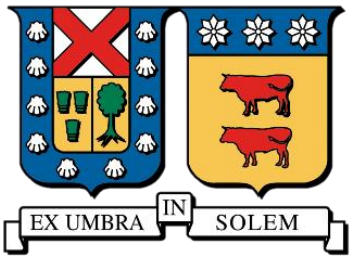
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTRÓNICA

VALPARAÍSO – CHILE



"GENERACIÓN DE VIDEO SINTÉTICO DE POSICIÓN CONFIGURABLE A PARTIR DE VARIOS VIDEOS FIJOS"

SEBASTIÁN IAN ANTONIO UBIERGO REYES

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICO

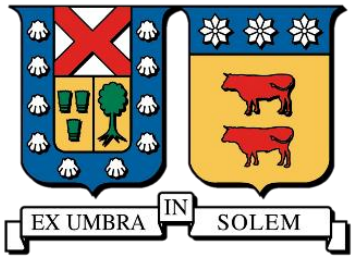
PROFESOR GUÍA: AGUSTÍN GONZÁLEZ VALENZUELA

ABRIL – 2021

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTRÓNICA

VALPARAÍSO – CHILE



“GENERACIÓN DE VIDEO SINTÉTICO DE POSICIÓN CONFIGURABLE A PARTIR DE VARIOS VIDEOS FIJOS”

SEBASTIÁN IAN ANTONIO UBIERGO REYES

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICO

PROFESOR GUÍA: AGUSTÍN GONZÁLEZ VALENZUELA

PROFESOR CO-REFERENTE: FERNANDO AUAT CHEEIN

ABRIL – 2021

Dedicado a mi padre, madre y hermanas

Resumen

En la actualidad cada vez son más las aplicaciones que aprovechan la realidad virtual para sumergir al usuario en un ambiente virtual con fines de entretención, educación, tratamientos psicológicos entre otros. Sin embargo, se ha detectado en varias de estas aplicaciones, que cuando se desea sumergir al consumidor en un escenario real, a este último se le limitada los lugares en que puede estar. Esta decisión de diseño resulta comprensible teniendo en cuenta que resulta imposible contar con cámaras para todas las posibles posiciones en que el usuario pueda llegar a posicionarse. Debido a esto último, y con la intención de que en futuros trabajos se puedan desarrollar aplicaciones de realidad virtual que permitan un libre movimiento en un escenario real, es que con este proyecto se busca poder generar una vista sintética del escenario a partir de señales de videos entregadas por cámaras fijas y finitas.

Para lograr el objetivo del proyecto, este se dividió en tres etapas de desarrollo. La primera etapa consiste en desarrollar un sensor de imagen-profundidad compuesto por dos cámaras con el cual se pueda extraer una imagen a color del escenario junto con su información de profundidad de cada pixel. La segunda etapa se centra en utilizar la información entregada por dos sensores de imagen-profundidad para generar un nuevo punto de vista de la escena, para ello se genera una imagen de la perspectiva deseada por cada sensor y luego se procede a combinarlas en una sola imagen. Finalmente se desarrolla de una interfaz gráfica que permita visualizar en un monitor el video sintético y poder manipular la posición en que se desea generar.

El proyecto logró resultados satisfactorios, permitiendo al usuario del sistema generar una vista sintética de la escena y generar un video a partir de esa perspectiva.

Palabras claves: Realidad virtual, vista sintética, sensor de imagen-profundidad, cámaras.

Abstract

Nowadays, there are more and more applications that take advantage of virtual reality to immerse the user in a virtual environment for entertainment, education, psychological treatments, among others. However, it has been detected in several of these applications, that when you want to immerse the consumer in a real scenario, the latter is limited in the places where he can be. This design decision is understandable considering that it is impossible to have cameras for all possible positions in which the user can be positioned. Due to the latter, and with the intention of developing virtual reality applications that allow free movement in a real scenario in future works, this project seeks to generate a synthetic view of the scenario from video signals delivered by fixed and finite cameras.

To achieve the objective of the project, it was divided into three stages of development. The first stage consists of developing an image-depth sensor composed of two cameras with which a color image of the scene can be extracted along with its depth information of each pixel. The second stage focuses on using the information provided by two image-depth sensors to generate a new viewpoint of the scene, generating an image of the desired perspective for each sensor and then proceeding to combine them into a single image. Finally, a graphic interface is developed to visualize the synthetic video on a monitor and to be able to manipulate the position in which it is generated.

The project achieved satisfactory results, allowing the system user to generate a synthetic view of the scene and generate a video from that perspective.

Keywords: Virtual reality, synthetic view, depth-image sensor, cameras.

Glosario

- **3D:** Tres dimensiones.
- **a:** Vector de ángulos de rotación que definen la matriz de rotación *R*.
- **A:** Matriz de la cámara.
- **B:** Separación entre los puntos focales de las cámaras.
- **BM:** Clase *"StereoBM*" de la librería de OpenCV [1]. La sigla viene de "Block Maching".
- **c_{py}:** Factor de corrección en el eje y.
- **c_{px}:** Factor de corrección en el eje x.
- **c_x:** Valor de la coordenada x del pixel central de la imagen entregada por el sensor izquierdo.
- **c_x':** Valor de la coordenada x del pixel central de la imagen entregada por el sensor derecho.
- **c_y:** Valor de la coordenada y del pixel central de la imagen entregada por el sensor izquierdo.
- **d(x,y):** Disparidad, diferencia (en pixeles) entre dos pixel que contengan la proyección del mismo punto del espacio, asociada al pixel (x,y).
- **DVR:** Grabador de video digital o "Digital Video Recorder".
- **f:** Distancia focal.
- **FPGA:** Matriz de puertas lógicas programable (en inglés "field-programmable gate array").
- **GPU:** Unidad de procesamiento gráfico (en inglés "graphics processing unit").
- **I_{xn}:** Intensidad del pixel *n* según la cámara *x*, pudiendo ser *d* (cámara derecha) , *i* (cámara izquierda) o *s* (cámara virtual).
- **k_x:** Parámetro de distorsión radial número *x*, pudiendo ser *1*, *2* o *3*.
- **P:** Punto en el espacio *(X,Y,Z)* de la reconstrucción de la imagen.
- **P_h:** Punto en el espacio (X_h,Y_h,Z_h,W_h) en coordenadas homogéneas.

- **P_n:** Punto (X_n,Y_n,Z_n) de la construcción en 3D de la imagen rotado y trasladado.
- **p_x:** Parámetro de distorsión tangencial número *x*, pudiendo ser *1* o *2*.
- **q:** Parámetro que indica en que posición se desea generar la vista sintética, siendo 0 la posición del sensor izquierdo y 1 la posición de la sensor derecho.
- **Q:** Matriz obtenida del proceso de calibración usada para transformar las coordenadas de una imagen a coordenadas en el espacio.
- **r:** Distancia, en pixeles, del pixel distorsionado al pixel central de la imagen.
- **R:** Matriz de rotación que se le deben aplicar a los puntos entregados por el sensor imagen-profundidad para que queden en la perspectiva deseada.
- **R_c:** Matriz de rotación que se le debe aplicar al sensor para que quede en la orientación deseada.
- **R_{id}:** Matriz de rotación para transformar el sistema de coordenada de la cámara izquierda al sistema de la cámara derecha.
- **R_x:** Matriz de rotación para llevar el plano del tablero de ajedrez al plano de la imagen de la cámara *x*, pudiendo ser *d* (derecha), *i* (izquierda) o *cu* (cualquiera).
- **s:** Factor escalar usado en el proceso de calibración de una cámara.
- **SGBM:** Clase *"StereoSGBM*" de la librería de OpenCV [2]. La sigla viene de "Semi Global Block Maching".
- **t:** Vector de traslación que se le deben aplicar a los puntos entregados por el sensor imagen-profundidad para que queden en la perspectiva deseada.
- **t_c:** Vector de traslación que se le debe aplicar al sensor para que quede en la ubicación deseada.
- **t_{id}:** Vector de traslación para transformar el sistema de coordenada de la cámara izquierda al sistema de la cámara derecha.
- **t_x:** Vector de traslación para llevar el plano del tablero de ajedrez al plano de la imagen de la cámara *x*, pudiendo ser *d* (derecha), *i* (izquierda) o *cu* (cualquiera).
- **URL:** Localizador de recursos uniforme (en inglés "uniform resource locator").
- **x_d:** Coordenada *x* del pixel distorsionado.
- **y_d:** Coordenada *y* del pixel distorsionado.

Índice de Contenidos

Índice de figuras

Índice de tablas

Capítulo 1

1. Introducción

Cuando se habla de realidad virtual se hace referencia a aquellas tecnologías que permiten al usuario sumergirse en un entorno virtual. Hoy en día dichas tecnologías están empezando a ser aprovechadas por distintas industrias, por ejemplo, en la industria del deporte, está el caso de Liga Nacional de Baloncesto de Estados Unidos, la cual junto con Intel lanzaron una aplicación que permite a los usuarios vivir desde sus hogares un partido de baloncesto mediante realidad virtual [3].

Otra industria que ha visto un gran valor en la realidad virtual es la del turismo, en donde gracias a esta tecnología se ha permitido que personas de todo el globo sean capaces de realizar recorridos turísticos sin la necesidad de viajar. En este contexto, como se indica en la noticia *"Turismo virtual, un modelo de negocios que llegó para quedarse"* [4], uno de los proyectos de la actualidad en Latinoamérica es *"Quito 360"*, una iniciativa Ecuatoriana que hoy en día ofrece múltiples tour virtuales por el país.

Observando el ámbito de la salud, en psicología, como bien se cuenta en el artículo *"Realidad virtual en el consultorio: Una revisión de Psious"* [5], la realidad virtual a significado un gran avance en las terapias cognitivo conductual (véase tratarse una fobia, ansiedad o estrés postraumático por decir algunos) ya que permite lograr mejores terapías de exposición al sumergir al paciente en un medio virtual controlado, quitando de la fórmula la capacidad imaginativa del paciente (lo que se llama terapía de exposición imaginaria).

Los casos anteriores no son únicos y se pueden seguir dando ejemplos de cómo la realidad virtual se ha ido haciendo un lugar en las diferentes industrias. Por lo mismo es que surge un especial interés en ir desarrollando y puliendo la realidad virtual con el fin de que el usuario tenga la mejor experiencia posible.

Una limitante que se ha detectado en este tipo de aplicaciones, cuando se trata de un escenario real, es que el usuario suele estar limitado a puntos específicos de la escena y, más allá de mirar a su alrededor, no tiene una libre movilidad en el escenario. Por lo mismo con este proyecto se busca que a partir de los datos entregados por puntos de mediciones finitos y que se encuentren fijos, poder generar una vista sintética de una nueva perspectiva del escenario como si hubiese sido capturada por una cámara y que dicha posición pueda ser manipulada por el usuario. Lograr este objetivo permitiría que en futuros trabajos se puedan desarrollar aplicaciones de realidad virtual más complejas en la que el usuario pueda desenvolverse mejor con un entorno real.

1.1. Objetivos y requerimientos

Es importante recalcar que en este proyecto no se busca crear una aplicación de realidad virtual, en su lugar se busca establecer una base que permita desarrollar aplicaciones de realidad virtual en la que el usuario sea capaz de moverse libremente en su entorno. Por lo mismo y dadas la cantidad de recursos disponibles, junto con el tiempo de desarrollo, se establecen las siguientes limitaciones para el proyecto:

- Los puntos de mediciones se limitarán a dos y la cámara virtual no se posicionara en cualquier punto, sino que estará limitada a estar en algún punto intermedio entre los puntos de medición.
- La vista sintética será visualizada en un monitor.

Teniendo en cuenta las limitaciones del proyecto, los objetivos a cumplir son:

1. Sintetizar una imagen única y su mapa de profundidad a partir de dos imágenes obtenidas de un par de cámaras adyacentes.
2. Generar la imagen sintética que se obtendría de una cámara virtual ubicada en un punto intermedio entre dos posiciones desde donde se tienen imágenes y mapas de profundidad correspondientes a una misma escena.
3. Desarrollar un sistema de despliegue del video sintético generado por una cámara virtual ubicada en una posición configurable por el usuario entre dos puntos fijos.

Capítulo 2

2. Estudio de técnicas para generación de vista sintética

"View synthesis" es la rama dentro de la visión artificial que busca generar vistas sintéticas a partir de varias imágenes fijas. Los artículos y estudios de esta área se pueden dividir en dos enfoques: vista interpolada y levantamiento 3D.

Vista interpolada son aquellas técnicas en la que mediante cámaras, y quizás algún sensor adicional, se deforman las imágenes para llevar la vista a un punto intermedio entre la cámaras y luego combinar las imágenes, usualmente de forma lineal, para generar la vista de esta nueva perspectiva.

En el caso del levantamiento 3D son aquellas técnicas, en la que usando cámaras u otro tipo de sensor, se recrea la escena en un ambiente 3D y la nueva vista que se desea generar es el resultado de proyectar esa escena en el plano deseado.

Teniendo en cuenta las tecnologías y los algoritmos usados para generar las vistas sintéticas, es importante investigar sobre algunas alternativas tecnológicas de las ya propuestas, tanto de hardware como de software, que pueden aportar a la captura, procesamiento y/o visualización de los datos.

2.1. Trabajos de vista sintética usando *vista interpolada*

Uno de los trabajos más interesantes en lo que respecta a la vista interpolada es el desarrollado por un grupo de investigadores de Microsoft [6], en el cuál usando una configuración de 8 cámaras, con resolución de 1024x768, logran obtener un punto de vista intermedio entra las cámaras. Dicho trabajo también describe claramente los pasos que se deben seguir si se desea obtener una vista sintética, los cuales se pueden resumir en los siguientes puntos:

1. Calibración de las cámaras.
2. Establecer los píxeles correspondientes[1] entre las imágenes.
3. Calcular la disparidad, que corresponde a la diferencia, en píxeles, de los píxeles correspondientes. Como cada pixel se le asigna un valor de disparidad al usar otra imagen como referencia, al conjunto de disparidades se le denomina mapa de disparidad.
4. Generar los mapas de profundidad usando los datos de disparidad.
5. Con la información de los mapas de profundidad, deformar las imágenes para que queden en el punto de vista deseado.
6. Mediante una combinación lineal obtener la imagen final.

Otro trabajo dentro de estas líneas es el mostrado en "*View Interpolation of Multiple Cameras Based on Projective Geometry*" [7], el cuál buscaba sintetizar vistas de una cancha de fútbol. Este artículo se aprovecha del conocimiento previo del escenario para dividirlo y facilitar el trabajo sobre cada elemento que compone la escena. Los componentes de la escena lo dividen en tres partes: El fondo; la cancha de fútbol, la cual tiene una ubicación y geometría conocida; y los jugadores, que corresponden a un elemento dinámico. En este artículo también se hace mención que a partir de tras cámaras se puede generar una vista desde cualquier punto del triángulo que forma las tres cámaras.

Un punto en común de los trabajos anteriores es que ambos necesitaban la información espacial de la escena para producir una vista intermedia. Obtener dicha información puede llegar a ser costoso computacionalmente y bajo este contexto, los trabajos que utilicen sensores de distancias para determinar dicha información pueden llegar a ser de interés, como el mostrado en "*Wide angle virtual view synthesis using two-by-two Kinect V2*" [8]. En este se muestra como usando una matriz de 2x2 de Kinect V2 se consigue generar una vista con la perspectiva de cualquier punto que se encuentre en el rectángulo que forme las Kinect.

[1] Los píxeles correspondiente son aquellos que contienen la proyección de un mismo punto en el espacio.

Si bien el uso de las Kinect soluciona la obtención de la información espacial, se debe tener en cuenta que esta implementación trae consigo una serie de desafíos. El trabajo [8] cuenta que el primer problema se debido a una limitante de hardware, donde fue necesario tener conectado cada Kinect a un computador y además fue necesario usar un software de terceros para sincronizar la hora de cada computador para lograr que la captura de datos fuera simultánea. El segundo problema vino del propio funcionamiento de la Kinect, como este instrumento primero captura la imagen y luego el mapa de profundidad se producía un pequeño desfase que se tuvo que arreglar a la hora de procesar los datos.

2.2. Trabajos de vista sintética usando *levantamiento 3D*

En cuanto al levantamiento 3D uno de los trabajos donde mejor se aprecia este enfoque es el "*Cinematized reality: Cinematographic camera controlling 3D free-viewpoint video*" [9]. Este trabajo, en el contexto de cinematografía, realizan una construcción de la escena en 3D para obtener otros ángulos de cámara y poder capturar en video eventos inesperados durante el rodaje. Para ello usan el método de "shape-from-silhouette" para la reconstrucción en 3D y se apoyan de la información que la escena, la ubicación de los actores y otros objetos de interés, para obtener mejores resultados.

2.3. Herramientas de hardware y software

En esta sección se verán algunas herramientas, tanto de hardware como software, que pueden aportar a la hora de diseñar la solución final.

Hasta ahora se ha visto que para obtener la información espacial de la escena se utilizan cámaras y sensores de profundidad. Si bien las cámaras Kinect se han difundido, existen más sensores que pueden cumplir está función. Un ejemplo de esto es el sensor LiDAR como se muestra en el trabajo "*A Two-step Method for Extrinsic Calibration between a Sparse 3D LiDAR and a Thermal Camera*" [10]. En dicho trabajo, y a pesar del nombre, los autores realizan una calibración entre el LiDAR y una cámara a color para luego calibrar la cámara a color con una térmica.

Otra herramienta de hardware con la que se pueden contar es la familia de las Raspberry Pi, puesto que estas son compatibles con varios tipo de cámaras y permite el procesamiento de imágenes como se puede observar en el video "*Depth Mapping with Arducam Stereo Camera and Raspberry Pi*" [11], en el cual se muestra cómo se calculan los mapas de profundidad usando una cámara estéreo.

En cuanto a las herramientas de software que pueden ayudar a construir la solución final, la principal es la librería OpenCV [12]. Esta librería cuenta con una gran variedad de recursos para procesamiento de imágenes, por ejemplo implementa métodos para calcular el mapa de profundidad. Entre los lenguajes que pueden manejar dicha librería se encuentran C++, Python y Java.

Capítulo 3

3. Diseño de la solución

Con el fin de desarrollar un sistema que sea capaz de generar un video de posición configurable, se ha optado por dividir el sistema en tres subsistemas según las tareas que se deben realizar.

El primer subsistema consiste en un sensor que debe ser capaz de capturar una imagen del escenario y entregar dicha imagen con un mapa de profundidad. A este sensor, de aquí en adelante, será llamado como sensor de imagen-profundidad.

El segundo subsistema es el encargado, a partir de los datos entregados por el sensor de imagen-profundidad, de generar una vista intermedia entre las imágenes entregadas. A este subsistema se le conocerá como *generador de vista sintética*.

El tercer subsistema es el encargado de ser el intermediario entre el sistema y el usuario, en otras palabras, es la *interfaz de usuario*. En la figura 3.1 se visualiza como interaccionan los tres subsistemas para formar la solución final.

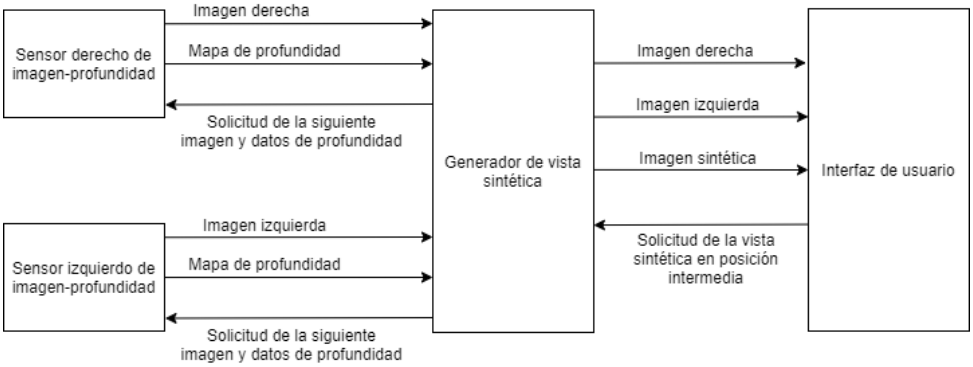


Figura 3.1: Diseño general de la solución

En las siguientes secciones se muestra el diseño de cada subsistema y los equipos a utilizar para que en posteriores capítulos ver la implementación de estos.

3.1. Diseño del sensor de imagen-profundidad

El sensor a diseñar debe ser capaz de capturar una imagen de la escena y entregar dicha imagen con su respectivo mapa de profundidad. Para que el sensor pueda cumplir con su función se propone una configuración como la mostrada en la figura 3.2, en la que se tienen un par de cámaras paralelas que envían las imágenes a un computador donde se realiza el procesamiento de estas para generar el mapa de profundidad. Es importante mencionar que dependiendo de las cámaras a usar puede ser necesario de contar con otro dispositivo que funcione de conector entre estas y el computador.

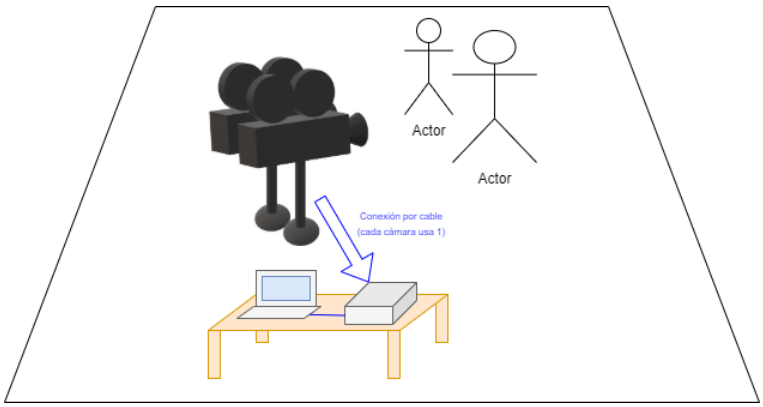


Figura 3.2: Configuración del sensor de imagen-profundidad

Antes de continuar, es importante aclarar que, de aquí en adelante, los ejes x e y de una imagen son como se muestran en la figura 3.3, siendo el punto de origen de coordenadas el pixel izquierdo superior.

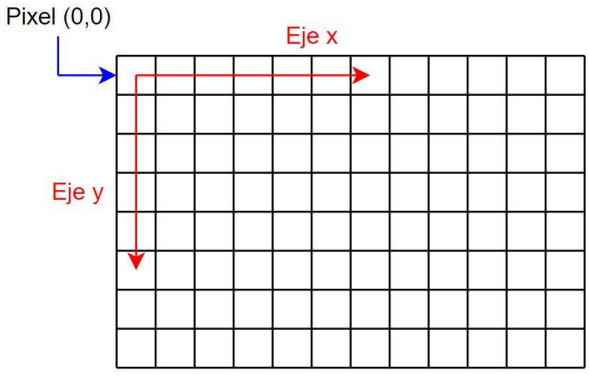


Figura 3.3: Sistema de coordenadas en la imagen

Una vez que las imágenes son recibidas por el computador, estas deben pasar por una fase de preparación para poder generar el mapa de profundidad. Esta fase de preparación conlleva dos tareas: la primera consiste en eliminar la distorsión de las imágenes; la segunda tarea es rectificar las imágenes. Este último proceso consiste en manipular las imágenes de tal forma que queden como si hubiesen sido tomadas por dos cámaras perfectamente paralelas y que se encuentren a la misma altura, lo que se traduce en que las imágenes estén alineadas en el eje y.

Habiendo concluido la fase de preparación, se procede a generar el mapa de profundidad. Los pasos que se deben seguir para cumplir este objetivo son:

- Calcular los valores de disparidad entre las dos imágenes, la cual consiste en la diferencia, en pixeles, entre dos pixel que contengan la proyección del mismo punto del espacio. Para este fin se utilizará un algoritmo de coincidencia de bloque, particularmente, los implementados por las clases de la librería OpenCV: "*StereoBM"* [1], la cual de aquí en adelante se le abreviara con las siglas BM de "Block Maching", y *"StereoSGBM"* [2], la cual de aquí en adelante será SGBM de "Semi Global Block Maching". En la sección 4.3 se explica con más detalle la coincidencia de bloque y la deferencia entre BM y SGBM.
- Filtrar los mapas de disparidad para generar un mapa más suavizado.
- Calcular las profundidades de los pixeles de la imagen. Como la configuración de las cámaras es la mostrada en la figura 3.4, para calcular las profundidades se usa la ecuación 3.1 (ecuación extraída de la documentación oficial de OpenCV [13]).

$$Z = \frac{f \cdot B}{x - x'} \qquad (3.1)$$

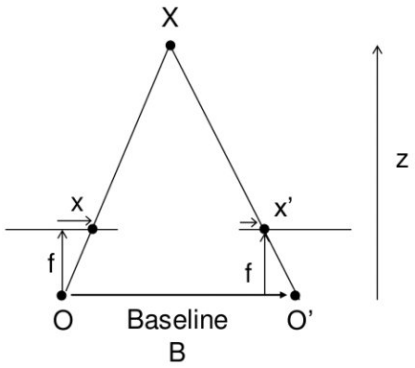


Figura 3.4: Configuración de las cámaras[2]

3.2. Diseño del generador de vista sintética

El generador de vista sintética tiene como objetivo transformar los datos entregados por los sensores de imagen-profundidad en una vista sintética con una nueva perspectiva de la escena. Para cumplir con este objetivo, se optó por seguir el camino de vista interpolada explicado en la sección 2.1.

Antes de entrar en detalle sobre las tareas que se debe realizar este módulo, se debe definir el parámetro *q*. El parámetro *q* es la que indicará en que posición y orientación en que se desea generar la vista sintética, siendo 0 la posición y orientación del sensor izquierdo y 1 la y orientación del sensor derecho. En la figura 3.5 se aprecia mejor el significado del parámetro *q*.

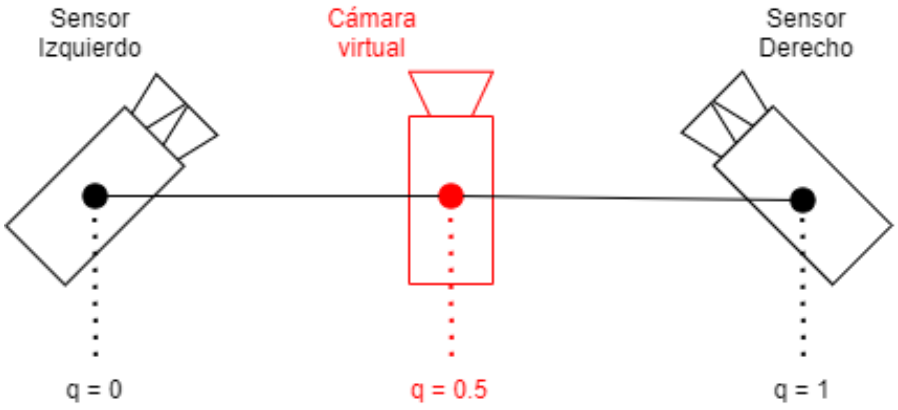


Figura 3.5: Significado del parámetro q

[2] Imagen extraída de *"Depth Map from Stereo Images"* [13] el 14 de Noviembre de 2020

Dicho lo anterior, la primera tarea realizada consiste en reubicar los puntos del mapa de profundidad con el fin de que queden en la perspectiva del punto de vista deseado y luego trasladar su correspondiente pixel de la imagen a la nueva posición. Este trabajo se debe realizar con los datos entregados tanto por el sensor derecho como izquierda, teniendo como resultado dos imágenes de una misma perspectiva. En la sección 5.2 se explica con más detalle cómo se realizan las transformaciones previamente descritas. Se deben resaltar dos casos que se pueden dar a la hora de generar estas imágenes:

- Algunas zonas quedarán sin puntos del mapa de profundidad y por consecuencia tampoco habrán información de que color rellenar la zona. Para estos casos se opta por rellenar con negro.
- El cambio de perspectiva puede provocar que en algunos sectores le corresponda más de un punto del mapa de profundidad. En estos casos se conserva el punto más cercano a la vista que se desea generar y se descartan los otros.

Una vez con las imágenes de la perspectiva deseada, se debe proceder a combinarlas. Para ello se opta por usar la interpolación presentada en el trabajo "*View Interpolation of Multiple Cameras Based on Projective Geometry*" [7]. Reescribiendo la formula ahí presentada, usando el parámetro *q* como peso y llamando I_{xn} la intensidad del pixel *n* según la cámara *x*, el valor de la intensidad del pixel *n* según la cámara virtual es:

$$I_{vn} = q \cdot I_{dn} + (1 - q) \cdot I_{in} \qquad (3.2)$$

La ecuación anterior tiene el detalle que es válida solo si en ese píxel hay información de parte de las dos imágenes que se desean combinar. Si solo una de las imágenes cuenta con esa información, se rellena directamente esta. Para esto se consulta si los píxeles son negros, ya que este color representa la falta de información. Si se da en caso que en ambas imágenes un mismo píxel es negro se puede deber que

las vistas no cuentan con información en ese píxel o que en efecto ese píxel debe ser negro; en ambos casos se resuelve rellenando con negro.

3.3. Diseño de la interfaz de usuario

La interfaz de usuario es la que permite que el usuario manipular el sistema y debe cumplir con las siguientes funcionalidades:

1. Poder indicar la posición donde se desea generar la vista sintética.
2. Muestre las imágenes entregadas por los sensores y la vista sintética.
3. Permita fijar los diferentes parámetros del sistema.
4. Guarde las imágenes y videos generados.

Si bien la interfaz desarrollada se muestra en el capítulo 6, en la figura 3.6 se presenta un boceto de la interfaz implementada:

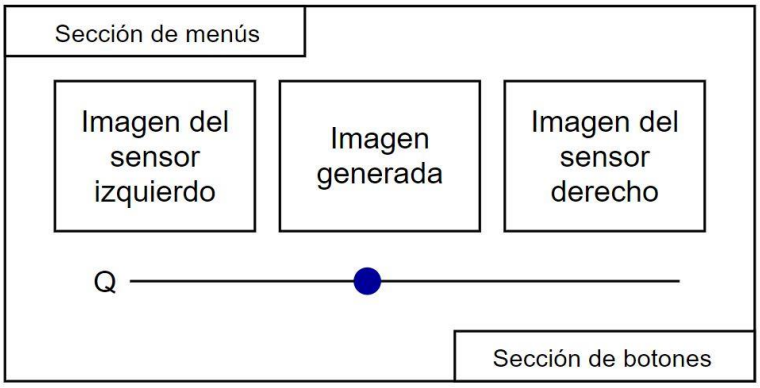


Figura 3.6: Boceto de la interfaz gráfica implementada

3.4. Software y equipos a utilizar

Para realizar el procesamiento, se utiliza un computador que cuenta con el sistema operativo Ubuntu 18.04.5 LTS [14]. Para la programación del sistema se utiliza el lenguaje C++ (La versión C++11) [15] junto con librería la OpenCV en la versión 3.2.0 [16] y para desarrollar la interfaz gráfica se utiliza el entorno de trabajo Qt en la versión 5.9.5 [17].

En cuanto a la captura de las imágenes, se opta por usar un sistema de cámaras de seguridad que cuenta con cuatro cámaras análogicas y un grabador de video digital o

DVR (por sus siglas en ingles “Digital Video Recorder”). Las cámaras van conectadas al DVR, el cual transforma las imágenes a digital con una resolución de 1280x720 píxeles y las envía al computador a travéz de una conexión de red. Para poder manipular el DVR, este último también necesita tener conectado un ratón de computadora y un monitor. En la figura 3.7 se muestran las cámaras, las cuales a pesar de ser de diferente modelo entregan la misma calidad de imagen; en la figura 3.8 se muestra el DVR y en la figura 3.9 se muestra un diagrama con la conexiones, en la que se omitieron las tomas de energía puesto que todos los equipos van conectados a la red de distribución de energía eléctrica.

(a) **(b)**

Figura 3.7: (a) Par de cámaras 1; (b) par de cámaras 2

(a) **(b)**

Figura 3.8: (a) DVR en vista frontal; (b) DVR en vista trasera

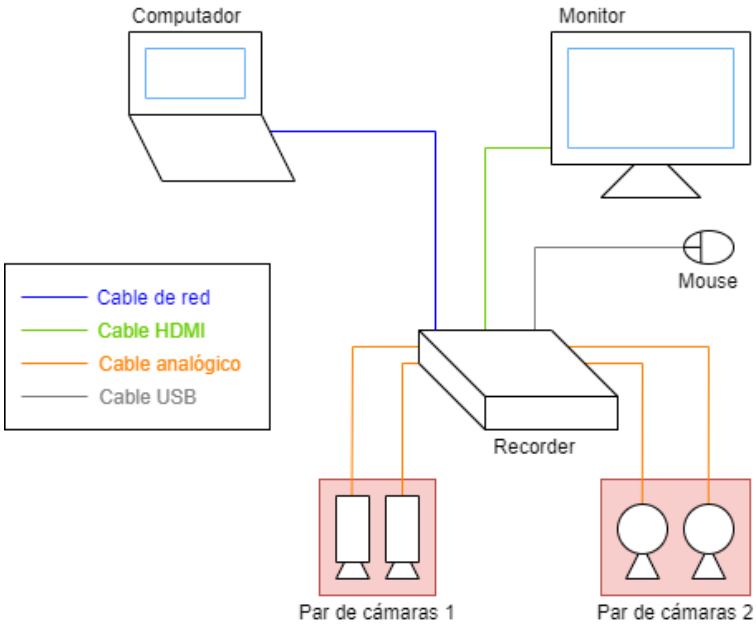


Figura 3.9: Diagrama de conexiones

Capítulo 4

4. Sensor de imagen-profundidad

A lo largo de este capítulo se explica la implementación del sensor de imagen-profundidad, desde la captura de las imágenes hasta la generación de los mapas de profundidad.

4.1. Captura de imágenes

La librería OpenCV cuenta con la clase *"VideoCapture"* [18], la cual es capaz de realizar una petición al DVR (y a cualquier cámara IP) siempre y cuando se le entregue el localizador de recursos uniforme (URL) correspondiente. Para el DVR utilizado dicho localizador presenta el siguiente formato:

rtsp://usuario:contraseña@IP/cam/realmonitor?channel=id&subtype=0

Es importante recalcar que el localizador de recursos uniforme no es único para todas las cámaras IP y DVR, si no que varían dependiendo del fabricante y del modelo.

Para evitar problemas de sincronización en la captura, se crea un hilo por cada cámara, los cuales se encargan de obtener de forma continua las imágenes desde el DVR y almacenarlo. Luego si el hilo principal del programa necesita las imágenes, este obtiene una copia de la última imagen que guardo el hilo.

4.2. Calibración de las cámaras

En la figura 4.2 se observa un ejemplo de las imágenes que se obtienen a partir de un par de cámaras que componen el sensor (las cuatros cámaras utilizadas presentan la misma calidad de imagen). Con el fin de obtener la información espacial de las imágenes, estas deben pasar por un proceso de preparación, el cual consiste en eliminar la distorsión introducida por las propias cámaras y rectificarlas para que queden como si hubiesen sido tomadas por dos cámaras perfectamente paralelas y alineadas en el eje y.

(a) **(b)**

Figura 4.1: (a) Imagen capturada por la cámara izquierda del sensor; (b) imagen capturada por la cámara derecha del sensor

Para realizar este proceso primero se deben determinar los parámetros intrínsecos de cada cámara que compone el sensor. Estos parámetros se dividen en dos, los correspondientes a la distorsión y la matriz de la cámara.

Cuando se habla de distorsión, como se indica en el tutorial *"Camera Calibration"* de la documentación oficial de OpenCV [19], esta se divide en dos tipos: la distorsión radial que produce que la imagen se curve cada vez más a medida que nos alejamos del centro de la imagen, en la figura 4.1 se puede observar este efecto; y la distorsión tangencial que se produce cuando el lento no se encuentra de forma paralela al plano donde se proyecta la imagen dentro de la cámara. El tutorial de la documentación oficial de OpenCV [19] modela la distorsión radial de la siguiente forma (donde x_d e y_d son los píxeles distorsionados y r es la distancia, en píxeles, del pixel distorsionado al píxel central de la imagen distorsionada):

$$x_d = x \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \quad (4.1)$$

$$y_d = y \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \quad (4.2)$$

El mismo tutorial [19] modela la distorsión tangencial como:

$$x_d = x + (2 \cdot p_1 \cdot x \cdot y + p_2 \cdot (r^2 + 2 \cdot x^2)) \quad (4.3)$$

$$y_d = y + (2 \cdot p_2 \cdot x \cdot y + p_1 \cdot (r^2 + 2 \cdot y^2)) \quad (4.4)$$

Entonces los parámetros que se deben obtener para poder eliminar la distorsión de la imagen son *k1*, *k2*, *k3*, *p1* y *p2*.

Por otro lado la matriz de la cámara contiene la información de la distancia focal en los ejes x e y, junto con los pixeles centrales en los ejes x e y. Está matriz es de gran importancia puesto que permite traducir un punto en el espacio en un pixel de la imagen.

Para determinar estos parámetros intrínsecos se siguen los pasos indicados en el blog *"Camera Calibration using OpenCV"* [20]. Primero capturan imágenes de un tablero de ajedrez en diferentes posiciones (se suele recomendar al menos 30 fotografías), como se muestra en la figura 4.2 en la que se aprovechó la cercanía de las cámaras que componen el sensor para capturar las imágenes al mismo tiempo, para luego detectar en que pixeles se ubican las esquinas de los cuadros del tablero de ajedrez. Luego con las imágenes, la ubicación de las esquinas de los cuadros y el tamaño del cuadro del tablero de ajedrez se determinan los parámetros intrínsecos usando alguna implementación del método expuesto por Zhengyou Zhang [21]. Dicho método, explicado en breves palabras, busca resolver el siguiente sistema (extraída del trabajo de Zhang [21]):

$$s \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = A \begin{bmatrix} R_{cu} & t_{cu} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (4.5)$$

Donde s es un factor escalar; x e y las coordenadas del pixel (aún hace falta aplicar la distorsión modelada en las ecuaciones 4.1, 4.2, 4.3 y 4.4); A la matriz de la cámara; R_{cu} y t_{cu} la matriz de rotación y el vector de traslación para llevar el plano del tablero de ajedrez al plano de la imagen respectivamente; Z, X e Y las coordenadas en el espacio.

(a) **(b)**

Figura 4.2: (a) Imagen del tablero de ajedrez capturada por la cámara izquierda del sensor; (b) imagen del tablero de ajedrez capturada por la cámara derecha del sensor

Para la detección de las esquinas de los cuadrados y la implementación del método de Zhang se utiliza la rutina expuesta por Sourish Ghosh en su blog *"Camera calibration using C++ and OpenCV"* [22].

Con los parámetros intrínsecos y la función *"undistort()"* de OpenCV [23] es posible eliminar la distorsión, obteniendo imágenes como las mostradas en la figura 4.3.

(a) **(b)**

Figura 4.3: A partir de las imágenes de la figura 4.2: (a) imagen del tablero de ajedrez capturada por la cámara izquierda del sensor sin distorsión; (b) imagen del tablero de ajedrez capturada por la cámara derecha del sensor sin distorsión

Ya habiendo obtenidos los parámetros intrínsecos, se calcular los denominados parámetros extrínsecos. Estos parámetros corresponden a la posición y orientación de una de las cámaras que conforma el sensor con respecto a la otra. Para poder encontrar estos parámetros se debe capturar imágenes de un tablero de ajedrez, que sea visto por las dos cámaras al mismo tiempo como se muestra en la figura 4.2, y luego establecer las esquinas de los cuadros del tablero en cada imagen. Luego teniendo las imágenes del tablero de ajedrez de las dos cámaras, la ubicación de las esquinas de los cuadros y el tamaño de estos, se pueden calcular estos parámetros. Según la documentación oficial de la función *"stereoCalibrate()"* [24] (función de la librería OpenCV que permite calcular los parámetros extrínsecos) el cálculo se realiza de la siguiente forma: dado que se puede computar la posición relativa del plano del tablero de ajedrez con el plano de la imagen usando el sistema de ecuaciones planteado en 4.5 teniendo en cuenta que en esta ocasión estos parámetros intrínsecos son dados), entonces se puede calcular la posición relativa de una cámara con respecto a otra resolviendo el sistema de

ecuaciones planteado por 4.6 y 4.7 (ambas extraídas de la documentación oficial de OpenCV [24]):

$$R_d = R_{id} \cdot R_i \quad (4.6)$$

$$t_d = R_{id} \cdot t_i + t_{id} \quad (4.7)$$

Donde R_i *y* t_i son la matriz de rotación y el vector de traslación para llevar el plano del tablero de ajedrez al plano de la imagen de la cámara izquierda; R_d *y* t_d es lo equivalente a R_i *y* t_i pero para la cámara derecha y finalmente R_{id} y t_{id} corresponden a la matriz de rotación y vector de posición traslación que transforma el sistema de coordenadas de la cámara izquierda al de la cámara derecha.

Una vez con los parámetros intrínsecos de cada cámara y los parámetros extrínsecos se pueden obtener los parámetros de rectificación. La importancia del proceso de rectificación radica en que, al proyectar las imágenes de cada cámara con el fin de que queden en un mismo plano y alineados en el eje y, los pixeles correspondientes (los pixeles que contienen la proyección del mismo punto en el espacio) entre las dos imágenes tengan la misma coordenada y, lo que reduce significativamente el espacio de búsqueda de dichos pixeles de una plano (la imagen) a una línea.

Para el cálculo de los parámetros de rectificación, OpenCV tiene la función *"stereoRectify()"* [25], la cual debe entregar las siguientes cinco matrices: por cada cámara entrega una matriz de rotación que rota la imagen entregada por cada cámara para que queden en el plano en común (en el capítulo *"Stereopsis"* del libro *"Introductory Techniques for 3-D Computer Vision"* [26] se detalla cómo obtener estás matrices a partir de los parámetros extrínsecos); también por cada cámara entrega una matriz con los nuevos parámetros intrínsecos que le corresponden a las imágenes rotadas y una matriz de mapeo de disparidad profundidad. Está ultima se verá con más detalle en la siguiente sección. Antes de seguir se debe indicar que está función tiene por defecto la opción que el pixel central de que las imágenes rectificadas tengan el mismo valor de la coordenada *x*.

Para el cálculo de los parámetros extrínsecos (incluyendo la identificación de las esquinas del tablero de ajedrez) y los parámetros de rectificación se utiliza la rutina expuesta por Sourish Ghosh en su blog *"Stereo calibration using C++ and OpenCV"* [27], dejando la opción por defecto que provoca que las imágenes rectificadas tengan la misma coordenada del pixel central.

Una vez con los parámetros intrínsecos, extrínsecos y de rectificación, se usa la función *"initUndistortRectifyMap()"* de OpenCV [28] para generar dos matrices de mapeo con las que luego se pueda llevar las imágenes originales a un par de imágenes rectificadas mediante la función *"remap()"* de OpenCV [29]. En la figura 4.4 se observa el par rectificado de las imágenes presentadas en la figura 4.2.

(a) **(b)**

Figura 4.4: A partir de las imágenes de la figura 4.2: (a) imagen del tablero de ajedrez capturada por la cámara izquierda del sensor rectificada; (b) imagen del tablero de ajedrez capturada por la cámara derecha del sensor rectificada

Si bien las imágenes de la figura 4.4 ya se pueden usar, se puede apreciar grandes áreas negras y áreas muy alteradas, por ejemplo la esquina inferior izquierda de la figura 4.4 (b). Con el fin de obtener mejores imágenes, una vez obtenidos los parámetros intrínsecos se procede a eliminar la distorsión de todas las imágenes para simular que las imágenes fueron tomadas por cámaras perfectas, que no introducen distorsión. Luego sobre las imágenes sin distorsión se calculan las parámetros intrínsecos, extrínsecos y de rectificación. Si bien este proceso agrega carga computacional, ya que a las imágenes obtenidas se le debe primero eliminar la distorsión para luego transformarlas en un par rectificadas, se obtienen mejores imágenes como se ve en la figura 4.5.

(a) **(b)**

Figura 4.5: A partir de las imágenes de la figura 4.2: (a) imagen del tablero de ajedrez capturada por la cámara izquierda del sensor rectificada con el proceso modificado; (b) imagen del tablero de ajedrez capturada por la cámara derecha del sensor rectificada con el proceso modificado

4.3. Generación del mapa de profundidad

Como se ha mencionado en la sección 3.1, para generar un mapa de profundidad primero se debe calcular un mapa de disparidad. Para la generación de los mapas de disparidad se optó por usar las clases de BM [1] y SGBM [2] que proporciona la librería de OpenCV.

La clase BM se le entrega el par de imágenes rectificadas y está devuelve una mapa de disparidad utilizando, según su documentación oficial [1], un algoritmo de "Block matching" para detectar los pixeles correspondientes. Este tipo de algoritmo funcionan de la siguiente manera: se tiene el pixel al que se le quiere detectar la correspondencia en la imagen base y se establece una ventana, por ejemplo un área de 5x5 pixeles siendo el pixel central al que se le quiere detectar su pixel correspondiente, luego se selecciona un pixel del área de búsqueda de la otra imagen, al ser imágenes rectificadas el área de búsqueda son los pixeles que tengan la misma coordenada *y* del pixel al que se le desea detectar correspondencia, y usando una ventana del mismo tamaño se compara los dos bloques de pixeles usando sumas por diferencias absolutas, como indica la respuesta del usuario Jenseb en el foro *"Big different between StereoSGBM and gpu::StereoBM_GPU"* [30], obteniendo una métrica de error, como se ilustra en la figura 4.6. El proceso se repite con todos los pixeles del área de búsqueda y el que presente una menor métrica de error se establece como correspondiente al pixel de la imagen base y restando los valores de la coordenada *x* de ambos pixeles se obtiene

finalmente la disparidad asociado a ese pixel de la imagen base. En el tutorial *"Stereo Vision Tutorial – Part I"* [31] se explica con más detalle el algoritmo de "Block matching".

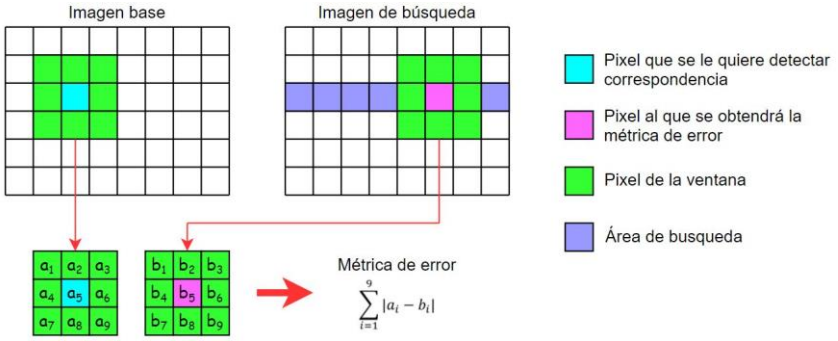


Figura 4.6: Obtención de la métrica de error en un algoritmo de coincidencia de bloque

Por otro lado la clase SGBM también necesita como entrada un par de imágenes rectificadas y retorna un mapa de disparidad. La diferencia de esta clase con la BM radica, como también indica el usuario Jenseb [30], en que se busca que los pixeles vecinos tengan una disparidad parecida. Esto último permite mejores resultados a costo de un mayor tiempo de ejecución.

Antes de seguir, es importante aclarar que el mapa de disparidad obtenido en ambos casos se encuentra alineado con la imagen rectificada de la cámara izquierda del sensor. Esto quiere decir que la disparidad del pixel que se encuentre en la coordenada (550, 100), por decir un ejemplo, de la imagen izquierda rectificada se encontrara en la misma coordenada (550, 100) del mapa de disparidad, como se ilustra en la figura 4.7.

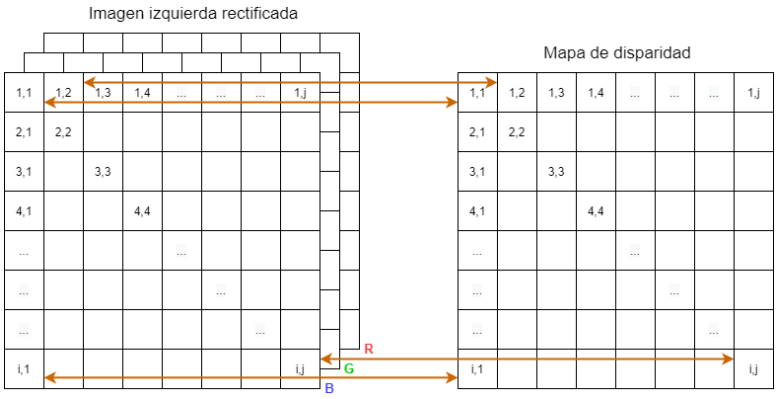


Figura 4.7: Relación entre la imagen izquierda rectificada y el mapa de disparidad

Después de generar los mapas de disparidad se realiza un proceso de filtrado para suavizar los mapas. Para ello se usa la clase *"DisparityWLSFilter"* [32], como se sugiere en el tutorial *"Disparity map post-filtering"* [33]. Este filtro basado en los mínimos cuadrados ponderados tiene la particularidad que permite la conservación bordes y esto lo consigue debido a que no solo se le debe entregar el mapa de disparidad, sino también la imagen con la que se alinea para que sirva de guía. De esta forma en el área que se esté filtrando, si los pixeles vecino presentan poca variación entre sí en la imagen guía, se da importancia a que el mapa de disparidad filtrado en esa área presente poca variación, en cambio si los pixeles vecino presentan gran variación (hecho que se da en los bordes), pierde importancia que el mapa de disparidad filtrado presente poca variación en esa área y se busca que la diferencia entre el mapa de disparidad original y el filtrado sea mínima. Para comprender mejor el fundamento matemático de este filtro, se recomienda leer el trabajo *"Fast Global Image Smoothing Base don Weighted Least Squares"* [34].

Una vez que se tiene el mapa de disparidad, se debe calcular el mapa de profundidad. Para ello se debe usar la matriz de mapeo de disparidad profundidad que se obtuvo del proceso de rectificación como se indicó en el sección 4.2. Dicha matriz es la siguiente:

$$Q = \begin{pmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/B & (c_x - c_x')/B \end{pmatrix} \qquad (4.5)$$

Donde c_x y c_y corresponden a las coordenadas x e y del pixel central de la imagen rectificada izquierda (en la figura 3.3 se muestran los ejes de la imagen); c_x' la coordenada del eje x del pixel central de la imagen rectificada derecha; f la distancia focal; B la distancia entre las dos cámaras[3].

Para encontrar la ubicación en el espacio de un pixel en la ubicación *(x,y)* de la imagen con un valor de disparidad asociado de *d(x,y)*, primero se debe obtener el punto

[3] A pesar que en la matriz de mapeo de disparidad profundidad da a entender que -1/B es negativo, al realizar el proceso de rectificación entrega un valor positivo.

en el espacio en coordenadas homogéneas, a dicho punto se nombra como $P_h=(X_h,Y_h,Z_h,W_h)$. Como se indica en la discusión *"Derivation for perspective transformation matrix (Q)"* [35] de un antiguo foro de OpenCV, P_h se encuentra de la siguiente forma:

$$\begin{pmatrix} X_h \\ Y_h \\ Z_h \\ W_h \end{pmatrix} = Q \cdot \begin{pmatrix} x \\ y \\ d(x,y) \\ 1 \end{pmatrix} \quad (4.6)$$

Considerando que, como se indicó en la sección 4.2, las imágenes rectificadas tienen su respectivo pixel central en la misma coordenada *x*, la ecuación 4.6 se resuelve de la siguiente forma:

$$\begin{pmatrix} X_h \\ Y_h \\ Z_h \\ W_h \end{pmatrix} = \begin{pmatrix} x - c_x \\ y - c_y \\ f \\ \frac{-d(x,y)}{B} \end{pmatrix} \quad (4.7)$$

Finalmente, y como se indica en la discusión *"Derivation for perspective transformation matrix (Q)"* [35], las coordenadas *(X,Y,Z)* del pixel son (En particular la ecuación 4.10 es una reescritura de la ecuación 3.1):

$$X = \frac{X_h}{W_h} = -\frac{B}{d(x,y)} \cdot (x - c_x) \quad (4.8)$$

$$Y = \frac{Y_h}{W_h} = -\frac{B}{d(x,y)} \cdot (y - c_y) \quad (4.9)$$

$$Z = \frac{Z_h}{W_h} = -\frac{B \cdot f}{d(x,y)} \quad (4.10)$$

Notar que las coordenadas *X* e *Y* se pueden escribir en función de la coordenada *Z*, o sea de la profundidad, obteniendo las siguientes formulas:

$$X = \frac{Z}{f} \cdot (x - c_x) \quad (4.11)$$

$$Y = \frac{Z}{f} \cdot (y - c_y) \quad (4.12)$$

Para realizar la conversión de pixel de cada a un punto en el espacio se utiliza la función *"reprojectImageTo3D"* [36], la cual recibe el mapa de disparidad junto con la matriz de mapeo de disparidad profundidad y retorna una matriz de 3 canales (uno por cada coordenada) y cada punto está en la misma ubicación de su correspondiente píxel en la imagen izquierda rectificada. La figura 4.8 ilustra mejor este concepto.

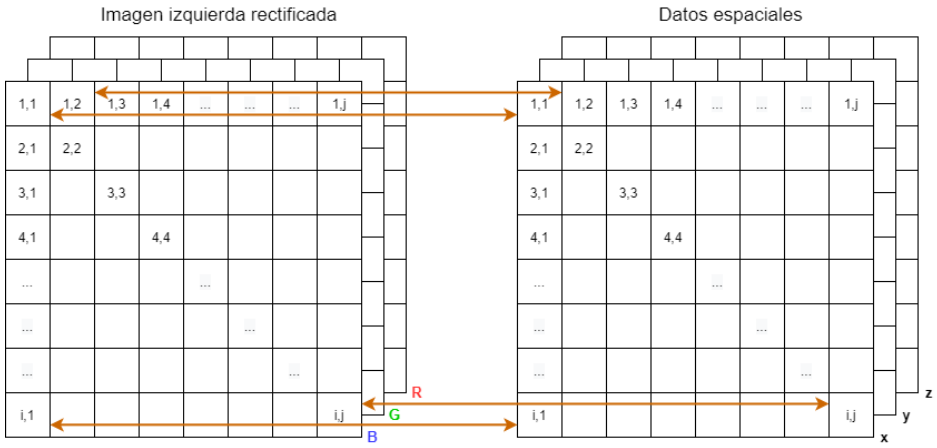
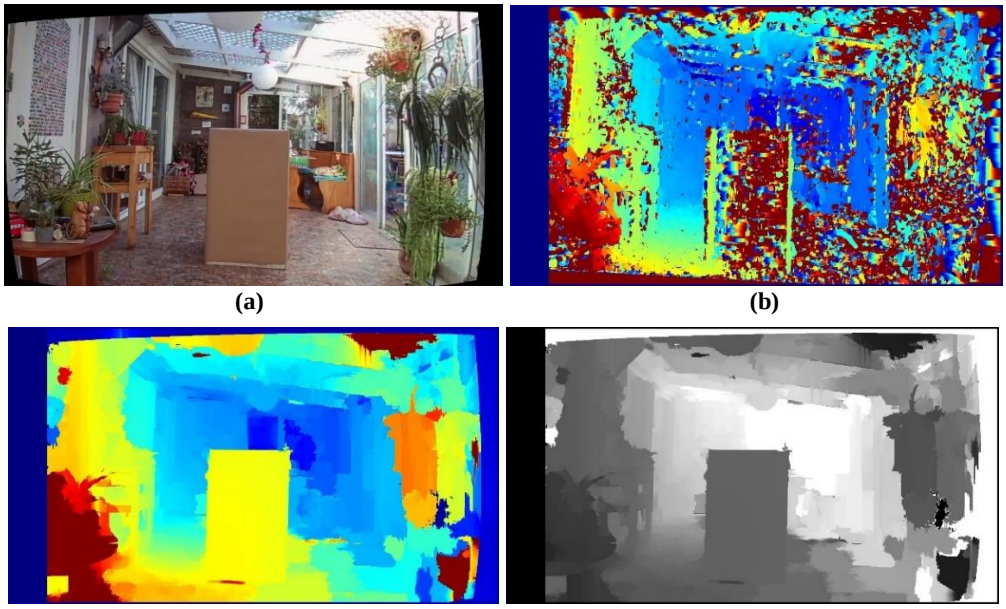


Figura 4.8: Relación entre la matriz de los datos espaciales y la imagen izquierda rectificada

Con lo anterior se tiene que el sensor de imagen-profundidad no solo entrega la profundidad, sino que entrega directamente la construcción en 3D del espacio.

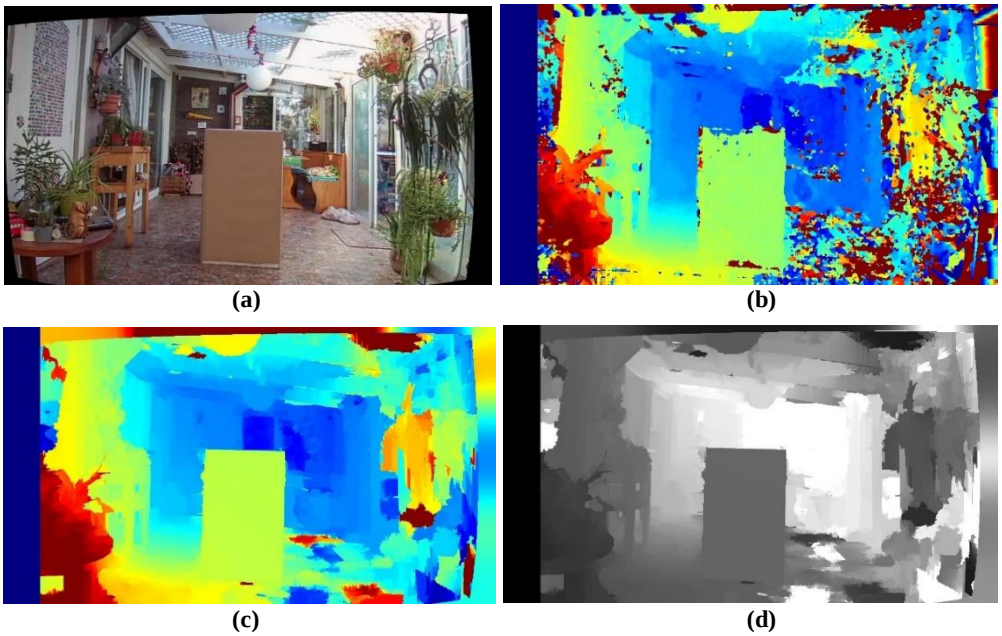
En la figura 4.9 se muestra la imagen que entrega el sensor (que corresponde a la imagen izquierda rectificada); el mapa de disparidad obtenido usando BM donde se coloreo con tonos más rojizo las disparidades mayores y tonos azulados las disparidades menores; el mapa de disparidad después del filtrado usando el mismo código de colores y finalmente el mapa de profundidad de la imagen, en la que se coloreo con tonos negros las zonas con menos profundidad y blanco las que tienen mayor profundidad. En la figura 4.10 se vuelve a mostrar la evolución del proceso de obtención del mapa de profundidad pero esta vez usando SGBM para la detección de disparidades.



(a) **(b)**

(c) **(d)**

Figura 4.9: (a) Imagen entregado por el sensor; (b) mapa de disparidad generado por BM; (c) mapa de disparidad generado por BM filtrado; (d) mapa de profundidad a partir de (c)



(a) **(b)**

(c) **(d)**

Figura 4.10: (a) Imagen entregado por el sensor; (b) mapa de disparidad generado por SGBM; (c) mapa de disparidad generado por SGBM filtrado; (d) mapa de profundidad a partir de (c)

La franja negra que se observa en los mapas de disparidad y de profundidad en las figuras 4.9 y 4.10 se debe a la implementación del algoritmo correspondencia de bloque

de BM y SGBM. Como BM y SGBM buscan correspondencia de pixeles usando como base la imagen que proporciona la cámara izquierda y las imágenes se encuentran rectificadas, es decir que se encuentran en un mismo plano, siempre existirá una granja en el lado izquierdo de la imagen que no será posible detectar correspondencia dado ese espacio no es visible en la imagen entregada por la cámara derecha. Por lo anterior BM y SGBM directamente no buscan correspondencia ni calculan disparidad en una franja del lado izquierdo de la imagen cuyo tamaño corresponde a la disparidad máxima permitida seteada por el usuario.

4.4. Resultados

Para ver cómo afecta la posición de las cámaras a los resultados entregados por el sensor, se hicieron tres pruebas: Se colocaron las cámaras a una distancia de 10 cm entre ellas de forma paralela; luego se aumentó dicha distancia a 30 cm e inclinando muy ligeramente las cámaras (no fue posible dejar las cámaras perfectamente paralelas con esa separación) y finalmente a manteniendo está última separación se procedió a inclinar aún más las cámaras. En la figura 4.11 se aprecia mejor cuando se habla de cámaras paralelas e inclinadas. En los tres casos se calculó el mapa de disparidad mediante SGBM.

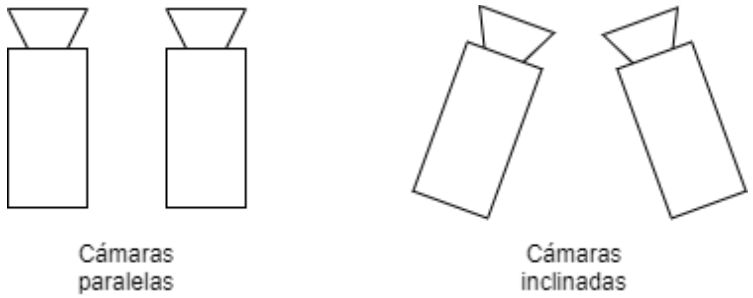


Figura 4.11: Configuración de cámaras paralelas (izquierda) e inclinadas (derecha)

El resultado de la pruebas descrita se puede apreciar en la figura 4.12, donde en la columna izquierda se sitúa la imagen entregada por el sensor y en la columna derecha el mapa de profundidad. En dicha figura se puede observar que a pesar que el algoritmo propuesto es capaz de calcular un mapa de profundidad en los tres casos, dado que los algoritmos de OpenCV requieren que las imágenes pasen por un proceso de

rectificación, el ir separando e inclinar las cámaras no aporta a que el sensor entregue mejores salidas, es más, hay más zonas en la que no se tiene información. Por lo tanto para obtener buenos resultados es recomendable tener las cámaras lo más juntas y paralelas posible.

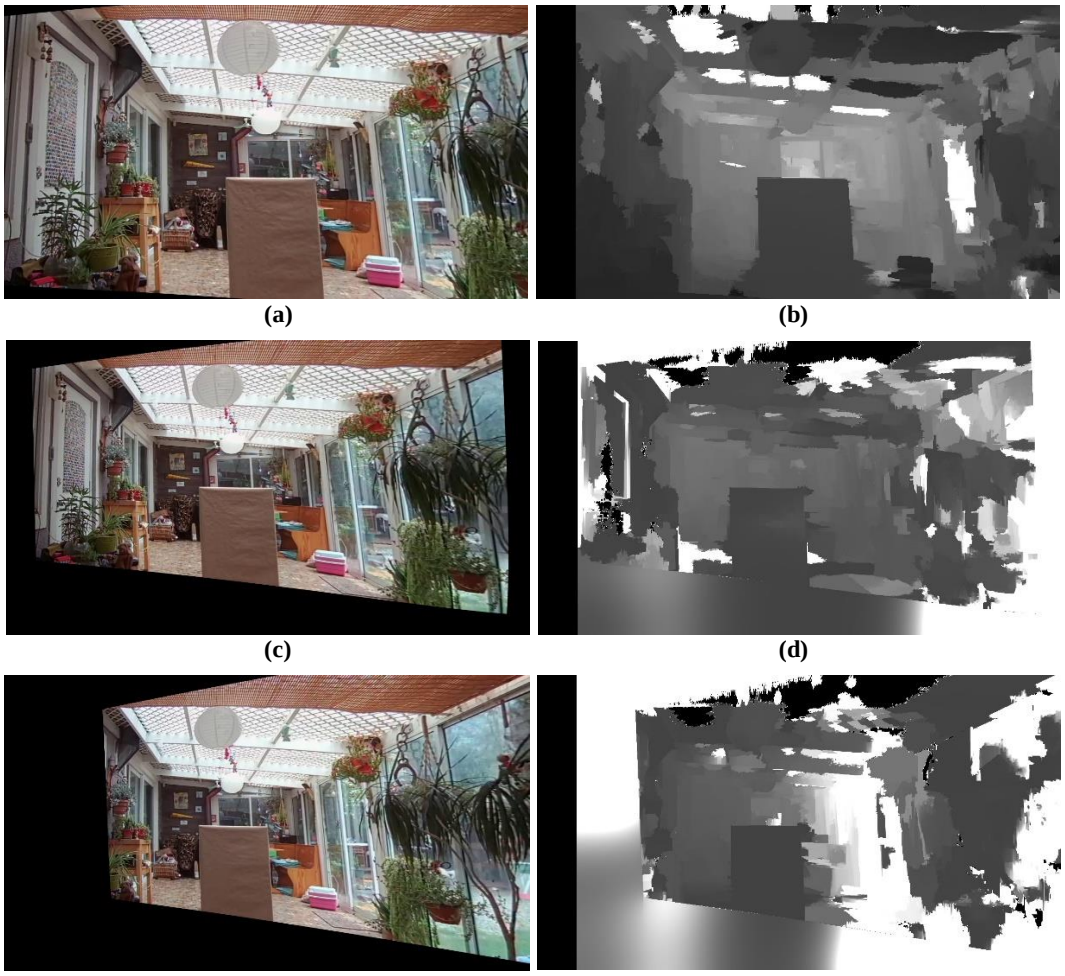


Figura 4.12: (a) Imagen entregada por el sensor con cámaras en paralelo y separadas por 10 cm; (b) mapa de profundidad de (a) usando SGBM; (c) imagen entregada por el sensor con cámaras ligeramente inclinadas y separadas por 30 cm; (d) mapa de profundidad de (c) usando SGBM; (e) imagen entregada por el sensor con cámaras inclinadas y separadas por 30 cm; (f) mapa de profundidad de (e) usando SGBM

Para apreciar los resultados entregados por el sensor de imagen-profundidad se posiciona una caja a una distancia conocida del par de cámaras paralelas y con las imágenes resultantes se calcula el mapa de profundidad. Los mapas de profundidad se almacenaron en un archivo en formato “csv” con el fin poder abrirlos como imágenes

en Matlab [37], ya que este tiene la opción de ver el valor de un píxel con solo seleccionarlo. En la figura 4.13 en la columna izquierda se sitúa la imagen entregada por el sensor y en la columna derecha el mapa de profundidad, habiendo calculado la disparidad con BM. En la figura 4.14 se muestra los resultados habiendo usado SGBM para calcular la disparidad.

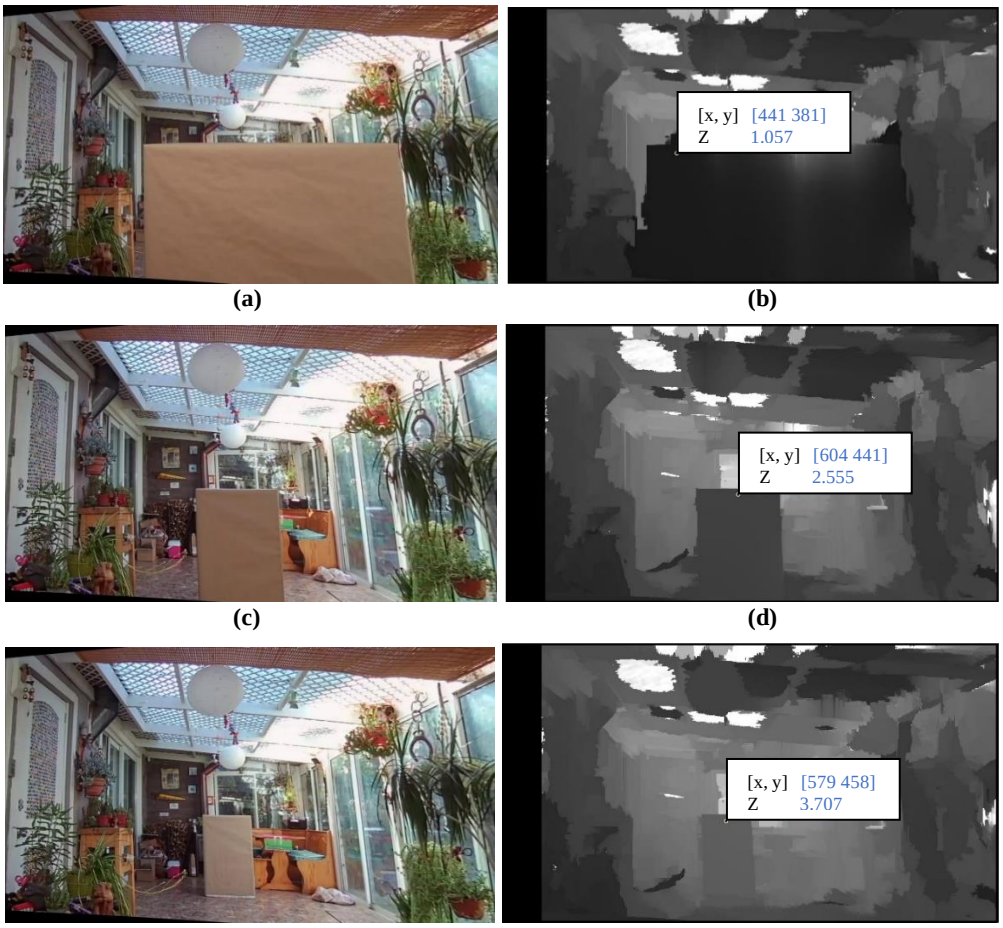


Figura 4.13: (a) Imagen entregada por el sensor con la caja ubicada a 1 m del sensor; (b) mapa de profundidad de (a) usando BM; (c) imagen entregada por el sensor con la caja ubicada a 3 m del sensor; (d) mapa de profundidad de (c) usando BM; (e) imagen entregada por el sensor con la caja ubicada a 5 m del sensor; (f) mapa de profundidad de (e) usando BM

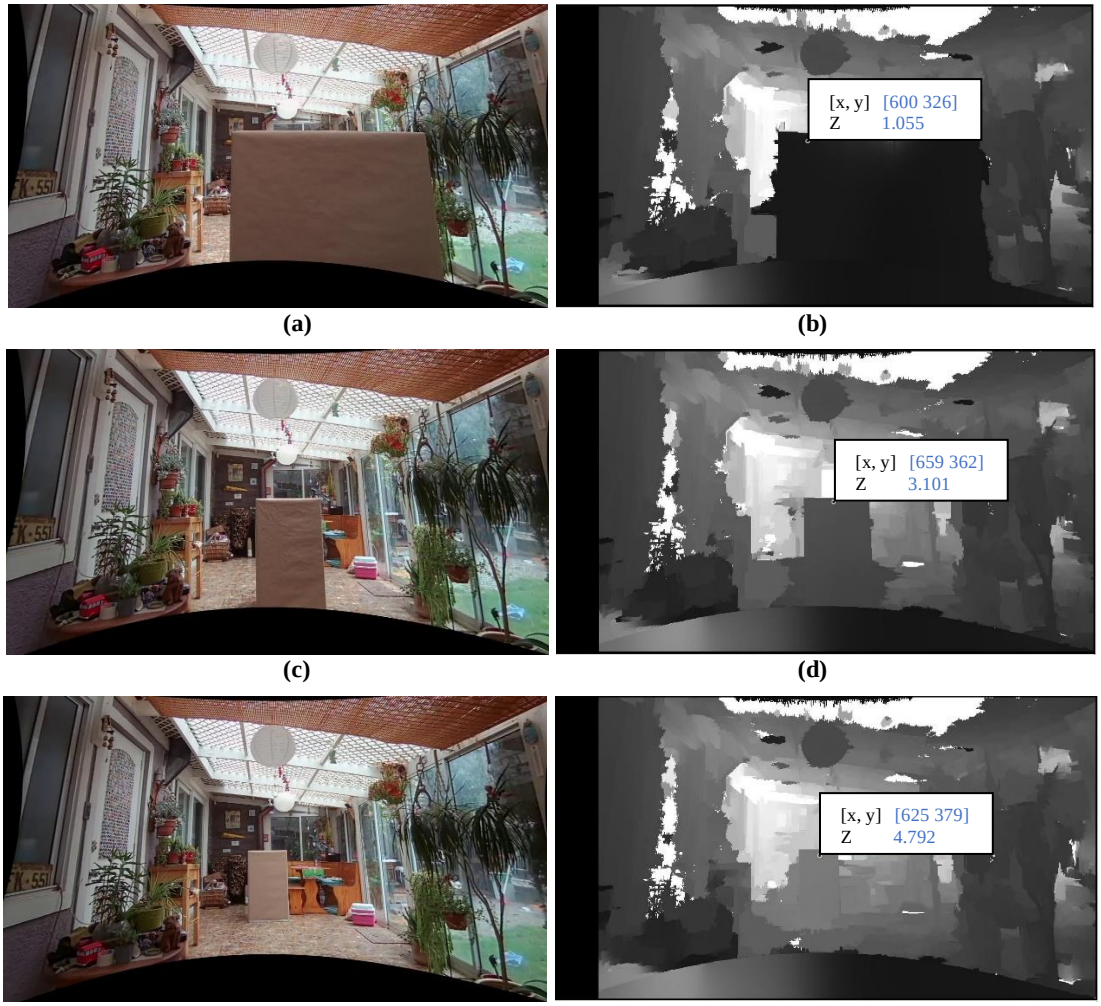
(a) **(b)**

(c) **(d)**

(e) **(f)**

Figura 4.14: (a) Imagen entregada por el sensor con la caja ubicada a 1 m del sensor; (b) mapa de profundidad de (a) usando SGBM; (c) imagen entregada por el sensor con la caja ubicada a 3 m del sensor; (d) mapa de profundidad de (c) usando SGBM; (e) imagen entregada por el sensor con la caja ubicada a 5 m del sensor; (f) mapa de profundidad de (e) usando SGBM

Se puede apreciar que existe un error significativo a medida que las caja se aleja del par de cámaras. Esto se debe a la decisión de eliminar la distorsión antes de realizar el proceso de calibración de las cámaras. Si se realiza el proceso de calibración sin ese paso y se calculan los mapas de profundidad, se obtienen resultados más cercanos a la realidad, como se muestra en la figuras 4.15 y 4.16.

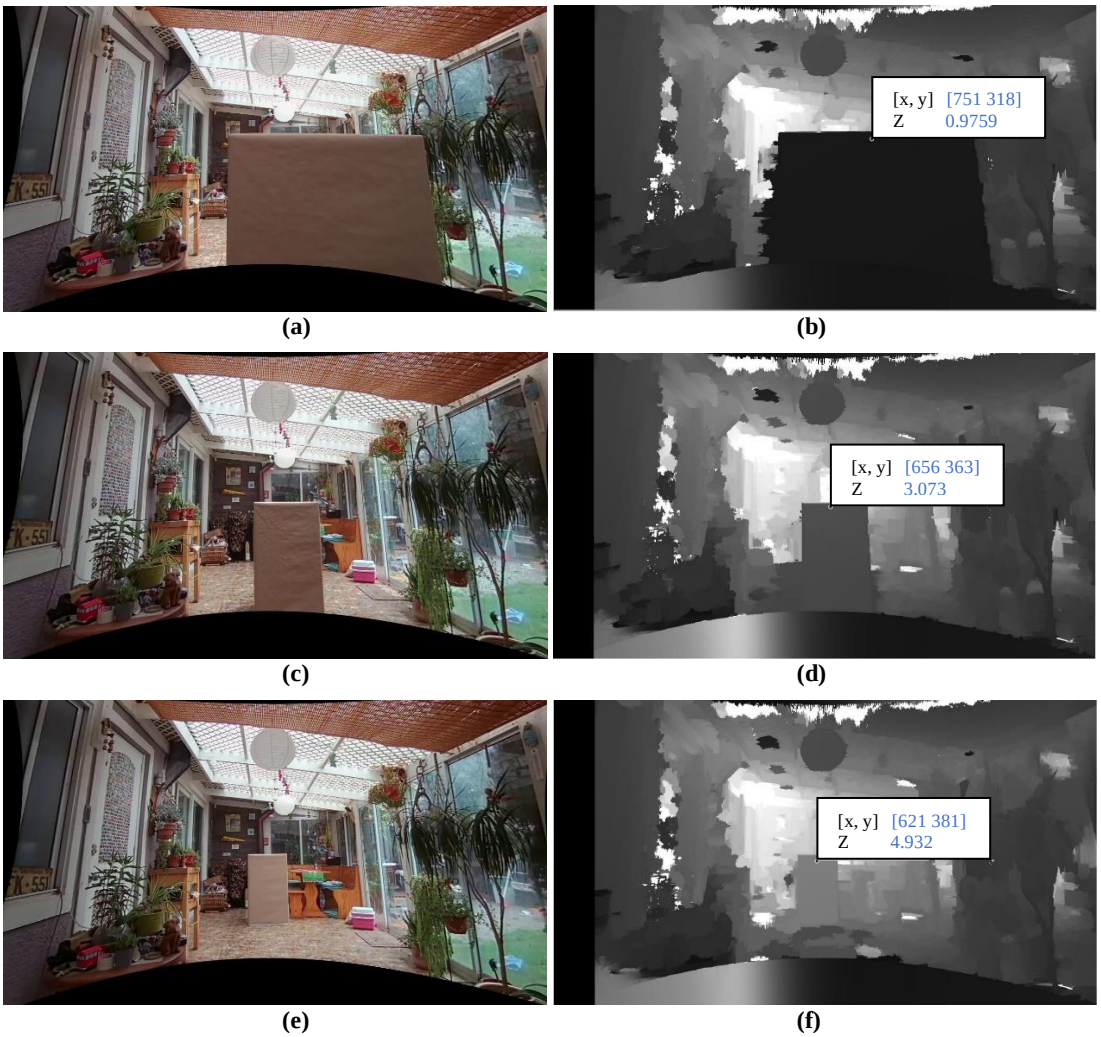


(a) **(b)**

(c) **(d)**

(e) **(f)**

Figura 4.15: (a) Imagen entregada por el sensor con la caja ubicada a 1 m del sensor, sin eliminación de distorsión previa; (b) mapa de profundidad de (a) usando BM; (c) imagen entregada por el sensor con la caja ubicada a 3 m del sensor, sin eliminación de distorsión previa; (d) mapa de profundidad de (c) usando BM; (e) imagen entregada por el sensor con la caja ubicada a 5 m del sensor, sin eliminación de distorsión previa; (f) mapa de profundidad de (e) usando BM



(a) **(b)**

(c) **(d)**

(e) **(f)**

Figura 4.16: (a) Imagen entregada por el sensor con la caja ubicada a 1 m del sensor, sin eliminación de distorsión previa; (b) mapa de profundidad de (a) usando SGBM; (c) imagen entregada por el sensor con la caja ubicada a 3 m del sensor, sin eliminación de distorsión previa; (d) mapa de profundidad de (c) usando SGBM; (e) imagen entregada por el sensor con la caja ubicada a 5 m del sensor, sin eliminación de distorsión previa; (f) mapa de profundidad de (e) usando SGBM

Finalmente mencionar que los datos espaciales que entrega el sensor no son perfectos, como se muestra en la figura 4.17 donde se encerraron con rojos algunas imperfecciones más visibles de los mapas de profundidad. Estás imperfecciones finalmente se traducirán en que al momento de generar las imágenes sintéticas estas presenten menor calidad.

(a)

(b)

(c)

Figura 4.17: (a) Imagen entregada por el sensor de imagen-profundidad; (b) mapa de profundidad de (a) usando BM; (c) mapa de profundidad de (a) usando SGBM. En (b) y (c) se encerraron con rojo algunos de los errores más visibles.

Capítulo 5

5. Generación de vista sintética

A lo largo de este capítulo se verá en detalle la generación de la vista sintética a partir de los datos entregados por los sensores de imagen-profundidad. Para ello se ha dividido este capítulo en cinco secciones. La primera sección, *"Calibración de los sistemas de capturas"*, consiste en la calibración de los equipos con el fin de obtener la posición y orientación relativa a un sensor de imagen-profundidad con respecto al otro.

La segunda sección, *"Transformación de la imagen"*, se centra de cómo, a partir de los datos entregados por lo sensores de imagen-profundidad, mapear la imagen para obtener una nueva perspectiva de la escena desde un punto intermedio entre los sensores.

La tercera sección, *"Combinación de las imágenes"*, aborda de cómo mezclar las imágenes sintéticas generadas a partir de los dos sensores para obtener una sola vista sintética de esa nueva perspectiva de la escena.

La cuarta sección, *"Generador de imágenes con JavaFX"*, se presenta un programa que se utilizó con fines de obtener imágenes de pruebas ideales, es decir, imágenes que no se le deba eliminar la distorsión, no se deban rectificar y se conozca las ubicaciones y orientaciones de las cámaras.

Finalmente en la sección de *"Resultados"* se muestran diferentes pruebas con las que ver el comportamiento del generador de vistas.

5.1. Calibración de los sistemas de capturas

El objetivo de este proceso es obtener la posición y orientación relativa de un sensor con respecto al otro, o más específicamente, se necesita la posición y orientación

relativa entre las cámaras izquierda de ambos sensores, pues la imagen que entrega el sensor desarrollado corresponde a la imagen obtenida de esta cámara.

Para cumplir con lo anterior, se repite el proceso descrito en el capítulo cuatro en la sección *"Calibración de las cámaras"*, pero entre las cámaras izquierdas de ambos sensores, pues este proceso entre los datos que entrega este proceso se encuentra un vector de traslación y una matriz de rotación entre las cámaras.

La matriz de rotación obtenida por sí sola no es suficiente ya que no se puede manipular para obtener una rotación intermedia entre los sensores. Para solucionar esto se utiliza la función *"Rodrigues"* de la librería OpenCV [38], a la cual si se le entrega un matriz de rotación esta retorna una matriz con los ángulos de rotación en cada eje. Dicha función también permite hacer la operación inversa.

Cabe mencionar que aparte del método ya mencionado, y siempre y cuando se cuenten con los equipos de medición adecuados, es posible determinar la ubicación y orientación relativa de un sensor con respecto al otro mediante medición directa sin pasar por el proceso de calibración. Para este caso no se contaba con algún equipo que permitiese detectar de forma confiable la diferencia de orientación entre las cámaras, por lo que se usó el método anteriormente descrito.

5.2. Transformación de la imagen

Como se mencionó en la sección 4.3, el sensor de imagen-profundidad entrega las coordenadas *(X,Y,Z)* de cada pixel teniendo como origen el sensor. Ahora se desea ver como esos puntos se proyectan en una nueva imagen si el sensor se trasladara y rotara para ver la escena desde otro punto de vista.

Sea t_c el vector de traslación y R_c la matriz de rotación que ubican y orientan respectivamente al sensor en la ubicación deseada. Para ver como quedarían los datos espaciales de la imagen original en esta nueva perspectiva, se supondrá que es el sensor el que se encuentra estático y es la escena la que se mueve alrededor de este, como se ilustra en la figura 5.1, donde la traslación y la rotación de los puntos en el espacio debe ser el inverso al que se le aplicaría al sensor.

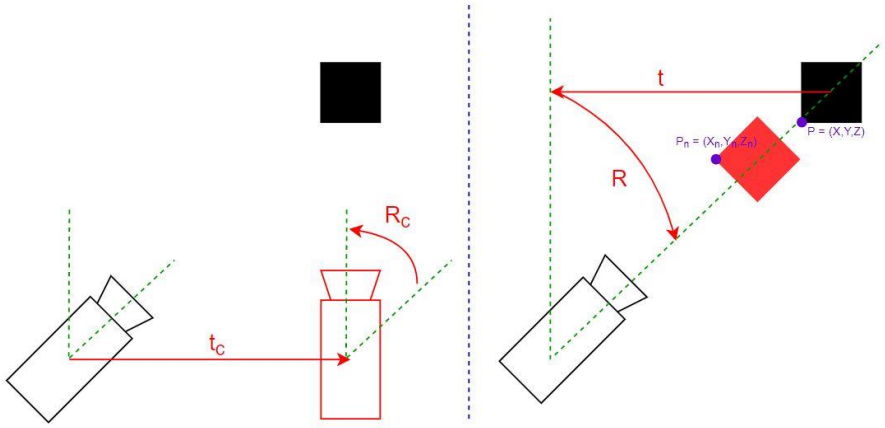


Figura 5.1: A la izquierda de la imagen el sensor se mueve alrededor de la escena y a la derecha de la imagen la escena se mueve alrededor del sensor

Sea *t* el vector de traslación y *R* la matriz de rotación que se le deben aplicar a los puntos entregados por el sensor imagen-profundidad para que queden en la perspectiva deseada. También sea $P=(X,Y,Z)$ un punto cualquiera del conjunto entregado el sensor, que se encuentra asociado a su respectivo pixel (x,y) de la imagen que entrega el sensor y $P_n=(X_n,Y_n,Z_n)$ su ubicación de dicho punto en la nueva perspectiva (la figura 5.1 ilustra un ejemplo). La forma de transformar el punto P al punto P_n es:

$$P_n = R \cdot P + t \qquad (5.1)$$

Para encontrar el pixel (x_n,y_n) que contiene la proyección de punto P_n, se realiza la siguiente conversión, las cuales vienen de las ecuaciones 4.11 y 4.12[4] :

$$x_n = \frac{X_n \cdot f}{Z_n} + c_x \qquad (5.2)$$

$$y_n = -\frac{Y_n \cdot f}{Z_n} + c_y \qquad (5.3)$$

Donde c_x y c_y son la coordenada del eje x e y del pixel central de la imagen y f la distancia focal. El valor de estos parámetros corresponden a los que contiene la matriz de mapeo de disparidad profundidad (ecuación 4.5).

[4] El signo negativo que aparece debido a que la función "reprojectImageTo3D" [36] invierte el eje y, dejando el eje y de la reconstrucción espacial en sentido opuesto al de la imagen (figura 3.3).

Entonces a la imagen que contenga la nueva proyección, se le debe asignar en el pixel (x_n,y_n), siempre y cuando este dentro de los límites de la imagen, el color del pixel (x,y) de la imagen que entrega el sensor.

En la figura 5.2 se puede observar el ejemplo de haber trasladado el sensor 1 metro a la derecha; en la figura 5.3 por haber rotado el sensor en 20° en sentido antihorario y en la figura 5.4 haber trasladado la cámara 1 metro a la derecha y al mismo tiempo haber rotado el sensor en 20° en sentido antihorario.

(a) **(b)**

Figura 5.2: (a) Imagen entregada por el sensor; (b) imagen (a) con traslación en 1 m del sensor a la derecha

(a) **(b)**

Figura 5.3: (a) Imagen entregada por el sensor; (b) imagen (a) con rotación 20° antihorario del sensor

(a) **(b)**

Figura 5.4: (a) Imagen entregada por el sensor; (b) imagen (a) con traslación en 1m de la cámara a la derecha y rotación 20° antihorario del sensor

Sea el vector *a*, el cual contiene los ángulos de rotación que define la matriz de rotación *R* y recordando que el parámetro *q* define en qué punto se desea generar la vista sintética (ver figura 3.5), entonces la transformación que se debe aplicar a los puntos de la reconstrucción espacial del sensor izquierda para llevarlos a la perspectiva de un punto intermedio entre ambos sensores es (modificación de la ecuación 5.1):

$$P_n = Rodriques(q \cdot a) \cdot P + q \cdot t \qquad (5.4)$$

De forma análoga, la transformación que se debe aplicar a los puntos de la reconstrucción espacial del sensor derecho para llevarlos a la perspectiva de un punto intermedio entre ambos sensores es:

$$P_n = Rodriques((q-1) \cdot a) \cdot P + (q-1) \cdot t \qquad (5.5)$$

5.3. Combinación de las imágenes

Una vez con las imágenes generadas en los puntos deseados, se deben combinar en una sola y para ello se implementan dos métodos. El primero consiste en el método explicado en la sección 3.2, el cual consiste en la interpolación presentada en el trabajo *"View Interpolation of Multiple Cameras Based on Projective Geometry"* [7]. A este método se le llamará de aquí en adelante como el método de interpolación.

Si bien el método anterior es efectivo, es necesario que la generación de los puntos de vistas sean precisas, pues de lo contrario la imagen sintética empieza a perder calidad pudiendo llegar al grado de tener una vista incomprensible. Por lo anterior se puede optar por implementar una versión simplificada de la técnica presentada en *"Wide angle virtual view synthesis using two-by-two Kinect V2"* [8], la que consistirá en ir rellenando la imagen sintética con la información del sensor más cercana y si esta no posee tal información, se rellena con la información que aporta el otro sensor. Para detectar si la vista tiene la información, en el momento de realizar el cambio de perspectiva de una imagen, se almacena el mapa de profundidad de la nueva perspectiva y si ese mapa en un punto en específico es cero, se interpreta como la falta de información. A este método se le llamará de aquí en adelante como el método de reemplazo.

Aparte de la implementación del método de reemplazo para solventar el problema de la precisión de los datos espaciales, también se implementó un sistema de compensación. Este sistema consiste en mover las imágenes sintéticas en forma proporcional al punto donde se genera la vista. En la práctica, si llamamos c_{px} y c_{py} los factores de compensación en el eje x e y respectivamente, se traduce en que la imagen sintética hecha a partir de los datos del sensor izquierdo sufra un corrimiento de $-q$ veces c_{px} en el eje x de la imagen y de $-q$ veces c_{py} en el eje y; y en el caso de la imagen formada por el sensor derecho, en lugar de multiplicar los factores por $-q$, se realiza con $1-q$. Es importante agregar que esta forma de compensación es un intento de arreglar las imágenes sintéticas y mientras se tenga una buena calibración de los equipos es recomendable dejar los valores de estos factores, c_{px} y c_{py}, como cero.

5.4. Generador de imágenes con JavaFX

Dado que el proceso de calibración altera las imágenes, y con el fin de poder probar solo lo relacionado al generador de vistas, es que surgió la necesidad de crear un programa el cual permita generar imágenes ideales con las que se puedan probar el sistema. Para ello se optó por usar el entorno de trabajo de JavaFX [39].

El programa creado para este fin pone uno o más cubos en un espacio y con el teclado manipular la posición de la cámara (con las flechas la cámara se traslada y con las teclas Q y W la cámara gira sobre su propio eje). Finalmente la imagen se obtiene con una captura de pantalla, lo que asegura que todas las imágenes tengan el mismo tamaño. La figura 5.5 muestra un ejemplo de las imágenes que se pueden obtener.

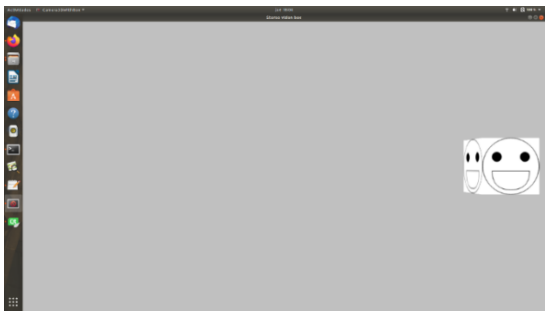
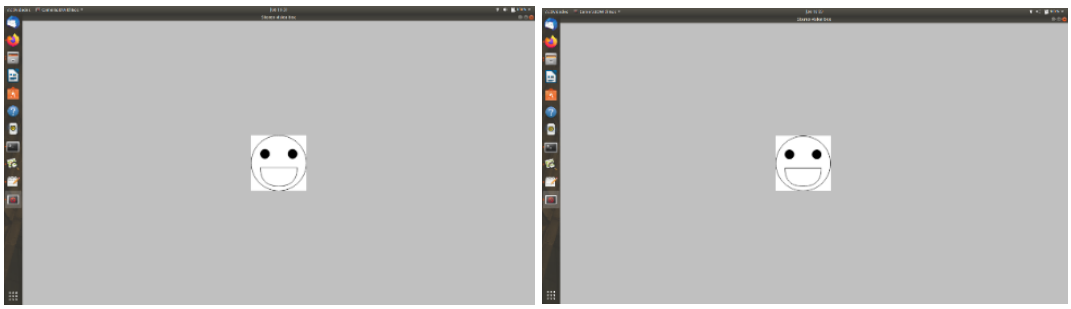


Figura 5.5: Imagen de un cubo en un espacio virtual

Antes de seguir es importante mencionar que la barra de tarea que se observa en la figura 5.5, pues está dentro del área donde BM y SGBM no calculan distorsión. Por otro lado la barra superior no afecta a la vista sintéticas del cubo como se aprecia en la siguiente sección.

Para poder procesar estas imágenes se necesita la matriz de la cámara y la separación entre ellas. Para la matriz de las cámaras, se consideró que todas las cámaras virtuales eran idénticas, con una distancia focal de 1000 (similar a la de las cámaras usadas) y con sus píxeles centrales ubicados en el centro del área gris. Por otro lado se consideró que cada vez que se realizaba una traslación de la cámara está sea de 10 cm y cuando se realizaba una rotación sea de 10°, de esta forma resulta sencillo conocer las ubicaciones de las cámaras en todo momento. Con lo anterior dicho, se optó que el par de cámaras virtuales que conformase el sensor de imagen-profundidad siempre estuviese separado por 10 cm, obteniendo imágenes como las mostradas en la figura 5.6.



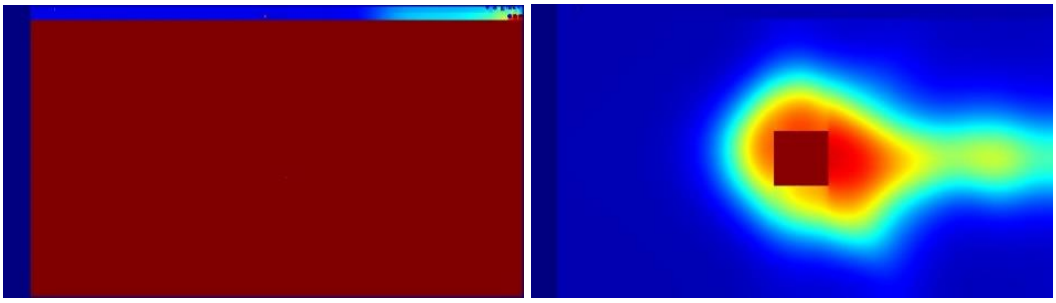
(a) **(b)**

Figura 5.6: Ejemplo de las imágenes obtenidas por las cámaras virtuales. (a) Imagen que entrega la cámara izquierda; (b) imagen que entrega la cámara derecha

Para calcular los mapas de disparidad se usó SGBM. Esta decisión se debe a, que si bien lo importante de las imágenes son los cubos, el fondo unicolor provoca problemas con el algoritmo. En la figura 5.7 se muestra el mapa de disparidad usando BM y SGBM usando como base las imágenes de la figura 5.6.

Esta prueba muestra como los algoritmos utilizados podrían llegar a fallar si en un escenario real se presentan áreas monótonas. Aun así, como se aprecia en la figura 5.7, el mapa de disparidad generado con el algoritmo de SGBM permite identificar el cubo,

por lo que se usará dicho algoritmo para realizar pruebas con las imágenes que genera este programa.

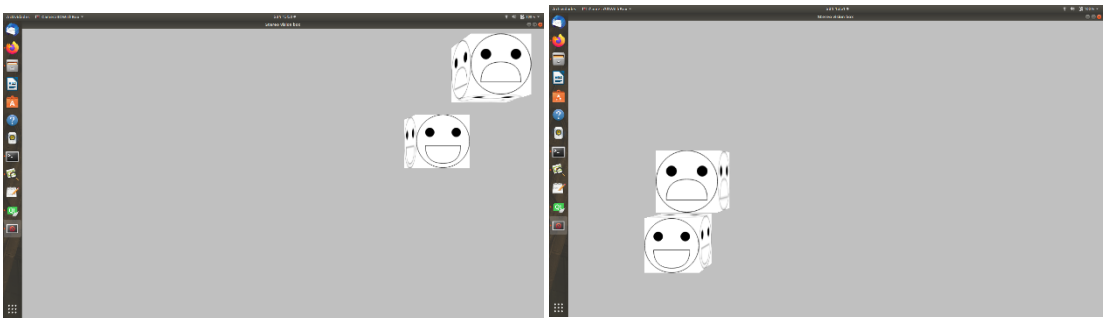


(a) **(b)**

Figura 5.7: A partir de las imágenes de la figura 5.6: (a) mapa de disparidad usando BM; (b) mapa de disparidad usando SGBM

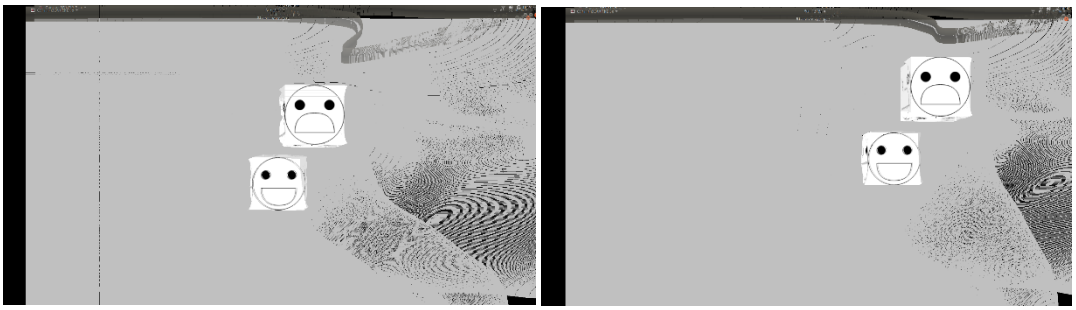
5.5. Resultados

Con el fin de probar el funcionamiento del generador de vistas, se procede a usar imágenes creadas con el programa explicado en la sección anterior. Primero se corrobora qué tan bien funciona el generador de vistas cuando cada par de cámaras solo se encuentran trasladadas entre sí. Para ello se usan las imágenes de la figura 5.8, las cuales corresponden a las imágenes que entrega cada sensor al generador de vistas, y en la figura 5.9 se pueden ver algunas vistas sintéticas usando el método de interpolación a la hora de juntar las vistas.



(a) **(b)**

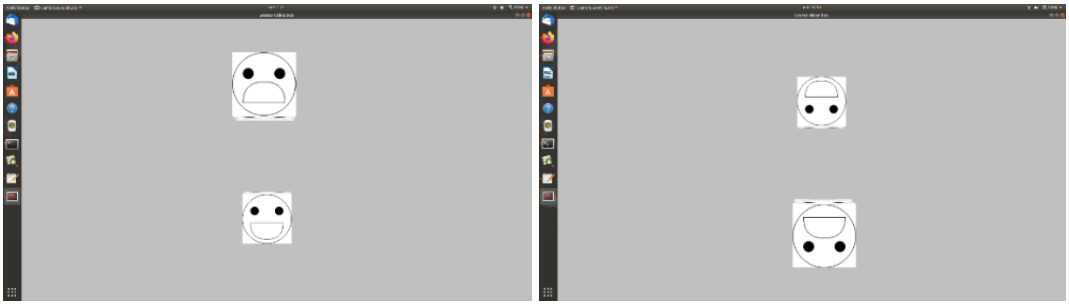
Figura 5.8: (a) Imagen que del sensor virtual izquierdo; (b) imagen del sensor virtual derecho (derecha)



(a) **(b)**

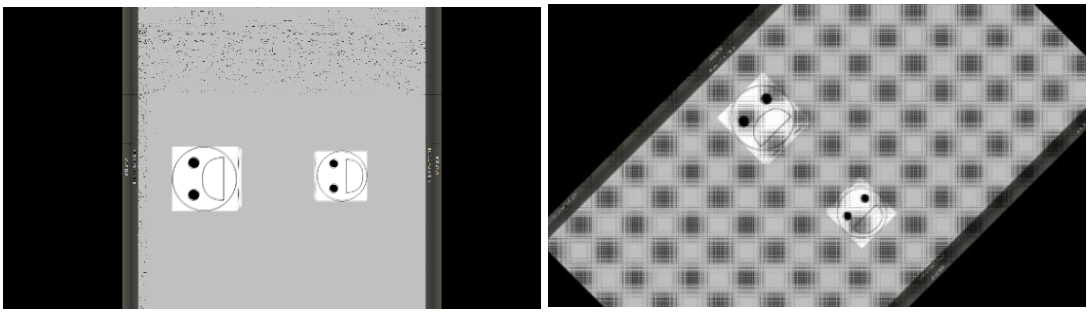
Figura 5.9: A partir de las imágenes de la figura 5.8 y usando la interpolación como método de combinación de imágenes: (a) vista sintética con q igual a 0.5; (b) vista sintética con q igual a 0.75

Para comprobar el funcionamiento de la rotación se usaron las imágenes de la figura 5.10, donde un par de cámara está rotado en 180° con respecto al otro, pero la cámara que proporciona la imagen que entrega el sensor de imagen-profundidad están ubicadas en el mismo punto. En la figura 5.11 se pueden apreciar algunos resultados de la generación de vistas sintéticas, volviendo a utilizar el método de interpolación a la hora de juntar las vistas.



(a) **(b)**

Figura 5.10: (a) Imagen del sensor sin voltear; (b) imagen del sensor volteado

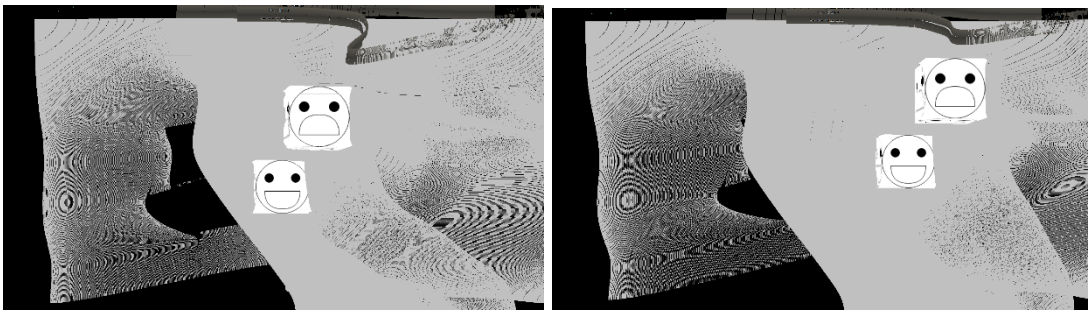


(a) **(b)**

Figura 5.11: A partir de las imágenes de la figura 5.10 y usando la interpolación como método de combinación de imágenes: (a) vista sintética con q igual a 0.5; (b) vista sintética con q igual a 0.75

Al observar las vistas generadas tanto en la figura 5.9 y 5.11, se puede apreciar que las fórmulas presentadas en la sección 5.2 permiten llevar las imágenes obtenidas por ambos sensores a un punto intermedio para finalmente juntarlas en un sola imagen.

Hasta ahora las imágenes se juntaron mediante el método de interpolación. Con respecto al método de reemplazo, en la figura 5.12 se obtienen algunas vistas sintéticas usando dicho método, a partir de las imágenes de la figura 5.8.



(a) **(b)**

Figura 5.12: A partir de las imágenes de la figura 5.8 y usando reemplazo como método de combinación de imágenes: (a) vista sintética con q igual a 0.5; (b) vista sintética con q igual a 0.75

En las figuras 5.9 y 5.12 se observa la obtención de una vista sintética usando el método de interpolación y del reemplazo respectivamente a la hora de juntar las imágenes. Si bien en ambos casos los cubos no presentan mayores diferencias, no se puede decir lo mismo del fondo, siendo que el método de interpolación obtuvo mejores resultados.

Ahora que se ha verificado el funcionamiento del generador de vistas, se procede a realizar pruebas con imágenes reales. Para ello primero se realizan 2 pruebas. La primera ubicando los sensores de imagen-profundidad con una distancia de 30 cm entre las imágenes de cada sensor y que se encuentren paralelas entre sí. La segunda prueba consiste en aumentar distancia a 150 cm. Las figuras 5.13 y 5.14 muestran las imágenes entregadas por cada sensor para cada caso.

(a) **(b)**

Figura 5.13: Con una separación de 30 cm entre los sensores y con los sensores en paralelo: (a) imagen entregada por el sensor izquierdo; (b) imagen entregada por el sensor derecho

(a) **(b)**

Figura 5.14: Con una separación de 150 cm entre los sensores y con los sensores en paralelo: (a) imagen entregada por el sensor izquierdo; (b) imagen entregada por el sensor derecho

Las figuras 5.15 y 5.16 muestran algunas vistas sintéticas para los casos de 30 cm y 150 cm respectivamente, siempre usando el método de reemplazo, ya que los errores en la reconstrucción espacial de las imágenes afecta en gran medida la vista sintética que se desea generar si se optara por usar el método de interpolación.

(a) **(b)**

Figura 5.15: A partir de las imágenes de la figura 5.13 y usando reemplazo como método de combinación de imágenes: (a) vista sintética con q igual a 0.25; (b) vista sintética con q igual a 0.5

(a) **(b)**

Figura 5.16: A partir de las imágenes de la figura 5.14 y usando reemplazo como método de combinación de imágenes: (a) vista sintética con q igual a 0.25; (b) vista sintética con q igual a 0.5

Al observar las figuras 5.15 y 5.16, se puede decir que si bien es posible generar vistas intermedias, los errores de la reconstrucción espacial afectan a la imagen que se obtiene y dichos errores de reconstrucción se acentúan a medida que se van separando los sensores de imagen-profundidad entre sí.

Las pruebas anteriores se realizaron dejando los sensores paralelos entre sí, sin embargo, con el fin de probar el generador de vistas de forma completa, se procedió a situar cada sensor a una distancia de 30 cm entre sus cámaras izquierda y con inclinación. En la figura 5.17 se observa la imagen entregada por cada sensor y en la figura 5.18 algunas vistas sintéticas generadas, las cuales no presentaron mayores errores a los ya mostrados en la figura 5.15.

(a) **(b)**

Figura 5.17: Con una separación de 30 cm entre los sensores y con los sensores inclinados: (a) imagen entregada por el sensor izquierdo; (b) imagen entregada por el sensor derecho

(a) **(b)**

Figura 5.18: A partir de las imágenes de la figura 5.17 y usando reemplazo como método de combinación de imágenes: (a) vista sintética con q igual a 0.25; (b) vista sintética con q igual a 0.5

El último caso a mencionar es cuando el parámetro *q* es 0 o 1, es decir, se genera una vista en la posición de la cámara izquierda de uno de los sensores. En la figura 5.19 se puede apreciar estos casos, donde se observan imágenes ruidosas. Esto debido a que el algoritmo programado al tomar las nubes de puntos, y después de realizar las operaciones pertinentes, no fue capaz de llevar cada punto a su posición original en la imagen. Estos casos críticos eventualmente se podría solucionar si se indica que en estos momento se muestre directamente la imagen entregada por el sensor correspondiente.

(a) **(b)**

Figura 5.19: A partir de las imágenes de la figura 5.13 y usando reemplazo como método de combinación de imágenes: (a) vista sintética con q igual a 0; (b) vista sintética con q igual a 1

Capítulo 6

6. Interfaz de usuario e integración de subsistemas

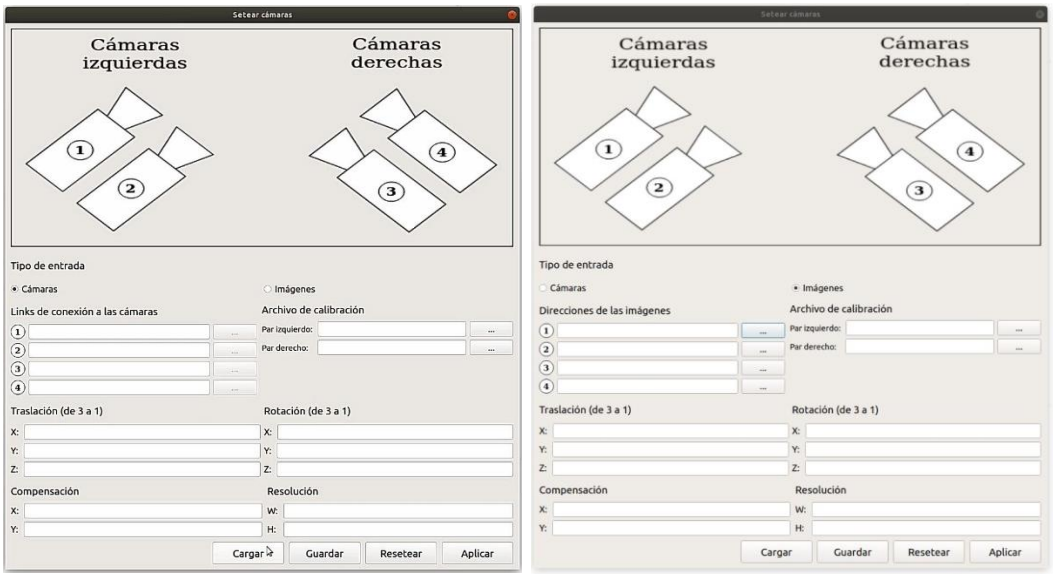
Con el fin de que el sistema pueda ser manipulado de forma sencilla, es imprescindible que este cuente con una interfaz de usuario. Para este proyecto se optó por desarrollar dicha interfaz usando el entorno de trabajo Qt.

La ventana principal, la cual se muestra en la figura 6.1, la cual está diseñada para mostrar hasta tres imágenes a la vez: A la izquierda muestra la imagen que entrega el sensor de imagen-profundidad izquierdo; a la derecha muestra la imagen que entrega el sensor de imagen-profundidad derecho; y en el centro la imagen sintética.

Figura 6.1: Ventana principal de la interfaz de usuario

En la parte superior de la ventana principal se encuentran tres menús despegables. El primero de ellos *"Cámaras"*, solo cuenta con la opción de *"Setear cámaras"* (figura 6.2) que permite configurar los parámetros de los pares de cámaras mediante la ventana que se muestra en la figura 6.3.

Figura 6.2: Menú *"Cámaras"* abierto



(a) **(b)**

Figura 6.3: Ventana para configurar parámetros de los pares de cámaras: (a) usando cámaras conectadas como entrada; (b) usando imágenes como entrada

Las ventanas que se muestran en la figura 6.3 poseen varias secciones. La primera *"Tipo de entrada"* indica si el sistema se usará con cámaras conectadas (seleccionando la opción de *"Cámaras"*) o si solo se trabajará con cuatro imágenes en específico (seleccionando la opción *"Imágenes"*). Si se trabaja con cámaras, en la sección de *"Links de conexión a las cámaras"* se debe ingresar los localizadores de recursos uniforme que permite la comunicación con cada cámara. En cambio si se trabaja con imágenes, en la sección *"Direcciones de las imágenes"* se debe ingresar las rutas de las imágenes a usar, la cual se puede escribir o bien apretando el botón *"..."* que está justo al lado para seleccionar la imagen desde el administrador de archivos.

La siguiente sección *"Archivos de calibración"* se deben indicar, ya sea escribiendo la ruta completa o apretando el botón *"..."* que está justo al lado para usar el gestor de archivos, el archivo con los parámetros de calibración de cada par de cámara.

En *"Traslación (de 3 a 1)"* se le debe indicar cuánto se debe trasladar en los ejes z, x e y, en metros, la imagen capturada por la cámara 3 para que quede en la misma posición de la cámara 1. De forma similar la sección *"Rotación (de 3 a 1)"* se le debe

indicar en cuánto se debe rotar en los ejes z, x e y, en radianes, para que la imagen de la cámara 3 quede en la misma orientación de la cámara 1.

Las últimas dos secciones son *"Resolución"*, en donde se indica la resolución de las cámaras en píxeles, y en *"Compensación"* se ingresan los parámetros para realizar la corrección explicada en el capítulo cinco.

La ventana también cuenta con cuatro botones: *"Cargar"*, permite seleccionar desde el gestor de archivos uno que cuente con la configuración deseada; *"Guardar"*, permite guardar en un archivo los parámetros colocados en la ventana (dicho archivo debe de tener por extensión .ini); *"Resetar"*, limpia todos los parámetros de la ventana; y *"Aplicar"*, el cual fija todos los parámetros entregados para que el sistema comience a funcionar.

El menú despegable de la ventana principal, *"Combinación de imágenes"* (figura 6.4), es usado para indicar quá tipo de combinación de imagen se desea usar.

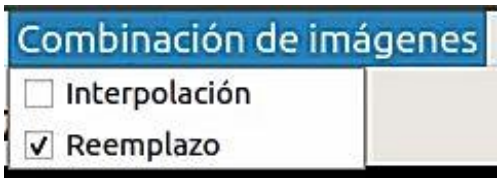


Figura 6.4: Menú "Combinación de imágenes" abierto

Finalmente el menú despegable correspondiente a *"Sensor de profundidad"* el cual, como se muestra en la figura 6.5, solo tiene la opción de *"Setear sensor"*, que despliega una ventana que permite indicar cuál de los algoritmos usar al momento de calcular los mapas de disparidad (BM o SGBM) junto con los parámetros que se desea usar para dichos algoritmos y los parámetros del filtro. Esta ventana en sus dos opciones variantes se muestra en la figura 6.6.

Figura 6.5: Menú "Sensor de profundidad" abierto

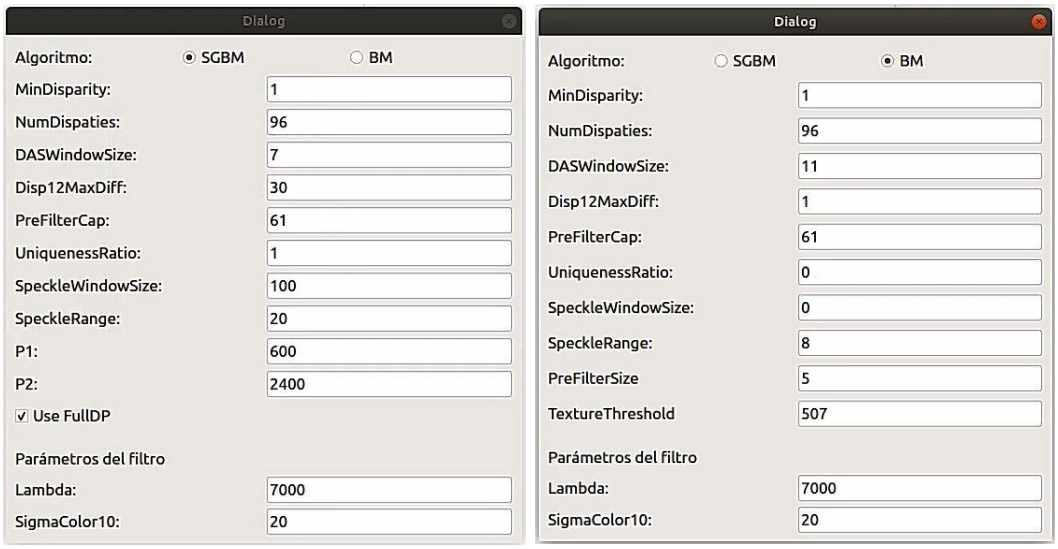
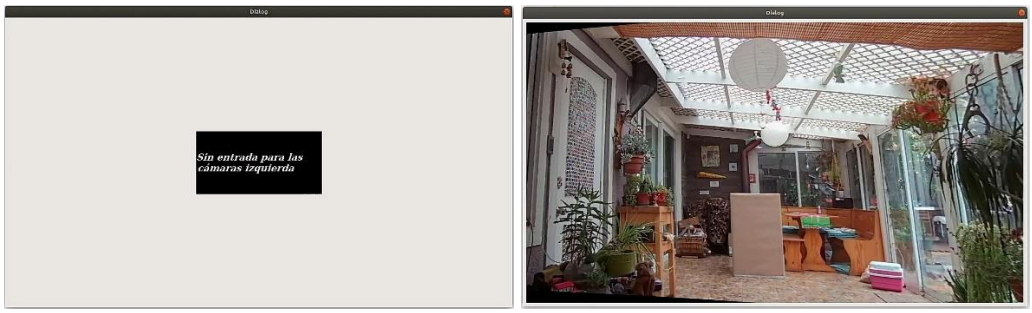


Figura 6.6: Ventana para configurar los parámetros que se usarán para calcular los mapas de disparidad: (a) ventana si se opta por usar SGBM; (b) ventana si se opta por usar BM

Siguiendo con la ventana principal, debajo de la zona de las imágenes, se encuentra una barra de desplazamiento horizontal que permite fijar el valor del parámetro *q*, o en otras palabras, indicar en qué punto se desea generar la imagen sintética. Inmediatamente a la derecha de la barra de desplazamiento, se muestra el valor actual del parámetro *q*.

Ya terminando con la ventana principal, en la parte inferior se encuentran cuatro botones: *"Expandir"*, *"Guardar"*, *"Video"* y *"Generar"*. El botón *"Expandir"* abre el menú de la figura 6.7, donde se selecciona que imagen (la del sensor izquierdo, derecho y/o la imagen virtual) se quiere mostrar en una ventana emergente (figura 6.8).

Figura 6.7: Menú que se abre al apretar el botón "Expandir"



(a) **(b)**

Figura 6.8: Ventana usada para mostrar imágenes más grande: (a) ventana en el caso que no se cuente con ninguna imagen; (b) ventana en caso de que ya exista un imagen que mostrar

El botón de *"Guardar"* abre el menú de la figura 6.9. En dicho menú se muestra una lista de lo que se permite almacenar en el computador y en la sección de *"Nombre de los archivos"* se debe escribir un nombre en común para los archivos. El botón de *"Guardar"* del menú de la figura 6.9 hace que el usuario seleccione la ubicación donde se desea almacenar los archivos y efectúa el guardado.

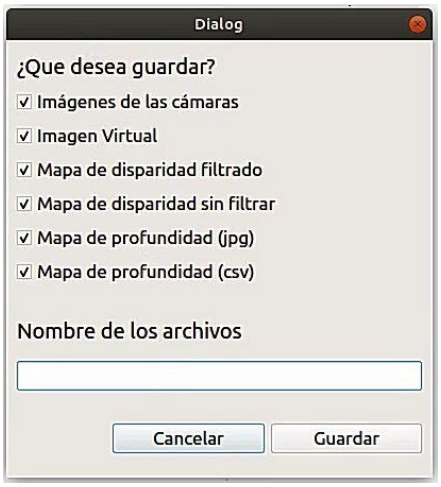


Figura 6.9: Menú de guardado

El siguiente botón de la ventana principal, *"Video"*, tiene una funcionalidad dependiente de que si se está usando como entrada cámaras o solo imágenes. Si se trabaja con cámaras, el sistema estará constantemente generando la vista sintética a partir de las imágenes que llegan desde las cámaras y se irá actualizando las vistas en la ventana principal. Este proceso se detiene cuando se vuelve a apretar el botón de *"Video"*, lo cual a su vez se abre el menú de la figura 6.10, en donde se da la opción

de guardar el video en un archivo en formato “avi” especificando un nombre y las imágenes por segundo que se desean. Al momento de apretar el botón de “Guardar” del menú de la figura 6.10, permite seleccionar el lugar donde se quiere almacenar usando el administrador de archivos y efectúa el guardado.

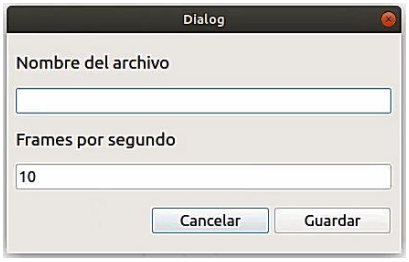


Figura 6.10: Menú para guardar video

Por otro lado, si se trabaja con la opción de solo imágenes, entonces al apretar el botón *“Video”* el sistema empieza a generar vistas sintéticas de la escena, empezando desde q igual a 0 hasta q igual a 1, ida y vuelta, con un paso de 0.01 y actualizando la vista de la cámara virtual cada vez. Una vez finalizado el proceso se vuelve a mostrar el menú de la figura 6.10 para tener la opción de guardar el video que se acaba de generar.

Finalmente el botón de *“Generar”* permite crear una vista sintética en un punto deseado usando como base las últimas imágenes con la que cuenta el sistema. Este botón junto con el de *“Guardado”* se encuentran deshabilitados mientras el sistema este generando un video.

6.1. Resultados

Para comprobar eficiencia del sistema, se ejecutan varias pruebas con el fin de medir el tiempo desde la captura de las imágenes hasta la generación de la vista sintética.

En total el programa se ejecutara 8 veces mientras se itera entre las siguientes opciones: Los dos tipos de entrada; los algoritmos para calcular los mapas de disparidad; y los dos algoritmos de combinación de imágenes. Durante cada ejecución se registran los siguientes tiempo: La adquisición de imágenes, el cual va desde obtener

las imágenes más recientes de las cámaras hasta la eliminación de la distorsión y rectificación de estas; el cálculo de los puntos en el espacio, que consiste en el cálculo de los mapas de disparidad y las matrices de tres canales con la información espacial de cada pixel; y finalmente el tiempo de combinación de imágenes.

Antes de seguir se debe indicar que en las pruebas que usaban las cámaras como entrada, lo que se realizó fue dejar el programa ejecutándose por un tiempo. Esto último junto con el hecho de que al usar BM para el cálculo de los mapas de disparidad el programa es más rápido, como se verá en los resultados finales, provoca que exista una diferencia en la cantidad de muestras de cuando se usa SGBM y BM. Por el contrario cuando se utiliza imágenes como entrada, como no se tiene que estar constantemente actualizando las imágenes, se realizó un número fijo de pruebas. En esta sección se optó por mostrar todos los datos recopilados, por lo que en los siguientes gráficos se observara esta diferencia en la cantidad de muestras.

En la figura 6.11 se muestran los tiempos registrados en la adquisición de imágenes, los cuales solo se midieron cuando la entrada del sistema correspondía a las cámaras. Los datos se agruparon en dos series: Cuando se usa el algoritmo de SGBM para calcular los mapas de disparidad (puntos azules) y cuando se usa BM (puntos naranjos).

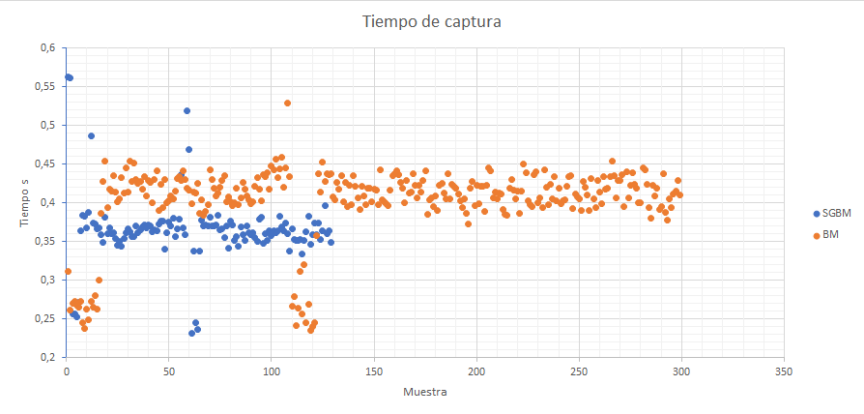


Figura 6.11: Tiempos de adquisición de imágenes registrados

En ambos casos cada vez que se inicializa el programa parten con tiempos de captura de alrededor de los 0.25 segundos, luego aumentan hasta un punto de estabilización. El tiempo de captura en la adquisición de imágenes cuando el programa

utiliza el algoritmo de SGBM se estabiliza en un punto más bajo que cuando se utiliza el algoritmo BM, siendo que el primero presenta una mediana de 0.36 segundos y el segundo una mediana de 0.41 segundos (Se usaron los tiempos ya estables para obtener la mediana).

En la figura 6.12 se muestran los tiempos registrados en el cálculo de los puntos espaciales de las imágenes. Al igual que el caso anterior, dichos tiempos fueron medidos solamente cuando la entrada del sistema corresponde a las cámaras y de nuevo se agruparon los datos según el algoritmo que se usó para calcular los mapas de disparidad: SGBM (puntos azules) y BM (puntos naranjos).

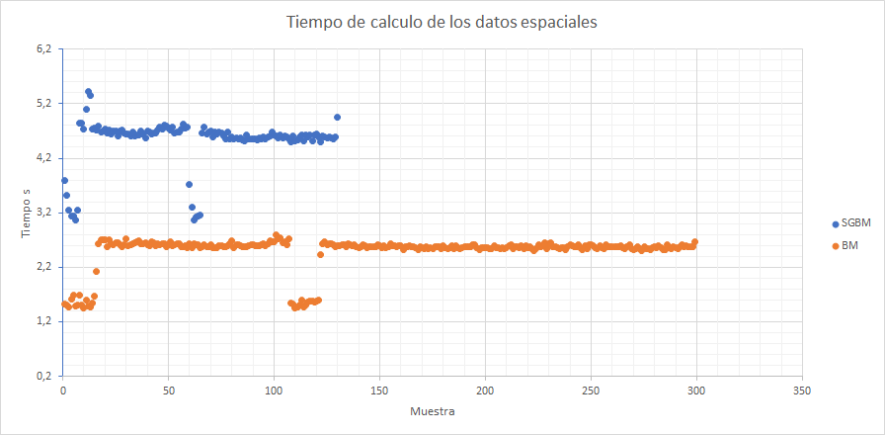


Figura 6.12: Tiempos de cálculo de los mapas de disparidad y las matrices de tres canales con la información espacial de cada pixel

Al observar los datos se aprecia que en ambos casos parten tiempos bajos y luego aumentan y se estabilizan en un punto. En esta ocasión cuando se usa el algoritmo BM el programa presenta un tiempo de calculó menor a que si se utiliza el algoritmo SGBM. La mediana cuando el tiempo de estabiliza usando SGBM es de 4.63 segundos y cuando se usa BM es de 2.58 segundos.

En la figura 6.13 se muestran los tiempos registrados al momento de combinar las imágenes usando el método del reemplazo. En este caso los tiempos se dividen en cuatro grupos: Usando las cámaras como entrada y el algoritmo SGBM (puntos azules); usando las cámaras como entrada y el algoritmo BM (puntos naranjas); usando

imágenes como entrada y el algoritmo SGBM (puntos grises); y usando imágenes como entrada junto con el algoritmo BM (puntos amarillos). Es importante recalcar en este punto que los tiempos registrados corresponden solo a los tiempos del algoritmo de combinación de imágenes mediante reemplazo, sin embargo, dicho tiempo se vio afectado dependiendo de que otras tareas debía realizar el programa, aunque en el caso que se usaron imágenes como entrada, al solo tener que calcular los mapas de profundidad una vez al comienzo, es donde los tiempos se vieron menos afectados.

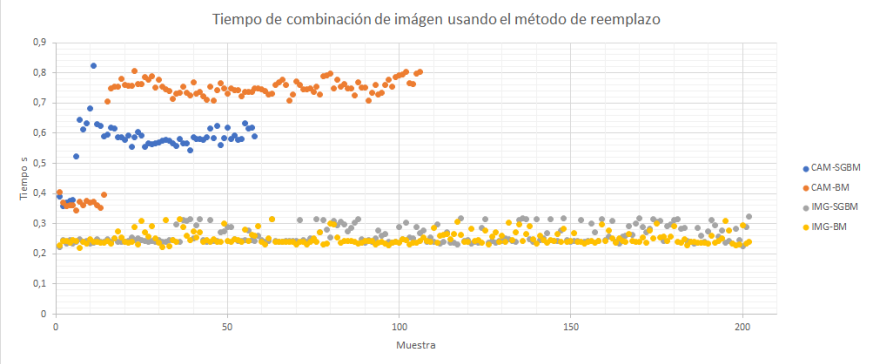


Figura 6.13: Tiempos de combinación de imágenes usando reemplazo

En este caso se puede observar que de forma similar a los anteriores casos cuando se utiliza las cámaras como entrada, después de cierto punto el tiempo de la operación aumenta y se queda estable. Por otro lado se observa que cuando se trabaja con imágenes como entrada, el tiempo de esta operación se mantiene estable a los largo de la ejecución del programa. La mediana de los tiempos es: Para el caso de que la entrada sea las cámaras y se use el algoritmo SGBM el tiempo es 0.58 segundos; para cuando la entrada son las cámaras y se use el algoritmo BM el tiempo es de 0.75 segundos; y finalmente cuando la entrada son imágenes, el tiempo es de 0.24 segundos.

En la figura 6.14 se muestran los tiempos registrados al momento de combinar las imágenes usando el método del interpolación. Los datos se agruparon de la misma forma que el caso anterior y al igual que el caso anterior los tiempos registrados corresponden solo a los tiempos del algoritmo de combinación de imágenes por interpolación, sin embargo, dicho tiempo se vio afectado dependiendo de que otras tareas debía realizar el programa.

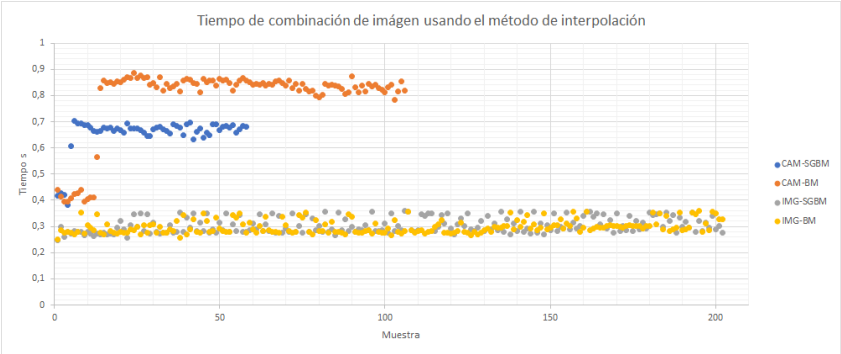


Figura 6.14: Tiempos de combinación de imágenes usando interpolación

Los tiempos presentan el mismo comportamiento del caso anterior, con la diferencia en la mediana de los tiempos: Para el caso de que la entrada sea las cámaras y se use el algoritmo SGBM el tiempo es 0.67 segundos; para cuando la entrada son las cámaras y se use el algoritmo BM el tiempo es de 0.84 segundos; y finalmente cuando la entrada son imágenes, el tiempo es de 0.29 segundos.

Considerando que cuando la entrada corresponde a solo la imágenes, el único tiempo importante corresponde al de combinación de imagen, ya que es la única operación que se repite, el tiempo que le toma al programa generar una vista sintética es:

Tabla 6.1: Tiempo de generación de vista sintética

Tipo de entrada	Algoritmo para obtener el mapa de disparidad	Método de combinación de imágenes	Tiempo en segundos
Cámara	SGBM	Reemplazo	5.57
Cámara	SGBM	Interpolación	5.67
Cámara	BM	Reemplazo	3.74
Cámara	BM	Interpolación	3.83
Imágenes	BM / SGBM	Reemplazo	0.24
Imágenes	BM / SGBM	Interpolación	0.29

Capítulo 7

7. Conclusiones y trabajos futuros

Construir un sistema de realidad virtual que permita al usuario navegar libremente en un escenario real no es una tarea trivial, sin embargo, con lo desarrollado en este proyecto se logró dar unos primeros pasos a esa meta. Se desarrolló un sensor de imagen-profundidad el cual a partir de dos cámaras paralelas entrega una imagen con los datos espaciales de esta; con dicha información se logró transformar las imágenes para que quedasen en una perspectiva intermedia entre los dos sensores y combinarlas en una sola vista sintética; y finalmente se desarrolló una interfaz de usuario que permitió obtener de forma continua información de los sensores e ir generando un vista sintética en una posición configurable por el usuario.

En cuanto a los futuros trabajos a realizar, tenemos lo siguiente:

- **Ajustar el proceso de calibración:** Como se mencionó en la sección 4.4, el hecho de eliminar la distorsión antes de realizar el procedimiento de calibración normal afecta en los resultados del sensor de imagen-profundidad. Para corregir este problema se recomienda realizar un proceso de calibración normal y al momento de mostrar la imagen al usuario, recortarla para solo mostrar el área de interés.
- **Expandir la cantidad de puntos de medición:** En el presente proyecto se utilizaron dos puntos de medición para obtener una vista sintética en un punto intermedio entre ellos. Con el fin de poder obtener más vistas posibles, resulta necesarios agregar más puntos de medición. Se recomienda primero agregar un tercer punto de medición para tener un plano de perspectivas sintéticas posibles, como lo recomienda en el trabajo *"View Interpolation of Multiple Cameras Based on Projective Geometry"* [7].

- **Mejora de calidad de imagen:** A lo largo de la memoria se pudo apreciar que las imágenes sintéticas presentaban ruido en forma de puntos negros. Para mejorar la calidad de la imagen resultante, se debe trabajar con el fin de poder rellenar dichos puntos con la información que tengan los píxeles vecinos.
- **Mejorar tiempos de procesamiento:** Como se vio en el capítulo anterior, procesar una sola imagen sintética toma varios segundos. Estos tiempos son particularmente malos si se busca realizar una aplicación que funcione en tiempo real. Si bien en general se debe trabajar para que todos los tiempos bajen, se debe dar prioridad a los tiempos de cálculo de las datos espaciales dado que son los que más retrasan el sistema. Tres alternativas para mejorar los tiempos del sistema son:
    - Cambiando el algoritmo de detección de pixeles correspondientes, a uno que segmente la imagen y detecte correspondencia entre segmentos, como se realiza en el trabajo *"High-Quality Video View Interpolation Using a Layered Representation"* [6], lo que bajaría significativamente la cantidad de correspondencia a detectar.
    - Implementar los diferentes algoritmos en una unidad de procesamiento gráfico (GPU), como lo hace en el mismo trabajo realizado por los investigadores de Microsoft [6], o en una matriz de puertas lógicas programable (FPGA) como se puede apreciar en el trabajo *"Stereo Vision Algorithms for FPGAs"* [40].
    - Cambiar el sensor de imagen-profundidad desarrollado por un sensor comercial, lo cual aliviaría la carga computacional. En el trabajo *"Wide angle virtual view synthesis using two-by-two Kinect V2"* [8] se propone el uso de una Kinect V2 la cual funciona a 30 fps y en un rango de 0.5 – 4.5 metros. Una alternativa a la Kinect son los sensores ofrecidos por la empresa "Orbbec" [41] los cuales cuentan con sensores que trabajan a 30 fps y tienen un rango de funcionamiento de 0.6 hasta 8 metros con precios de alrededor de 150 dolares. Otra alternativa son los sensores de la empresa "StereoLabs" [42], su sensor más económico tiene un valor

de 350 dolares, sin embargo es capaz de entregar mapas de profundidad a 100 fps (como máximo) y con un rango de funcionamiento de 0.3 a 25 metros. Independiente del sensor, si se opta por adquirir un sensor comercial se tener en cuenta que es posible encontrar los problemas similares a los expuestos en *"Wide angle virtual view synthesis using two-by-two Kinect V2"* [8] (y explicados en la sección 2.1).

- **Implementación de visión estereoscópica:** Con el fin de avanzar a una aplicación de realidad virtual, al sistema se le debe realizar las modificaciones pertinentes para generar dos imágenes sintéticas con una pequeña separación, de unos 6 cm, y así poder generar una visión estereoscópica.

Bibliografía

[1] cv::StereoBM Class Reference [en línea]
<https://docs.opencv.org/3.2.0/d9/dba/classcv_1_1StereoBM.html>
[consulta: 30 Enero 2021]

[2] cv::StereoSGBM Class Reference [en línea]
<https://docs.opencv.org/3.2.0/d2/d85/classcv_1_1StereoSGBM.html>
[consulta: 30 Enero 2021]

[3] Frías, G. NBA ofrece una nueva experiencia virtual desde tu sofá. [en línea] CNN en Español. 27 de Marzo, 2019
<https://cnnespanol.cnn.com/video/nba-realidad-aumentada-nuevos-juegos-vo-portafolio-cnnee/>
[consulta: 29 Enero 2021]

[4] Moreira, M. Turismo virtual, un modelo de negocios que llegó para quedarse. [en línea]. La Hora. 25 de Diciembre, 2020
<https://lahora.com.ec/noticia/1102336461/turismo-virtual-un-modelo-de-negocios-que-llego-para-quedarse>
[consulta: 29 Enero 2021]

[5] Maero F. Realidad virtual en el consultorio: Una revisión de Psious. [en línea] Psyciencia. 31 de Octubre, 2016
<https://www.psyciencia.com/realidad-virtual-en-el-consultorio-un-analisis-a-psious/>
[consulta: 31 Enero 2021]

[6] Zitnick L., Kang S., Uyttendaele M., Winder S. y Szeliski R. High-Quality Video View Interpolation Using a Layered Representation. En: SIGGRAPH04: Special Interest Group on Computer Graphics and Interactive Techniques (2004, Los Angeles California) ACM SIGGRAPH 2004 Papers, Nueva York, Estados Unidos, Association for Computing Machinery, 2004, pp. 600-608

[7] Makoto K., Saito H., Yaguchi S. y Inamoto N. View Interpolation of Multiple Cameras Based on Projective Geometry. En: International Workshop on Pattern Recognition and Understanding for Visual Information. 2002

[8] Chang H. y Hang H. Wide angle virtual view synthesis using two-by-two Kinect V2. En: APSIPA ASC 2017 (9°, 2017, Kuala Lumpur) 2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Asia-Pacific Signal and Information Processing Association, 2017, pp. 1083-1091

[9] Kitahara I., Sakamoto R., Satomi M., Tanaka K. y Kogure K. Cinematized reality: Cinematographic camera controlling 3D free-viewpoint video. En: CVMP 2005 the 2nd IEE European Conference on Visual Media Production (2°, 2005, Londres) Visual Media Production, 2005. CVMP 2005. The 2nd IEE European Conference on, Institution of Engineering and Technology, 2005, pp. 154-161

[10] Zhang J., Siritanawan P., Yue Y., Yang C., Wen M. y Wang D. A Two-step Method for Extrinsic Calibration between a Sparse 3D LiDAR and a Thermal Camera. En: International Conference on Control, Automation, Robotics and Vision (ICARCV) (15°, 2018, Singapur) 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), Institute of Electrical and Electronics Engineers, 2018, pp. 1039-1044

[11] Arducam. Depth Mapping with Arducam Stereo Camera and Raspberry Pi [en línea] <https://www.youtube.com/watch?v=eBZm40z7E8Y> [consulta: 21 Noviembre 2019]

[12] Bradski G. The OpenCV Libray. Dr. Dobb's Journal of Software Tools. 25(11): 120, 122-125, Noviembre, 2000

[13] Depth Map from Stereo Images [en línea] <https://docs.opencv.org/3.4/dd/d53/tutorial_py_depthmap.html> [consulta: 14 Noviembre 2020]

[14] Ubuntu Desktop Guide [en línea] <https://help.ubuntu.com/18.04/ubuntu-help/index.html> [consulta: 30 Enero 2021]

[15] ISO/IEC. International Standard ISO/IEC 14882:2011(E) – Information technology – Programming Language – C++, Ginebra, Suiza, 2011, p. 1338

[16] OpenCV modules [en línea] <https://docs.opencv.org/3.2.0/> [consulta: 30 Enero 2021]

[17] Qt 5.9 [en línea] <https://doc.qt.io/archives/qt-5.9/index.html> [consulta: 31 Enero 2021]

[18] cv::VideoCapture Class Reference [en línea] <https://docs.opencv.org/3.2.0/d8/dfe/classcv_1_1VideoCapture.html> [consulta: 30 Enero 2021]

[19] Camera Calibration [en línea] <https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html> [consulta: 21 Marzo 2021]

[20] Mallick S. y Sadekar K. Camera Calibration using OpenCV [en línea] <https://learnopencv.com/camera-calibration-using-opencv/> [consulta: 22 Marzo 2021]

[21] Zhang Z. A Flexible New Technique for Camera Calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence. 22(11): 1330-1334, Noviembre, 2000

[22] Ghosh S. Camera calibration using C++ and OpenCV [en línea] <https://sourishghosh.com/2016/camera-calibration-cpp-opencv/> [consulta: 22 Marzo 2021]

[23] Geometric Image Transformations [en línea] undistort() <https://docs.opencv.org/3.2.0/da/d54/group__imgproc__transform.html> [consulta: 22 Marzo 2021]

[24] Camera Calibration and 3D Reconstruction [en línea] stereoCalibrate() <https://docs.opencv.org/3.2.0/d9/d0c/group__calib3d.html> [consulta: 23 Marzo 2021]

[25] Camera Calibration and 3D Reconstruction [en línea] stereoRectify() <https://docs.opencv.org/3.2.0/d9/d0c/group__calib3d.html> [consulta: 23 Marzo 2021]

[26] Trucco E. y Verri A. Stereopsis. En su: Introductory Techniques for 3-D Computer Vision. Estados Unidos, Prentice Hall, 1998, pp. 139-176.

[27] Ghosh S. Stereo calibration using C++ and OpenCV [en línea] <https://sourishghosh.com/2016/stereo-calibration-cpp-opencv/> [consulta: 7 Diciembre 2020]

[28] Geometric Image Transformations [en línea] initUndistortRectifyMap() <https://docs.opencv.org/3.2.0/da/d54/group__imgproc__transform.html> [consulta: 22 Marzo 2021]

[29] Geometric Image Transformations [en línea] remap() <https://docs.opencv.org/3.2.0/da/d54/group__imgproc__transform.html> [consulta: 22 Marzo 2021]

[30] thanhit08. 2014. Big different between StereoSGBM and gpu::StereoBM_GPU [en línea] <https://answers.opencv.org/question/29392/big-different-between-stereosgbm-and-gpustereobm_gpu/> [consulta: 1 Abril 2021]

[31] McCormick, C. Stereo Vision Tutorial – Part I [en línea] <http://mccormickml.com/2014/01/10/stereo-vision-tutorial-part-i/> [consulta: 14 Abril 2021]

[32] cv::ximgproc::DisparityWLSFilter Class Reference [en línea] <https://docs.opencv.org/3.2.0/d9/d51/classcv_1_1ximgproc_1_1DisparityWLSFilter.html> [consulta: 30 Enero 2021]

[33] Disparity map post-filtering [en línea] <https://docs.opencv.org/master/d3/d14/tutorial_ximgproc_disparity_filtering.html> [consulta: 7 Diciembre 2020]

[34] Min D., Choi S., Lu J., Ham B., Sohn K. y Do M. Fast Global Image Smoothing Based on Weighted Least Squares. IEEE Transactions on Image Processing. 23(12): 5638-5653, Diciembre, 2014

[35] 2ros0. 2018. Derivation for perspective transformation matrix (Q) [en línea] <https://answers.opencv.org/question/187734/derivation-for-perspective-transformation-matrix-q/> [consulta: 31 Enero 2021]

[36] Camera Calibration and 3D Reconstruction [en línea] reprojectImageTo3D() <https://docs.opencv.org/3.2.0/d9/d0c/group__calib3d.html> [consulta: 30 Enero 2021]

[37] MATLAB. MATLAB version 9.5.0.1049112 (R2018b) Update 3. Natick, Massachusetts, The Mathworks, Inc., 2018

[38] Camera Calibration and 3D Reconstruction [en línea] Rodrigues() <https://docs.opencv.org/3.2.0/d9/d0c/group__calib3d.html> [consulta: 30 Enero 2021]

[39] JavaFX [en línea] <https://openjfx.io/> [consulta: 31 Enero 2021]

[40] Mattoccia S. Stereo Vision Algorithms for FPGAs. En: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshop (CVPRW) (2013, Portland, Oregón, Estados Unidos) 2013 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshop, Portland, Oregón, Estados Unidos, Institute of Electrical and Electronics Engineers, 2013, pp. 636-641

[41] Orbbec [en línea] <https://orbbec3d.com/> [consulta: 04 Abril 2021]

[42] StereoLabs [en línea] <https://www.stereolabs.com/> [consulta: 04 Abril 2021]