ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Estudio e implementación de algoritmo de generación de mapas de profundidad a partir de imágenes estéreo

(Study and implementation of depth map from Stereo Matching algorithm)

Para acceder al Título de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Juan Antonio Fernández de la Granja

Octubre - 2017

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Juan Antonio Fernández de la Granja
Director del TFG: Pablo Pedro Sánchez Espeso
Título: Estudio e implementación de algoritmo de generación de mapas de profundidad a partir de imágenes estéreo.
Title: "Study and implementation of depth map from Stereo Matching algorithm"

Presentado a examen el día: 30 de octubre de 2017

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Sánchez Espeso, Pablo Pedro
Secretario (Apellidos, Nombre): Ruiz Lombera, Rubén
Vocal (Apellidos, Nombre): Ugarte Olano, Iñigo

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Trabajo Fin de Grado Nº
(a asignar por Secretaría)

Agradecimientos

Quiero agradecer a mi familia, especialmente a mis padres, por haberme apoyado incondicionalmente todos estos años, haciendo posible que haya llegado a donde estoy ahora. También quiero agradecer especialmente a mis abuelos toda la ayuda y apoyo proporcionados estos 4 años de Titulación que me han acogido en su vivienda.

Por supuesto, a mi tutor, Pablo, por darme la oportunidad de realizar este trabajo sobre una temática que realmente despierta mi interés, por la confianza, las horas depositadas en mí, y todos los conocimientos proporcionados no sólo durante este TFG, si no durante los últimos dos años de la carrera. No puedo olvidarme de otros profesores del departamento como Iñigo Ugarte o Álvaro Díaz, por ofrecer su ayuda de forma desinteresada.

Y por último a mi pareja y amigos, por poder contar siempre con ellos y ser un apoyo fundamental.

Resumen

En este trabajo, se presenta un estudio del algoritmo *Stereo Matching* y un análisis teórico de las técnicas más utilizadas para su implementación. Además, se desarrolla dicho algoritmo en C++ para obtener mapas de profundidad a partir de una escena con imágenes estéreo en tiempo real. Estos mapas de profundidad representan modelos 3D sencillos de interpretar y sirven como punto de partida para reconstrucciones 3D más complejas. Finalmente, se analizan los resultados obtenidos y se plantean opciones de mejora en cuanto al rendimiento del algoritmo y a la calidad de los mapas de profundidad, con el fin de ayudar a alcanzar mejores implementaciones futuras del mismo en tiempo real utilizando sistemas embebidos.

Palabras clave: *Stereo Matching*, Modelo 3D, Visión Éstereo (*Stereo visión*), Visión binocular, Macrobloque, *frame*, Calibración de cámara, Geometría Epipolar, Rectificación, Triangulación, Correspondencia, Coste, Mapa de Disparidad (*Disparity map*), Mapa de Profundidad (*Depth map*), *Matching*.

Abstract

This document presents a study of the *Stereo Matching* algorithm, with a theoretical analysis of all the techniques used on its development. After that, a C++ implementation is suggested in order to obtain real-time Depth maps from Stereo Cameras. Finally, the results of the implementation are evaluated, so improvements regarding reconstruction quality and performance can be proposed with the purpose of improving future implementations in embedded systems.

Keywords: Stereo Matching, 3D Model, Stereo vision, Binocular Vision, Window, Frame, Camera Calibration, Epipolar Geometry, Rectification, Triangulation, Correspondence, Cost, Disparity map, Depth map, Matching.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1
Introducción:

1.1 Motivación

En los últimos años, la tecnología en tres dimensiones (3D) han ido ganando popularidad entre la mayoría de la población. El interés despertó con las películas en tres dimensiones alrededor del año 2003. Según crecía la popularidad de esta tecnología, los dispositivos 3D fueron introduciéndose en el mercado en forma de juegos y otras aplicaciones tanto de ocio como profesionales. El uso de las impresiones en 3D ha mejorado significativamente el ámbito de la industria y el diseño. La mejora de los modelos 3D, concretamente de los modelos realísticos de reconstrucción 3D, han sido la clave para dar lugar a estas mejoras y hacer crecer la popularidad de la tecnología 3D.

Durante las últimas décadas, se han desarrollado multitud de métodos en el ámbito de la reconstrucción 3D, los cuales se pueden dividir en dos clases: métodos activos y pasivos. Los métodos activos interactúan con el objeto a reconstruir, sea de forma mecánica o radiométrica. Un ejemplo común de un método mecánico activo sería el uso de un calibre para medir las dimensiones de un objeto en rotación, cubriendo así todas sus perspectivas. También se puede realizar el mismo proceso utilizando un láser que interfiera con el objeto. Por otro lado, los objetos pasivos no interaccionan con el objeto a reconstruir, se usa un sensor que mida la radiación reflejada o emitida por la superficie del objeto para conocer su estructura 3D. El ejemplo de método pasivo más básico es grabar o tomar desde varias perspectivas imágenes del objeto, contra el cual incide luz visible, para obtener como salida un modelo 3D del mismo.

Los métodos pasivos son los más empleados ya que resultan más sencillos y más baratos de implementar. Sólo son necesarias una o dos cámaras, rara vez es necesario un sistema de varias cámaras, con las que obtener así la secuencia de imágenes que nos dé información 3D del objeto. En el caso de un sistema con una única cámara, sería necesario tomar imágenes consecutivamente

alrededor del objeto. El principal inconveniente de este método es que el camino que siga la cámara alrededor del objeto debe ser conocido para facilitar el proceso de reconstrucción. Sin embargo, a día de hoy existen métodos que son capaces de implementar el modelo 3D bajo un movimiento libre de la cámara, como por ejemplo "*Structure from motion*, (*SFM*)". Normalmente este método se emplea para reconstruir objetos estáticos. La estructura básica de un sistema de *Structure from motion* se presenta en la figura 1.1.

Este sistema se puede dividir en tres partes: (1) calibración de la cámara, que obtiene los parámetros de dicha cámara a partir de las vistas disponibles, (2) *Stereo Matching*, que busca una correspondencia entre píxeles de las diferentes imágenes y calculan la profundidad de la escena, y (3) modelación 3D, que reconstruye el objeto a partir de los mapas de profundidad.

También existen métodos pasivos de reconstrucción 3D que usan dos cámaras o una cámara estéreo. En este caso, se poseen dos perspectivas diferentes de cada punto de la escena, y existe una relación de desplazamiento entre dichas perspectivas. Para buscar la correspondencia entre estos puntos en los sistemas de doble cámara se vuelve a utilizar *Stereo Matching*. El valor de discrepancia entre las diferentes perspectivas de cada cámara se traduce en una disparidad entre puntos gracias a *Stereo Matching*. Esta disparidad es inversamente proporcional a la distancia de dichos puntos con respecto a las cámaras. A continuación, es posible calcular la profundidad de todos los puntos de la escena gracias a un proceso de triangulación. Los sistemas con cámaras estéreo son capaces de obtener la profundidad de los objetos en escenas dinámicas con movimiento. Antes de computar la disparidad, puede ser necesario un proceso de calibración de las dos cámaras, aunque es común que si se usa una cámara estéreo este paso no sea necesario.

Por otro lado, los sistemas multi-cámara (más de dos), que ofrecen múltiples vistas de los objetos en el mismo momento de tiempo, son capaces de reconstruir cuerpos dinámicos en 3D ya que se tienen múltiples perspectivas del objeto en dicho momento. Esto es algo que no ocurre en los sistemas de dos cámaras, donde sólo se puede obtener información de la profundidad de la escena. El *modus operandi* en los sistemas multi-cámara también se puede dividir en tres pasos: calibración de las cámaras, *Stereo Matching* y modelación 3D. Tras la calibración de las cámaras, se computan los mapas de profundidad de las diferentes vistas utilizando *Stereo Matching* y finalmente se obtiene, a partir de las profundidades, un modelo 3D de buena calidad.

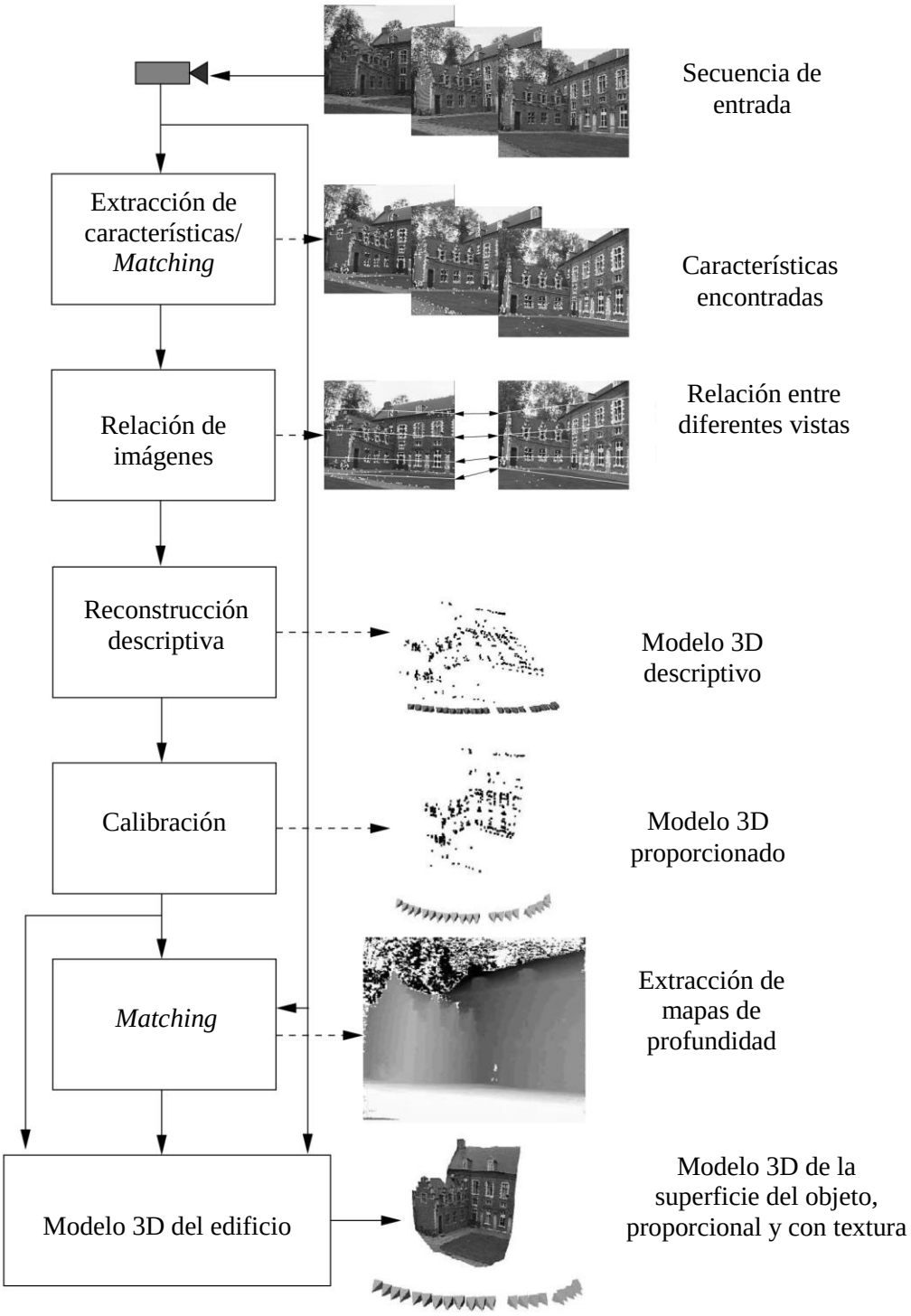


FIGURA 1.1: SISTEMA DE RECONSTRUCCIÓN 3D BASADO EN IMÁGENES PRESENTADO EN [5].

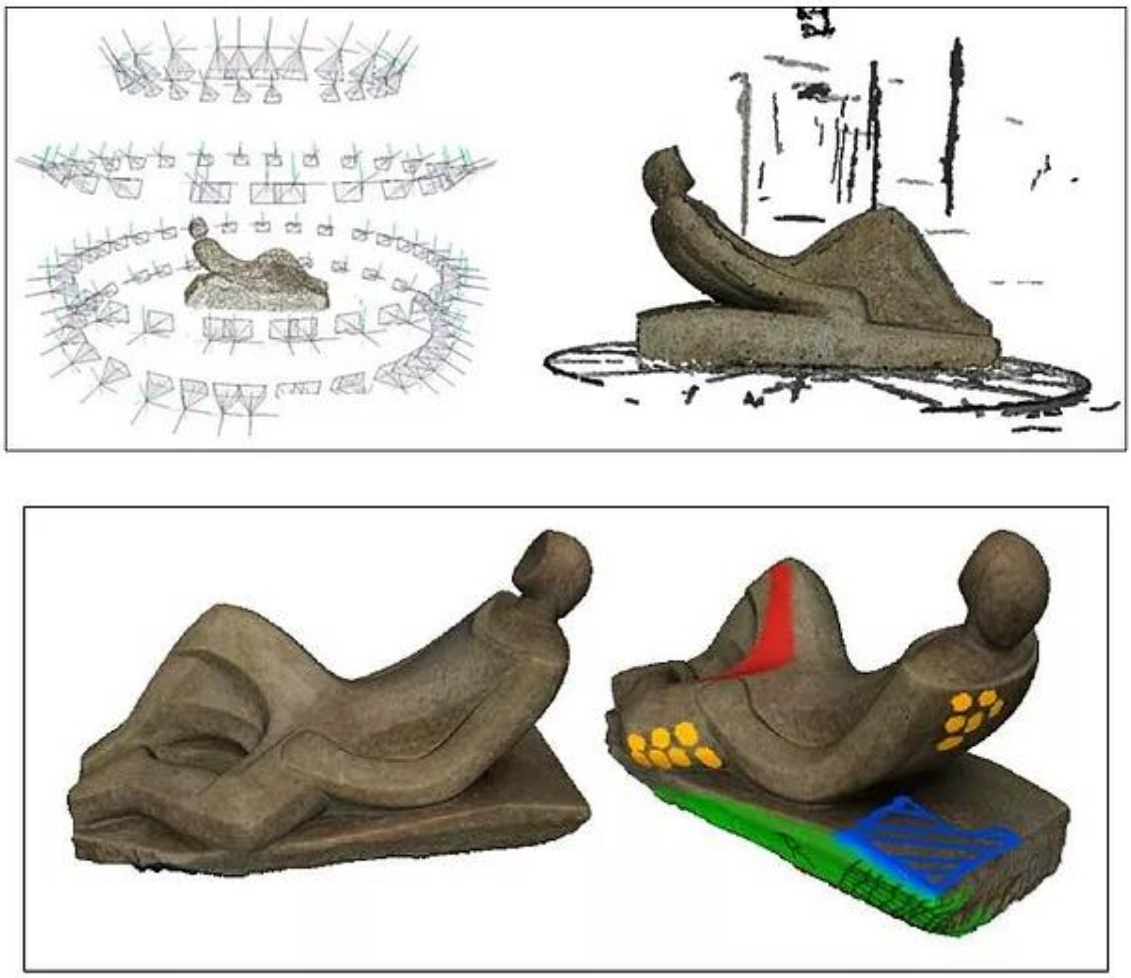


FIGURA 1.2: EJEMPLO DE RECONSTRUCCIÓN A PARTIR DE UN SISTEMA MULTICÁMARA, DESTINADO A LA RECONSTRUCCIÓN DE OBJETOS ESTÁTICOS. [23]

El método *Stereo Matching* resulta una tarea común y fundamental para cualquier tipo de sistema de reconstrucción 3D, sea cual sea el nº de cámaras utilizadas en el sistema o si se trata de una escena estática o en movimiento. Se puede observar que el interés principal reside en encontrar la correspondencia adecuada entre los puntos de la escena y *Stereo Matching* resuelve este problema. No resulta una tarea sencilla, pero desde hace varios años se viene buscando la implementación óptima para este *Stereo Matching*. Sin embargo, la búsqueda de los mismos puntos en diferentes imágenes sería demasiado complicado sin antes preparar las imágenes para aumentar nuestras probabilidades de éxito. En la mayoría de los sistemas de reconstrucción 3D también es necesario un proceso de calibración de las cámaras. Ambos pasos resultan procesos fundamentales, pero a la vez poco triviales en la reconstrucción 3D [5] [6]. Se puede decir que son pasos intuitivos, pero difíciles de imaginar cómo abordarlos. Es por eso que, en este trabajo, se va a realizar una implementación sencilla de un sistema pasivo con cámara estéreo de reconstrucción 3D, donde estas tareas resultan básicas en un campo tan popular hoy en día, pero a la vez conocido a fondo por sólo unos pocos.

1.2 Objetivos

En este trabajo, se pretende analizar el algoritmo de correspondencia estéreo *Stereo Matching*, que supone un método sencillo de obtención de información 3D de una escena, un campo que está a la orden del día en el ámbito del procesado de imagen y de la reconstrucción 3D. Se tratará previamente el marco teórico que envuelve el algoritmo y se analizarán las técnicas y principios fundamentales del mismo con cierto grado de implicación. Se pretende así proporcionar las claves necesarias para entender el desarrollo de la técnica *Stereo Matching*.

También se desarrollará una implementación de un sistema de *Stereo Matching*, con la que se espera obtener mapas de profundidad a partir de imágenes estéreo. El enfoque que se planteará de este algoritmo irá enfocado a obtener un sistema que ofrezca mapas de profundidad en tiempo real. El desarrollo de *Stereo Matching* en tiempo real podría abarcar un abanico de aplicaciones considerable, sobre todo en los ámbitos de la Inteligencia Artificial en la robótica y de la conducción automática en el campo de la automoción.

Durante el proceso de implementación, se utilizará el popular lenguaje de programación orientado a objetos C++, el cual no tiene recogido su enseñanza en ninguna de las guías docentes de las asignaturas ofertadas en el Grado de Tecnologías de Telecomunicación de la Universidad de Cantabria. De esta forma, se adquirirán competencias de exploración y documentación dentro de un lenguaje desconocido por parte del alumno para la implementación de un sistema complejo.

1.3 Estructura del trabajo

El desarrollo de este TFG se divide en cuatro capítulos principales:

- **Capítulo 2**: se presenta el estado del arte del algoritmo *Stereo Matching*, donde se mencionan las principales aplicaciones de este método, la teoría que envuelve al algoritmo y una explicación de la visión estéreo y sus fundamentos.
- **Capítulo 3**: se realiza una explicación exhaustiva de los pasos y principios que componen *Stereo Matching* y se exponen las técnicas más populares para su implementación. De la misma forma, se señalan las técnicas que se van a tener en cuenta en el desarrollo posterior del sistema en este TFG.
- **Capítulo 4:** se propone una implementación del algoritmo *Stereo Matching* orientada a establecer una salida en tiempo real, utilizando técnicas ligeras computacionalmente pero que proporcionen un buen grado de exactitud.
- **Capítulo 5:** se exponen las principales conclusiones obtenidas del desarrollo del trabajo.

Capítulo 2
Estado del arte:

2.1 Aplicaciones de la visión estéreo, *stereo matching* y los mapas de profundidad

Las aplicaciones relacionadas con el mundo de la reconstrucción 3D están ganando cada vez más popularidad en el día a día: películas y juegos 3D, impresión 3D, mapas de navegación 3D, reconocimiento de objetos, etc. Muchas de estas aplicaciones requieren de modelos 3D realistas y completos. Sin embargo, los mapas de profundidad que se esperan obtener en este trabajo gracias a la técnica de *Stereo Matching* no nos ofrecen toda la información necesaria para implementar un modelo completo que sea apto para una reconstrucción 3D. A pesar de ello resulta información útil para llegar a dicha reconstrucción, pudiendo afirmar que los mapas o modelos de profundidad son el paso previo para un modelo de reconstrucción 3D. Por otro lado, los mapas de profundidad tienen otras aplicaciones directas, además de la reconstrucción tridimensional, las cuales se van a tratar a continuación.

2.1.1 *Tracking* 3D

Tal y como se acaba de afirmar, los mapas de profundidad suponen un paso previo a modelos de reconstrucción 3D más completos. Con estos mapas de profundidad se obtiene información de las posiciones relativas en el espacio de todos los objetos de la escena. El termino *tracking* de imagen hace referencia a la identificación y seguimiento de determinados puntos dentro de una imagen que cumplan determinadas características. *Stereo Matching* trata la imagen en unidad de píxeles y su finalidad es buscar los píxeles en la otra imagen del par de imágenes. Mientras tanto, el *tracking* de imagen utiliza información previa para identificar partes completas dentro de una misma imagen, y así aislarlas y su posición a lo largo de una secuencia de imágenes.

El *tracking 3D* utiliza los mapas de profundidad obtenidos con *Stereo Matching* para refinar y mejorar la información de la escena obtenida a partir de las imágenes. Es capaz de ofrecer

información de objetos concretos en la imagen, pudiendo identificarlos por separado dentro del mapa [29]. Dicha información se utiliza para conocer la trayectoria y posición de estos objetos cuando se tiene indicios de que están en movimiento. Para ello son necesarios varios mapas de profundidad en diferentes momentos de tiempo. El tracking 3D se utiliza para aplicaciones de reconstrucción 3D de objetos, seguridad y vigilancia donde se necesitan conocer las posiciones en el espacio de individuos en la escena, y otras muchas más aplicaciones.

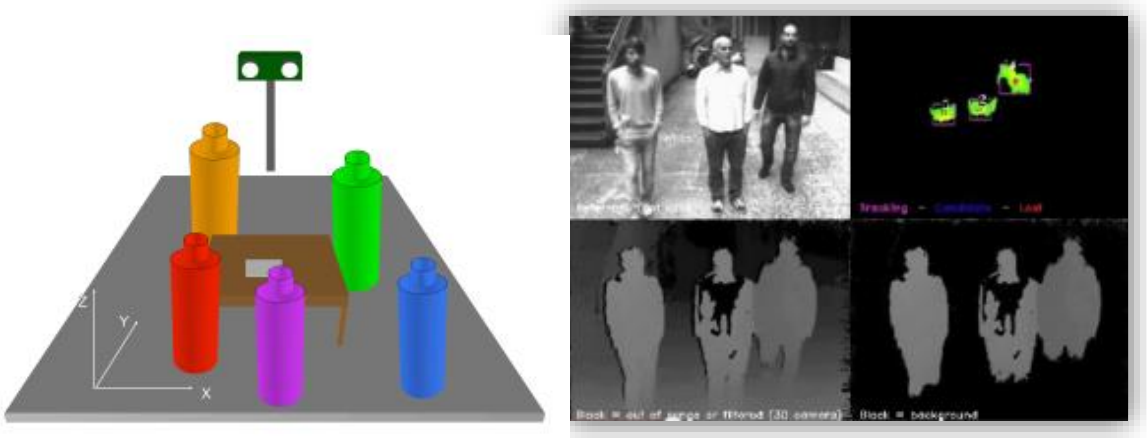


FIGURA 2.1: EJEMPLO DE *TRAKING* 3D, DONDE SE LOCALIZA LA POSICIÓN DE PERSONAS. FUENTE: GOOGLE

2.1.2 SLAM Y SFM

En este caso, vuelve a ocurrir que los mapas de profundidad obtenidos con *Stereo Matching* proporcionarán información 3D adicional a las imágenes tomadas para técnicas más sofisticadas de modelado y reconstrucción 3D. La técnica SLAM tiene como objetivo la resolución simultánea de: (1) la localización y mapeado de elementos en el entorno, es decir, la resolución de las características tridimensional de una escena, y proporcionar información sobre las posiciones desde la que se toman las imágenes. SLAM intenta ofrecer una estructura tridimensional en tiempo real empleando información que proviene de distintos sensores o sistemas, siendo los mapas de profundidad uno de estos sistemas de entrada.

Existe otra variante, el vSLAM (*visual SLAM*), que emplea únicamente imágenes para resolver la reconstrucción 3D y obtener la posición. Este método es una aplicación de la técnica mencionada en la introducción de este trabajo *Structure from Motion (SFM)*. Sin embargo, esta última es capaz de ofrecer una resolución en gran medida más completa que SLAM, aunque más difícil de implementar en tiempo real.

2.1.3 Conducción automática

Esta aplicación abarca varios tipos de sistemas en los que se puede implementar, desde pequeños robots hasta grandes vehículos, pero todos motorizados. El ámbito de la conducción automática en automóviles resulta un tema de gran interés hoy en día, y que los vehículos sean capaces de guiarse y moverse por las vías de forma autónoma supone el futuro en el campo automovilístico.

Dotando al robot, dron o vehículo de visión estéreo, es posible analizar las disparidades de los diferentes planos y objetos del camino o vía, y así poder reaccionar de forma adecuada ante ellos. Para conseguir esto, dichos sistemas se valen de la técnica *Stereo Matching*. De esta forma, permite obtener información de las profundidades de la escena, así como identificar objetos en la misma y sus posiciones.

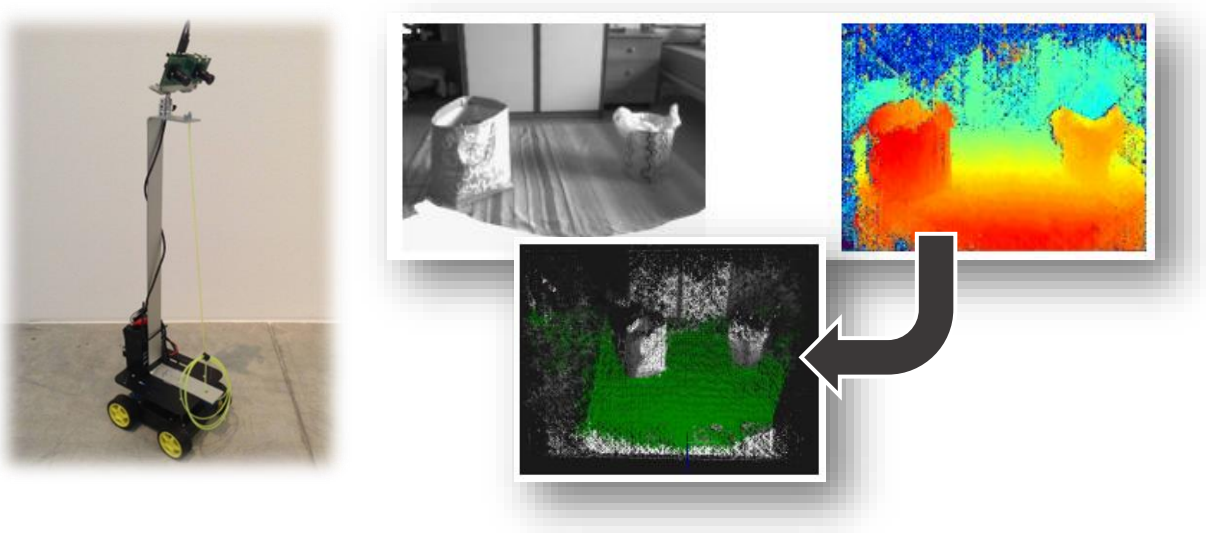


FIGURA 2.2: EJEMPLO DE MAPA DE PROFUNDIDAD GENERADO POR EL SISTEMA ACOPLADO A UN ROBOT CAPAZ DE MOVERSE CON AUTONOMÍA GRACIAS A *STEREO MATCHING*. FUENTE: GOOGLE

Hoy en día solo se pueden encontrar prototipos de grandes aparatos que dispongan de conducción automática. Aún no existe el desarrollo suficiente en esta técnica como para llevar al mercado dichos sistemas, ya que requieren de una fiabilidad máxima y de un procesado en tiempo real. La técnica vSLAM antes mencionada también es empleada en el guiado de vehículos autónomos, drones y robots; aunque, al igual que los mapas de profundidad, no está lo suficientemente desarrollado como para integrarse en productos comerciales.

2.1.4 Asistencia a la movilidad de personas con problemas de visión

Esta iniciativa ha sido expuesta en uno de los canales principales de la televisión italiana [30]. Se han desarrollado implementaciones de sistemas que procesan la información obtenida por los mapas de profundidad para ayudar a personas con dificultades en la visión a moverse de forma autónoma por su ambiente y desarrollar un estilo de vida normalizado.

La forma en la que se emplean los mapas de profundidad en estos sistemas es la misma que en los sistemas de conducción automática, se analizan los datos recogidos por una cámara estéreo, elaborando los mapas de profundidad con la finalidad de identificar la profundidad de objetos de la escena. La reacción ante los objetos se regirá por las decisiones que tome el cerebro humano, que ha sido avisado mediante señales auditivas y táctiles de la distancia a objetos de la escena que puedan interferir con el individuo.

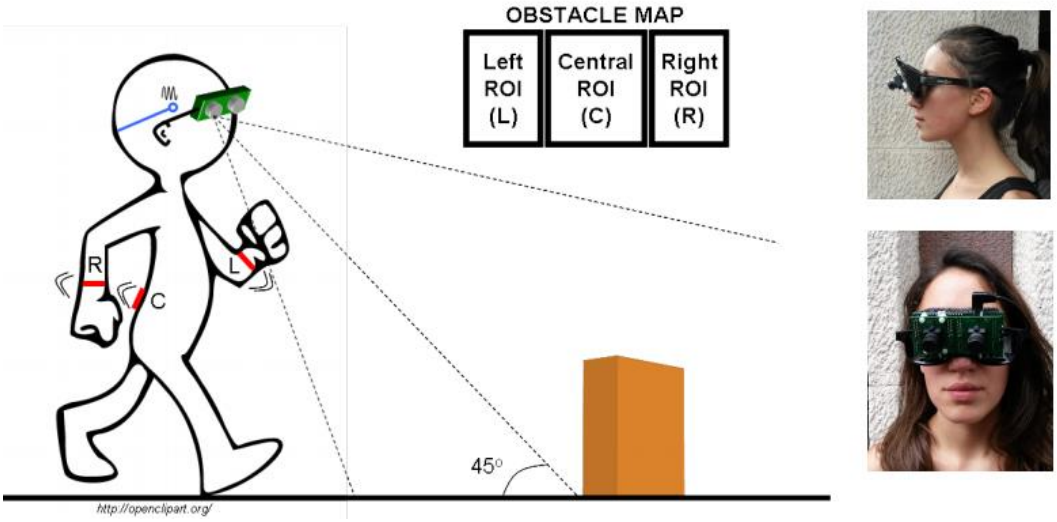


FIGURA 2.3: MUESTRA DE UN SISTEMA DE CÁMARAS ESTÉREO QUE IMPLEMENTAN *STEREO MATCHING* Y ASISTEN A PERSONAS CON PROBLEMAS DE VISIÓN. FUENTE: GOOGLE

2.2 La visión binocular humana como base de la visión estéreo

Tras leer la introducción de este trabajo, se puede afirmar que, para obtener profundidad de una escena, es fundamental disponer, como mínimo, de dos imágenes de un objeto desde diferentes perspectivas o de dos cámaras tomando imágenes al mismo tiempo desde diferentes ángulos. En esta implementación de un sistema de visión estéreo, se utilizarán dos cámaras como cámara estéreo para la toma de imágenes de la escena. Dichas cámaras serán los *ojos* de nuestro sistema de reconstrucción 3D, donde cada una tomará imágenes desde una perspectiva diferente. De esta forma, una cámara estéreo funciona de manera análoga a los ojos humanos. Esto es así ya

que el diseño de las cámaras estéreo está basado en un modelo cuya base teórica se toma de la visión humana.

El ojo humano ayudado de un proceso cognitivo es capaz de percibir profundidad. Aún se desconoce el mecanismo que utiliza nuestro cerebro para interpretarla y no hay duda de que se trata de un proceso complejo que es objeto de estudio en el ámbito de la neurología. Es por eso que en este trabajo sólo se mencionará lo básico sobre este tema. Aunque una visión dual resulta muy útil en la tarea de estimación de profundidad o *Depth Estimation*, éste no es el único método válido. El conocimiento *a priori* de las dimensiones de los objetos o ver como dos líneas de paralelas convergen en la lejanía son ejemplos que nos ayudan a obtener sensación de profundidad. Sin embargo, guiarnos por los patrones de estos dos ejemplos resultaría demasiado complejo si se quiere obtener una implementación de *Depth Estimation* de forma computacional.

Hawkins [1] identificó algunos fenómenos visuales dentro del campo de la *Estereoscopía*. Las dos imágenes percibidas por ambos ojos, izquierdo y derecho, se funden en una única imagen que contiene información de profundidad no cuantificada. Si esta fusión de las imágenes falla, se puede dar lugar a la Diplopía, produciendo dos imágenes "fantasma" del objeto, sensación bien conocida por los Astigmáticos. La Diplopía se produce cuando la visión ocular no es capaz de converger en un mismo objeto, como por ejemplo cuando lo tenemos demasiado cerca de los ojos como para poder enfocarlo. A continuación, se muestra la explicación gráfica de dicho fenómeno:

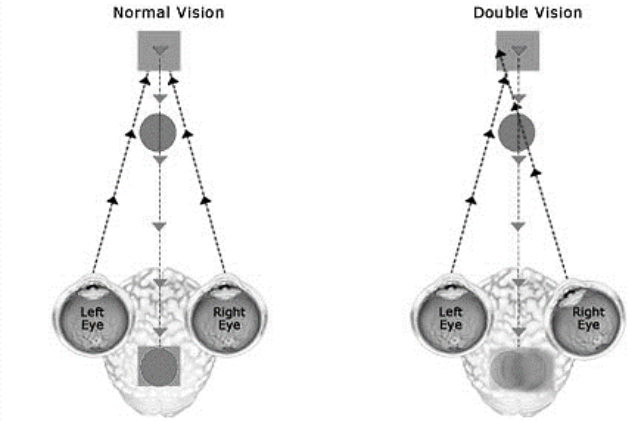


FIGURA 2.4: EXPLICACIÓN GRÁFICA DEL FENÓMENO DE DIPLOPÍA. FUENTE: GOOGLE

Por último, la variable principal para la visión 3D es el Paralaje. Este concepto hace referencia a la diferencia de posición de un objeto en la escena cuando existen varias vistas desde diferentes perspectivas, y se usa para localizar los diferentes objetos de la imagen con respecto al plano de fondo o de visión. Bruder et al., [2] explican cómo se puede obtener una imagen 3D virtual donde los objetos se crean a partir del plano de visión (cero paralaje), tratándose de paralaje

positivo cuando se crean más allá de dicho plano o paralaje negativo cuando se crean más cerca de dicho plano.

El efecto del paralaje en la visión humana provoca que, cuando la visión enfoca a un objeto, éste pasa a ocupar el plano de visión. A partir de aquí, si nos desplazamos lateralmente con

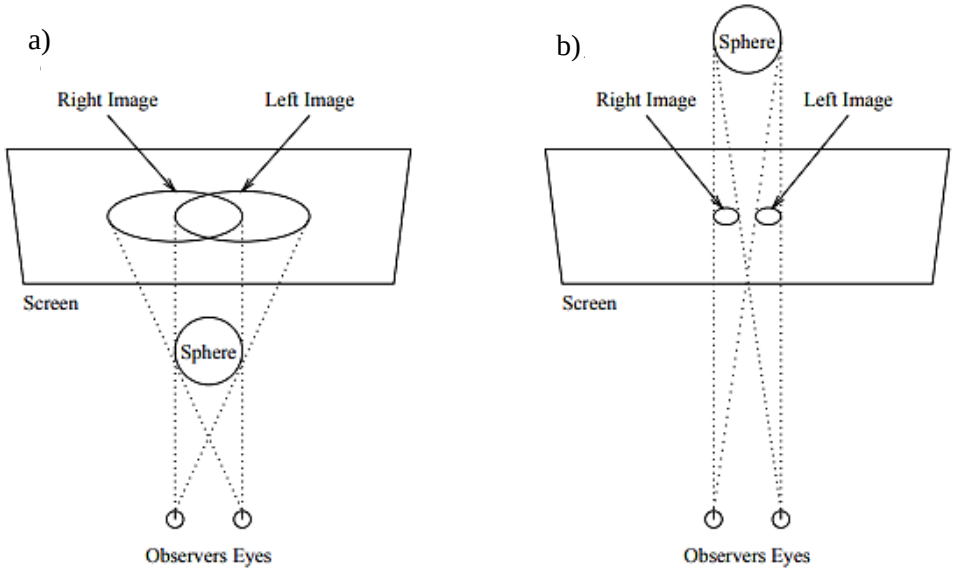


FIGURA 2.5: DISPARIDAD EN LA VISIÓN ESTÉREO PRODUCIDO POR A) PARALAJE NEGATIVO Y B) PARALAJE POSITIVO [1]

respecto al objeto, se puede percibir como los objetos con paralaje positivo se mueven en el mismo sentido de desplazamiento, y, por el contrario, tenemos la sensación de que los objetos con paralaje negativo se desplazan en el sentido opuesto. Además, estos objetos se desplazan a velocidades diferentes, es decir, cuanto mayor paralaje por parte del segundo objeto, sea paralaje negativo o positivo, más rápido se desplazará en el eje horizontal, sea o no en el mismo sentido de desplazamiento de nuestra visión.

Esto se refleja en el siguiente fenómeno: Si se observa la escena con un ojo cerrado se tiene una perspectiva concreta de los objetos que la componen. Si, a continuación, se cierra dicho ojo y se abre el otro, se observa que los objetos de la escena se desplazan instantáneamente en el eje horizontal, sea en sentido negativo o positivo. Este compendio de ideas estudiadas por la Estereoscopía bajo el nombre de paralaje describen los procesos empleados por el cerebro humano para crear la sensación de profundidad en la visión humana y suponen la base de los mapas de profundidad.

A partir de este punto, cuando se hable de visión estéreo en este trabajo, se referirá a las cámaras estéreo o unión de dos cámaras como entradas del sistema de reconstrucción 3D. Utilizando *Stereo Matching*, es decir, identificando la posición de los elementos de una imagen en la otra y obteniendo su desplazamiento se tratará de traducir el paralaje en mapas de

profundidad 3D, convirtiendo así a *Stereo Matching* en el proceso cognitivo de nuestro sistema de reconstrucción capaz de producir información 3D.

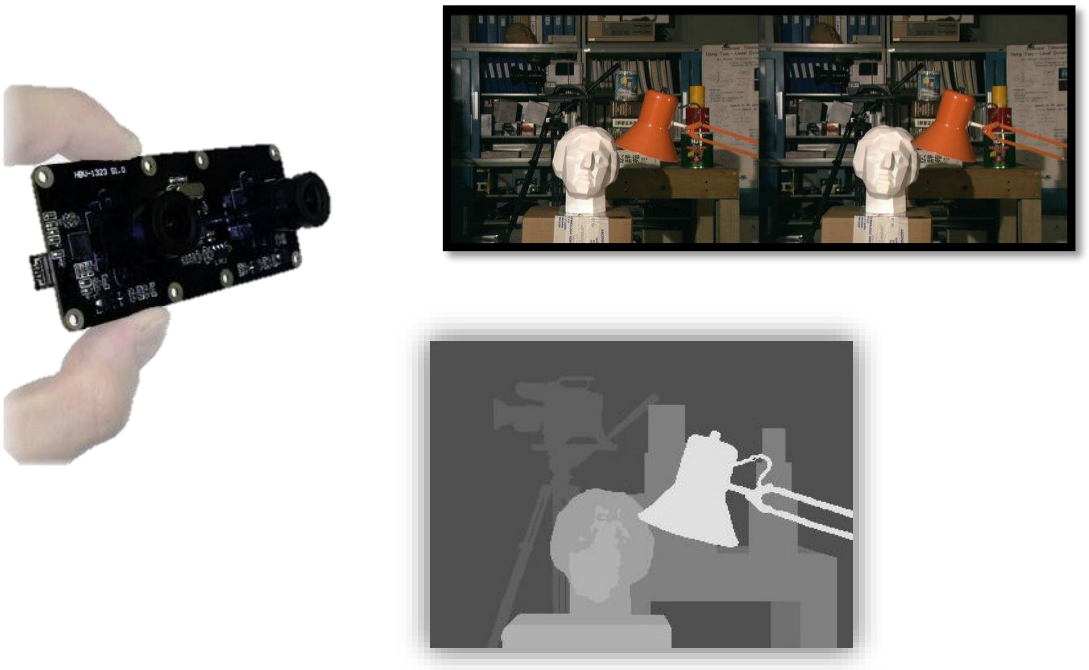


FIGURA 2.6: A LA IZQUIERDA, SISTEMA TÍPICO DE CÁMARA ESTERO. A LA DERECHA, EJEMPLO DE *STEREO MATCHING* A PARTIR DE LAS IMÁGENES DE TSUKUBA UTILIZANDO *GROUND TRUTH*[1].

2.3 Las Cámaras Estéreo y sus fundamentos

En esta parte se presentará la teoría fundamental a tener en cuenta cuando hablamos de cámaras estéreo y visión estéreo. Se tratará el modelo de cámara fundamental *Pinhole Camera*, la geometría epipolar, la rectificación de imagen, el *Stereo Matching*, y la Triangulación de las cámaras. La siguiente información se dará a un nivel general e introductorio dentro del campo, por lo tanto, la lectura de estos apartados no resulta trascendental para aquellas personas familiarizadas con estos conceptos. Para aquellos lectores interesados en estos aspectos se les referencian algunas fuentes donde podrán ampliar su perspectiva sobre la visión estéreo [3] [4].

Las cámaras estéreo son un sistema de toma de imágenes sin sensores adicionales de profundidad. Tal y como se ve en la figura 2.3, está compuesta por dos lentes acopladas a una placa donde deben colocarse en el mismo eje óptico. Cuando se utilizan, es común asumir que las imágenes grabadas estarán corregidas y alineadas, sin embargo, no tiene porqué ser así.

[1] El famoso método *Ground Truth* de *Stereo Matching* pertenece a la Universidad de Middlebury y está bajo patente. Se considera de los mejores algoritmos en cuanto a mapas de profundidad. En este trabajo se tomará Ground Truth como el algoritmo que crea los mapas de profundidad de referencia.

Dependerá de la calidad de las mismas y del fabricante que esto ocurra o no. En *Stereo Matching* es fundamental que las imágenes de entrada estén alineadas y sin distorsión, por lo que en la mayoría de los casos es necesario calibrar las cámaras. Este paso se vuelve muy recomendable cuando se utilizan dos cámaras individuales acopladas para suplir las cámaras estéreo. El esquema general de un sistema de visión estéreo, objeto de ser implementado en este TFG, se ilustra en la Figura 2.7.

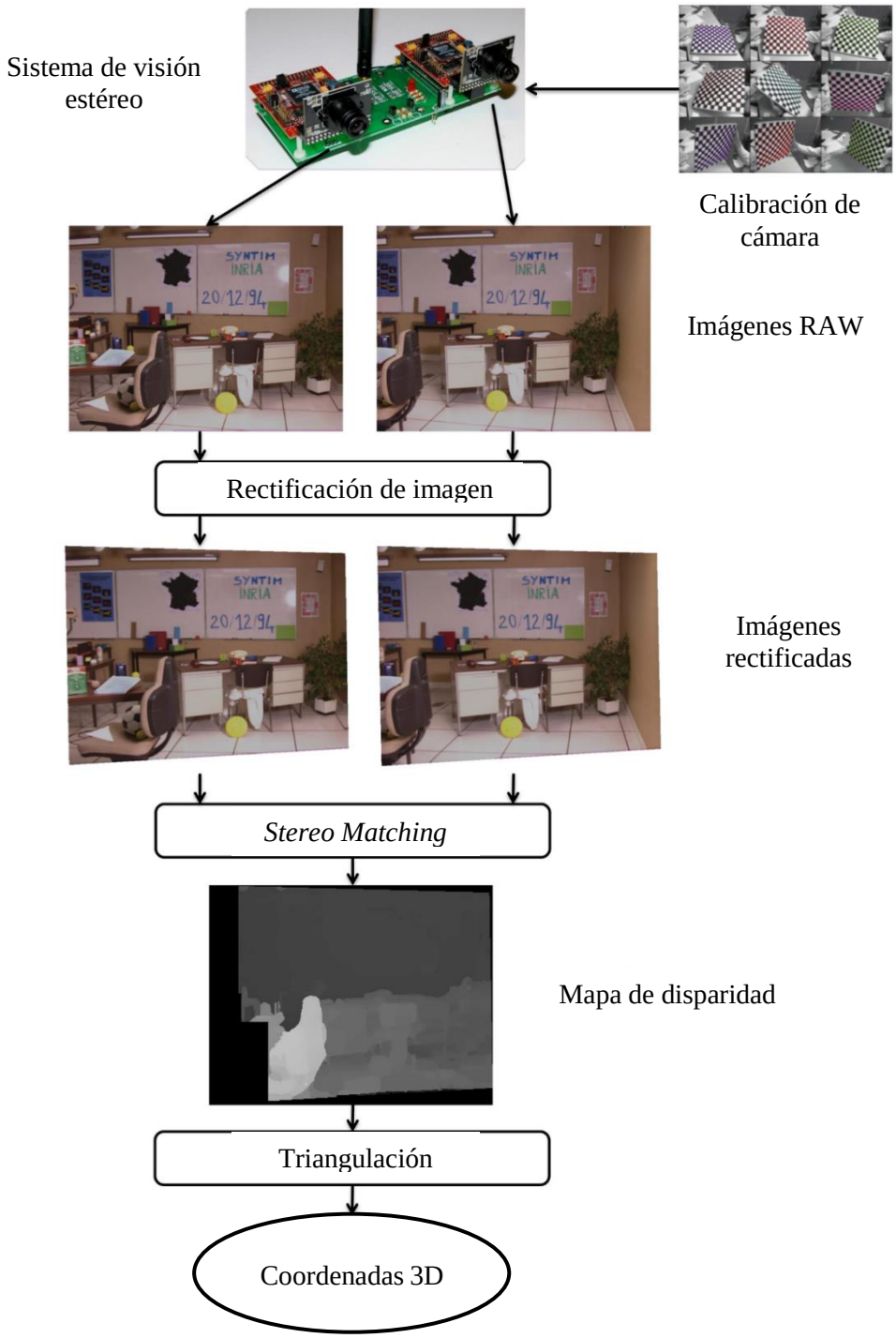


FIGURA 2.7: ESQUEMA GENERAL DE UN SISTEMA DE VISIÓN ESTÉREO QUE OBTIENE MAPAS DE PROFUNDIDAD.

Como se ha mencionado en el apartado 2.2, la idea de visión estéreo fue motivada y se hizo a semejanza del sistema de visión humano, que es capaz de percibir la profundidad de la escena. La visión dual de los humanos ofrece dos imágenes que son procesadas por el cerebro para hacernos llegar información 3D del ambiente. Ya se debatió anteriormente algunos de los métodos por los cuales somos capaces de percibir esta profundidad. De forma análoga a la visión binocular humana, un sistema de visión estéreo se vale de dos cámaras (dos ojos) que capturan simultáneamente la misma escena desde diferentes perspectivas, y serán los componentes *Hardware*. Por otra parte, el *software* del sistema estéreo llevará a cabo la función del "cerebro" de dicho sistema al computar la información 3D desde estas dos vistas.

A continuación, se va a explicar de forma secuencial las partes de este sistema de visión estéreo (figura 2.7). En primer lugar, un punto de la superficie de un objeto es proyectado sobre los planos de las dos cámaras respectivamente. Esta proyección de un punto 3D a los pixeles 2D se puede entender mediante el modelo de *Pinhole Camera*, que se explica en el apartado 2.3.1. La esencia de este modelo de cámara es una Matriz de Proyección de dimensiones 3x4, cuyas componentes se basan en parámetros intrínsecos y extrínsecos de la cámara.

Después de obtener dichos parámetros, el problema inherente de la visión estéreo empieza a salir a la luz, la correspondencia entre píxeles; es decir, establecer donde se encuentra cada parte o pixel de una imagen en la otra. Este proceso de búsqueda 2D será parte importante de este trabajo. Se podrá ver que dicha tarea es modelable como una búsqueda en una única dimensión (1D) gracias a la restricción de la Geometría Epipolar, introducida en el capítulo 2.3.2, donde se buscarán correspondencias en la misma línea epipolar. Además, este problema de correspondencia se verá simplificado por la técnica de Rectificación de Imagen, introducida en el apartado 2.3.3.

Tras esta rectificación, se finaliza un pequeño proceso de calibración entre pares de cámaras. La correspondencia entre pares se realizará en la misma línea epipolar que será también totalmente horizontal, a la misma altura, ya que ambas imágenes se habrán situado en el mismo plano y en el mismo eje óptico. Tras esto, se procede a realizar un exhaustivo *Stereo Matching*, que será introducido ahora, pero tratado con mucha más profundidad en el capítulo 3. La salida del módulo de *Stereo Matching* será un Mapa de Profundidad o *Depth Map*, que se basará en la diferencia de posición entre la localización de un pixel de una imagen en la otra del mismo par. Como último paso, se obtendría la coordenada 3D de cada pixel del Mapa de Profundidad utilizando el método de Triangulación, explicado de forma sencilla en el apartado 2.3.5. En este esquema, se considera la parte de *Stereo Matching* como la más trascendental del proceso.

2.3.1 Modelo de cámara Pinhole:

El modelo de cámara *Pinhole* representa una relación matemática entre las coordenadas de un punto 3D y su proyección 2D en el plano de la imagen. Esta relación matemática se representa en la figura 2.8. El centro de la cámara, punto C, donde se sitúa la apertura de la cámara, representa el origen de un sistema de coordenadas Euclideo. El eje Z, que apunta a la dirección de visión de la cámara, se tomará como eje principal del espacio. El plano de la imagen será paralelo a los ejes X e Y, y se situará a una distancia f del centro de la cámara C en dirección al eje Z.

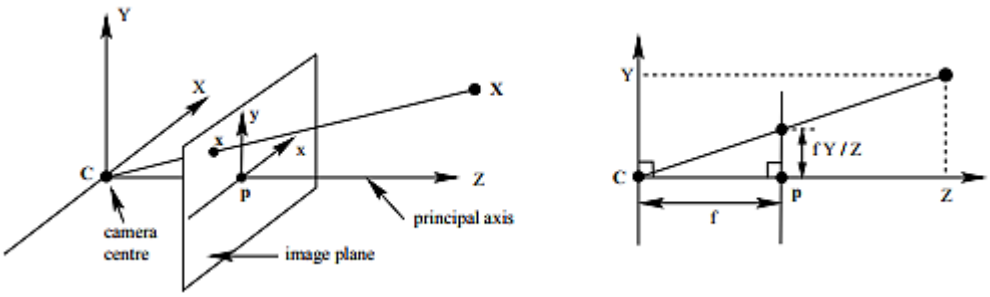


FIGURA 2.8: GEOMETRÍA DEL MODELO DE CÁMARA *PINHOLE*.

En la figura, el plano de imagen se observa que se encuentra a una distancia $Z = f$, donde f hace referencia a la distancia focal. El punto p, intersección entre el plano de la imagen y el eje Z será el punto principal de la imagen, coincidiendo con el centro de la misma. Dado un punto X en el espacio 3D se define su coordenada homogénea como $(X, Y, Z, 1)'$. Este punto es proyectado al plano de imagen en el punto x, que será el punto donde la línea que une la coordenada X con el centro C de la cámara intersecta con el plano de imagen. Utilizando semejanza de triángulos, la coordenada de x de puede definir como:

$$\frac{f}{Z} = \frac{x}{X} = \frac{y}{Y} \tag{2.1}$$

que queda:

$$x = \frac{fX}{Z}$$
$$y = \frac{fY}{Z} \tag{2.2}$$

Por tanto, las coordenadas homogéneas del punto X en el plano de imagen serían $(f * X/Z, f * Y/Z, 1)'$. Esta proyección se puede escribir como una multiplicación de matrices:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} fX \\ f\mathrm{Y} \\ \mathrm{Z} \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{pmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \\ 1 \end{pmatrix} \tag{2.3}$$

La expresión de arriba, ecuación 2.3, asume que el origen de coordenadas del sistema 2D es la intersección entre el eje Z y el plano de la imagen. Pero este no tiene por qué ser el centro de dicho sistema. Necesitamos escribir las coordenadas del punto x con respecto al origen 2D de coordenadas. Llamaremos a esa traslación (px, py)'. Por tanto, la coordenada x será ahora:

$$\begin{aligned} x &= \frac{fX}{Z} + p_x \\ y &= \frac{fY}{Z} + p_y \end{aligned} \tag{2.4}$$

obteniendo una expresión parecida a la ecuación 2.3,

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} fX + Zp_x \\ f\mathrm{Y} + Zp_y \\ \mathrm{Z} \end{pmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{pmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \\ 1 \end{pmatrix} \tag{2.5}$$

En la expresión de arriba, ecuación 2.5, las coordenadas tienen unidades de milímetros (mm). Necesitamos saber la resolución de la cámara en píxeles/mm. Si los píxeles son cuadrados, la resolución será idéntica para ambas direcciones, ejes X e Y. Sin embargo, para una aproximación más general, las coordenadas Euclideas pueden tener escalas desiguales en ambos ejes y se están asumiendo píxeles rectangulares con resolución mx y my pixeles/mm en ambos ejes respectivamente. Por lo tanto, para medir el punto x en pixeles, las coordenadas se deben multiplicar por mx y my respectivamente de forma que:

$$\begin{aligned} x &= m_x \frac{fX}{Z} + m_x p_x \\ y &= m_y \frac{fY}{Z} + m_y p_y \end{aligned} \tag{2.6}$$

Esto se puede expresar en una ecuación de matrices:

$$\begin{pmatrix} x \\ y \\ \mathrm{z} \end{pmatrix} = \begin{bmatrix} m_x f & 0 & m_x p_x & 0 \\ 0 & m_y f & m_y p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{pmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \\ 1 \end{pmatrix} = \begin{bmatrix} a_x & 0 & q_x & 0 \\ 0 & a_y & q_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{pmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \\ 1 \end{pmatrix} \tag{2.7}$$

Donde *ax* y *ay* representan la distancia focal de la cámara en dimensiones de píxeles para ambas direcciones X e Y. En algunos casos, si los ejes X e Y no son ortogonales entre ellos,

entonces se necesita un parámetro de *skew*. Por tanto, la matriz general que proyecta los puntos del espacio 3D al plano de imagen 2D quedaría:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} a_x & s & q_x & 0 \\ 0 & a_y & q_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.8}$$

Los parámetros (*ax*, *ay*, *s*, *qx*, *qy*) son los parámetros intrínsecos de la cámara, que abarcan la distancia focal (*ax*, *ay*), el punto principal o de referencia (*qx*, *qy*) y el coeficiente de *skew s*. Existen parámetros intrínsecos no lineales, como la distorsión de las cámaras, que también se han de tener en cuenta, sin embargo, no se pueden incluir en el modelo lineal de cámara descrito en la ecuación 2.8. Existen algoritmos que estiman el valor de estos parámetros internos no lineales.

La relación entre un punto del espacio 3D (X, Y, Z, 1) y su proyección en 2D (x, y, z) se ha realizado suponiendo que el centro de coordenadas coincide con el centro de la cámara C. En general, el centro de la cámara y el de coordenadas no tiene porqué ser el mismo punto, pero ambos puntos se pueden relacionar mediante un desplazamiento de rotación y otro de traslación, que harán referencia a los parámetros externos de la cámara (Figura 2.9). A partir de ahora, $\tilde{X}$ denota las nuevas coordenadas del anterior punto X para un caso aún más general que el de la ecuación 2.8. La relación entre ambos puntos se puede expresar de la siguiente forma:

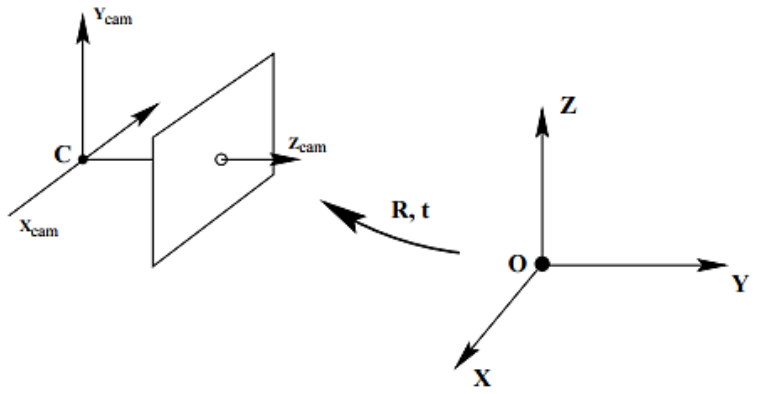


FIGURA 2.9: RELACIÓN ENTRE LAS COORDENADAS DE LA CÁMARA Y EL ESPACIO DE TRIDIMENSIONAL.

$$X = R * \tilde{X} + t \tag{2.9}$$

donde *t* representa una matriz de traslación de dimensiones 3x1 y *R* una matriz de rotación de 3x3. Esta relación se puede escribir en términos de multiplicación de matrices:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} \tilde{X} \\ \tilde{Y} \\ \tilde{Z} \\ 1 \end{pmatrix} \tag{2.10}$$

De esta forma, en un espacio general de coordenadas, el mapeo del espacio 3D a un espacio 2D vendría descrito por:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} a_x & s & q_x & 0 \\ 0 & a_y & q_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} \tilde{X} \\ \tilde{Y} \\ \tilde{Z} \\ 1 \end{pmatrix} \qquad (2.11)$$

donde la ecuación del punto *x* se puede reescribir como: $x = K[R|t]\,\tilde{X}$, siento las matrices

$$K = \begin{bmatrix} a_x & s & q_x \\ 0 & a_y & q_y \\ 0 & 0 & 1 \end{bmatrix}, \qquad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad y \quad t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

En la literatura sobre el modelo de cámara *pinhole* [7], se define la matriz de proyección de la cámara *P*, donde $P = K[R|t]$ dando lugar así a una matriz de 3x4 que contiene el modelo de cámara *pinhole*. La matriz *K* contiene los parámetros intrínsecos de la cámara y las matrices *R* y *t* guardan los parámetros extrínsecos de la misma [3], que se pueden calcular mediante un proceso muy popular de calibración de la cámara [8].

2.3.2 Geometría epipolar

El modelo de cámara *pinhole* [7] representa el mapeo de un punto del espacio 3D a un punto del plano 2D de la imagen. Los sistemas de visión estéreo incluyen dos cámaras *pinhole*, que capturan en el mismo momento de tiempo dos imágenes con perspectivas diferentes de la escena. La relación geométrica que tienen estas dos vistas se puede representar con la Geometría Epipolar. La geometría Epipolar se refiere a la geometría de proyección entre dos vistas. Se supone un punto *X* en el espacio 3D que se traduce en dos puntos x_1 y x_2 en las imágenes izquierda y derecha respectivamente. Tras realizar la calibración de la cámara, se obtienen los parámetros intrínsecos de ambas cámaras y se computan las dos matrices de proyección para cada vista P_r y P_l. A continuación, se calculan las coordenadas de los puntos x_1 y x_2 con la ecuación $x_n = P_n\,\tilde{X}$ donde $P_n = K_n[R_n|t_n]$ siendo n=1,2 en función de si se habla de la cámara izquierda o la derecha, respectivamente.

Ahora, el problema reside en encontrar la correspondencia entre ambos puntos, x_1 y x_2, de las dos imágenes, que supone la parte clave de cualquier sistema de visión estéreo. La Geometría Epipolar nos facilita la respuesta, ya que demuestra que para un punto x_1 con coordenadas conocidas en una vista, sus coordenadas en la otra vista están limitadas a una línea epipolar, tal y como se puede ver en la figura 2.10.

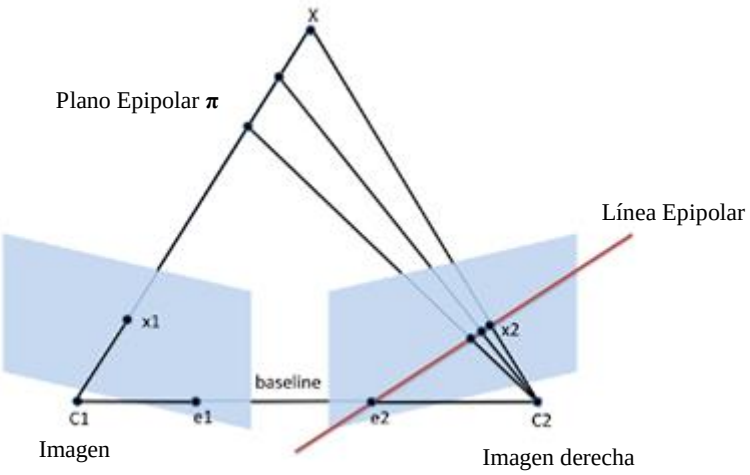


FIGURA 2.10: GEOMETRÍA EPIPOLAR

Este fenómeno se explica de la siguiente forma: las líneas que unen $\overline{X, x1, C1}$, $\overline{X, x2, C2}$ y $\overline{C1, C2}$ definen el plano epipolar, que contiene los centros de las cámaras C_1 y C_2, el punto X del espacio 3D y las proyecciones de dicho punto en ambas vistas. Los puntos epipolares (***e1*** y ***e2***) representan la intersección de la línea que une los centros de las cámaras con los planos de las imágenes, por lo tanto, también forman parte del plano epipolar. La línea epipolar (***l***) definida por un punto X coincide con la intersección del plano epipolar con el plano de una imagen. La línea epipolar en una vista será la proyección de la línea que une el punto en la otra vista con el punto X del espacio 3D, de forma que delimita las coordenadas del punto proyectado en la otra vista a esta línea epipolar.

Esta restricción geométrica se puede interpretar como una proyección de puntos, de forma que para cada punto $\boldsymbol{x_1}$ en una imagen, existe una línea epipolar (***l***) en la otra vista en la que se encontrará $\boldsymbol{x_2}$.

2.3.3 Rectificación de Imagen

Aunque la geometría epipolar simplifica en gran medida el problema de correspondencia entre los puntos de proyección x_1 y x_2, ya que delimita sus posiciones a una única línea, la orientación arbitraria de la línea epipolar hace que salga a la luz otro problema importante a la hora de comparar los píxeles entre ambas imágenes. Es entonces cuando se hace necesaria una técnica de rectificación.

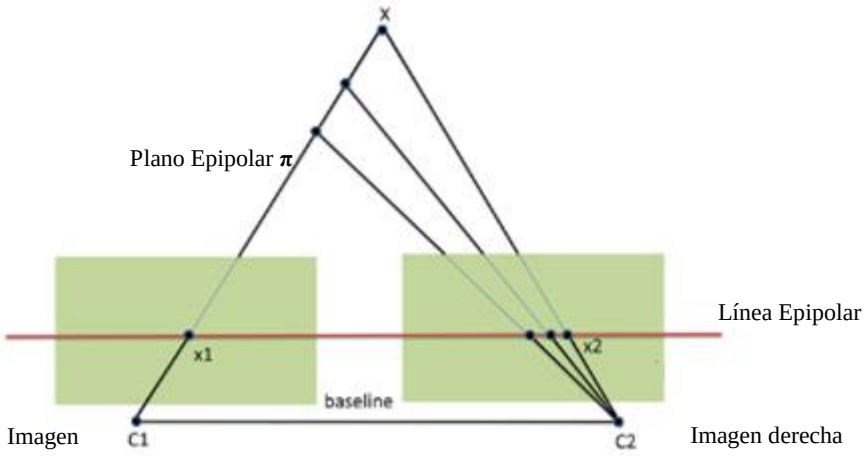


FIGURA 2.11: RECTIFICACIÓN DE IMAGEN

La rectificación de imagen es un proceso de transformación usado para proyectar dos imágenes en un plano de imagen común, y conseguir entonces que el par de líneas epipolares pasen a ser co-lineales y tener la misma dirección, tal y como se puede observar en la figura 2.11. Es por esto que con la rectificación de imágenes también se corrigen las distorsiones en las mismas. La idea detrás de la rectificación de imagen es definir dos nuevas matrices de proyección P_r' y P_l', de forma que se roten de nuevo las imágenes sobre el centro de las cámaras para que los planos de imagen se vuelvan co-planares, y paralelos a la *baseline* que conecta C_1 y C_2. Cualquier par de imágenes puede ser transformado de forma que las líneas epipolares pasen a ser paralelas y horizontales en cada imagen. Las nuevas matrices P_r' y P_l' tendrán los mismos parámetros intrínsecos, la misma orientación pero diferente posición. Tal y como se analiza en [15], las nuevas matrices de las cámaras serán:

$$P_l' = A[R|-RC1]$$
$$P_r' = A[R|-RC2] \qquad (2.12)$$

Los parámetros de la matriz A son los mismos para ambas cámaras y se pueden establecer arbitrariamente. *C1* y *C2* denotan el centro de la cámara. La matriz R hace referencia a la que proporciona la posición de las cámaras y será la misma para ambas. Los parámetros de *R* se especifican por vectores de filas: $R = [r1', r2', r3']'$ correspondiéndose cada vector con los ejes X, Y y Z respectivamente. El nuevo eje X tiene que ser paralelo a la *baseline*, el nuevo eje Y debe ser ortogonal a X y a un vector unitario y arbitrario *k*, y el nuevo eje Z debe ser perpendicular a XY. Por lo tanto, se obtiene:

$$r1 = (\mathrm{c1} - \mathrm{c2})/||\mathrm{c1} - \mathrm{c2}|| \qquad r2 = k \times r1 \qquad r3 = r1 \times r2 \qquad (2.13)$$

A la hora de rectificar la imagen, tomando como ejemplo la imagen izquierda, se necesita mapear el anterior plano de imagen en un nuevo plano, para lo que se usará la matriz de

transformación T. Partiendo de la anterior matriz P_l (apartados 2.3.1 y 2.3.2), se redefine con la forma $P_l = [Q_1|q_1]$, donde Q_1 es una matriz 3x3 y q_1 es una matriz 3x1. También se redefine la nueva matriz $P_l' = A[R|-RC2]$ que pasa a tener la forma $P_l' = [Q_1'|q_1']$. La matriz de transformación T_l que rectificará la imagen izquierda será:

$$T_l = Q_1' {Q_1}^{-1} \tag{2.14}$$

2.3.4 Stereo Matching:

El objetivo de *Stereo Matching* es identificar la correspondencia entre píxeles de un par o conjunto de imágenes. En este caso de un sistema de visión estéreo, se buscan las correspondencias entre una imagen izquierda y otra derecha. Tras la rectificación de las imágenes, la tarea de *Stereo Matching* estéreo se vuelve mucho más sencilla ya que la localización de los píxeles de la primera imagen en la segunda se limita únicamente a buscar en la misma línea horizontal de píxeles (con la misma altura *y*). *Stereo Matching* es la tarea más importante de un sistema de visión estéreo, cuyo estado del arte se revisa en el Capítulo 2. La salida del *Stereo Matching* es una imagen en blanco y negro, también denominado *Disparity Map*, con los valores de disparidad entre los correspondientes píxeles del par de imágenes, donde **disparidad** se refiere a la distancia horizontal absoluta entre la posición de un píxel en la imagen izquierda con la posición de su píxel correspondiente en la imagen derecha.

2.3.5 Triangulación:

Tras haber obtenido el mapa de disparidad utilizando *Stereo Matching*, la profundidad de cada pixel de dicho mapa puede ser determinada usando la técnica de triangulación, mostrada en la figura 2.10. *P* hace referencia a un punto en el espacio 3D, cuyas coordenadas son (X, Y, Z) y su proyección en la cámara izquierda, por ejemplo, es $p1 = (x1, y1)$. En la cámara derecha la proyección del punto *P* se define como $p2 = (x2, y2)$.

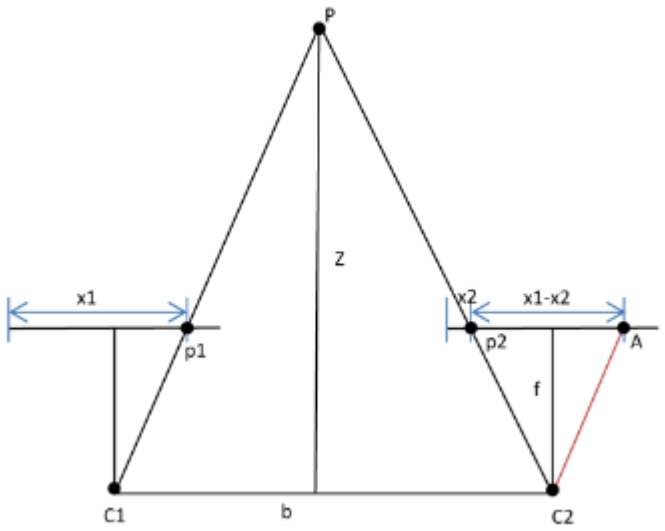


FIGURA 2.12: REPRESENTACIÓN GRÁFICA DE LA TRIANGULACIÓN

Gracias a la rectificación de cámara descrita en la sección 2.3.3, ahora las líneas epipolares son horizontales y la correspondencia entre píxeles caerá en la misma línea horizontal de píxeles, a una altura *y* del par de imágenes ($y1 = y2$). La *baseline b* representa la distancia entre los centros de las dos cámaras C_1 y C_2, f es la distancia focal, *Z* denota la profundidad del punto *P*, y es la incógnita del sistema de la figura 2.12. Si dibujamos una línea auxiliar paralela a $\overline{P, C1}$ desde el punto C_2, se obtiene el punto *A* y la línea $\overline{A, C2}$ que ayuda a delimitar el triángulo $A, \widehat{C2, p2}$, el cual es semejante al triángulo $P, \widehat{C1, C2}$, por lo que se puede escribir la siguiente ecuación:

$$\frac{Z}{b} = \frac{f}{x1 - x2} \tag{2.15}$$

donde x_1-x_2 es la disparidad calculada. La profundidad del punto P se describe como:

$$Z = \frac{f * b}{x1 - x2} \tag{2.16}$$

Gracias a esta ecuación, de deduce que el mapa de disparidad es inversamente proporcional al mapa de profundidad, que está relacionado con el fenómeno del **paralaje**, descrito en el apartado 2.2 sobre la visión binocular humana.

Capítulo 3
Estudio del algoritmo:
Stereo Matching

El algoritmo de *Stereo Matching*, que es empleado para buscar la correspondencia entre píxeles de un par de imágenes y obtener un mapa de profundidad, se viene perfeccionando desde hace varias décadas y aún se exploran mejoras que permitan implementarlo en tiempo real. En este capítulo se van a analizar las principales técnicas y estrategias que abarca este método, además de introducir las técnicas que se emplearán en la implementación y se desarrollarán más a fondo en el Capítulo 4.

3.1 Introducción a *Stereo Matching*:

La parte de *Stereo Matching* es la base de la visión estéreo. De acuerdo con una taxonomía básica propuesta por Scharstein y Szeliski [16], los métodos existentes para implementar *Stereo Matching* se pueden englobar en dos categorías: métodos globales y métodos locales. Los métodos globales computan todas las disparidades simultáneamente tras definir una función de energía que supone la suavidad de la imagen y después va aplicando algoritmos de optimización. Ejemplos de métodos globales de *Stereo Matching* son *graph cut* [17], *belief propagation* [18], *scanline optimization* [19] and *dynamic programming* [20]. Los métodos globales tienden a ofrecer mejores resultados de correspondencia, con menos cantidad de errores en la disparidad, pero por lo general son procesos muy caros computacionalmente debido a la naturaleza iterativa de los mismos, es decir, recorren varias veces la imagen para optimizar el resultado final después de la predicción de las profundidades.

Por otro lado, los métodos locales se valen del concepto de Correlación entre patrones de intensidad del par de imágenes. Usan una pequeña ventana de búsqueda para determinar la disparidad de cada píxel, y después se refina el resultado en función de una disparidad óptima. Al contrario que los métodos de *Stereo Matching* globales, los métodos locales ofrecen resultados

finales más rápidamente, pero generalmente tienen peor tasa de errores. En los últimos años los métodos locales han ido ganando popularidad ya que algunos de estos van incorporando estrategias que ofrecen un resultado más satisfactorio en comparación con otros métodos globales. [21] [22]

Todos los métodos que implementan *Stereo Matching*, tanto globales como locales, normalmente actúan en 4 pasos diferenciados. Estos 4 pasos son (1) Cómputo de la Correspondencia, (2) Asignación del coste, (3) Disparidad óptima, (4) Refinado de la Disparidad. El cumplimiento de los 4 pasos enumerados depende del algoritmo en cuestión, ya que se trata de una generalización; es decir, los métodos globales por ejemplo no incluyen un paso de asignación de la disparidad, ya que estos procesos la van asumiendo iterativamente. De la misma manera, no todos los métodos locales incluyen un refinado de las disparidades obtenidas. Más adelante se expondrán las diferentes técnicas utilizadas hoy en día dentro de estos 4 pasos que componen los diferentes algoritmos existentes.

3.2 Cómputo de la Correspondencia

Este primer paso es necesario para ambos métodos, tanto locales como globales. El cálculo de la correspondencia o *Matching Cost* valora la similitud entre dos píxeles de la escena dado un par de imágenes, por lo que cada pixel poseerá un *Matching Cost* propio. Los métodos locales se encargan a continuación de realizar una suma de los *Matching Cost* de los píxeles de una determinada área. Esta suma o agregación se explicará en el apartado 3.3. Los métodos globales usarán directamente esta similitud entre pares de píxeles en el paso (3) de búsqueda de disparidad óptima, introducido en el apartado 2.4.3.

A continuación, se enumeran las funciones de cálculo de la correspondencia más populares dentro del *Stereo Matching* [24]: Diferencias absolutas (o *Absolute differences*, AD), Diferencia de cuadrados (o *Squared Differences*, SD), Truncamiento de diferencias absolutas de color y gradiente (o *Truncated Absolute Difference of Color and Gradient*, TADCG), métodos basados en transformadas de Fourier, etc. Para nuestro conocimiento, la función de correspondencia AD será la más utilizada a lo largo de este TFG de la misma forma que es la más referenciada en numerosos artículos. [21] [22]

3.2.1 Diferencias absolutas y diferencia de cuadrados:

Las funciones de cálculo de la correspondencia denominadas Diferencias absolutas (AD) y Diferencia de cuadrados (SD) son simples computacionalmente y fáciles de implementar, lo que hace que resulten de las técnicas más empleadas. También se usarán en esta implementación de *Stereo Matching*. Estas técnicas usan los valores de luminancia (*Y*) de los píxeles de las imágenes izquierda y derecha de entrada. Es importante saber que AD y SD son utilizadas en otras

aplicaciones relacionadas con el procesado de imagen, como en *motion estimation* dentro de la compresión de vídeo [27], *tracking* de objetos [29], etc. Ambas operaciones se realizan a nivel de píxel entre las imágenes.

La AD entre un píxel p de la imagen izquierda I_l y un píxel d en la imagen derecha I_r viene descrita por la siguiente expresión:

$$C_{AD}(p,d) = |\, I_l(p) - I_r\,(d)|, \tag{3.1}$$

y para el caso de SD se puede escribir de forma similar:

$$C_{SD}(p,d) = (\, I_l(p) - I_r\,(d))^2, \tag{3.2}$$

3.2.2 Truncamiento de diferencias absolutas de color y gradiente (TADCG)

Las técnicas AD y SD resultan muy ligeras computacionalmente, sin embargo, no están a prueba de discordancias entre los niveles de exposición de las imágenes. Son técnicas que asumen una corrección de las imágenes de entrada y en caso contrario ponen en riesgo la fiabilidad y exactitud de la correspondencia entre píxeles. Es por eso que surge TADCG para lidiar con la discrepancia o diferencia de brillo de las imágenes del par de cámaras. En esta técnica, se tiene en cuenta la información de gradiente, que funciona más robustamente frente a cambios en los brillos que la información de la luminancia. TADCG se divide en dos partes, color y gradiente. La parte del color se procesa a nivel de píxel con la misma ecuación 3.1 que AD, mientras que la parte del gradiente se procesa siguiendo la siguiente ecuación:

$$C_{gradient}(p,d) = |\nabla_x\, I_l(p) - \nabla_x I_r\,(d)|, \tag{3.3}$$

donde ∇_x es el gradiente en función de *x*. Después, estas dos partes se combinan mediante truncamiento para que la función final de truncamiento de diferencias absolutas de color y gradiente sea:

$$\begin{aligned} C_{TADGG}(p,d) = (1 - \theta)\ \text{x}\ \min\ (|I_l(p) - I_r\,(d)|, \tau_1) \\ + \theta\ \text{x}\ \min(|\nabla_x\, I_l(p) - \nabla_x I_r\,(d)|, \tau_2) \end{aligned} \tag{3.4}$$

donde θ iguala las partes de color y de gradiente, τ_1 y τ_2 son valores de truncamiento que ayudarán a reducir la influencia negativa de zonas ocultas[2] en las imágenes. Sin embargo, esta

[2] Con zonas ocultas en una imagen se refiere a que, por diferencias de perspectiva de las cámaras, en una vista se verán partes del objeto que en otras no, por lo tanto, resulta complicado realizar la correspondencia en esa parte de la imagen.

técnica resulta computacionalmente costosa, y aleja del objetivo de *Stereo Matching* de ofrecer una respuesta en tiempo real.

3.3 Asignación del Coste

La asignación del coste es el paso más importante en los métodos locales de *Stereo Matching*, ya que de este paso depende la eficacia y eficiencia del proceso completo. Como ya se ha apuntado anteriormente, este paso no se lleva a cabo en los métodos globales. La literatura sobre la asignación del coste ya empezó a publicarse en los años 70 [11] [12] [13]. En este paso, cada píxel es asignado a una ventana de píxeles o macrobloque de un tamaño determinado, donde dentro de dicha ventana se realizará una suma de las correspondencias de los píxeles dando lugar a un coste global. Esto lleva a que, para el proceso de *Stereo Matching*, se tenga en cuenta a todo el bloque en lugar de un sólo píxel, con el objetivo de mejorar la fiabilidad del método. Este último paso resulta necesario para realizar un *Matching* correcto píxel a píxel, ya que cada píxel no es único dentro de una imagen, lo que dificulta en gran medida la localización del píxel de la imagen en la otra del mismo par. Al tener en cuenta un macrobloque de píxeles, se aumenta sustancialmente la probabilidad de que un bloque sea único dentro de la imagen, o al menos en una parte de ella.

En este apartado se incluyen varias prácticas de Asignación de coste analizadas en [31] y [32], como *fixed window approach*, *shiftable window approach*, *variable window approach*, *multiple window approach* y *adaptive support weight based approach* [9] [21] [22] [8]. La aparición de estos procedimientos de generación de Macrobloques y la asignación del coste ha sido secuencial, de forma que el próximo ha nacido siempre como mejora del anterior. Es por eso que, para la obtención de resultados con mayor calidad, el método más utilizado es el *adaptive support weight based approach*.

3.3.1 Fixed Window Approach

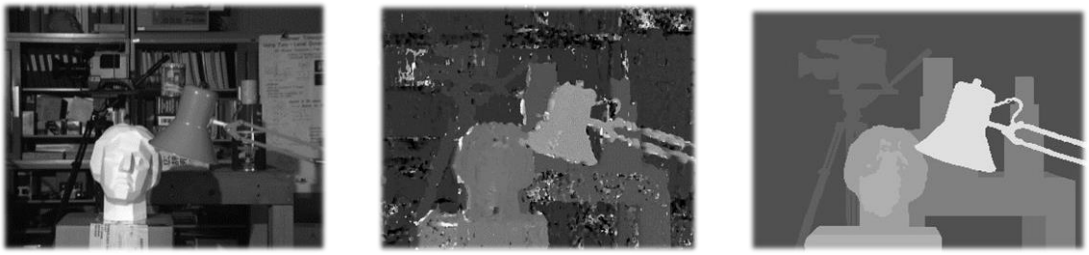
La técnica de *fixed window* es la más tradicional de las mencionadas anteriormente, además de ser la que menos tiempo consume [32]. Un macrobloque de píxeles de tamaño fijo, normalmente de dimensiones cuadradas, es creado para cada píxel, teniendo como inicio dicho pixel. El coste de dicha ventana es asignado bajo la siguiente expresión,

$$C_{aggr}(p,d) = \sum_{q \in \omega_p} C_{raw}(q,d) \tag{3.5}$$

donde ω_p denota un macrobloque de píxeles cuya esquina superior izquierda está en el píxel *p*, *q* es un píxel dentro de ω_p y $C_{raw}(q,d)$ es el coste de dicho píxel, que puede ser calculado

utilizando AD, SD o TADCG. Si se elige AD para calcular el coste, entonces la función de asignación de coste utilizada se conocerá como Suma de diferencias absolutas (SAD).

Esta técnica, *fixed window*, ofrece de una forma computacionalmente sencilla capacidad para los métodos locales a obtener en el siguiente paso disparidad a nivel de píxel. Sin embargo, tiene un problema importante, y es que asume que todos los píxeles incluidos en la ventana fija tienen el mismo valor de disparidad. De hecho, es muy probable que esto no ocurra debido a su tamaño fijo y forma no variable, violando así esta asunción. Kanade y Okutomi [35] afirman que el método *fixed window* será propenso a cometer fallos cuando los píxeles del macrobloque no tengan la misma disparidad, produciendo así inexactitudes en las zonas de la imagen donde haya bordes y saltos entre objetos. Sin embargo, se demuestra que es el método más rápido en ofrecer el mapa de profundidad (Tabla 3.1), ya que requerirá de menos operaciones de cómputo que otros métodos alternativos.



a) b) c)

FIGURA 3.1: A) IMAGEN DE REFERENCIA B) MAPA DE PROFUNDIDAD OBTENIDO CON LA IMPLEMENTACIÓN DESARROLLADA EN ESTE TFG, QUE UTILIZA *FIXED WINDOWS* DE LA IMAGEN DE TSUKUBA DEL *DATASET* DE MIDDLEBURY. C) MAPA OBTENIDO UTILIZANDO *GROUND TRUTH*.

Técnica de Asignación de coste	Tiempo (mm:ss)
Adaptive weight	18:14
Fixed Window	< 1s
Shiftable Window	00:15
Multiple Window (5W)	00:07
Multiple Window (9W)	00:09
Multiple Window (25W)	00:17
Variable Window	00:25

TABLA 3.1: TABLA CON UN ANÁLISIS COMPARATIVO DEL RENDIMIENTO DE LAS DIFERENTES TÉCNICAS EXISTENTES DE ASIGNACIÓN DE COSTE, REALIZADO POR TOMBARI ET AL. EN [32].

En este trabajo se tiene como objetivo implementar un algoritmo de *Stereo Matching* en tiempo real. Dada la comparativa de rendimiento presentado en [32] la mejor opción para ello es *Fixed Window*. Es por esto por lo que se elige dicha técnica para la implementación del algoritmo en lugar de las otras mencionadas en la Tabla 3.1 y explicadas a continuación.

3.3.2 Otras técnicas de asignación de coste

Shiftable window approach

En esta técnica, se crean varias ventanas centradas en diferentes localizaciones de la misma, como se muestra en la figura 3.3, por cada píxel de la imagen. De todas las ventanas creadas, se selecciona la que menos coste medio tenga. La teoría detrás de esta idea es que la ventana que encierre un menos coste medio en su interior, es la que más probabilidad tiene de que las disparidades de sus píxeles sean iguales. De esta manera, el concepto de disparidad controla la selección de una ventana o macrobloque más apropiado. Tal y como se muestra en la figura 3.2.

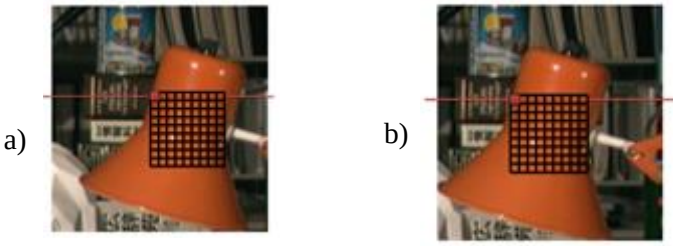


FIGURA 3.2: A) IMAGEN IZQUIERDA B) IMAGEN DERECHA. EN ESTA FIGURA SE MUESTRA EL QUE SERÍA EL MACROBLOQUE CON MENOR COSTE MEDIO, QUE EN ESTE CASO TIENE COMO REFERENCIA EL PÍXEL ROJO.

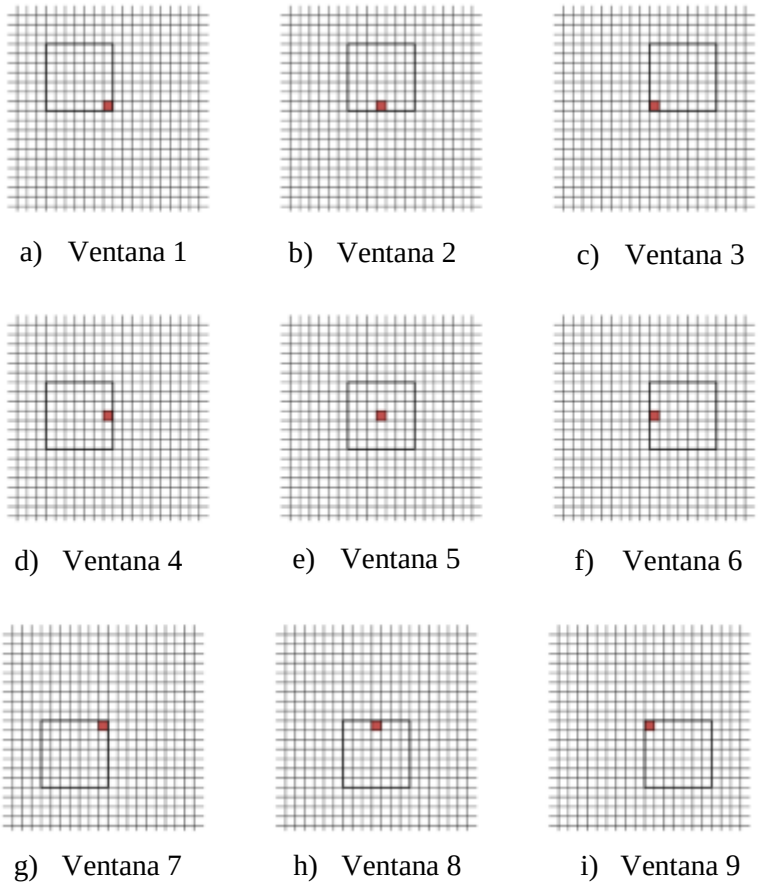


FIGURA 3.3: LAS 9 VENTANAS DE *SHIFTABLE WINDOW*. SE CALCULA LA DISPARIDAD DEL PÍXEL ROJO Y SE ESCOGE LA VENTANA ÓPTIMA PARA ELLO EN FUNCIÓN DE CUAL TIENE MENOR COSTE.

Multiple window approach

En esta técnica, se tiene como objetivo evitar las discontinuidades entre regiones de la imagen cambiando la forma de la ventana. Se proponen tres configuraciones diferentes de unión de bloques para formar una estructura desigual lejos de componen una ventana fija e invariante: macrobloques de 5 ventanas, 9 ventanas o 25 ventanas. Tomando la configuración de 9 ventanas como ejemplo, que se muestra en la figura 3.6, se establecen 9 sub-ventanas candidatas para cada píxel, marcando una con un color diferente. A continuación, se seleccionan 5 de las 9 sub-ventanas para determinar un macrobloque final. En esta selección, se debe incluir la sub-ventana central, por lo que las otras 4 se seleccionarán a partir de las 8 restantes. Por lo tanto, resultará un macrobloque de forma variable en función del contenido local de la imagen. Se presentan 6 combinaciones posibles en la imagen 3.4. En un caso ideal, los píxeles del macrobloque resultante tendrán todos el mismo valor de disparidad, como se muestra en la figura 3.5.

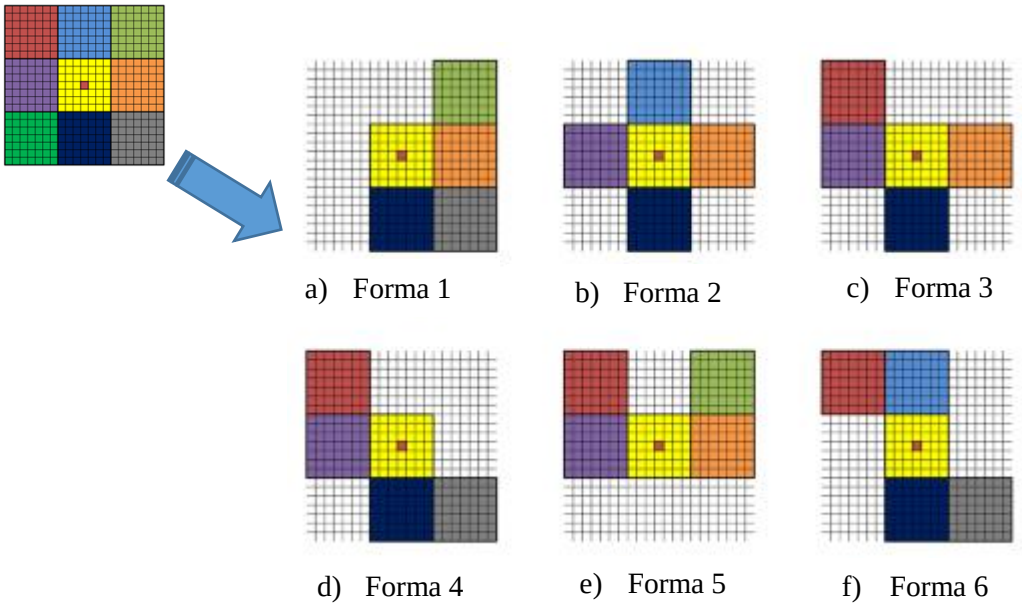


FIGURA 3.4: LAS 9 VENTANAS DE *MULTIPLE WINDOW*, SIENDO SELECCIONADAS CON DIFERENTES FORMAS PARA ABARCAR LA ZONA CON MENOR COSTE ENTRE TODAS LAS POSIBLES.

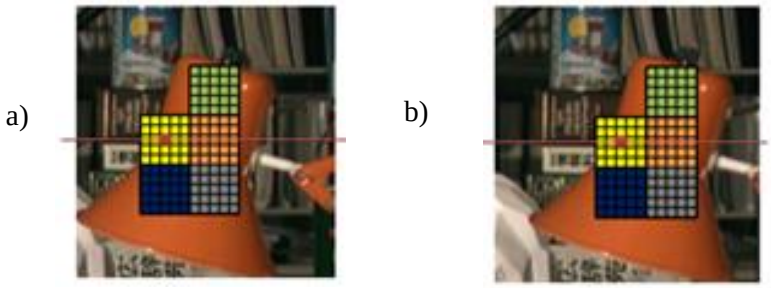


FIGURA 3.5: A) IMAGEN IZQUIERDA B) IMAGEN DERECHA. SE ESCOGE LA FORMA CON MENOR COSTE.

Variable window approach

Esta técnica afronta los cambios abruptos de disparidades en la imagen mediante el cambio del tamaño del macrobloque y de su forma. La teoría detrás de este método tiene dos premisas. (1) un macrobloque debe ser lo suficientemente grande como para recoger variaciones de la intensidad en la imagen, pero debe ser lo suficientemente pequeño como para que los píxeles que contenga tengan todos la misma disparidad.

En la figura 3.6, se presentan varias implementaciones de mapas de profundidad obtenidos con diferentes tamaños de ventana o macrobloque. Se observa que cuanto mayor sea el tamaño del bloque, mejor actúa el algoritmo frente a zonas sin textura clara, ya que se concentra más información dentro del macrobloque, tal y como se ilustra en la figura 3.7. También se observa que un tamaño de macrobloque más pequeño resuelve bien las zonas con fuertes variación de profundidad, ya que reduce la probabilidad de incluir píxeles con diferentes disparidades dentro del macrobloque. Por lo tanto, lo óptimo será un tamaño de ventana variable acorde con el contenido local de la imagen.

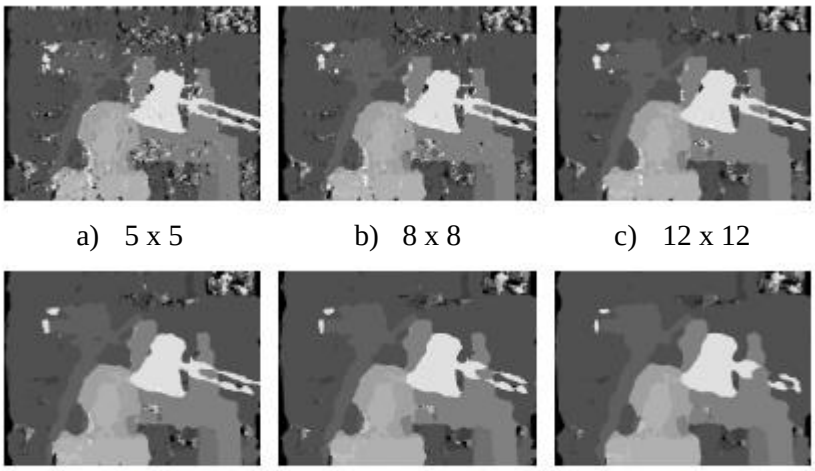


FIGURA 3.6: MAPAS DE PROFUNDIDAD DE LA IMAGEN DE TSUKUBA PARA DIFERENTES TAMAÑOS DE MACROBLOQUE

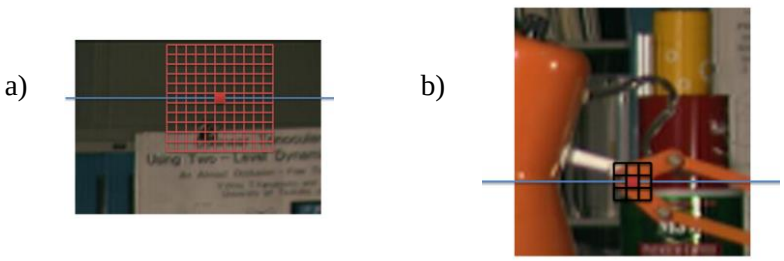


FIGURA 3.7: A) VENTANA MÁS GRANDE PARA TRATAR LAS ZONAS CON POCA TEXTURA. B) VENTANA MÁS PEQUEÑA PARA TRATAR LAS ZONAS CON SALTOS ENTRE OBJETOS Y PROFUNDIDADES.

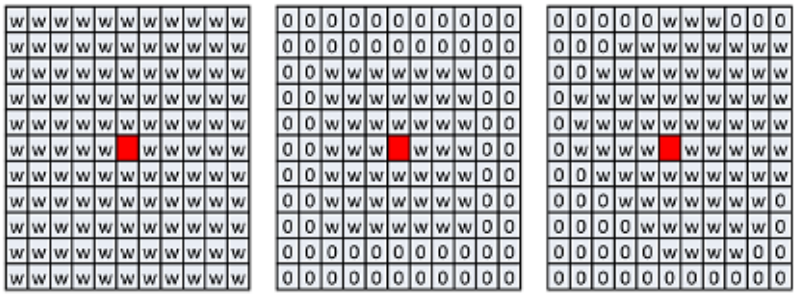
Como segunda premisa, (2) cerca de los saltos de disparidad, es necesario evitar que los diferentes macrobloques, con tamaño variables, crucen a la vez dichos saltos, por lo que se deben implementar diferentes formas para los bloques. La idea de cambiar la forma de la ventana es la misma que en *Shiftable window*.

Adaptive support weight

Esta técnica fue propuesta por Yoon and Kweon [8], y trata de mejorar los anteriores métodos de asignación de coste. La teoría de esta técnica defiende que se debe asignar un peso a cada píxel dentro de la ventana escogida y después asignar el coste siguiendo la siguiente expresión:

$$C_{aggr}(p,d) = \sum_{q \in \omega_p} \omega\,(p,q) \cdot C_{raw}(q,d), \qquad (3.6)$$

donde ω_p denota es macrobloque centrada en el píxel p; q es un píxel dentro de ω_p y $\omega(p,q)$ es el peso del píxel q. El peso de un píxel representa la probabilidad para dicho píxel de tener el mismo valor de disparidad que el píxel central de la ventana. Esto quiere decir, que cuanto más cerca esté el píxel del centro de la ventana, más probabilidad tendrá de que se le atribuya un peso. Este proceso de asignar o no un peso a los píxeles del macrobloque es más flexible y potente que cambiar simultáneamente el tamaño y la forma del bloque como hacen los anteriores métodos. Éste argumento se ilustra en la figura 3.8. En (a), se establece el macrobloque centrado en un píxel, donde *w* hace referencia al peso y será el mismo para todos los píxeles. Si el peso de los píxeles más exteriores se establece a 0, tal y como se muestra en (b), se reduce así el tamaño de la ventana dinámicamente. Si el peso de otros píxeles se establece a 0 de la que se muestra en (c) se puede cambiar la forma del macrobloque. En resumen, se puede cambiar el tamaño y la forma de la ventana asignando peso nulo de los píxeles dentro de ella.



a) Ventana grande b) Ventana pequeña c) Forma diferente

FIGURA 3.8: ADAPTIVE SUPPORT WEIGHT OFRECE LA POSIBILIDAD DE UNA VENTANA VARIABLE EN FORMA Y TAMAÑO.

El método de *adaptive support weight* es el que más calidad ofrece en los mapas de profundidad, tal y como se puede ver en un estudio comparativo realizado por Gong et al. en [31] mostrado en la figura 3.9. Sin embargo, se debe establecer un compromiso entre calidad de resultados y cómputo de los mismos, y es por eso por lo que esta técnica se desecha y se elige *Fixed Windows*. *Adaptive weight* computa una salida demasiado lento (ver tabla 3.1), mientras que *Fixed Windows* si ofrece una salida adecuada en comparación con las demás técnicas, tal y como se muestra en la figura 3.9:

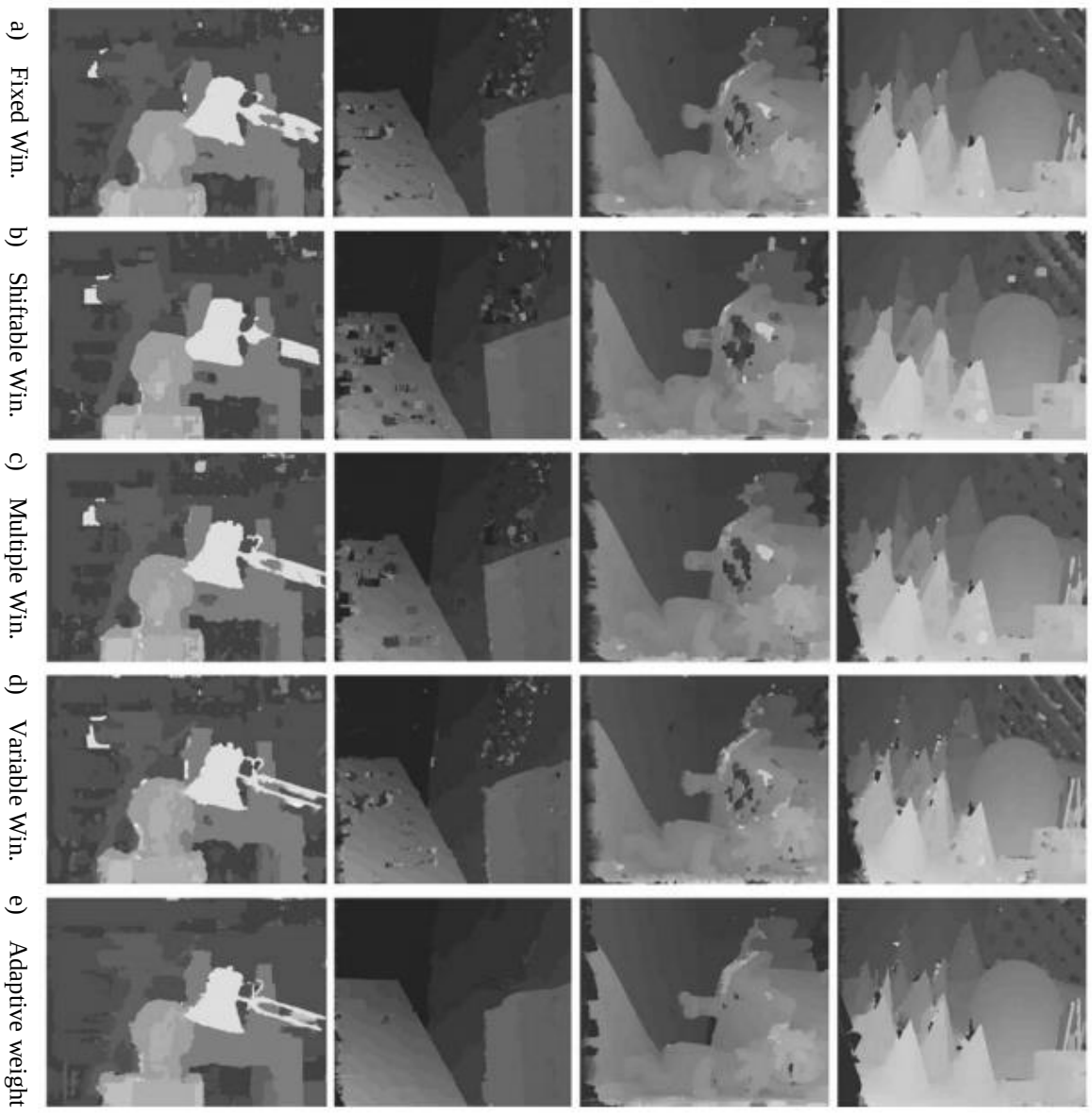


FIGURA 3.9: ANÁLISIS CUALITATIVO COMPARATIVO DE LA CALIDAD DE LOS MAPAS DE PROFUNDIDAD OBTENIDOS CON LAS DIFERENTES TÉCNICAS EXISTENTES DE ASIGNACIÓN DE COSTE, REALIZADO POR G ET AL. EN [31] PARA DIFERENTES IMÁGENES ESTÉREO.

3.4 Disparidad Óptima

Este paso del *Stereo Matching* hace referencia a todos aquellos métodos o procedimientos que se encargan de obtener el valor óptimo de disparidad para cada píxel perteneciente a un macrobloque. La ejecución de esta parte resulta diferente si se está hablando de un método local o de un método global, residiendo aquí la principal diferencia entre ambos métodos.

3.4.1 Optimización Local:

El método usado para la obtención del valor de disparidad correcto para cada píxel, *D(p)*, resulta bastante sencillo en un método local. La técnica más popular es *winner-taken-all strategy* [16], donde la disparidad óptima se escoge en función de qué macrobloque de una imagen tiene la mínima diferencia de coste con otro macrobloque de la imagen del mismo par. Es decir, se establece dónde se encuentra un macrobloque de una imagen en la otra en función de cómo de semejante sea su valor de coste. A continuación, la diferencia de posición entre la localización del mismo bloque en ambas imágenes se traducirá en un valor de disparidad óptimo.

$$D(p) = \arg\min_{d}\left(C_{aggr}(p,d)\right) \tag{3.7}$$

Si se está empleando el método de SD para el cómputo de correspondencia, entonces se utiliza la máxima suma de costes en lugar de la mínima para obtener la disparidad. Lo primero que se puede pensar una vez llegados a este punto es que sólo se va a obtener disparidades a nivel de macrobloque y no a nivel de pixel, sin embargo, se recuerda que, en los métodos de asignación de coste descritos en el apartado 3.3, se creaba un nuevo macrobloque por cada píxel. De esta manera, se llega a dar un valor de disparidad para cada píxel, ya que existe un bloque asociado a cada píxel. Esto supone un objetivo fundamental de *Stereo Matching*.

3.4.2 Optimización Global:

Los métodos globales emplean casi todo su esfuerzo en este paso, en la obtención de la disparidad óptima, y omiten, normalmente, el paso de asignación de coste, ya que no agrupan los píxeles para generar bloques más grandes, si no que tratan con todo el coste de la imagen. El objetivo de este método es encontrar una función de disparidad d que minimice el coste global,

$$E(c) = E_{data}(c) + \lambda E_{smooth}(c) \tag{3.8}$$

donde λ es una constante. El término $E_{data}(c)$ mide el grado de concordancia entre la correspondencia *c* y el par de imágenes de entrada, donde su valor se obtiene de la diferencia de las intensidades entre los correspondientes píxeles,

$$E(c) = E_{data}(c) + \lambda E_{smooth}(c) \quad (3.9)$$

donde I_l denota al conjunto de pixeles de la imagen izquierda y el coste de la imagen raw C_{raw} se puede calcular utilizando una función de coste de las mencionadas en el apartado 3.1. El término de suavidad $E_{smooth}(c)$ tiene en cuenta que los píxeles vecinos en la imagen tienden a tener correspondencias similares y se restringen únicamente a medir las diferencias entre correspondencias de píxeles vecinos. El término de suavidad basado en el modelo de método de Scharstein y Szeliski [16] es:

$$E_{smooth}(c) = \sum_{(x,y)} \rho(d(x,y) - d(x+1,y)) + \rho(d(x,y) - d(x,y+1)) \quad (3.10)$$

donde ρ es una función cuadrática de cálculo de disparidad de forma monoatómica, es decir, se aplica pixel a pixel, y crea una suavidad de la correspondencia en todas partes de la imagen. Esto puede llevar a resultados de disparidad pobres si hay muchos objetos en la imagen en diferentes planos de profundidad, sin embargo, los algoritmos globales actuales tratan adecuadamente este problema, como por ejemplo *Graph Cuts* [17].

3.5 Refinado de la Disparidad

El refinado o post-procesado de la disparidad es el último paso del método *Stereo Matching*, que está destinado a eliminar los fallos e inexactitudes en el cálculo de las disparidades de cada píxel. El hecho de que aparezcan fallos a la hora de calcular la disparidad puede tener que ver con dos factores. El primero, por las características de las imágenes de entrada. A continuación se enumeran algunos aspectos que pueden dar problemas en el cálculo de la disparidad de un par de imágenes: variaciones en la exposición o en el contraste de las dos imágenes, ruido en el sensor de la cámara, zonas quemadas en la imagen o con demasiado brillo, diferencias o distorsiones de perspectiva en los objetos de la imagen, zonas u objetos sin textura clara, estructuras o texturas repetitivas, reflejos o transparencias en los objetos de la imagen (cristales), oclusiones de objetos por la perspectiva, etc.

El segundo de los factores tiene que ver con los propios métodos y técnicas que componen *Stereo Matching* (es decir, errores en los tres pasos anteriores), ya que no existe la técnica perfecta que dé total exactitud en el cálculo de las disparidades. Sin embargo, con los métodos existentes es común ver buenos resultados de *Matching* en multitud de artículos que abordan este algoritmo.

Las técnicas de post-procesado para el refinamiento de los resultados obtenidos son por lo general métodos muy costosos computacionalmente. El modo de operación de estas técnicas viene dividido generalmente en tres pasos. En primer lugar (1), se debe disponer de un mapa de disparidad para la imagen izquierda y otro para la imagen derecha, es decir, haber computado dos

mapas, primero tomando como referencia la imagen izquierda y después la imagen derecha. En segundo lugar (2), se mide la consistencia entre los mapas de las imágenes izquierda y derecha [14]. De esta forma, se establecen píxeles erróneos en caso de que la disparidad en ambas imágenes no sea lo misma para dicho píxel, o bien que haya una diferencia entre ambas superior a un determinado umbral. En último y tercer lugar (3), se recalcula el valor de disparidad para los píxeles errores tomando como referencia los píxeles vecinos.

En la literatura sobre este paso, se pueden encontrar hasta tres ejemplos de métodos de refinamiento, que son *unweighted median filtering* [16], *the weighted median filtering* [9], y *reaggregation-based method* [21].

Capítulo 4
Implementación del sistema y evaluación

En este capítulo se propone una implementación del algoritmo *Stereo Matching* orientada a establecer una salida en tiempo real, utilizando técnicas sencillas computacionalmente pero que proporcionen un buen grado de exactitud. También se analiza su rendimiento y opciones de posibles mejoras. La implementación se puede modificar en un futuro trabajo con el fin de mejorar la calidad de los mapas de profundidad obtenidos, pero sin perder en gran medida la velocidad total de procesado.

4.1 Descripción general

Tal y como se ha comentado en el capítulo 3, se implementará un *Stereo Matching* dividido en 4 ideas fundamentales, donde las técnicas a emplear a lo largo de la implementación se eligen en función de su complejidad computacional y la calidad del resultado. También se implementa un proceso de calibración de las cámaras (apartado 2.3) y otras técnicas de pre-procesado de las imágenes. Es por eso que la estructura completa de la implementación se puede dividir en varias partes donde se diferencian dos módulos independientes (ver figura 4.1). El primer módulo realiza las siguientes tareas:

- CALIBRACIÓN ESTÉREO: Se obtienen los parámetros tanto intrínsecos como extrínsecos entre ambas cámaras (matrices ***K***, ***R*** y ***t*** del apartado 2.3.1).
- RECTIFICACIÓN[3]: Se aplica la teoría de rectificación de las imágenes con el fin de obtener 4 matrices que nos servirán en este proceso, una matriz de transformación y una nueva matriz de proyección por cada imagen.

[3] Ver apartado 2.3.3

El primer módulo no se incluye como parte del algoritmo *Stereo Matching*, aunque su implementación es necesaria ya que se utilizan dos cámaras *pinhole* como entrada en lugar de una cámara estéreo. Al no utilizar una cámara estéreo, las probabilidades de tomar imágenes no corregidas, distorsionadas y que no se encuentran en el mismo eje óptico aumentan considerablemente. Esto hace necesario calibrar las dos cámaras empleadas, que se acoplarán a una montura de madera que las alinee horizontalmente para reducir la probabilidad de fallo del proceso de calibración, ofrecer mayor ángulo de visión y calidad en las imágenes. Por otro lado, el segundo módulo aplicará los parámetros de las matrices obtenidas a las imágenes de entrada y les aplicará el algoritmo *Stereo Matching*, dando como salida un mapa de profundidad en formato de imagen en escala de grises. Para conseguir esto se distinguen las siguientes tareas:

- ALINEACIÓN Y RECORTE: Se aplica la información guardada en las matrices de rectificación a las cámaras, y se confeccionan las imágenes ya que tras la rectificación se pueden obtener tamaños diferentes de las imágenes y zonas deformadas. Esto hace necesario recortar las imágenes e igualar los bordes.
- FILTRADO DE LAS IMÁGENES: con el fin de optimizar el proceso de *Stereo Matching*, se utilizan estrategias que igualen el brillo de las imágenes y traten las zonas con poca textura en las mismas.
- *STEREO MATCHING*: donde se diferencian dos tareas,
    - CORRELACIÓN DE LAS IMÁGENES: Se aplica la teoría de las técnicas recogidas en los apartados 3.2, 3.3 y 3.4 para obtener la disparidad adecuada de cada píxel de la imagen.
    - GENERACIÓN DEL MAPA DE PROFUNDIDAD: teniendo en cuenta la información de disparidad obtenida anteriormente, en esta tarea se confecciona el mapa de profundidad final en escala de grises.

El segundo módulo actúa de forma que, procesa las imágenes de entrada según se vayan obteniendo y no captura unas nuevas hasta proporcionar la salida.

A continuación, se describe en un diagrama de bloques los diferentes módulos y tareas del sistema en la figura 4.1. Algunas partes del diagrama relacionadas con la parte de la calibración se pueden subdividir en tareas más atómicas tal y como se muestra en la figura 4.2.

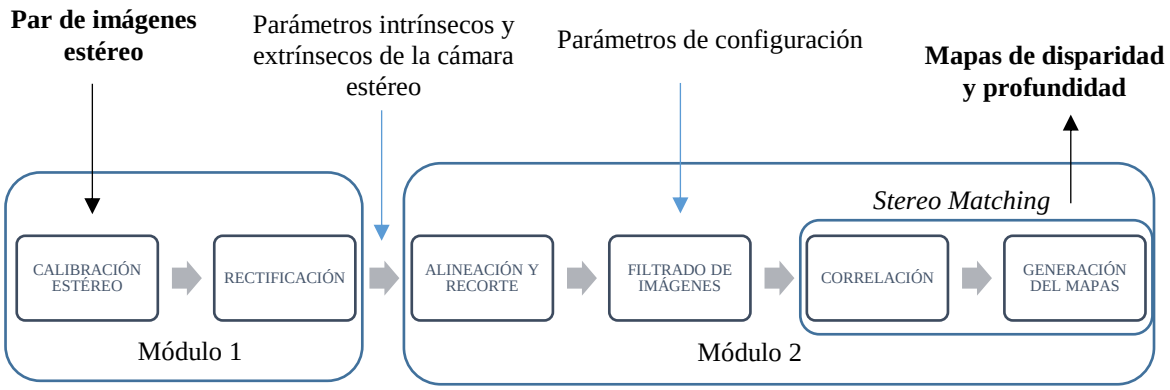


FIGURA 4.1: DIAGRAMA DE BLOQUES DE LA IMPLEMENTACIÓN DESARROLLADA

FIGURA 4.2: ESTRUCTURA DE LA PARTE DE CALIBRACIÓN ESTÉREO

4.2 Desarrollo de la implementación

En este apartado, se van a dar más detalles sobre la implementación realizada. Para el desarrollo del código se ha utilizado el lenguaje de programación C++. Se ha escogido este lenguaje debido a que ofrece un fácil manejo de los datos dentro de los algoritmos de procesado de imagen y sencillez a la hora de integrar la librería OpenCV. Para la implementación del sistema se utiliza el programa *Eclipse*. A continuación, se analizan las limitaciones previas tenidas en cuenta sobre este algoritmo. Se presentan también las funciones de OpenCV incluidas y su funcionalidad; y, por último, se comentarán todas las técnicas empleadas en la implementación. En total se implementan dos proyectos de Eclipse, uno por cada módulo de la figura 4.1, donde se han escrito 790 líneas de código en total y se realizan todas las tareas descritas en el apartado 4.1.

4.2.1 Limitaciones

Al utilizar dos imágenes provenientes de cámaras independientes, se pueden presentar considerables diferencias: distintas escalas de brillo, distintas reflexiones espectrales, una imagen puede estar peor enfocada que la otra, etc. En la implementación desarrollada de *Stereo Matching*

en este trabajo, se utilizan las luminancias de las imágenes, por lo que el algoritmo necesita asumir lo siguiente:

- En las imágenes solo aparecen objetos opacos, sin reflexiones aparentes.
- En las imágenes no existen zonas sin ninguna textura aparente mayores que el tamaño de macrobloque.
- Las imágenes están enfocadas de igual forma.
- Las cámaras a graban con el mismo brillo.
- Las cámaras grabaran la escena alineadas entre sí y en el mismo eje óptico.

El no cumplimiento de estas condiciones puede llevar a errores en las correspondencias. Sin embargo, dadas las características del montaje de las cámaras, es muy poco probable que no se incumpla ninguna de estas premisas. Aunque es verdad que algunas de las técnicas recogidas a lo largo del capítulo 3 combate algunas de estas limitaciones, existen otras técnicas de pre-procesado de las imágenes más sencillas y rápidas que se encarguen, por ejemplo, de normalizar los histogramas de contraste para añadir textura o de aplicar técnicas de compensación de iluminación. Por otro lado, no se debe olvidar el hecho de que se van a utilizar dos cámaras individuales, lo que hace que no exista sincronización entre ellas. Esto hace que sea imposible tomar imágenes en el mismo momento de tiempo, aunque sí será posible hacerlo con muy poca diferencia. Por tanto, se debe tener en cuenta que en escenas con objetos en movimiento la correspondencia entre píxeles será peor.

Además, las cámaras pueden no estar alineadas ni corregidas al no tratarse de una cámara estéreo, lo que obliga a añadir a esta implementación un módulo previo de calibración de las cámaras. En este trabajo, se ha limitado a realizar el proceso de calibración y de pre-procesado de las imágenes utilizando funciones de la librería OpenCV 3.1.0 [33]. También se utiliza esta librería en el proceso de toma de imágenes por parte de las cámaras. A continuación, se expone la batería de funciones utilizadas de esta librería junto con la funcionalidad de las mismas dentro de la implementación.

4.2.2 OpenCV

OpenCV es una librería *Open Source* creada en 1999 que ofrece gran variedad de recursos en el ámbito del procesado de imágenes. Incluye numerosas funciones que facilitan el desarrollo de programas de alta complejidad, así como algoritmos completos de visión por pantalla. Está orientada a la programación *SW* utilizando el lenguaje C++. En el anterior sub-apartado se ha mencionado en qué partes de nuestra implementación se usan funciones de la librería OpenCV. También se utilizan clases de OpenCV para almacenar imágenes o elementos durante la implementación, siendo las más usada la clase *Mat* que trata la variable como una matriz de datos

y es útil para almacenar imágenes, la clase *Vector* que representa una matriz unidimensional capaz de manejar diferentes tipos clases y variables, y otras clases representar coordenadas y correspondencias entre píxeles.

4.2.2.1 Funciones de calibración estéreo

Entre las funciones incluidas en OpenCV, existen varias funciones que realizan por completo la calibración tanto de una cámara *pinhole single* como de una cámara estéreo. Para dichas funciones se deben facilitar ciertos parámetros de entrada, sin embargo, también existen funciones de OpenCV que proporcionan dichos parámetros con tal sólo capturar con la cámara, por ejemplo, la imagen de un tablero de ajedrez. De esta forma, OpenCV ofrece implícitamente un proceso sencillo de calibración de cámaras *single* y estéreo. La calibración que se implementa en nuestro sistema con la ayuda de OpenCV sigue la teoría expuesta en el capítulo 2.3 y se estructura acorde con los diagramas de las figuras 4.1 y 4.2. Echando la vista atrás en este trabajo, en la figura 2.4, que muestra un sistema general de *Stereo Matching,* también se incluye el proceso de calibración implementado aquí.

La primera de las funciones a analizar es ***StereoCalibrate*** [28]. Esta función ofrece varios parámetros de salida, al igual que necesita numerosos parámetros de entrada. Con *StereoCalibrate* se pueden obtener varios parámetros de calibración ya mencionados en el apartado 2.3.1 de este trabajo. Entre ellos, destacan los parámetros intrínsecos (matriz *K*) para cada cámara con las variables *cameraMatrix1* (M_1) y *cameraMatrix2* (M_2). Como en esta función se suponen los ejes X e Y del espacio 3D ortogonales entre ellos, el parámetro *s* de *skew* será igual a 0. Los vectores de salida *distCoeffs1* (D_1) y *distCoeffs2* (D_2) ofrecen los parámetros de distorsión de cada cámara para corregirla posteriormente. También se obtienen las matrices de rotación (*R*) y de traslación (*t*) que conforman los parámetros extrínsecos de las cámaras, la matriz fundamental[4] (F) y la elemental (E). Estas últimas dos matrices no se utilizarán en el proceso de rectificación.

Para la obtención de estos parámetros, es necesario dar como entradas (1) un vector de puntos del espacio 3D (parámetro *objectPoints*), (2) las proyecciones de dichos puntos en la primera imagen (*imagePoints1*), (3) las proyecciones de dichos puntos en la segunda imagen (*imagePoints2*) y otros parámetros relacionados con criterios y *flags* que dependen de qué se quiera obtener con *StereoCalibrate* y a partir de qué parámetros. En nuestra implementación, se incluyeron los *flags cv_calib_same_focal_length, y cv_calib_fix_focal_length* que establecen los parámetros intrínsecos óptimos a partir de los puntos 3D introducidos; y *cv_calib_zero_tangent_dist y cv_calib_fix_principal_point* que quitan la distorsión. En cuanto a

[4] Ver apartado 2.3.2

los parámetros de *TermCriteria* se establecen los valores recomendados por la documentación de OpenCV 3.1.0 [28].

Los parámetros de entrada *objectPoints, imagePoints1* e *imagePoints2* están relacionados con la teoría explicada en el apartado 2.3.2 sobre geometría epipolar. Para obtener los valores de estos tres parámetros de entrada, en la documentación de OpenCV se plantea un método sencillo que plantea cómo elegir unos puntos determinados del espacio 3D de una escena que tengan proyecciones fáciles de identificar en las vistas izquierda y derecha de las cámaras. Se plantea utilizar un tablero de ajedrez con cuadrados blancos y negros alternos y bien diferenciados, como el de la figura 4.3, y grabarlo con las cámaras a calibrar (también en la figura 4.3) con el fin de encontrar las esquinas en la imagen izquierda y derecha y sacar las coordenadas 3D y 2D de esos puntos.

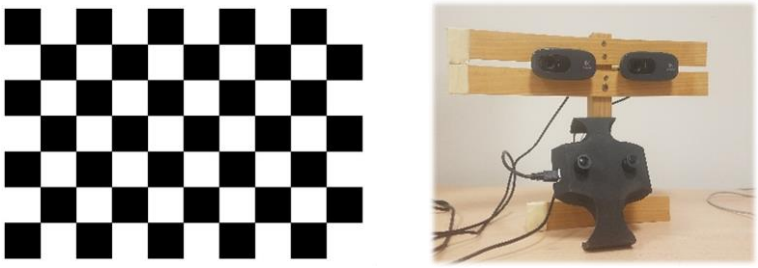


FIGURA 4.3: TABLERO UTILIZADO PARA CALIBRAR LAS CÁMARAS DE LA IMAGEN DERECHA CON LA AYUDA DE OPENCV.

Las coordenadas 3D de las esquinas se inicializan a 0 como objetos de la clase *Point3f* ya que se consideran el centro del espacio 3D. Los puntos 2D serán las proyecciones de dichos puntos 3D en cada imagen y se forzará a que se correspondan con las esquinas del tablero para ambas vistas. Para esto, se va a utilizar la función de OpenCV ***findChessboardCorners*** [28]. Dicha función es capaz de encontrar las esquinas internas del tablero grabado a partir de una imagen de dicho tablero y de las dimensiones del mismo. Es una función de tipo booleano, por lo que devuelve un valor $True = 1$ o $False = 0$ en función de si la localización de las esquinas ha sido correcta. La función trata las imágenes de una en una, por lo que se debe utilizar esta función tantas veces como *frames* se capturen del tablero y una vez por cada cámara. La imagen del tablero (*image*) y las dimensiones del mismo (*patternSize*) se establecen como parámetros de entrada, y se obtienen como parámetros de salida las coordenadas en las imágenes izquierda (*corners*) o derecha de las esquinas internas del tablero. Las coordenadas se obtienen como objetos de la clase *Point2f.*

Tras esto, se usan las funciones de OpenCV ***cornerSubPix*** y ***drawChessboardCorners*** para comprobar que la función *findChessboardCorners* se implementa correctamente. *CornerSubPix* refina las coordenadas de las esquinas internas del tablero, y *drawChessboardCorners* dibuja un foco redondo encima de ellas para poder comprobar visualmente la correcta localización de las

mismas. De esta forma, tras usar estas tres funciones de OpenCV, se obtiene una imagen del tablero con sus esquinas internas localizadas y marcadas con un círculo además de los parámetros *imagePoints1* e *imagePoints2* que se necesitan para *StereoCalibrate.*

4.2.2.2 Funciones de Rectificación

Siguiendo con la teoría del apartado 2.3 y el esquema de las figuras 2.7 y 2.4, tras realizar la calibración estéreo de las dos cámaras, se procede a realizar la rectificación de las imágenes. Para ello, se emplea la función de OpenCV ***stereoRectify*** [28]. Con esta función, se obtienen 5 matrices que nos proporcionan parámetros de rectificación que se aplicarán posteriormente a las imágenes de las cámaras. Se tiene como salidas una matriz de transformación por cada imagen (T_l y T_r), una nueva matriz de proyección por cada imagen (P_l y P_r) y una matriz de transformación[5] (Q). Como entradas de esta función se deben proporcionar las matrices con los parámetros intrínsecos de cada cámara (*cameraMatrix1* y *cameraMatrix2*), los vectores de distorsión (*distCoeffs1* y *distCoeffs2*), el tamaño de la imagen calibrada, las matrices de rotación (R) y traslación (t), el *flag cv_calib_zero_disparity* que establecerá los puntos principales de cada cámara en la misma línea epipolar, y el valor entero *alpha* igual a 0 para recortar la imagen quitado la parte deformada y corregir los márgenes de las imágenes rectificadas.

Tras esto, se utilizan dos funciones más de OpenCV capaz de aplicar todos los parámetros de rectificación a las imágenes tomadas por las cámaras. Estas funciones son ***initUndistortRectifyMap***, que aplica los parámetros de rectificación y corrección a un mapa de transformación (el cual se puede guardar en un objeto de clase *Mat*), y ***remap***, que fusiona dicho mapa de parámetros a una imagen de entrada (en este caso, las imágenes de la cámara estéreo). Se obtiene un mapa de rectificación diferente por cada cámara, por lo que se emplea cada función dos veces.

4.2.2.3 Funciones de lectura de imágenes

Para capturar imágenes de ambas cámaras en nuestro programa se utilizará la clase *VideoCapture* de OpenCV. Con esta clase se pueden leer secuencias de *frames* procedentes de una cámara incluida en el sistema, de un vídeo ya grabado o incluso de una secuencia de imágenes [34]. Esta clase permite una amplia variedad de *codecs* para la lectura de un video y de las imágenes. Para su utilización, se declara un objeto que tendrá como tipo esta clase, Cada vez que se quiera capturar un *frame* se emplea el signo >> para guardar dicho *frame* en otro objeto de la clase *Mat*. En nuestro sistema, se usa *VideoCapture* de esta forma, declarando dos objetos (uno por cámara) de esta clase después de obtener los parámetros de calibración.

[5] Ver apartado 2.3.3

4.2.3 Desarrollo atómico de los módulos del sistema

Primer módulo:

Este módulo se implementa en un proyecto independiente, donde se espera obtener como salida los parámetros de calibración y rectificación de las cámaras a partir de imágenes de un tablero de ajedrez.

Calibración Estéreo:

El primer paso será realizar la calibración estéreo de las cámaras. Se instancian dos objetos de la clase *VideoCapture* con los que tomar un número determinado de *frames* de cada cámara mientras se apunta a un tablero de ajedrez impreso. Se crea un ciclo anidado en el que, por cada iteración, se capturar un *frame* de cada cámara y se localizan las esquinas del tablero en cada imagen. Se recuerda que para localizar estas esquinas se utilizan la función *findChessboardCorners* que devolverá el valor booleano 1 o 0 dependiendo de si se han encontrado las esquinas o no. En caso afirmativo, se utiliza *CornerSubPix* para refinar las coordenadas de las esquinas, se realiza la comprobación visual con *drawChessboardCorners* y se almacenan las coordenadas de las esquinas en el vector del tipo *Point2f* para cada imagen de cada cámara.

El número de iteraciones de dicho ciclo deben ser las suficientes como para cubrir el mayor número de vistas posibles del tablero, pero no demasiadas para no producir redundancias. A continuación, se instancia una nube de puntos 3D del tamaño del número de esquinas del tablero como objetos de la clase *Point3f* y se inicializan a 0, ya que se situarán en el centro de coordenadas del espacio 3D. Por último, con todos los parámetros necesarios ya obtenidos, se utiliza la función *StereoCalibrate* de la forma descrita en el apartado 4.2.2.1, obteniendo así los parámetros intrínsecos y extrínsecos de la calibración estéreo. En el desarrollo de la implementación, los parámetros se escriben y se guardan en ficheros del tipo "*.yml"*, que es una variante especial de fichero de datos que ofrece una forma sencilla de almacenar y visualizar matrices.

Rectificación:

En esta parte, se obtienen las matrices de transformación y de proyección de cada cámara. A partir de los parámetros de calibración, guardados en un fichero "*.yml"*, éstos son dados como entradas para la función de OpenCV *stereoRectify*, cuya funcionalidad se explica en el apartado 4.2.2.2. Así se obtienen todos los parámetros de rectificación necesarios, y se guardan en otro fichero "*.yml"*, que supone la salida global de este primer proyecto. Este módulo sólo es necesario utilizarle cada vez que se quiera calibrar las cámaras. Supone un procedimiento independiente a *Stereo Matching*.

Segundo módulo:

Alineación y recorte de las imágenes:

La primera tarea de este segundo proyecto, será aplicar los parámetros de rectificación obtenidos en el módulo anterior. Este paso no es necesario en el caso de que se utilice una cámara, vídeo o imágenes estéreo ya calibrados para *Stereo Matching*. Se comienza leyendo del fichero "*.yml"* los parámetros de rectificación e introduciéndolos en la función *initUndistortRectifyMap*[6] de OpenCV. Con esto se obtiene un mapa de alineamiento aplicable a cualquier imagen con la ayuda de la función *remap*.

A continuación, se comienza con la captura de imágenes a las que aplicar el algoritmo *Stereo Matching*. Se crea un ciclo anidado en el que se instancia un objeto de la clase *VideoCapture* para cada cámara y se re-mapea cada *frame* con el mapa de rectificación obtenido. Dentro de este ciclo se incluye el desarrollo completo del algoritmo *Stereo Matching,* de forma que hasta que no se finaliza para un par de imágenes no se vuelve a capturar otro par. La captura de las imágenes izquierda y derecha de la escena se realiza en el mismo momento de tiempo y se convierten a imágenes en blanco y negro, donde el valor de los píxeles se corresponde directamente con el de su luminancia.

Filtrado de las imágenes:

Tras rectificar las imágenes, se implementa un sencillo filtrado o procesado de las imágenes, con la finalidad de combatir algunas limitaciones expuestas en el apartado 4.2.1. Las imágenes se pre-procesan para reducir los problemas de zonas con poca textura y la desigualdad en el brillo de la mismas. Una técnica utilizada es la de normalizar los histogramas de contraste para añadir textura a zonas donde esta sea imperceptible, para lo que se usa la función de OpenCV *equalizeHist*. Otra técnica empleada para filtrar las imágenes será añadir una de compensación de iluminación entre las vistas. Para ello, se emplea la función *set()*de la clase *VideoCapture*, con los *flags cap_prop_brightness y cap_prop_exposure*. Esta función se debe ejecutarse antes de capturar las imágenes, ya que se con esto se fuerza a las cámaras a capturar con el mismo brillo y exposición.

Correlación de las imágenes:

Tras pre-procesar las imágenes de entrada para favorecer el correcto funcionamiento de *Stereo Matching,* se comienza con la implementación del mismo aplicando la teoría recogida en el capítulo 3. A continuación, se van a combinar los dos primeros pasos propios de *Stereo*

[6] Ver figuras 4.1. y 4.2

Matching, (cómputo de la correspondencia y Asignación del coste) en una única tarea. Primero, se delimita una ventana o macrobloque de dimensiones *NxN* en la imagen izquierda, teniendo como referencia el píxel de la esquina izquierda de dicha ventana, llamado p_l (tarea propia de *Asignación de coste*). Dicho bloque será tratado ahora como una unidad. Después se delimita otro bloque de las mismas dimensiones en la imagen derecha teniendo como referencia el píxel de la esquina izquierda de dicha ventana p_r. Las coordenadas p_l y p_r deberán ser la misma, o p_l deberá ser igual a p_r menos un valor de desplazamiento d en el eje X (d hace referencia a máxima disparidad determinada). Es decir, que el bloque de la imagen izquierda deberá empezar en el mismo píxel que el de la imagen derecha o desplazado un valor $-d$ en el eje X.

Tras esto, se aplica la técnica SAD, combinación del *cómputo de la correspondencia* y de la *asignación del coste*. Esto es, los píxeles con mismas coordenadas *x* e *y* de cada bloque se restan entre ellos y la diferencia absoluta obtenida se acumula para todos los píxeles, esto se describe con la siguiente expresión:

$$C_{bloque}(p, d) = \sum | w_l(p) - w_r(n)|, \qquad (4.1)$$

donde p es un píxel del bloque en la imagen izquierda w_l y n un píxel del bloque en la imagen derecha w_r ; p y n tienen la misma coordenada dentro del bloque. Hay que darse cuenta que esta expresión es la misma que describe la **correlación** entre dos señales, y a partir de este punto se denominará de esta forma a la operación descrita entre macrobloques del par de imágenes. A continuación, se desplaza la ventana un valor d en la imagen derecha un píxel hacia la izquierda, y se vuelve a repetir la correlación entre la ventana izquierda anterior y la nueva en la imagen derecha. Se repetirá este último paso hasta haber desplazado el bloque de la imagen derecha *d* píxeles hacia la izquierda, habiendo guardado las SAD para cada caso.

Tras esto, se implementa el tercer paso de *Stereo Matching*, *disparidad óptima*. Se selecciona de entre todas las SAD realizadas la de menos valor, lo que significará que la diferencia de costes entre el bloque izquierdo w_l y el bloque derecho w_r habrá sido la mínima, por lo que dichos bloques serán los más parecidos. El valor *d* del SAD mínimo será, por tanto, la disparidad del píxel p_l.

$$D(p_l) = \arg\min_d \left(C_{bloque}(p_l, d)\right) \qquad (4.2)$$

Como último paso del algoritmo explicado en el capítulo 3 (refinado de la disparidad) se corrigen los errores que hayan surgido en el mapa de disparidad. Hasta ahora se ha calculado dicho mapa para la imagen izquierda, así que se calcula también para la imagen derecha. Después, se comparan ambos mapas de profundidad y se comprueba que para cada píxel se tenga la misma

disparidad o que la diferencia de disparidades no supere un determinado umbral. Si la disparidad es diferente, es decir, existe un error en el cálculo de la disparidad del píxel, se aplica la técnica *unweighted median filtering*, que corrige el píxel erróneo calculando la media de los píxeles circundantes y asignando el valor de la media a dicho píxel.

Finalmente, se escalan los valores de disparidad para obtener el mapa de disparidad en escala de grises. Para ello, se aplica la siguiente expresión:

$$D_{epth}(p_l) = [D(p_l)/\max(D)] * 255 \quad (4.3)$$

donde $\max(D)$ hace referencia a la máxima de las disparidades obtenidas entre todos los píxeles de la imagen. Por último, se aplica la triangulación del apartado 2.3.5, con la ecuación 2.16, para crear el modelo 3D que suponen los mapas de profundidad.

4.3 Evaluación de los resultados

El sistema implementado genera una gran cantidad de resultados intermedios a parte del mapa de profundidad final. Los resultados obtenidos en las diferentes partes del sistema se pueden agrupar de la siguiente forma:

- Resultados de localización de puntos (esquinas del tablero) en imágenes.
- Resultados de calibración estéreo y rectificación
- Generación de mapas de profundidad

Tras presentar el desarrollo del algoritmo y conocer todos los conceptos teóricos que éste abarca, en este apartado se realiza una evaluación de los diferentes resultados obtenidos en el sistema. Se analiza la calidad de los resultados, el rendimiento del sistema y el grado de éxito obtenido en el desarrollo del trabajo. Se buscarán los puntos débiles de la implementación para localizar posibles fallos o mal funcionamiento de las técnicas empleadas y analizar otros métodos alternativos. También se analiza la relación entre el tamaño de los macrobloques y la calidad de los resultados y rendimiento del sistema. Por último, se evaluará cómo el tamaño de las imágenes tomadas afecta a la calidad de los resultados, el rendimiento del sistema y el grado de cumplimiento de los objetivos finales de la implementación.

4.3.1 Calidad de los resultados

En este apartado se va a evaluar la calidad de los resultados que se obtendrán durante la implementación. En primer lugar, se evalúan los resultados obtenidos en la localización de las esquinas del tablero. Las cámaras utilizadas en el sistema se corresponden con dos *webcam* Logitech capaces de capturar imágenes con una resolución máxima de 960x720 píxeles. El tablero

utilizado tiene unas dimensiones de 9x6 esquinas, con unos cuadrados cuyas aristas miden 30mm. Hay que mencionar que se van a re-escalar las imágenes tomadas a calidad VGA, 640x480 píxeles, ya que se considera una resolución lo suficientemente grande como para obtener buenos mapas de profundidad y lo suficientemente pequeña como para no tardar un tiempo excesivamente elevado en computarlos.

Se obtienen las siguientes imágenes donde se localizan las esquinas del tablero en ambas imágenes:

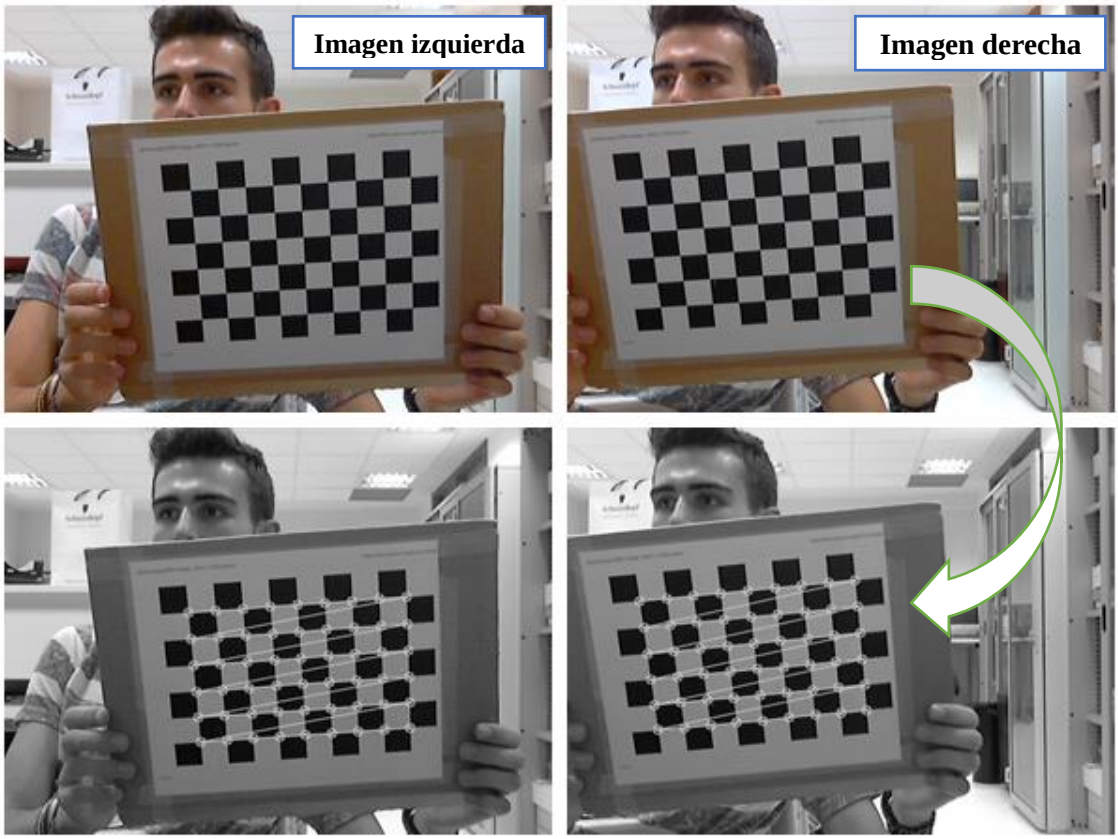


FIGURA 4.4: PRUEBA DE LA LOCALIZACIÓN DE LAS ESQUINAS DEL TABLERO

Se observa *a priori* una localización correcta de las esquinas del tablero en ambas imágenes. En la implementación de capturas hasta 30 *frames* y se localizan las esquinas en todas ellas de forma correcta. De esta forma, se continúa con la evaluación de los resultados pasando ahora a evaluar todos los parámetros de calibración estéreo obtenidos (función *StereoCalibrate*).

Los parámetros intrínsecos asociados a cada cámara se presentan en las tablas 4.1 y 4.2. También se adelantan los parámetros de rectificación (matrices de transformación T_l y T_r y de proyección P_l' y P_r' en 2.3.3). Como estas imágenes se han sido tomadas desde cámaras con una disposición casi binocular, la matriz de rotación R de la rectificación tiende a parecerse a la identidad. Finalmente, como al rectificar es muy probable modificar las dimensiones de la zona

útil de la imagen, este cambio se ve reflejado en las matrices *P*, que son diferentes a las matrices *M*.

Cámara Izquierda		Cámara Derecha	
Parámetros	**Valores**	**Parámetros**	**Valores**
$\mathbf{M_1}$	$\begin{bmatrix} 795.71 & 0 & 319.50 \\ 0 & 795.71 & 239.50 \\ 0 & 0 & 1 \end{bmatrix}$	$\mathbf{M_2}$	$\begin{bmatrix} 795.71 & 0 & 319.50 \\ 0 & 795.71 & 239.50 \\ 0 & 0 & 1 \end{bmatrix}$
$\mathbf{D_1}$	$[0.04 \quad -2.21 \quad 0 \quad 0 \quad 23.14]$	$\mathbf{D_2}$	$[0.04 \quad -2.21 \quad 0 \quad 0 \quad 23.14]$
$\mathbf{T_l}$	$\begin{bmatrix} 0.999 & 0.026 & -0.003 \\ -0.013 & 0.999 & -0.007 \\ 0.003 & 0.008 & 0.999 \end{bmatrix}$	$\mathbf{T_r}$	$\begin{bmatrix} 0.999 & 0.023 & -0.002 \\ -0.026 & 0.999 & 0.008 \\ 0.001 & -0.007 & 0.999 \end{bmatrix}$
$\mathbf{P_l}$	$\begin{bmatrix} 979.33 & 0 & 335.12 & 0 \\ 0 & 979.33 & 239.33 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	$\mathbf{P_r}$	$\begin{bmatrix} 979.33 & 0 & 335.12 & -4757.5 \\ 0 & 979.33 & 239.33 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

TABLA 4.1: PARÁMETROS INTRÍNSECOS Y DE RECTIFICACIÓN OBTENIDOS PARA LAS CÁMARAS LOGITECH

Parámetros	Valores
R	$\begin{bmatrix} 0.999 & 0.013 & 0.031 \\ -0.013 & 0.999 & -0.015 \\ -0.031 & -0.015 & 0.999 \end{bmatrix}$
t	$[-4.854 \quad -0.061 \quad 0.156]^T$

TABLA 4.2: PARÁMETROS EXTRÍNSECOS OBTENIDOS PARA LAS CÁMARAS LOGITECH

Por otro lado, los parámetros intrínsecos que relacionan ambas cámaras se pueden ver en la tabla 4.1. Los valores de *R* y t validan la configuración espacial de las cámaras, pues R se asemeja a la identidad y t representa un valor solo significativo en su primera componente. Las imágenes tomadas ofrecen una calibración adecuada con el algoritmo de calibración y rectificación proporcionado por OpenCV. A continuación, se evalúa la rectificación aplicada a las imágenes de forma cualitativa. En la figura 4.5 se recogen las imágenes rectificadas y recortadas a partir de los parámetros anteriores.

Cualitativamente, se observa en las imágenes de partida cómo las cámaras no se encuentran en el mismo eje óptico, ya que en una cámara graba zonas del techo que la otra no, y ocurre lo mismo con el suelo, por lo tanto, una se encuentra desplazada en el eje Y con respecto a la otra. Además, también es apreciable como no se encuentran alineadas una con la otra, ya que las líneas que atraviesan la imagen por completo (las líneas del techo, las mesas, el suelo etc…) no son paralelas horizontalmente. Sin embargo, si se observan las imágenes rectificadas izquierda y derecha se observa cualitativamente como se ha corregido la alineación, se han encuadrado en el mismo eje *Y* y no se observa distorsión alguna en la imagen. Aunque es apreciable también cómo

se ha reducido el ángulo de visión de las imágenes, lo que empeora la resolución de las mismas, y habrá que comprobar cómo afecta este factor al algoritmo *Stereo Matching*.

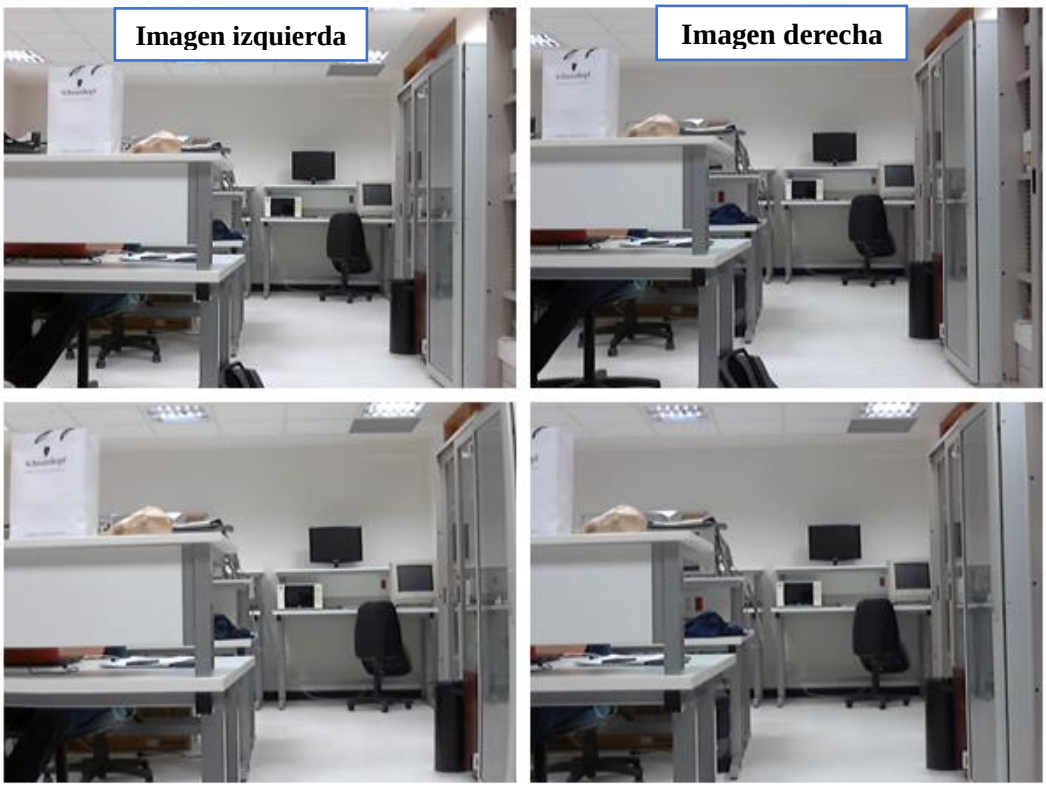


FIGURA 4.5: IMÁGENES RECTIFICADAS OBTENIDOS PARA LAS CÁMARAS LOGITECH

En cuanto a la evaluación de los resultados de los mapas de profundidad, se van a realizar dos tipos de prueba. Primero se va a evaluar el resultado final a partir de *frames* capturados de videos e imágenes estéreo de diferentes *Datasets* y después se va a estudiar el resultado obtenido al aplicar *Stereo Matching* en las imágenes procedentes de las dos cámaras Logitech en una resolución de grabación VGA.

En primer lugar, se utilizarán imágenes estéreo procedentes de ficheros del *Dataset* de *Middlebury benchmark* [26], ampliamente aceptado en el ámbito de la visión estéreo y del *Stereo Matching*. El famoso método *Ground Truth* de *Stereo Matching* pertenece a la Universidad de Middlebury y está bajo patente. Se considera de los mejores algoritmos en cuanto a mapas de profundidad. En este trabajo se tomará *Ground Truth* como el algoritmo que crea los mapas de profundidad de referencia. En la figura 4.6 se muestran los resultados obtenidos a partir de una imagen de este *Dataset*, las cuales se asumen perfectamente corregidas y alineadas.

Se observa que el resultado obtenido es adecuado ya que se consigue un mapa de profundidad donde se identifican todas las profundidades gradualmente y se pueden reconocer los objetos

perfectamente. Sin embargo, los bordes de los objetos y algunos saltos de profundidad en las imágenes son inexactos y poco claros.

A lo largo del trabajo, ya se explicó que el método que se iba a implementar no trataba bien estos cambios de profundidad y saltos entre objetos, pero se espera ganar tiempo de ejecución a cambio de una pequeña inexactitud en los mapas de profundidad. Cabe mencionar que, en este ejemplo, en las zonas de la pared la cual carece de textura, se pueden ver errores que no han sido capaz de eliminarse, aunque se haya utilizado tanto pre-procesado (ecualización del histograma de las imágenes) como post-procesado (cálculo de la disparidad en ambos sentidos y *unweighted median filtering*). Sin aplicar ninguna de estas técnicas de refinamiento se obtendría el resultado de la figura 4.7.

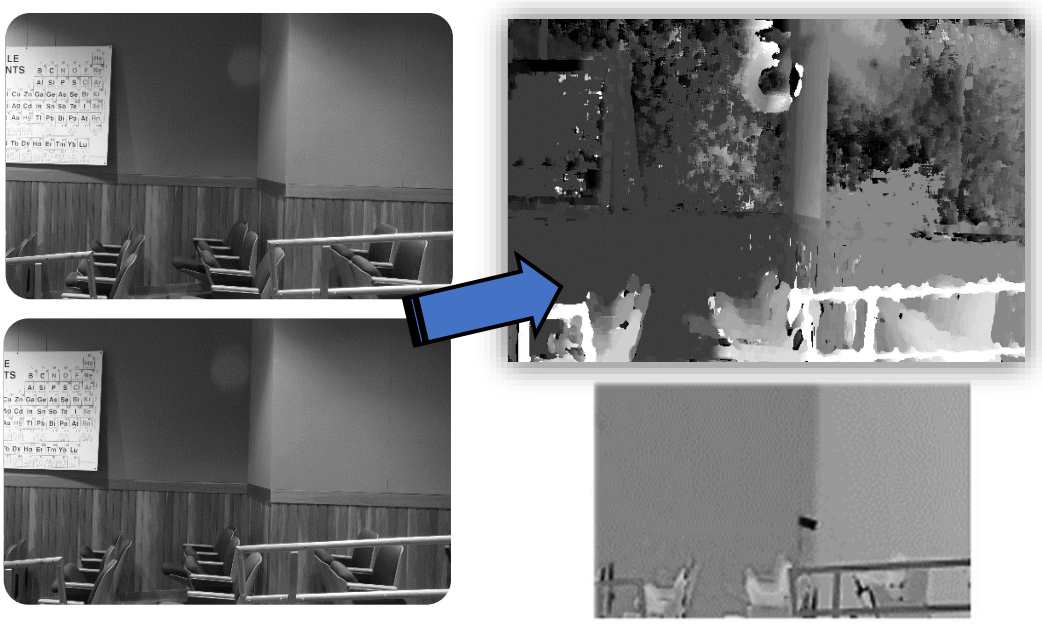


FIGURA 4.6: MAPA DE PROFUNDIDAD OBTENIDO A PARTIR DE UNA IMAGEN DEL DATASET DE MIDDLEBURY. EN EL RECUADRO DE ABAJO A LA DERECHA SE PUEDE OBSERVAR EL MAPA OBTENIDO CON GROUND TRUTH.

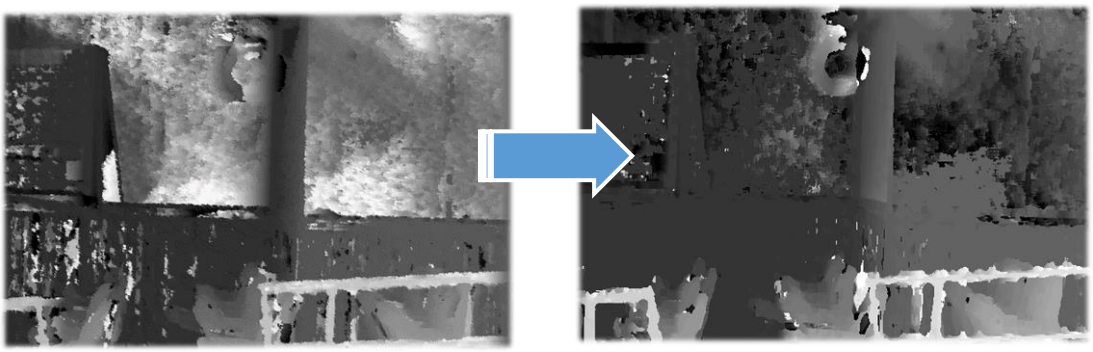


FIGURA 4.7: EN LA PRIMERA IMAGEN NO SE HA APLICADO NINGUNA TÉCNICA DE POST-PROCESADO, Y SE OBSERVAN PEORES RESULTADOS EN LAS ZONAS CON MENOS TEXTURAS.

A continuación, se analiza el mapa de profundidad obtenido a partir de un vídeo estéreo extraído del *Dataset* de la Universidad RMIT, denominado RMIT3DV [25]. En éste se pueden encontrar más de 30 vídeos estéreo en *FULL HD* perfectamente alineados y corregidos. En la figura 4.8 se muestra el resultado obtenido al aplicar *Stereo Matching* en uno de los videos del *Dataset* RMIT3DV. En esta prueba se ha re-escalado las imágenes de entrada a un tamaño de 640x360 píxeles:

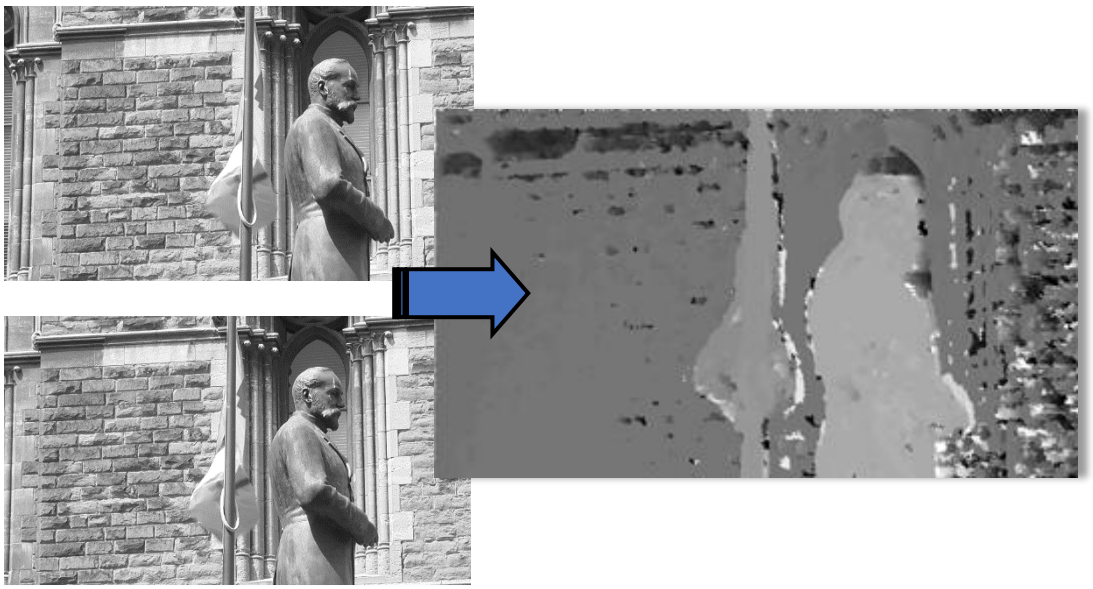


FIGURA 4.8: MAPA DE PROFUNDIDAD OBTENIDO A PARTIR DE LOS *FRAMES* DE UN VIDEO ESTÉREO DEL *DATASET* DEL DATASET DE RMIT3DV.

Se aprecia que el resultado obtenido en este caso también es adecuado, ya que se diferencian todos los objetos a diferentes profundidades, aunque sigue teniendo errores de inexactitudes en los bordes, algo que es de esperar debido a las técnicas escogidas. Este ejemplo posee una pared en la escena con una textura marcada y clara pudiendo se observa como *Stereo Matching* trabaja correctamente y otorga la misma disparidad a toda la pared, apreciando en el mapa de disparidad que se le da el mismo tono de gris.

Por último, se examinan los resultados obtenidos en el caso de obtener las imágenes estéreo desde una cámara estéreo o desde dos cámaras individuales acopladas a una montura horizontal. En este caso, será necesario computar los parámetros de calibración obtenidos en el primer módulo del sistema. En un primer momento se utilizaron unas cámaras estéreo de la marca *OVRVision* para obtener las imágenes (figura 4.3). Sin embargo, las cámaras estéreo poseían una deformación de tipo ojo de pez muy fuerte y además estaban descentradas, no centrando las dos lentes en el mismo eje óptico. Todo esto hacía que, después de calibrar las imágenes, el ángulo de visión final y la resolución eran muy poco adecuados para aplicar *Stereo Matching*, por lo que se optó por utilizar dos webcam Logitech (también en la figura 4.3, acopladas al poste de madera)

capaces de capturar imágenes con una resolución máxima de 960x720 píxeles. Tras adaptar el programa para que el sistema capture las imágenes de las cámaras y las calibre, se ejecuta y se obtienen los siguientes resultados:

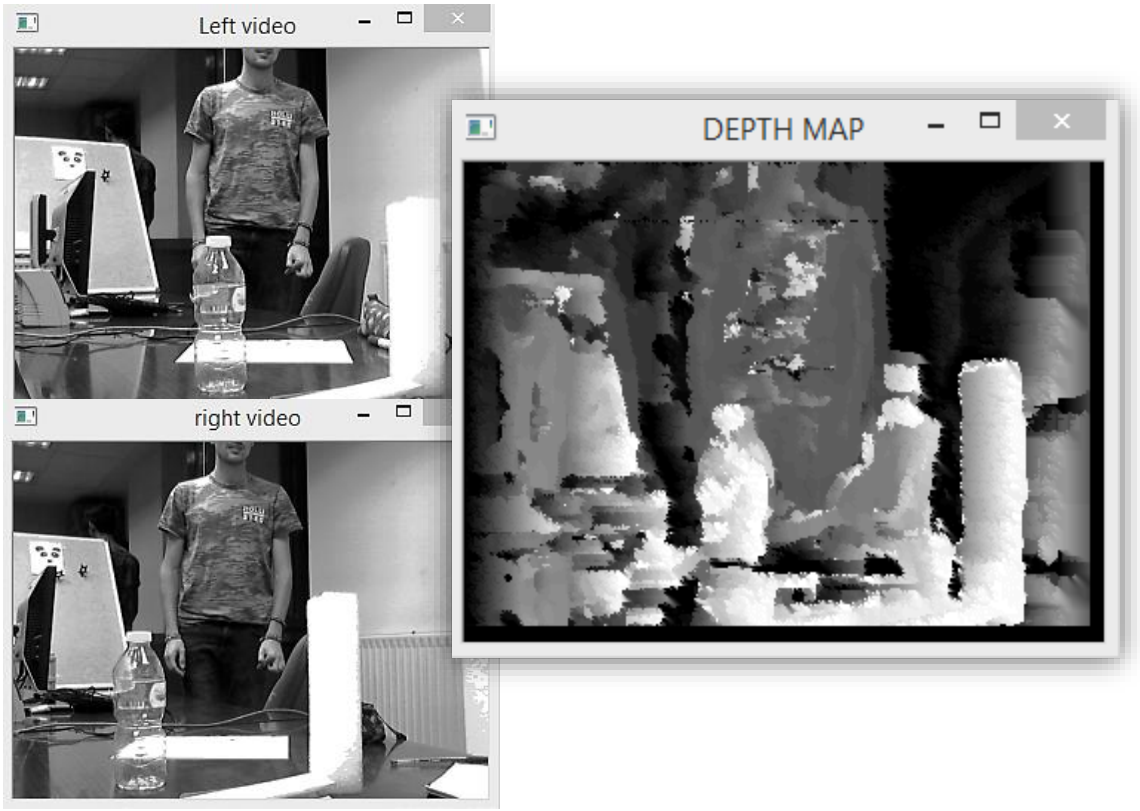


FIGURA 4.9: MAPA DE PROFUNDIDAD OBTENIDO A PARTIR DE IMÁGENES TOMADAS CON DOS *WEBCAM* ALINEADAS Y CALIBRADAS. LA IMAGEN PROCESADA ES DE UN TAMAÑO DE 320X240 CON UN TAMAÑO DE MACROBLOQUE DE 8X8.

Se observa que la calidad del mapa de profundidad obtenido a partir de las cámaras Logitech no posee tanta calidad como los mostrados anteriormente, aunque sí se pueden apreciar objetos en el mapa y las diferentes profundidades están bien calculadas. Además, se observará en la figura 4.14 que el tiempo de cómputo de este mapa de profundidad ronda los 0,6 segundos, por lo que se obtiene una implementación bastante rápida.

4.3.2 Rendimiento del sistema

En este apartado se evaluará el tiempo que tarda el sistema en obtener el mapa de profundidad de salida. Para este análisis, se debe tener en cuenta el tiempo parcial de todos los procesos y ver donde se encuentra el cuello de botella de nuestra implementación. Nos centraremos en el análisis temporal del módulo que implementa el algoritmo *Stereo Matching*, y no del módulo de calibración y rectificación de las cámaras.

Resulta intuitivo pensar que el tiempo de cómputo empleado por el sistema para la obtención del mapa de profundidad será proporcional al tamaño de la imagen. Se analiza de forma cualitativa la relación entre el tamaño de la imagen y el tiempo de cómputo de las diferentes partes del algoritmo. De esta manera se espera reforzar también la identificación de las zonas sensibles del algoritmo y cuellos de botella. Por último, se reforzará la evaluación del rendimiento del sistema evaluando el grado de influencia que tiene el tamaño del macrobloque en el algoritmo. Todas las pruebas aquí mencionadas se realizarán de forma Software en un PC cuyas características técnicas son las siguientes: Procesador de 2,2 GHz *Inter Core* i5 CPU de 4 *cores* y 8 GB de memoria RAM. La velocidad de cómputo es muy elevada y los 4 *cores* del procesador permiten que el código se paralelice hasta 4 hilos.

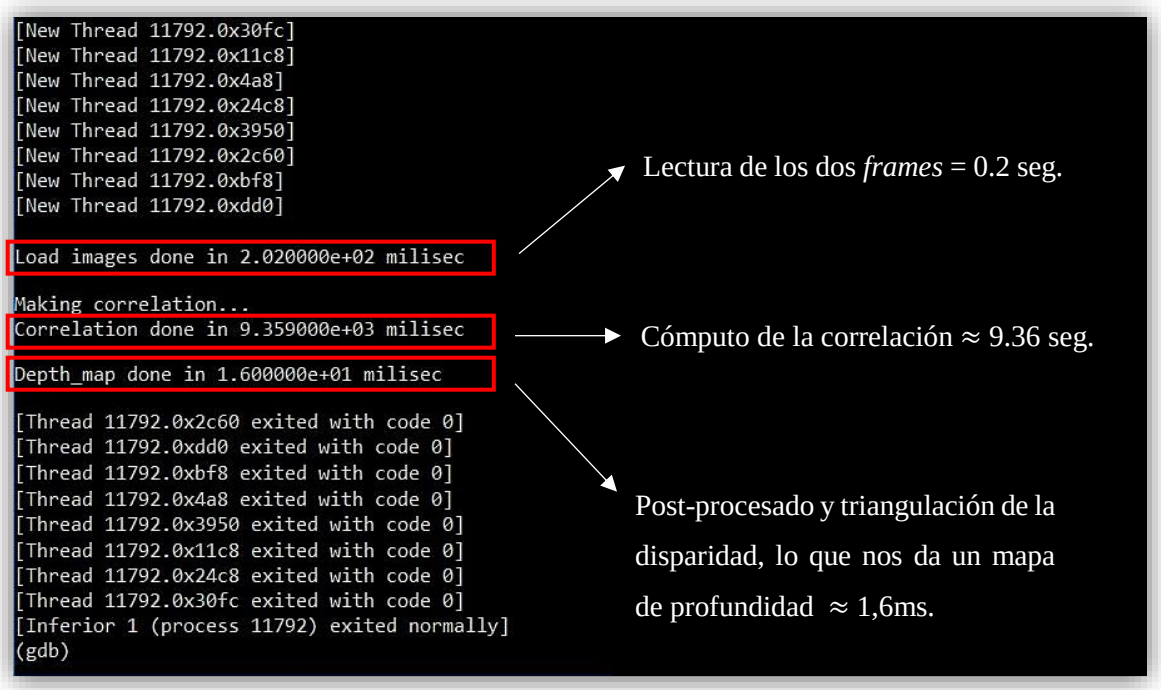


FIGURA 4.10: EJECUCIÓN DEL SISTEMA REALIZADA EN *CYGWIN* DONDE SE PUEDEN VER LOS RESULTADOS OBTENIDOS.

Antes de realizar las pruebas temporales del sistema, se espera que el principal cuello de botella del algoritmo se localice en la parte de la correlación, donde se implementan las tareas de cómputo de la correspondencia, asignación del coste, y cálculo de la disparidad óptima; 3 de los 4 principios de *Stereo Matching*. Para la ejecución del sistema implementado se utiliza la consola de Cygwin, que es una herramienta a la semejanza de Linux o Windows que permite la ejecución de ficheros. En la figura 4.10 se muestran los resultados temporales parciales (es decir, de las diferentes partes del segundo módulo) obtenidos durante la obtención del mapa de profundidad de una imagen de tamaño 640x360 con un tamaño de macrobloque de 8x8 y una ventana de disparidad máxima de 60 píxeles.

De dicha figura, se puede deducir rápidamente que el principal cuello de botella del sistema se encuentra en la parte de la correlación, tal y como se había predicho. El tiempo de cómputo de esta parte es tan grande con respecto al de las otras partes que hace que los demás tiempos obtenidos sean despreciables. Para el caso de una imagen de tamaño 640x360 con un tamaño de macrobloque de 8x8 y una ventana de disparidad máxima de 60 píxeles se obtiene un tiempo de cómputo de la correlación de 9,36 segundos aproximadamente. Este valor temporal hace que nuestro sistema se aleje del objetivo de implementar un *Stereo Matching* que compute en tiempo real. La única opción de acelerar nuestra implementación sin modificar parámetros como el tamaño de imagen, tamaño de macrobloque o ventana de disparidad máxima es paralelizar el código con alguna técnica software.

En nuestro sistema se utiliza **OpenMP** para conseguir paralelizar el código. Esta supone la mejor opción de paralelización para este caso, ya que resulta una técnica sencilla y apropiada dado los recursos *Hardware* (HW) que nos ofrece nuestro PC (4 *cores* y una velocidad de procesado elevada de 2,2GHz). El estándar OpenMP se basa en la creación de nuevos *threads* o hilos mediante el uso de directivas o *pragmas*. Se ejecutan así secciones de código de forma paralela. Cada *thread* utilizará uno de los *cores* de la tarjeta gráfica de nuestro PC y será capaz de ejecutar el código o parte de él de forma paralela al resto de *threads*. Cada *thread* puede poseer memoria de datos privada y tiene acceso a la memoria global del programa. Para paralelizar con OpenMP habrá que tener cuidado con lectura de las variables compartidas. Las variables privadas de cada *thread* no corren peligro de dependencia (lo que puede hacer que el resultado final sea erróneo) sin embargo las variables de entorno o globales sí. Es por eso que será necesario estudiar el programa que nos ocupa y determinar qué opción de palalelización con OpenMP es la mejor.

Teniendo esto en cuenta, la mejor estrategia para la paralelización de nuestro sistema será; en primer lugar, (1) hacer las lecturas de las imágenes de forma simultánea, forzando a que un *core* del procesador se encargue de la lectura de la imagen derecha y otro *core* diferente, al mismo tiempo, lea la imagen izquierda. Después, (2) se paralelizará el paso de la correlación, que supone la tarea más compleja temporalmente de nuestro algoritmo. Se dividirá la imagen horizontalmente en cuatro porciones iguales, de forma que cada uno de los *cores* disponibles (hasta cuatro) se encargue de computar una porción diferente. Esta es la máxima paralelización posible de realizar con OpenMP dadas las características HW de nuestro PC (4 *cores*).

Una vez implementado esto en el código, se realiza una nueva prueba y se examina la mejora en el tiempo de cómputo del mapa de profundidad. Se vuelve a tomar como ejemplo un video estéreo de tamaño 640x360 con un tamaño de macrobloque de 8x8 y una ventana de disparidad máxima de 60 píxeles:

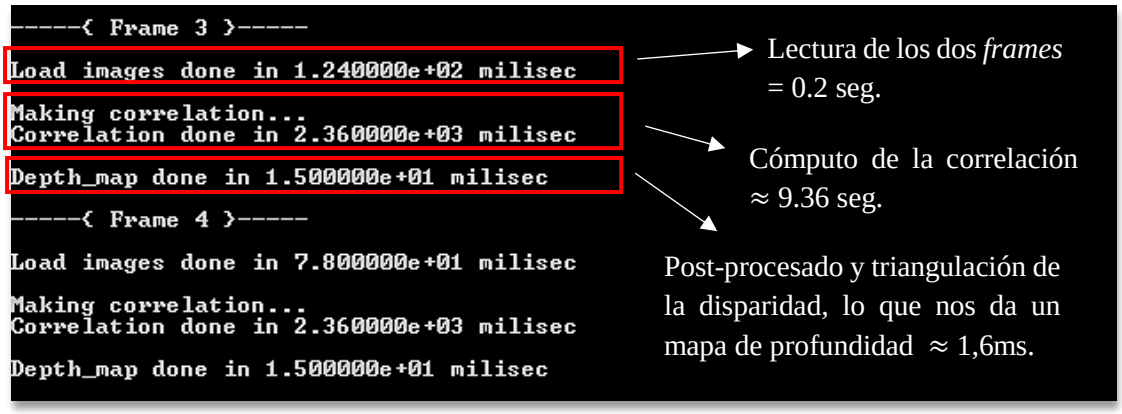


FIGURA 4.11: EJECUCIÓN DEL SISTEMA REALIZADA EN *CYGWIN* DONDE SE PUEDEN VER LOS RESULTADOS OBTENIDOS DEL SISTEMA PARALELIZADO CON OPENMP.

Se visualiza una mejora muy notable, obteniendo un tiempo de 2,36 segundos en la correlación de las dos imágenes, que es la parte que se ha paralelizado con OpenMP. Esto supone haber reducido en aproximadamente un factor 4 el tiempo de cómputo de esta parte, que es lo esperado ya que se ha paralelizado dicho proceso en cuatro hilos. En las siguientes figuras se recogen los resultados temporales obtenidos variando parámetros como el tamaño de la imagen procesada y el tamaño de bloque por separado, pudiendo estudiar así la influencia de estos factores en el rendimiento del sistema.

En la figura 4.12, se puede apreciar como el tiempo de cómputo de la correlación aumenta según crece el tamaño de las imágenes de entrada, estableciendo una relación de proporcionalidad directa entre ambos factores. A continuación, se muestra en la figura 4.13 el mismo tipo de experimento de análisis pero variando el tamaño del macrobloque:

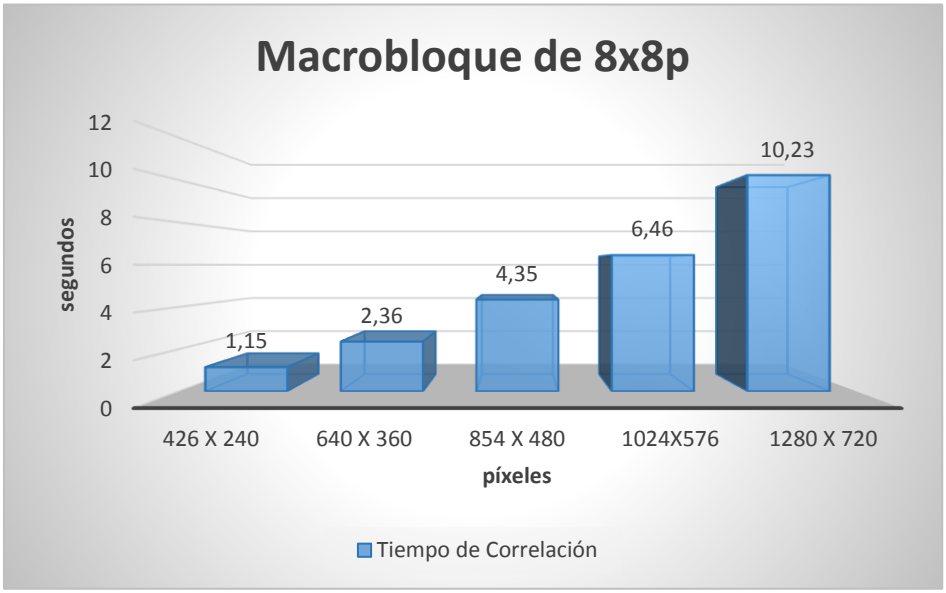


FIGURA 4.12: TIEMPOS DE CÓMPUTO DE LA CORRELACIÓN PARA DIFERENTES TAMAÑOS DE IMAGEN.

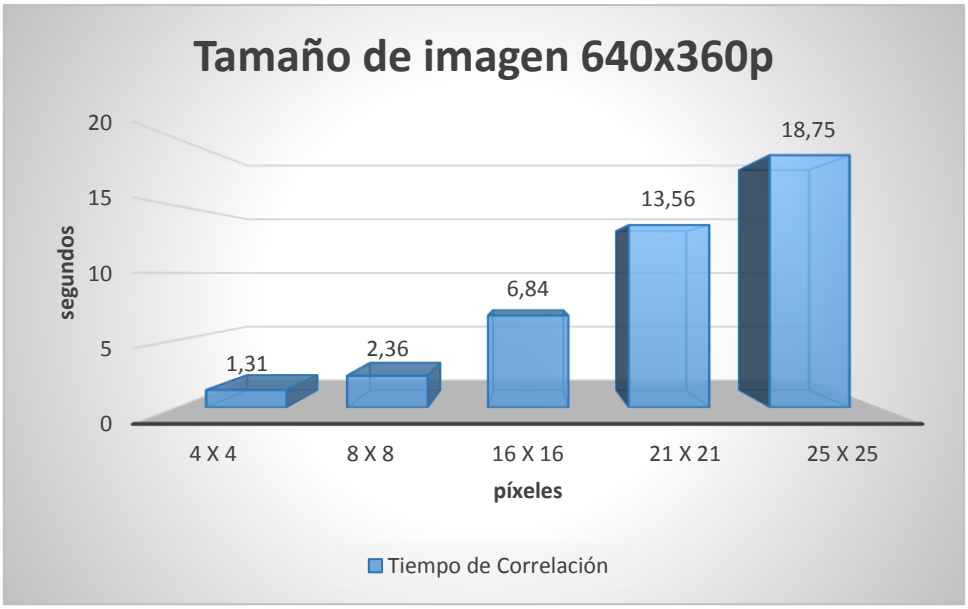


FIGURA 4.13: TIEMPOS DE CÓMPUTO DE LA CORRELACIÓN PARA DIFERENTES TAMAÑOS DE IMAGEN.

Se observa como el tiempo de cómputo de la correlación aumenta de forma aparentemente exponencial según incrementa el tamaño de macrobloque establecido. Esto es algo esperado, ya que al aumentar bien el tamaño de imagen (figura 4.12) o el tamaño de macrobloque (4.13) el ciclo anidado que recorre las imágenes y computa su correlación va aumentando el número de iteraciones. Se observa que para un tamaño de imagen de 426x240 con macrobloques de 8x8 el tiempo de ejecución se acerca a un mapa de profundidad por segundo, y lo mismo ocurre en el caso de imágenes de 640x360 con macrobloques de 4x4, por lo que una combinación de estos valores en una prueba puede acercarnos a mejores resultados temporales. Esto se demuestra en el ejemplo ejecutado en la figura 4.9, donde se obtienen los siguientes resultados temporales:

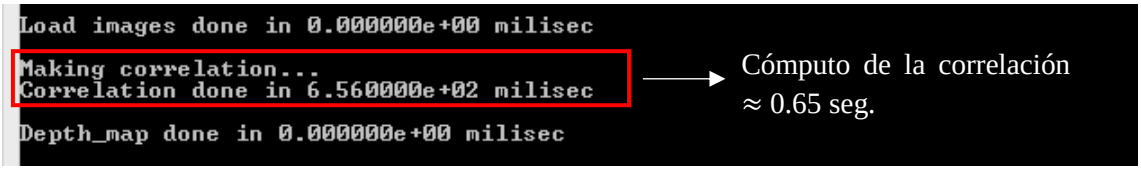


FIGURA 4.14: RESULTADOS TEMPORALES DE LA EJECUCIÓN MOSTRADA EN LA FIGURA 4.9 CON RESOLUCIÓN DE 320X240 PÍXELES Y CAMBIANDO EL TAMAÑO DE MACROBLOQUE A 4X4 PÍXELES.

4.3.3 Conclusión

Tras la evaluación de todos los resultados obtenidos, se llega a una primera conclusión en relación con el rendimiento del sistema implementado. El paso de la correlación supone ser el más importante en la propuesta de *Stereo Matching* desarrollada, ya que además de recoger en un solo paso 3 de los 4 principios teóricos del algoritmo, también se ha visto que computacionalmente es el parte más exigente. Se ha comprobado que el tamaño de la imagen es

fundamental a la hora de obtener una implementación en tiempo real, y que un tamaño cada vez más grande de macrobloques afecta muy negativamente al tiempo de cómputo del algoritmo, además de favorecer un mapa de profundidad final más inexacto en las zonas con bordes marcados y saltos entre objetos, favoreciendo únicamente las zonas con poca textura (ver figura 3.6).

Por otro lado, se concluye que los mapas de profundidad se obtienen con una calidad aceptable, sobre todo por parte de imágenes de los *Dataset,* que resultan videos e imágenes estéreo perfectamente calibrados, sin distorsiones, alineados y con una nitidez muy adecuada. Los errores producidos en este caso tienen que ver con correspondencias erróneas provocados por oclusiones de objetos, periodicidades o zonas sin textura. En el caso de los mapas de profundidad a partir de las cámaras Logitech, el problema de correspondencia errónea tiene que ver con el brillo desigual de las vistas, objetos ocluidos y una falta de calidad y excesivo ruido en las imágenes tomadas. Además, se concluye que ha sido posible alcanzar un tiempo de cómputo de 0,6 segundos por cada mapa de profundidad con imágenes de un tamaño de 320x240 píxeles y tamaño de macrobloque a 4x4 píxeles, por lo que se alcanza el objetivo de realizar un sistema que obtenga la salida en tiempo real, o al menos a más de una salida por segundo.

4.4 Propuestas de mejora

Tras el desarrollo de la implementación en todos los apartados anteriores, se van a presentar algunas mejoras que se pueden llevar a cabo en nuestro sistema. Estas tendrán que ver con la calidad de los mapas de profundidad, con el rendimiento del sistema y con la geometría de las cámaras.

4.4.1 Calidad de los mapas de profundidad

Las mejoras en este aspecto pueden ir desde la inclusión de nuevas técnicas de post-procesado del mapa de profundidad hasta el empleo de técnicas diferentes en todos los pasos del algoritmo implementado. Sin embargo, hay que tener en cuenta que el hecho de mejorar la calidad de la salida lleva inherente la consecuencia de un mayor tiempo de ejecución de la implementación. Los errores en los mapas de profundidad provienen de los factores recogidos en el apartado 4.2.1. Tras evaluar el resultado obtenido en el apartado 4.3.1 sobre la calidad de los mapas de profundidad, se proponen las siguientes mejoras:

- El empleo del gradiente en lugar de SAD.
- El problema de las oclusiones se puede solucionar con un algoritmo de detección de bordes
- Problema de inexactitud en los saltos entre objetos.

La principal ventaja de nuestro sistema es que se vale únicamente de los niveles de gris de las imágenes de entrada para obtener el mapa de profundidad. Un método más refinado de caracterización y correspondencia de puntos necesita más información que la que ofrece el canal de luminancia. Una propuesta sería emplear los colores de la imagen para la búsqueda de correspondencias en las imágenes.

4.4.2 Rendimiento del sistema

El margen de mejora en este aspecto del sistema es más pequeño, ya que se escogieron las técnicas más rápidas y sencillas computacionalmente. Entre las pocas opciones posibles de mejora, la mejor vista será la implementación y programación del sistema en otras plataformas HW a parte del PC. El cómputo de la correlación ralentiza en gran medida el tiempo final de ejecución del programa. Si bien, este cómputo tiene alto grado de paralelización, bien dividiendo las imágenes en multitud de trozo horizontales o paralelizando las iteraciones de la correlación que sí son independientes entre sí. Para ambos casos, la implementación se vuelve natural en una GPU (*Graphic processing Unit*), donde se disponen de cientos de procesadores. Cada uno podrá procesar distintas partes de la imagen a la vez, logrando una paralelización considerable y una reducción del tiempo de cómputo muy importante.

4.4.3 Geometría y otras características de las cámaras

Con el fin de generar un mapa de profundidad representativo y preciso, es fundamental contar con una buena estimación de los parámetros asociados a las cámaras. En este trabajo, se limitó el proceso de calibración al uso de funciones de la librería OpenCV. Si bien la calibración parecía adecuada de forma cualitativa, se ha visto que a la hora de utilizar las imágenes capturadas con las cámaras Logitech, el algoritmo *Stereo Matching* ha ofrecido más errores de los aceptables, algo que no ha ocurrido en las pruebas con imágenes de videos e imágenes estéreo sacadas de *Datasets* donde se emplearon cámaras estéreo corregidas para la toma de las mismas.

Es necesario la búsqueda de un mecanismo que resuelva esta tarea eficazmente, aunque este problema dista de ser nuevo y ha sido tratado en abundante literatura. Además, ha ocurrido que la zona útil de las imágenes se ha reducido considerablemente tras aplicar la calibración, lo que dificulta aún más el proceso de *Matching*. Es posible que una aplicación directa de la teoría explicada en el apartado 2.3 ayude a obtener una mejor calibración. Sin embargo, todo apunta a que se deben utilizar cámaras con una calidad mayor.

Capítulo 5
Conclusión

A lo largo de este trabajo, se ha analizado una técnica de estimación de la correspondencia entre píxeles de imágenes estéreo basado en *Stereo Matching,* paso necesario en la generación de modelos 3D. En primer lugar, se ha presentado el estado del arte del algoritmo, se ha hablado del marco teórico que le envuelve y se han analizado las técnicas y principios fundamentales del algoritmo. El marco teórico y el estado del arte han sido tratados con bastante profundidad, con la clara intención de proporcionar las claves necesarias para entender el desarrollo de un algoritmo de *Stereo Matching*.

También se ha desarrollado una implementación del algoritmo de *Stereo Matching*, permitiendo la evaluación de resultados, identificando ventajas/desventajas y permitiendo poder identificar líneas de mejora. Uno de los requisitos del proyecto era el desarrollo de una implementación de *Stereo Matching* en tiempo real, lo cual permite un amplio rango de aplicaciones. Del desarrollo de este trabajo, se extraen las siguientes conclusiones:

- Es posible obtener mapas de profundidad de calidad aceptable de imágenes estéreo a alta velocidad de un sistema correctamente calibrado. Se ha proporcionado una solución adecuada para el caso particular de imágenes perfectamente corregidas.
- Se ha explorado la utilización de *Webcam* comerciales para la toma de imágenes, con las cuales se han obtenido mapas de profundidad de calidad aceptable, a pesar de que el sistema de soporte de las cámaras era poco robusto.
- Ha sido posible calibrar las cámaras Web utilizando el modelo de cámara *Pinhole* y la teoría de rectificación con la ayuda de la librería OpenCV. Se ha solucionado así el problema de desajuste entre los ejes ópticos de las cámaras además de las distorsiones en las vistas, lo que ha contribuido en gran medida a la obtención de mapas de profundidad adecuados.
- Surge la necesidad de utilizar otras técnicas que ofrezcan mejor calidad en los mapas de profundidad, lo cual se plantea en los puntos de propuestas de mejora. Si se desea mejorar

la calidad los mapas de profundidad, se debe aumentar el tiempo de ejecución, el cómputo y mejorar el algoritmo de correspondencia.

- Que gracias a las técnicas de pre-procesado y post-procesado planteadas en la literatura sobre este algoritmo, se han podido mejorar los resultados de mapa de profundidad obtenidos con el algoritmo de *Stereo Matching* propuesto. Estas técnicas no perjudican en gran medida el tiempo de ejecución del sistema, y funcionan adecuadamente tanto para el caso de imágenes estéreo de *Datasets* como para imágenes estéreo grabadas desde las cámaras *Web*.

BIBLIOGRAFÍA

[1] Rhys Hawkins, Digital Stereo Video: display, compression and transmission, Australian National University, 2002.

[2] Gerd Bruder, Frank Steinicke, and Wolfgang Sturzlinger, "Effects of visual conflicts on 3d selection task performance in stereoscopic display environments", in 3D User Interfaces (3DUI), 2013 IEEE Symposium on. IEEE, 2013.

[3] D. A. Forsyth and J. Ponce, Computer Vision A Modern Approach. Prentice Hall, Upper Saddle River, 2003.

[4] R. I. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision. Cambridge University Press, 2000.

[5] M. Pollefeys, "Visual 3d modeling from images," http://www.cs.unc.edu/ ~marc/tutorial/index.html, tutorial Notes, Accessed November 4, 2014.

[6] T. Beeler, B. Bickel, P. Beardsley, B. Sumner, and M. Gross, "High-quality single-shot capture of facial geometry," ACM Transactions on Graphics, vol. 29, no. 3, pp. 40:1–40:9, 2010.

[7] Hartley, R; Zisserman, A., *Multiple view geometry in computer vision*, 2nd ed. Cambridge: Cambridge University Press, 2005.

[8] OpenCV 2.4 Documentacion: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html?highlight=stereocalibrate

[9] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz, "Fast costvolume filtering for visual correspondence and beyond," in Proceedings of IEEE Bibliography Conference on Computer Vision and Pattern Recognition, 2011, pp. 3017– 3024.

[10] F. Tombari, S. Mattoccia, L.D. Stefano, E. Addimanda. Classification and evaluation of cost aggregation methods for stereo correspondence. IEEE International Conference on Computer Vision and Pattern Recognition, Anchorage, Alaska (2008), pp. 1-8

[11] M. Harville, Stereo person tracking with adaptive plan-view templates of height and occupancy statistics Image and Vision Computing 22(2) pp 127-142, February 2004

[12] R. D. Arnold, “Automated stereo perception,” Ph.D. dissertation, Stanford University, 1983. 25

[13] M. D. Levine, D. A. O’Handley, and G. M. Yagi, “Computer determination of depth maps,” Computer Graphics and Image Processing, vol. 2, no. 2, 1973.

[14] P. Fua, “A parallel stereo algorithm that produces dense depth maps and preserves image features,” Machine Vision and Applications, vol. 6, no. 1, pp. 35–49, 1993.

[15] A. Fusiello, E. Trucco, and A. Verri, “A compact algorithm for rectification of stereo pairs,” Machine Vision and Applications, vol. 12, no. 1, pp. 16–22, Jul. 2000.

[16] D. Scharstein and R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” International Journal of Computer Vision, vol. 47, no. 1-3, pp. 7–42, Apr. 2002. 21, 25, 33, 49, 50, 51, 53, 55, 66, 84, 85, 106

[17] Y. Boykov, O. Veksler, and R. Zabih, “Fast approximate energy minimization via graph cuts,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 11, pp. 1222–1239, Nov. 2001.

[18] T. Meltzer, C. Yanover, and Y. Weiss, “Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation,” in Proceedings of IEEE International Conference on Computer Vison, 2005, pp. 428–435.

[19] S. Mattoccia and F. Tombari, “Stereo vision enabling precise border localization within a scanline optimization framework,” in Proceedings of Asian Conference on Computer Vision, 2007.

[20] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister, “High-quality realtime stereo using adaptive cost aggregation and dynamic programming,” in Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT’06), 2006, pp. 798–805.

[21] Q. Yang, “A non-local cost aggregation method for stereo matching,” in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 1402–1409.

[22] “Recursive bilateral filtering,” in Proceedings of European Conference on Computer Vision, 2012.

[23] G. M. Bilmes, M. Morita. “Digitalización 3d para documentación de objetos patrimoniales e instalaciones de media art”, 2016, http://conversaonline.wixsite.com/conversa/morita

[24] H. Hirschmuller, “Evaluation of cost functions for stereo matching,” in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[25] E. Cheng, P. Burton, J. Burton, A. Joseski, I. Burnett, "RMIT3DV: Pre-Announcement of a Creative Commons Uncompressed HD 3D Video Database," in*Proc. 4th International Workshop on Quality of Multimedia Experience (QoMEX 2012)*, Yarra Valley, Australia, 5-7 July 2012.

[26] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nesic, X. Wang, and P. Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German Conference on Pattern Recognition (GCPR 2014), Münster, Germany,* September 2014.

[27] K. Sun, “Adaptive motion estimation based on statistical sum of absolute difference,” in Proceedings of IEEE International Conference on Image Processing, 1998.23

[28] OpenCV 3.0 Documentation:
https://docs.opencv.org/3.0-beta/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

[29] I. Haritaoglu, D. Harwood, and L. S. Davis, “W 4 s: A real-time system for detecting and tracking people in 2 1/2d,” in Proceedings of Proceedings of European Conference on Computer Vision, 1998.23

[30] RAI TV: http://www.rai.it/dl/RaiTV/programmi/media/ContentItem-fbb80bea-9d96-44ea-ae62-1fa3b5e572a5-tgr.html?refresh_ce

[31] M. Gong, R. Yang, L. Wang, and M. Gong, “A performance study on different cost aggregation approaches used in real-time stereo matching,” International Journal of Computer Vision, vol. 75, no. 2, pp. 283–296, 2007.

[32]F. Tombari, S. Mattoccia, L. D. Stefano, and E. Addimanda, “Classification and evaluation of cost aggregation methods for stereo correspondence.” In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[33] OpenCV 3.1 Documentation: https://docs.opencv.org/3.1.0/

[34] OpenCV 3.1 Documentation:
https://docs.opencv.org/3.1.0/d8/dfe/classcv_1_1VideoCapture.html

[35]T. Kanade and M. Okutomi, “A stereo matching algorithm with an adaptive window: Theory and experiment,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, no. 9, 1994.