojo de pez) se deben tomar todos los términos.

En sistemas reales se pueden encontrar más tipos de distorsiones. Sin embargo, serán mucho menos notables que los ya descritos. Una forma de ver la distorsión de una cámara es mediante los gráficos que se muestran a continuación:

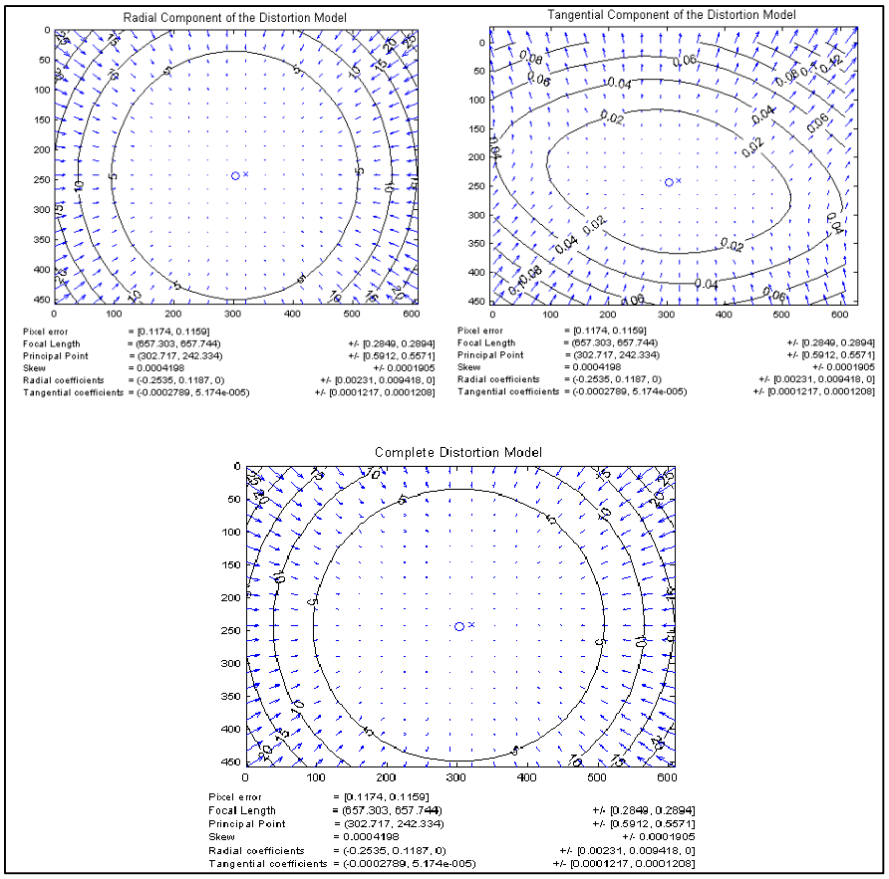

Figura 5: *De arriba a abajo y de izquierda a derecha, modelo de distorsión radial, tangencial y completo.*

Para corregir estas irregularidades es necesario calibrar la cámara. Como antes se dijo, es un proceso importante para que el posterior tratamiento de las imágenes brinde un resultado adecuado.

3.1.3 Calibración

La cámara y los puntos que se quieren calibrar no están en el mismo plano de referencia. Por tanto, un primer paso es rotar y trasladar el sistema de referencia de los puntos al plano de la cámara. Para ello, primero habrá que rotar en torno al eje z, el eje y y el eje x. Para conseguirlo, se han de aplicar las siguientes transformaciones matriciales:

$$R_x(\vartheta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\vartheta & \sin\vartheta \\ 0 & -\sin\vartheta & \cos\vartheta \end{bmatrix}$$

$$R_x(\varphi) = \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{bmatrix} \qquad \textbf{Ecuación 3.A.6}$$

$$R_x(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Las rotaciones se van aplicando a cada eje. Empezando por rotar alrededor del eje z, el eje y y el eje x. A continuación, se tendría que trasladar el sistema de coordenadas rotado hasta el plano de la cámara. Se representará como un vector para desplazarse de un punto a otro:

$$T = (x, y, z)_{objeto} - (x, y, z)_{camara} \qquad \textbf{Ecuación 3.A.7}$$

Con la matriz de rotación y el vector de traslación se pueden pasar todos los puntos de la escena a la cámara. Combinado con los parámetros intrínsecos de la cámara, se obtendrá la calibración que se busca.

En general, un proceso de calibración se basa en comparar los datos tomados por un sensor (o instrumentos en general), que para este caso será una cámara y un patrón de referencia (la imagen que se formaría usando una lente ideal). Para conseguirlo, se tiene un patrón de referencia fácilmente interpretable por software: un tablero de ajedrez.

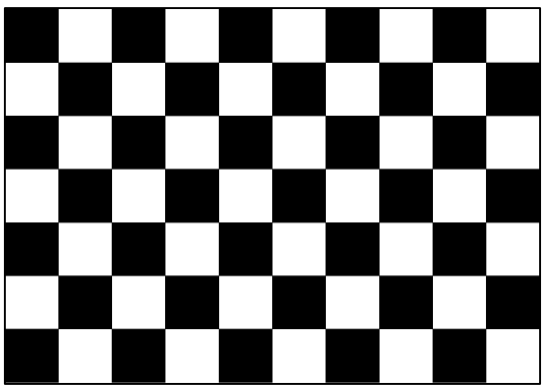

Figura 6: *Patrón de calibración, tablero de ajedrez.*

La interpretación es muy sencilla para el software, pues está formado por colores opuestos, los bordes de la imagen están muy bien definidos y las esquinas serán fácilmente identificables. Se puede usar cualquier otro objeto; sin embargo, los tableros son bastante populares y ampliamente utilizados.

A partir de la imagen tomada, hay que plasmarla en el sensor a través de una homografía [20]. Para ello se necesita una matriz de homografía y dos puntos, uno de la escena tomada q' y otro del lado del sensor q. Se usarán coordenadas homogéneas como ya se explicó anteriormente:

$$q' = [x', y', z', 1]^T \qquad q = [x, y, 1]^T \qquad \textbf{Ecuación 3.A.8}$$

Se puede expresar la homografía como sigue, introduciendo un valor de escala s que da a entender que la matriz de homografía se define sólo por ese factor:

$$q = sHq' \qquad \textbf{Ecuación 3.A.9}$$

La matriz de homografía H se divide en una parte física (dónde se localiza el objeto en la imagen) y una segunda parte para introducir los parámetros intrínsecos de la cámara. Luego, la relación anterior quedará como:

$$q = sMWq' \qquad \textbf{Ecuación 3.A.10}$$

La parte física de la transformación es la matriz W, que será la suma del efecto de la rotación y la traslación descrita anteriormente. La rotación venía dada por una matriz, sin embargo, la traslación era un vector. Pero como se está trabajando en un sistema de coordenadas homogéneo, se podrían combinar:

$$W = [R \quad T] \qquad \textbf{Ecuación 3.A.11}$$

Y la parte que introduce los parámetros intrínsecos es M, que tal y como se introdujo, queda definida como sigue:

$$M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \qquad \textbf{Ecuación 3.A.12}$$

Sin pérdida de generalidad, se puede decir que el plano del objeto se corresponda con el plano $z = 0$. Así se obtienen simplificaciones interesantes de cara al cálculo:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = s \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} [r_1\ r_2\ r_3\ \overline{t}][x\ y\ 0\ 1]^T = sM[r_1 \quad r_2 \quad \overline{t}][x \quad y \quad 1] = H[x \quad y \quad 1]$$

Ecuación 3.A.13

La matriz de homografía se convierte en una matriz cuadrada de tres filas y columnas. Nos describe la posición de un punto del plano de la imagen a un punto del plano del sensor por una simple relación matricial:

$$q = Hq' \rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = H \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Ecuación 3.A.14

Mediante algoritmia que se explicará en las siguientes secciones, se puede obtener una imagen rectificada y sin distorsión basándose en las matrices y parámetros descritos hasta ahora.

3.2 Visión estéreo

Si en técnicas de visión monocular se usa una única cámara como sensor, en las técnicas de visión estéreo serán necesarias dos cámaras bien configuradas. En este caso, se pretende reconstruir la profundidad tal y como lo hacen los humanos. Dos cámaras hacen las veces de ojos, y la algoritmia que las controla hará de cerebro.

Las capturas de dos cámaras separadas y, en principio, no situadas en el mismo plano, darán lugar a dos capturas diferentes. La base para recuperar la dimensión perdida es, precisamente, la diferencia entre esas dos imágenes. El proceso a seguir es:

- Obtener las imágenes del par estéreo rectificadas y sin distorsión. Tal y como se explica para visión monocular, pero aplicado ahora a dos cámaras.

- Buscar correspondencias entre puntos de ambas imágenes. Así se obtiene un mapa de disparidad. Donde la disparidad es la diferencia en el eje x entre dos puntos correspondientes de la

imagen izquierda y derecha:

$$d = x_l - x_r$$ ***Ecuación 3.A.15***

- Con la geometría de las cámaras, podemos triangular los puntos. A éste proceso se le llama re-proyección [21], y lo que se obtiene es un mapa de profundidad.

El modelado de las dos cámaras es el mismo que para una única cámara, pero realizado de forma independiente para cada uno de ellos. Se omite el estudio del modelado para no ser redundantes, pues se expuso previamente en la sección de visión monocular.

3.2.1 Configuraciones de pares estéreo

Por lo general, un par estéreo se forma con dos cámaras. La orientación y la distancia entre ellas puede ser aleatoria; sin embargo, tener una buena configuración de par facilita el procesamiento. La configuración más genérica es:

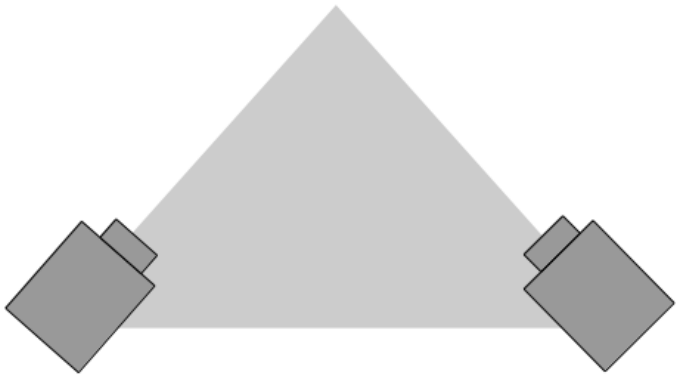

Figura 7: *Configuración genérica de un par estéreo.*

En esta tesis se ha optado por un modelo que simplifica bastante el proceso de calibración (de nuevo será crítico para obtener resultados correctos) y la geometría del problema: las cámaras se sitúan en el mismo plano. Así los sensores estarán a la misma distancia de la escena. La configuración escogida es, por tanto la de la Figura 8 pues al estar las cámaras alineadas en el mismo plano facilita toda la eficiencia de la lógica posterior (principalmente el cálculo de puntos característicos y el matching):

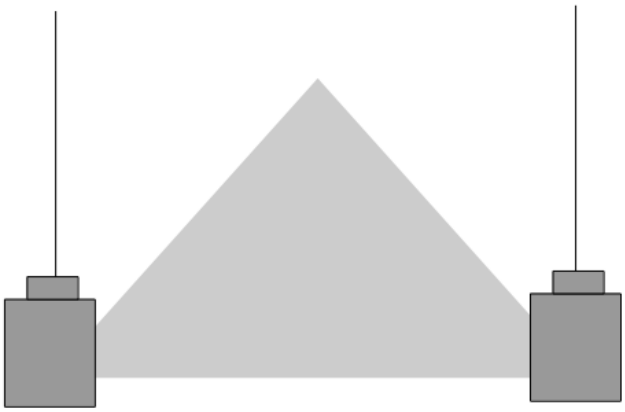

Figura 8: *Configuración simplificada del par estéreo.*

Se asume que los planos imagen de las cámaras son coplanarias, el eje óptico de ambos sistemas es paralelo e igual distancia focal para las dos cámaras. Además, se supone que los puntos principales de ambas cámaras han sido bien rectificados, y tienen las mismas coordenadas.

Lo más interesante a la hora de crear un par estéreo es la triangulación y el problema de correspondencia. Por ello, se va a suponer que las imágenes están corregidas (sin distorsión, alineadas y rectificadas) tal y como se explica en la sección de técnicas monoculares.

3.2.2 Triangulación

En la configuración coplanaria descrita anteriormente, el escenario es el siguiente:

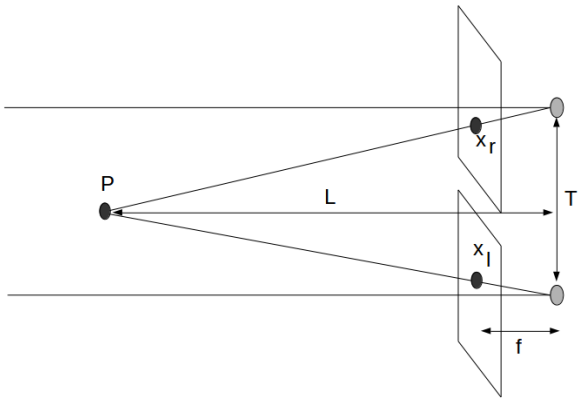

Figura 9: *Triangulación en par estéreo coplanaria.*

Por simple triangulación, se puede sacar la siguiente relación:

$$\frac{T-(x_l-x_r)}{L-f}=\frac{T}{L}\rightarrow L=\frac{fT}{x_l-x_r}\rightarrow L=\frac{fT}{d} \qquad \textbf{Ecuación 3.A.16}$$

Se puede ver que la profundidad (L) es inversamente proporcional a la disparidad (d). Es decir, los objetos más cercanos tendrán mayor disparidad (menos profundidad) y objetos lejanos tendrán menor disparidad (mayor profundidad).

Como cabe esperar, se puede modelar el par estéreo matemáticamente. En ésta ocasión, se tendrán dos matrices que darán información sobre los parámetros intrínsecos de las cámaras y la posición relativa entre ambas cámaras en el espacio. Las matrices que dan dicha información son:

- **Matriz esencial** [22]. Es la que indica dónde se localiza un mismo punto del espacio en la cámara izquierda y en la derecha. Informa sobre conceptos puramente geométricos de la estructura. Relaciona un punto del espacio (P) con su proyección en los sensores de la siguiente forma:

$$p_d^T E p_i = 0 \qquad \textbf{Ecuación 3.A.17}$$

- **Matriz fundamental** [23]. Relaciona los puntos del plano imagen de una cámara en puntos en el plano imagen de la otra cámara (en coordenada de píxeles). De forma análoga, se obtiene que la relación entre los puntos en el plano imagen de ambas cámaras es:

$$q_d^T F q_i = 0 \qquad \textbf{Ecuación 3.A.18}$$

  Si se tienen las imágenes completamente rectificadas y normalizadas por su distancia focal, se cumplirá que:

$$E = F \; y \; M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \textbf{Ecuación 3.A.19}$$

3.3 OpenCV

Para llevar a cabo lo anteriormente explicado, se ha usado Open Computer Vision Library (en adelante OpenCV), una biblioteca de visión artificial, utilizada en muchas aplicaciones de visión por computador actualmente. Su popularidad ha ido aumentando desde su lanzamiento en 1999. Se distribuye bajo licencia Berkeley Software Distribution (en adelante BSD), permitiendo ser usada libremente. Incluye usos comerciales y desarrollos de Investigación, Desarrollo e Innovación (en adelante I+D+I).

Escrito para C, C++, Python y Java, fue escrito pensando en eficiencia. Además, es multiplataforma, lo que permite desarrollar fácilmente para diversas máquinas. Entre la multitud de funciones que posee, las más interesantes para el estudio objeto de esta tesis han sido las relativas a calibración, búsqueda de bordes y correspondencia entre imágenes.

3.3.1 Calibración con OpenCV

Como se viene haciendo hasta ahora, se va a diferenciar entre el caso de cámara monocular y par estéreo. Aunque existen elementos comunes en ambos casos, por la naturaleza de las configuraciones, no van a ser procesos equivalentes.

3.3.1.1 Calibración monocular

OpenCV tiene muchas funciones para la calibración de cámaras usando el tablero de ajedrez citado anteriormente. Los pasos a seguir para conseguir una buena calibración son los siguientes:

- Buscar en las capturas el patrón escogido, el tablero de ajedrez. Se detecta y se dibuja en caso de ser necesario:

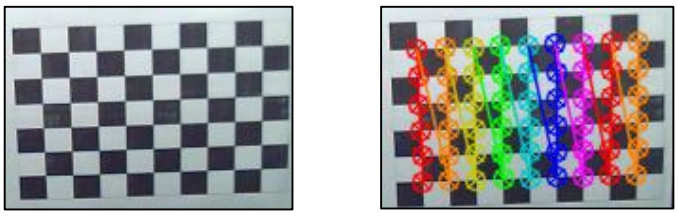

Figura 10: *Patrón de referencia a la izquierda y esquinas detectadas a la derecha.*

- Como el patrón es fácilmente reconocible y se sabe como tiene que ser, se procede al cálculo de la matriz de parámetros intrínsecos (M) y el vector de distorsión.

- Se obtiene una matriz de igual tamaño a la imagen original para el mapeo de puntos sin distorsión. Tras este punto, se obtiene la imagen rectificada.

3.3.1.2 Calibración estéreo

El proceso de calibración estéreo es esencialmente el mismo que el anterior, pero aplicado a dos cámaras. Sin embargo, tiene la diferencia de que el resultado han de ser dos imágenes perfectamente alineadas, rectificadas y sin distorsión para posterior tratamiento y obtención de profundidad. El proceso seguido es:

- Encontrar el patrón en las capturas de cada una de las cámaras. Como en el caso anterior, se pueden dibujar sobre la imagen en caso de requerirlo.

- Obtención de los vectores de distorsión y matriz de parámetros intrínsecos de ambas cámaras.

- Cálculo de la matriz esencial y la fundamental, así como la rotación y traslación de los sistemas de coordenadas necesario para la posterior rectificación.

- Cómputo de las coordenadas ideales de los puntos a partir de las imágenes observadas. Proceso a aplicar en ambas cámaras por separado.

- Cálculo de la correspondencia entre líneas de ambas imágenes (siguiendo geometría epipolar [24]).

- Finalmente, se quita la distorsión de las imágenes con el vector de distorsión obtenido y se mapean las nuevas coordenadas de los píxeles rectificados. Se obtienen las deseadas.

3.3.2 Correspondencia entre imágenes

El proceso de correspondencia (aplicable a pares estéreo) es uno de los más costosos en cuanto a consumo de recursos de la máquina. Por ello, una buena calibración de las cámaras y rectificación de las imágenes es necesaria. El proceso será más liviano si los dos objetivos anteriores se cumplen.

Se puede obtener la profundidad a partir de la disparidad entre dos píxeles de las imágenes rectificadas. Para ello, se ha hecho uso de técnicas de Block Matching (en adelante BM) [25], que usa una ventana para el cálculo de la diferencia absoluta entre píxeles. Como es un algoritmo que toma puntos con alto contraste de texturas, para uso en exteriores la mayoría de los puntos tendrán asociada una profundidad. En interiores, un menor número de puntos serán asociados con una profundidad.

El algoritmo usado por OpenCV (con BM muy rápido y optimizado) funciona para imágenes rectificadas y sin distorsión. De ahí la importancia de la calibración, rectificación y eliminado de la distorsión. En general, el algoritmo sigue tres fases:

- Procesado inicial de las imágenes para realzar las texturas y obtener, posteriormente, mejor reconstrucción de la profundidad.

- Búsqueda de correspondencia horizontalmente a lo largo de las líneas de las imágenes. En éste punto se usa la citada ventana para el cálculo de la diferencia absoluta de píxeles.

- Procesado final para eliminar malas correspondencias y falsas medidas de profundidad.

La disparidad [26]en la captura de una cámara tiene que ocurrir en la misma línea de la captura de la otra cámara. Mediante el algoritmo aplicado, podemos limitar la búsqueda a cierto número de píxeles:

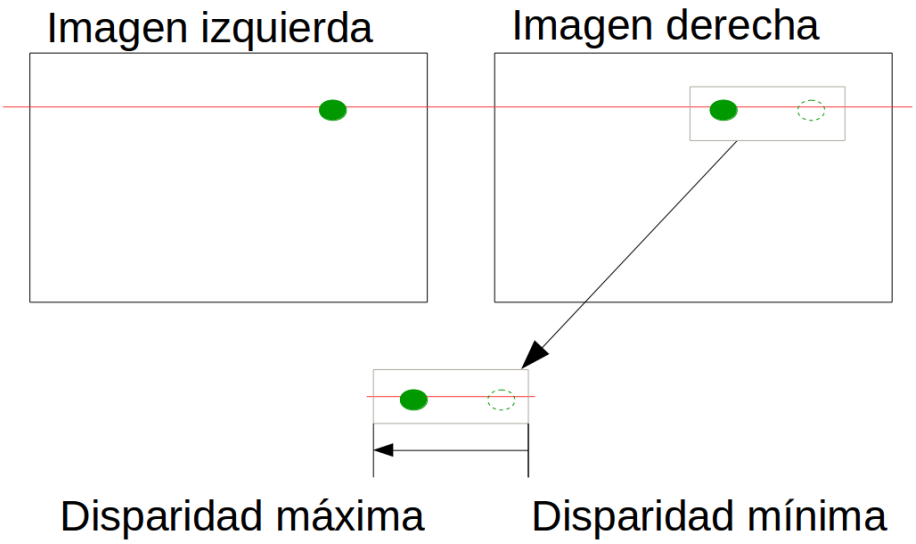

Figura 11: *Disparidad entre capturas de la cámara izquierda y derecha.*

En la figura 11 se puede observar que la imagen capturada por una y otra cámara difieren. Mediante el algoritmo proporcionado por OpenCV, se puede modificar la disparidad máxima y mínima a detectar, así como el rango de disparidad a tener en cuenta, el número de píxeles a buscar la disparidad, el tamaño de la ventana deslizante usada y demás parámetros. Modificar unos parámetros u otros, dará lugar a mapas de disparidad diferentes.

Mediante la disparidad se llega al objetivo final, reconstruir la profundidad:

$$D = f\frac{b}{d} \qquad \textbf{Ecuación 3.A.20}$$

Donde D es la profundidad, f es la distancia focal de la cámara, b es la separación entre las cámaras del par estéreo y d es la disparidad calculada.

Además, si se tiene un rango de disparidad Δd, se puede obtener el rango máximo de profundidad obtenible aplicando la siguiente relación (profundidad máxima que se puede sacar de la escena):

$$\Delta D = f\frac{D}{b}\Delta d \qquad \textbf{Ecuación 3.A.21}$$

Resultados obtenidos de éste proceso se pueden observar en las siguientes imágenes:

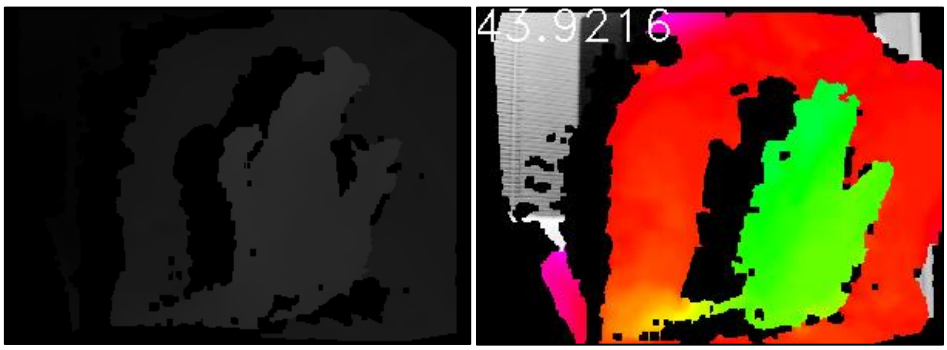

Figura 12: *De izquierda a derecha, mapa de profundidad en blanco y negro y mapa de profundidad en Red Green Blue (en adelante RGB) junto con la distancia (en cm.) al objeto más cercano.*

En este caso, la disparidad en blanco y negro muestra que cuanta más componente blanca, mayor cercanía del objeto. En la imagen en RGB, el color verde corresponde a una mayor cercanía.

3.3.3 Técnicas de cálculo de disparidad

Existen diversas técnicas para el cálculo de la disparidad a partir de pares estéreo, algunas basadas en el ya citado algoritmo de BM, aunque también algunas otras basadas en métodos diferentes.

En cuanto al BM hay dos métodos muy utilizados que también se ha utilizado para la obtención de los mapas de disparidad y cálculo de profundidad en las diferentes pruebas realizadas. Estos son el método Semi Global Block Matching (en adelante SGBM) [27] y el método BM [28]. Se diferencian fundamentalmente en la forma que tienen cada uno de calcular la disparidad.

El SGBM divide la imagen en ventanas, pero a la hora de calcular emparejamientos no busca similitudes individuales entre píxeles, sino que intenta minimizar una función de energía a nivel de ventana. El algoritmo BM, por su parte, primero se encarga de filtrar y normalizar la intensidad de la imagen, luego busca correspondencias en las líneas epipolares y finalmente filtra los resultados para eliminar falsos positivos. Si bien ambos algoritmos calculan la disparidad de manera diferente, el resultado que podemos obtener de ambos es bastante similar.

Por otro lado, existe otra técnica basada en el algoritmo GraphCuts [29] que suele ser muy utilizada para obtener una gran precisión en el cálculo de correspondencias debido a que aplica optimizaciones sucesivas basadas en el teorema max-flow min-cut [30] aplicadas sobre los parámetros de color. Tras dichas iteraciones se obtiene un grafo de podas mínimas, pero con un flujo máximo, que se corresponde con la estimación máxima a posteriori de la solución.

El problema de este método es definitivamente su gran lentitud, en comparación con los métodos de BM. Sin embargo, con este método se puede obtener cálculos de disparidad muy precisos, lo cual implica una mayor exactitud para el cálculo de distancias, sobre todo cuando éstas son elevadas.

Una técnica mucho más eficiente en términos de rendimiento computacional consiste en el cálculo de profundidad a partir de la correspondencia de intensidad lumínica línea a línea de los píxeles del par estéreo. Sin embargo, hay que tener en cuenta que las líneas epipolares de las imágenes tienen que coincidir con líneas horizontales. Es decir, el punto de partida son dos imágenes lo más perfectamente rectificadas y con la menor distorsión.

El algoritmo se basa en la luminosidad de los píxeles de las imágenes del par estéreo convertidas a escala de grises. Es decir, se tendrán ristras de números enteros entre cero y doscientos cincuenta y cinco que habrá que analizar.

El algoritmo sigue varios pasos bien diferenciados:

- Cálculo del mínimo coste del matching entre dos píxeles. Si cada línea de una imagen se corresponde con la línea de la otra, buscar correspondencia entre píxeles será recorrer la línea donde se encuentre. Se establece un máximo número de píxeles sobre los que buscar (una disparidad máxima). Se almacenará la mínima diferencia entre intensidades.

- Establecer un coste mínimo y máximo. Permitirá elegir qué píxeles son los más parecidos y lo que se encuentra entre ellos. Se define una diferencia entre intensidades máxima y mínima. Si se encuentra entre esos márgenes, el píxel se toma como un buen matching, y el resto como píxeles diferentes.

- Obtención de la disparidad. Una vez se obtienen los píxeles que, en teoría serán los mismo, se cuentan los píxeles que los separan. Esta medida es la disparidad.

- Escalado. A partir de la disparidad, con un simple escalado para aprovechar los doscientos cincuenta y seis valores posibles obtendremos la aproximación a la profundidad.

Comparado con los métodos usado por OpenCV y dependiendo del tamaño de la imagen se ha obtenido mejoras de entre cien y ciento veinte milisegundos. Sin lugar a dudas, la profundidad obtenida no será tan precisa, pero el objetivo es una detección rápida capaz de correr en un procesador de tipo Advanced RISC Machine (en adelante ARM) o incluso en un microcontrolador.

Para ello, se ha intentado independizar lo más posible de OpenCV y de cualquier biblioteca de visión por computador. El algoritmo se limita a calcular operaciones en un vector de números enteros. Hasta el momento, lo único para lo que se usa OpenCV es para copiar la imagen a una matriz y para mostrar los resultado.

Se han hecho pruebas para obtener la imagen a través de la segunda versión de Video for Linux (en adelante V4L) para independizar completamente de OpenCV. No se obtuvieron buenos resultados, aunque se ha comprobado que es factible.

Para cada par estéreo habrá que configurar los parámetros que más se adapten. Dependerá de la resolución de la imagen a analizar, el sincronismo entre capturas y el rango de búsqueda que queramos tener en cuenta. Algunos resultados del algoritmo explicado se muestran a continuación:

Figura 13: *Diferentes resultados para el algoritmo de disparidad desarrollado.*

En la segunda imagen se puede ver un shell indicando el tiempo que tarda en calcular la profundidad medida en milisegundos. Se puede observar que ronda entre quince y veinte milisegundos.

Sin embargo, aún no se ha basado la detección y el mapeado con este algoritmo por falta de pruebas y obtención de resultados que pueden ser mejorados.

3.3.4 Extracción de características de la imagen y búsqueda de contornos

En OpenCV existen muchos métodos para realizar la detección y extracción de puntos característicos de la imagen, así como optimizaciones para quedarnos solo con los que nos sean más útiles para procesamiento posterior. Por ejemplo, se pueden obtener puntos invariantes de la imagen con el detector de esquinas Harris [31] o mediante el algoritmo Kanade-Lucas-Tomasi (en adelante KLT) [32].

Ambos métodos han sido testeados y se han obtenido resultados como los mostrados en la siguiente imagen:

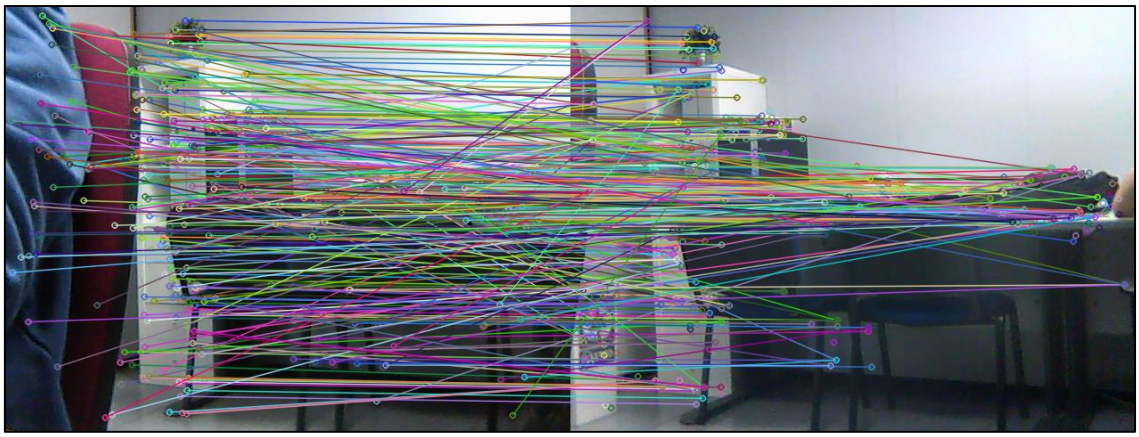

Figura 14: *Búsqueda de puntos característicos en imágenes estéreo.*

Hay que destacar que dado que el cálculo de puntos característicos conlleva un tiempo de ejecución elevado, y que no todos los puntos de la imagen cumplen con las cualidades necesarias para ser un punto característico, la extracción de estos puntos no debe utilizarse para el cálculo de disparidad.

En cambio, y aprovechando la invariabilidad de los puntos característicos en la aplicación de distintas transformaciones, estas

técnicas se utilizan sobre todo para realizar la estimación de poses (la combinación de la posición y orientación de los objetos relativos a algún sistema de coordenadas) en relación al movimiento de la cámara, para así poder estimar precisamente la posición y orientación de la cámara en cualquier instante. Este problema, conocido en el ámbito de la robótica como odometría visual [33] es uno de las partes fundamentales en la solución del problema de la Localización y Mapeado Simultaneo (en inglés Simultaneous Localization and Mapping, y en adelante SLAM) [34] [35].

Un método ampliamente utilizado para la detección y extracción de puntos característicos es el algoritmo Scale-Invariant Feature Transform (en adelante SIFT) [36]. Como su propio nombre indica, los puntos de interés que obtiene son invariantes al escalado. Además, son muy fiables a cambios en la iluminación y el ruido, debido a que la información para obtenerlos se saca de los contrastes de iluminación de la imagen. Más adelante, en el BLOQUE B de esta tesis, se explicarán más detalladamente el algoritmo SIFT, así como otros métodos de extracción de características.

Otro problema recurrente a la hora de enfrentarse a detección mediante visión artificial es la segmentación de imágenes. Para dicha tarea es de gran utilidad las técnicas de búsqueda de contornos. Para superar esta brecha, se ha hecho uso de diversas técnicas. Desde el tradicional algoritmo de Canny [37] para la búsqueda de contornos hasta técnicas más sofisticadas basadas en la reconstrucción del árbol completo de contornos.

En la siguiente imagen se muestra una captura de un vuelo real, con el correspondiente procesado de la imagen en búsqueda de los contornos más característicos:

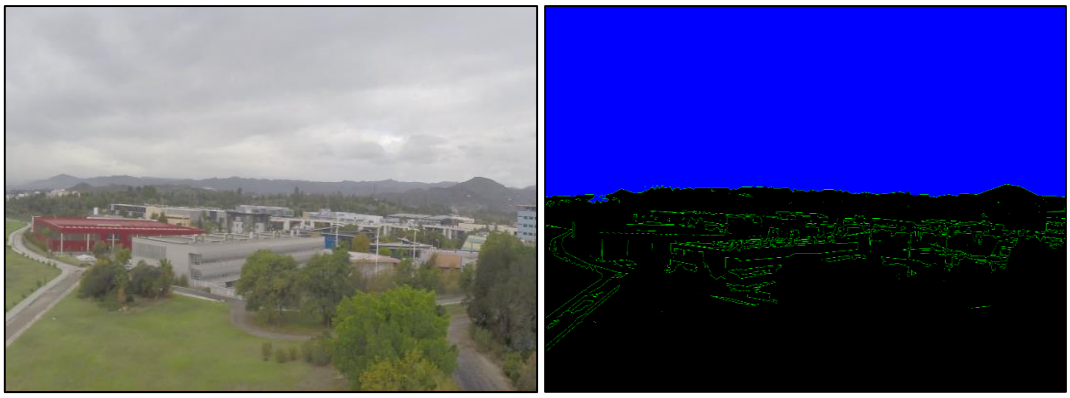

Figura 15: *Detección de bordes en un video en vuelo real.*

En una imagen en alta definición calcula y dibuja los contornos en tiempo real. Además, se calcula los momentos de la imagen para posteriores cálculos, por ejemplo, el área de los contornos encontrados.

Las capturas anteriores trabajan sobre una configuración monocular. Es una técnica aplicable a configuración estéreo. De hecho, ha sido también probada con éxito para dicha configuración, obteniéndose resultados muy positivos aplicados a la imagen de disparidad directamente.

3.3.5 Seguimiento del objeto más cercano

El seguimiento de objetos es también un problema a resolver en la visión artificial, más aún cuando el fondo de la imagen no es estático, como en este caso. Para la configuración estéreo, se puede obtener la posición del objeto más cercano haciendo uso de múltiples funciones de OpenCV. La siguiente captura muestra la posición del siguiente objeto:

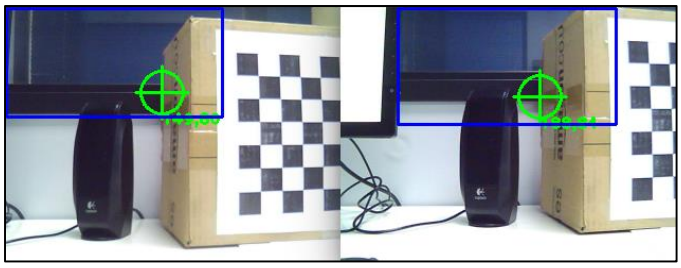

Figura 16: *Objeto más cercano de la escena (la caja con el patrón de calibración) y la posición de un pixel.*

Haciendo uso de varias técnicas simultáneamente, se puede seguir el objeto más cercano y saber a qué distancia se encuentra. La técnica de seguimiento se usa tanto para cámaras estéreo como para configuraciones monoculares. Lo único que lo diferencia es que para configuraciones estéreo, hay que buscar la correspondencia entre los puntos encontrados.

3.3.6 Flujo de píxeles

Referente a la odometría visual, que se explicará en el BLOQUE B relativo al módulo de posicionamiento y mapeado, es de gran utilidad el conocimiento del flujo de píxeles, o flujo óptico. En este caso, se ha elaborado un algoritmo para configuración monocular. En la siguiente imagen se puede ver el resultado:

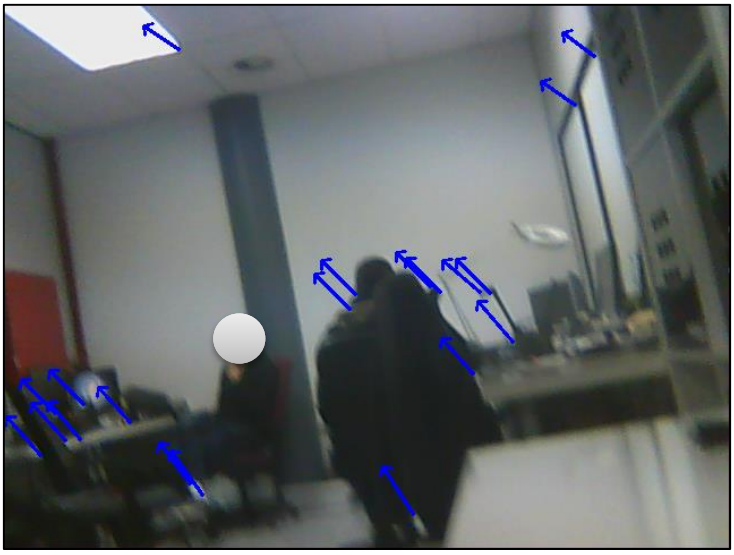

Figura 17: *Flujo de píxeles en configuración monocular.*

Para éste algoritmo, se toman los mejores puntos característicos de la imagen en un cuadro, se adquiere el siguiente cuadro del flujo y se calculan los puntos característicos. En general, como la captura se realiza en unos pocos milisegundos, los puntos tomados en uno y otro cuadro serán los mismos. Finalmente, se hace un matching de puntos.

El resultado visual es una flecha que apunta al punto característico del siguiente cuadro, y tiene el ángulo y la magnitud del movimiento

realizado. En la figura 17, el flujo de píxeles muestra un movimiento de la cámara no muy pronunciado hacia arriba y ligeramente ladeado hacia la izquierda.

3.4 Sensores para visión artificial

Para llevar a cabo lo anteriormente descrito, se han empleado configuraciones estéreo y monoculares. Las cámaras utilizadas se describen a continuación.

3.4.1 Minoru 3D Webcam

Una cámara estéreo de bajo coste, con prestaciones suficientes para obtener mapas de profundidad para la detección de objetos. Sus características técnicas más importantes son las listadas a continuación:

- Sensor Complementary Metal-Oxide Semiconductor [CMOS] (Rolling Shutter [38]).
- Separación entre cámaras: 6 cm.
- Tasa de cuadros máxima: 30 fps.
- Control de la distancia focal manual (desde 10 cm.).
- Campo de visión de 42º.
- Calibración requerida.

Figura 18: *Frontal de la cámara Minoru 3D Webcam.*

Actualmente en funcionamiento, es capaz de obtener distancias tanto en exteriores como en interiores. Diseñada como cámara web, trabaja mucho mejor en las condiciones de iluminación de interiores. Sin embargo, es operativa en exteriores.

3.4.2 Logitech Webcam C210

Cámara web con mayores prestaciones que la anterior. Con posibilidad de usar tanto en configuración estéreo como monocular. Sin embargo, se obtienen peores resultados debido a la dificultad a la hora de calibrar. Por ello, ha sido descartada en cuanto a configuración estéreo.

Figura 19: *Logitech Webcam C210.*

Para el caso de configuración monocular, también se ha descartado, puesto que no es una cámara Global Shutter. No se van a obtener mejoras sustanciales en relación a usar una única cámara del par estéreo que forma Minoru 3D Webcam.

3.4.3 IDS UI-1221LE

Las cámaras UI-1221LE están equipadas con un sensor Wide Video Graphics Array (en adelante WVGA) monocromo y una lente fisheye que permite un ángulo de visión de, aproximadamente, 180º en la horizontal. Una de las características más importantes de estas cámaras es que tienen sensor Global Shutter [39].

Un sensor Global Shutter tiene la característica de que cada pixel en el sensor comienza y termina su exposición simultáneamente, de

forma que proporciona una capacidad de captura de imágenes en el sensor en una vez, lo que proporciona gran eficiencia y sincronización, especialmente en la creación de escenas en 3 dimensiones (en adelante 3D).

Por otro lado, los sensores Rolling Shutter definen un array de diferentes líneas para la imagen, que son expuestas en tiempos diferentes tanto en cuando la función de lectura va desplazándose a lo largo del sensor barriéndolo línea por línea. Al menos se requieren 20 milisegundos desde que se activa la exposición hasta que se realiza la lectura, por lo que puede afectar significativamente a la sincronización y a la distorsión espacial.

Por tanto, para que los resultados sean más fiables es recomendable el uso de cámaras Global Shutter en vez de Rolling Shutter.

Para visión artificial, se han conectado dos de estas cámaras mediante un soporte para permitir la visión estéreo. Las características de estas cámaras son:

- Sensor CMOS (Global Shutter*)*.
- Separación entre cámaras: 12 cm.
- Resolución máxima: 0.36 Mpx. (752x480)
- Frecuencia máxima de imagen: 87.2 fps
- Campo de Visión: ~180º
- Calibración requerida

Estas cámaras dan la posibilidad de ajustar la exposición y ganancia lumínica de forma automática, por lo que son capaces de ajustar la imagen dependiendo de la fuente de luz que se tenga en el entorno.

Figura 20: *Módulo de dos cámaras IDS UI-1221LE en modo estéreo.*

3.5 Robot Operating System

3.5.1 Introducción

Robot Operating System (en adelante ROS) ***[²]*** es un sistema operativo que proporciona múltiples librerías para el desarrollo de aplicaciones relacionadas con la robótica. ROS se desarrolló originalmente en 2007 bajo el nombre de switchyard por el Laboratorio de Inteligencia Artificial de Stanford para dar soporte al proyecto del Robot con Inteligencia Artificial de Stanford: STAIR2). Desde 2008, el desarrollo continúa en Willow Garage, un instituto de investigación robótico con más de veinte instituciones colaborando en un modelo de desarrollo federado.

ROS proporciona, entre otras cosas, drivers para conectar con dispositivos hardware, librerías que implementan multitud de algoritmos, herramientas de visualización e interfaces para interactuar con las aplicaciones. Además, ROS también incluye un sistema de manejo de paquetes para facilitar la integración de nuevas aplicaciones en el sistema.

ROS tiene dos partes básicas: la parte del sistema operativo (ROS) y ros-pkg, una suite de paquetes aportados por la contribución de usuarios que implementan las funcionalidades tales como localización y mapeo simultáneo, planificación, percepción, simulación, etc.

ROS es software libre bajo términos de licencia BSD Esta licencia permite libertad para uso comercial e investigador. Las contribuciones de los paquetes en ros-pkg están bajo una gran variedad de licencias diferentes.

A continuación, se mostrará una lista con las áreas de la ingeniería en las que se utiliza ROS:

- Percepción
- Identificación de Objetos
- Segmentación y reconocimiento

[²] *http://www.ros.org/ (último acceso: 15-sept-2015)*

- Reconocimiento facial
- Reconocimiento de gestos
- Seguimiento de objetos
- Odometría visual
- Comprensión de movimiento
- Estructura de movimientos (en inglés Structure from Motion, y en adelante SFM)
- Visión estereoscópica
- Movimientos
- Robots móviles
- Control
- Planificación
- Agarre de objetos

ROS es especialmente útil en el área de visión artificial, ya que proporciona librerías tanto para el manejo de cámaras estereoscópicas y monoculares, como para la visualización de mapas 3D.

Una versión de ROS puede ser incompatible con otra. Normalmente están referidas por un nombre de distribución en vez de por una versión numérica. Las versiones, desde la más actual a la primera versión, son:

- 22/Julio/2014 - Indigo Igloo
- 04/Septiembre/2013 - Hydro Medusa
- 31/Diciembre/2012 - Groovy Galapagos
- 23/Abril/2012 - Fuerte

- 30/Agosto/2011 - Electric Emys
- 02/Marzo/2011 - Diamondback
- 03/Agosto/2010 - C Turtle
- 01/Marzo/2010 - Box Turtle
- 22/Enero/2010 – ROS 1.0

En los siguientes apartados se hablará sobre los elementos de ROS que se han utilizado para la detección de riesgos y se describirán sus características más importantes.

3.5.2 Sistema de Archivos de ROS

En el sistema de archivos de ROS se puede encontrar:

- Paquetes: Los paquetes son la unidad de organización software del código ROS Cada paquete puede contener librerías, ejecutables, scripts u otros artefactos.
- Manifest (paquete .xml): Un archivo manifest contiene la descripción de los paquetes. Esto nos sirve para definir dependencias entre paquetes y para obtener los metadatos e información sobre los paquetes como la versión, las licencias, etc.

Para controlar el sistema de archivos de ROS se necesita un sistema de construcción, que es el responsable de generar ejecutables, a partir de código fuente, que puedan ser utilizados por un usuario final. Estos ejecutables pueden estar en forma de librerías, programas ejecutables, scripts, interfaces (por ejemplo un archivo de cabecera C++) o de cualquier otro tipo que no sea código estático. En términos de ROS, el código fuente está organizado en paquetes, donde cada paquete consiste en uno o más archivos ejecutables.

Para construir ejecutables, el sistema necesita información como la localización de los componentes, localización del código fuente, las dependencias del código, las dependencias externas, etc. Esta información se encuentra normalmente en los ficheros de configuración, los CMakeLists.txt, que serán usados por el compilador

para construir el código fuente y generar los ejecutables en el orden adecuado.

ROS utiliza un sistema de construcción propio llamado Catkin. Catkin combina CMake y scripts de Python para proporcionar funcionalidad al flujo de trabajo. Catkin fue diseñado para tener una mejor distribución de los paquetes de ROS, un mejor soporte de compilación y una mejor portabilidad.

El sistema de archivos de Catkin tiene la siguiente estructura:

```
workspace_folder/        -- WORKSPACE
  src/                   -- SOURCE SPACE
    CMakeLists.txt       -- 'Toplevel' CMake file, provided by catkin
    package_1/
      CMakeLists.txt     -- CMakeLists.txt file for package_1
      package.xml        -- Package manifest for package_1
    ...
    package_n/
      CMakeLists.txt     -- CMakeLists.txt file for package_n
      package.xml        -- Package manifest for package_n
```

3.5.3 Elementos de ROS

ROS dispone de diferentes elementos que pueden interconectarse entre sí para realizar diversas tareas. Los más importantes son:

- Nodos: Un nodo de ROS es realmente un archivo ejecutable que se encuentra dentro de un paquete ROS Los nodos de ROS utilizan librerías de cliente ROS para comunicarse con otros nodos. Cada nodo de ROS puede enviar o recibir información de un topic. Los nodos también pueden proporcionar o utilizar un servicio.

- Nodelet: Un nodelet proporciona una forma de ejecutar múltiples algoritmos en un mismo proceso sin realizar cambios de contexto. Normalmente son llamados por un nodo.

- Mensajes: Información que un nodo puede usar cuando se suscribe a un topic o cuando publica datos en un topic.

- Topic: Los topic son conexiones a través de los cuales los nodos envían mensajes. Los topic tienen una semántica anónima de publicación/suscripción, lo que significa que un nodo no sabe realmente a través de que topics se está comunicando.

- Servicio: Un servicio es otra forma a través del cual los nodos pueden comunicarse entre sí. A diferencia de los topic, en un servicio un nodo envía peticiones y recibe respuestas a las mismas.

3.5.4 Grafo de Conexiones

Como se ha comentado en el apartado anterior, ROS proporciona múltiples elementos para la interconexión de servicios. Dado que es muy posible que en la aplicación tengamos muchos nodos y conexiones activas, ROS proporciona una herramienta para poder visualizar el esquema del inter-conexionado: el grafo de conexiones.

El grafo de conexiones puede iniciarse ejecutando el nodo rqt_graph, y mostrará todos los nodos que están ejecutándose en el sistema y todas las conexiones activas. En la siguiente imagen se puede ver un ejemplo del grafo de conexiones.

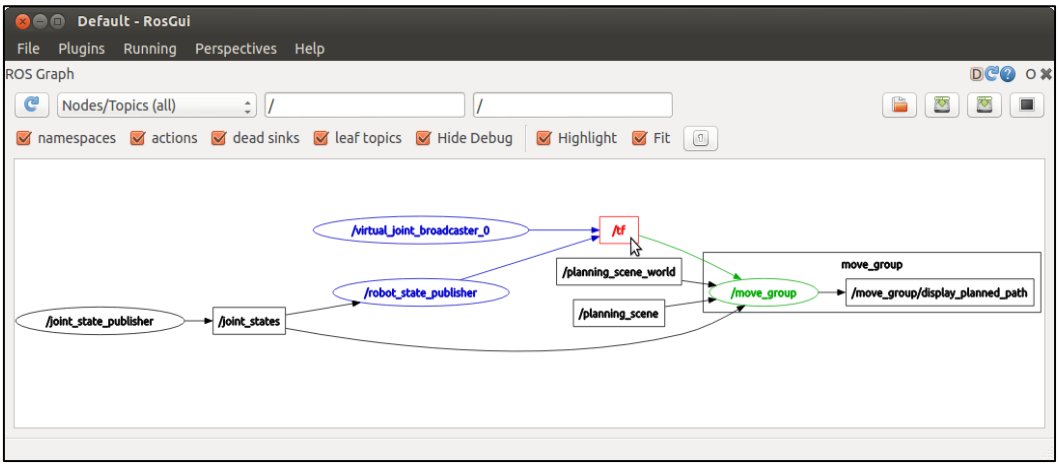

Figura 21: *Grafo de conexiones entre nodos en un sistema ROS.*

3.5.5 Módulo ROS para procesamiento de imagen

Para realizar el procesamiento de imágenes, ROS proporciona varios nodos con una funcionalidad bastante completa, el módulo image_pipeline. Este paquete se encarga del procesamiento de videos capturados a partir de una cámara tanto para visión monocular como para visión estereoscópica. A su vez, el stack (conjunto de paquetes integrados bajo una estructura específica) image_pipeline contiene 6 paquetes, como se puede ver en la imagen siguiente.

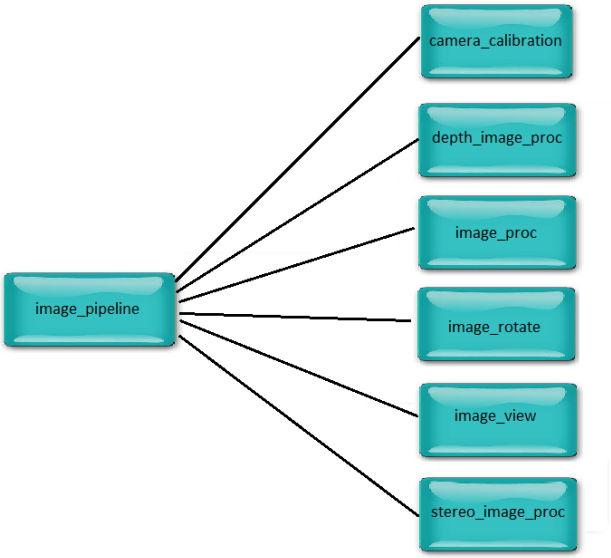

Figura 22: *Subdivisión del stack image_pipeline en paquetes.*

- camera_calibration: Las cámaras deben calibrarse para poder establecer una relación entre las imágenes que reproducen y el mundo real 3D. Con la calibración se ajustan varios parámetros necesarios para que la cámara funcione correctamente. El nodo camera_calibration proporciona herramientas para calibrar tanto cámaras monoculares como cámaras estereoscópicas en nuestro sistema ROS

- depth_image_proc: Este paquete contiene nodelets que se encargan del procesado de imágenes captadas con cámaras con sensor de profundidad, como las producidas por la cámara OpenNI o Kinect. Su funcionalidad va desde crear imágenes de disparidad

hasta la creación de la nube de puntos de la escena 3D que la cámara esté captando.

- image_proc: Este paquete contiene el nodo image_proc, que actúa como puente entre el driver de la cámara y los nodos de procesamiento de imagen. El nodo image_proc también procesa la imagen para eliminar la distorsión de la cámara y, si es necesario, convierte el formato de imagen Bayer o YUV422 a color. Este nodo funciona para cámaras monoculares.

- image_rotate: Este paquete contiene un nodo que permite rotar la imagen de forma que minimiza en ángulo entre un vector de un frame arbitrario al vector del frame de la cámara. El frame resultante es publicado por un nodo. El nodo image_rotate permite visualizar las imágenes obtenidas por la cámara con una orientación que es más intuitiva para el ser humano que la orientación de la imagen que el hardware de la cámara establece.

- image_view: Este paquete proporciona nodos para visualizar imágenes en ROS Permite visualizar tanto imágenes con cámaras monoculares como imágenes con cámaras estéreo. También permite mostrar el mapa de disparidad de una imagen.

- stereo_image_proc: Este paquete contiene el nodo stereo_image_proc, que actúa como puente entre el driver de la cámara y los nodos de procesamiento de imagen. Este nodo realiza todas las funciones que realiza el nodo image_proc, pero para las dos lentes de la cámara estereoscópica. Para cámaras que estén correctamente calibradas, se tienen imágenes sin distorsión y rectificadas, lo que permite un procesamiento estéreo más exacto. El nodo stereo_image_proc también es capaz de computar imágenes de disparidad de un par estéreo usando algoritmos de OpenCV. Este nodo también produce nubes de puntos que pueden verse con rviz y procesarse con Point Cloud Library (en adelante PCL).

3.5.6 El paquete Rosbag

ROS proporciona un paquete llamado rosbag, que permite entre otras cosas, guardar y visualizar topics activos del roscore. Entre la funcionalidad que rosbag ofrece tenemos:

- Grabar en un archivo '.bag' la ejecución de todos los nodos activos del *roscore*. Para ello se tiene que ejecutar el comando:

  >> rosbag record –a

- Ejecutar un archivo '.bag' para reproducir la ejecución de los nodos del *roscore*. Para ello se tiene que ejecutar el comando:

  >> rosbag play <nombre_archivo>

- Mostrar información sobre un archivo '.bag'. para ello se tiene que ejecutar el comando:

  >> rosbag info <nombre_archivo>

- Comprobar si un archivo '.bag' es ejecutable o si por el contrario está dañado o tiene algún fallo. Para ello se tiene que ejecutar el comando:

  >> rosbag check <nombre_archivo>

- Reparar un archivo '.bag' dañado. Para ello se tiene que ejecutar el comando:

  >> rosbag fix <nombre_archivo_dañado> <nombre_archivo_salida>

- Reparar archivos '.bag' que hayan sido incorrectamente cerrados o que tengan problemas con la indexación. Para ello se tiene que ejecutar el comando:

  >> rosbag reindex <nombre_archivo>

En este caso, la mayor utilidad del paquete rosbag será la posibilidad de almacenar la ejecución del nodo stereo_node, ya que eso permitirá disponer del video capturado por una cámara

estereoscópica para luego poder procesarlo y obtener la imagen de disparidad.

Para poder visualizar con ROS el mapa de disparidad obtenido a través de una cámara estereoscópica cuya ejecución ha sido almacenada en un archivo '.bag', los nodos que se deben ejecutar son:

- *stereo_view*: Nodo para visualizar el mapa de disparidad y el video grabado con ambas lentes de la cámara estereoscópica.

- *stereo_image_proc*: Nodo para procesar la imagen estereoscópica y crear el mapa de disparidad.

- *play*: Nodo para reproducir el video desde un archivo .bag. El archivo .bag contiene además la información de todos los nodos de ROS que estuvieron activos durante la grabación del video.

En la siguiente imagen se puede ver cómo quedaría el grafo de conexiones durante la ejecución de los nodos que se han mencionado para procesar el video obtenido desde el archivo .bag.

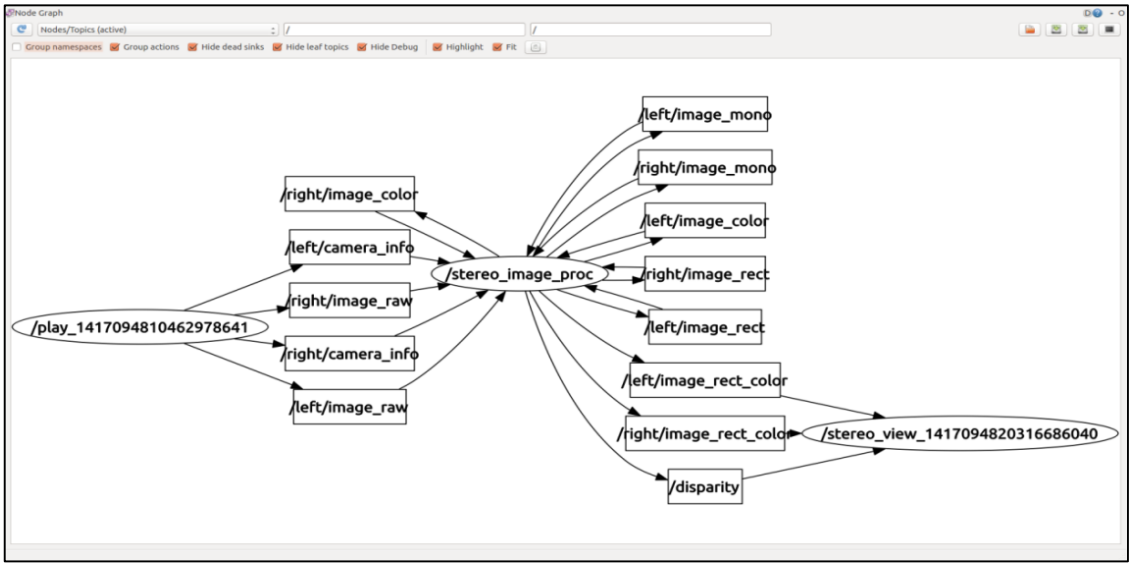

Figura 23: *Grafo de conexiones para la visualización de la imagen de disparidad a partir de un archivo bag.*

3.5.7 Conexión con las cámaras

Para la conexión de una cámara, ya sea monocular o estereoscópica, ROS implementa el paquete "uvc_camera", que proporciona los drivers necesarios para utilizar cámaras conectadas mediante el puerto Universal Serial Bus (en adelante USB).

Para capturar la imagen que proporciona una cámara monocular se ejecutará el nodo "camera_node". La ejecución de dicho nodo no mostrará la imagen de la cámara, para ello se tendrá que utilizar el nodo *"image_view"* que, como se comentó anteriormente, es el nodo encargado de mostrar la imagen por pantalla.

Los parámetros que se pueden modificar a la hora de ejecutar el nodo "camera_node" son los siguientes:

~camera_info_url: Dirección Uniform Resource Locator [URL] al archivo de calibración de la cámara.

~device (por defecto: /dev/video0): Ruta del archivo de la cámara.

~fps (por defecto: 10): Frames por segundo que procesa la cámara.

~width (por defecto: 640): Anchura de la imagen de la cámara.

~height (por defecto: 480): Altura de la imagen de la cámara.

Si, por el contrario, lo que se quiere es capturar la imagen que proporciona una cámara estereoscópica, se tendrá que ejecutar el nodo "stereo_node". Al igual que se ha comentado anteriormente para el nodo "camera_node", la ejecución del "stereo_node" no mostrará la imagen de la cámara, para ello se tendrá que utilizar el nodo "image_view".

Para procesar una imagen estereoscópica ROS selecciona por defecto la cámara 'dev/video0' como cámara izquierda y '/dev/video1' como cámara derecha. Por tanto, no se necesita disponer propiamente de una cámara estereoscópica para utilizar visión estéreo, sino que se puede simular la imagen estéreo usando dos cámaras conectadas al puerto USB

Los parámetros que se pueden modificar a la hora de ejecutar el nodo "stereo_node" son los siguientes:

~left/camera_info_url: Dirección URL al archivo de calibración de la cámara izquierda.

~right/camera_info_url: Dirección URL al archivo de calibración de la cámara derecha.

~left/device (por defecto: /dev/video0): Ruta del archivo de la cámara izquierda.

~right/device (por defecto: /dev/video1): Ruta del archivo de la cámara derecha.

~fps (por defecto: 10): Frames por segundo que procesa la cámara.

~left/rotate (por defecto: false): Si el valor es true, la imagen izquierda de la cámara se rotará 180º grados.

~right/rotate (por defecto: false): Si el valor es true, la imagen derecha de la cámara se rotará 180º grados.

~width (por defecto: 640): Anchura de la imagen de la cámara.

~height (por defecto: 480): Altura de la imagen de la cámara.

3.5.8 Calibración de las cámaras

Antes de aplicar el procesamiento a las imágenes capturadas, es necesario realizar una calibración de la(s) cámara(s) a través de transformaciones que permitan pasar los puntos de la escena al mismo plano de referencia de la cámara.

El paquete camera_calibration de ROS utiliza la librería de calibración proporcionada por OpenCV para la calibración de cámaras tanto monoculares como estéreo. Para ello, se utiliza el nodo Python cameracalibrator.py proporcionándole como parámetros el tamaño del tablero de ajedrez en celdas, la longitud del lado de la celda del tablero y la imagen o imágenes en raw (del inglés en "bruto" o en "crudo") proporcionadas.

De esta forma, si por ejemplo se quisiera calibrar una cámara utilizando un tablero de ajedrez de 8x6 y una longitud de celda de 24 milímetros se podría lanzar una orden como la siguiente:

```
rosrun camera_calibration cameracalibrator.py --size 8x6 --square
0.024 image:=/my_camera/image camera:=/my_camera
```

Al ejecutar la orden anterior se mostrará una aplicación para poder realizar el calibrado de la cámara utilizando el tablero de ajedrez con las medidas indicadas anteriormente.

En la siguiente imagen se puede observar un ejemplo de calibración de dos cámaras en modo estéreo haciendo uso de la aplicación anterior:

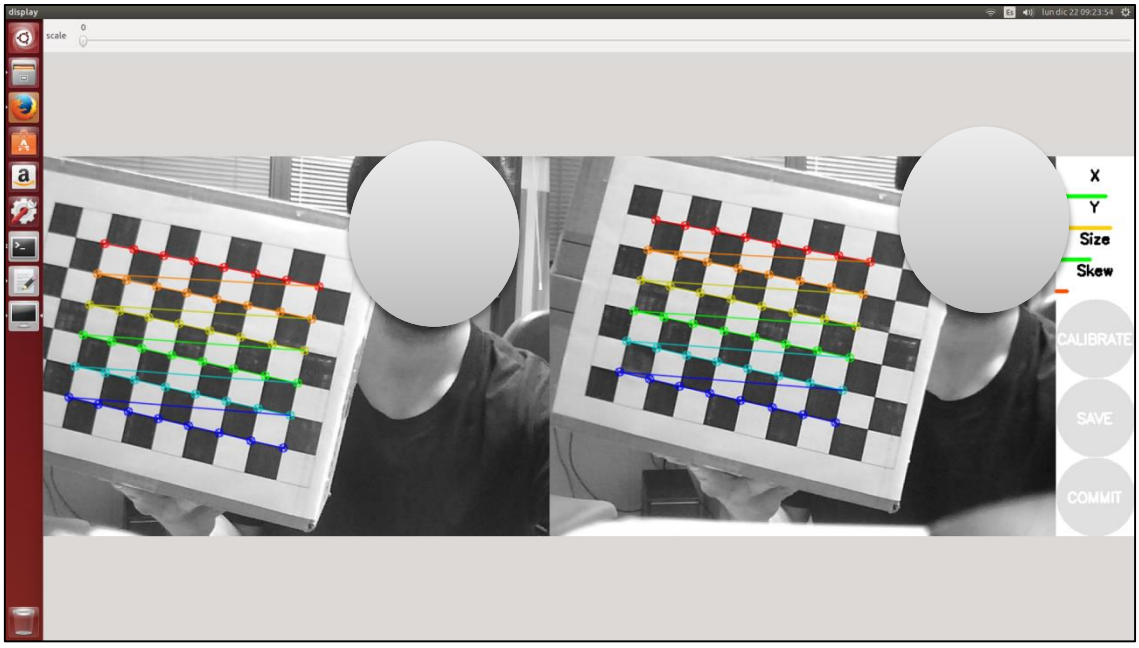

Figura 24: *Ejemplo de calibración de dos cámaras en modo estéreo con el nodo camera calibrator.*

Si se observa la parte derecha de la captura anterior, aparecen cuatro parámetros que llevan asociadas unas barras de color que se van rellenando y cambiando de rojo a verde, conforme vamos capturando frames de calibración con el tablero de ajedrez. Estos parámetros (X, Y, Size y Skew), indican respectivamente transformaciones en el posicionamiento en el eje X, en el eje Y, transformaciones de tamaño (Size) y oblicuidad (Skew). Una vez todas las barras estén en verde se habrá obtenido un número de frames válidos para la calibración,

por lo que se pulsará sobre el botón CALIBRATE. Finalmente, y una vez calibrada, se pulsará en SAVE para guardar el/los fichero/s de calibración.

Estos ficheros deben ser parseados y copiados al directorio .ros en la ruta /home del sistema dentro a su vez de un subdirectorio que deberá llamarse camera_info. En los experimentos se ha utilizado un parseador realizado en Perl que adapta los ficheros de calibración al formato necesario y los copia directamente en la ruta especificada.

Realizando un nuevo encendido y conexión con la(s) cámara(s) a través del paquete uvc_camera, se cargarán los nuevos ficheros de calibración, y por tanto, a partir de ese momento la cámara quedará calibrada.

3.5.9 Visualización del Mapa de Disparidad

Para poder visualizar, con ROS, el mapa de disparidad obtenido a través de una cámara estereoscópica que se encuentra grabando video en tiempo real, los nodos que se deben ejecutar son los siguientes:

- stereo_view: Nodo para visualizar el mapa de disparidad y el video grabado con ambas lentes de la cámara estereoscópica.

- stereo_image_proc: Nodo para procesar la imagen estereoscópica y crear el mapa de disparidad.

- uvc_camera_stereo: Nodo para ejecutar una cámara conectada a través del puerto USB

En la siguiente imagen se puede ver cómo quedaría el grafo de conexiones durante la ejecución de los nodos que se han mencionado para procesar el video obtenido desde una cámara conectada por USB.

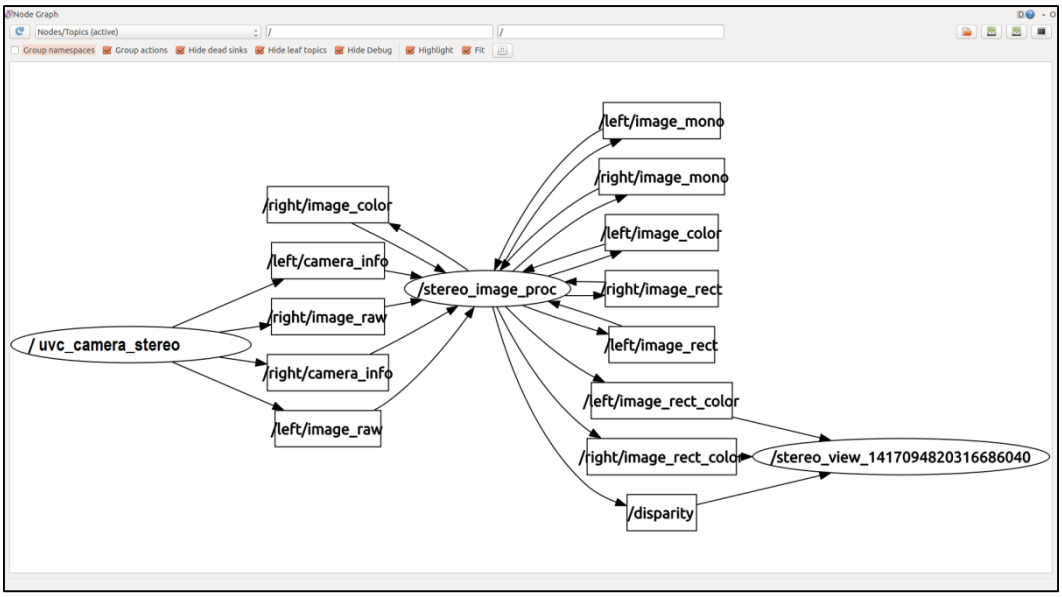

Figura 25: *Grafo de conexiones para la visualización de la imagen de disparidad a partir de una cámara estéreo calibrada.*

3.5.10 Visualización con Rviz

ROS también proporciona un visor gráfico llamado Rviz, que permite visualizar una malla con el eje de coordenadas, el mapa 3D o la nube de puntos obtenida del mapa de disparidad, entre otras cosas.

Rviz se ejecuta como un nodo más de ROS En la siguiente imagen se puede ver un ejemplo de la pantalla de visualización de Rviz con la nube de puntos cargada.

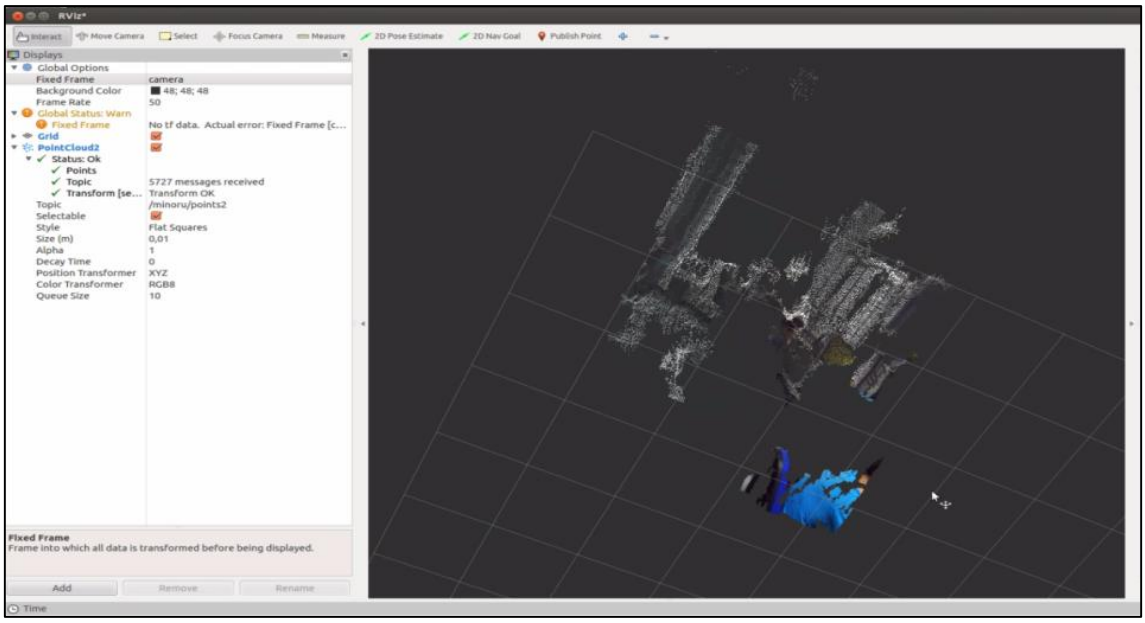

Figura 26: *Representación de una escena 3D en Rviz a partir de una nube de puntos extraída de la disparidad calculada utilizando la Minoru 3D Webcam.*

Como se puede apreciar, gracias a la nube de puntos generada por el nodo stereo_image_proc se puede ver la profundidad de la escena.

Rviz también tiene un componente llamado Marker Array que permite visualizar gráficamente las librerías OctoMap [40]. Las librerías OctoMap utilizan la nube de puntos para convertir los puntos de la escena en voxels (unidades cúbicas básicas), y permitir así una visualización más clara de la profundidad de la escena 3D.

En la siguiente imagen se puede ver un ejemplo de la visualización de OctoMap con Rviz:

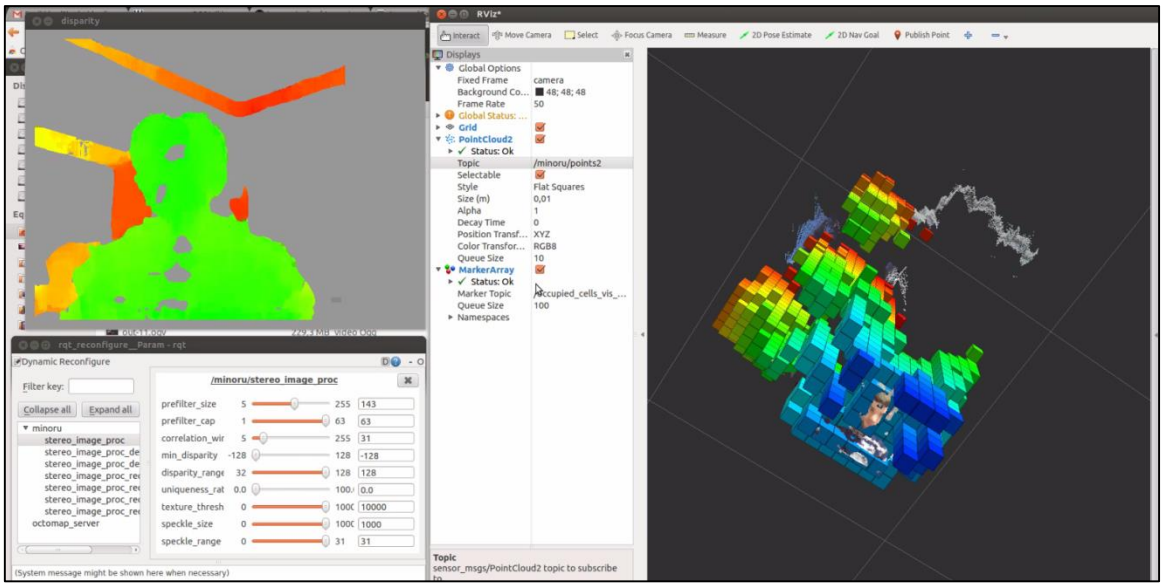

Figura 27: *Superposición de un Marker Array con OctoMap sobre la nube de puntos.*

3.6 Obtención y visualización del mapa de disparidad y cálculo de profundidad

3.6.1 Solución con cámara Minoru 3D Webcam

A continuación se va a explicar cómo realizar el cálculo del mapa de disparidad y las operaciones necesarias para obtener la profundidad a partir de la disparidad. Para este apartado se explicará el proceso utilizando una cámara estéreo Rolling Shutter, concretamente se utilizará la webcam Minoru anteriormente mencionada.

3.6.2 Obtención de la imagen de disparidad

En los apartados anteriores se ha hablado sobre el paquete image_pipeline y sobre la funcionalidad de los nodos que éste contiene. A continuación se va a hablar de cómo se obtiene la imagen de disparidad y cómo se puede detectar profundidades en la imagen.

Anteriormente ya se vio como ejecutar los nodos correspondientes del paquete image_pipeline para visualizar la imagen estéreo y la imagen de disparidad. El siguiente paso es configurar la disparidad para que se puedan obtener buenos resultados. Para ello se utilizará

el nodo rqt_reconfigure. En la siguiente imagen se puede ver la ventana gráfica de rqt_reconfigure.

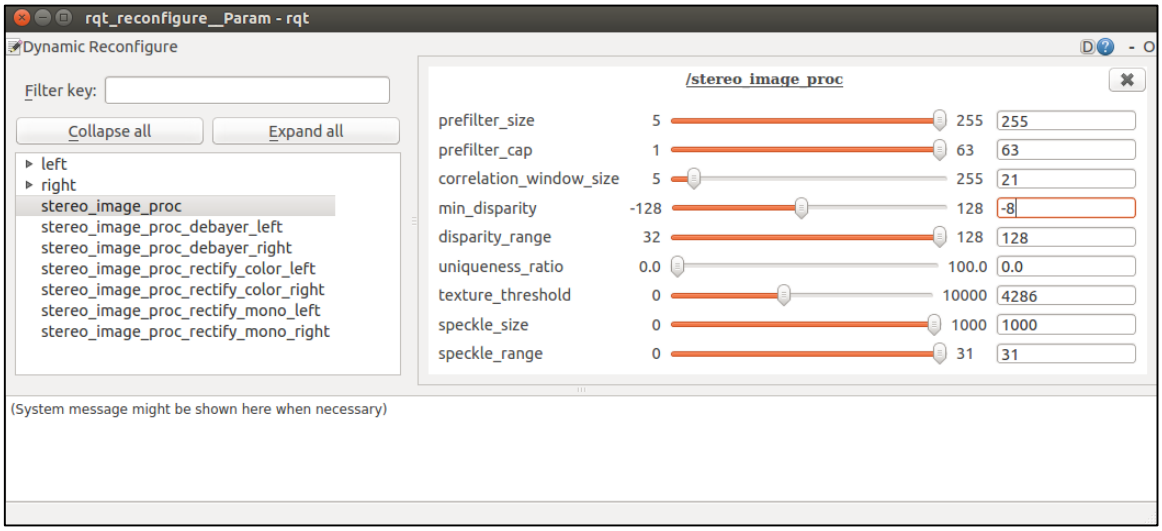

Figura 28: *Parámetros de correspondencia estéreo para configurar la imagen de disparidad.*

Los parámetros más importantes son:

- **Correlación**: la correlación indica la fuerza y la dirección de una relación lineal y proporcionalidad entre dos variables estadísticas. Se considera que dos variables cuantitativas están correlacionadas cuando los valores de una de ellas varían sistemáticamente con respecto a los valores homónimos de la otra: si se tienen dos variables (A y B) existe correlación si al aumentar los valores de A lo hacen también los de B y viceversa. La correlación entre dos variables no implica, por sí misma, ninguna relación de causalidad.

- **Disparidad Mínima**: Este parámetro indica la proporcionalidad que se tomará como referencia para calcular la disparidad.

- **Rango de Disparidad**: Este parámetro definirá la proporción de distancia entre objetos de una escena 3D.

- **Umbral de Textura**: Este parámetro definirá el tamaño mínimo que debe tener un objeto para ser tenido en cuenta en el mapa de disparidad. Esto es importante para evitar el ruido que puede producirse en la disparidad.

- **Tamaño/Rango del Moteado**: Este parámetro definirá el nivel de filtrado de los puntos de disparidad.

Una vez configurados todos los parámetros de la imagen de disparidad, se obtendrán resultados óptimos como el ejemplo que se puede ver en la figura 29.

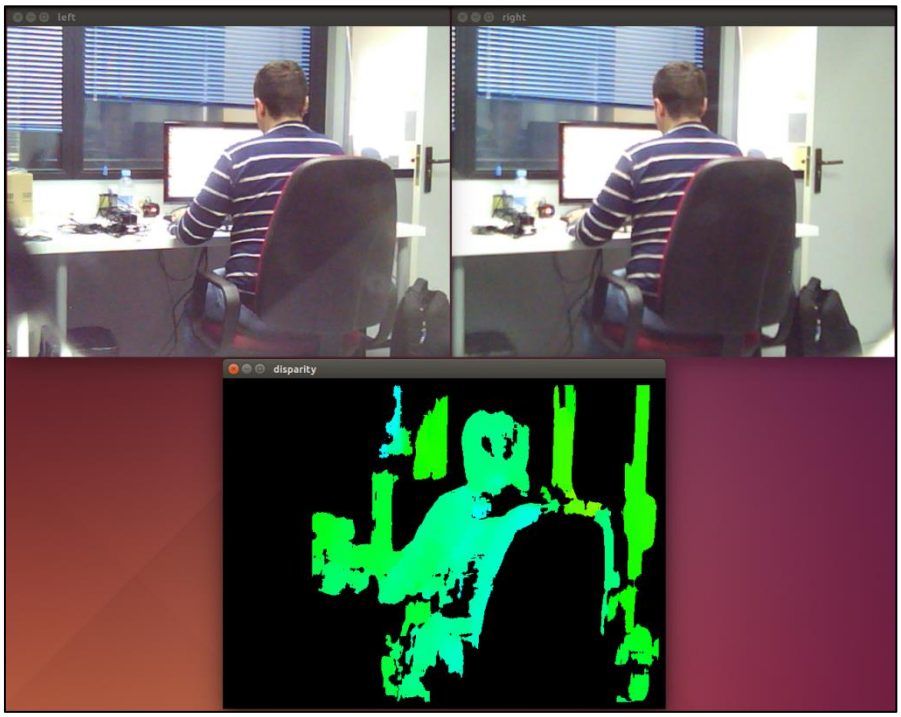

Figura 29: *Par de imágenes estéreo y su imagen de disparidad corregida asociada.*

Como se puede observar, los colores definen la distancia a la que se encuentran los objetos de la escena 3D. La escala de colores va desde el azul turquesa (más cercano) al amarillo (más lejano).

Para realizar cambios en el flujo de información que se recibe de la imagen de disparidad, se debe modificar el fichero *stereo_view.cpp*, que se encuentra del paquete *image_view*. Dicho fichero se encarga de mostrar la visión estéreo y lanzar la imagen de disparidad para su visualización.

3.6.3 Calculo de Distancia

Para calcular la profundidad de la imagen, primero se tendrá que filtrar qué objetos de la escena se deben procesar y cuáles no. Para ello se seguirán los siguientes pasos:

- Cálculo de la disparidad.
- Cálculo de contornos para seleccionar los objetos de la escena.
- Cálculo del área de los objetos seleccionados para eliminar el ruido de la imagen.
- Cálculo del tono del mapa de color más cercano de cada objeto para fijar el objeto más cercano de la escena 3D.

  Una vez que se han realizado todos estos cálculos, se tendrá fijado el objeto que se encuentra más cerca de la cámara, que es en realidad el objeto de interés. Una vez que se dispone de toda esta información, se podrá calcular la distancia a la que se encuentra dicho objeto. Para ello se utilizará una técnica de interpolación lineal sobre el mapa de colores que define la distancia de la cámara al objeto en cuestión. Los pasos para calcular la profundidad son:

- Definir el color más cercano a la cámara. En este caso será el azul turquesa, con RGB (0, 249, 255).
- Definir el color más lejano a la cámara. En este caso será el amarillo, con RGB (253, 255, 0).
- Definir un punto de referencia a una distancia fijada.
- Utilizar la función de interpolación lineal para calcular la distancia para cada tono del mapa de colores.

$$f(x|x_1;x_2) = f(x_1) + \frac{f(x_2) - f(x_1)}{(x_2 - x_1)}(x - x_1) \qquad \textbf{Ecuación 3.A.22}$$

Una vez que se conoce la información que se necesita, se tiene que modificar el nodo stereo_view para añadirle dicha funcionalidad. Para ello se modificará el archivo stereo_view.cpp y se implementará la función de cálculo del objeto más cercano, para posteriormente realizar el cálculo de profundidad. De este modo, al procesar la imagen de disparidad, devolverá la distancia numérica a la que se encuentra dicho objeto.

En la siguiente imagen se puede ver la imagen estéreo, la imagen de disparidad, y un terminal en el que se imprime en cada frame el color RGB, el índice tono del color RGB dentro del mapa de colores y la distancia a la cámara del objeto más cercano.

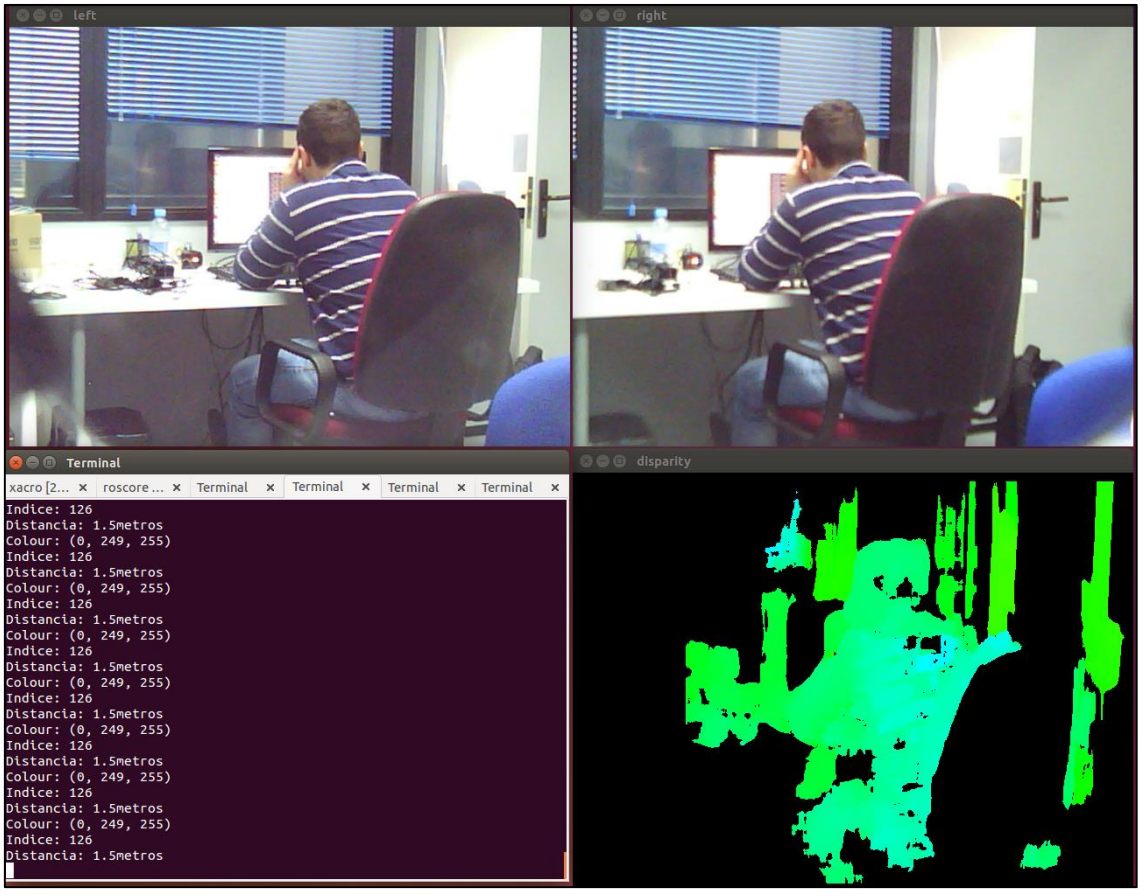

Figura 30: *Algoritmo modificado para el cálculo de distancias obtenidas del mapa de disparidad.*

3.6.4 Solución con el módulo estéreo de cámaras Imaging Development System IDS UI-1221LE

A continuación se va a explicar cómo realizar el cálculo del mapa de disparidad y las operaciones necesarias para obtener la distancia a partir de la disparidad. Para este apartado se explicará el proceso utilizando una cámara estéreo Global Shutter, concretamente se utilizarán las cámaras del fabricante Imaging Develpment System (en adelante IDS) conectadas en estéreo que se han mencionado anteriormente.

3.6.4.1 Obtención de la imagen de disparidad

Para el cálculo de la imagen de disparidad, el procedimiento es distinto a como se venía haciendo con la webcam Minoru. Las cámaras IDS usan lentes de ojo de pez (fisheye), que provocan distorsión de barril en la imagen, por lo que la forma de calibrar estas cámaras es distinta. Además, la forma de obtener y procesar las imágenes obtenidas con las cámaras es diferente.

El primer paso es obtener la imagen raw de las cámaras. Con una cámara USB convencional, se puede acceder directamente a la imagen de la cámara mediante el buffer de linux "/dev/video", pero con las IDS es diferente. Las cámaras IDS tienen un driver propio que el Sistema Operativo (en adelante SO) no reconoce automáticamente. La instalación de dichos drivers permitirá acceder a la funcionalidad de las cámaras.

Para interactuar con las cámaras, acceder a sus parámetros de configuración y obtener las imágenes, el driver de las cámaras IDS instala una librería en el sistema llamada "ueye.h". Esta librería contiene una gran variedad de funciones, pero las que más se han usado son:

- Funciones de inicialización de las cámaras: Permiten montar la cámara en un dispositivo virtual y definir los parámetros iniciales de configuración como resolución de la imagen, modo de color, modo de video, etc.

- Funciones de configuración en tiempo de ejecución: Permiten cambiar parámetros de la cámara en tiempo de ejecución, como

los frames por segundo, el tiempo de exposición, la ganancia, el corrector gamma, etc.

- Funciones para acceso a las imágenes: Permiten definir el modo en que se almacenarán las imágenes en memoria y, más tarde, el modo en que podremos leer dichas imágenes de memoria para procesarlas y utilizarlas como convenga.

Para poder realizar el primer paso de forma automática, se ha creado una clase de C++, UeyeCam, que se encargará de acceder a la librería de ueye.h y permiterá acceder a las cámaras más fácilmente.

El segundo paso es calibrar las cámaras. A diferencia de una cámara normal con lente de tipo pinhole, las lentes fisheye provocan distorsión de barril en la imagen. La distorsión de barril consiste en la curvatura hacia el exterior de aquellas líneas que deberían aparecer rectas. Esta es causada por lentes con ángulos de visión superiores a 90º, ya que la lente intenta plasmar objetos de enorme amplitud en un plano de imagen finito.

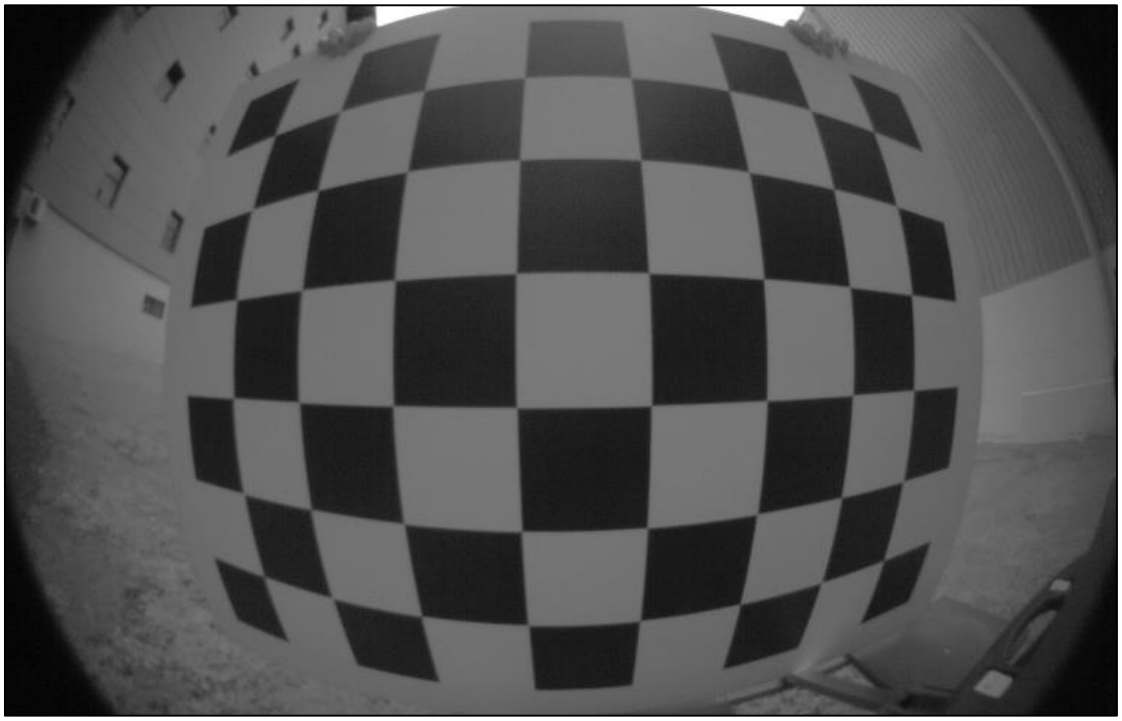

Figura 31: *Ejemplo de distorsión de barril.*

Debido a esto, antes de calibrar las cámaras en modo estéreo, hay que calibrarlas en modo monocular para eliminar la distorsión de barril. Con esta calibración monocular se realizarán diferentes

transformaciones geométricas para rectificar la imagen (tal y como se indica en [41]). El resultado obtenido tendrá los bordes de la imagen aún con distorsión, por lo que se debe realizar un recorte en la fotografía. Esto conllevará una perdida en el ángulo de visión, que será proporcionalmente inferior al campo de visión anterior.

En la siguiente imagen se puede apreciar cómo, al rectificar la imagen, los bordes aparecen un poco distorsionados. Realizando luego un recorte de bordes se solucionará este problema.

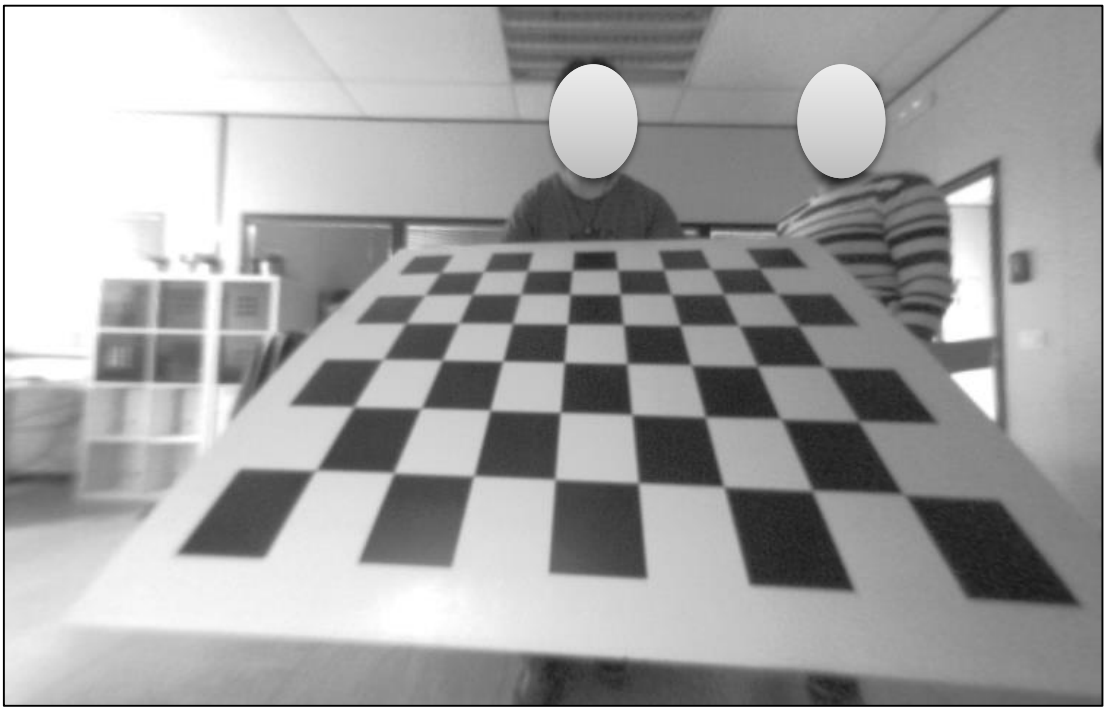

Figura 32: *Distorsión de barril rectificada.*

Una vez se ha eliminado la distorsión, se pueden calibrar las cámaras en estéreo como ya se ha explicado anteriormente.

Después de calibrar las cámaras en modo estéreo, se pasa al cálculo de la disparidad. Para ello, se han realizado pruebas tanto con el método SGBM y el método BM que ya se comentó en secciones anteriores.

En la siguiente imagen se pueden ver los parámetros que se utilizan para el cálculo de la disparidad, tanto para el algoritmo SGBM como para el BM

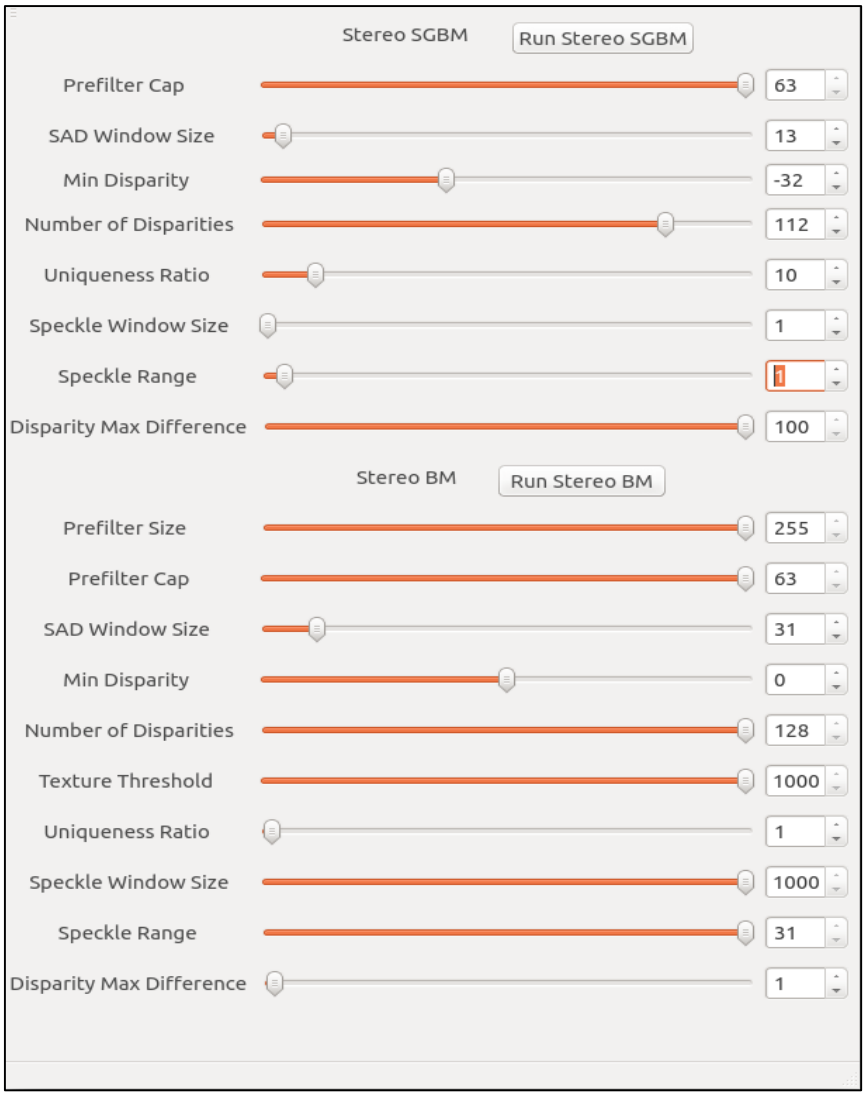

Figura 33: *Parámetros para el cálculo de disparidad con los algoritmos Stereo SGBM y Stereo BM*

Los parámetros más importantes son:

- **Correlación**: la correlación indica la fuerza y la dirección de una relación lineal y proporcionalidad entre dos variables estadísticas. Se considera que dos variables cuantitativas están correlacionadas cuando los valores de una de ellas varían sistemáticamente con respecto a los valores homónimos de la otra: si se tienen dos variables (A y B) existe correlación si al aumentar los valores de A lo hacen también los de B y viceversa. La correlación entre dos variables no implica, por sí misma, ninguna relación de causalidad.

- **Disparidad Mínima**: Este parámetro nos indica la proporcionalidad que se tomará como referencia para calcular la disparidad.

- **Rango de Disparidad**: Este parámetro definirá la proporción de distancia entre objetos de una escena 3D.

- **Umbral de Textura**: Este parámetro definirá el tamaño mínimo que debe tener un objeto para ser tenido en cuenta en el mapa de disparidad. Esto es importante para evitar el ruido que puede producirse en la disparidad.

- **Tamaño/Rango del Moteado**: Este parámetro definirá el nivel de filtrado de los puntos de disparidad.

Una vez se ajustan los parámetros, se puede calcular la disparidad. En este caso se emplea la ayuda de la librería OpenSource OpenCV. El resultado se puede ver en la siguiente imagen:

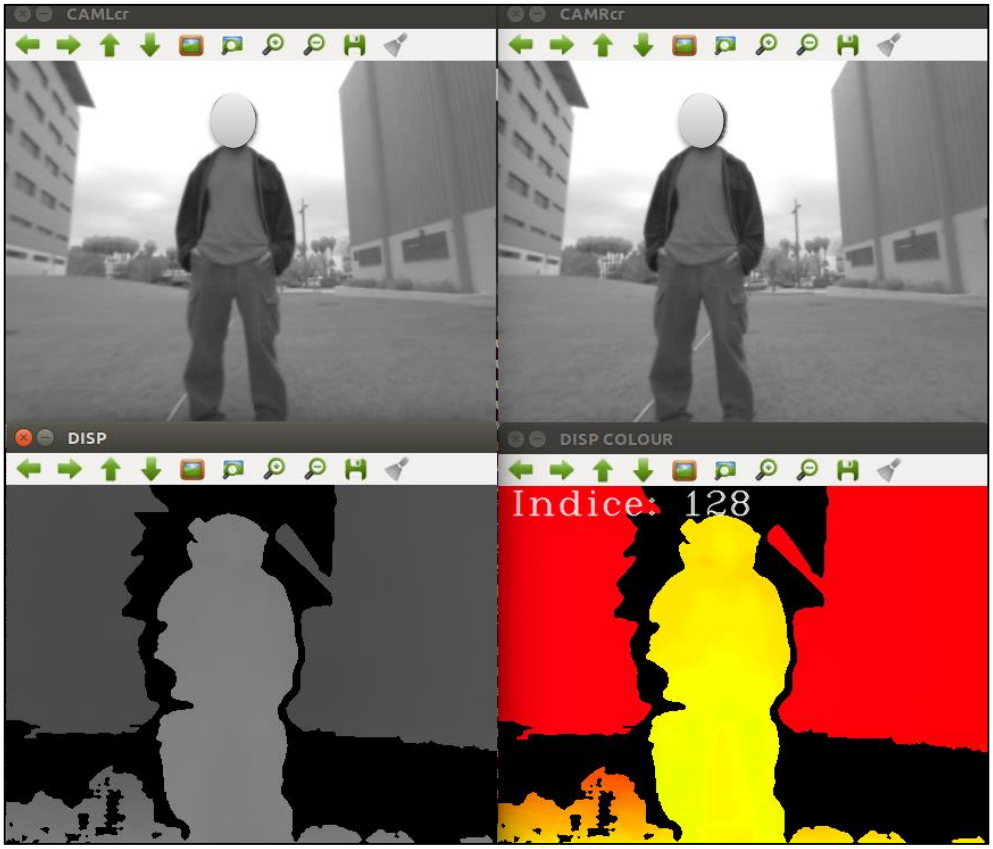

Figura 34: *Mapa de Disparidad obtenido con módulo estéreo IDS UI-1221LE aplicando los algoritmos SGBM y BM*

3.6.4.2 Calculo de Distancia

Para poder realizar un cálculo de distancias a partir de la imagen de disparidad, se necesita que ésta sea fiable. Para ello no basta con simplemente calcular la disparidad, hay que procesar dicha imagen. Para ello se tiene que realizar un filtrado de Speckles [42], que consiste en eliminar las pequeñas motas que aparecen en la imagen y que pueden provocar falsos positivos a la hora de calcular distancias.

Una vez se ha filtrado la imagen de disparidad se procede a realizar un cálculo de contornos. Con ello se fraccionará la imagen de disparidad en contornos rectangulares que se corresponderán aproximadamente con objetos de la escena. De este modo se localizarán los objetos potenciales.

Para finalizar, calculamos el índice de color de cada objeto potencial y se escogerá el menor. Los índices de disparidad van de menor a mayor, por lo que, al escoger el índice menor, se estará calculando el objeto más cercano al módulo estéreo.

Para calcular la distancia que se corresponde a cada índice se utilizará una función de interpolación lineal. De ese modo se obtendrán distancias que tendrán su correspondencia con cada uno de los 255 índices de disparidad.

En la figura 35 se puede ver un ejemplo del cálculo de la distancia al objeto más cercano con el módulo estéreo IDS UI-1221LE:

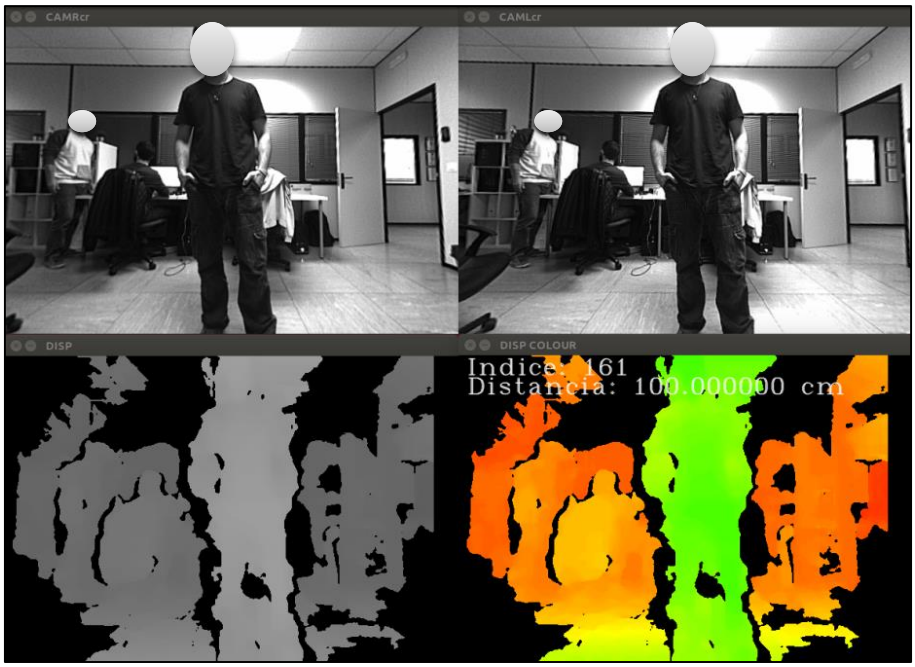

Figura 35: *Cálculo de la distancia al objeto más cercano con módulo estéreo IDS UI-1221LE.*

3.7 Solución al cálculo de distancias con módulo estéreo IDS UI-1221LE integrada con ROS

En la solución planteada en la sección anterior para el cálculo de distancias con el módulo estéreo de dos cámaras IDS UI-1221LE se ha realizado una implementación que hace uso de los algoritmos basados en BM de OpenCV. Sin embargo, dicha solución no tiene integración con ROS, pues se ha dedicado exclusivamente al cálculo de disparidad y al cálculo de la distancia al objeto más cercano.

Para dotar de escalabilidad al sistema, y conectar los diferentes nodos y la distinta información que se genera a partir de las técnicas de visión utilizadas, se ha desarrollado una solución adaptativa del paquete uEye de ROS (disponible con licencia BSD) que ofrece funcionalidad básica para utilizar las cámaras IDS UI-1221 LE dentro del sistema de nodos de ROS

El paquete uEye dispone del nodo camera, que es el nodo principal que incluye el driver de la cámara y que dispone de los diferentes parámetros de la cámara tales como el tiempo de exposición, frame rate, etc., y que pueden ser modificados utilizando el nodo rqt_reconfigure. Además, el paquete uEye dispone de otro nodo llamado stereo que está implementado para sincronizar dos cámaras

IDS en modo estéreo, publicando los respectivos mensajes de imagen y de calibración (camera_info) de ambas cámaras en los frames left y right.

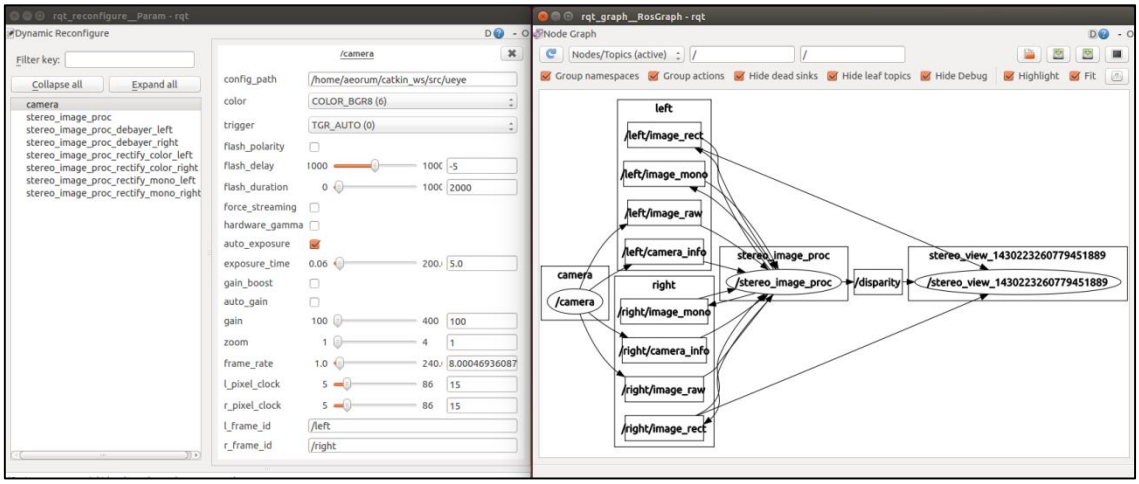

Figura 36: *Modificación de parámetros del módulo estéreo (izquierda) con rqt_reconfigure y grafo de interconexiones entre nodos (derecha) haciendo uso del paquete uEye.*

El nodo *stereo* del paquete *uEye* define una ruta específica (parámetro *config_path*) donde se deben depositar los ficheros de calibración de las cámaras izquierda y derecha con un nombre específico al modelo de las cámaras IDS En este caso los nombres de los ficheros serían *Cal-UI121LE-M-1-4102805686* y *Cal-UI121LE-M-1-4102805688,* para las cámaras izquierda y derecha respectivamente. Estos nombres están definidos en base al siguiente formato:

Cal-[modelo de la cámara]-[Color(C) o Monocromo(M)]-[Zoom]-[Id Cámara]

El valor de zoom, dependerá del que se haya asignado en los parámetros al iniciar el nodo. Si la cámara no dispone de zoom (como es nuestro caso), un valor mayor que 1 provocará un re-escalado de la imagen a una menor resolución, por lo que es conveniente dejar el valor por defecto (_zoom:=1). El modelo es el mismo en las dos cámaras, pero para poder diferenciarlas existe un número de identificación (Id) para cada una de ellas. En nuestro caso el terminado en 86 corresponderá a la cámara izquierda y el acabado en 88 a la cámara derecha.

Estos ficheros de calibración serán generados con el calibrador estéreo proporcionado por el paquete de ROS camera_calibration. En este caso, al disponer un valor de baseline (línea de separación entre

las cámaras) mayor que en el caso de la cámara Minoru 3D Webcam, se utiliza un patrón donde el tamaño de las celdas del tablero sea de 108 mm (0.108 cm).

No obstante, esta calibración estéreo no deberá realizarse directamente sobre las imágenes publicadas por el nodo stereo. Del mismo modo que se ha explicado en apartados anteriores, las cámaras IDS UI-1221LE incorporan sendas lentes de ojo de pez (fisheye) para obtener un mayor ángulo de visión. Sin embargo, estas lentes producen una elevada distorsión de barril, la cual puede ser corregida. Por ello, previamente a la calibración en modo estéreo, se deben calibrar las cámaras en modo monocular para eliminar la distorsión de barril.

Lo anterior implica tener que realizar modificaciones en el código del paquete uEye para aplicar la rectificación monocular a cada uno de los mensajes publicados por el nodo (identificados por el topic image_raw) para cada una de las dos cámaras. Los nuevos mensajes generados serán mapeados al mismo nombre de forma que el proceso sea transparente. Es decir, el nodo seguirá publicando mensajes a través de topics con el nombre /<camara>/image_raw, pero estos mensajes serán los de las imágenes sin distorsión.

Así pues, y una vez aplicadas las modificaciones oportunas, se arranca el nodo stereo de uEye y se realiza la calibración estéreo sobre las imágenes raw sin distorsión de barril. A continuación, se utiliza el nodo stereo_image_proc para obtener las imágenes con rectificación estéreo y el mapa de disparidad:

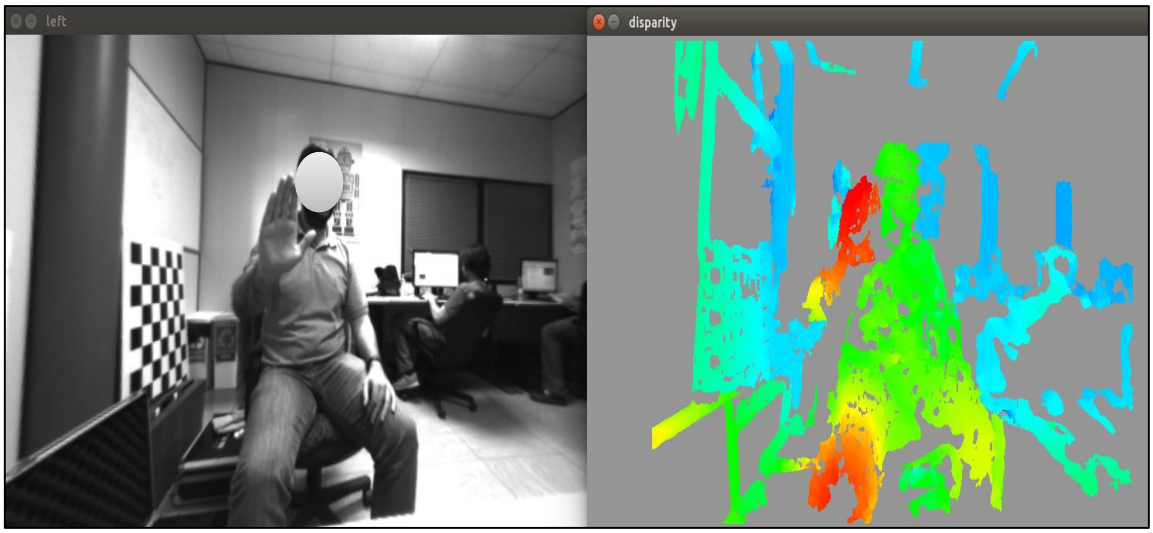

Figura 37: *Resultado de la imagen de disparidad a partir del módulo estéreo IDS UI-1221LE, utilizando el nodo stereo del paquete uEye y el nodo stereo_image_proc.*

Al igual que se hacía con la cámara Minoru 3D WebCam, se utiliza el nodo stereo_view del paquete image_view para visualizar la imagen de disparidad. Además, igualmente, se modifica también este nodo para aportar un procesamiento consistente en mostrar distintos rangos de distancias en función de la lejanía de los objetos.

Así pues, se divide la imagen de disparidad en tres imágenes con tres rangos de índices distintos (tonos rojos, verdes y azules). A continuación, se aplica el algoritmo de búsqueda de contornos, generando formas rectangulares sobre los diferentes objetos a diferentes distancias. Finalmente, sobre dichas formas y en función de la cercanía, se muestra un texto indicando la distancia de dicho objeto. Este proceso se realizará sobre la ventana que visualiza la imagen izquierda.

Por otro lado, en la ventana que visualiza la disparidad, se muestra un texto indicando el objeto más cercano (al igual que se hacía en las soluciones anteriores). No obstante, en esta solución, el texto se muestra en un color (rojo, verde y azul) en función de la distancia a la que se encuentre dicho objeto. En este caso, el color rojo representa mayor cercanía, el color verde cercanía media y el color azul menor cercanía.

En la siguiente imagen se puede ver el resultado obtenido:

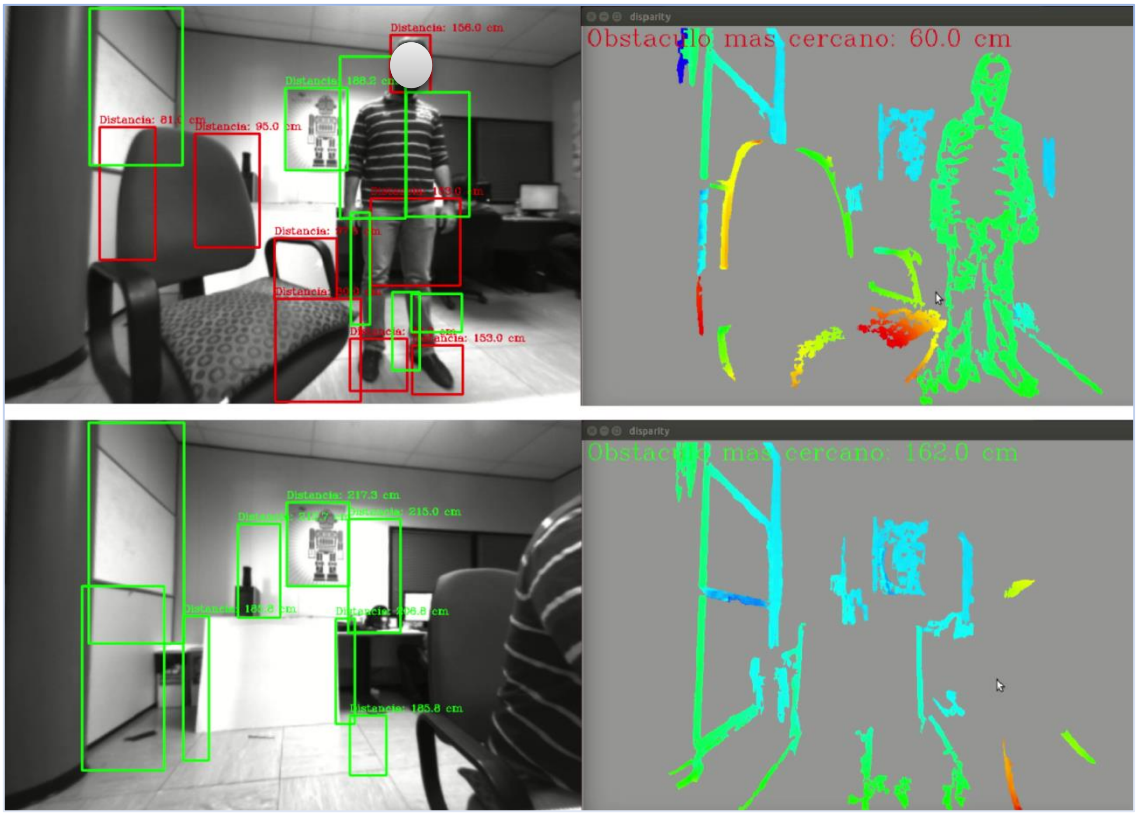

Figura 38: *Cálculo de la distancia al objeto más cercano con módulo estéreo IDS UI-1221LE integrado con ROS Se visualizan tres niveles de color (rojo, verde y azul) en función de la distancia.*

3.8 Detección y evasión mediante láser

Una posible solución a la detección de obstáculos se basa en el uso de un láser. En primer lugar se dispone del modelo *SF02* de *Parallax* Inc.. Las especificaciones técnicas más importantes son:

- Rango de medida: entre 0 y 40 metros.
- Resolución: 1 cm.
- Hasta 12 lecturas por segundo.
- No requiere calibración.

De las características listadas se deduce que la precisión que se obtendrá de la distancia al obstáculo será alta. Sin embargo, se tiene un único punto de detección, a diferencia de otros métodos como ultrasonidos o Light Detection and Ranging (en adelante LIDAR).

Posteriormente se testeó un nuevo láser LIDAR Lite. Las características principales del segundo láser son:

- Rango de medida: entre 0 y 40 metros.
- Precisión: +/- 0.025 m.
- Hasta 100 lecturas por segundo.
- Conexión por I^2C (Inter-Integrated Circuit) o Pulse Width Modulation [PWM].

El sistema se compone además de un sistema empotrado. Los que se han testeado han sido Raspberry Pi B, Odroid XU3 y Arduino.

3.8.1 Raspberry Pi B

Debido a su bajo coste y la disponibilidad de la misma, Raspberry Pi fue la primera opción. Una de sus funciones es abrir una conexión serie para leer los datos del láser. De forma concurrente, se mantiene una comunicación a través del puerto de entrada y salida analógico con el Pixhawk, usando como protocolo de comunicación Micro Air Vehicle Communication Protocol [MAVLink] [43]. El esquema más simplificado del sistema se muestra en la Figura 1.

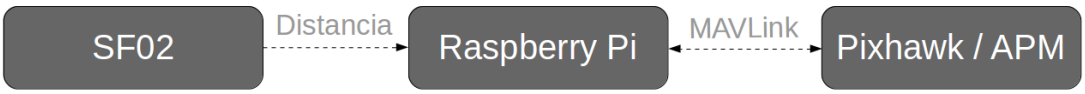

Figura 39: *Esquema simplificado del sistema de detección de obstáculos basado en Raspberry Pi B.*

Una vez detectado un obstáculo, se pueden implementar diversas técnicas de evasión. Para ello, se hará uso de los comandos y mensajes proporcionados por MAVLink. La lógica implementada para la evasión se aplicará en el modo de vuelo automático, teniendo en cuenta el protocolo de intercambio de waypoints de MAVLink. Además, se dispone de una palanca del mando radiocontrol para activar y desactivar la detección.

Figura 40: *Palanca para activar y desactivar la detección de obstáculos.*

A través del General Purpose Input/Output (en adelante GPIO) se conecta al receptor/transmisor del radiocontrol. Haciendo una lectura de la salida PWM se lleva a cabo la acción correspondiente: activar o desactivar la funcionalidad.

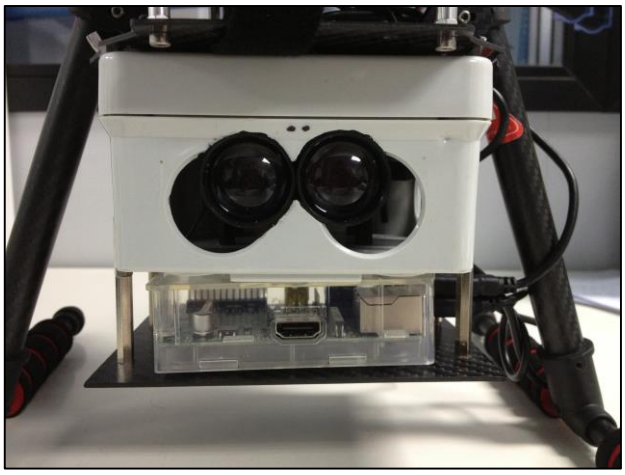

Figura 41: Imagen real del sistema con Raspberry Pi.

Adicionalmente se llevó a cabo comunicación de la telemetría a través de *XBEE*. XBEE es una solución integrada que proporciona un medio inalámbrico para la comunicación entre dispositivos. Los módulos XBEE utilizan el protocolo de red IEEE 802.15.4 para crear redes punto a multipunto o para redes punto a punto.

El resultado de la prueba fue la incapacidad del empotrado para alimentar el módulo XBEE a través de sus puertos USB Además, la alimentación del láser tampoco era suficiente, dando lugar a medidas erróneas y aparición de falsos positivos en detección. Por estos motivos se descartó su uso.

3.8.2 Odroid XU3

La arquitectura del sistema es esencialmente el mismo. Sin embargo,

las mayores prestaciones que ofrece este empotrado darán lugar a aumentar las prestaciones del diseño. La siguiente figura 42 un esquema simplificado:

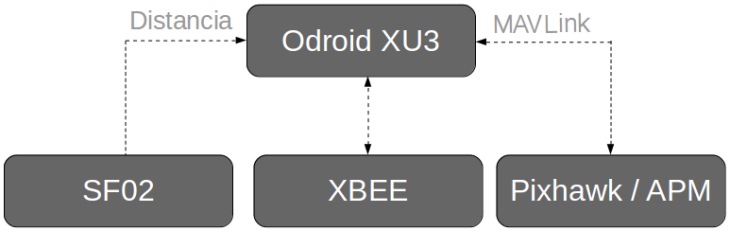

Figura 42: *Esquema simplificado del sistema de detección de obstáculos basado en Odroid XU3.*

En este caso, todas las conexiones se realizan a través de los puertos *USB*, que no tienen problemas para alimentar los dispositivos conectados. Además, no será posible usar la palanca de activación para la detección del mando, debido a la diferencia de voltaje de funcionamiento del puerto de entrada de Odroid y el nivel de la señal *PWM* del mando. Pero se usará una de las palancas que pueden comunicarse desde el mando a la controladora.

Figura 43: *Imagen real del sistema con Odroid XU3.*

Aunque el consumo de Odroid es mayor que el de Raspberry, se ha comprobado que no es un factor crítico del diseño. La lógica de evasión también será, en principio, basada en waypoints. Posteriormente, se realizará una lógica más sofisticada.

3.8.3 Arduino Mega 2560

Debido a la baja carga computacional requerida por la lógica de detección y evasión, además de la motivación del ahorro de espacio, llevó a portar el código realizado a Arduino. Las ventajas principales son la reducción de espacio, menor consumo y manejo de los

dispositivos por serial.

El esquema más simplificado del sistema con Arduino se muestra en la siguiente figura:

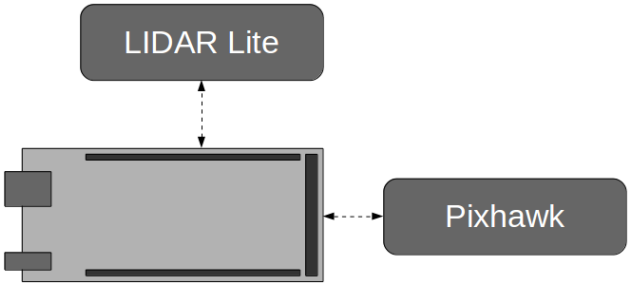

Figura 44: *Esquema simplificado del sistema de detección de obstáculos basado en Arduino.*

El sistema real una vez montado es el siguiente:

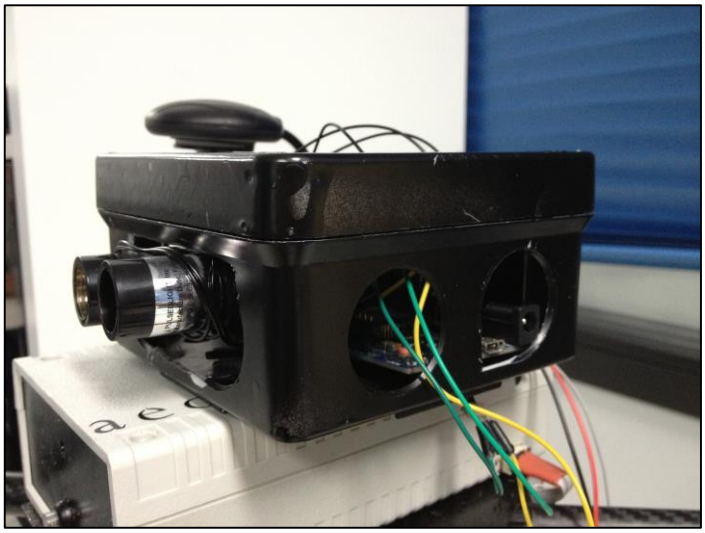

Figura 45: *Imagen real del sistema con Arduino.*

3.9 Lógica de evasión

Una vez detectado el obstáculo, hay que tomar decisiones sobre el comportamiento del dron. Las soluciones implementadas se pueden ordenar en dificultad creciente dependiendo de si se basan en actuar sobre el modo de vuelo actual o en la modificación de waypoints.

3.9.1 Modificación de la posición del dron

Como versión inicial para la detección y evasión de obstáculos se diseñó un sistema capaz de modificar uno de los ángulos del dron. En

este caso se modificó el yaw [44] de la posición en el momento de la detección en cuarenta y cinco grados sexagesimales.

Con ello se pretende detectar, girar el ángulo correspondiente y seguir adelante en la misión. Sin embargo, es una lógica muy limitada y poco dinámica. Por ello, se descarta y se toma como el punto de partida para nuevas lógicas de evasión y detección.

3.9.2 Evasión actuando sobre el modo de vuelo

La primera aproximación es la más directa: si se encuentra un obstáculo a una distancia previamente fijada, se detiene inmediatamente hasta que el obstáculo desaparezca. Para ello, se obliga al dron a salir de modo automático y entrar en modo loiter.

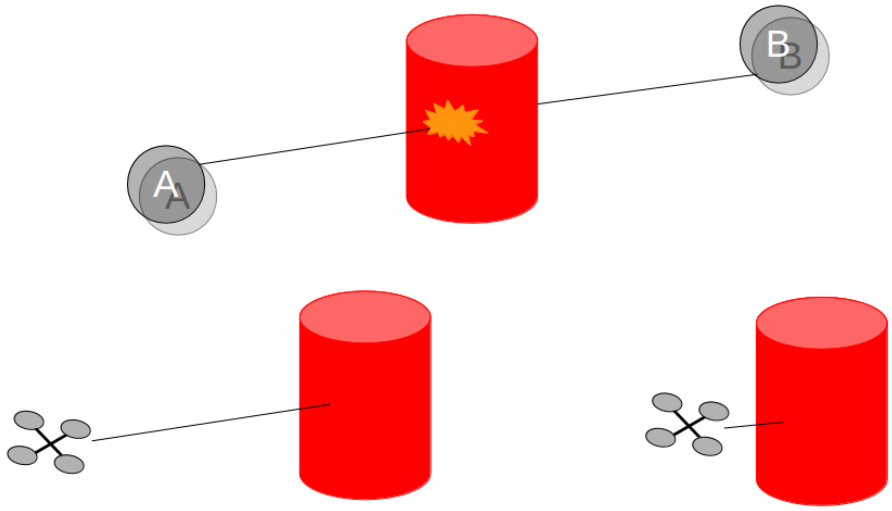

Figura 46: *Primera técnica de actuación ante un obstáculo.*

En la parte superior de la figura, se observa una ruta definida desde *A* hasta *B*, con un obstáculo intermedio. Una vez en modo automático, se pasará a modo loiter cuando se encuentre a una distancia previamente fijada.

Hay que tener en cuenta la velocidad de reacción del sistema y la distancia a la que se fija la detección. Por ejemplo, una configuración de dos metros por segundo con una distancia de diez metros de detección, se detendrá a unos cinco o seis metros del obstáculo.

Esta técnica de detección se testeó sobre la primera configuración del sistema: Raspberry Pi y el láser SF02 como sensor de distancia.

3.9.3 Evasión modificando la trayectoria

Una segunda estrategia consiste en modificar los waypoints de una ruta previamente establecida.

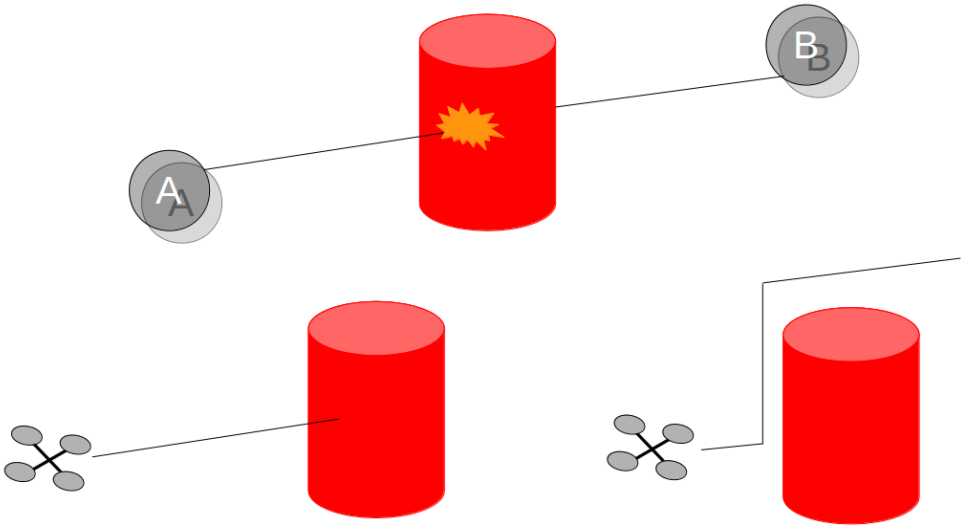

Figura 47: *Técnica de evasión basada en modificar los waypoints.*

La misma ruta vista en la figura 47 muestra un obstáculo entre los waypoints A y B. Sin embargo, la forma de actuar ahora es diferente. Entra en modo loiter, se crea un nuevo waypoint con más altura (fijada por el usuario previamente) y sube hacia los siguientes waypoints la misma altura. Cuando ha superado el obstáculo, sigue su ruta modificada.

Esta técnica de detección se testeó sobre la primera y segunda configuración del sistema: tanto Raspberry Pi y el láser SF02 como con Odroid y láser SF02.

3.9.4 Evasión basada en modo guiado

Sin embargo, la mejor forma de implementar la evasión se encontró en el uso de modo guiado. Se basa el mandar waypoints con la posición a la que se quiere ir [45]. El dron se desplazará automáticamente hasta ese punto ayudado por su Global Positioning System (en adelante GPS).

De esta forma, se puede obtener un comportamiento mucho más dinámico en cuanto a evasión del obstáculo. Se puede manipular fácilmente el dron para que se desplace en todas las direcciones del

espacio (arriba, abajo, derecha, izquierda, adelante y atrás).

3.10 Pruebas de detección y evasión

3.10.1 Raspberry Pi B y láser SF02

Las primeras pruebas se llevaron a cabo con la primera aproximación a la detección y evasión y la primera configuración del sistema (Raspberry Pi con láser SF02). Se probó tanto en interiores como exteriores.

En entornos interiores se obtuvieron resultados satisfactorios de forma inmediata. Sin embargo, en el exterior los inconvenientes estuvieron relacionados con el formato en que el láser SF02 transmite la medida por la conexión serie. El formato es $bx.xx\,m$, donde b es un número entre uno y cuatro o un espacio si la medida es menos de diez metros, x es un número entre cero y nueve y m es la unidad de medida (metros).

Para el caso de que la media sea mayor de los cuarenta metros permitidos, el sensor nos devolverá $--.--\,m$, lo cual hizo fallar el sistema en el exterior. Al recibir una medida mayor a su máximo rango, y no tener ese caso contemplado, el dron comenzaba a girar sin llegar a parar por constante falso positivo de detección.

A nivel software también se resolvieron inconvenientes. Para que el sistema funcione de forma correcta, la lectura del láser debe de ir en otro hilo de ejecución diferente al hilo principal. En otro caso, las operaciones a realizar por MAVLink pueden ser descartadas y llevar a comportamientos no deseados.

Además, esta solución se basa en sobrescribir valores del mando (a través del comando RC_CHANNELS_OVERRIDE de MAVLink), operación arriesgada de cara a retomar el control si no se libera adecuadamente. Por seguridad se intentará no llevar a cabo soluciones basadas en la toma de control del mando.

Tras descartar la solución inicial, se pasa a desarrollar la descrita una un poco más sofisticada que la anterior. Ya no se actuará sobre el mando, se cambiará el modo de vuelo cuando la distancia al obstáculo sea menor que cierto umbral mediante un command_long de MAVLink. Como la detección se llevará a cabo en modo

automático, se pasará a modo loiter cuando se detecte el obstáculo.

La principal limitación de esta nueva técnica será la velocidad de respuesta al cambio de modo. Dependerá tanto de la velocidad que lleve el dron como del umbral de detección y lo que tarde el empotrado en mandar el comando y el procesado en la controladora de abordo. Datos orientativos de la respuesta a la detección se puede ver en la siguiente tabla:

Umbral de detección (m)	Velocidad en modo automático (m/s)	Distancia a la que se detiene
5 y 10	5	Colisión
5 y 10	3	Pocos centímetros
5 y 10	2	Detención segura

Tabla 1: *Respuesta a la detección en función de la velocidad y del umbral de detección*

Además de responder a la detección parando el vuelo, se aborda la evasión desde el punto de vista de modificar los waypoints de la misión durante el vuelo. Dicha operación es delicada en cuanto al momento de modificar la ruta como el contenido de los nuevos puntos. Un dato erróneo puede dar lugar a un fallo al cargar la ruta.

En una primera aproximación la evasión se plantea en varios pasos:

- Detención del *dron* mediante cambio del modo de vuelo.
- Creación de dos nuevos waypoints en la coordenada GPS donde se ha detectado: uno a la altura actual y otro una cantidad de metros editable por el usuario.
- Modificación del resto de waypoints a la su altura más la cantidad de metros que se ha programado.

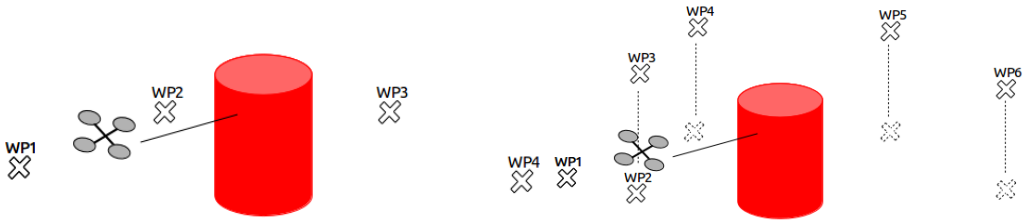

Figura 48: *Evasión mediante modificación de altura. De izquierda a derecha: ruta programada y ruta*

En la Figura 48 se muestra la técnica de evasión explicada. La principal ventaja es que se obtiene una ruta libre de obstáculos, aunque no sea la óptima. Sin embargo, para cada obstáculo se incluyen dos waypoints nuevos, y el hecho de cargar y modificar la ruta puede ser arriesgado.

El sistema también es capaz de generar la ruta que ha seguido el dron en formato GPX (compatible con Google Earth y otros programas) y los waypoints donde ha encontrado obstáculos. Además, para detectar fallos durante el vuelo se añadió un módulo XBEE para transmitir los datos más importantes.

Debido al consumo de los dispositivos conectados, el empotrado no podía alimentar de forma correcta el láser. A partir de este punto la detección obtenía falsos positivos continuamente, por lo que la primera solución fue emplear un sistema empotrado más potente.

3.10.2 Odroid y láser SF02

Para el nuevo empotrado se puede usar la misma estrategia que la descrita anteriormente. Además, la alimentación de los dispositivos a través de los puertos USB no dará ningún problema.

El tiempo de respuesta no varía mucho respecto a la Raspberry Pi B, pues lo que más limita es la temporización de MAVLink y la pérdida de paquetes en vuelo debido a sobrecarga de la conexión serie. Por ello, se hace un nuevo software con una lógica encargada de asegurar la recepción de los comandos MAVLink en la controladora y en el empotrado.

Precisamente, debido a la alta probabilidad de fallo que puede tener la constante lectura, modificación y envío de la ruta, se desarrolla una evasión basada en el modo guiado (guided mode). Así no es necesario modificar la ruta, ya que mediante la introducción de waypoints directamente, el dron irá a la posición indicada.

Una primera versión de esta técnica es detener el vuelo, y desplazarse hacia un lado hasta no encontrar obstáculo. Una vez no se detecte, seguir hacia adelante. La siguiente secuencia de figuras muestra la técnica descrita:

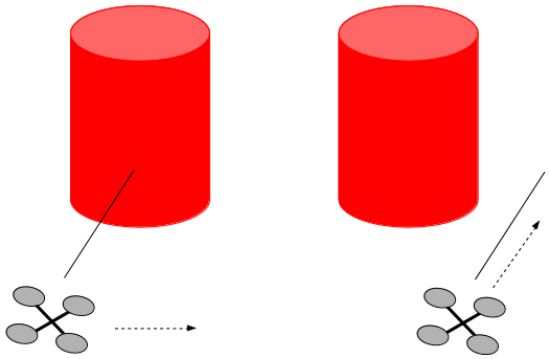

Figura 49: *Primera evasión lateral. De izquierda a derecha: detección y movimiento hacia adelante cuando no hay detección.*

Hay que tener en cuenta un margen de seguridad tras dejar de detectar, para no colisionar con las hélices. Todo ello a base de modo guiado, es decir, sin pedir la ruta al controlador de abordo, lo que reduce las posibilidades de fallo.

3.10.3 Odroid, láser SF02 y ultrasonidos

Al sistema anterior también se le puede dotar de detección mediante ultrasonidos. La idea de este sistema es como guarda al láser. Es efectivo cuando el obstáculo ya se encuentra muy cerca. Además, hace falta introducir un Arduino Nano para hacer la lectura.

La evasión que se plantea para detección por ultrasonidos también es basada en el modo guiado. Cuando el obstáculo ya está demasiado cerca, se opta por insertar un waypoint en modo guiado detrás del dron. Así se separa del obstáculo lo más rápidamente posible.

3.10.4 Arduino Mega 2560, láser SF02 y ultrasonidos

El uso de un empotrado de las características de Odroid XU3 es una solución sobredimensionada al problema propuesto. Las mejora obtenidas son varias: tamaño del sistema final, consumo, comunicación por serie de todos los elementos (no es necesario USB y los problemas de alimentación a través de ellos se erradican) y la solución se ajusta más a las dimensiones del problema.

Para la evasión, se mejora la técnica descrita en la Figura 49. Si existiese algún obstáculo en el lateral hacia el que se desplaza, el dron impactaría contra él. Por ello, primero se orienta el láser hacia el lado al que se desplaza y posteriormente comienza el movimiento

lateral.

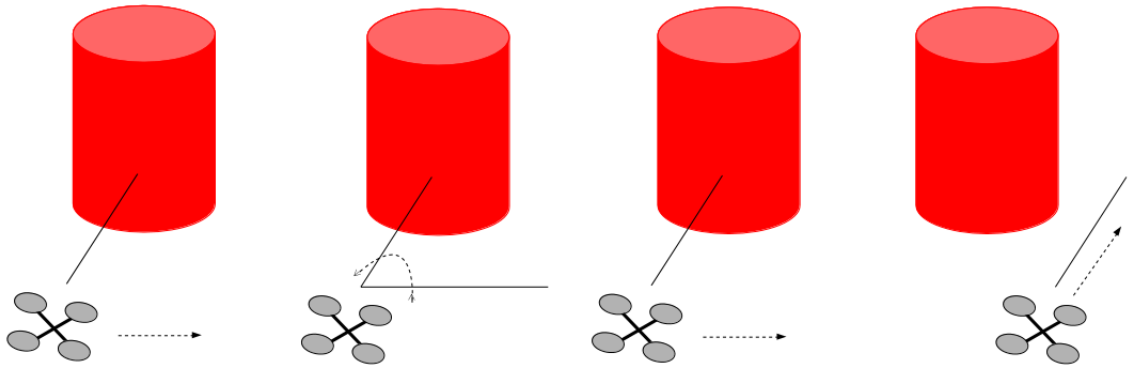

Figura 50: *Evasión lateral modificada. De izquierda a derecha: detección, giro para comprobar obstáculos laterales, movimiento lateral, continua hacia adelante.*

3.10.5 Arduino Mega 2560, láser LIDAR Lite y servo

El segunda láser testeado es LIDAR Lite. Como ya se ha descrito, es de un tamaño mucho más reducido que el SF02, por lo que sigue ganando en tamaño. Las especificaciones técnicas lo hacen más apto para el sistema.

El formato en que devuelve la medida tomada es mucho más manejable que el SF02. De todas formas, también hay que tener precaución con el caso extremo: para la máxima distancia alcanzable se tendrá una lectura de cero metros.

Este método representa una gran ventaja pues proporciona gran precisión y un coste moderado.

La desventaja de basar la detección y evasión en las medidas tomadas por un láser es que sólo existe un punto de detección. Las decisiones se basan en la medida de distancias de un único punto. Para tener más referencias, se puede incluir un servo en el sistema, lo que hace LIDAR Lite la mejor opción. Con ello, se podrá tener una estimación de la anchura del obstáculo. Además, habría que decir también que proporciona sólo un análisis puntual en una única dirección. Tiene la posibilidad de ampliar el rango analizado por medio de un sistema de servo-motores. La lectura de los datos se realiza de forma secuencial y es necesario una calibración exacta de los servos y el láser para obtener medidas útiles.

3.11 Ayudas a la detección

La visión artificial no es la solución única a la detección de obstáculos. De hecho, hay que usar más métodos de apoyo a la detección. Debido a que las cámaras nos proporcionan un rango de actuación máximo y mínimo, es posible que no se tenga certeza de si existen obstáculos en el espacio más cercano.

3.11.1 Detección mediante ultrasonidos

Para la detección de objetos que pueden no ser vistos mediante visión artificial, se puede hacer uso de ultrasonidos [46]. Por ejemplo, las ventanas o superficies transparentes o cables de la red eléctrica. Mediante un sensor de ultrasonidos pueden suplirse éstas carencias.

El sensor que se ha probado ha sido el SRF10. Sensor de ultrasonidos conectado mediante I2C, con un rango máximo de detección de once metros, configurado para detectar hasta dos metros:

Figura 51: *Vista del sensor de ultrasonidos SRF10.*

Funciona correctamente en interiores en un espacio cercano (hasta dos metros). Las pruebas se han realizado sobre un *Arduino MEGA 2560*, desarrollado bajo la premisa de ser hardware y software libre:

Figura 52: *Arduino MEGA 2560.*

Como ventaja, cabría destacar que son sensores muy económicos y con una interfaz de uso sencilla. Sin embargo, no es posible obtener medidas exactas de posicionamiento de obstáculos y en situaciones de ruido ambiente el rango de funcionamiento de los ultrasonidos utilizados puede estar saturado, eliminando cualquier posibilidad de medida.

3.11.2 PX4FLOW

Durante el desarrollo de esta tesis, se ha estudiado el uso de ésta plataforma para ayudas a la detección:

Figura 53: *Vista superior del PX4FLOW.*

Equipado con una cámara de flujo óptico (global shutter), calcula el flujo de píxeles entre cuadros. Las primeras pruebas se han realizado bajo condiciones de laboratorio, pues la lente es muy sensible a la luz, y hay que aclimatarla. Aunque no se han obtenido resultados

concluyentes, se propone este sistema como objeto de futuros estudios.

La utilidad de éste sistema es la gran precisión con la que obtiene la altura. Además, de cara al seguimiento de la cámara en la parte de reconstrucción tridimensional será de gran utilidad disponer de un sistema parecido.

3.12 Búsqueda de caminos y generación de planes

Para conseguir dotar a vehículos no tripulados de autonomía, es esencial tener en cuenta que debe ser capaz de moverse en un escenario sin colisionar con obstáculos o seguir rutas previamente dictadas evadiendo los obstáculos que pudiese encontrar.

Para ello es necesario implementar algoritmos para la navegación autónoma (motion planning), tanto de forma global como de forma local:

- Algoritmos para la navegación en **caminos completos**. Se refiere a ir de un punto origen a un punto destino evadiendo obstáculos considerados grandes. Se tiene un conocimiento de la misión en un nivel de abstracción alto. En la figura 54 se muestra una planificación de una misión a alto nivel, donde se pueden tener en cuenta los obstáculos más grandes:

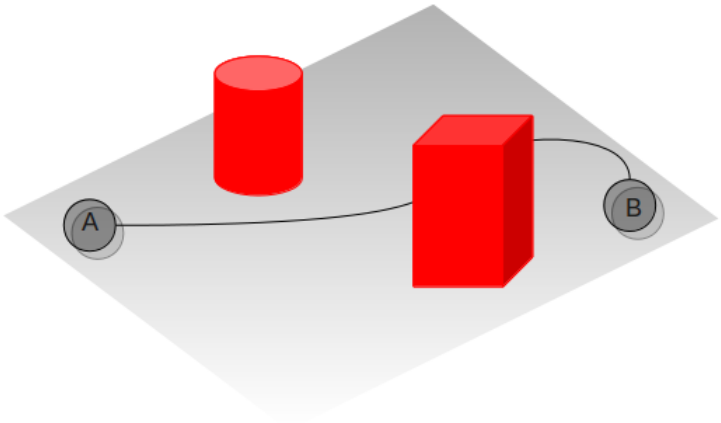

Figura 54: *Plan de una misión a nivel global.*

- Algoritmos para la navegación **dentro del camino completo**. A nivel local, se pueden encontrar objetos de menor tamaño, pero

que también son necesarios de esquivar. Por ello, hay que tener un plan de actuación dentro del camino global para posibles incidencias no contempladas inicialmente. En la figura 55 se puede ver un ejemplo de la planificación local:

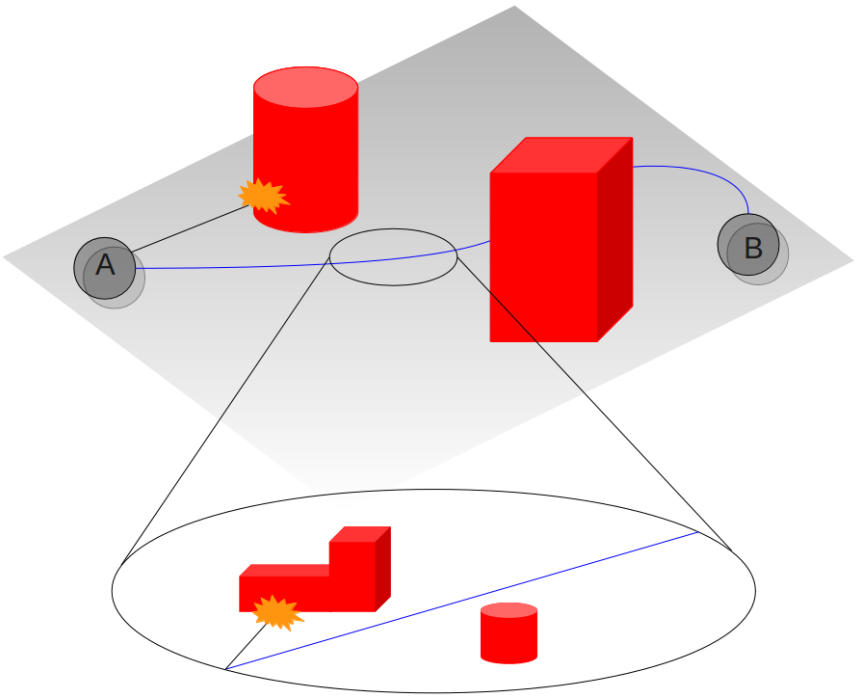

Figura 55: *Plan local dentro del plan global.*

Es de suma importancia no confundir los algoritmos de navegación y búsqueda de caminos con los algoritmos para representar el mapa del camino. Estos últimos pretenden almacenar el mapa del escenario, mientras que los primeros pretenden navegar el escenario de forma autónoma. El primero puede usarse para completar al segundo.

3.12.1 Camino completo

Por lo general, para navegar de forma autónoma un escenario, se necesita hacer un estudio previo de los obstáculos. Una vez gestionados los obstáculos, se pueden saber los espacios libres de obstáculos.

Como este problema es de naturaleza tridimensional, se tendrá que trabajar en un sistema euclídeo especial de orden 3 (SE(3)), que vendrá dado por seis parámetros:

$$SE(3) = R^3 \; x \; SO(2) \rightarrow (x, y, z, \alpha, \beta, \gamma) \qquad \textbf{Ecuación 3.A.23}$$

Es decir, se necesita describir el estado del vehículo aéreo no tripulado mediante su posición (x, y, z) y sus ángulos de Euler (α, β, γ).

Una vez obtenidos los obstáculos del escenario, se podrá generar mapas de ocupación (de espacio libre si se invierte la notación), márgenes de seguridad en torno a los obstáculos para la geometría del vehículo e incluso mallas de nodos para buscar un camino (ya sea óptimo o no), tal y como se realiza aplicando por ejemplo el algoritmo KPIECE [47]:

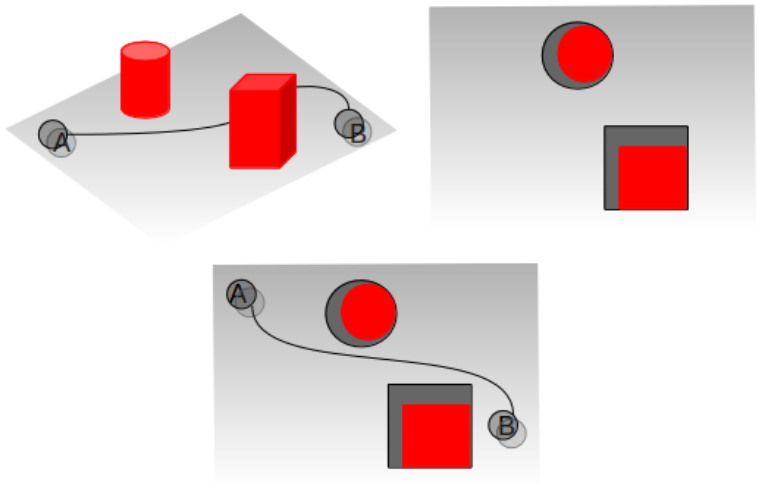

Figura 56: *De arriba a abajo y de izquierda a derecha, escenario original, mapa de ocupación con márgenes de seguridad y camino sobre el mapa de ocupación.*

La mayor parte de algoritmos para búsqueda de caminos se basan en crear mallas y buscar un camino basado en algún heurístico. No se obtiene la solución exacta, pues computacionalmente no es viable, pero se obtiene un camino desde el objetivo al destino [48] [49].

La terminología que se suele usar para formular el problema es:

- **Configuración**. Una posición específica del vehículo dada por los seis parámetros descritos anteriormente. Su posición en el espacio y los ángulos de Euler correspondientes.

- **Espacio de configuraciones**. Es el conjunto de todas las configuraciones posibles del vehículo.

- **Camino**. Mapa entre la configuración inicial y la configuración final.

- **Espacio de configuraciones libres de obstáculos**. Espacio de configuraciones donde no hay obstáculos.

- **Configuración libre de obstáculos**. Cualquier configuración del espacio de configuraciones libres de obstáculos.

- **Mapa libre de obstáculos**. Mapa continuo de configuraciones libres de obstáculos desde la configuración libre inicial a la configuración libre objetivo.

Una vez formulado el lenguaje usado para el problema, se puede resumir un algoritmo de búsqueda de camino en la siguiente figura:

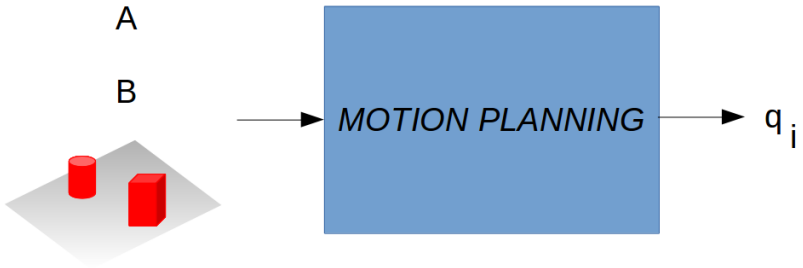

Figura 57: *Algoritmo de búsqueda de camino.*

La entrada a la planificación es la configuración inicial (A), la configuración final (B) y la reconstrucción de la escena calculada previamente en la etapa del SLAM

3.12.2 Dentro del camino completo

Es posible que aparezcan obstáculos nuevos mientras que el vehículo está llevando a cabo una misión. Para gestionar este tipo de incidencias será necesario detectar obstáculos y planificar la evasión de los mismos.

Esta parte compete más al área de visión artificial y detección de obstáculos que a la reconstrucción tridimensional. Por ello, esta tesis se centra más en los algoritmos de camino completo.

3.13 Registro de pruebas realizadas

3.13.1 Prueba de disparidad estéreo con Minoru webcam

Para esta simulación se ha utilizado la webcam Minoru en modo estéreo. Se ha realizado un programa en C++ utilizando las librerías OpenCV. El cálculo de disparidad está realizado con el algoritmo SGBM La cámara está sin calibrar, pero se le ha aplicado una transformación matricial para corregir parcialmente el desfase de las cámaras estéreo.

En la figura 58 se puede ver una prueba del resultado de la prueba.

Figura 58: *Prueba de disparidad con webcam Minoru.*

3.13.2 Prueba de calibración estéreo con Minoru webcam

En esta simulación se ha probado el software de calibración que se ha diseñado. Para calibrar las cámaras, del modo que se ha explicado en apartados anteriores, se ha desarrollado un programa en C++ que permite automatizar el proceso de calibración. En la figura 59 se puede ver una captura de cómo realizar las correspondencias de las imágenes usando el modelo clásico del tablero de ajedrez.

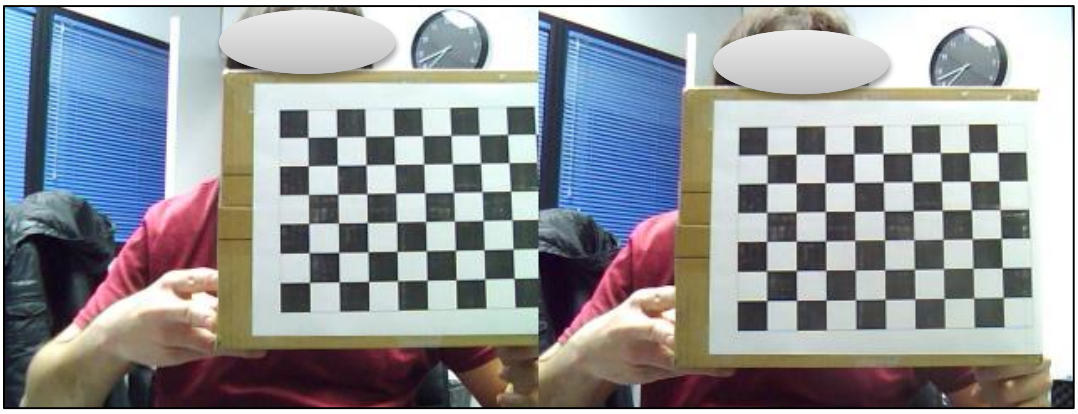

Figura 59: *Calibrando con webcam Minoru.*

Una vez se calibran las cámaras, se utilizan las matrices de proyección y distorsión para rectificar la imagen y conseguir así una mejor imagen de disparidad. En la figura 60 se puede ver un ejemplo con la imagen normal, la imagen rectificada, y la imagen de disparidad. A su vez, se ha creado un algoritmo para convertir la disparidad en blanco y negro a escala de colores RGB

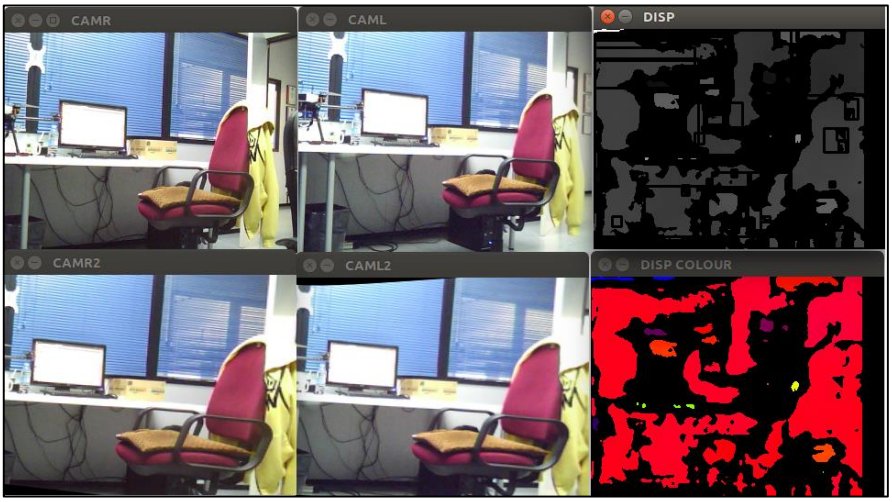

Figura 60: *Imagen real, imagen rectificada e imagen de disparidad con webcam Minoru.*

3.13.3 Cálculo de nube de puntos con webcam Minoru

En la siguiente simulación se ha ampliado nuestro software para realizar el cálculo de nube de puntos. Para ello se ha utilizado la librería PCL ***[3]***, que es una librería de código abierto para manejo de nubes de puntos. Utilizando la imagen rectificada, la imagen de disparidad y la matriz de profundidad Q, es posible realizar la

[3] *http://www.pointclouds.org/ (último acceso: 12-oct-2015)*

reproyección de los píxeles de la imagen y obtener una nube de puntos con profundidad. En la figura 61 se puede ver un ejemplo de esta prueba.

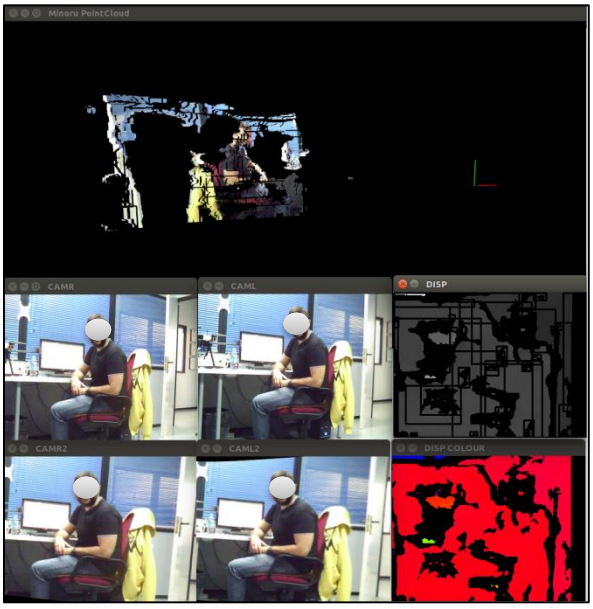

Figura 61: *Cálculo de nube de puntos con webcam Minoru.*

3.13.4 Prueba de calibración y cálculo de disparidad con ROS y webcam Minoru

Para esta simulación se ha utilizado ROS, en lugar de programar directamente el software en C++. ROS incluye un calibrador propio, y además no permite calcular la disparidad sin haber calibrado las cámaras primero. Por ello el primer paso consiste en calibrar las cámaras con el software que incluye ROS, que funciona de manera similar al que se creó anteriormente en C++. En la figura 62 se puede ver un ejemplo del calibrador de ROS

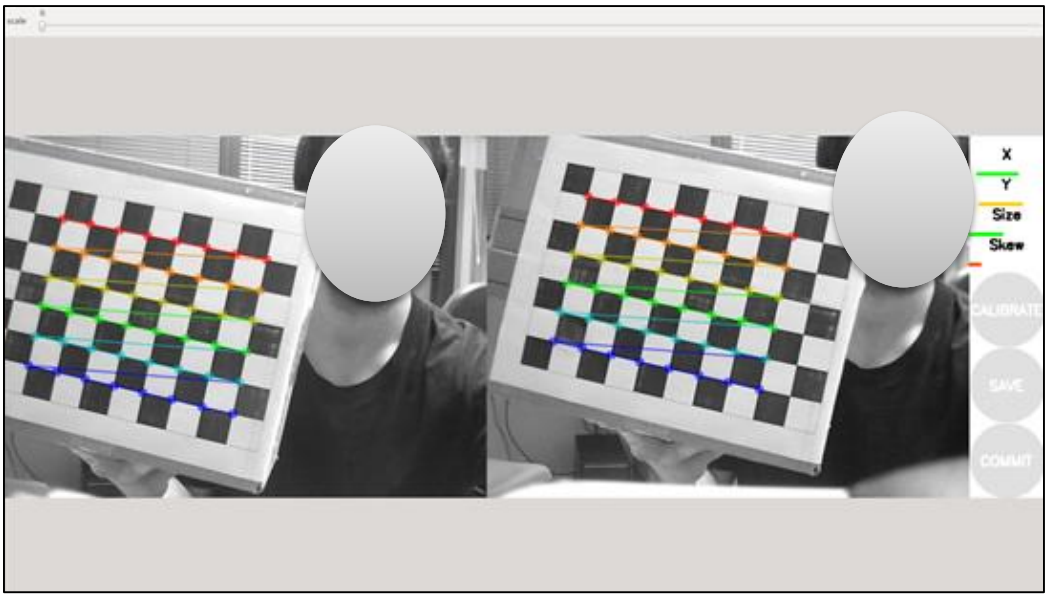

Figura 62: *Calibrando con ROS.*

Una vez que se han calibrado las cámaras, se puede ver la imagen de disparidad que es calculada por el nodo stereo_image_proc. Si además se ejecuta el nodo rqt_reconfigure, se pueden ajustar los parámetros de disparidad para obtener exactamente el resultado deseado. En la figura 63 se puede ver un ejemplo de la prueba.

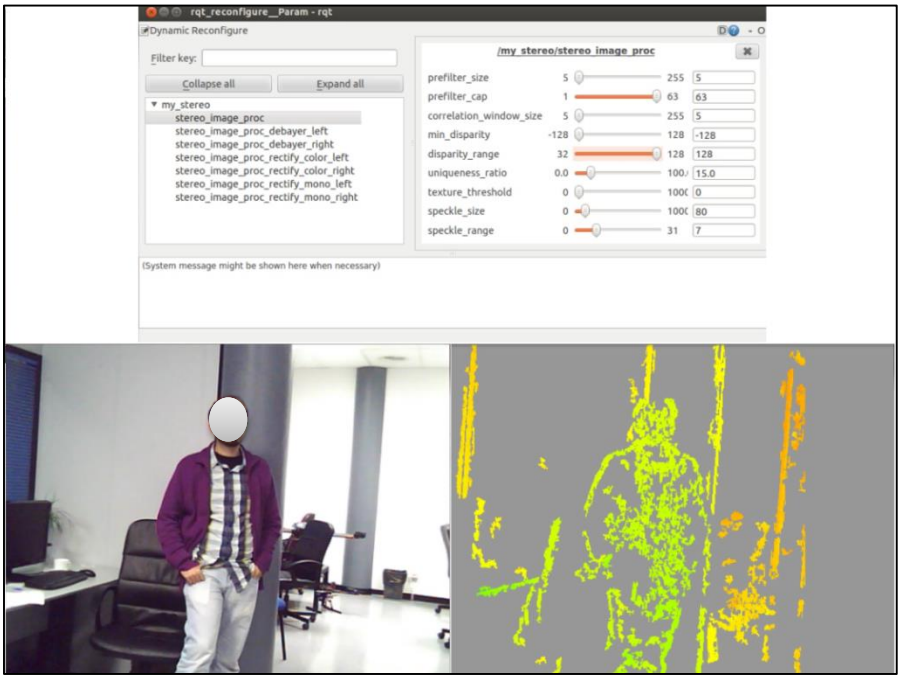

Figura 63: *Cálculo de disparidad con ROS y webcam Minoru.*

3.13.5 Cálculo de nube de puntos con ROS y webcam Minoru

En esta simulación se utiliza ROS para el cálculo de disparidad. Además, ROS contiene un nodo llamado rviz, que permite visualizar escenas 3D. El nodo de ROS stereo_image_proc, aparte de calcular la disparidad, también es capaz de calcular la nube de puntos en formato pointCloud. Se utiliza por tanto rviz para visualizar la nube de puntos, como se puede ver en la figura 64.

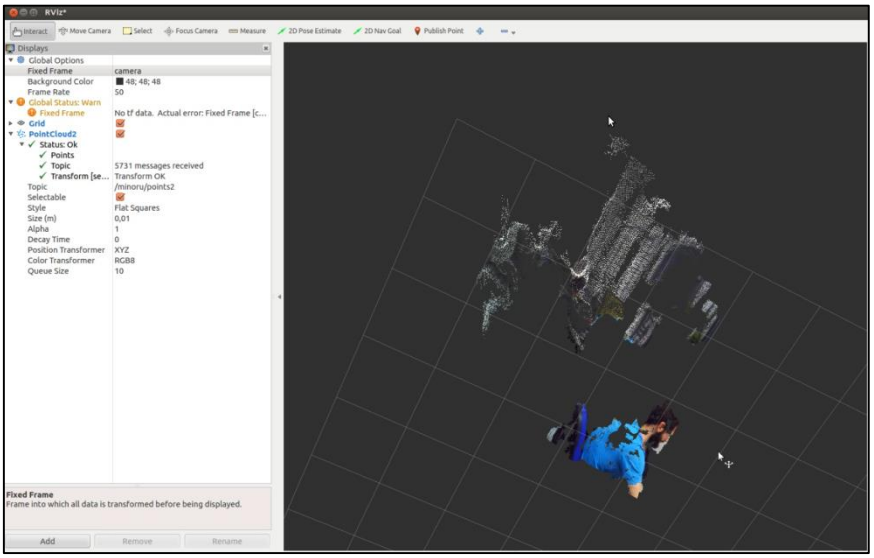

Figura 64: *Visualización de la nube de puntos con ROS y webcam Minoru.*

3.13.6 Cálculo de nube de puntos con ROS visualización de Octomap

Con la prueba anterior se conseguía visualizar la nube de puntos con el nodo rviz de ROS Dado que las nubes de puntos pueden ser muy densas, ROS permite utilizar la librería Octomap, que es una librería de código abierto que simplifica la nube de puntos en cubos de un tamaño específico, disminuyendo así la carga de procesamiento. Utilizando nuevamente rviz, se pueden visualizar estos cubos, como se puede ver en la figura 65.

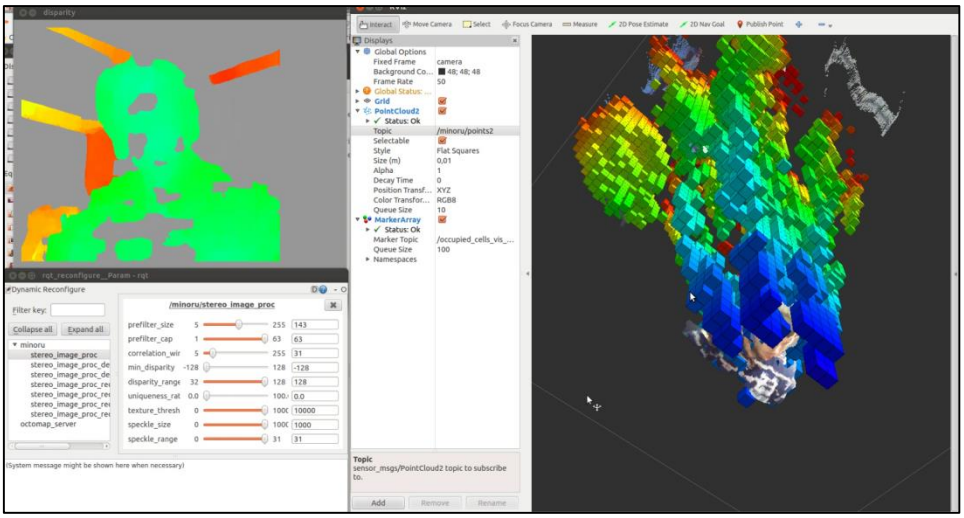

Figura 65: *Visualización de la nube de puntos y Octomap con ROS.*

3.13.7 Prueba de cálculo de distancias con ROS y webcam Minoru

Una vez se ha conseguido calcular un mapa de disparidad estable, se pueden obtener mediciones reales sobre él. Utilizando índices como referencia, se ha desarrollado un algoritmo para calcular distancias a partir del mapa de disparidad. Para ello se ha realizado una búsqueda de contornos en la imagen de disparidad, y se han filtrado los objetos por área, obteniendo siempre el índice del objeto más cercano a la cámara. Tomando ciertos índices de color como referencia, y utilizando interpolación lineal, se ha conseguido obtener mediciones fiables de distancias. En la figura 66 se puede ver un ejemplo de esta prueba.

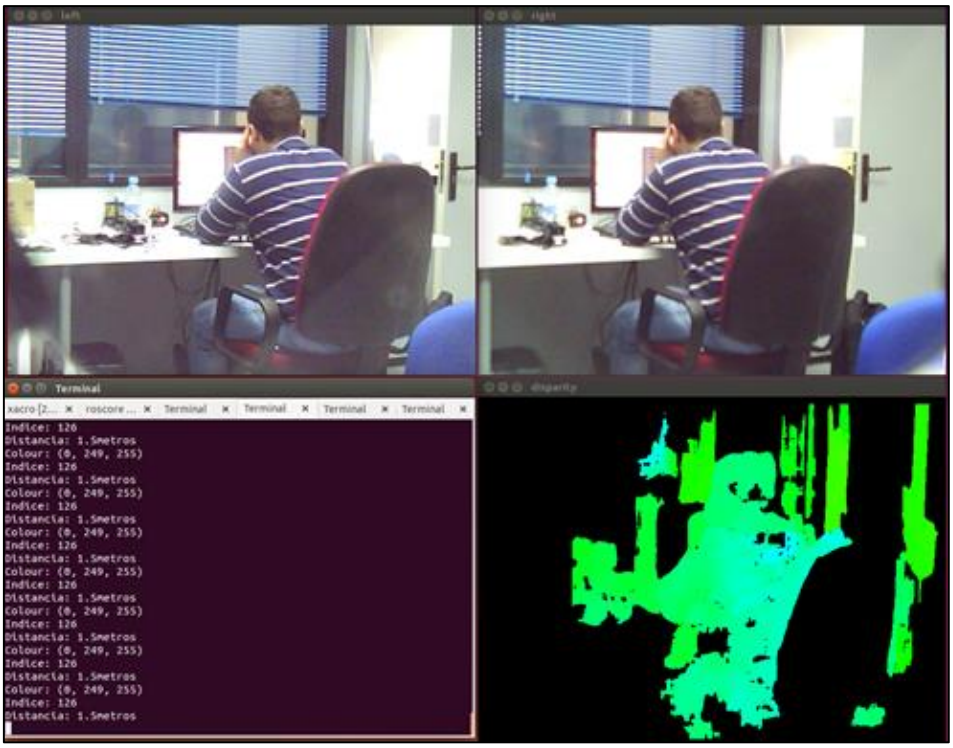

Figura 66: *Cálculo de distancias con ROS y webcam Minoru.*

3.13.8 Prueba de webcam Minoru en exteriores

Una vez se han realizado todas las pruebas oportunas en interiores, se procede a probar el estado de la imagen que se obtiene de la webcam estéreo Minoru en el exterior, y se comprueba así si las condiciones lumínicas afectan a la calidad de la imagen. Como se puede ver en la figura 67, la luz del sol satura la imagen de la cámara, y por tanto el cálculo de disparidad que se realiza es bastante pobre.

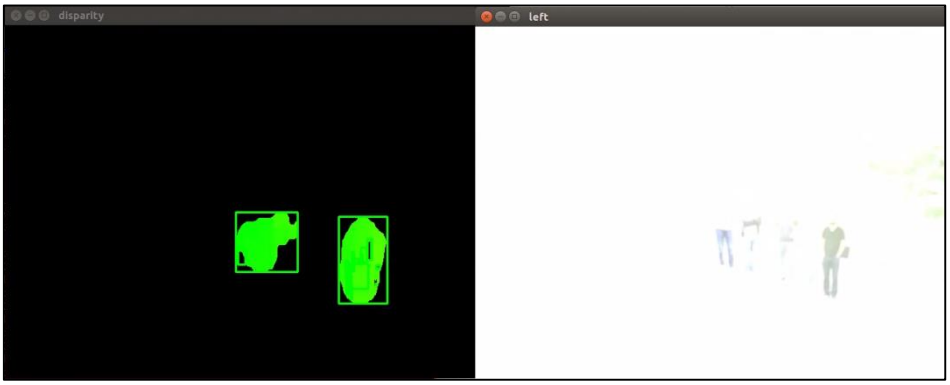

Figura 67: *Prueba con webcam Minoru en exteriores.*

Queda comprobado por tanto que la webcam Minoru no es apta para exteriores, ya que es incapaz de ajustar su visión a la luminosidad exterior. Además, tampoco permite modificar parámetros de la cámara para ajustarse a condiciones externas.

3.13.9 Prueba con Minoru en exteriores con radiografía

Dado el resultado anterior obtenido con la webcam Minoru en exteriores, ha quedado demostrado que esta cámara no es apta para obtener imágenes bajo la luz del sol. Para solucionar este problema se ha utilizado una radiografía. Poniendo dos trozos de radiografía sobre las lentes de la cámara, se ha conseguido que la imagen no se vea afectada tanto por las condiciones lumínicas del exterior, de modo que sea viable procesar imágenes con la webcam Minoru bajo la luz del sol.

En la figura 68 se puede ver un ejemplo de la webcam Minoru grabando en exteriores.

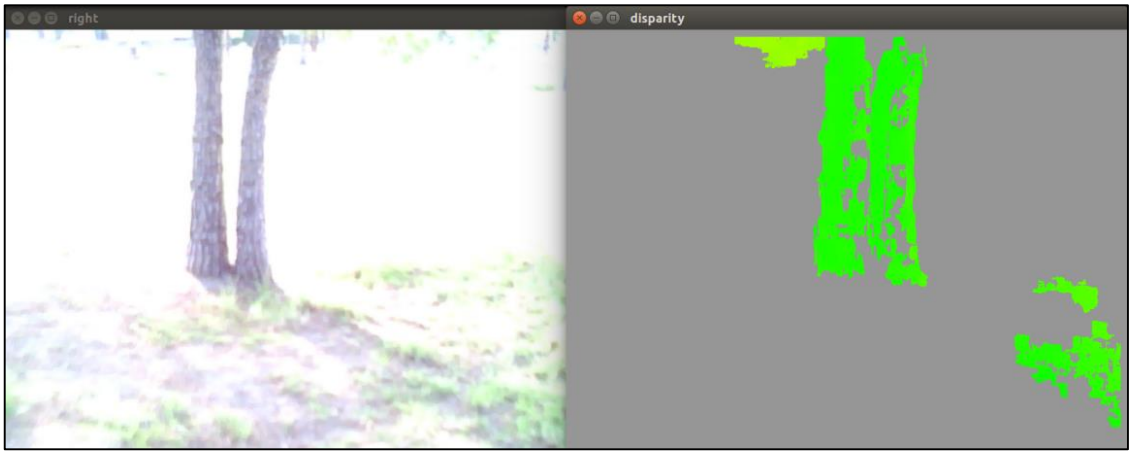

Figura 68: *Prueba con webcam Minoru y radiografía en exteriores.*

Al utilizar la radiografía, la imagen se quema menos, pero aún no se obtiene un resultado realmente bueno. Por tanto, se puede concluir que la webcam Minoru no es apta para exteriores.

3.13.10 Prueba de cámaras IDS en modo estéreo

Las cámaras IDS son individuales y se conectan al PC mediante USB Como ya se ha mencionado anteriormente, para poder obtener imágenes de estas cámaras tenemos que instalar los drivers propios

de IDS Estos drivers proporcionarán acceso a la librería ueye.h, que es la que se utilizará para interactuar con las cámaras y obtener sus imágenes.

Ya que las cámaras son individuales, el primer paso es conectarlas en modo estéreo, y para ello se han colocado las cámaras en una pieza de carbono. No obstante la imagen de las cámaras, al ser individuales, deben ser sincronizadas para que, al recibir la imagen estéreo, ambas imágenes se correspondan con el mismo instante de tiempo.

Para sincronizar las cámaras se ha utilizado un hilo (del inglés "thread") que actúa como un temporizador, de modo que cuando el temporizador llegue a cero se obtenga un nuevo frame de ambas cámaras a la vez. En la figura siguiente se puede ver un ejemplo de la imagen estéreo de las cámaras IDS

Figura 69: *Imagen estéreo de las cámaras IDS.*

3.13.11 Prueba de disparidad estéreo con cámaras IDS

Una vez que las cámaras IDS están conectadas en modo estéreo, se puede computar la imagen de disparidad. Para ello se ha realizado un programa en C++ que capture las imágenes de ambas cámaras y realice el cálculo de disparidad mediante el algoritmo SGBM

En la figura 70 se puede ver un ejemplo de la imagen de disparidad obtenida. Dado que la imagen presenta una fuerte distorsión de barril (debido a las lentes de ojo de pez), se puede observar que la disparidad no es buena. Se debe, por tanto, calibrar las cámaras y quitar la distorsión a la imagen.

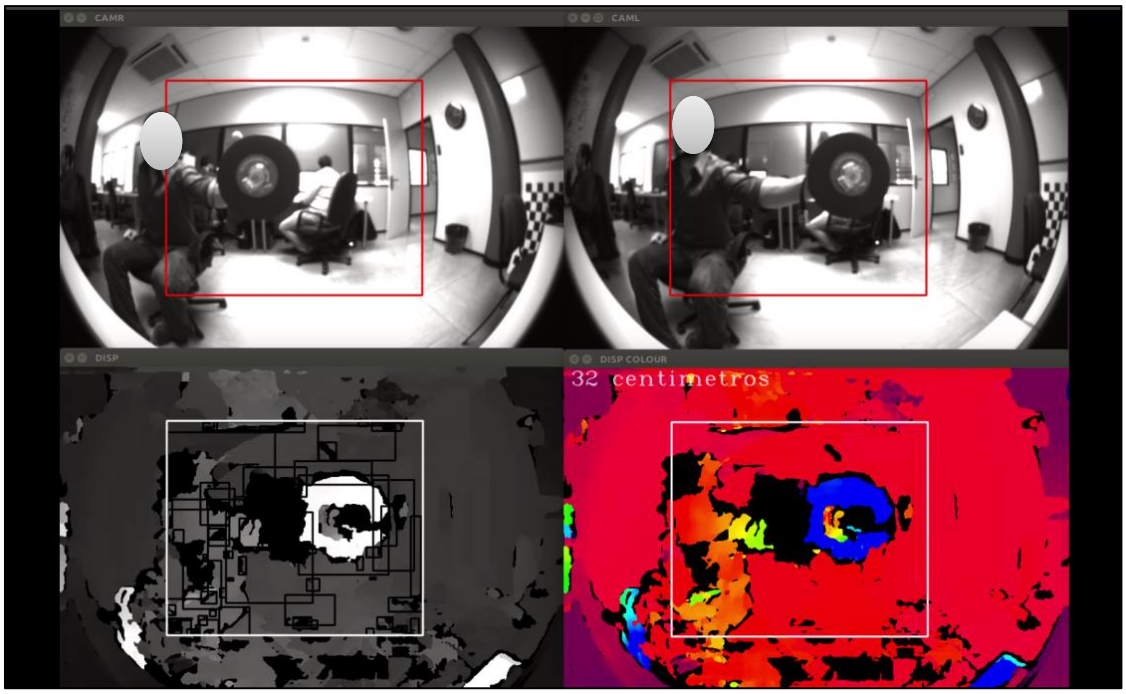

Figura 70: *Imagen de disparidad de las cámaras IDS.*

3.13.12 Calibración de las cámaras IDS en modo estéreo

Como se ha comprobado en la prueba anterior, las cámaras IDS necesitan ser calibradas para poder eliminar la distorsión de barril que presentan las imágenes. Para resolver este problema, se ha creado un programa en C++ específico para calibrar cámaras con lente fisheye. El programa actúa sobre cada cámara por separado y requiere la utilización del clásico tablero de ajedrez para calibrar. En la figura 71 se puede ver un ejemplo de imagen estéreo normal e imagen estéreo con la distorsión eliminada.

Como se puede observar, al eliminar la distorsión de barril, la imagen pierde información en los bordes, por lo que, en este caso, las cámaras estarían perdiendo parte de su ángulo de visión a cambio de la obtención de una imagen corregida.

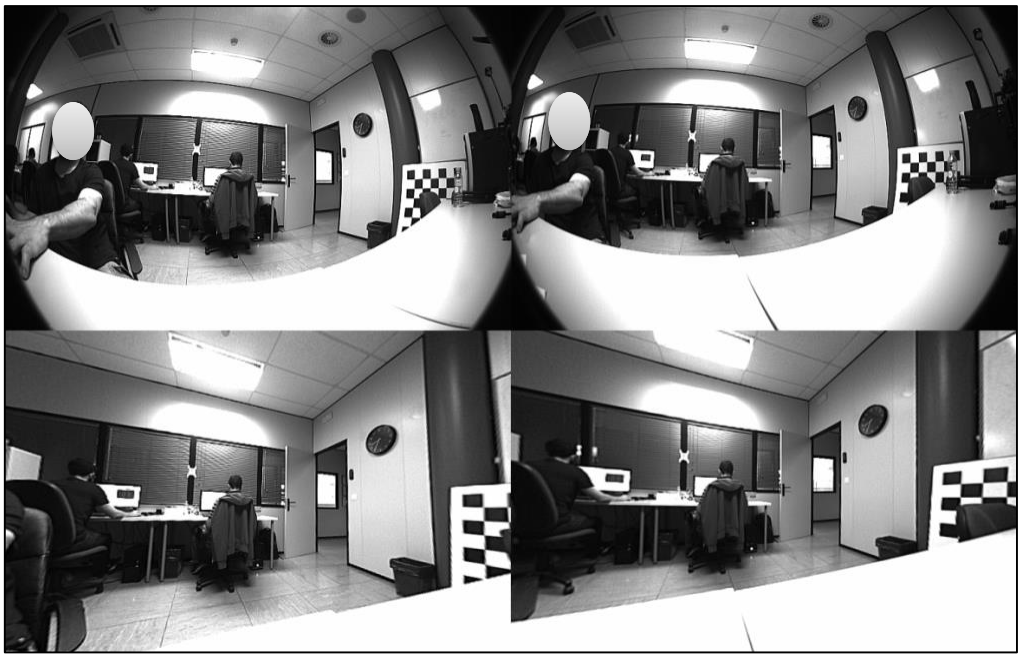

Figura 71: *Arriba: imagen raw de las cámaras IDS. Abajo: Imagen sin distorsión de las cámaras IDS.*

3.13.13 Prueba de disparidad estéreo con cámaras IDS

Una vez que están las cámaras calibradas correctamente y se puede recibir la imagen sin distorsión y rectificada, se procede a realizar nuevamente el cálculo de la imagen de disparidad. Para ello se ha creado un programa C++ que utiliza las matrices de calibración monocular para quitar la distorsión a la imagen y las matrices de calibración estéreo para rectificar la imagen. Finalmente, calcula la imagen de disparidad utilizando el algoritmo SGBM En la figura 72 se puede ver un ejemplo.

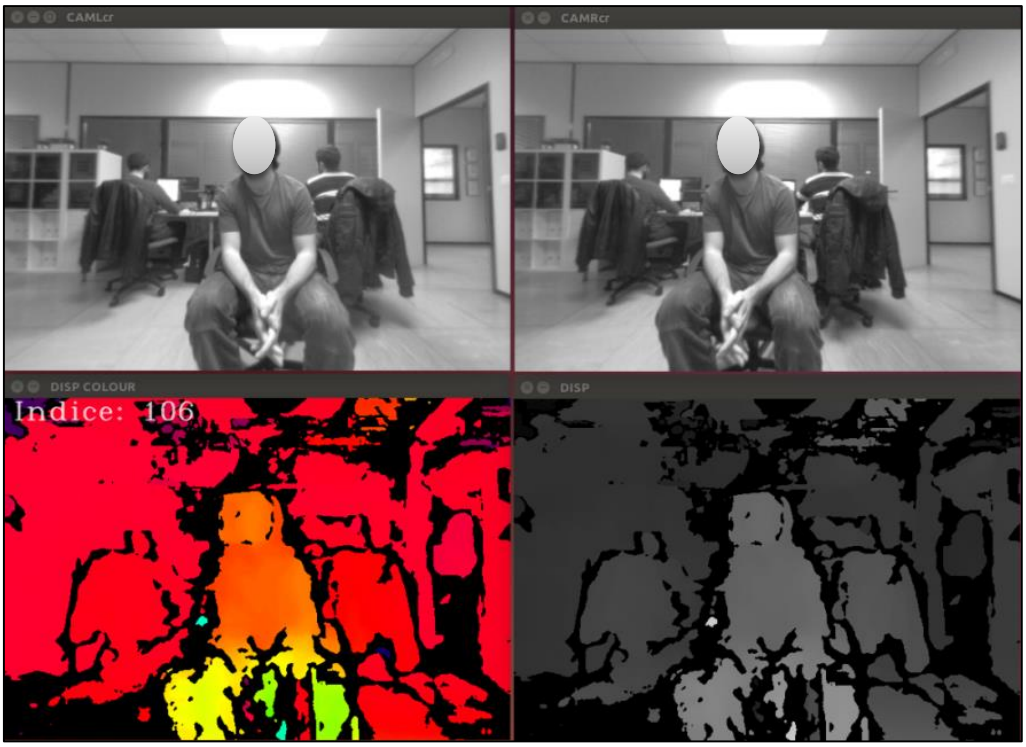

Figura 72: *Imagen de disparidad con las cámaras IDS.*

3.13.14 Prueba de cálculo de distancia con cámaras IDS

Una vez que se ha conseguido calcular un mapa de disparidad estable con las cámaras IDS, se puede proceder a realizar la computación de mediciones reales. Para ello se ha desarrollado una función en C++ que filtra la imagen de disparidad, busca el contorno de los objetos que aparecen, y calcula el índice máximo (más cercano) de entre todos los objetos que aparecen en la imagen. Para calcular la distancia se han tomado ciertos índices como referencia y se ha utilizado interpolación lineal para calcular la distancia correspondiente a cada uno. El resultado puede verse en la figura 73.

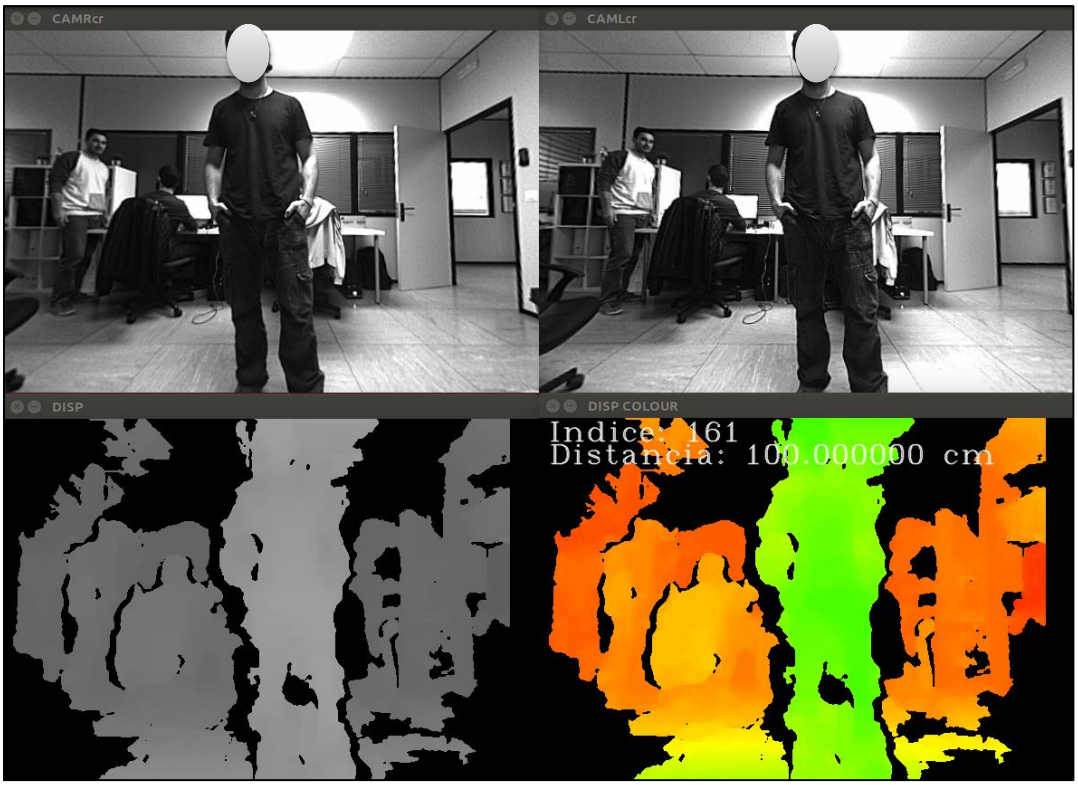

Figura 73: *Cálculo de distancias con cámaras IDS.*

3.13.15 Prueba de cámaras IDS en exteriores

Una vez que las cámaras IDS están funcionando en interiores y con el cálculo de distancias realizado, el siguiente paso es probarlas en el exterior para ver si son capaces de funcionar correctamente bajo la luz del sol, o si por el contrario se tienen problemas similares a los que experimentamos con la webcam Minoru. Como se puede ver en la figura 74, la imagen de las cámaras IDS muestran mejor comportamiento ante la luz del sol que la webcam Minoru, pero aun así el sol es capaz de afectar a la imagen y, por consiguiente, a la corrección del mapa de disparidad.

Figura 74: *Imagen de las cámaras IDS en el exterior.*

Para solucionar este problema, se ha utilizado una de las funcionalidades de la librería *ueye.h*, que consiste en la capacidad de modificar los parámetros de configuración de la cámara. Concretamente, para ajustar la imagen a las condiciones lumínicas exteriores, se ha configurado el tiempo de exposición de la cámara.

El obturador se abre en el momento de disparar y limita el tiempo que el rayo de luz penetra en la cámara y alcanza el sensor digital. El tiempo que la luz está alcanzando el sensor digital es lo que se llama tiempo de exposición. Simplificando, se puede decir que el tiempo de exposición es el tiempo que está haciéndose la foto. Ajustando el tiempo de exposición a valores mínimos la cámara responderá mejor ante la luz del sol. Además si se ajusta el HW gamma (es un parámetro que se puede modificar en las cámaras Global Shutter que permite maximizar el contraste de la imagen con lo que se consiguen mejores resultados en el cálculo de la disparidad) y la ganancia, la imagen se vuelve más nítida. En la figura 75 se puede ver como la imagen en el exterior ha mejorado notablemente configurando el tiempo de exposición, el HW gamma y la ganancia.

Figura 75: Imagen de las cámaras IDS configuradas en el exterior.

3.13.16 Prueba de calibración de cámaras IDS en exteriores

Para esta prueba, se han sacado las cámaras IDS a una explanada en el exterior y se ha comprobado si es posible realizar la calibración de las cámaras en condiciones de luz solar. En la figura 76 se puede ver el escenario de la prueba.

Figura 76: *Calibrando las cámaras IDS en el exterior.*

Dada la corrección de parámetros que se han realizado, se ha conseguido que la imagen en el exterior sea lo suficientemente buena como para permitir una buena calibración. En la figura 77 se puede ver un ejemplo de imagen rectificada en el exterior.

Figura 77: *Imagen rectificada de las cámaras IDS en el exterior.*

3.13.17 Prueba de cálculo de distancia en el exterior con cámaras IDS

Recapitulando lo conseguido hasta el momento, se dispone ya de la calibración monocular y estéreo de las cámaras IDS, del cómputo de la imagen de disparidad y el algoritmo de cálculo de distancias. La siguiente simulación es, por tanto, probar todo el cálculo de distancias para el exterior. Para ello se han sacado las cámaras a una explanada en condiciones lumínicas altas y se ha colocado una cinta métrica en el suelo para comprobar las mediciones. En la figura 78 se puede ver el escenario de la prueba.

Figura 78: *Escenario de la prueba para el cálculo de distancias con cámaras IDS en el exterior.*

Una vez dispuesto el escenario, uno de los miembros del equipo se ha colocado en distintas posiciones junto a la cinta métrica para comprobar las distancias que mide la cámara, desde distancias más cercanas, hasta donde las cámaras siguen siendo capaces de discernir disparidad. En la figura 79 se puede ver un ejemplo de esta prueba.

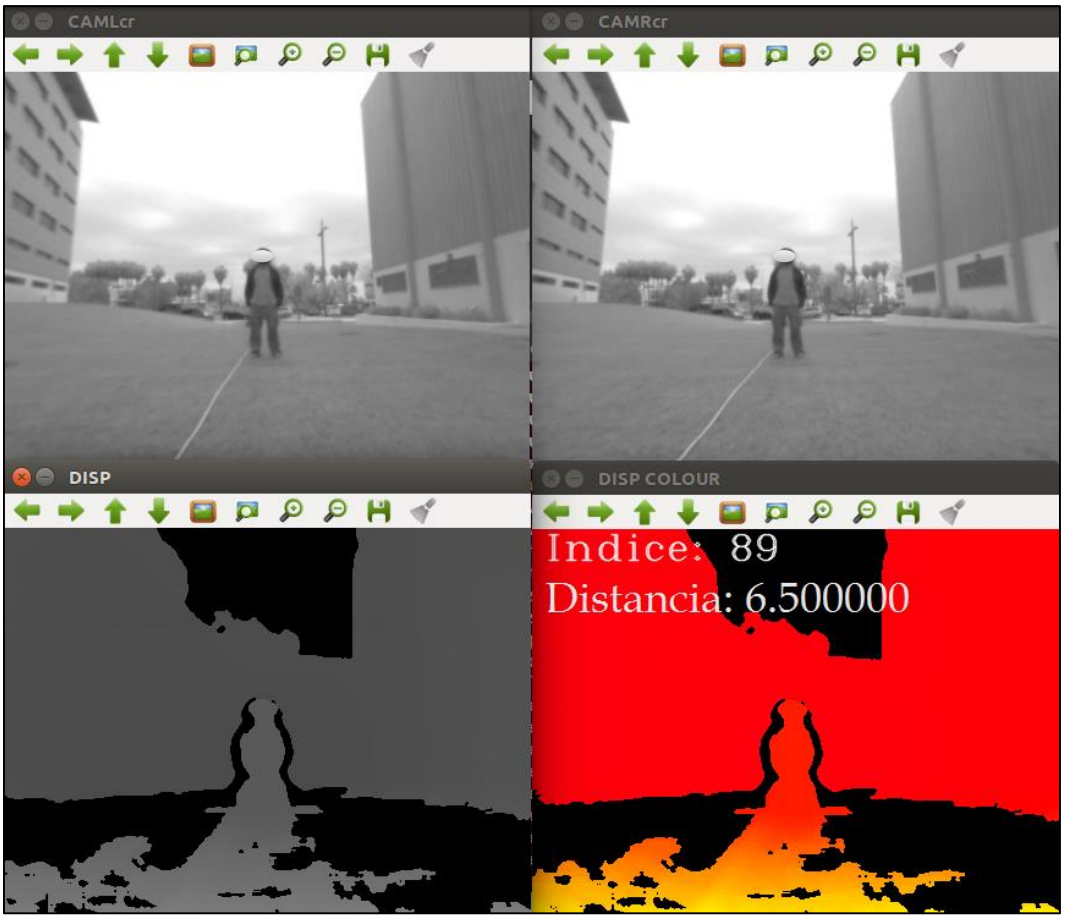

Figura 79: *Escenario de la prueba para el cálculo de distancias con cámaras IDS en el exterior.*

Tras la prueba, se puede concluir que el cálculo de distancias funciona bien y con un margen de error de pocos centímetros. Además, las cámaras IDS son capaces de captar distancias de hasta 7 metros.

BLOQUE B: MAPEO ROBÓTICO Y POSICIONAMIENTO

Uno de los objetivos a la hora de crear un vehículo no tripulado es su capacidad de ser autónomo. Bajo un escenario dado, autonomía es sinónimo de conocer el terreno y saber moverse sin necesidad de interacción humana.

Para cumplir dicho objetivo es necesario, por un lado, que el robot sea capaz de reconstruir un escenario de forma que sea comprensible para el vehículo y por otro, mantener el posicionamiento relativo de forma simultánea a la dicha reconstrucción. Las técnicas que tratan de dar solución al problema anterior se conocen como técnicas de SLAM, las cuales serán estudiadas en detalle en los siguientes puntos.

3.14 Técnicas de SLAM

El término *SLAM* (Simultaneous Localization And Mapping) ya introducido anteriormente, hace referencia al problema de establecer un robot móvil en una posición y entorno inicial desconocidos, y que este sea capaz de realizar una reconstrucción incremental de un mapa del entorno, de forma que utilice dicho mapa para determinar su propia localización.

Una de las mayores dificultades a la hora de implementar cualquier técnica de SLAM es el problema conocido como cierre de lazo (loop closure), que surge como consecuencia del error acumulativo producido en la estimación de los *poses* del robot móvil. La dificultad reside en detectar cuando el robot ha vuelto a una posición en la cual ya ha estado anteriormente. Para solucionarlo, se suele utilizar un algoritmo que realiza comparaciones entre similitudes capturadas por el sensor que mide la trayectoria (en el caso de Visual SLAM se suelen emplear comparaciones de características de la imagen a partir de cada una de las posiciones ya visitadas).

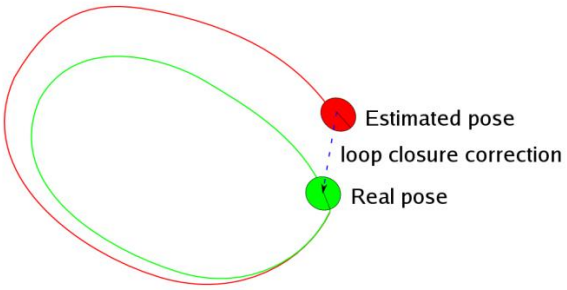

Figura 80: *Idea simplificada de corrección de la pose estimada aplicando el cierre de lazo.*

Existen diferentes técnicas de SLAM que tratan de dar solución al problema anterior desde distintos enfoques, aunque todos ellos están basados en modelos de probabilidad bayesianos, debido a la incertidumbre que normalmente existe en las medidas tomadas por los sensores y la imperfección de los modelos empleados. Así, mediante mecanismos de inferencia basados en la información contenida en el sistema compuesto por el mapa y el robot (partiendo de un estado inicial) y un conjunto de medidas adquiridas por un sensor, se puede estimar de manera incremental un nuevo estado del sistema, permitiendo la generación en tiempo real de un modelo del entorno y de la posición estimada del robot móvil.

3.14.1 Desde EKF-SLAM a LSD-SLAM

A pesar del corto periodo de tiempo que lleva estudiándose este problema, existen distintas soluciones que han evolucionado a lo largo de los años y que introducen nuevas técnicas que mejoran la eficiencia y rapidez en la reconstrucción, reduciendo en gran medida el error acumulativo. Entre las diferentes técnicas se puede destacar:

- EKF-SLAM [50] [51]: Ha sido el método de SLAM más utilizado en la última década aunque ha sido sustituido más recientemente por FastSLAM. Utiliza un filtro de Kalman extendido (Extended Kalman Filter), por lo que se realiza una predicción de estados a través de estimaciones sobre la media y variancia de una distribución normal a partir de los puntos característicos del entorno (utilizando cámaras, láseres u otros dispositivos). La eficiencia de este método depende del nivel de incertidumbre del estado posterior, ya que si

este es muy elevado (sistemas no lineales), la precisión disminuye considerablemente.

- Fast SLAM [52]: Es un método evolucionado del EKF-SLAM que reduce el factor de incertidumbre mediante la aplicación de filtros de partículas en la asociación de datos, reduciendo el nivel de complejidad computacional a escala logarítmica. Este método ha permitido generar mapas de una escala y precisión muy elevados y ha sido aplicado satisfactoriamente en entornos dinámicos muy diferentes, ofreciendo solución incluso al problema de la detección de personas.

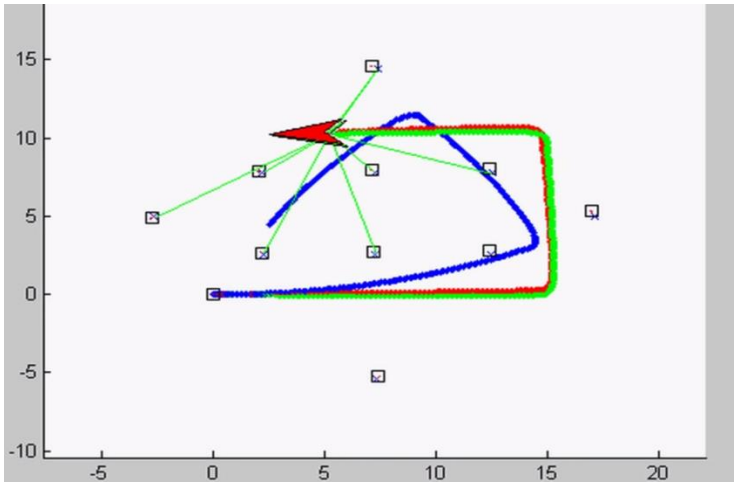

Figura 81: *Captura de un momento en la ejecución de un programa realizado en Matlab donde se aprecia el funcionamiento del algoritmo Fast SLAM.*

- Graph SLAM [53]: Se trata de otra técnica de SLAM evolucionada de las anteriores, donde se utiliza un grafo de interdependencias entre observaciones que contienen información sobre los mismos puntos característicos. Cada nodo representa la posición del robot, y los arcos entre ellos se corresponderán con las restricciones dadas por las medidas probabilísticas entre posiciones sucesivas en función del alineamiento entre las observaciones obtenida. Una de las soluciones más extendidas basadas en la técnica de Graph SLAM es Real-Time Appearance-Based Mapping (en adelante RTAB-Map) que utiliza hipótesis de detección de cierre de lazo calculadas en base a un modelo de bolsas de palabras o Bag-of-Words (en adelante BOW). Cada vez que se acepta un cierre de lazo, una nueva restricción se añade al grafo. Finalmente, un algoritmo optimizador minimiza los errores en el mapa generado.

Más adelante se explica con más detalle esta solución aplicándola a cámaras que incorporan sensor de profundidad (Kinect®), así como a cámaras con modo de visión estéreo.

- LSD-SLAM [54]: Large-Scale Direct Monocular SLAM, es una técnica de SLAM basada en visión monocular que, en lugar de utilizar funciones para la detección y extracción de puntos característicos, opera directamente sobre la intensidad de la imagen completa, buscando las diferencias entre intensidades para obtener el tracking de la cámara y una estimación de un mapa de profundidad semi-denso a través de comparaciones sucesivas de pares estéreo.

A continuación realizaremos una breve introducción al concepto de odometría visual, así como una descripción más exhaustiva de los distintos pasos a dar para la estimación de *Poses*, que incluyen la detección y extracción de características sobre la imagen, y el cálculo de odometría visual basada en matching utilizando la librería FLANN (Acrónimo en del inglés Fast Approximate Nearest Neighbor). Analizaremos con más detalle la técnica LSD-SLAM, mostrando su funcionamiento a través de varias pruebas que se han realizado con la cámara Kinect®. Finalmente, propondremos una solución que hace uso del algoritmo RTAB-Map básada en Graph-SLAM, aplicada a cámaras con sensor de profundidad Kinect® y cámaras estéreo (módulo IDS UI-1221LE).

3.14.2 Odometría Visual

El objetivo de la odometría visual es la determinación del movimiento de la cámara en un espacio de coordenadas 3D mediante el uso de secuencias de imágenes que permiten estimar la localización, orientación y rotación del robot. Normalmente, esta técnica es utilizada en conjunción con otros dispositivos tales como un Intertial Messurement Unit (en adelante IMU), que permite estimar la posición a través de la velocidad, direccionalidad y tiempo, o sitemas de geolocalización GPS La odometría visual es un problema común a todos los métodos de reconstrucción tridimensional.

Se basa en la extracción de características (*Key Features*) de los *frames* obtenidos por la cámara, y la construcción del flujo óptico por

el método de Lucas-Kanade para la estimación del movimiento de la cámara, lo cual incluye estimación de posición mediante el filtro de Kalman y minimización de la función de coste con muestras aleatorias simples.

La utilización de la odometría visual en el SLAM es muy frecuente, debido a la facilidad de integrarla en un computador sin necesitar de otros dispositivos a excepción de una cámara. No obstante, los algoritmos que implementan esta funcionalidad usualmente producen un coste computacional elevado, haciendo que en determinados casos la ejecución en entornos de tiempo real sea inviable. Además, es necesario que la cámara disponga de ciertas características que proporcionen un flujo óptico elevado, permitiendo la grabación de imágenes completas en un tiempo mínimo (global shutter).

Para paliar el inconveniente del uso elevado de recursos en la odometría visual, es recomendable la utilización de cámaras estereoscópicas. En este tipo de cámaras el procesamiento se reduce considerablemente debido a que proporcionan de forma directa información sobre la profundidad y la escala a través del par de imágenes utilizado en la producción de cada frame, reduciendo el error y mejorando la velocidad de procesamiento.

Como se explicará más adelante, el primer paso consiste en obtener un conjunto de puntos característicos utilizando métodos de cálculo sobre el escalado de la imagen, filtros de esquinas (corner filter), valores de intensidad de los puntos, rotación, etc. A continuación, se seleccionan aquellos puntos que mejor representan el conjunto de la imagen, y se aplica una técnica conocida como matching (también suele emplearse el término stereo-matching), que consiste básicamente en obtener las correspondencias entre los puntos característicos de dos imágenes estereoscópicas. Este paso requiere también la aplicación de filtros respecto a valores umbrales que descarten las correspondencias que puedan contener errores.

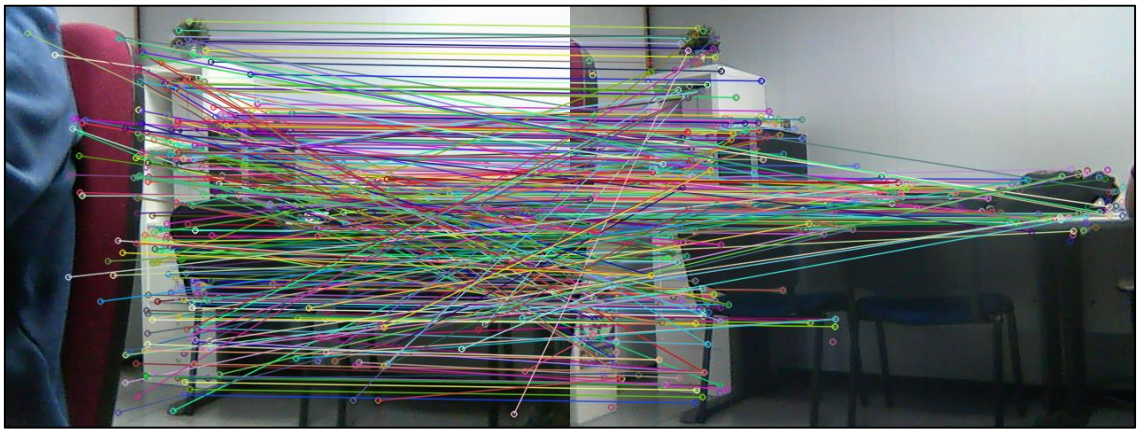

Figura 82: *Obtención del matching de puntos característicos en un par de imágenes obtenidas con dos cámaras en modo estéreo. Se puede observar la existencia de errores en algunas correspondencias.*

En esencia, la odometría visual pretende hacer un seguimiento de las correspondencias entre los puntos característicos para cada par de imágenes obtenidas, pertenecientes a una secuencia de imágenes capturadas a lo largo del tiempo. Los pequeños cambios producidos en estas correspondencias debidos al movimiento de la cámara permitirán estimar con bastante precisión su localización en el espacio tridimensional (relativo a ciertos parámetros intrínsecos como la distancia focal) y la orientación.

Dependiendo de la configuración utilizada en el proceso existirán sutiles diferencias.

Como se puede observar, con la configuración estéreo se calculan los puntos de la imagen tomada con ambas cámaras, previamente rectificadas y sin distorsión. Esto permite hacer un seguimiento de las correspondencias obtenidas a través de los puntos característicos en cada par de imágenes. Simultáneamente, se puede obtener la disparidad entre ambos puntos y obtener la profundidad en el espacio tridimensional.

El inconveniente fundamental de esta configuración reside en la pérdida de puntos característicos que no se pueden determinar debido a la oclusión y a la pérdida de coincidencias entre pares de imágenes estéreo.

Con la configuración monocular, los puntos característicos se toman cuadro a cuadro, y se siguen del cuadro previo al siguiente. Es una

forma de simular un par estéreo, ya que se toman cuadros consecutivos con una separación dada por la frecuencia a la que la cámara es capaz de capturar imágenes. Esto último se conoce como flujo óptico. En la mayor parte de los casos se podrá considerar que el flujo óptico se comporta como un par estéreo con un valor de baseline (distancia entre cámaras) muy pequeño.

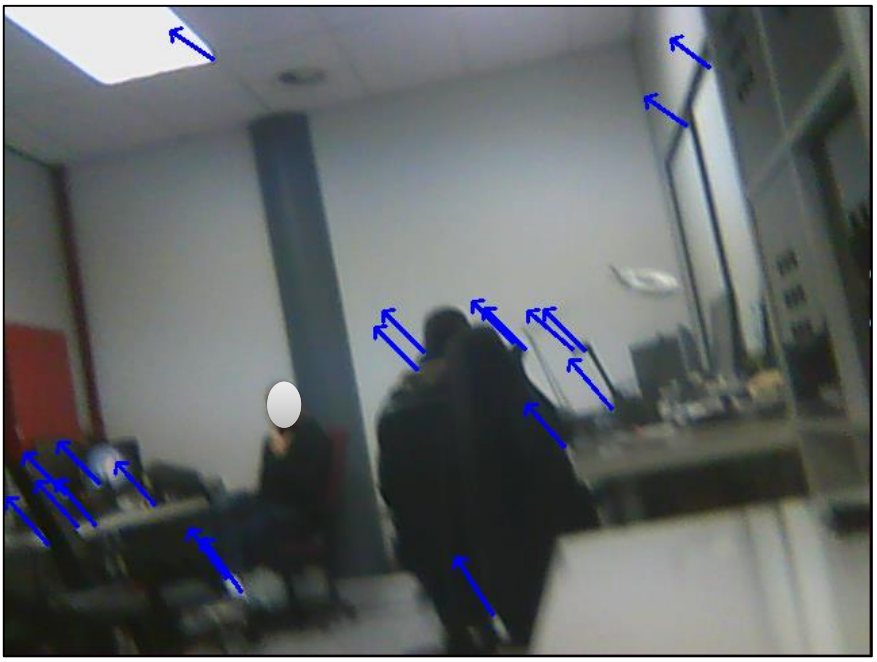

Figura 83: *Medición del flujo óptico haciendo uso de una configuración monocular para estimar la trayectoria de la cámara.*

En resumen, cuanto mejor sea la técnica utilizada para obtener la odometría visual (para lo cual es absolutamente necesario partir de una calibración de las cámaras casi perfecta) mejor será el resultado que obtengamos al tratar de solucionar el problema del SLAM, por lo que este proceso es crucial a la hora de utilizar cualquier técnica de SLAM de las ya comentadas anteriormente.

Es posible usar odometría mono o estéreo. Para cada una de estas técnicas cabría destacar:

- **Odometría monomodal:**
  Ventajas: Uso de cámaras sencillas y económicas. Desventajas: Se necesitan imágenes precisas y rangos de tiempos determinados para poder realizar el análisis. Los movimientos bruscos pueden suponer pérdida de

sincronismo entre las imágenes tomadas de forma secuencial.

- **Odometría estéreo:**
  Ventajas: Elimina la necesidad del tratamiento de imágenes secuenciales.
  Desventajas: Necesita la calibración de las cámaras y la implementación de sistemas de matching de imágenes para la obtención de mapas de disparidad precisos.

3.14.3 Detección y extracción de características de la imagen aplicadas al cálculo de la odometría visual

Como ya se ha comentado, el primer paso a realizar en cualquier técnica de odometría visual para SLAM, es el de la detección y extracción de puntos característicos del entorno, bien a través de una clasificación de los puntos escaneados por un láser observando parámetros como la dispersión, o bien aplicando técnicas de visión por computador sobre las imágenes capturadas por una o varias cámaras.

En este apartado se analizan los distintos métodos de detección y extracción de características sobre la imagen, de forma que, aplicando posteriormente diferentes métodos de estimación podamos realizar un mapeado de la escena, así como una estimación de la posición de la(s) cámara(s) en todo momento. La precisión de las estimaciones basadas en la detección de características serán determinantes para las correcciones realizadas en el cierre de lazo.

Hay que diferenciar entre dos conceptos: el de detección de características y el de descripción de características. El primero se refiere a las técnicas de procesamiento de imagen para localizar ciertas características en la imagen (usualmente conjuntos de pixeles o partes de la imagen que no varían tras diferentes transformaciones de las imágenes, u otras técnicas). Del resultado del proceso anterior se obtiene un vector de características, también conocido como descriptor de características, que aporta la información necesaria para comparar a posteriori las mismas características en todas las

imágenes y alinearlas, combinarlas, o realizar cualquier operación con ellas.

Entre los métodos más ampliamente utilizados se pueden destacar:

3.14.3.1 SIFT (Scale-Invariant Feature Transform)

El algoritmo Scale-Invariant Feature Transform [55] (en adelante SIFT) es posiblemente uno de los detectores y descriptores de características en imágenes más utilizados en aplicaciones de visión por computador. Es un algoritmo patentado y propiedad de la Universidad de British Columbia de Estados Unidos, por lo que su uso comercial queda restringido, y por lo tanto la mayoría de las aplicaciones lo incluyen bajo licencia General Public License (en adelante GPL). Utiliza un detector de características que no varían en diferentes transformaciones de la imagen tales como la escala, traslación, rotación, etc.

El algoritmo SIFT consta de 4 pasos:

1. Se realiza la identificación de aquellos objetos o puntos que son identificables desde distintos puntos de visualización. Esta operación se lleva a cabo mediante una función gaussiana de escala-espacio. Se define con la siguiente fórmula:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$ ***Ecuación 3.B.1***

Donde * es el operador de convolución, $G(x, y, \sigma)$ es una variable de escalado gaussiano y $I(x, y)$ es la imagen de entrada.

Se pueden usar varias técnicas para detectar puntos característicos estables. La diferencia gaussiana es una de esas técnicas, la cual computa la diferencia entre dos imágenes, una de ellas escalada en un factor k veces más que la otra, para así localizar el extremo escala-espacio, $D(x, y, \sigma)$.

$D(x, y, \sigma)$ se define como:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$ ***Ecuación 3.B.2***

Para localizar los mínimos y máximos locales de D(*x*, *y*, σ), cada punto se compara con los ocho vecinos más próximos en la misma escala, y con los nueve vecinos más cercanos de la horizontal con escala 1. Si el valor es el mínimo o el máximo de todos esos puntos, entonces dicho punto es un punto extremo.

2. Se realiza un filtrado de puntos, eliminando aquellos que tengan un contraste bajo o no estén suficientemente correspondidos con esquinas. Para ello se realiza el cálculo del laplaciano para cada punto encontrado en el paso 1. La localización del extremo z se consigue con la siguiente fórmula:

$$\mathbf{z} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}}$$

Ecuación 3.B.3

Si el valor de la función en z está por debajo de un umbral, entonces el punto es excluido. Con esto eliminaremos puntos extremos con contraste bajo. Para eliminar puntos con baja correspondencia hay que tener en cuenta que hay una larga curvatura a lo largo de la esquina pero una pequeña curvatura en la dirección perpendicular de la función de diferencia gaussiana. Si esta diferencia está por debajo del ratio del vector de *eigen* más alto al más bajo, desde la matriz Hessiana 2x2, el punto es rechazado.

3. Se asigna una orientación a los puntos característicos basándonos en las características de la imagen. El punto característico puede entonces ser representado de forma relativa a su orientación, consiguiendo ser invariante a la rotación. Para ello se calcula la magnitud del gradiente *m* de la siguiente forma:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

Ecuación 3.B.4

Acto seguido, se calcula la orientación θ de la siguiente manera:

$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1))/(L(x+1, y) - L(x-1, y)))$$

Ecuación 3.B.5

A continuación, se forma un histograma de orientaciones mediante la orientación de los gradientes. Se selecciona entonces el pico más alto

y algunos picos más con un valor que sea como mínimo igual al 80% del primero. Se crea un punto característico con esa orientación y se interpola su posición con el ajuste de parábola de los 3 valores del histograma más cercanos a dicho punto.

4. Los datos del gradiente también se utilizan para crear los descriptores. Los datos del gradiente se rotan para alinearlos con la orientación del punto característico y se le añade un peso gaussiano de valor 1,5 multiplicado por la escala del punto característico. Estos datos se usan para crear un conjunto de histogramas centrados en el punto característico.

   Los descriptores normalmente usan un conjunto de 16 histogramas alineados en mallas de 4x4. El resultado es un vector de características de 128 elementos.

   Los vectores resultantes se conocen como SIFT keys, los cuales son utilizados partiendo de un enfoque basado en el vecino más cercano para la identificación de posibles objetos en la imagen.

3.14.3.2 SURF (Speeded Up Robust Features)

Speeded Up Robust Reatures [56] [57] (en adelante SURF) es otro de los algoritmos de extracción de puntos característicos más populares en visión por computador. Es un detector y un descriptor de alto rendimiento de los puntos de interés de una imagen, donde se transforma la imagen en coordenadas, utilizando una técnica llamada multi-resolución. Consiste en hacer una réplica de la imagen original de forma Piramidal Gaussiana o Piramidal Laplaciana, y obtener imágenes del mismo tamaño pero con el ancho de banda reducido. De esta manera se consigue un efecto de borrosidad sobre la imagen original, llamado Scale-Space. Esta técnica asegura que los puntos de interés son invariantes en el escalado.

El algoritmo SURF está basado en el predecesor SIFT, y es distribuido bajo licencia GPL por la Universidad de Zurich.

Para el cálculo de puntos característicos, el algoritmo SURF utiliza integrales de una imagen para realizar más rápidamente el cálculo de la convolución. La imagen integral se define como:

$$I_{\Sigma}(\mathbf{x}) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i,\, j) \qquad \textbf{Ecuación 3.B.6}$$

La suma de la imagen original dentro de un rectángulo D de la imagen se puede evaluar rápidamente utilizando esta imagen integral. I (i, j) sumada sobre el área seleccionada requiere 4 evaluaciones de S (i, j) (A, B, C, D).

SURF utiliza un detector de Binary Large Object (en adelante BLOB) basado en la matriz Hessiana para encontrar puntos de interés. El determinante de la matriz Hessiana expresa la extensión de la respuesta y es una expresión de un cambio local alrededor del área. El detector se basa en la matriz Hessiana, debido a su buen desempeño en la precisión. Más precisamente, se detectan estructuras BLOB en lugares donde el factor determinante es el máximo. En contraste con el detector de Hess - Laplace para Mikolajczyk y Schmid, se basa en el determinante de la Hessiana también para la selección de escala, como se hace por Lindeberg. Dado un punto x = (x, y) en una imagen I, la matriz H Hessiana (x, σ) en x a escala σ se define de la siguiente manera:

$$\mathcal{H}(\mathbf{x},\, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x},\, \sigma) & L_{xy}(\mathbf{x},\, \sigma) \\ L_{xy}(\mathbf{x},\, \sigma) & L_{yy}(\mathbf{x},\, \sigma) \end{bmatrix} \qquad \textbf{Ecuación 3.B.7}$$

Donde $Lxx = (x,\sigma)$ es la convolución del segundo orden derivativo $\partial x/\partial x^2 g(\sigma)$ con la imagen en el punto x, y de manera similar para $Lxy = (x,\sigma)$ y $Lyy = (x,\sigma)$.

Los *box filters* de 9x9 son aproximaciones de un Gaussiano con σ = 1.2 y representa la escala más baja (resolución espacial más alta) para computarizado de los mapas de respuesta BLOB

Se denota Dxx, Dyy, Dxy. Las ponderaciones aplicadas a las regiones rectangulares son mantenidas por la eficiencia de la Unidad Central de Procesamiento (del inglés Central Processing Unit, y en adelante C.P.U.):

Se calculan las imágenes: -Dxx (x, y) a partir de I (x, y) y Gxx (x, y) -Dxy (x, y) a partir de I (x, y) y Gxy (x, y) -Dyy (x, y) a partir de I (x, y) y Gxyyx, y)

Después se genera la siguiente imagen:

$$\det(\mathcal{H}_{\text{approx}}) = D_{xx}D_{yy} - (wD_{xy})^2.$$

Ecuación 3.B.8

La ponderación relativa (w) de la respuesta del filtro es utilizado para balancear la expresión por el determinante Hessiano. Es necesario para la conservación de la energía entre los kernels Gaussianos y los kernels Gaussianos aproximados.

$$w = \frac{|L_{xy}(1.2)|_F|D_{yy}(9)|_F}{|L_{yy}(1.2)|_F|D_{xy}(9)|_F} = 0.912... \simeq 0.9,$$

Ecuación 3.B.9

La aproximación del determinante de la matriz Hessiana representa la respuesta BLOB en la imagen en la localización x. Estas respuestas son almacenadas en el mapa de respuestas BLOB sobre diferentes escalas.

Para encontrar los puntos de interés en la imagen y a lo largo de la escala, se utiliza un sistema de vecino cercano de 3x3x3. El máximo del determinante de la matriz Hessiana son los puntos interpolados en la escala y la imagen. La interpolación en el espacio-escala es muy importante, ya que la diferencia de escala entre la primera capa de cada octava es relativamente grande. En la siguiente imagen podemos ver el histograma de la detección de escalas y los puntos de interés detectados:

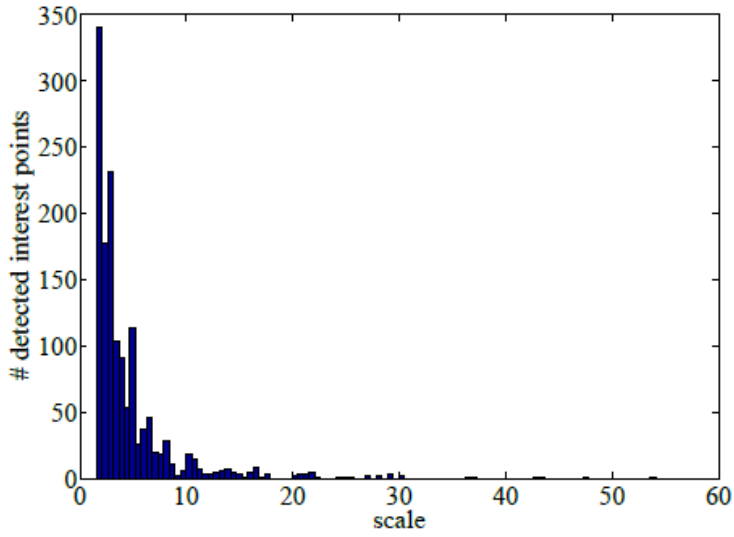

Figura 84: *Histograma de escalas.*

Para obtener los descriptores, una vez calculado el escalado, lo primero que debemos realizar es el cálculo de la orientación del punto de interés. Para obtener un punto invariante a las rotaciones, iluminación y orientación se utiliza el wavelet de Haar ***[4]*** sobre las direcciones de x e y en una región circular de radio 6s, siendo s la escala del punto de interés. Los puntos de interés de SURF tienen la característica principal de repetitividad, que quiere decir que si un punto es considerado fiable, el detector encontrará el mismo punto bajo diferentes puntos de vista (escala, orientación, rotación, etc.).

Se tiene una posición (x, y) para cada punto de interés.

Realizadas las respuestas wavelet Haar con un Gaussiano centrado en el punto de interés, las respuestas son representadas como puntos en el espacio, donde la respuesta horizontal la tenemos en la abscisa y la respuesta vertical en la ordenada.

Una vez calculados para todos los vecinos, se estima la orientación dominante calculando la suma de todos los resultados dentro de una ventana deslizante que cubre un ángulo de π/3. Las características Haar consisten en que las cejas son negras, las mejillas blancas, etc. para detectar una cara definimos características y vemos si la imagen cumple. A la imagen se le pasa una ventana y (slideing window) se

[4] *http://cnx.org/contents/fzuC-eK1@7/The-Haar-Transform (último acceso:15-septiembre-2015)*

comprueba si hay cara. Hay algoritmos que ven con escala. El algoritmo lo que hace es poner una máscara de la característica haar con la imagen, y si el patrón casa y hay coincidencias suficientes para ver que una característica en una función concreta, entonces ahí hay una cara. Lo del pi/3 es lo mismo pero para calcular la orientación de la cámara. Para unos puntos característicos, se le pasa una ventana deslizante y ve si tiene la orientación concreta. Se le pasa un tamaño de ventana pi/3. Esos tamaño de ventana.

Se suman la respuesta horizontal y vertical con la ventana. El vector más largo de las ventanas es lo que define la orientación del punto de interés. El tamaño de esta ventana debe tomarse en cuenta ya que si es demasiado pequeña, tendrá un solo gradiente dominante y si es demasiado grande, tenderá a dar tamaños de vector máximo que no representan la realidad correctamente.

Ahora, el cálculo del descriptor se realiza construyendo, primeramente, una región cuadrada centrada en el punto de interés y con un tamaño de 20s.

La región de interés se divide regularmente en 4x4 subregiones cuadradas, y para cada una de ellas se calculan unas características simples, el Wavelet de Haar para x e y, y se suavizan los resultados mediante un filtro Gaussiano (para ofrecer una mayor robustez a deformaciones, ruido y traslaciones), obteniendo dx y dy. El tamaño de este es de 2s, La verticalidad (si es horizontal o vertical) se define ahora respecto la orientación del punto de interés.

Para cada sub-región se suman los resultados dx y dy, además de calcularse su valor absoluto |dx| y |dy|. De esta manera, cada subregión proporciona un vector v, que estará compuesto por:

$$v = (\sum dx, \sum dy, \sum |dx|, \sum |dy|)$$ ***Ecuación 3.B.10***

El cual es distintivo y al mismo tiempo robusto al ruido, errores y deformaciones geométricas o fotométricas. El descriptor del SURF obtiene mediante la unión de los vectores de las subregiones.

3.14.3.3 ORB (Oriented-FAST and Rotated-BRIEF)

Oriented-FAST and Rotated BRIEF [58] (en adelante ORB) es un algoritmo de extracción de puntos característicos de una imagen que se basa en las técnicas Features from Accelerated Segment Test (en adelante FAST) y Binary Robust Independent Elementary Features (en adelante BRIEF). El método ORB está implementado en las librerías de OpenCV para su uso libre por parte de cualquier usuario. Su funcionamiento es el siguiente:

Para la primera parte, la elección de puntos característicos, el algoritmo ORB utiliza el método FAST, que es una técnica de elección de puntos característicos de una imagen, en tiempo real, que utiliza esquema piramidal para escalar la imagen y el filtro de esquinas de Harris para mejorar la elección de puntos. La elección de los puntos se realiza utilizando la intensidad del centroide y para ello, primero se deben seleccionar los momentos con la siguiente fórmula:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \qquad \textbf{Ecuación 3.B.11}$$

Usando los momentos calculados por la anterior fórmula, se puede calcular el centroide, es decir, el centro de masa, con la siguiente fórmula:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \qquad \textbf{Ecuación 3.B.12}$$

Una vez tenemos calculado el centroide, podemos calcular el vector de dirección que va desde la esquina hasta el centroide. Tendremos por tanto la orientación del punto con la siguiente fórmula:

$$\theta = \text{atan2}(m_{01}, m_{10}) \qquad \textbf{Ecuación 3.B.13}$$

De este modo, el punto es invariante a cualquier transformación de rotación. Una vez obtenido esto, comparamos el método de cálculo del centroide con dos medidas del gradiente, BIN y MAX: MAX selecciona el gradiente máximo para cada punto característico, y BIN

forma un histograma de la dirección del gradiente en un intervalo de 10 grados. Se selecciona, por tanto, el BIN de mayor valor.

Para la segunda parte, el descriptor de características, el algoritmo ORB utiliza la técnica BRIEF. El método BRIEF demuestra ser robusto frente a los cambios de intensidad de luz, a la borrosidad de la imagen, a la distorsión por perspectiva y a las variaciones por rotación del plano de la imagen. BRIEF es un método que realiza el descriptor de características usando tests binarios de entre pixeles de un marco nítido de la imagen, utilizando la siguiente fórmula:

$$\tau(\mathbf{p};\mathbf{x},\mathbf{y}) := \begin{cases} 1 & : \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 & : \mathbf{p}(\mathbf{x}) \geq \mathbf{p}(\mathbf{y}) \end{cases}, \qquad \textbf{Ecuación 3.B.14}$$

Donde p(x) es la intensidad de 'p' en el punto 'x'. La característica está definida como el siguiente vector de 'n' tests binarios:

$$f_n(\mathbf{p}) := \sum_{1 \leq i \leq n} 2^{i-1} \tau(\mathbf{p};\mathbf{x}_i,\mathbf{y}_i) \qquad \textbf{Ecuación 3.B.15}$$

De esta forma es como el algoritmo ORB selecciona los puntos característicos de una imagen. Para realizar el 'matching' entre los puntos característicos de dos imágenes, se deberá usar el algoritmo de matching FLANN.

En la figura 85 podemos ver un ejemplo de cálculo de puntos característicos en una imagen usando el algoritmo ORB.

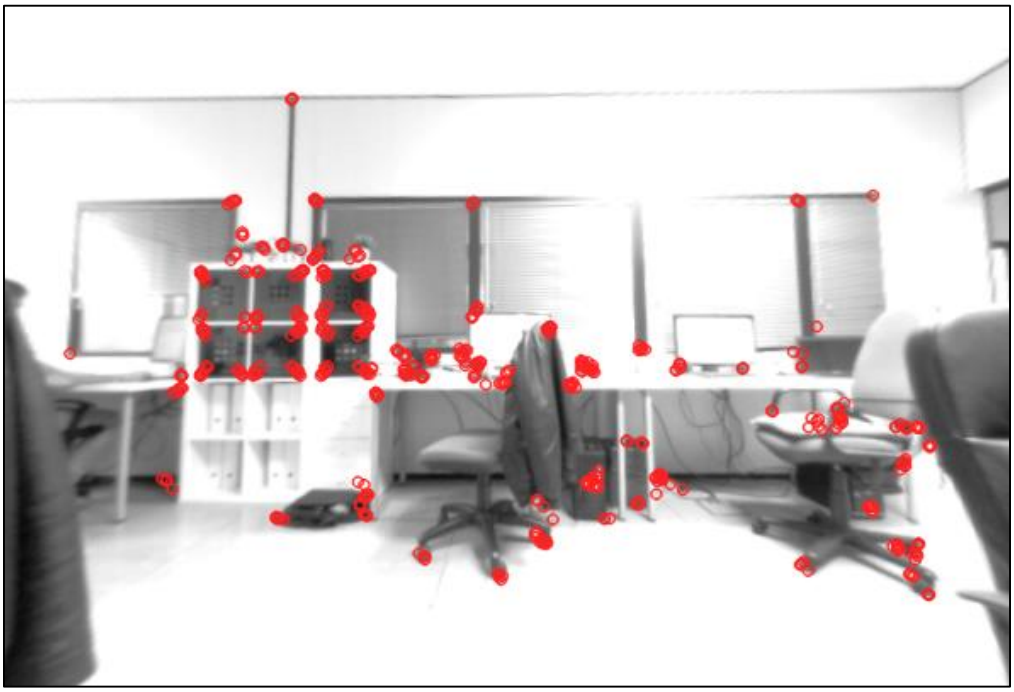

Figura 85: *Cálculo de puntos característicos con ORB*

Hasta este momento se ha explicado diferentes métodos de extracción de puntos característicos de una imagen o un par estéreo como primer paso para obtener la odometría visual. El siguiente paso sería por tanto realizar el matching entre esos puntos característicos para posteriormente poder triangular la posición de la cámara con respecto a unos puntos fijos. Para realizar el matching de puntos característicos utilizaremos la librería FLANN, que es una de las más populares y además es de código abierto.

3.14.3.4 FLANN (Fast Library for Approximate Nearest Neighbors)

FLANN [59] es una librería que implementa búsquedas rápidas para aproximación del vecino más cercano en grandes espacios multidimensionales. Definimos la búsqueda del vecino más cercano como, dado un conjunto de puntos P={p1, p2, ..., pn} en un espacio métrico M, y un punto de búsqueda q ∈ M, encontrar el elemento NN(q, P) ∈ P que es el más cercano a 'q' con respecto a la distancia métrica d: M × M → R.

$$\mathrm{NN}(q, P) = argmin_{x \in P}\ d(q, x).$$

Ecuación 3.B.16

De entre la funcionalidad que proporciona la librería FLANN será especialmente de utilidad el algoritmo de matching "FLANN Based Matcher", que se usa en conjunto con el algoritmo ORB para realizar la correspondencia entre los puntos característicos de dos imágenes. Su funcionamiento es el siguiente:

Primero el algoritmo FLANN Based Matcher halla la correspondencia entre los descriptores de dos imágenes que han sido calculados anteriormente con ORB Luego se filtrarán esas correspondencias con respecto a un valor umbral que se seleccionará dependiendo de las características de la imagen, de este modo se descartan correspondencias que puedan contener errores. Finalmente, estas correspondencias se usan para calcular qué puntos característicos de la primera imagen así como qué puntos característicos de la segunda imagen pertenecen al mismo punto del espacio 3D. Esta información es vital para el cálculo de la odometría visual.

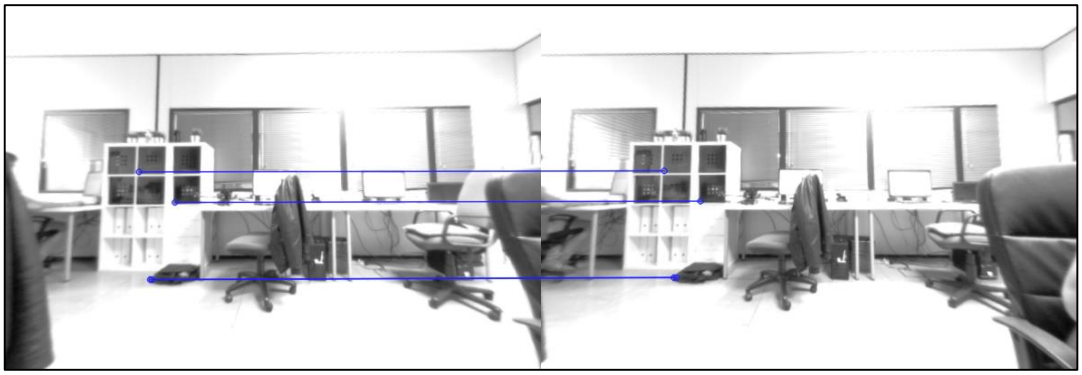

Figura 86: *Cálculo de correspondencias con FLANN*

Una vez que se sabe cómo realizar el matching de puntos característicos de dos imágenes, ya se puede triangular la posición de la cámara. Para ello seguiremos los siguientes pasos:

- Capturar dos pares de imágenes estéreo: IL(k-1), IR(k-1) y IL(k), IR(k).
- Extraer puntos característicos y realizar el matching en las imágenes IL(k-1) y IL(k).
- Triangular las coordenadas de los puntos calculados en el paso anterior utilizando la imagen estéreo.

- Calcular los vectores de rotación y traslación que definen el cambio de posición de la cámara desde el instante k-1 al instante k.

- Por último, concatenar los vectores de rotación y traslación con los de la posición anterior para obtener las coordenadas de la siguiente posición de la cámara.

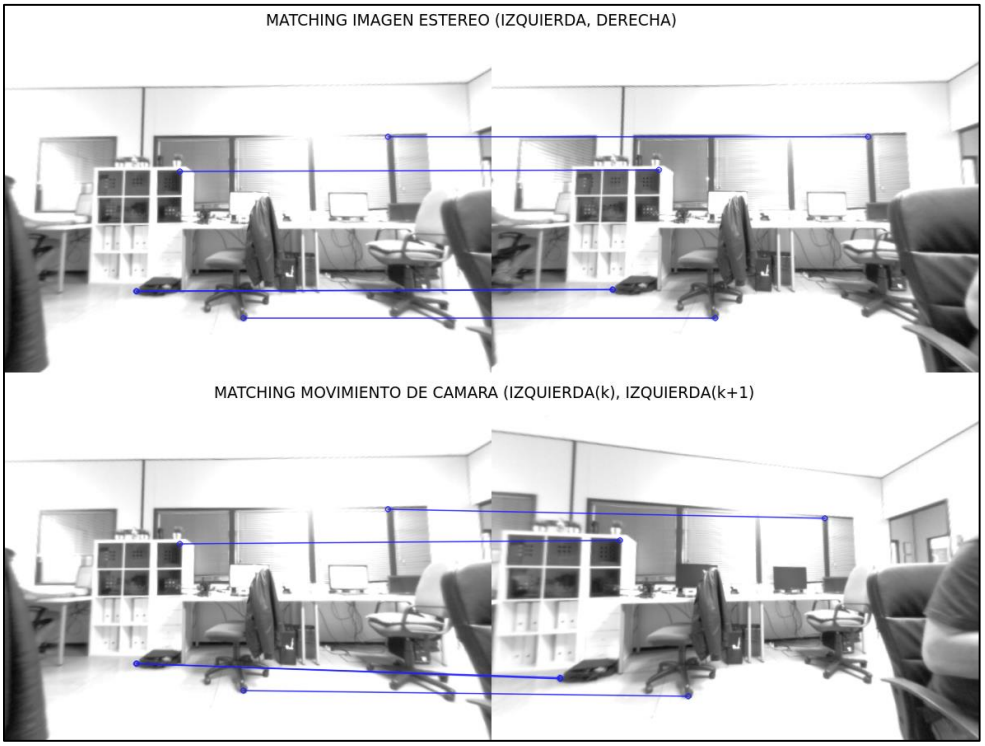

Figura 87: *Matching de características entre dos imágenes estereoscópicas y matching en dos instantes de tiempo diferentes tras el movimiento de la cámara.*

Tal y como puede apreciarse en la figura 87, se ha calculado el matching para el cambio de posición de la cámara en un instante de tiempo. Las nuevas coordenadas se obtienen en base a la concatenación de los vectores de las dos imágenes mediante triangulación.

3.15 Visual SLAM

Visual SLAM hace referencia al problema de construir un modelo de una escena 3D a través de las imágenes capturadas por una cámara, siendo capaz de almacenar la trayectoria de la cámara de manera

simultánea. En los últimos años ha habido un creciente interés en desarrollar soluciones a este problema, especialmente para vehículos aéreos no tripulados.

Principalmente, existen dos formas de implementar Visual SLAM, lo cual viene directamente determinado por el tipo de cámara a utilizar: SLAM monocular y SLAM estéreo.

SLAM estéreo hace uso de cámaras estereoscópicas que permiten la reconstrucción de mapas densos y semi-densos gracias a la estimación de la posición y orientación de la cámara (conocido como Pose Estimation), evitando la ambigüedad en la escala y el empleo de técnicas complejas en el cálculo de la profundidad. Esta estimación se obtiene fundamentalmente utilizando técnicas de odometría visual, tal y como hemos visto anteriormente. En este caso, el coste de cálculo derivado de la odometría visual se reduce significativamente, ya que al estar las imágenes rectificadas e, idealmente sin distorsión, el cálculo de la profundidad se simplifica sobremanera.

El inconveniente de este tipo de técnicas surge a raíz de dos elementos importantes: las limitaciones impuestas por el sistema estéreo que pueden afectar tanto a la precisión como a la operatividad del mismo (véase calibración, disposición limitada de las celdas del sensor Charge-Coupled Device, mejora del Campo de Visión (en inglés Field of View, y en adelante FOV), etc.), así como las oclusiones que impiden que ciertas características no se puedan determinar a través de la triangulación estéreo.

Por el contrario, las técnicas de SLAM monocular utilizan una única cámara para el cálculo de la posición, tracking (trazado y almacenaje de la trayectoria) y reconstrucción de un mapa 3D. Para ello se extrae un conjunto de características (Key Features) de la imagen, y se estima la información geométrica a lo largo de una serie de observaciones que realizan cálculos de matching basados en el flujo óptico.

El problema de estas técnicas radica en la dificultad que supone la estimación del espacio multidimensional a través de la extracción de puntos característicos (problema 5-dimensional o Random Sample Consensus de cinco puntos, en adelante RANSAC). Las subsecuentes

observaciones de los puntos característicos constituyen una progresión no lineal que va acumulando error en el cálculo de los *Poses*, lo que añadido al coste supone la no convergencia del mapa generado a lo largo del tiempo.

No obstante, el SLAM monocular suele ofrecer una mejor aproximación que otras técnicas debido a que el cálculo de los keypoints espaciales no depende del punto de observación (tal y como pasa con la visión estéreo donde las oclusiones hacen que ciertas características no sean visibles), sino de la precisión de los algoritmos de medida utilizados.

3.15.1 LSD-SLAM

LSD-SLAM (Large-Scale Direct Monocular SLAM) es una novedosa técnica que utiliza un algoritmo de cálculo de profundidad basado en el nivel de intensidad de las imágenes que utiliza toda la información de la imagen. Además, en lugar de obtener mapas de profundidad densos, usa técnicas semidensas, para aliviar el cómputo y conseguir tiempo real.

Para la estimación de la profundidad, el algoritmo pretende simular una cámara estéreo. Mediante la captura de imágenes consecutivas de forma rápida (para que la separación entre el par estéreo simulado sea pequeño) es posible obtener una imagen de disparidad. De esta forma, la profundidad es estimada por el flujo óptico, y no por la correspondencia entre puntos característicos.

El algoritmo cuenta de tres partes: seguimiento de cuadros, estimación de la profundidad y mapeo de los cuadros. Tras una fase de inicialización para calibrar la profundidad, se toma un cuadro de referencia, del que se sabe un Pose inicial estimado. Al tomar el siguiente cuadro, gracias a la odometría visual, se puede extraer el Pose del siguiente cuadro.

Como ya se ha comentado anteriormente, la captura sucesiva de cuadros se puede considerar como un par estéreo. Es aquí donde se obtiene una estimación de la profundidad, para, finalmente, plasmar en la reconstrucción tridimensional los puntos obtenidos con su respectiva profundidad.

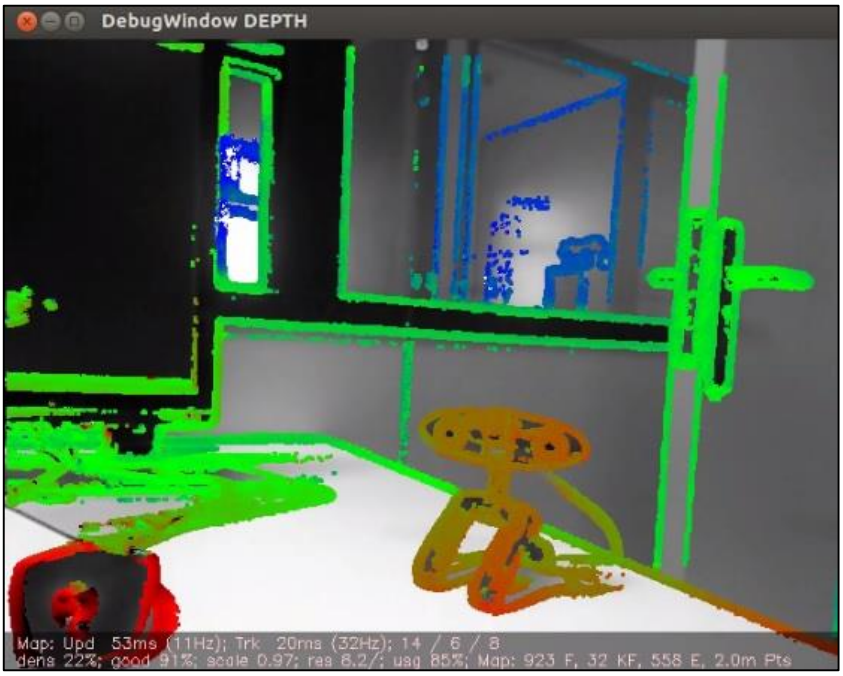

Figura 88: *Estimación de la profundidad utilizando LSD-SLAM*

Sin embargo, existen problemas con las medidas, pues se pierde la escala absoluta de las imágenes. Producirá errores en la escala de las imágenes, derivas en las medidas y problemas de cierre de lazo. Pero se introduce una forma de eliminar estos errores mediante un novedoso método de correspondencia entre sucesivas estimaciones de la profundidad.

Para el seguimiento de los Poses se utiliza una mejora del espacio euclídeo especial SE(3). La mejora consiste en aplicar el algoritmo de Gauss-Newton con pesos sobre ese espacio para obtener el espacio sim(3). Los cálculos de pasar de un espacio a otro no representan una sobrecarga adicional de cómputo. Para el cierre de lazo (loop closure) se utilizan varios posibles cuadros para comparar. Y para evitar falsos cuadros de cierre de camino se compara también de forma estadística el error. Finalmente se usa el de menor error para cerrar el camino. De esta forma, el mapa generado está formado por una serie de cuadros (con su correspondiente profundidad) y una serie de *Poses* asociados seguidos mediante sim(3). Dicho mapa está en continua mejora mediante la optimización de los *Poses* realizada en segundo plano.

Se ha experimentado en condiciones de laboratorio utilizando cámaras convencionales y la cámara Kinect®. Un ejemplo de los resultados obtenidos se pueden observar en la siguiente figura:

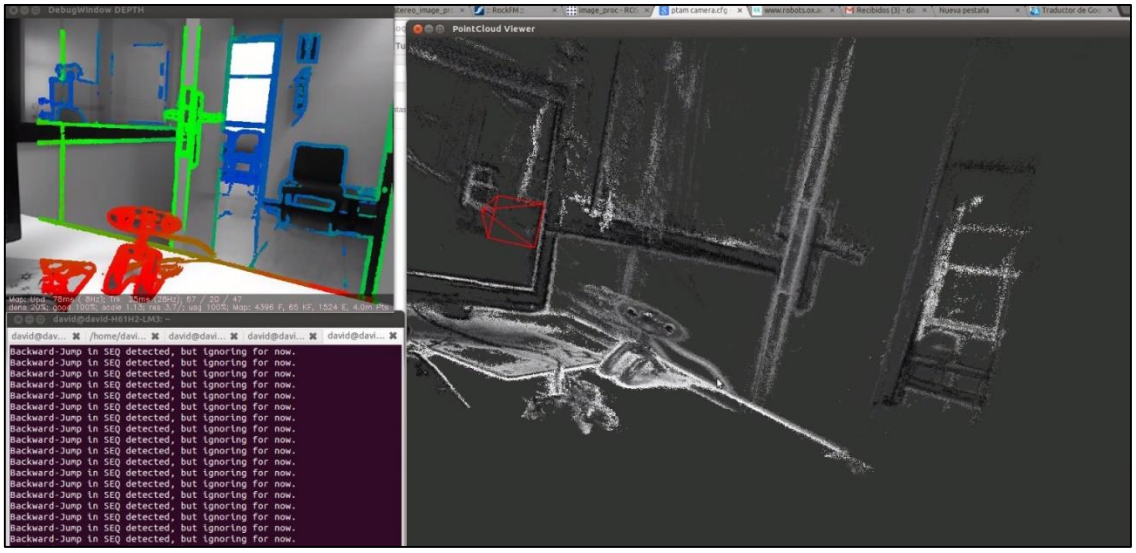

Figura 89: *Cálculo de profundidad y reconstrucción 3D con LSD-SLAM*

Dichos resultados producen un resultado muy preciso en la reconstrucción del espacio 3D tal y como se puede apreciar en la figura 89. No obstante, el nivel de procesamiento requerido es muy elevado, debido precisamente a dicha precisión. Además, el tracking de los *Poses* en los distintos cuadros calculados es muy susceptible a perderse, no dando lugar a restablecerse (cosa que no ocurre con RTAB-Map, tal y como se verá más adelante), y provocando en la mayoría de los casos la pérdida parcial o total de la reconstrucción y como consecuencia la necesidad de reiniciar la aplicación.

Para obtener buenos resultados se recomienda utilizar una cámara tipo Global Shutter con lentes que permitan un amplio FOV

Además, la cámara debe funcionar a un alto framerate (al menos 30 fps), ya que de esta forma el algoritmo de cálculo de tracking para el movimiento de la cámara permite generar más cuadros por segundo, y, por lo tanto, se evitarán pérdidas en la reconstrucción. Asimismo, pueden ajustarse parámetros tales como peso de distorsión provocado por el filtro de Kalman (KFDistWeight) que determinan como se obtienen los diferentes cuadros. Al decrementar estos valores la generación de cuadros será mayor, y por lo tanto la calidad del tracking y de la reconstrucción (mapping) será mucho mayor.

El algoritmo es bastante susceptible a perder el tracking con los giros alrededor del eje perpendicular al eje óptico de la cámara (en términos de ángulos de navegación equivaldría al Yaw). Esto quiere decir, que si se realizan rotaciones de la cámara sin realizar una traslación al mismo tiempo, el algoritmo no funcionará. Generalmente, el movimiento de izquierda a derecha es el mejor, y dependiendo también de la anchura del FOV, el movimiento hacia delante y atrás es igualmente bueno. La rotación alrededor del eje óptico (Roll) tampoco causa problemas.

En las pruebas realizadas con la cámara Global Shutter (IDS UI-1221LE) y con la cámara Kinect® se han obtenido buenas aproximaciones de los Poses, obteniendo una relación de profundidad de los elementos del entorno 3D con tasas de exactitud muy elevadas, tal y como se puede apreciar en la siguiente imagen:

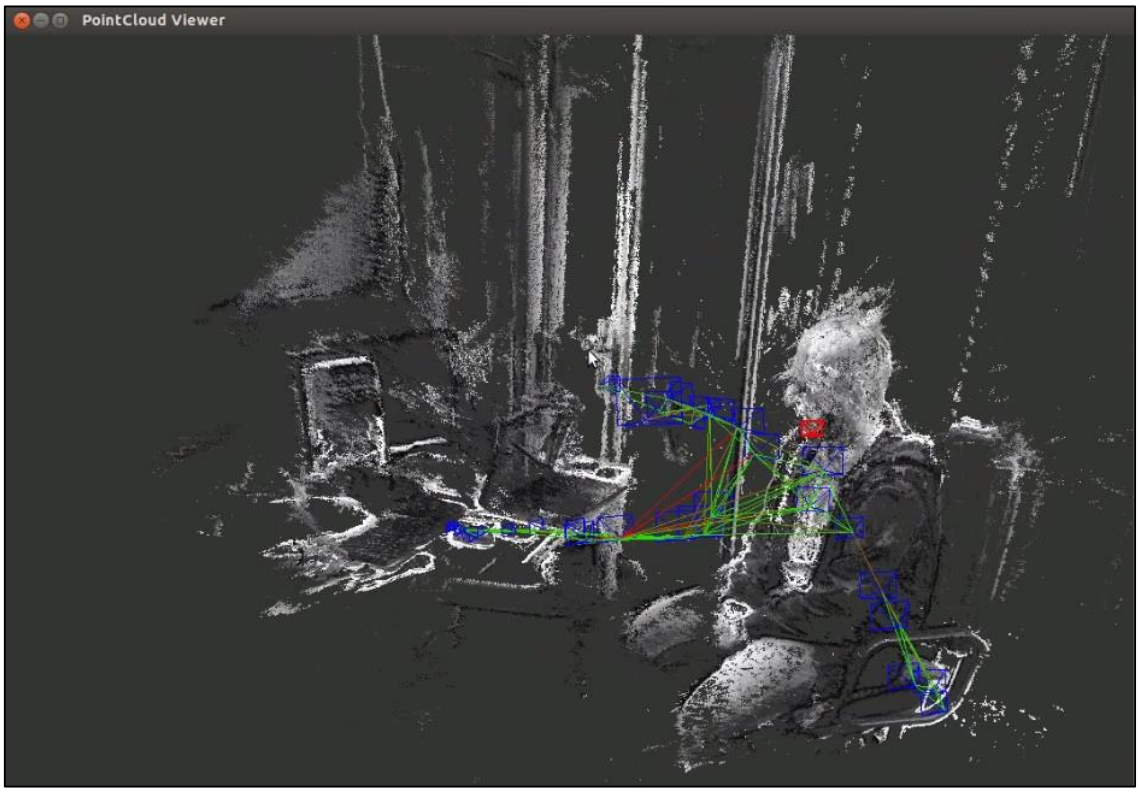

Figura 90: *Reconstrucción 3D y representación de los Poses de la cámara*

Se puede observar que los Poses calculados en base al tracking realizado en base a la información de los cuadros obtenidos por la cámara mantienen la relación de profundidad, dirección y orientación en base a los elementos del entorno observado.

3.15.2 Técnicas de SLAM estéreo

Como se ha comentado anteriormente, el SLAM estéreo (obtenido a partir de la técnica de triangulación estereoscópica) es difícilmente de implementar debido entre otras cosas a la pérdida de características de la imagen obtenida en el cálculo de disparidad. Esto puede generar rangos de distancias en los que se producen oclusiones o distorsiones en la escala, lo cual hace que la estimación de los Poses, así como la reconstrucción 3D, puedan verse afectados. No obstante, a la imagen obtenida puede aplicarse ciertas correcciones que mejoran en gran medida los problemas anteriores. Realizando un post procesamiento a la imagen de disparidad con funciones específicas para corregir el grado de correlación, el rango de disparidad o el umbral de texturizado, se puede mejorar notablemente la imagen obtenida.

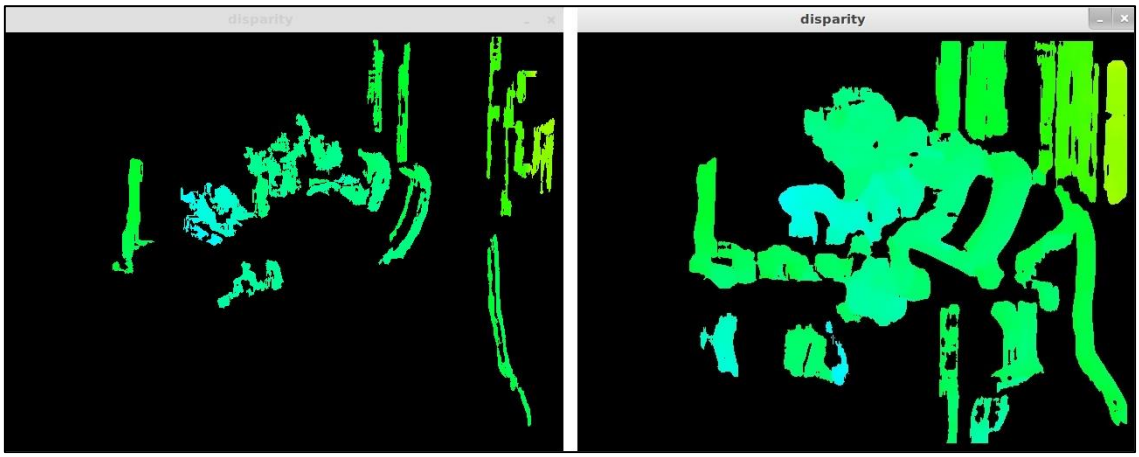

Figura 91: *Ejemplo de corrección de imagen de disparidad obtenida con técnicas de visión estéreo*

Utilizando los módulos ROS para el procesamiento de imagen, se han conseguido obtener imágenes de disparidad muy definidas que representan un rango de profundidades amplio y bastante preciso.

Para el SLAM estéreo se han estudiado algunos algoritmos basados en odometría visual adaptada a imágenes con rectificación estéreo. Uno de ellos está incluido dentro de un paquete ROS llamado Stereo SLAM.

Este algoritmo basado en la técnica Graph SLAM fue diseñado y probado en un principio para su utilización en entornos subacuáticos

para vehículos autónomos subacuáticos (en inglés Autonomous Underwater Vehicle, en adelante AUV). Una de sus principales características consiste en la utilización la librería de odometría visual libviso2 que también se encuentra como paquete independiente en ROS: viso2. El paquete viso2 dispone de varios nodos que estiman el movimiento de la cámara partiendo de imágenes rectificadas.

Por un lado, el nodo mono_odometer para visión monocular genera una matriz de correspondencias utilizando el método iterativo por consenso (RANSAC) que, aplicando mínimos cuadrados en muestras atípicas a una transformación del plano del suelo a la cámara, estima los Poses en cada iteración. Este método puede generar problemas debido por un lado, a la pérdida de tracking por el error generado en la estimación del plano del suelo y por el escalado de estos puntos, así como a la insuficiencia de características de la imagen que se puede detectar en la rotación perpendicular al eje de la cámara (al igual que ocurre con LSD-SLAM.

Por el contrario, el nodo stereo_odometer de viso2 calcula el movimiento de la cámara únicamente a través de las correspondencias extraídas de los pares estéreo. Esto hace que no existan limitaciones de cálculo de características. El resultado, como siempre en estos casos, depende totalmente de una buena calibración del sistema estéreo, evitando pérdidas de puntos por oclusión y/o escalado.

Realizando transformaciones con el paquete tf de ROS sobre el base_link (coordenadas base del robot), se puede obtener la posición relativa de las cámaras instaladas y la distancia entre ellas. A partir de esa información, el nodo stereo_odometry estima los Poses y envía a la salida una nube de puntos (point cloud) producto de las correspondencias 3D calculadas. En la siguiente imagen extraída de un ejemplo de aplicación de viso2 en un robot subacuático con dos cámaras en modo estéreo, se puede ver como gracias a stereo_odometry se puede realizar una reconstrucción 3D de la escena manteniendo la localización y orientación de las cámaras representadas por los frames de transformación (tf) stereo_forward y stereo_down.

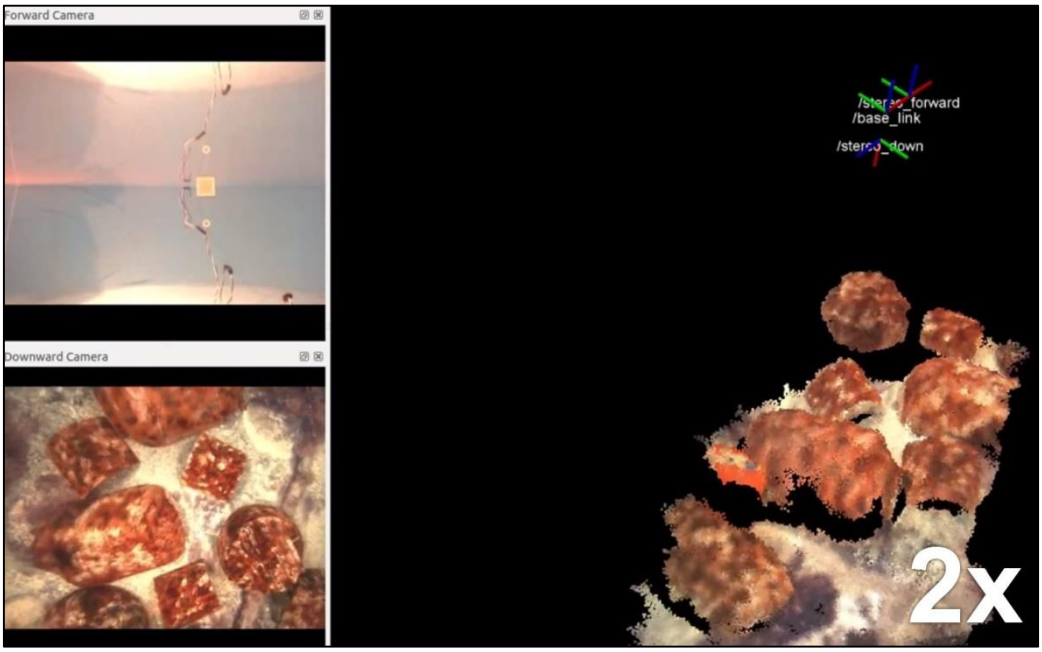

Figura 92: *Reconstrucción 3D a través de las correspondencias obtenidas por dos cámaras en modo estéreo que utiliza el algoritmo viso2 para el cálculo de la odometría visual.*

El paquete stereo_SLAM dispone de un nodo con el mismo nombre que utiliza los nodos de odometría visual del paquete viso2. Sobre los Poses calculados, se representa la trayectoria del robot sobre un gráfico de coordenadas XYZ, a la vez que se realiza la reconstrucción del mapa 3D en función de estos puntos. El nodo stereo_SLAM incorpora una librería de optimización de grafos (G2O) y una librería basada en hashing de características de la imagen (libhaloc), que implementa una solución al problema de cierre de lazo. Este algoritmo va comprobando que en la generación de la última imagen y en función de las imágenes anteriores, exista un posible cierre de lazo. Para ello, se genera un grafo en el que cada nodo representa el cálculo del Pose estimado. Para predecir un cierre de lazo, se repasan los nodos anteriores y se busca alguna coincidencia cercana con el último nodo además de otros factores basados en técnicas de probabilidades sucesivas.

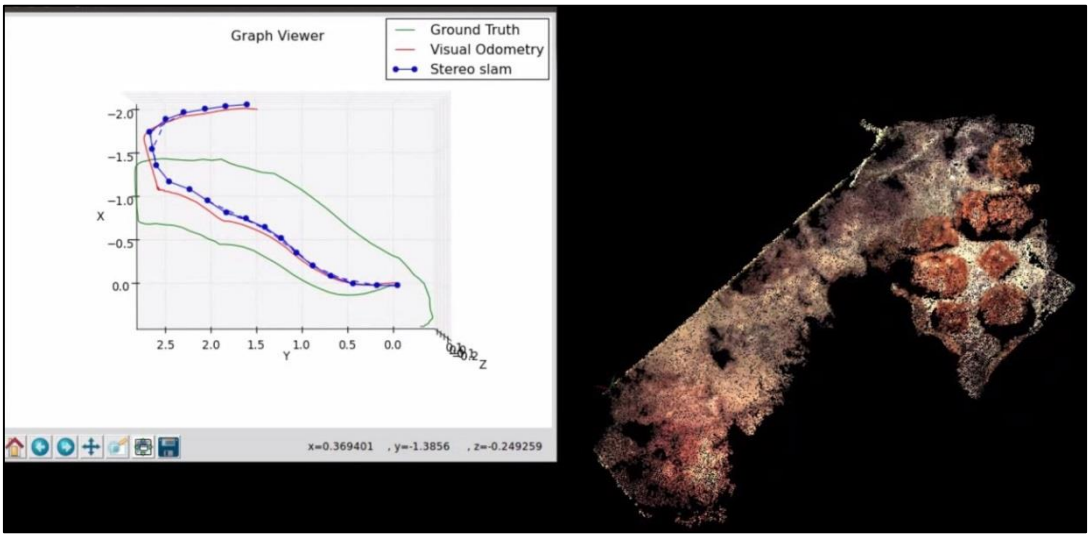

Figura 93: *Ejemplo de reconstrucción 3D junto a una gráfica de estimación de la posición haciendo uso del nodo Stereo SLAM, en la que se observan los nodos registrados para el cálculo del cierre de lazo.*

El inconveniente de este método como ya se ha dicho es que está adaptado para ser utilizado en entornos subacuáticos, y su precisión viene supeditada al uso externo de un algoritmo concreto de odometría visual, en este caso viso2. Este algoritmo tiene una licencia GPL, lo que supone restricciones para su uso comercial.

Como ya se ha explicado en secciones anteriores, otra técnica de *SLAM* Stereo basada en Graph-SLAM es RTAB-Map (Real-Time Appearance-Based Mapping). Además se mostrará una solución integrada en ROS en la que se ha utilizado una unidad de medida inercial o IMU (Inertial Measurement Unit) junto con el módulo de dos cámaras estéreo (IDS UI-1221LE) para realizar una composición de odometría visual y cálculos rotacionales basados en el giroscopio y acelerómetro que incorpora un IMU.

3.16 Path planning y recorridos óptimos a partir de un mapa del entorno

Si el sistema automatizado dispone de un mapa del entorno que lo rodea puede utilizarlo para determinar cuál es el recorrido óptimo y libre de obstáculos para alcanzar una meta o una configuración específica a la que se llamará goal (objetivo). A esta actividad se la denomina path planning y desde los años setenta ha sido objeto de investigación con óptimos resultados para la base de cualquier aplicación real de la robótica.

Para poder utilizar este mapa a la hora de generar un recorrido, es necesario que el sistema disponga de otros dos datos: una descripción geométrica del espacio que ocupa el robot, de modo que puedan verificarse correctamente posibles colisiones, y la posición de partida dentro del mapa. Si es el propio robot el que ha creado el mapa, se dispondrá implícitamente del segundo dato, pues para reconstruir el mapa de un lugar es necesario que el robot sepa dónde se encuentra.

Las trayectorias generadas por los algoritmos de planificación no tienen necesariamente en cuenta los límites cinemáticos del robot y de los operadores, sino que a menudo se trata de soluciones puramente geométricas, de ahí que tras la fase de determinación de la trayectoria suela existir una de refinamiento que tiene como objeto generar la secuencia de comandos necesarios para conducir el sistema a la configuración de goal; en este caso se habla de path planning geométrico [60]. Existen métodos que ya tienen en cuenta la dinámica del robot y generan una serie de posibles comandos para alcanzar el goal, en cuyo caso se habla de path planning orientado al control (también llamado path planning reactivo [61]).

3.16.1 Definición del problema

Desde un punto de vista matemático, el robot a pilotar y los obstáculos a evitar pueden esquematizarse en forma de figuras geométricas, bien sean polígonos (si el robot solo puede moverse en un entorno 2D o vinculado a un plano), bien poliedros (en caso de moverse en un espacio 3D).

Dado un mapa, se define W como el conjunto de todos los puntos que lo componen. Por tanto es posible definir dos subconjuntos: O (la región en la que hay obstáculos presentes) y su complementario F (espacio libre). De forma paralela también se define el conjunto C (constituido por todas las configuraciones o posiciones dentro del entorno en las que puede encontrarse el robot) y su subconjunto Cfree (que contiene todas las configuraciones correspondientes a F). Dada una configuración inicial del robot (start) y una final (goal) incluidas dentro de Cfree, calcular el recorrido entre ambas equivale a buscar la secuencia de elementos incluidos en Cfree que permite pasar del punto de partida al de llegada.

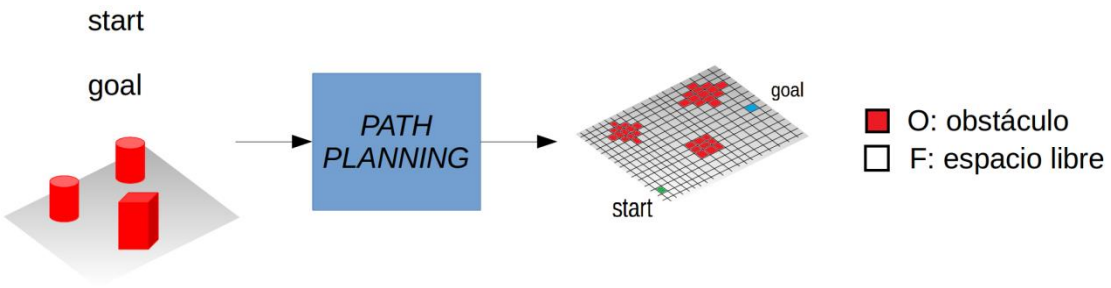

Figura 94: *Esquema representativo del espacio de configuraciones en un problema de path planning*

3.16.2 Soluciones geométricas

Estas soluciones tienen como único objetivo determinar la trayectoria para pasar de start a goal, pero no proporcionan ninguna información sobre los comandos que hay que suministrarle al robot, por tanto se habla de path planning geométrico. Para que el robot se mueva será necesario refinar después la trayectoria y convertirla en comandos efectivos que suministrarle al robot. A lo largo de los años se han propuesto numerosas soluciones a este problema, la mayor parte de ellas basadas en la creación de grafos o árboles que describen el entorno en el que se mueve el robot.

Una notable excepción a este enfoque es el uso de campos potenciales o potential fields [62]; el robot se esquematiza en forma de punto sometido a un campo de fuerza, el goal genera una fuerza de atracción y los obstáculos otra de repulsión, de manera que es posible calcular el vector fuerza que actúa sobre el robot (es decir, el esfuerzo necesario para acercarlo a la meta) para cada una de sus configuraciones. Así, no solo es posible determinar una trayectoria libre de obstáculos, también el vector fuerza que hay que aplicarle al robot para conducirlo hacia la meta en cada punto de dicha trayectoria.

La solución, por tanto, parece ideal: es sencilla de aplicar y capaz de determinar tanto la trayectoria como los comandos necesarios para alcanzar el goal. Sin embargo, presenta un grave defecto: los "puntos de mínimo" o puntos en los que la suma de las fuerzas es igual a cero; si el robot se encuentra en uno de dichos puntos, será incapaz de salir de él y por tanto se quedará inmóvil. Una solución sencilla podría ser introducir ruido al azar en los comandos, de forma que sea imposible que el robot permanezca inmóvil en un punto por tiempo

indefinido. Otro problema es que para obtener la trayectoria es necesario integrar el vector fuerza, operación computacionalmente costosa. Con frecuencia resultaría útil poder separar la trayectoria de su implementación.

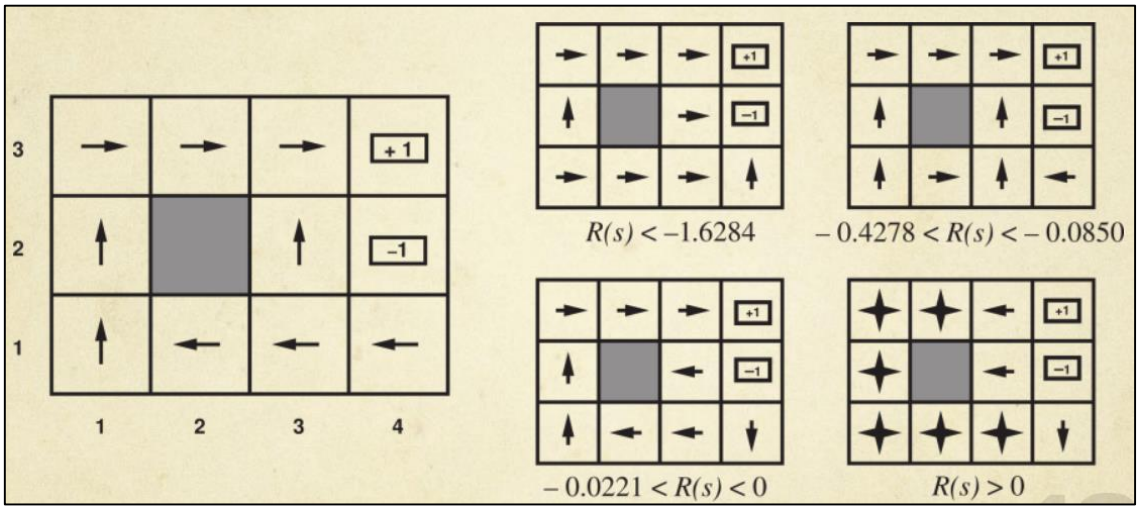

Figura 95: *Diferentes políticas utilizando un método por recompensas basado en campos potenciales.*

En la figura 95 se pueden observar diferentes políticas a seguir para, estando el robot en una configuración cualquiera, pasar a una nueva configuración que acerque o aleje al robot del objetivo en función de un valor de recompensa (en la figura, la configuración goal podría equivaler a la casilla con el valor +1). Para este caso concreto, la política óptima sería la indicada en el tablero situado más a la izquierda. Este método utiliza un algoritmo basado en un modelo de espacio de estados markoviano [63], el cual necesita gran cantidad de procesamiento para obtener la política óptima, siendo este proceso más complejo en la medida que el tamaño del mapa aumenta.

Es por ello que en la práctica suelen utilizarse algoritmos basados en el uso de grafos, los cuales solucionan el problema en dos pasos:

1) Se crea un grafo que representa una discretización de todas las configuraciones posibles en el espacio en el que se mueve el robot.

2) Se conectan el start y el goal a la estructura generada, la cual denominaremos roadmap.

De esta forma se simplifica notablemente el problema de planning, que se convierte en un problema de enrutamiento sobre un grafo muy estudiado y resuelto de forma óptima.

A continuación se describen distintos algoritmos de generación de grafos para path planning.

3.16.3 División del mapa en celdas

Se subdivide en celdas todo el mapa (poligonales si el mapa es bidimensional y poliédricas si es tridimensional) y se crea un grafo que representa el espacio libre, tomando como nodos los centros de las celdas libres y conectando entre sí únicamente las celdas adyacentes. A continuación solo resta determinar en qué celda se encuentran el start y el goal para obtener un roadmap consistente. Este método permite alcanzar en un tiempo finito la solución al problema (si existe) o determinar con seguridad que no existe, por tanto el algoritmo es completo. Sin embargo, esta solución requiere analizar exhaustivamente todo el mapa, lo cual supone en muchos casos costes computacionales elevados.

Existen muchos algoritmos para subdividir el mapa en celdas:

1. **Celdas homogéneas**: todo el mapa se subdivide en celdas de igual dimensión, la resolución del mapa es única y se establece a priori, al igual que las dimensiones del grafo resultante.

2. **Celdas poligonales**: los obstáculos se esquematizan en forma de polígonos o de poliedros y a partir de cada vértice se trazan líneas rectas paralelas a uno de los ejes (a dos si es 3D) hasta dar con un obstáculo o con el borde del mapa. Como nodos se utilizan un punto interno dentro de cada celda y uno en las rectas; el punto interno se unirá mediante ramas a aquellos que se encuentran en las líneas que delimitan su celda. De esta forma se obtiene una subdivisión óptima del mapa en celdas que permite minimizar las dimensiones del grafo, pero los costes computacionales de la fase de creación son muy altos.

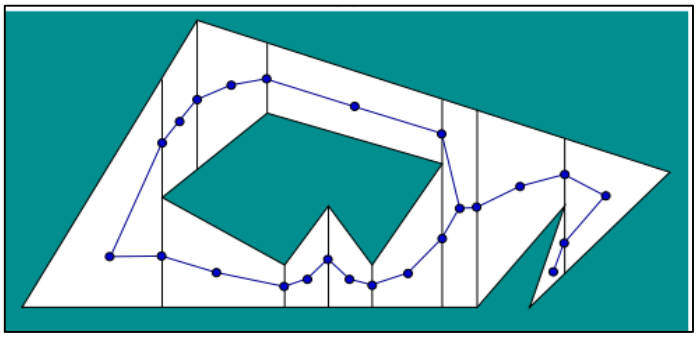

Figura 96: *División de un mapa en celdas poligonales para la obtención de un grafo de recorrido.*

3. QuadTree/Octree: el mapa se subdivide recursivamente en cuadrantes (si es 2D) o en cubos (si es 3D) de modo que sea posible representarlo mediante un árbol cuaternario u octal, respectivamente; la recursión se interrumpe en función de la precisión deseada para esa área determinada. De esta forma la dimensión de las celdas no es fija, sino que hay que adaptarse dinámicamente a la zona a representar; intuitivamente, las celdas cercanas a los obstáculos serán más pequeñas y las de la zona libre serán más grandes.

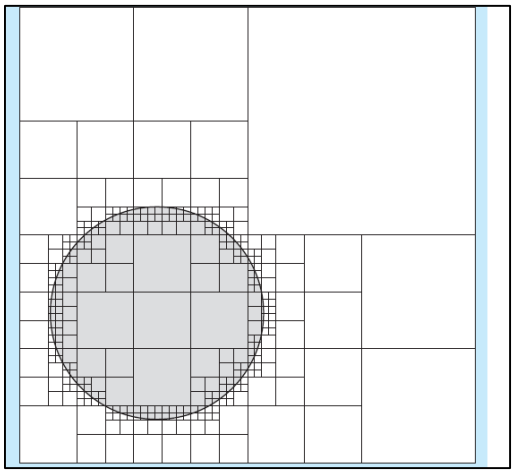

Figura 97: *División de un mapa utilizando Octrees.*

4. Diagrama de Voronio: se subdivide el mapa en áreas, cada una de las cuales está identificada por un punto característico denominado "centro". Para que un punto pertenezca a una casilla debe estar más cerca de su centro que de cualquier otro punto del conjunto de centros. Colocando los centros en el interior de los obstáculos, se obtiene un roadmap óptimo, en el que los nodos son los vértices de las celdas y los arcos son los bordes. Este roadmap es

óptimo porque, por definición, los trayectos obtenidos a partir de él se encuentran a la máxima distancia posible y permiten por tanto pasar a una distancia de seguridad de los obstáculos. La fase de definición de las celdas, sin embargo, es larga y costosa.

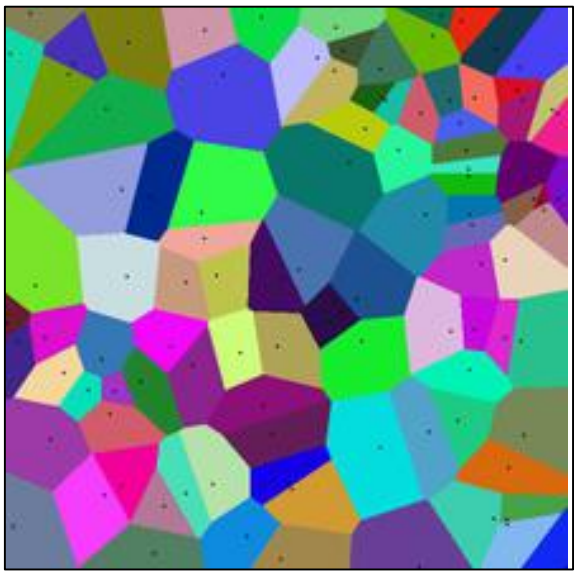

Figura 98: *Delimitación de celdas con un diagrama de Voronio. Si los centros representan los obstáculos, cualquier camino a lo largo de los bordes de las áreas definidas es óptimo.*

5. Grafo de visibilidad: los obstáculos se esquematizan en forma de polígonos o de poliedros; se considera cada vértice un nodo del grafo y se conecta a todos los vértices "visibles" desde su posición, es decir, a todos los vértices a los que está unido por una esquina o hasta los cuales es posible trazar una línea recta de unión que no tropiece con ningún obstáculo. El resultado es un roadmap que permite calcular los trayectos más breves posibles para evitar los obstáculos, pero que resulta difícil de usar porque pasan muy cerca de ellos.

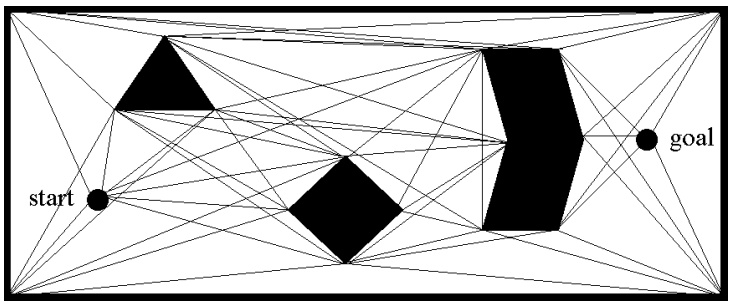

Figura 99: *Delimitación de caminos en un mapa mediante un grafo de visibilidad.*

3.16.4 Probabilistic Roadmap (PRM)

Representa una alternativa a la creación de grafos respecto a los enfoques combinatorios basados en la división en celdas, que son eficaces pero requieren una potencia computacional alta porque analizan siempre el mapa completo, incluso cuando bastaría con disponer de un conocimiento mucho más somero del entorno. De ahí la importancia de los enfoques probabilísticos para resolver este problema: los nodos del grafo no se eligen siguiendo un criterio específico, sino que se seleccionan al azar de entre las posibles configuraciones del robot incluidas dentro de Cfree; la parte computacionalmente más compleja pasa a ser la creación de las conexiones dentro del grafo. Cada vez que se añade un nodo, se conecta con todos los nodos accesibles mediante un trayecto recto y libre de obstáculos existentes dentro de un radio n (donde n se define a priori) cuyo centro es el nodo. Un enfoque alternativo consiste en conectar el nodo con los k nodos más cercanos, independientemente de la distancia a la que se encuentren.

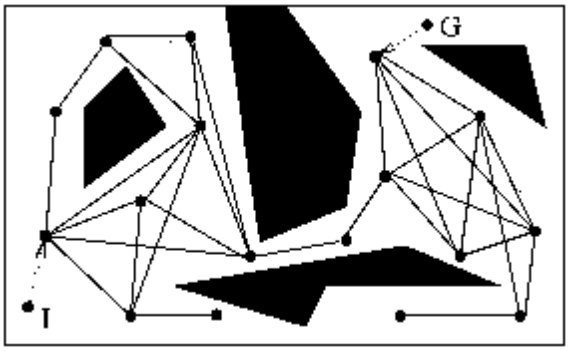

Figura 100: *Representación de un grafo de roadmap basado en un enfoque probabilístico.*

Cuando el roadmap se considera suficientemente denso, dejan de añadirse nodos; a partir de ese momento comienza el uso propiamente dicho. Para crear un trayecto solo hay que añadir las configuraciones de start y goal al roadmap, conectarlas con los nodos más cercanos y usar el grafo para obtener el trayecto.

Esta solución no es completa, ya que no es posible por un lado determinar con seguridad si alcanzar el goal es posible, ni tampoco si se va a encontrar siempre una solución en tiempo finito, ya que el algoritmo de búsqueda incluye un componente de azar. Sin embargo,

es "probabilísticamente completa", es decir, la probabilidad de no resolver un problema es tan baja que es una de la soluciones más empleadas en la práctica, ya que, además, presenta otra ventaja: una vez creado un roadmap puede usarse para muchas queries o peticiones de movimiento del robot sobre el camino trazado, amortizando así el coste de la fase de creación.

Existen variantes optimizadas de este algoritmo [64]:

1) PRM* (el asterisco indica 'óptimo'): el valor del parámetro *n*, definido en el algoritmo estándar, varía dinámicamente, disminuyendo intuitivamente al aumentar las dimensiones del mapa.

2) LazyPRM: funciona como el PRM normal, pero la ausencia de obstáculos en el camino que une el start y el goal solo se verifica cuando es necesario. La fase de creación del mapa es más rápida, pero la query individual es más lenta [65].

3.16.5 Rapidly-exploring Random trees (RRT)

La exploración rápida y aleatoria de árboles (RRT) es de otro enfoque de creación de grafos en el que se utiliza un árbol en lugar de un grafo. La raíz del árbol suele coincidir con la configuración de start y en cada iteración se elige una configuración al azar: si forma parte de Cfree, se busca el nodo más cercano y, si es posible trazar un trayecto recto libre de obstáculos desde el nodo a dicha configuración, se une ésta al árbol; en caso contrario, se descarta. Tras un número predeterminado de iteraciones se intenta conectar la configuración de goal al árbol. Si se consigue, el problema está resuelto y basta aplicar un algoritmo de búsqueda sobre los árboles para obtener el camino; en caso contrario, la expansión del árbol continúa. Lo que caracteriza a esta técnica es la exploración, es decir, la capacidad de tornar accesibles regiones del mapa a gran velocidad: se ha demostrado que el algoritmo es capaz de cubrir buena parte del mapa en poquísimas iteraciones (del orden de centenas) [66].

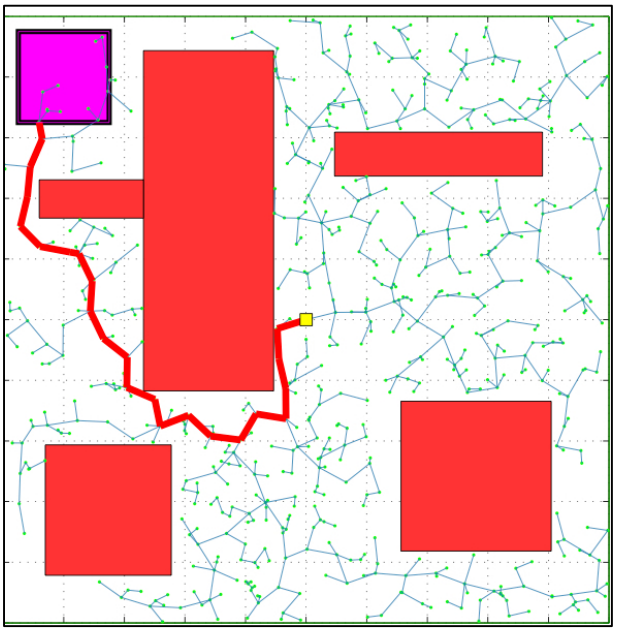

Figura 101: *Exploración en un mapa del camino hasta el objetivo (goal) a través de la expansión de un árbol de ramificación aleatoria (random tree).*

A diferencia de los roadmaps, en este caso la estructura de datos creada es single-query, es decir, cada vez que se realiza una petición de planning se crea una nueva. Esto puede parecer una limitación, pero el coste computacional de la creación del árbol de exploración es mucho menor que el de un roadmap, ya que el número de nodos y sobre todo el número de ramas es inferior y además la estructura no es cíclica.

El elemento fundamental en este método es el local planner, cuya labor consiste en unir los nodos y crear las ramas. Si éste es capaz no solo de ver si los nodos se pueden unir mediante trayectos libres de obstáculos sino también qué comandos hay que suministrarle al robot para que vaya del primero al segundo, es posible realizar un algoritmo que también tenga en cuenta las dinámicas del sistema a coste cero. En ese caso, solo se unirían los nodos efectivamente accesibles para el robot desde la configuración de partida.

Existen algunas variantes de este algoritmo:

1) RRT-connect: incluye la expansión simultánea de dos árboles, uno con raíces en el start y otro con raíces en el goal. La expansión es alterna y tras un cierto número de

iteraciones se intenta conectar ambas estructuras. Cuando esto se consigue el problema queda resuelto [67].

2) RRG (rapidly exploring random graph): incluye la posibilidad de ciclos en la estructura. Es una vía intermedia entre el RRT y el roadmap. La expansión tiene lugar como en el algoritmo base, pero cada vez que se añade un nodo se buscan a su alrededor posibles nodos con los que conectarlo dentro de un radio predeterminado. La estructura se convierte por tanto en un grafo, pero el coste de creación sigue siendo inferior al del roadmap [68].

3) Optimal RRT (RRT*): funciona como el RRG, pero se evitan los ciclos eliminando las ramas redundantes, es decir, las que no forman parte del camino más corto desde las raíces del árbol hasta un vértice. Finalmente se obtiene un árbol optimizado. De esta forma se conjugan la eficiencia a la hora de determinar el trayecto de los grafos y la practicidad de la gestión de los árboles [69].

3.16.6 Árboles de búsqueda por expansión

Una última alternativa a la solución geométrica del problema consiste en un enfoque basado en árboles de búsqueda por expansión (en inglés Expansion Search Tree (EST)). La expansión no se produce totalmente al azar como ocurría en el método RRT, sino que está guiada para que el árbol solo se expanda a áreas que sabemos que son accesibles. Se utiliza una cuadrícula de celdas para discretizar Cfree y tener en cuenta las configuraciones que ya han sido exploradas (aquellas a las que se puede acceder desde las raíces siguiendo uno de los caminos dentro del árbol). Los nodos por los que expandir el árbol se toman de las celdas libres próximas al borde de la región de la cuadrícula que ya ha sido explorada. Se garantiza que en cada iteración el árbol se expanda a regiones que antes eran inaccesibles, pero la función de selección del punto con el que expandir el árbol se vuelve más compleja.

Este algoritmo también incluye la creación simultánea de dos árboles, uno con raíces en *start* y otro con raíces en *goal*. Existen variantes que no verifican la validez del trayecto hasta que no es necesario, por ejemplo: Single-query Bidirectional Lazy collision checking [SBL]. En este algoritmo, en la fase de expansión los nodos solo se conectan en función de su cercanía relativa; una vez que se ha creado un trayecto entre *start* y *goal*, se comprueban todas las ramas para verificar posibles colisiones, si no las hay, el problema está resuelto, en caso contrario, se eliminan las ramas inválidas y se procede a una nueva expansión [70].

3.16.7 Algoritmos de búsqueda sobre árboles y grafos

Una vez obtenido un esquema del entorno que comprenda también las configuraciones de start y de goal, entran en juego los algoritmos de búsqueda sobre grafos o árboles, que han sido objeto de investigación durante muchos años, pues resultan fundamentales para muchísimas aplicaciones informáticas, tanto que se han conseguido estándares de eficiencia óptimos.

El algoritmo que se suele utilizar en las distintas aplicaciones es A* [71]. Se trata de un algoritmo de búsqueda informada que consiste en visitar los nodos del árbol o del grafo en función del coste que supone llegar hasta ellos. Además de la función coste (representada por *g(n)*, donde n es el nodo actual), la decisión de explorar por un camino u otro viene determinada por una función heurística (*h(n)*) formulada en base al mínimo coste que resulta de ir desde el nodo n hasta el nodo objetivo (goal). Por tanto, el coste para llegar desde cualquier nodo al objetivo vendría reflejado por la función *f(n):*

$$f(n) = g(n) + h(n) \qquad \textbf{Ecuación 1}$$

De esta forma, la decisión de expandir la búsqueda por un camino u otro estará determinada por el menor valor de *f* que se pueda encontrar.

Se mantienen constantemente dos listas para cada iteración, una de nodos visitados y otra de nodos por visitar. Cuando se visita un nodo, se añade a la lista de visitados y se expande la otra lista con los nuevos nodos accesibles, se calcula la función *f* de mínimo coste para

los nuevos elementos y se escoge con qué nodo seguir la expansión en función de estos. Así, este algoritmo es capaz de hallar el camino de menor coste al objetivo siempre y cuando exista, es decir, se trata de un algoritmo óptimo, ya que si la solución existe el algoritmo la encuentra y además la solución será la de menor coste.

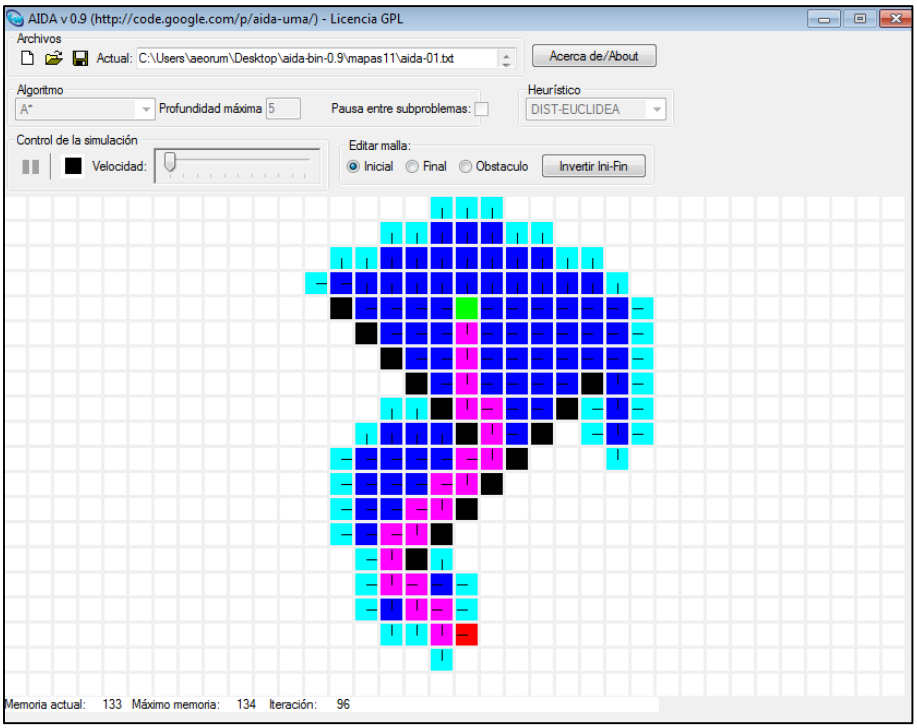

Figura 102: *Ejemplo de ejecución de algoritmo A* utilizando como heurístico la distancia euclídea.*

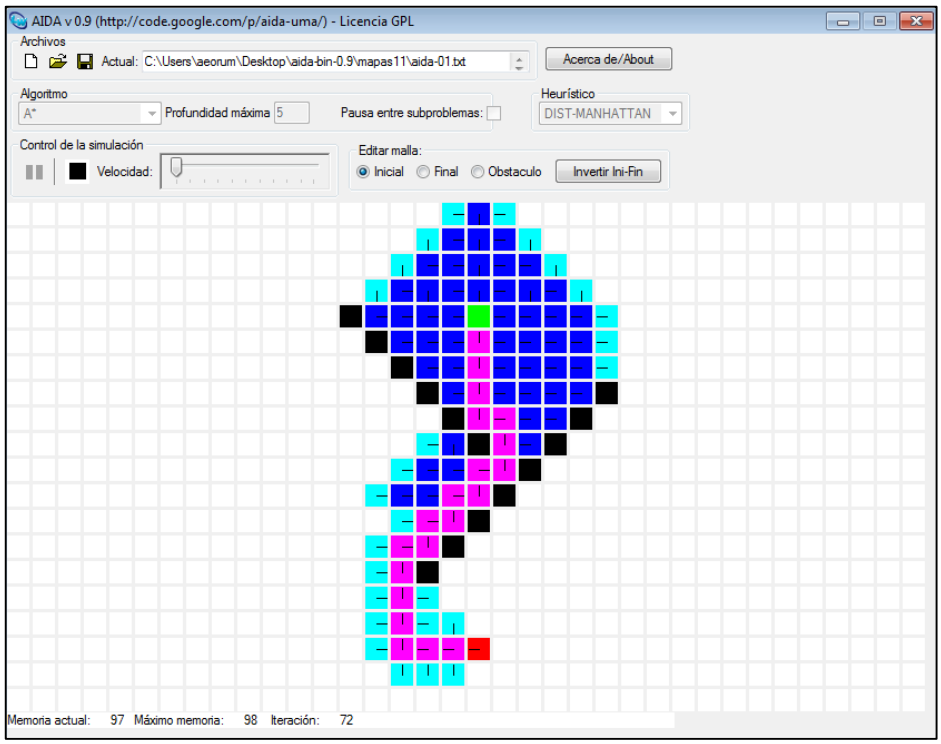

Figura 103: *Ejemplo de ejecución de algoritmo A* utilizando como heurístico la distancia manhattan.*

En las figuras anteriores se pueden observar dos ejemplos de ejecución del algoritmo A* sobre un mapa 2D. Las celdas blancas representan los espacios libres, las negras son los obstáculos, la celda verde es el start y la roja el goal. Al comenzar el algoritmo, se va explorando el mapa en función del mínimo coste calculado en cada una de las configuraciones. Las celdas exploradas se representan con el color azul más oscuro. En cambio, las celdas aun por explorar se representan por el color azul más claro. Finalmente, y una vez alcanzado el objetivo, el camino es señalizado con el color magenta.

De las dos imágenes anteriores se puede concluir que la función heurística elegida es determinante a la hora de minimizar el coste computacional y el tiempo que tarda el algoritmo en obtener la solución.

3.16.8 Soluciones basadas en el control

Un enfoque diferente es el de modelar los vínculos cinemáticos y dinámicos característicos del robot controlado ya en fase de path planning. De esta forma, es posible obtener una trayectoria que el robot pueda seguir de forma efectiva y los comandos necesarios para

seguirla con un solo algoritmo, evitando reelaboraciones posteriores. El peso computacional necesario para resolver cada query individual sin embargo aumenta. Además, es necesario poder modelar con precisión las características dinámicas del sistema.

Intuitivamente es necesario discretizar de alguna forma los inputs que pueden enviarse al robot desde el exterior. El enfoque más sencillo para hacerlo es establecer una unidad mínima de tiempo t, aplicar todas las posibles combinaciones de input para un tiempo t partiendo de la configuración de start, calcular todas las configuraciones accesibles dentro de Cfree y repetir el procedimiento para cada estado al que se acceda. La estructura así obtenida será por tanto un árbol con raíces en el start y en cuyas ramas se guardarán los comandos necesarios para pasar de un nodo al siguiente. Este procedimiento se repetirá para cada nuevo nodo obtenido hasta alcanzar la configuración de goal.

El algoritmo Kinodinamic Panning by Interior-Exterior Cell Exploration (en adelante KPIECE) [47] es una versión optimizada de este procedimiento. A diferencia del exhaustivo enfoque estándar, tiene como objetivo dirigir la expansión del árbol de modo que con cada iteración se alcancen áreas que no habían sido visitadas antes. Para ello, se representa el espacio de los estados mediante una cuadrícula de celdas multinivel y multirresolución que tiene en cuenta el espacio que ya ha sido explorado. Ahora no se hace hincapié en el espacio físico mapeado, sino en los estados que el robot puede alcanzar dentro de dicho espacio.

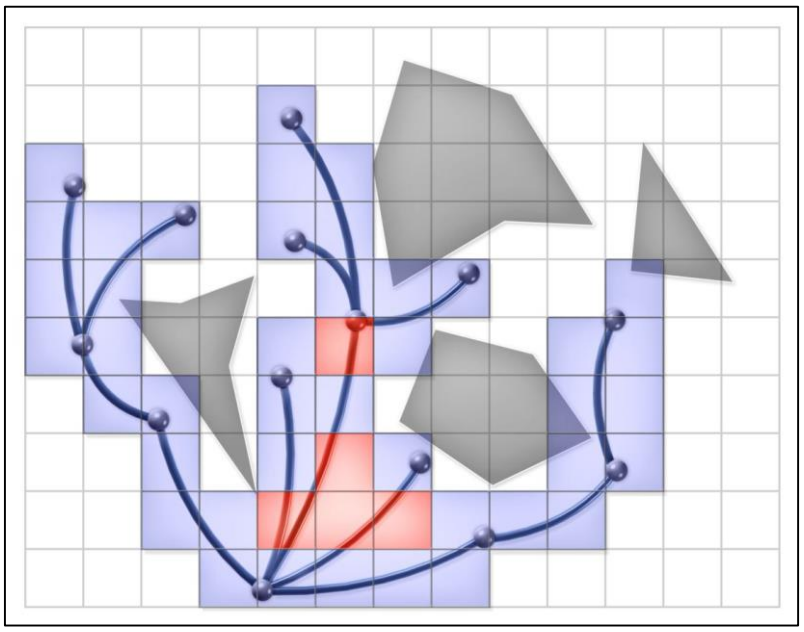

Figura 104: *Funcionamiento del algoritmo KPIECE donde se aprecia el espacio explorado (celdas azules), y celdas de partida (rojas) calculadas en base a las distintas resoluciones.*

Esto es posible gracias a la diferenciación entre celdas externas e internas. Cuando se llama al algoritmo se inicia la cuadrícula, pero no las celdas, que solo se crean cuando una rama del árbol las visita durante una fase de expansión. Se denomina celda interna aquella con *2*n* celdas inicializadas contiguas y con *n* dimensiones del espacio de estados. Se denomina celda externa a todas las demás.

El punto a partir del que se expande el árbol no se elige al azar, sino que se establece a través de un valor asociado a cada celda para así dirigirla hacia celdas externas como punto de partida. De esta forma, el árbol puede cubrir el espacio disponible rápidamente. Las distintas resoluciones de la cuadrícula permiten acelerar la elección de la celda de la que partir, es decir, primero se comparan las celdas de nivel más alto, las más grandes; después, una vez se encuentra la de mayor valor, se repite el procedimiento recursivamente en su interior hasta llegar a la mínima resolución posible; se disminuye así el número de comparaciones respecto a las que serían necesarias si se usara directamente la cuadrícula con las celdas más pequeñas.

Se ha demostrado que este algoritmo es capaz de producir óptimos resultados y alcanzar el goal en un tiempo razonable con una estructura de datos muy ligera en comparación con las obtenidas mediante otros algoritmos. Aunque se presenta como un algoritmo

basado en el control, es fácil usarlo en el planning geométrico simple tomando R^3 como espacio de estados y todas las posibles roto-traslaciones en tres dimensiones como posibles comandos.

CAPÍTULO 4: PROPUESTA Y DESARROLLO

En función de las pautas establecidas en el capítulo 2, se ha realizado el análisis y validación de distintos métodos para la detección automatizada de obstáculos y su mapeo. El método seleccionado para el desarrollo de la presente tesis es el basado en el análisis de imágenes por medio de cámaras de espectro visible y su tratamiento automatizado. Este análisis se realiza mediante dos métodos fundamentales:

- Haciendo uso de una única cámara y un conjunto de fotogramas consecutivos. Esta aproximación permite la detección de obstáculos basado en las diferencias relativas entre los distintos fotogramas.
- Usando cámaras estéreo y realizando el cálculo de la disparidad entre las imágenes captadas.

El estado actual de la técnica establece las pautas básicas para realizar el análisis estéreo, pero el cálculo efectivo de la disparidad resulta en una tarea compleja y muy dependiente del sistema de cámaras elegido. En esta tesis se plantea el uso de un sistema normalizado para la toma de medidas y su procesamiento. Para ello, la toma de datos se realizará por medio de dos cámaras dispuestas en un bastidor y calibradas para maximizar la distancia máxima de análisis. Adicionalmente, se incorpora un sistema hardware para la toma de medidas de forma coherente entre dichas cámaras aumentando la capacidad de proceso del sistema en su conjunto.

Las cámaras utilizadas en el presente desarrollo corresponden al modelo UI-1221LE-M-GL del fabricante IDS, cuya principal característica es la inclusión de un sensor global shutter tal y como se analizó en el apartado 2.4.3.

En este contexto, se ha realizado el desarrollo HW y SW de la solución para la calibración y toma de imágenes coherentes por sincronización basada en interfaz hardware.

4.1 Solución RTAB-Map integrada en ROS utilizando módulo estéreo con IMU

4.1.1 Módulo integrado

Para realizar las pruebas que se tratarán más adelante, el módulo módulo estéreo con dos cámaras IDS UI-1221LE, ha sido ampliado añadiéndose una unidad de medida inercial o IMU conectada a una placa Arduino de la siguiente forma:

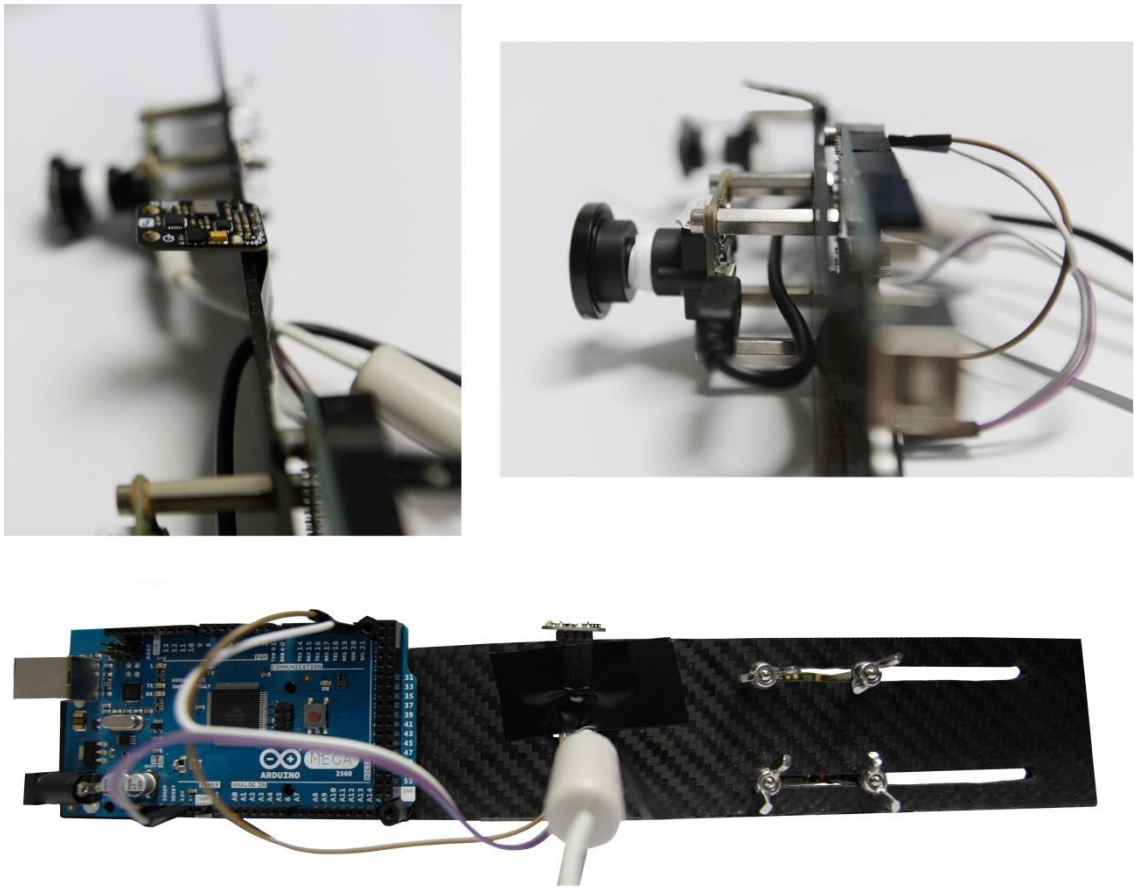

Figura 105: *Módulo estéreo IDS UI-1221LE con unidad de medida inercial (IMU) integrada.*

Se trata de un módulo con varios sensores basado en el módulo integrado Sparkfun Razor IMU 9DOF, compuesto por un sensor IMU de 9 grados de libertad que usualmente dispone de un magnetómetro de 3 ejes, un giroscopio y un acelerómetro, y por un módulo Arduino, integrado todo dentro de la misma placa. En nuestro prototipo, se ha conectado el sensor IMU a una placa Arduino convencional a través de los pines específicos tal y como se puede apreciar en la imagen anterior.

Se ha modificado el Firmware del Arduino para incluir el Attitude Heading Reporting System firmware (en adelante AHRS), que funciona perfectamente con el paquete de ROS que vamos a utilizar. Este paquete es el razor_imu_9dof e incorpora un driver específico para arrancar el sistema a través del puerto serie, y poder lanzar la ejecución de los nodos específicos para la monitorización y control del IMU

A través de un fichero launch se puede ejecutar el programa y mostrar un entorno de visualización del funcionamiento del IMU que incluye la monitorización de los distintos parámetros. En la siguiente imagen se puede apreciar el sistema en funcionamiento:

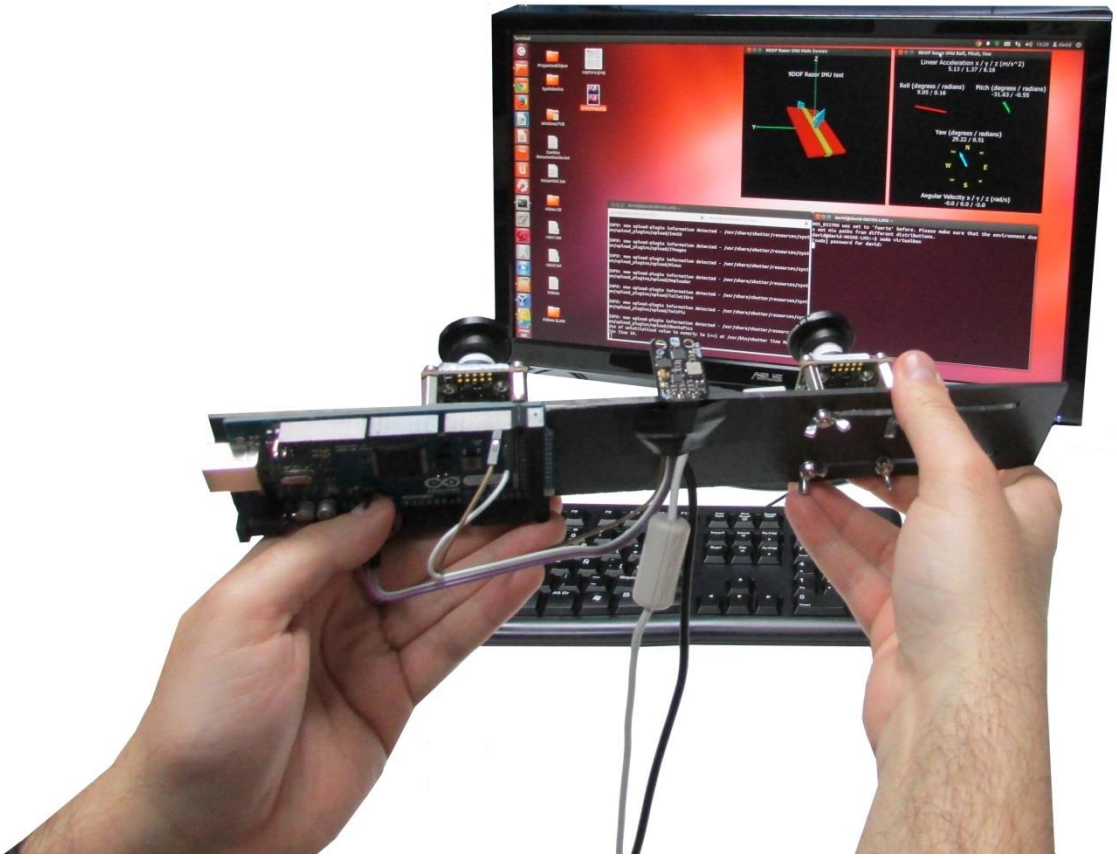

Figura 106: *Monitorización de la distinta información aportada por el IMU, incluidos los ángulos de navegación (Yaw, Pitch, Roll), magnetómetro y valores de aceleración.*

La integración con ROS permite interconectar los diferentes mensajes publicados por los diferentes sensores. En este caso entre la cámara estéreo y el IMU La información del IMU aparece reflejada en los mensajes de tipo sensor_msgs/Imu, que pueden obtenerse directamente abriendo una consola y escribiendo el comando rostopic

echo /imu. El topic "/imu" por el que ROS publica la información de los elementos del sensor (magnetómetro, acelerómetro y giroscopio) se utilizará para generar un nuevo mensaje que resulta de la composición de dicha información con la información de odometría visual aportada por RTAB-Map. Para ello se hace uso del paquete robot_pose_ekf, que estima los Poses en el espacio 3D utilizando distintas medidas de Poses parciales obtenidas de los diferentes sensores. Posteriormente se verá cómo generar el mensaje compuesto, pero antes estudiará la solución RTAB-Map para ROS, describiendo sus diferentes partes entre las que se incluye el nodo stereo_odometry para el cálculo de la odometría visual.

4.1.2 Paquete rtabmap_ros

La implementación de la solución RTAB-Map para SLAM está disponible como paquete de ROS con el nombre de rtabmap_ros. El código de RTAB-Map está disponible bajo licencia BSD, lo cual supone una gran ventaja de cara a su uso, modificación y ampliación para un uso comercial. No obstante, al utilizar OpenCV para la aplicación de los distintos algoritmos de odometría visual, hay que tener en cuenta que si se incluye el módulo nonfree, utilizado por los detectores de características SURF y SIFT, la licencia pasa a ser GPL, lo cual el uso comercial estaría sujeto a los desarrolladores de estos programas. Por otro lado, el algoritmo ORB no pertenece al módulo nonfree, por lo que su uso no está restringido. Este algoritmo ha demostrado ser bastante eficiente en diferentes entornos, aportando un alto índice de características detectadas.

La utilización de uno u otro algoritmo para la detección de características, matching, o parámetros tales como la tasa de detección o número de palabras del BOW (Bag of Words), pueden ser configurados en el fichero launch creado para ejecutar los distintos nodos. Como ya se ha comentado, el paquete rtabmap_ros incluye un nodo llamado stereo_odometry, que se corresponde con la parte del código de RTAB-Map que implementa la odometría visual para cámaras estéreo. Inicialmente, RTAB-Map fue diseñado como una solución para sensores para procesamiento RGBD, es decir para dispositivos que incorporan una cámara RGB y sensores de profundidad, como es el caso de la Kinect®. La odometría visual para este tipo de dispositivos es calculada utilizando el nodo RGBD

odometry, que utiliza información aportada por el sensor de infrarrojos para la profundad, así como detección de características en la imagen. No obstante, posteriormente fue incorporada la funcionalidad para cámaras estéreo lo cual ampliaba las posibilidades de la aplicación.

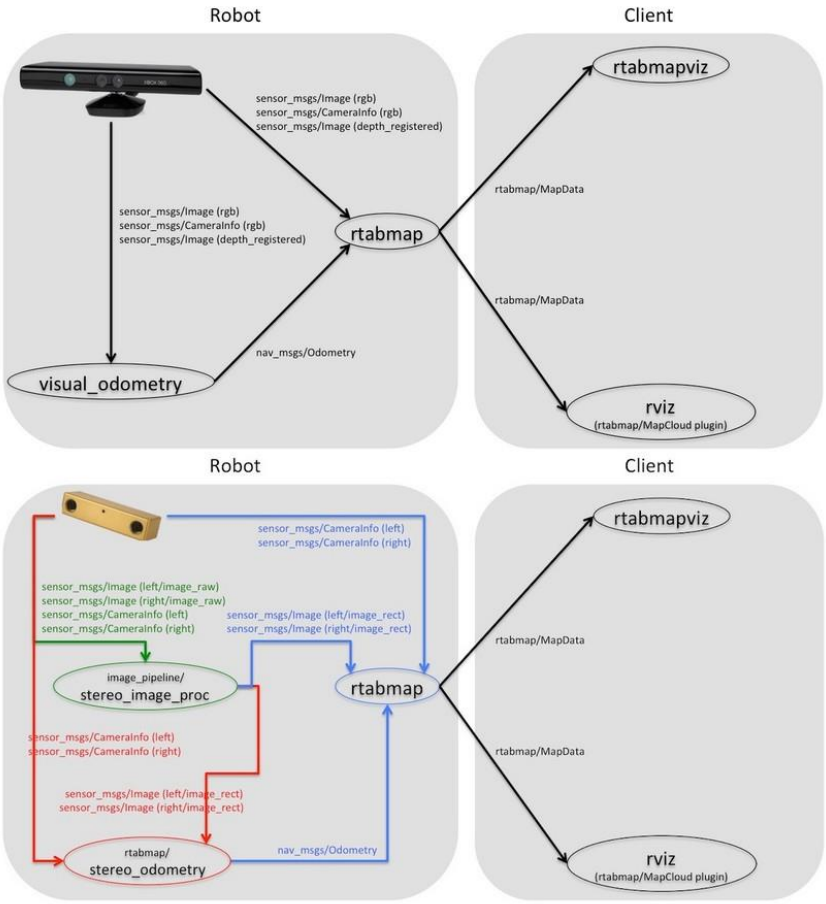

Figura 107: *Esquema de interconexión entre los distintos nodos para el cálculo de odometría visual para RTAB-Map, utilizando RGBD (arriba) y configuración estéreo (abajo).*

En el esquema inferior (configuración estéreo) de la imagen anterior se pueden ver los mensajes generados por cada uno de los nodos que intervienen en el sistema. El nodo stereo_odometry dispone de dos topics que subscriben la información de las imágenes estéreo rectificadas de la cámara (izquierda y derecha), así como otros dos para sus respectivos Camera_Info (información de calibración). Estos mensajes son publicados por el nodo stereo_image_proc.

En base a esta información, el nodo stereo_odometry hace el cálculo de odometría visual haciendo uso de los algoritmos de OpenCV especificados en la configuración. En este caso, se utilizará ORB para

la detección de características y un cálculo de odometría realizando un matching con FLANN sobre imágenes BOW que representan un conjunto de características que se engloban dentro de una bolsa o diccionario que mapea los puntos característicos más cercanos.

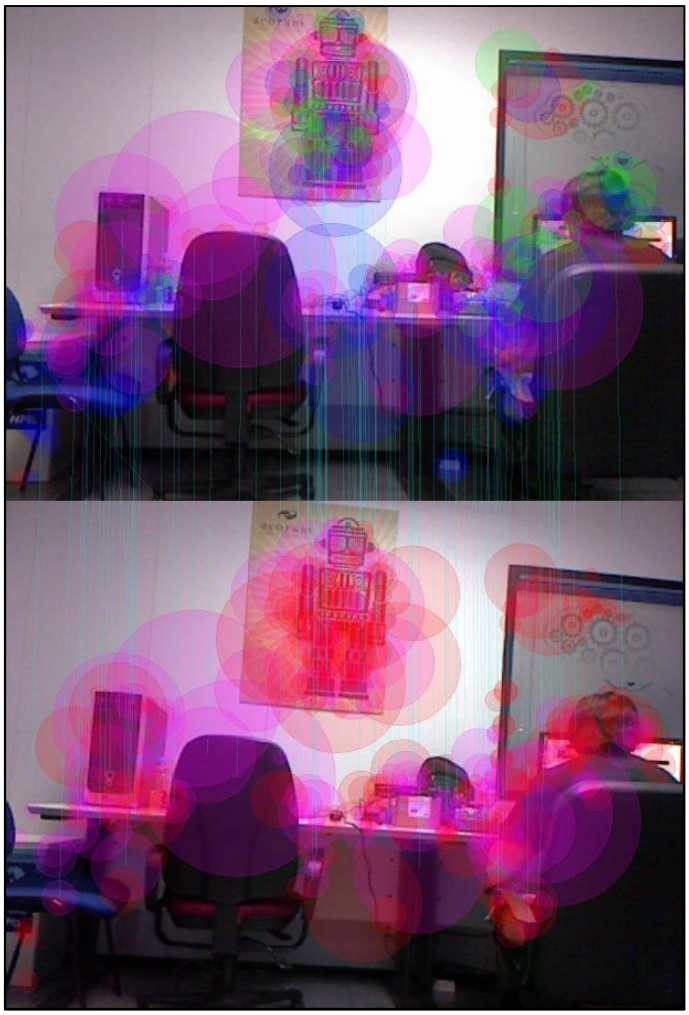

Figura 108: *Estimación de Pose en RTAB-Map utilizando matching FLANN y ORB para la extracción de características. Los círculos represantan los distintos BOWs o conjuntos de características.*

El nodo principal del paquete rtabmap_ros (llamado precisamente rtabmap), construye de forma incremental, a partir de los mensajes de las imágenes y de la odometría visual, un grafo que representa la reconstrucción del mapa 3D con la información de los Poses calculados en cada iteración. Esta información es publicada a través del topic mapData. Sobre el mapa se van realizando optimizaciones también de manera incremental cada vez que se detecta un cierre de lazo. Además, usado en combinación con el nodo map_optimizer es posible realizar optimizaciones del grafo cada cierto tiempo de

manera global, lo cual permite aplicar optimizaciones de manera independiente sobre un mapa global en vez del mapa local sobre el que trabaja el nodo rtabmap.

Finalmente, el nodo map_assembler subscribe los mensajes del grafo generado (mapData) y los ensambla incrementalmente en un mapa 3D representado con una nube de puntos (mensaje sensor_msgs/PointCloud2) y publicado con el topic assembled_clouds. Además, este nodo genera una rejilla ocupacional (OccupancyGrid) para representar el mapa en dos dimensiones calculado a partir de la proyección del mapa 3D generado en el plano xy.

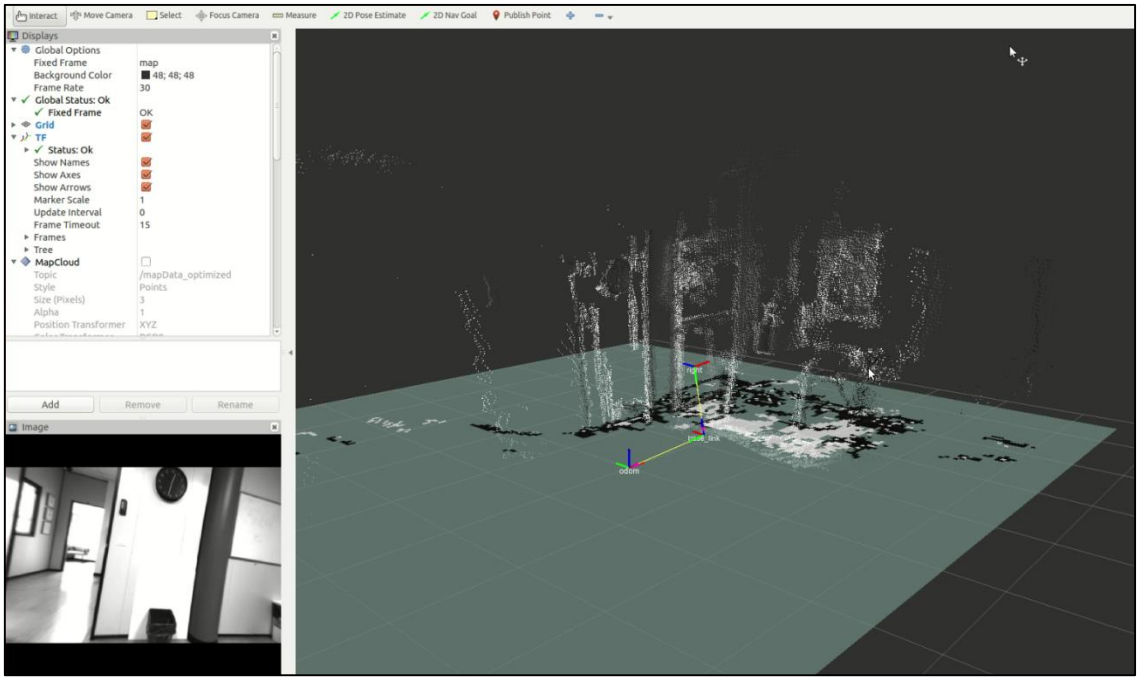

Figura 109: *Visualización del mapa 3D ensamblado con el nodo map_assembler a partir del grafo generado con el nodo rtabmap. Puede observarse también la proyección del mapa sobre la rejilla 2D (OccupancyGrid).*

4.1.3 Composición de odometría

Como ya se ha comentado, el paquete robot_pose_ekf, estima los Poses del sistema a partir de medidas de Poses obtenidas de diversas fuentes, tal y como puede ser la información aportada por un IMU, la odometría obtenida de unas ruedas mecánicas, o bien directamente de la odometría visual. El nodo principal implementa un filtro de

Kalman extendido para determinar el Pose del robot a partir de una composición de las diferentes fuentes de odometría y/u orientación. Los Poses combinados se calculan en función de las diferencias relativas entre los Poses obtenidos de cada sensor, que en cada pasada iteran actualizando el filtro de Kalman extendido buscando una convergencia. Si la precisión en la toma de medidas es lo suficientemente buena, se reducirá considerablemente la pérdida de certidumbre en el cálculo de Poses lo que mejorará el cálculo de la localización.

Los resultados obtenidos con esta Poses combinados mejoran considerablemente el tracking del sistema robótico debido a que no depende de la odometría visual para el cálculo de la orientación (situación que como ya se ha comprobado anteriormente produce errores significativos sobre todo en giros perpendiculares a los ejes ópticos), sino de los cálculos que nos ofrece el IMU.

Por otro lado, los mensajes publicados por el nodo robot_pose_ekf, son directamente el resultado del cálculo de los Poses del sistema y son del tipo geometry_msgs/PoseWithCovarianceStamp. El inconveniente es que el nodo rtabmap no acepta como entrada mensajes de los Poses calculados en crudo, sino que utiliza aquellos que están definidos dentro de un mensaje más genérico llamado nav_msgs/Odometry. Por tanto, es necesario realizar una conversión de este mensaje con el Pose combinado a un mensaje de odometría genérico en el que estén representados la posición y orientación obtenida de los distintos sensores.

Se crea por tanto un nuevo nodo en el paquete robot_pose_ekf al que llamaremos conversor_odom y que básicamente consistirá en un conversor de los mensajes de Poses combinados a mensajes de odometría simples. El nodo esperará un mensaje, extraerá el Pose y lo almacenará en un mensaje nav_msgs/Odometry. Acto seguido el mensaje se publica con el topic rtabmap/odom para así identificar que es la odometría que finalmente leerá el nodo rtabmap. A continuación podemos ver el grafo de nodos del sistema ROS donde se puede observar la conversión realizada por el nodo robot_pose_ekf para obtener la odometría para el nodo rtabmap a partir de la información de orientación aportada por el IMU y la odometría visual aportada por las cámaras:

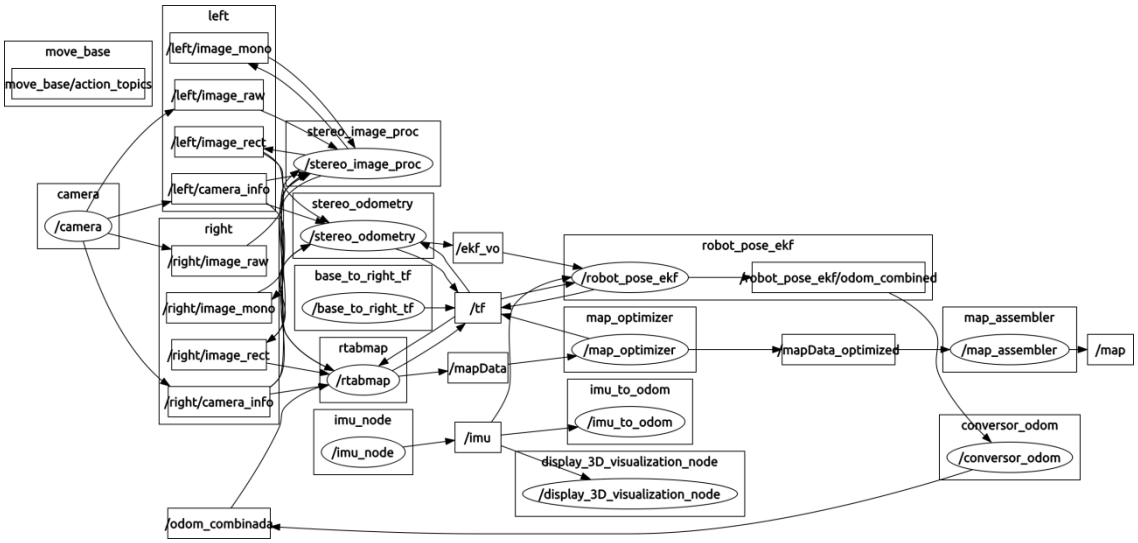

Figura 110: *Grafo de conexiones entre nodos del sistema de odometría combinado y SLAM con RTAB-Map.*

En el fichero launch se incluye además una transformación (tf) del frame correspondiente a la cámara estéreo sobre el frame base_link para inicializar la representación del sistema robótico en el visualizador Rviz. De esta forma, la reconstrucción del mapa 3D aparecerá inicialmente sobre la rejilla 2D sobre la que se proyecta el mapa en el eje xy (Occupancy Grid). A partir de aquí, y tomando como referencia el base_link, se realiza la transformación del frame odom sobre el frame anterior, y finalmente de odom sobre el mapa (frame map). En la siguiente captura se aprecia como aparecen representadas las transformaciones de los diferentes frames sobre el mapa, y de cómo la reconstrucción 3D se va realizando en referencia a las mismas:

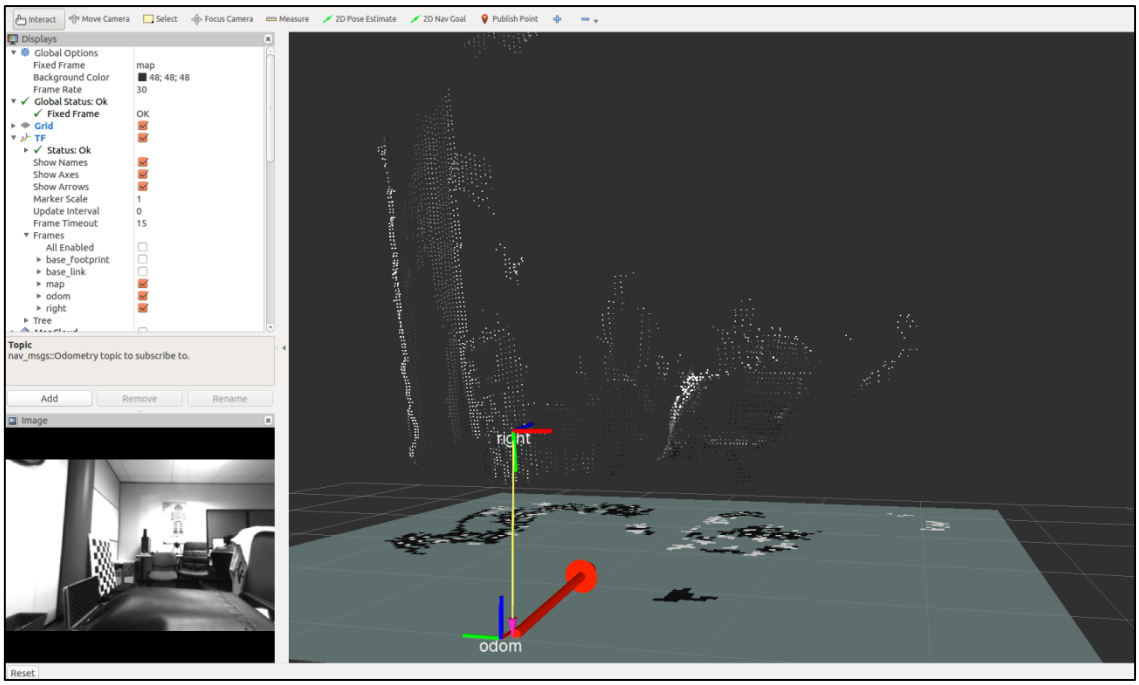

Figura 111: *Transformaciones de frames de odometría y cámara derecha para referenciar la posición y orientación del módulo estéreo en el espacio 3D.*

A continuación se muestra una prueba de reconstrucción realizada con el sistema completo funcionando, superponiendo sobre la nube de puntos ensamblada una capa de MarkerArray que representa una voxelización con Octomap (en la siguiente sección se describe este tipo de representaciones):

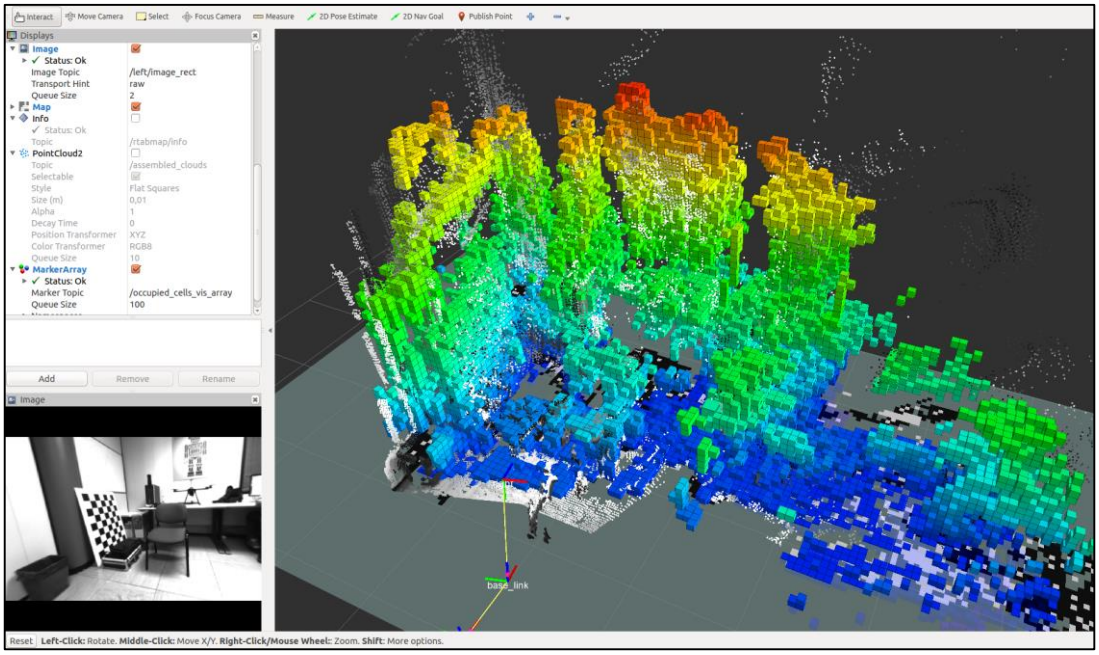

Figura 112: *Reconstrucción 3D con RTAB-Map utilizando el sistema integrado.*

4.1.4 RTAB-Map con procesamiento RGB+D

El procesamiento RGBD hace referencia a las diferentes técnicas de visión por computador que se pueden aplicar haciendo uso de una cámara RGB integrada junto con un sensor infrarrojo [IR] para el cálculo de la profundidad (Depth). El sistema más conocido hoy en día de este tipo es Microsoft® Kinect®. Este sistema de procesamiento RGBD es capaz de inferir la posición de un cuerpo sólido en un proceso de dos etapas.

En la primera utiliza un mapa de profundidad (similar a los mapas de disparidad estéreo) haciendo un análisis sobre un patrón de puntos obtenidos del sensor infrarrojo mediante un escáner de luz estructurada. Este cómputo se realiza a través de un hardware desarrollado por la compañía PrimeSense® y los detalles de la implementación no están disponibles públicamente.

Kinect® combina el escáner de luz estructurada con dos técnicas de visión artificial: distancia basada en enfoque y distancia estéreo. La primera, utiliza el principio de que las cosas que están más difuminadas están más lejanas. Para ello utiliza una lente "astigmática" especial con diferente distancia focal en las direcciones x e y. Estas lentes proyectan un círculo que se convierte en una elipse cuya orientación depende de la profundidad. Por otro lado, la técnica de distancia estéreo se basa en lo que ya hemos explicado en secciones anteriores. Básicamente, obteniendo imágenes de diferentes ángulos, la diferencia de las cercanías entre objetos es mayor cuando más cerca están los objetos, y menos cuanto más alejados. La técnica de distancia estéreo que utiliza Kinect® se denomina parallax.

En la segunda etapa Kinect® es capaz de inferir las partes de un cuerpo a través de algoritmos de decisión aleatorios aprendidos mediante el entrenamiento de cerca de un millón de ejemplos. Esta parte no interviene en la solución para SLAM de RTAB-Map, aunque podría ser útil para muchos otros propósitos.

Para conectar el sistema Kinect® con RTAB-Map en ROS se necesita hacer uso del driver de Kinect® que incorpora el hardware PrimeSense® para el cálculo de profundidad. El paquete ROS que

incorpora todo los mensajes relativos a la cámara RGB y al sensor de profundidad se llama openni_launch, cuyo fichero openni.launch incluye todo el procesamiento comentado anteriormente. Es necesario establecer el parámetro depth_registration a True (originalmente está a false). De esta forma se activa la funcionalidad que permite registrar el mapa de profundidad haciendo uso del escáner de luz estructurada.

Una vez arrancado el driver, el nodelet generado publica varios topics que deberemos mapearlos en RTAB-Map a través del nodo rgbd_odometry y el propio nodo principal rtabmap. Estos topics serán rgb/image y rgb/camera_info, relativos a la imagen RGB y a la calibración de la cámara RGB, y el topic depth/image, correspondiente al sensor de profundidad. Finalmente, el resultado obtenido con la ejecución de un launch adaptado para realizar la reconstrucción 3D con los puntos ensamblados (assembled cloud) y la rejilla 2D (Occupancy Map), haciendo uso de la odometría obtenida por las técnicas RGBD puede apreciarse en la siguiente captura:

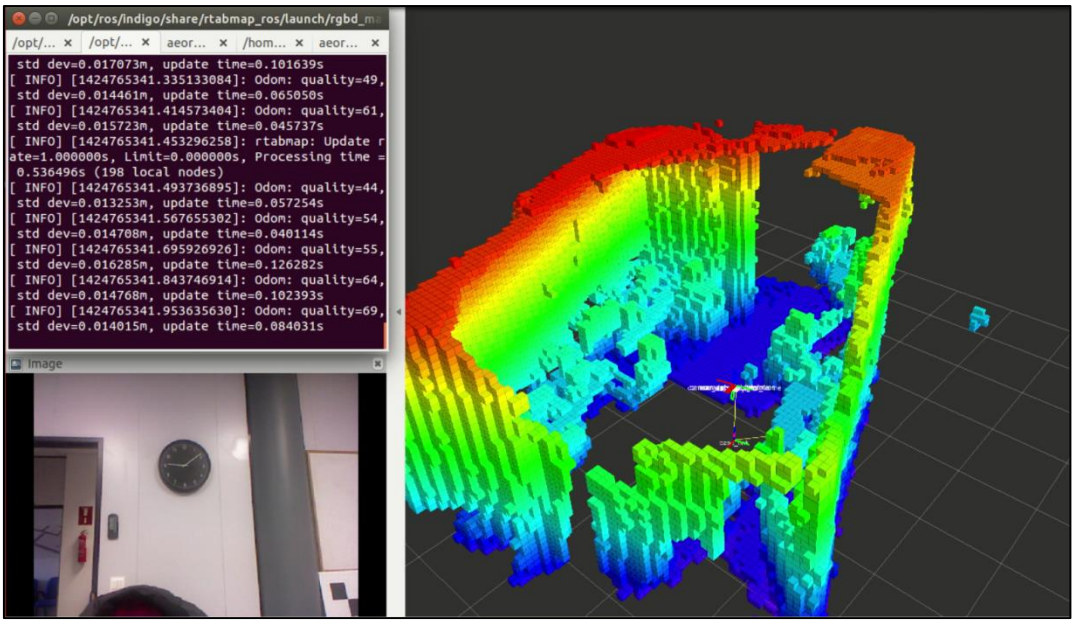

Figura 113: *Reconstrucción con RTAB-Map haciendo uso del sistema Kinect®.*

Gracias a la precisión del sensor de profundidad del sistema *Kinect®* los resultados obtenidos con RTAB-Map son muy buenos. Sin embargo, el inconveniente de esta solución radica sobre todo a su utilización en exteriores. Al ser un sensor de infrarrojos, la luz solar hace irrealizable el cálculo de profundidad en entornos donde el espectro infrarrojo de la luz está presente. Este hecho no ocurre con la solución estéreo.

4.1.5 Optimización de la reconstrucción 3D en tiempo real

Independientemente de las soluciones estudiadas para el problema del SLAM, otro de los factores a tener en cuenta en lo que respecta al mapeado del entorno consiste en el elevado consumo de recursos que supone la reconstrucción de la escena 3D en tiempo real. Esta limitación puede ser salvada haciendo uso de transformaciones sobre los mensajes obtenidos de la nube de puntos generada (point cloud), consiguiendo mapeados basados en la ocupación que tienen los objetos en la escena y no en su representación. Uno de estos mapeados se conoce con el nombre de OctoMap.

Se trata de un framework que realiza una implementación de mapeado 3D basada en octrees. Los octrees son árboles de datos donde cada nodo tiene exactamente ocho hijos. En el espacio tridimensional este árbol es representable mediante la partición de un hexaedro o cubo en ocho subdivisiones, definiendo de forma recursiva más subdivisiones de los nodos en función de los objetos que aparecen en la escena.

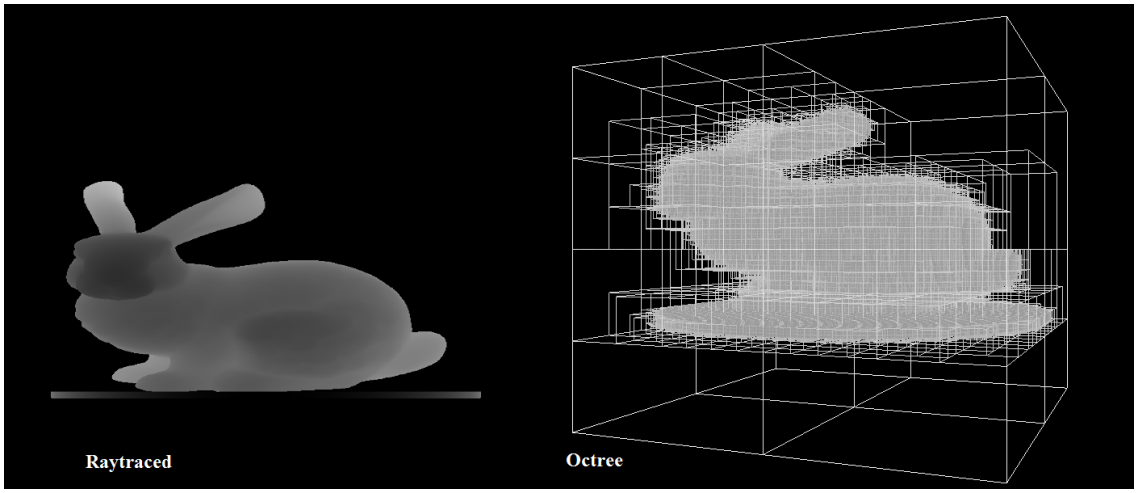

Figura 114: *Obtención de un Octree a partir de un modelo 3D.*

La potencia de OctoMap reside en la capacidad de codificar de forma implícita áreas desconocidas del entorno, siendo capaz además de distinguir entre las zonas libres y ocupadas. Esto último aumenta las posibilidades para la navegación dinámica, ya que permite añadir nuevos cambios en la escena en cualquier momento, así como la expansión de la misma. Aunque la mayor ventaja de utilizar OctoMap

consiste sin duda en la eficiencia de almacenamiento de la información tanto en memoria como en disco.

Mediante la técnica de voxelización (transformación a unidades cúbicas básicas), multitud de formatos para la renderización de mallas poligonales en 3D, tales como 3Ds de Studio Max, VRML, etc., pueden ser convertidos a un fichero OctoMap, reduciendo considerablemente su tamaño en disco.

En ROS puede utilizarse el paquete octomap_server para convertir directamente una nube de puntos obtenida a partir de la imagen de disparidad calculada con visión estereoscópica o de la generada a través de un láser en una voxelización. Para ello, basta con conectar los nodos de entrada y salida, y utilizar en Rviz el componente Array Marker. En la figura 115 se puede observar un ejemplo de reconstrucción 3D obtenido haciendo uso de la solución RTAB-Map, cuya nube de puntos (assembled_cloud) se ha mapeado en una voxelización (OctoMap).

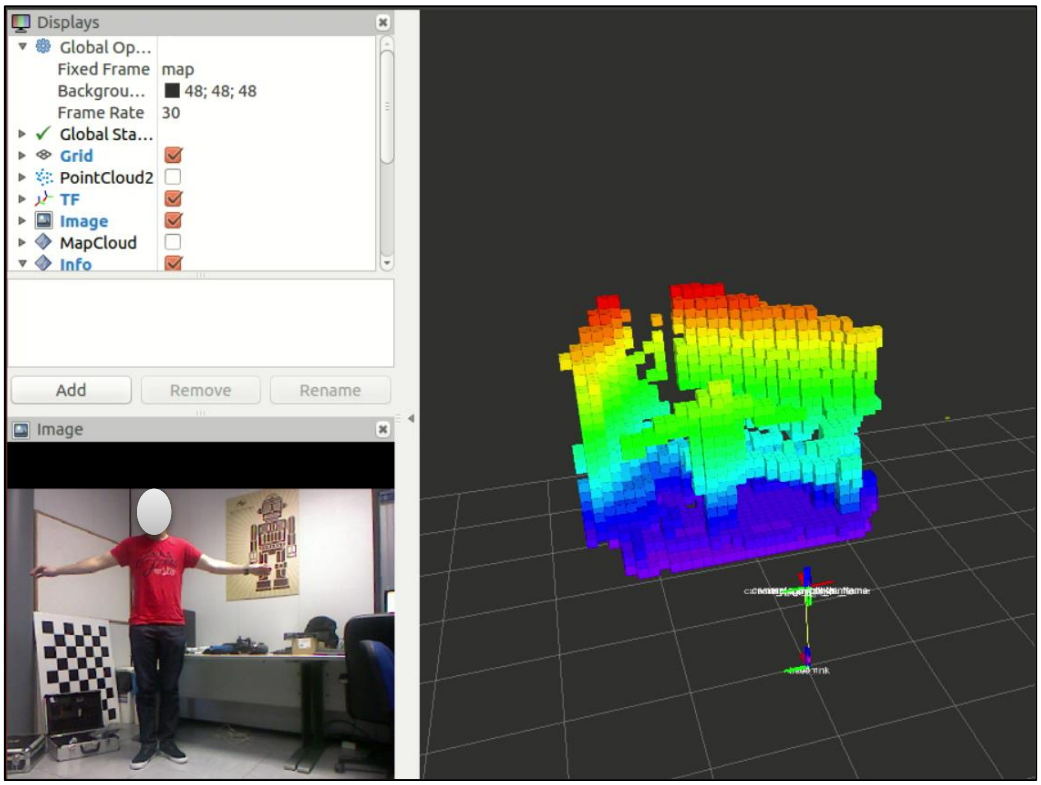

Figura 115: *Utilización de OctoMap para la optimización de la reconstrucción 3D en tiempo real.*

4.2 Desarrollo de sistema de sincronización Hardware

El procesamiento de imágenes estéreo se basa en el análisis de dos imágenes tomadas desde dos cámaras diferentes situadas a una cierta distancia. El objetivo en dicho procesamiento es la obtención de las distancias a los objetos de la escena.

Para cumplir dicho objetivo, es importante que las imágenes se tomen en el mismo instante de tiempo o con el menor retardo posible. El retraso entre capturas puede dar lugar a resultados erróneos y a inestabilidades en el sistema de procesamiento.

Con las cámaras utilizadas, existen dos caminos diferentes para generar el trigger que permite capturar las imágenes. Con las cámaras bajo estudio, UI-1221LE-M-GL, el trigger puede ser generado tanto vía software como hardware.

4.2.1 Trigger software

La primera solución pasa por sincronizar a nivel software las cámaras, junto con el resto de procesado de las imágenes. Para ello, las cámaras adquieren imágenes únicamente cuando se le da la orden correspondiente mediante comando, por lo que es necesario sincronizar ambas cámaras utilizando varias hebras y estableciendo sincronismo entre las mismas

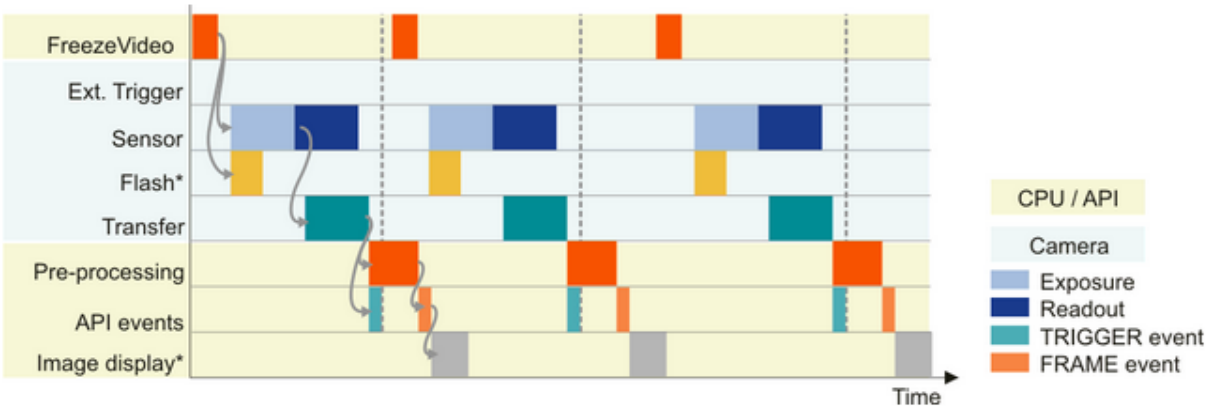

Figura 116: *Esquema temporal de trigger software. (Fuente: manual de la cámara, https://es.ids-imaging.com/manuals/uEye_SDK/EN/uEye_Manual/index.html).*

Mediante este método, la captura entre una cámara y la otra se retrasa entre 25 y 40 milisegundos, inaceptable para la aplicación a desarrollar. Para aplicaciones de visión artificial como la que se desarrolla en la presente tesis, es un retraso demasiado alto para obtener buenos resultados.

El motivo es que, para calcular distancias en un sistema de visión estereoscópico, es necesario que ambas imágenes sean lo más parecidas posibles. Como en general se usa en escenarios donde el sistema se encuentra en movimiento (vehículos autónomos), la falta de sincronismo hará que ambas imágenes no sean realmente de la misma escena.

4.2.2 Sincronización software

Además del trigger software que proporciona el fabricante, se ha desarrollado una solución, capaz de disminuir el retardo hasta el orden de un milisegundo. Para ello, se usan hebras y semáforos de POSIX. En este caso, se utilizan ambas cámaras en modo video, y, utilizando hebras guiadas por un timer, se captura la imagen de ambas cámaras cada cierto periodo de tiempo. No obstante, con este método sigue habiendo un pequeño retardo entre la captura de la imagen de una cámara y la otra, ya que, por software, no se pueden adquirir ambas de forma simultánea.

Aun así, este pequeño retraso de 1 ms sigue haciendo que el sistema de visión artificial no funcione como debería, ya que, cuando la cámara se desplaza, este retardo puede provocar una desincronización en las imágenes de ambas cámaras debido al movimiento. Por ello, se opta por desarrollar el último método, la sincronización hardware.

4.2.3 Sincronización a través del Trigger HW

La sincronización hardware se divide en dos partes bien diferenciadas: la primera es la generación de un trigger HW mediante algún dispositivo para la sincronización de las cámaras, y la segunda, el manejo de las cámaras para capturar y tratar las imágenes obtenidas.

Para que las cámaras respondan a dicho trigger, es necesario estudiar

cómo las cámaras responden a dicha señal. Para ello, en la figura 117 se puede ver la descripción del conector de entrada/salida de las cámaras.

Pin	Description
1	USB power supply (V_{CC}) 5 V
2	USB ground (GND)
3	Trigger input without optocoupler (+)
4	Flash output without optocoupler (+)
5	Power supply of the internal voltage transformer 3.3 V or 3.0 V*
6	USB ground (GND)
7	General Purpose I/O (GPIO) 1
8	General Purpose I/O (GPIO) 2
9	I^2C bus clock signal (internal 2k2 pull-up to V_{INT}*)
10	I^2C bus data signal (internal 2k2 pull-up to V_{INT}*)

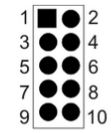

Fig. 442: USB uEye LE PCB version - I/O connector (PCB back view)

Figura 117: Esquema conector I/O. (Fuente: manual de la cámara, especificaciones eléctricas, https://es.ids-imaging.com/manuals/uEye_SDK/EN/uEye_Manual/index.html)

Como se puede ver, el pin número 3 es usado como entrada de trigger (sin optoaclopador). Dicho pin entenderá como nivel bajo una tensión en el rango entre 0 V y 0.8 V. El nivel alto será a partir de 2.0 V. Y el rango de entrada está comprendido entre 0 V y 5.25 V. El trigger HW se podrá activar para operar en flancos de bajada de dicha señal eléctrica o en flancos de subida.

ADVERTENCIA: Como se puede observar, los pines 1 y 5 del esquema son de salida. Aplicar tensión en dichos pines puede causar la destrucción de la cámara.

La secuencia de operaciones que se llevan a cabo en el modo de trigger HW de las cámaras también es relevante de cara a su sincronización.

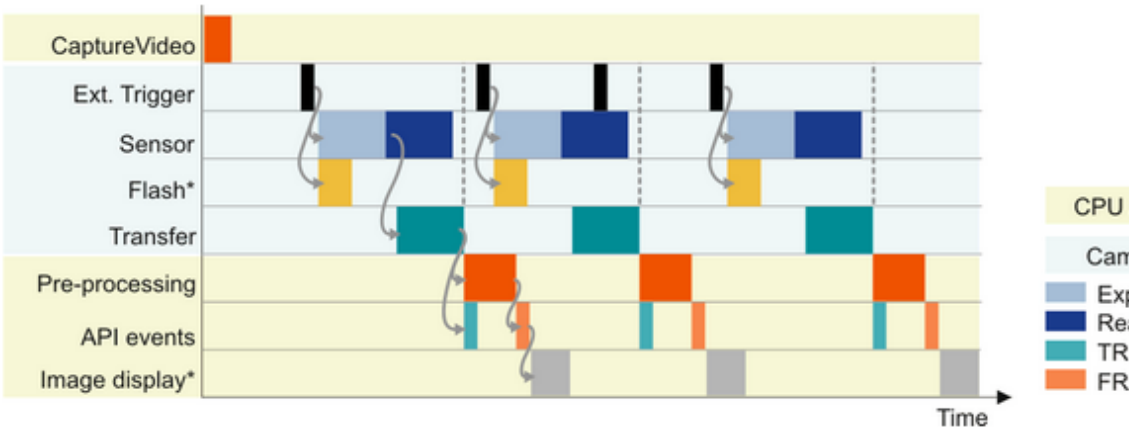

Figura 118: *Esquema temporal de trigger hardware. (Fuente: manual de la cámara, https://es.ids-imaging.com/manuals/uEye_SDK/EN/uEye_Manual/index.html).*

4.2.4 Generación del trigger en Arduino

La primera opción que se plantea es el uso de un Arduino. Un dispositivo de tamaño relativamente reducido, con un microcontrolador ATmega328P de Atmel y gran documentación sobre su uso y librerías que se pueden usar.

Dicho dispositivo dispone de tres timers configurables, dominados por un oscilador de 16 MHz. Los timers son el 0 (no se recomienda usar debido a que es el que usa internamente para las operaciones de entrada y salida serie), el 1 (usado por la biblioteca para el manejo de servos, que puede ser utilizado debido a que no se manejarán servos) y el 2 (usado para la función tone()). El Timer0 y el Timer2 son de 8 bits, mientras que el Timer1 es de 16 bits. En principio, el que se ha utilizado es el 1, puesto que no está previsto usar servos para la generación del trigger HW.

En una primera aproximación, se opta por generar el disparo con funciones predefinidas de Arduino y bibliotecas de terceros (TimerOne). Mediante dicha biblioteca, se generan interrupciones del Timer1 con un periodo dado y se asigna una rutina de tratamiento, donde se genera mediante escrituras en los puertos el flanco de 5 V.

Sin embargo, aprovechando que Arduino monta un microcontrolador,

se lleva a cabo un desarrollo optimizado, donde se sustituyen las llamadas a funciones de bibliotecas de terceros por la manipulación de los registros de dicho microcontrolador. Así, la segunda aproximación a la generación del trigger HW es el uso del Timer1 configurándolo accediendo a los registros del microcontrolador.

En principio, el modo de cuenta utilizado será el modo de cuenta normal. Dicho modo de cuenta se basa en llegar a una cuenta máxima y habilitar el flag de overflow. Para capturar dicho cambio, habrá que definir la ISR (Interrupt Services Routines) asociada a dicho flag de overflow.

Los periodos de cuenta que se pueden obtener vienen dados por:

$$T = \frac{1}{16Mhz} * Preescaler * 2^N \qquad \textbf{Ecuación 4.1}$$

Donde N representa el número de bits del contador y prescaler es el escalado del timer que se ha configurado. Dependiendo del número de bits (8 o 16 bits) y el prescaler definido (1, 8, 64, 256 o 1024), se podrán conseguir diferentes periodos de cuenta. De la ecuación 41 se puede obtener:

N/Prescaler	**1024**	**256**	**64**	**8**	**1**
16 bits	4.19s	1.05s	0.26s	32.8ms	4.09ms
8 bits	16.4ms	4.09ms	1.024ms	0.128ms	16us

Tabla 2: *Periodos de cuenta en función del número de bits (N) y del Prescaler*

Si se requiere una mayor precisión en la frecuencia del trigger, hay que hacer uso del modo de cuenta por comparación (CTC). Dicho modo compara la cuenta con un valor predefinido que viene dado por:

$$OCR1A = 16Mhz/Prescaler/f_{deseada}(2) \qquad \textbf{Ecuación 4.2}$$

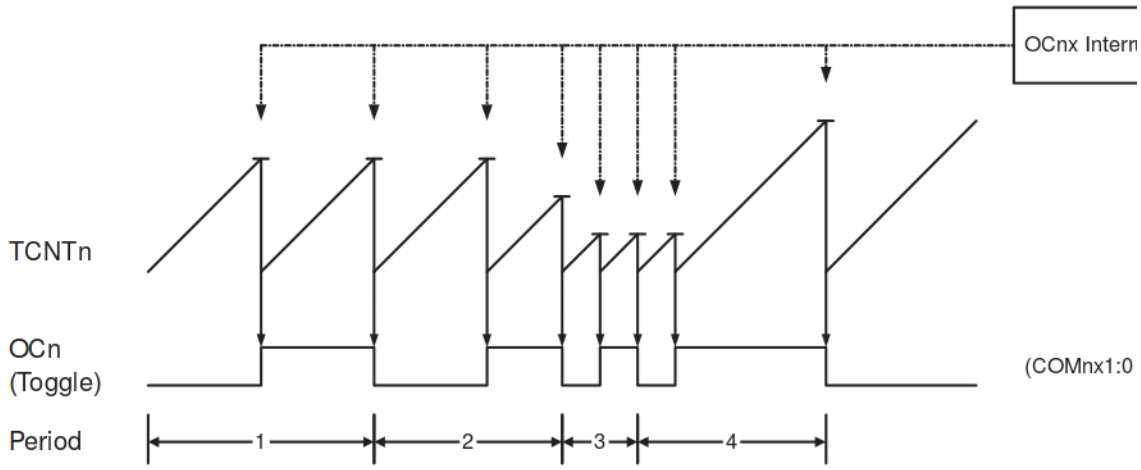

Figura 119: *Modo de cuenta CTC. Fuente: datasheet ATMEGA328P, pág. 100, http://www.atmel.com/Images/doc8161.pdf (Último acceso 18-Jun-2015).*

Además, en los periodos de tiempo en que no se está generando el trigger, se pasa a uno de los modos de bajo consumo del microcontrolador (IDLE). La interrupción del Timer1 será el que saque al Arduino del bajo consumo y regrese a su operación normal.

En cuanto a la alimentación del Arduino, podría hacerse a través de las cámaras, pues tienen un pin capaz de proporcionar 5V necesarios para ello y tierra. Sin embargo, la corriente que es capaz de proporcionar está entre 100 mA y 150 mA.

Del datasheet del microcontrolador, el consumo sin realizar ninguna operación es de 2.5mA a 5V. Por cada pin activo, unos 40mA (se usan 2 pines para los trigger, es decir, 80mA). Además, habría que alimentar los LEDs y demás elementos de la placa, y además añadir un margen de seguridad. Probablemente, alimentar el Arduino desde las cámaras no es viable.

4.2.5 Petición del trigger por comunicación serie

En una primera solución al problema, se opta por pedir el trigger a Arduino mediante comunicación serie. Es decir, manejando puertos en desde el programa encargado de analizar las imágenes. Para ello, es

necesario abrir y configurar el puerto que se encargará de comunicarse con Arduino.

Cada vez que se requieran nuevos frames, se manda una cadena fija desde el programa, que Arduino escuchará a través del puerto USB y generará el trigger.

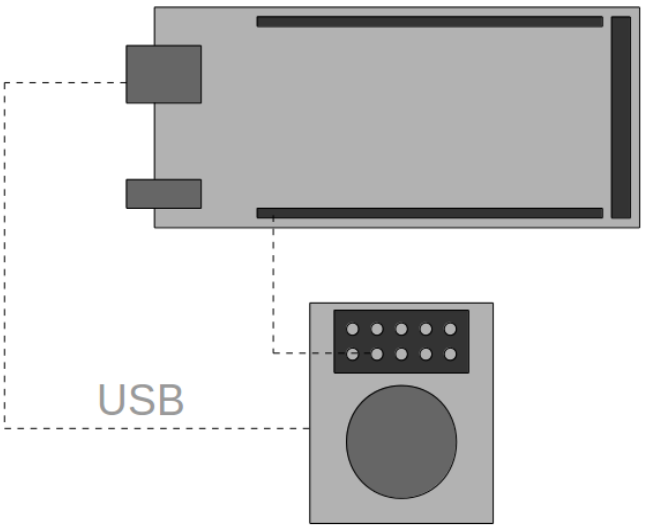

Figura 120: *Petición de trigger mediante USB (Fuente: elaboración propia).*

En la imagen superior se puede ver la petición mediante USB del trigger. Un pin de Arduino se conecta al pin 3 de la cámara, encargado de recibir los trigger hardware.

4.2.6 Petición del trigger desde las cámaras

Para evitar el manejo de puertos y la comunicación serie con un Arduino, se aprovecha los puertos GPIO de los pines de la cámara. Cuando se requiera nuevas capturas, en uno de los dos pines de propósito general se generará un nivel alto. En un dispositivo como Arduino, se estará chequeando constantemente un pin que tenga como entrada dicho pin de la cámara. En el momento que se reciba un nivel alto, se generará el trigger.

Las placas de Arduino alimentadas a 5 V, interpretan como nivel alto una tensión superior a 3 V. Los pines de propósito general de las cámaras, general una tensión a nivel alto de 3.3 V. Por ello, Arduino interpretará dicho voltaje como alto, permitiendo así la generación de un nuevo trigger.

Así, la comunicación entre las cámaras y el dispositivo encargado de generar el trigger es independiente de cómo se traten las imágenes. Los dos dispositivos anteriores se sincronizan entre sí sin necesidad de un mediador.

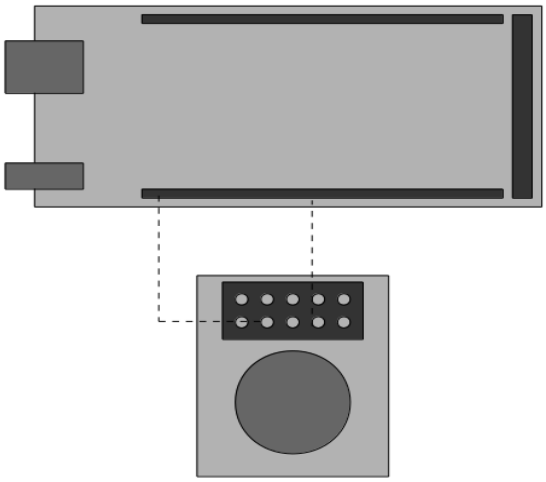

Figura 121: *Petición hardware de triger. (Fuente: elaboración propia).*

En la figura 121 se puede ver la petición hardware del trigger. La generación por parte de la cámara del nivel alto se lleva a cabo en el pin 7 (GPIO1), que será recibido en Arduino y generará un trigger al pin 3 de la cámara. Se puede ver como un trigger desde la cámara a Arduino y otro trigger desde Arduino a la cámara.

4.2.7 Generación del trigger entre las cámaras

Otra posible solución es, desde el código que maneja las cámaras, poner a nivel alto uno de los pines del GPIO de las cámaras. Dicho nivel se realimenta directamente a los pines de entrada de trigger hardware de ambas cámaras.

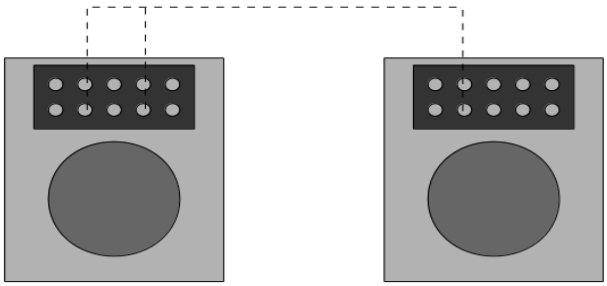

Figura 122: *Configuración maestro-esclavo para petición de trigger. (Fuente: elaboración propia).*

De esta forma, se evita el uso de un sistema para la generación del trigger. La salida del pin 7 de una de las cámaras actúa como trigger de entrada en ambas. Además, se gana en velocidad de captura y por consiguiente, velocidad en cada pasada del bucle. El retraso entre las capturas de las dos cámaras es, idealmente, el que introduce la propagación de la señal eléctrica por cada uno de los cables.

Esta última opción, por su sencillez en cuanto a código (exclusivamente ordenar a una de las cámaras poner a nivel alto uno de los pines de su GPIO) y por ahorro de un dispositivo que genere el trigger será la opción que se usará. Resulta una configuración maestro-esclavo, donde una de las cámaras maneja la captura de ambas.

4.2.8 Manejo de cámaras

Una vez generado el trigger, es necesario controlar la captura de la imagen mediante la API (Acrónimo del inglés API Application Programmer Interface) proporcionada por el fabricante de las cámaras. Al no ser compatibles con V4L, no se puede usar VideoCapture de OpenCV de forma tradicional. La API será el nexo de unión con el procesado usando OpenCV.

A la hora de manejar las cámaras, hay dos opciones: manejar cada una con un thread (de pthread) o hacer la gestión de ambas en un

único hilo. La principal desventaja de llevar cada cámara en un thread es la gestión de memoria compartida y la sincronización entre los threads. Por ello, las líneas a seguir serán:

- Sincronización usando un thread por cámara.
- Uso de librería boost.
- Memoria compartida (POSIX, shm).
- Un thread para proceso y otro que alimente imágenes y sirva de cola.
- Sincronización usando un único hilo.
- Pedir un nuevo trigger a dispositivo externo.
- Generación del trigger entre cámaras (configuración maestro-esclavo).

En ambos casos, la imagen será capturada con OpenCV, una vez se obtenga la imagen de la memoria de la cámara usando la API del fabricante. Para extraer dicha imagen de la memoria, se controlará la generación del evento IS_SET_EVENT_FRAME.

Para evitar el problema de la memoria compartida y sincronización entre threads, se puede optar por manejar ambas en el mismo hilo. Para ello, se espera a que las imágenes estén disponibles, se extraen de la memoria de las cámaras, se procesan y se pide la generación otro trigger. El frame rate no se podrá controlar, pues se generará un nuevo trigger cuando se termine de procesar las imágenes. Depende de la velocidad con la que se hagan las operaciones con las imágenes.

Pero, en principio, no tiene sentido de otra forma, pues para el objetivo perseguido el análisis tiene que ser lo más rápido posible. Guardar una mayor cantidad de frames si no se van a poder analizar no aporta ventaja alguna ante una detección y evasión rápida.

El retardo entre que una imagen está lista en una cámara y en la otra es del orden de nanosegundos. El trigger se puede hacer tanto a través del USB como puramente hardware. Preferiblemente se optará por la hacerlo vía hardware para evitar pérdidas de tiempo en operaciones de comunicación serie.

4.2.9 Resultados: Mejoras en la generación de imagen de disparidad y cálculo de distancias

Tras la instalación del trigger Hardware [HW] en el módulo estéreo IDS UI-1221LE, se ha incluido en el código una capa de comunicación para utilizar las imágenes sincronizadas con integración sobre ROS (en sustitución del paquete uEye de ROS, ya que este solo funciona en modo de sincronización Software [SW]). De igual forma que se hacía con el paquete uEye, se dispone de un nodo que publica las imágenes sincronizadas, además de los CameraInfo con los parámetros de calibración obtenidos a partir del algoritmo de calibración estéreo de ROS, habiéndose eliminado previamente la distorsión de barril provocada por las lentes fish eye.

Disponiendo de los topics de las imágenes raw sincronizadas, se utiliza el paquete stereo_image_proc para obtener las imágenes rectificadas necesarias para calcular la imagen de disparidad. Para visualizar esta última utilizamos el nodo stereo_view del paquete image_view.

A continuación, se muestra una comparativa de dos pares de imágenes de disparidad tomadas en dos instantes de tiempo del orden de milisegundos. De esta forma es posible observar la comparativa de imagen de disparidad entre capturas (sincronicidad) para la sincronización por Trigger HW y Trigger SW:

Capturas comparativas

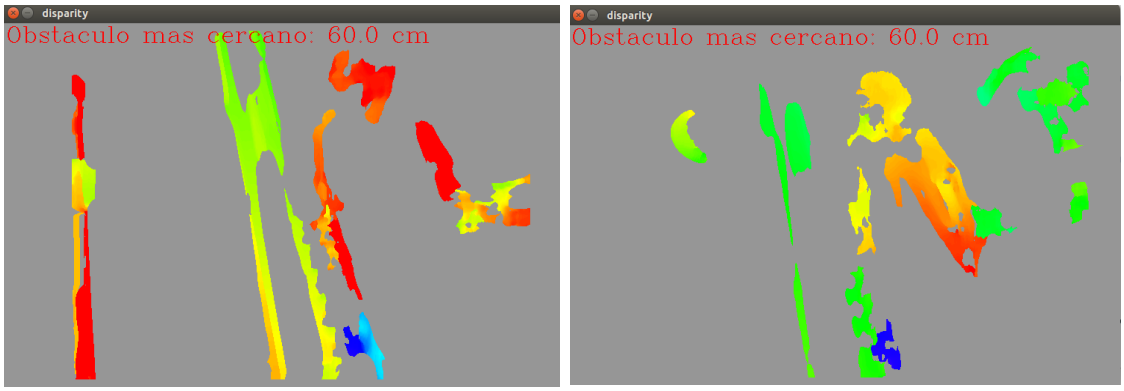

Figura 123: *Sincronización SW*

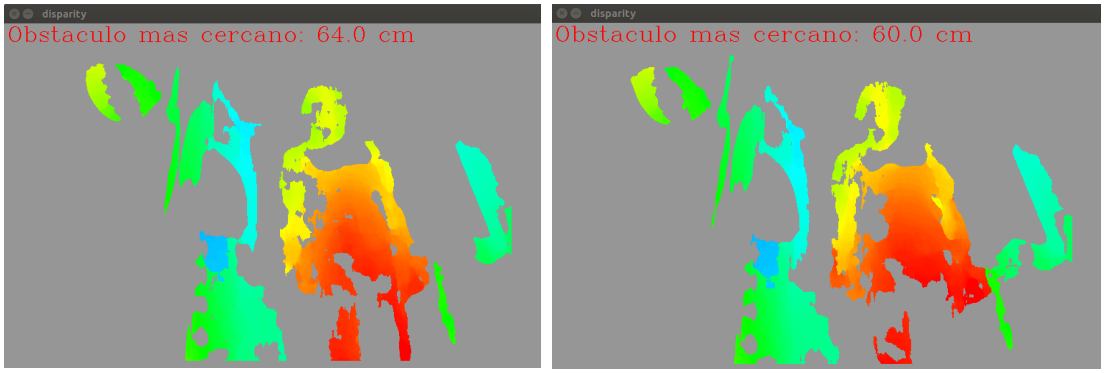

Figura 124: *Sincronización HW*

El primer par está tomado a partir del paquete uEye modificado, utilizando las mejoras de sincronización Software. El segundo par de imágenes está tomado a partir del nuevo código que utiliza la sincronización a través del trigger Hardware.

Como se puede apreciar, aunque los tiempos entre capturas son iguales en ambos casos, en el resultado obtenido utilizando la sincronización SW se aprecia un cambio momentáneo del rango de disparidad en el que los índices no se ajustan a la distancia (la posición no cambia, sólo cambia levemente la orientación). En cambio, en el segundo caso no se aprecia cambio alguno. Se comprueba que el resultado obtenido con la sincronización HW mejora sustancialmente la estabilidad de la imagen de disparidad entre capturas, aun cuando existen movimientos bruscos del módulo estéreo.

	Tamaño Imagen	**Tiempos para Cálculo de Disparidad en ms (milisegundos)**		**Media (ms)**	**Factor preescalado (16 bits)**	**Frames descartados**
Modo SincroHw	752x480	Eliminar distorsión	(42 - 53) ms	177 ms	1	42
		Remaping	(4.5 - 6) ms		8	4
		Disparidad	(120 - 128) ms		64	0
	592x400	Eliminar distorsión	33 - 41.7 ms	139 ms	1	33
		Remaping	(3.5 - 4.7) ms		8	3
		Disparidad	(94 - 101) ms		64	0
	432x320	Eliminar distorsión	(16 - 20.3) ms	68 ms	1	16
		Remaping	(1.7 - 2.3) ms		8	1
		Disparidad	(45.9 - 49) ms		64	0

Tabla 3: *Frames descartados según el factor de preescalado del Trigger HW*

Debido a distintos factores como la transferencia secuencial de las imágenes o la exposición del sensor, es posible que el frame rate con trigger hardware sea menor que en modo freerun. El tiempo requerido para capturar un frame en modo trigger se puede aproximar por la siguiente expresión:

$$T_{cap} = T_{exp} + \frac{1}{FR_{max}}$$ ***Ecuación 4.3***

Donde $\mathrm{T_{cap}}$ se refiere al tiempo de captura, $\mathrm{T_{exp}}$ al tiempo de exposición y $\mathrm{FR_{max}}$ el frame rate máximo en modo freerun.

Existen tres tipos de sincronización para las cámaras IDS:

1. Sincronización SW en modo captura

Este modo de sincronización no usa flujo de video en las cámaras, sino que se encarga de guardar en memoria las imágenes de las cámaras en un momento dado. Cada cámara tendrá asignado un tiempo de frame rate, que definirá el tiempo que ha de transcurrir, como mínimo, entre que se obtiene una imagen de una cámara hasta que se puede solicitar la siguiente.

El problema de este modo de sincronización viene dado por el retardo de la función de captura. El tiempo que se tarda, entre que se obtiene una imagen de una cámara y se guarda en memoria, hasta que se puede realizar la misma operación con la segunda cámara, es muy alto, entre 25 ms y 40 ms. Tras las pruebas se ha confirmado que este modo no es factible para tiempo real.

2. Sincronización SW con Timer

Este modo de sincronización utiliza flujo de video en las cámaras, por lo que cada cámara tendrá constantemente imágenes listas para volcar en memoria tan pronto como sea posible. Para sincronizar las imágenes de ambas cámaras se utiliza un timer, de modo que cuando el contador llega a cero se obtienen las imágenes de memoria de ambas cámaras.

El problema de este modo de sincronización viene dado por el retardo software al obtener las imágenes de memoria. El tiempo que transcurre entre que se obtiene la imágen de una cámara de la memoria del sistema, hasta que se obtiene la imagen de la cámara siguiente oscila entre 150 us y 250 us. Dado que las cámaras se encuentran en modo flujo de video, es altamente probable que, tras capturar la imagen izquierda, la derecha pertenezca a un frame diferente.

3. Sincronización por Trigger HW

Este modo de sincronización utiliza un disparador hardware para capturar las imágenes de las cámaras. Dicho disparador es una señal eléctrica generada por un dispositivo externo o por el propio GPIO de las cámaras. Este modo es similar al modo de *Sincronización SW en Modo Captura*, con la diferencia de que éste funciona por hardware y es por tanto mucho más eficiente. El retardo en recibir el trigger es la

propagación de la señal eléctrica a través del soporte físico utilizado. El disparador hardware definirá el momento en el que cada cámara almacena su imagen en memoria.

Este método de sincronización es el más eficiente que se ha obtenido. El retardo de la obtención de imágenes entre cámaras es entre 600 ns y 900 ns, por lo que mejora la sincronización en un orden de magnitud con respecto a la sincronización software.

Modo	Tamaño de Imagen	Tiempo
Sincronización SW en Modo Captura	**752 x 480**	**(25 000 000 - 40 000 000) ns**
	592 x 400	**(19 600 000 - 31 500 000) ns**
	432 x 320	**(9 600 000 - 15 500 000) ns**
Sincronización SW con Timer	**752 x 480**	**(150 000 - 250 000) ns**
	592 x 400	**(118 000 - 195 000) ns**
	432 x 320	**(57 500 - 95 000) ns**
Sincronización por Trigger HW	**752 x 480**	**(600 - 900) ns**
	592 x 400	**(472 - 710) ns**
	432 x 320	**(230 - 345) ns**

Tabla 4: *Tiempos entre captura de frame izquierdo y derecho*

4.2.10 Conclusiones

Mediante el uso de trigger hardware, el retraso entre captura de las dos cámaras es del orden de nanosegundos. Se mejora en seis órdenes de magnitud respecto al sincronismo software (del oden de milisegundos). Con esta mejora, el retardo es casi despreciable, mejorando sobremanera las prestaciones del sistema de visión artificial.

Gracias a la mejora, el objetivo buscado, la distancia a objetos, así como los procesos que partan de ella, se consiguen de manera más fiable y estable. En esencia, se mejora la búsqueda de correspondencias entre los frames, que son la base para el cálculo de disparidad, a partir del cual se construye el mapa de profundidad.

CAPITULO 5: CONCLUSIONES Y LINEAS FUTURAS DE INVESTIGACIÓN

5.1 Conclusiones

El desarrollo de la presente tesis ha permitido el análisis del estado del arte en sistemas de detección de obstáculos. Los sistemas de detección de obstáculos son fundamentales para sistemas de agentes automáticos que interaccionen con el entorno y sean capaces de realizar tareas desatendidas. Dentro de los sistemas de detección se han determinado tres grandes grupos, con distintas características y capacidades:

- Sistemas sónicos: muy económicos y con una interfaz de uso sencilla. No es posible obtener medidas exactas de posicionamiento de obstáculos y en situaciones de ruido ambiente el rango de funcionamiento de los ultrasonidos utilizados puede estar saturado, eliminando cualquier medida.
- Sistemas LIDAR: análisis extenso en tiempo real de áreas y generación de nubes de puntos con los obstáculos. Requieren de un hardware dedicado para la gestión de las medidas y son muy caros.
- Sistemas láser mini LIDAR: análisis puntual en una única dirección. Posibilidad de ampliar el rango analizado por medio de un sistema de servo-motores. La lectura de los datos se realiza de forma secuencial y es necesario una calibración exacta de los servos y el láser para obtener medidas útiles.
- Sistemas basados en visión artificial:
    - Odometría monomodal: usa cámaras simples y realiza un análisis secuencial de las distintas imágenes captadas a lo largo del tiempo. El coste es reducido, pero requiere de imágenes precisas y rangos de tiempos determinados para realizar el análisis. Además, movimientos bruscos pueden suponer la pérdida de sincronismo entre las imágenes tomadas de forma secuencial.
    - Odometría estéreo: usa cámaras estéreo o un par de cámaras simples sincronizadas. Elimina la necesidad del tratamiento de imágenes secuenciales. Necesita la

calibración de las cámaras y la implementación de sistemas de matching de imágenes para la obtención de mapas de disparidad precisos.

Las conclusiones del análisis e implementación de los anteriores métodos reseñados arrojan las siguientes conclusiones:

- El método más eficaz para la detección de obstáculos en tiempo real y el que requiere un menor esfuerzo en implementación software es el uso de sensores LIDAR El principal problema es que requiere un hardware muy específico y caro que lo hace inviable para su uso en conjuntos de agentes distribuidos. Además, supone la inclusión de un sobrecoste en cuanto al peso incluido en cada uno de los agentes que puede penalizar características como la autonomía.
- El método más sencillo es la detección por láser. Este método sin embargo, solo es capaz de realizar medidas puntuales y requiere de estrategias adicionales para la obtención de medidas adicionales. Estas estrategias no permiten la obtención en tiempo real de información detallada de la escena y penaliza la velocidad máxima de operación de los agentes.
- El método más flexible es el del análisis de imágenes estéreo, extrayendo de ellas la información relativa a los volúmenes detectados en un arco vertical y horizontal que hacen eficiente su uso en agentes autónomos.

Centrados en este último método, se ha implementado una metodología propia para la calibración y ajuste de las cámaras necesarias para el análisis, así como la implementación de los métodos para la generación de mapas de disparidad eficientes para la obtención de mapeados de obstáculos en tiempo real. Para ello se ha implementado un sistema de cámaras estéreo basado en el modelo UI-1221LE-M-GL del fabricante IDS montado sobre un bastidor. Sobre este montaje software, se ha implementado la extensión de trigger hardware, comparándolo con el método software habitual.

El resultado global de la experiencia indica un aumento en la eficiencia de la toma de medidas en torno a 4 grados de magnitud. El

resumen de cálculo de tiempos para la obtención de las imágenes estéreo queda:

Modo	Tamaño de Imagen	Tiempo
Sincronización SW en Modo Captura	752 x 480	(25 000 000 - 40 000 000) ns
	592 x 400	(19 600 000 - 31 500 000) ns
	432 x 320	(9 600 000 - 15 500 000) ns
Sincronización SW con Timer	752 x 480	(150 000 - 250 000) ns
	592 x 400	(118 000 - 195 000) ns
	432 x 320	(57 500 - 95 000) ns
Sincronización por Trigger HW	752 x 480	(600 - 900) ns
	592 x 400	(472 - 710) ns
	432 x 320	(230 - 345) ns

Tabla 5: *Resumen de cálculo de tiempos para la obtención de las imágenes estéreo*

La disminución del tiempo entre medidas de ambas cámaras implica el obtener imágenes coherentes en ese rango de tiempo. En aplicaciones que involucran agentes móviles, es determinante para obtener medidas precisas que sean capaces de congelar la escena y obtener medidas precisas aun cuando los agentes se mueven por el medio a gran velocidad.

Como conclusión principal, se ha de destacar la versatilidad de los métodos basados en visión artificial frente a los centrados en aspectos hardware (láser y sensores sónicos). El aumento en la precisión de las medidas de toma secuencial de imágenes es fundamental para poder realizar medidas precisas sobre los datos obtenidos sobre las cámaras. En la presente tesis se presenta el método de sincronización hardware para medidas coherentes de cámaras Global Shutter, obteniendo tiempos de sincronización que mejoran en varios órdenes de magnitud las medidas obtenidas con anterioridad.

5.2 Líneas futuras de investigación

Las futuras líneas de investigación se pueden dividir en tres niveles fundamentales: bajo, medio y alto nivel.

A bajo nivel, los trabajos posteriores se centrarán en la optimización del cálculo de mapas de disparidad entre imágenes y la optimización de los métodos de eliminación de distorsión. Los mapas de disparidad permiten determinar volúmenes por medio de la comparación de imágenes estéreo estáticas. La optimización de los cálculos permite obtener una mayor frecuencia de medidas y, por tanto, una mejora en la velocidad de operación de los agentes móviles. El cálculo de los mapas de disparidad supone la mayor parte del tiempo de proceso para el cálculo de distancia, en torno al 70% del tiempo, por lo que optimizaciones en este tiempo redunda en una mejora global en la eficiencia de los procedimientos. Por otro lado, la fase de eliminación de distorsión de las lentes utilizadas penaliza los cálculos de toma de medidas en torno a un 20% del tiempo. Mejoras en estos cálculos pueden ayudar también a reducir el tiempo total de cálculo de frames.

A medio nivel, una vez desarrollados algoritmos lo suficientemente óptimos para el procesamiento de imágenes, podemos acelerar su desempeño por dos vías fundamentales:

- Uso de coprocesadores gráficos (GPU) incluidos cada vez en más sistemas empotrados. Pueden ayudar enormemente en operaciones complejas realizadas con matrices.
- Traslación de partes de los algoritmos a implementaciones FPGA para acelerar su ejecución y obtener medidas rápidas en las partes más críticas o costosas.

Por último, a alto nivel, el análisis de imágenes por medio de algoritmos de visión artificial, permite no solo realizar medidas sobre el mundo real, sino tomar consideraciones cualitativas sobre el entorno. Para ello será necesario seguir aumentando la capacidad de cálculo de los sistemas empotrados existentes en el mercado, que permitirán cada vez una mayor capacidad de proceso con un coste energético y de peso acotados. El análisis en tiempo real de las imágenes del entorno será el método con mayor campo de crecimiento dentro de los sensores de obstáculos. Cabe destacar que

el análisis más fiable hoy en día, el realizado por LIDAR, está limitado a la obtención de una nube de puntos representativa del entorno. El análisis por medio de algoritmos de visión artificial permite ir más allá, pudiendo diferenciar obstáculos en función de su naturaleza, por ejemplo:

- Determinar rutas óptimas en función de las distribuciones de los obstáculos.
- Evitar obstáculos definidos como críticos o de vital importancia como personas.
- Evitar zonas previsiblemente problemáticas como las corrientes de convección sobre un incendio.
- Establecer estrategias complejas en la realización de tareas en grupo, por ejemplo, el recorrido de la cubierta de un puente, limitándose a esta parte.

El campo de investigación en el ámbito de la visión artificial va acelerando su desarrollo, conformándose como parte fundamental de los sistemas inteligentes autónomos y su relación con el entorno. Con el desarrollo de métodos robustos para la obtención de medidas coherentes como los presentados en la presente tesis se establecen las bases necesarias para favorecer el avance en dicha disciplina.

Índice de referencias bibliográficas

[1] Centro Nacional para la Protección de Infraestructuras Críticas. Ministerio de Interior. Gobierno de España., «http://www.cnpic.es,» [En línea]. Available: http://www.cnpic.es. [Último acceso: 8 Marzo 2015].

[2] Comisión Europea, «Libro Verde sobre el Cambio Climático,» 2011.

[3] IEA - International Energy Agency, «World Energy Outlook,» 2011.

[4] El Mundo, «http://www.elmundo.es,» [En línea]. Available: http://www.elmundo.es/elmundo/2009/08/17/internacional/1250508094.html. [Último acceso: 17 agosto 2009].

[5] ONU - Organización de Naciones Unidas, «Foro Mundial del Agua,» Marsella, 2012.

[6] H. Zlotnik, *Directora de la División de Población de la Naciones Unidas,* Nueva York, 2009.

[7] Oficina Española de Cambio Climático. Ministerio de Medio Ambiente, Rural y Marino, «Segundo Informe de Seguimiento del Plan Nacional de Adaptación al Cambio Climático,» 2011.

[8] Comisión Europea, Libro Verde sobre protección de los bosques e información forestal de la UE: Preparación de los bosques al cambio climático., Bruselas, 2010.

[9] A. M. Arrufi, La construcción sostenible y el cambio climático, 2011.

[10] MINAE - Ministerio de Ambiente y Energía. Gobierno de Costa Rica., *Segunda comunicación nacional a la convención marco de las Naciones Unidas sobre el cambio climático,* San José de Costa Rica, 2009.

[11] Gobierno de España, *Ley 8/2011, de 28 de abril, por la que se establecen medidas para la protección de las infraestructuras críticas,* Madrid: Boletín Oficial del Estado, 2011.

[12] Comisión Europea - Consejo Europeo de Tesalónica, «Estrategia Europea de Seguridad: "Una Europa más segura en un mundo mejor",» Tesalónica, 2003.

[13] Comisión Europea, «EPICP - Programa Europeo para la Protección de Infraestructuras Críticas,» EUR-Lex, Bruselas, 2006.

[14] Comite Ejecutivo para el Mando Unificado - Gobierno de España, *Plan de prevención y protección antiterrorista (revisión),* BOE, 2003.

[15] V. Popescu, P. Rosen, L. Arns, X. Tricoche, C. Wyman y C. M. Hoffmann, «The General Pinhole Camera: Effective and Efficient Nonuniform Sampling for Visualization,» *IEEE Transactions on Visualization & Computer Graphics,* vol. 16, nº 5, pp. 777-790, 2010.

[16] J. Bloomenthal y J. Rokne, «Homogeneous coordinates,» *The Visual Computer,* vol. 11, nº 1, pp. 15-26, January 1994.

[17] A. Heyden y K. Åkström, «Minimal conditions on intrinsic parameters for Euclidean reconstruction,» *Computer Vision,* vol. 1352, pp. 169-176, 29 July 2005.

[18] A. Woods, T. Docherty y R. Koch, «Image Distorsions in Stereoscopic Video Systems,» *Stereoscopic Displays and Applications,* vol. 1915, nº 4, February 1993.

[19] C. Hughes, M. Glavin, E. Jones y P. Denny, «Review of Geometric Distortion Compensation in Fish-Eye Cameras,» de *Signals and Systems Conference, 208. (ISSC 2008). IET Irish,* Galway, 2008.

[20] D. S. Kumar y C. V. Jawahar, «Robust Homography-Based Control for Camera Positioning in Piecewise Planar Environments,» *Computer Vision, Graphics and Image Processing,* vol. 4338, pp. 906-918, 2006.

[21] L. Yang, Y.-C. Tse, P. V. Sander, J. Lawrence, D. Nehab, H. Hoppe y C. L. Wilkins, «Image-based bidirectional scene reprojection,» *CM Transactions on Graphics,* vol. 30, nº 6, December 2011.

[22] R. Hartley, «An Investigation of the Essential Matrix,» G.E. CRD, Schenectady, NY, 1992.

[23] Q.-T. Luong y O. D. Faugeras, «The fundamental matrix: Theory, algorithms, and stability analysis,» *International Journal of Computer Vision,* vol. 17, nº 1, pp. 43-75, January 1996.

[24] Z. Zhang, «Determining the Epipolar Geometry and its Uncertainty: A Review,» *International Journal of Computer Vision,* vol. 27, nº 2, pp. 161-195, March 1998.

[25] J. Karathanasis, D. Kalivas y J. Vlontzos, «Disparity Estimation using Block Matching and Dynamic Programming,» de *Electronics, Circuits, and Systems, 1996. ICECS '96., Proceedings of the Third IEEE International Conference*, Rodos, 1996.

[26] P. Didyk, T. Ritschel, E. Eisemann, K. Myszkowski y H.-P. Seidel, «A perceptual model for disparity,» *ACM Transactions on Graphics,* vol. 30, nº 4, July 2011.

[27] H. Hirschmüller, «Semi-Global Matching – Motivation, Developments and Applications,» HPGU, Oberpfaffenhofen, 2011.

[28] I. Nahhas y M. Drahansky, «Analysis of Block Matching Algorithms with Fast Computational and Winner-Update Strategies,» *International Journal of Signal Processing, Image Processing and Pattern Recognition,* vol. 6, nº 3, June 2013.

[29] V. Kolmogorov y R. Zabih, «Graph Cut Algorithms for Binocular Stereo with Occlusions,» de *Mathematical Models in Computer Vision: The Handbook*, Springer-Verlag, 2005.

[30] T. Leighton y S. Rao, «Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms,» *Journal of the ACM,* vol. 46, nº 6, pp. 787-832, November 1999.

[31] K. G. Derpanis, «The Harris Corner Detector,» Technical Report, York University , 2004.

[32] C. Xuejun, «A New, Self-Adaptative, KLT-Based Algorithm for Visual Tracking,» *International Journal of Signal Processing, Image Processing and Pattern Recognition,* vol. 8, nº 1, pp. 61-68, 2015.

[33] D. Scaramuzza y F. Fraundorfer, «Visual Odometry,» *Robotics & Automation Magazine,* vol. 18, nº 4, pp. 80-92, December 2011.

[34] H. Durrant-Whyte y T. Bailey, «Simultaneous Localization and Mapping (SLAM): Part I The Essential Algorithms,» *Robotics & Automation Magazine,* vol. 13, nº 2, pp. 99-110, June 2006.

[35] H. Durrant-Whyte y T. Bailey, «Simultaneous Localization and Mapping (SLAM): Part II State of the Art,» *Robotics & Automation Magazine,* vol. 13, nº 3, pp. 108-117, September 2006.

[36] D. Lowe, «Object Recognition from Local Scale-Invariant Features,» de *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference*, Kerkyra, 1999.

[37] J. Canny, «A Computational Approach to Edge Detection,» *Pattern Analysis and Machine Intelligence,* vol. 8, nº 6, pp. 679-698, November 1986.

[38] O. Saurer, K. Köser, J.-Y. Bouguet y M. Pollefeys, «Rolling Shutter Stereo,» de *Computer Vision (ICCV), 2013 IEEE International Conference,* Sydney, 2013.

[39] G. Meinants, «Global shutter image sensors,» Solid State Technology, Antwerp, Belgium, 2014.

[40] K. M. Wurm, M. Bennewitz, C. Stachniss y W. Burgard, «OctoMap: an efficient probabilistic 3D mapping framework based on octrees,» *Autonomous Robots,* vol. 34, nº 3, pp. 189-206, 2013.

[41] J. Wei, L. Chen-Feng, H. Shi-Min, R. M. Ralph y T. Chiew-Lan, «Fisheye Video Correction,» *IEEE Transactions on Visualization and Computer Graphics,* vol. 18, nº 10, pp. 1771-1783, 2012.

[42] M. V. Sarode y P. R. Deshmukh, «Reduction of Speckle Noise and Image Enhancement of Images Using Filtering Technique,» *International Journal of Advancements in Technology,* pp. 30-38, 2011.

[43] L. Meier y J. Camacho, «Mavlink: Micro air vehicle communication protocol,» 2013.

[44] J. Diebel, «Representing attitude: Euler angles, unit quaternions, and rotation vectors,» *Matrix,* vol. 58, pp. 15-16, 2006.

[45] J. Israelsen y M. Beall, «Automatic Collision Avoidance for Manually Tele-operated Unmanned Aerial Vehicles,» *International Conference on Robotics & Automation,* p. 6, 2014.

[46] D. Holz y M. Nieuwenhuisen, «TOWARDS MULTIMODAL OMNIDIRECTIONAL OBSTACLE DETECTION FOR AUTONOMOUS UNMANNED AERIAL VEHICLES,» *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.,,* p. 6, 2013.

[47] I. A. ŞUCAN y L. E. KAVRAKI, «Kinodynamic motion planning by interior-exterior cell exploration,» *Algorithmic Foundation of Robotics VIII. Springer Berlin Heidelberg,* pp. 449-464, 2010.

[48] W. Zeng y R. L. Church, «Finding shortest paths on real road networks: the case for A*,» *International Journal of Geographical Information Science,* vol. 23, nº 4, pp. 531-543, 2009.

[49] P. E. Hart, N. J. Nilsson y B. Raphael, «A Formal Basis for the Heuristic Determination of Minimum Cost Paths,» *IEEE Transactions on Systems Science and Cybernetics SSC4,* vol. 4, nº 2, pp. 100-107, 1968.

[50] J. Solá, «Simulataneous localization and mapping with the extended Kalman filter,» 2014.

[51] P. J. J. T. a. J. N. L.M. Paz, «EKF SLAM updates in O(n) with Divide and Conquer SLAM,» Universidad de Zaragoza, 2007.

[52] S. T. D. K. B. W. Michael Montemerlo, «FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem,» Carnegie Mellon University & Stanford University, Pittsburgh & Stanford, 2002.

[53] C. C. a. J. N. Yasir Latif, «Detecting the correct graph structure in pose graph SLAM,» 2013.

[54] J. E. a. T. S. a. D. Cremers, «LSD-SLAM: Large-Scale Direct Monocular SLAM,» Technical University Munich, 2015.

[55] D. G. Lowe, «Distinctive Image Features from Scale-Invariant Keypoints,» University of British Columbia, B.C., Canada, Vancouver, 2004.

[56] M. E. N. E.-S. Bassem Sheta, «ASSESSMENTS OF DIFFERENT SPEEDED UP ROBUST FEATURES (SURF) ALGORITHM RESOLUTION FOR POSE ESTIMATION OF UAV,» *International Journal of Computer Science & Engineering Survey (IJCSES),* vol. 3, nº 5, pp. 15-41, 2012.

[57] A. M. S. Sadgi Sharma, «Image Retrieval Using Speeded Up Robust Feature: An Effort to Improvement,» *IJCSNS International Journal of Computer Science and Network Security,* vol. 14, nº 11, pp. 102-107, 2014.

[58] V. R. K. K. G. B. Ethan Rublee, «ORB: an efficient alternative to SIFT or SURF,» Willow Garage, Menlo Park, California, 2011.

[59] D. L. Marius Muja, «FLANN - Fast Library for Approximate Nearest Neighbors. User Manual,» University of British Columbia, Canada, 2009.

[60] D. Z. Chen, «Developing algorithms and software for geometric path planning problems,» *ACM Computing Surveys (CSUR) - Special issue: position statements on strategic directions in computing research,* vol. 28, nº 18, 1996.

[61] F. Belkhouche y T. A. I. Syst. Eng. Program, «Reactive Path Planning in a Dynamic Environment,» *Robotics, IEEE Transactions on,* vol. 25, nº 4, pp. 902-911, 2009.

[62] S. Ge y Y. Cui, «Dynamic Motion Planning for Mobile Robots Using Potential Field Method,» *Autonomous Robots,* vol. 13, nº 3, pp. 207-222, 2002.

[63] J. K. Satia y R. E. Lave, «Markovian Decision Processes with Probabilistic Observation of States,» *Management Science,* vol. 20, nº 1, pp. 1-13, 1973.

[64] G. Roland y M. H. Overmars, «A Comparative Study of Probabilistic Roadmap Planners,» *Algorithmic Foundations of Robotics V,* vol. 7, pp. 43-57, 2004.

[65] R. Bohlin, C. U. o. T. G. S. Dept. of Math. y L. Kavraki, «Path planning using lazy PRM,» *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on,* vol. 1, pp. 521-528, 2000.

[66] S. M. LaValle y J. J. Kuffner Jr., «Rapidly-Exploring Random Trees: Progress and Prospects,» *Algorithmic and Computational Robotics : New Directions,* pp. 293-308, 2000.

[67] J. Kuffner, S. U. C. U. Dept. of Comput. Sci. y S. LaValle, «RRT-connect: An efficient approach to single-query path planning,» *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on,* vol. 2, pp. 995-1001, 2000.

[68] K. R., «Rapidly-exploring Random Graphs: Motion Planning of Multiple Mobile Robots,» *Advanced Robotics,* vol. 27, nº 14, pp. 1113-1122, 2013.

[69] S. Karaman, M. I. o. T. C. M. U. Lab. for Inf. & Decision Syst. y E. Frazzoli, «Optimal kinodynamic motion planning using incremental sampling-based methods,» *Decision and Control (CDC), 2010 49th IEEE Conference on,* pp. 7681-7687, 2010.

[70] G. Sánchez y J.-C. Latombe, «A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking,» *Springer Tracts in Advanced Robotics ,* vol. 6, pp. 403-417, 2003.

[71] W. Zeng y R. L. Church, «Finding shortest paths on real road networks: the case for A*,» *International Journal of Geographical Information Science,* vol. 23, nº 4, pp. 531-543 , 2009.

[72] ONU - Organización de Naciones Unidas, «Foro Mundial del Agua,» Marsella, 2012.