UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE

"Desarrollo de software para sintetizar la vista de una cámara virtual a partir de múltiples cámaras Kinect"

LUIS ANDRÉS MUÑOZ ROMÁN

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICO

PROFESOR GUIA: **AGUSTÍN GONZÁLEZ V.**

PROFESOR CORREFERENTE: **MARCOS ZÚÑIGA B.**

“Desarrollo de software para sintetizar la vista de una cámara virtual a partir de múltiples cámaras Kinect”

LUIS ANDRÉS MUÑOZ ROMÁN

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICO

PROFESOR GUIA: AGUSTÍN GONZÁLEZ V.

Resumen

El objetivo principal de este trabajo es elaborar un software, llamado Virtualview, que permita componer la imagen que vería una cámara virtual, utilizando la información proveniente de múltiples cámaras Kinect. En la práctica solo se utilizaron dos cámaras, con el fin de virtualizar escenas y mostrarlas en las gafas de realidad virtual Oculus Rift.

La estrategia para desarrollar esta aplicación se basa en el uso de la tecnología de medición de distancia que proveen las cámaras Kinect, con las cuales es posible obtener información métrica de la escena, permitiendo reproyectar los pixeles de color y obtener una aproximación a una vista virtual.

Para conseguir la vista virtual es necesario unificar la información proveniente desde las cámaras mediante procesos de calibración, lo que permite encontrar las posiciones relativas entre estas, además de encontrar sus parámetros intrínsecos y extrínsecos. Adicionalmente se exploraran estrategias para combinar la información, ponderando las contribuciones de cada cámara por parámetros como la proximidad o discrepancia angular entre estas.

Para finalizar este experimento se despliegan las imágenes generadas en las gafas de realidad virtual Oculus Rift, donde se observan imágenes que debido a la falta de post-procesamiento no logra convencer de que es una cámara móvil lo que se observa.

Glosario

1. Virtualizar: Este concepto se refiere a llevar toda la información de color y disposición espacial de los objetos frente a la cámara, a un objeto en la memoria del computador, de modo de poder ser recreada posteriormente.

2. Cámaras TOF (time of flight): Las cámaras TOF corresponden a un sistema de medición de distancia entre la cámara y los puntos de una escena, la cual se resuelve a través del conocimiento de la velocidad de la luz.

3. Renderizar: Este concepto hace referencia al proceso de generar una imagen, en base a un modelo 3D y una iluminación determinada.

4. API (Application Programming Interface): Corresponde a un conjunto de rutinas, protocolos y herramienta para desarrollar un software. Provee un conjunto de bloques de construcción que el programador ensambla para conseguir solucionar su problema.

5. Campo de visión: Es la extensión del mundo observable en un momento dado.

6. IDE (Integrated Development Environment): Es una sigla en inglés que significa ambiente de desarrollo integrado, que habitualmente contiene un editor de texto, un depurador, un compilar, entre otros.

7. Sobre entrenar: Este concepto se refiere al error que se producido al ajustar un modelo matemático demasiado ajustado con datos insuficientes o que los datos utilizados estén altamente correlacionados. Lo que produce que el error del modelo sea pequeño entorno a los datos utilizados, pero grande lejos de estos.

8. DLT (Direct Linear Transformation): Corresponde a una estrategia para resolver sistemas de ecuaciones que no corresponden a ecuaciones lineares ordinarias, ya que las ecuaciones pueden diferir en un factor multiplicativo desconocido. Para

resolver este problema las ecuaciones son reescritas de forma homogénea utilizando una matriz antisimétrica, posteriormente es posible resolver el sistema de ecuaciones utilizando las estrategias estándares de las ecuaciones lineare ordinarias. La combinación de reescribir las ecuaciones en conjunto con la solución mediante métodos tradicionales se conoce como el algoritmo DLT.

9. Travelling: Corresponde a un concepto utilizado en la industria cinematográfica para describir un movimiento de cámara, en el cual la cámara se desplaza de su ubicación mientras realiza su registro.

Índice general

1.. Introducción 3
1.1. Motivación 3
1.2. Descripción del problema 3
1.3. Objetivos de la memoria 4
1.4. Estructura de la memoria 5

2.. Estado del arte en visión 3D 6
2.1. Tecnologías relacionadas con la realidad virtual 6
2.2. Desarrollos recientes en visión 3D 9
2.3. Tecnologías compatibles con Virtual View 10

3.. Análisis del problema y solución propuesta 19
3.1. Definición de requerimientos 19
3.2. Solución Propuesta 20

4.. Evaluaciones de Virtual View 27
4.1. Calibración de parámetros de las cámaras Kinect 27
4.2. Mejoras a las imágenes de profundidad 28
4.3. Resultados de las vistas generadas por las cámaras virtuales 29
4.4. Rendimientos a 'tiempo real' 31
4.5. Despliegue de imágenes en el Oculus Rift 31

5.. Reflexiones finales 33
5.1. Conclusiones 33

5.2. Trabajos futuros 34

6.. Bibliografía 36

7.. Anexos 39

1. INTRODUCCIÓN

1.1. Motivación

Con la aparición de las gafas de realidad virtual se abre un mundo de aplicaciones que aprovechen su potencial, una de ellas es la idea de mi profesor guía, quien imaginó la posibilidad de presenciar un partido de fútbol como si se estuviese de pie en la cancha. Con esa idea en mente se considera realizar una aplicación que genera vistas intermedias entre dos cámaras, permitiéndonos 'ver' desde una nueva perspectiva, consiguiendo así una aproximación a esta idea.

Esta memoria nace como un desafió de integración entre la virtualización de espacios en tiempo real, a través del uso de cámaras Kinect, y el despliegue de estos en gafas de realidad virtual, lo cual se realizó utilizando las gafas Oculus Rift[20]. Además se busca explorar caminos alternativos de solución, por lo que se decide utilizar solo imágenes para conseguir el objetivo, evitando realizar levantamientos 3D, debido a lo cambiante que resultan los escenarios reales y el computo requerido para realizar cada levantamiento 3D.

1.2. Descripción del problema

El propósito de esta memoria es explorar la posibilidad de virtualizar un espacio en tiempo real, permitiendo recorrer esta escena, a través de las gafas de realidad virtual, sin intervenirla, lo que nos llevaría a un nuevo modo de presenciar contenido multimedia, tales como eventos deportivos, series de televisión o cualquier otro tipo de contenido que actualmente es transmitido por televisión.

En base a la búsqueda realizada, no se encontró ningún software que permita recorrer una escena real en tiempo real, por lo que se buscara encontrar cuáles son las dificultades técnicas que no han permitido que esta tecnología se desarrolle.

Actualmente se están desarrollando diversas gafas de realidad virtual, como por ejemplo el Oculus Rift, Vive de HTC[18] o gafas exclusivas para consolas de videojuegos como lo son las Playstation VR[19]; además se desarrollan mejores procesadores y tarjetas gráficas, por lo que las dificultades técnicas que podrían limitar la posibilidad de recorrer espacios virtualizados a tiempo real podrían no existir el día de mañana, lo que permitiría realizar de esta idea un prototipo comercial.

El software desarrollado, llamado Virtualview, se basó en uso de las cámaras Kinect, ya que estas proveen información métrica de la escena, lo que permite generar nuevas vistas de la escena, distintas a las que proveen las cámaras estáticas, a través del uso de transformaciones perspectivas. El despliegue de estas imágenes se realiza en un Oculus Rift.

1.3. Objetivos de la memoria

Para concretar el propósito de esta memoria, es necesario completar una serie de etapas que permitirán obtener resultados positivos, estos pasos esenciales que debe cumplir el software se pueden definir como los objetivos de este trabajo. Estos objetivos son:

1. Lograr calibrar un par de cámaras de modo de ubicar espacialmente la información proveniente de ambas.

2. Conseguir crear dos imágenes virtuales, en base a la información obtenida desde las cámaras reales, simulando dos cámaras ubicadas entre las reales, y que posean la separación interpupilar, para simular los ojos de un espectador.

3. Desplegar ambas imágenes virtuales en una gafas de realidad virtual, de modo de generar la sensación de estar presente en la escena.

Estos tres objetivos puedes ser desagregados en varias etapas, las cuales se analizarán en detalle una vez se plantee la solución propuesta.

1.4. Estructura de la memoria

Esta memoria se compone esencialmente de cuatro partes, se comienza con un análisis de las tecnologías compatibles, es decir los recursos tecnológicos necesarios para lograr el objetivo planteado, además de técnicas de calibración posibles a implementar. Posteriormente se describirá el camino seguido en el desarrollo de la aplicación que aproxima el objetivo planteado con los recursos disponibles. Finalmente se analizan los resultados y se concluirá en posibles mejoras para el software desarrollado.

2. ESTADO DEL ARTE EN VISIÓN 3D

Para conocer el contexto en que se enmarca este trabajo, es necesario mencionar aquellas tecnologías relacionados a este, como lo son las gafas de realidad virtual o las cámaras de video con sensores de distancia. Adicionalmente se presentan herramientas para el desarrollo de software, bibliotecas y algoritmos que pueden ser utilizados para resolver el problema planteado.

2.1. Tecnologías relacionadas con la realidad virtual

Para el desarrollo de esta memoria hay dos elementos fundamentales, las cámaras de vídeo y los lentes de realidad virtual. Los lentes de realidad virtual se pueden describir como monitores montados en la cabeza, los cuales despliegan imágenes independientes en cada ojo para generar la sensación de profundidad e inmersión. Actualmente existen una gran cantidad de dispositivos disponibles, los cuales tienen campo de visión de alrededor de 100 o 110 grados, en comparación a los 135 grados promedio de un ser humano. Algunos además poseen sensores de posición, seguimiento ocular, entre otros, para generar una mayor inmersión.

A continuación se describirán algunos de los lentes de realidad virtual existentes:

- Google Cardboard[17] es probablemente el más sencillo de las gafas de realidad virtual, ya que consiste en utilizar un soporte de cartón para sostener un smartphone frente a nuestros ojos. Esta idea funciona ya que en los smartphones actuales se encuentran todos los sensores necesarios para permitirnos detectar la posición de la cabeza y poder así disfrutar de una experiencia de realidad virtual.

Fig. 2.1: Google Cardboard

- Oculus Rift[20] es un dispositivo de realidad virtual que cuenta con su propio monitor y sensores, incorpora el uso de lentes para simular un mayor campo de visión. En su primera versión posee una pantalla de 1280 por 800 pixeles, la cual fue aumentada a una pantalla de 1920 por 1080 pixeles en su segunda versión. Su primera versión cuenta con un campo de visión de 110 grados, y una tasa de refresco de 60 Hz. El prototipo Crescent Bay incorpora audio al hardware añadiendo un nuevo sentido a la realidad virtual.

  Como el Oculus Rift hay bastante otros dispositivos, como Sony PlayStation VR, HTC Vive, Samsung Gear VR, Microsoft HoloLens, Avegant Glyph, etc. Los cuales tienen distintas tasas de refresco, o distintas resoluciones, pero no poseen diferencias sustanciales.

Fig. 2.2: Oculus Rift

- FOVE VR[21] aporta un nuevo elemento a la realidad virtual, y esto es el seguimiento de pupila, lo que permite modificar la escena presentada de modo que se genere el enfoque y desenfoque de elementos, dando un paso más hacia el realismo, aumentado la inmersión.

Fig. 2.3: FOVE VR

Para las cámaras de vídeo se busca que además de entregarnos una imagen a color, nos provea de información acerca del entorno. Para conseguir esto es posible utilizar algún sistema que incorpore un mecanismo de detección de profundidad, como lo hacen las cámaras Kinect de Microsoft, las cámaras Xtion PRO de Asus o las cámaras

Senz3D de Creative. Estas cámaras proveen de una imagen a color y una imagen de profundidad provenientes de sensores infrarrojos. Como alternativa a esta combinación[6], es posible utilizar cámaras de alta resolución y cámaras TOF para obtener imágenes de buena resolución y alta tasa de refresco. Lamentablemente la tecnología de las cámaras TOF aún es muy costosa.

2.2. Desarrollos recientes en visión 3D

Un trabajo con varios elementos en común con lo que se intenta desarrollar, pero con una perspectiva distinta, es la Holoportation[15] que desarrolla Microsoft. La cual se describe como un sistema de captura 3D, la cual permitiría generar modelos 3D de personas en alta resolución, para ser comprimidos y transmitido a cualquier parte del mundo en tiempo real. Combinados con los HoloLens[16], permite una experiencia de ver y escuchar al otro participante como si estuvieran en el mismo espacio físico. Este proyecto tiene la diferencia fundamental con el trabajo desarrollado de que su dispositivo final de reproducción es de realidad combinada como ellos la llaman, ya que a diferencia de los lentes de realidad virtual, no teletransportan a otro lugar, ya que son transparentes y puedes seguir viendo tu entorno, pero proyecta elementos adicionales en el lugar donde te encuentras.

Fig. 2.4: Demostración simulada del funcionamiento de la ”holoportacion”[16].

2.3. Tecnologías compatibles con Virtual View

Las cámaras Kinect de Microsoft poseen la habilidad de inclinarse mediante un motor, proveer diversos stream de imágenes, tales como imágenes de color, profundidad o infrarrojo. Pero el modo de conseguir controlarla puede variar dependiendo del API que se utilice. Dependiendo del sistema operativo o lenguaje de programación que se desee utilizar existen varias opciones, algunas de las cuales se describirán a continuación:

- OpenKinect es una biblioteca desarrollada para MacOS la cual fue creada con el objetivo de permitir extraer la información 3D cruda, antes de ser procesada y transformada al sistema métrico, donde es aproximada. Además esta fue creada para la Kinect para xbox, por lo que la API para la versión de Windows de la cámara Kinect no estaba disponible.

- dLibs.freenect es una biblioteca basada en OpenKinect para el sistema operativo Windows. Permite su uso con el lenguaje de programación Java e incorpora una variedad de estructuras de datos para disponer de las imágenes de color y

profundidad. Extiende el control del motor para ajustar la posición de la cámara. Incorpora mecanismos de calibración multi-cámara y mapeo color/profundidad.

- Kinect SDK es la biblioteca oficial de Microsoft para el desarrollo de aplicaciones que utilicen las cámaras Kinect para Microsoft Windows. Esta biblioteca permite obtener los streams de vídeo de cada cámara que componen a la cámara Kinect. Proporciona información acerca de la dirección desde donde proviene el sonido mediante su arreglo de cuatro micrófonos, permite obtener el stream del esqueleto de las personas frente a la cámara, entre muchas otras funciones, posibilitando extraer la información que sus sensores proveen en aplicaciones de alto nivel, donde se utiliza información pre-procesada, o en aplicaciones de bajo nivel, donde se utiliza la información cruda entregada por los sensores.

Para conseguir observar el proceso de la generación de imágenes, y ver resultados intermedios, es necesario realizar una aplicación que permita visualizar lo que ocurre en tiempo real. Es por esto que es necesario desarrollar una aplicación que posea una interfaz de usuario amigable, que permita acceder a los datos en tiempo de ejecución. Es posible encontrar una variedad de API que facilitan el desarrollo de las interfaces gráficas, a continuación se describen algunas de las opciones disponibles:

- Visual Studio es un IDE que posee un compilador necesario para la compilación de la biblioteca propietaria de Kinect. En cuanto a sus capacidades para el desarrollo de interfaces gráficas, posee una gran variedad de elementos gráficos prediseñados, los cuales se pueden ensamblar fácilmente, ya que es posible desarrollar la interfaz de manera gráfica, arrastrando los elementos al lugar deseado, luego solo es necesario realizar unas pocas líneas de código para volverla funcional.

- Gtkmm es una interfaz para C++ de la biblioteca GTK+, dentro de sus principales características destaca un sistema de seguridad para los punteros, de modo que el compilador pueda advertir de punteros erróneos. Posee una gama

de elementos gráficos para utilizar, los cuales pueden ser fácilmente modificados mediante herencia. Adicionalmente es posible utilizar el diseñador Glade, el cual permite desarrollar interfaces gráficas de manera visual.

- Qt es una plataforma ampliamente utiliza, la cual se caracteriza por ser multiplataforma, alcanzando a todos los desarrolladores que desean utilizarla. Al igual que las otras opciones descritas, Qt incorpora una gran cantidad de elementos gráficos prediseñados y fáciles de modificar. Adicionalmente es posible utilizar el IDE Qt Creator, el cual permite desarrollar estas interfaces de manera gráfica, simplemente arrastrando los elementos a las posiciones deseadas.

La calibración de cámaras es un paso esencial en la visión por computadora, especialmente cuando se requieren información métrica. Existen bastantes métodos de calibración de cámaras, los cuales pueden ser clasificados como lineales, no lineales o de dos pasos, cuando se calculan algunos parámetros preliminares utilizando programación lineal, y el resto de los parámetros se calcula de manera iterativa[1].

Algunos de los métodos de calibración más comunes son: el método de Hall[2], el método de Faugeras-Toscani[3], el método de Faugeras-Toscani con distorsión radial[4], el método de Tsai[5], el método de Weng[7], entre otros.

Todos estos métodos se basan en resolver la ecuación del modelo pinhole, considerando diferentes parámetros de distorsión. El modelo pinhole se puede expresar matricialmente como:

$$\begin{pmatrix} s^I X_d \\ s^I Y_d \\ s \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \end{pmatrix} \begin{pmatrix} {}^W X_w \\ {}^W Y_w \\ {}^W Z_w \\ 1 \end{pmatrix}$$

En donde a la derecha de la igualdad, la matriz de 3x4 es una matriz de proyección que multiplica el vector de coordenadas homogéneas de un punto en el mundo real para obtener las coordenadas de aquel punto en coordenadas de la imagen. El uso

de coordenadas homogéneas agrega un elemento adicional al vector de coordenadas espaciales, lo que permite utilizar una matriz proyectiva de 3x4 elementos, la cual integra las tres transformaciones geométricas: rotación, traslación y escala.

Este modelo habitualmente se descompone en cuatro etapas:

- Relacionar el punto en coordenadas del mundo real, a coordenadas de la cámara, utilizando una matriz de rotación y un vector de traslación.

$$\begin{pmatrix} {}^{C}X_w \\ {}^{C}Y_w \\ {}^{C}Z_w \end{pmatrix} = {}^{C}R_W \begin{pmatrix} {}^{W}X_w \\ {}^{W}Y_w \\ {}^{W}Z_w \end{pmatrix} + {}^{C}T_W$$

- El segundo paso es llevar el punto en coordenadas de la cámara al plano de la imagen, utilizando la transformación proyectiva.

$${}^{C}X_u = f\frac{{}^{C}X_w}{{}^{C}Z_W},\ {}^{C}Y_u = f\frac{{}^{C}Y_w}{{}^{C}Z_W}$$

- El tercer paso modela la distorsión del lente, basado en la disparidad de la proyección real. Por lo que el punto del plano de la imagen es trasladado al punto de la proyección real.

$${}^{C}X_u = {}^{C}X_d + \delta_x,\ {}^{C}Y_u = {}^{C}Y_d + \delta_y$$

  Dependiendo de la técnica de calibración estos coeficientes varían. Métodos simples como el de Faugeras y Toscani no modelan la distorsión de los lentes, por lo que se cumple que:

$$\delta_x = 0,\ \delta_y = 0$$

  El método de Faugeras-Toscani puede ser mejorado incorporando un modelo para la distorsión radial. El método de Tsai también incorpora esta distorsión. Esta distorsión puede ser modelada como:

$$\delta_{xr} = k_1 {}^C X_d ({}^C X_d^2 + {}^C Y_d^2),\ \delta_{yr} = k_1 {}^C Y_d ({}^C X_d^2 + {}^C Y_d^2)$$

Modelos más complejos como el de Weng, modelan además de la distorsión radial, una distorsión de descentrado además de una distorsión de prisma delgado. La suma de estas tres distorsiones se puede expresar como:

$$\delta_x = (g_1 + g_3) {}^C X_u^2 + g_4 {}^C X_u {}^C Y_u + g_1 {}^C Y_u^2 + k_1 {}^C X_u ({}^C X_u^2 + {}^C Y_u^2),$$

$$\delta_y = g_2 {}^C X_u^2 + g_3 {}^C X_u {}^C Y_u + (g_2 + g_4) {}^C Y_u^2 + k_1 {}^C Y_u ({}^C X_u^2 + {}^C Y_u^2)$$

- El cuarto y último paso consisten en transformar el punto corregido en el plano de la imagen, desde el sistema de coordenadas métrico al sistema de coordenadas en pixeles.

$$^I X_d = -k_u {}^C X_d + u_0,\ ^I Y_d = -k_v {}^C Y_d + v_0$$

El siguiente diagrama representa los cuatro pasos mencionados.

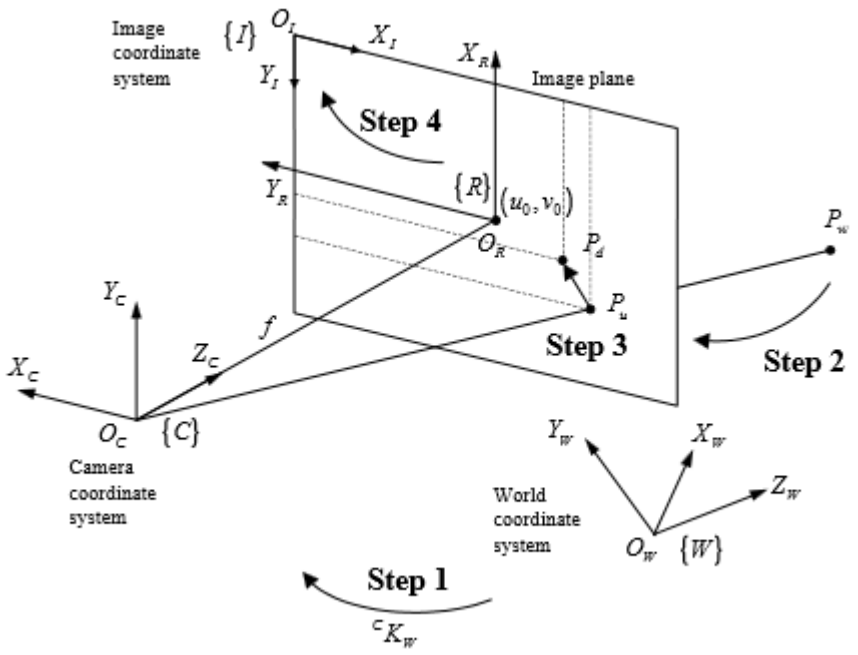

Fig. 2.5: Relación geométrica entre un objeto 3D y su imagen proyectada 2D[1].

Un paso esencial para conseguir una buena virtualizacion, es determinar de manera precisa la posición relativa entre las cámaras, lo que permite unificar la información proveniente de las distintas cámaras de manera correcta. Para poder realizar esta calibración es necesario considerar la geometría epipolar, la cual nos relaciona un punto en una imagen con una recta en la segunda. El siguiente diagrama gráfica esta idea:

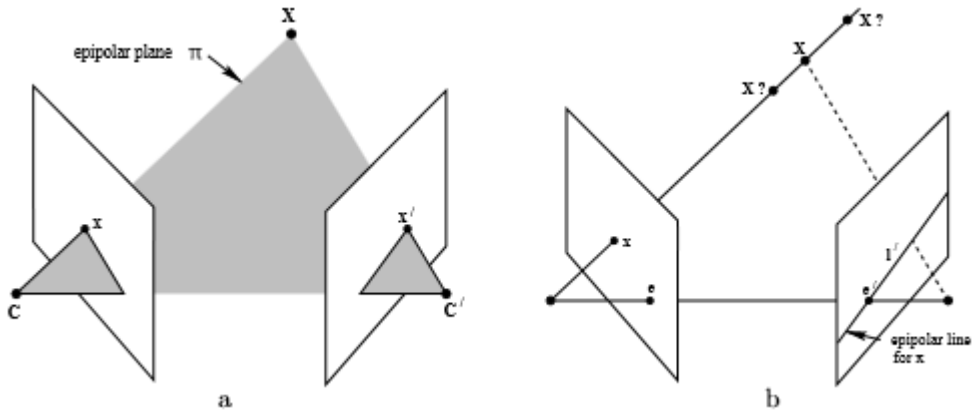

Fig. 2.6: (a) Las dos cámaras están indicadas por sus centros C y C' y los planos de la imagen. Los centros de las cámaras, el punto tridimensional X y sus imágenes x y x' yacen en el mismo plano π. (b) Un punto en la imagen x, se retro-proyecta en un rayo en el espacio tridimensional, definido por el centro de la cámara C y el punto en la imagen x. Este rayo es observado como la línea l' en la segunda vista. El punto tridimensional X que se proyecta en x debe yacer en este rayo, por lo que la imagen de X debe yacer sobre la línea l' en la segunda vista[10].

Encontrar la matriz proyectiva que nos permita relacionar ambas imágenes, es un problema de encontrar los pares de puntos en ellas. Esta matriz proyectiva tiene que tener la particularidad de ser de rango dos, para que las rectas epipolares no se dispersen. Al imponer esta condición, la matriz proyectiva, habitualmente llamada matriz fundamental, queda con siete grados de libertad, por lo que para obtenerla será necesario encontrar al menos siete pares de puntos.

Un aspecto importante a notar en las cámaras Kinect, es el hecho de que existe una separación evidente entra la cámara de color y la cámara de profundidad. Esta separación causa que la información proveniente de la cámara de profundidad no pueda ser mapeada directamente en la imagen de color. Para corregir esta separación física es

necesario realizar una transformación proyectiva entre estas imágenes. Para realizar esta calibración existen dos metodologías estándares que permiten obtener los pares de puntos necesarios para esta. Uno de ellos es utilizar algún patrón de calibración con relieves o perforaciones que permitan identificar fácilmente los puntos en las imágenes de color y de profundidad. Otra alternativa es solicitarle a la cámara un stream infrarrojo en vez del de profundidad.

El método de Zhang[8], utiliza un patrón plano del tipo chessboard pero de un modo creativo, en vez de obtener los puntos de entre las intersecciones de los cuadrados, se calcula un plano estimado para el patrón en la imagen de color, ya que se sabe que este debe ser co-planar con el plano en la imagen de profundidad. Adicionalmente se seleccionan puntos de manera manual entre ambas imágenes para obtener información confiable. Para determinar los parámetros se utiliza el método de maximum likehood.

El método utilizado en 'Depth-camera calibration toolbox' (DCCT)[9], es un método de calibración que utiliza un patrón esférico, y posee tres etapas. La primera etapa consiste en obtener las características de la imagen de color y profundidad, permitiendo identificar el borde de la esfera. El segundo paso consiste en utilizar Direct Linear Transformation (DLT) para obtener una primera solución de la matriz de rotación y el vector de traslación entre las cámaras, además de la matriz de parámetros intrínsecos de la cámara de color. El tercer paso consiste en refinar esta solución utilizando una combinación de funciones ponderadas, las cuales se basan en el error de re-proyección de algunos puntos de la imagen.

Un método descrito por Woonwo Lee en su blog[11], utiliza un patrón de tablero de ajedrez para realizar su calibración, pero a diferencia del método de Zhang, este método utiliza un stream de imágenes infrarroja en vez del de profundidad de la cámara Kinect, lo que permite ver los cuadros del patrón de ajedrez y usar métodos automatizados para la obtención de puntos en ambas imágenes. Ningún método para resolver el sistema de ecuaciones que se genera al utilizar el modelo pinhole es especificado.

Existen muchas más técnicas y métodos para la calibración de las cámaras, pero estas no son realmente el foco de esta memoria. En este trabajo se utilizan algunos de los métodos ya implementados para obtener una solución a este problema, pero no se busca en profundidad como mejorar estos resultados.

Para realizar la calibración de las cámaras existen algunas alternativas ya implementadas y distribuidas como lo son el API de OpenCV[12], el toolbox de calibración de MATLAB[13], 'Integrating Vision Toolkit (IVT)'[14], entre otros.

OpenCV es una biblioteca de código abierto, gratuita tanto para fines comerciales como académico, tiene interfaces en C, C++, Java y Python. Posee soporte para los sistemas operativos Windows, Linux, Mac OS, iOS y Android. La comunidad de personas detrás de esta biblioteca supera las cuarenta y siete mil, por lo que se mantiene en constante desarrollo tanto en el software como en la documentación y tutoriales de la misma.

MATLAB es un software diseñado para científicos e ingenieros, el cual posee herramientas para trabajar en aprendizaje de máquinas, procesamiento de señales, procesamiento de imágenes, visión por computadora, comunicaciones, robótica, ente otros. Utiliza su propio lenguaje de programación, pero permite ser integrado con C, C++, Python y Java. Este software requiere una licencia para ser utilizado con fines comerciales.

Integrating Vision Toolkit (IVT) es una biblioteca de visión por computadora, de código abierto, multiplataforma que posee una arquitectura orientada a objetos, desarrollada en C++, la cual implementa una serie de rutinas de procesamiento de imágenes rápidas, funciones matemáticas y variadas estructuras de datos prácticas para el trabajo de imágenes. Adicionalmente es posible utilizar una interfaz gráfica multiplataforma que tienen disponible.

Dentro de nuestra universidad también es posible encontrar trabajos relacionados en el área de la calibración de cámaras Kinect. Un buen ejemplo de esto es la memoria desarrollada por Luis Fuentes[23], en la cual a través de un objeto de calibración sugerido en ROS[24], logra realizar una calibración estéreo entre un par de cámaras

Kinect, además de corregir la separación entre las cámaras de color y profundidad.

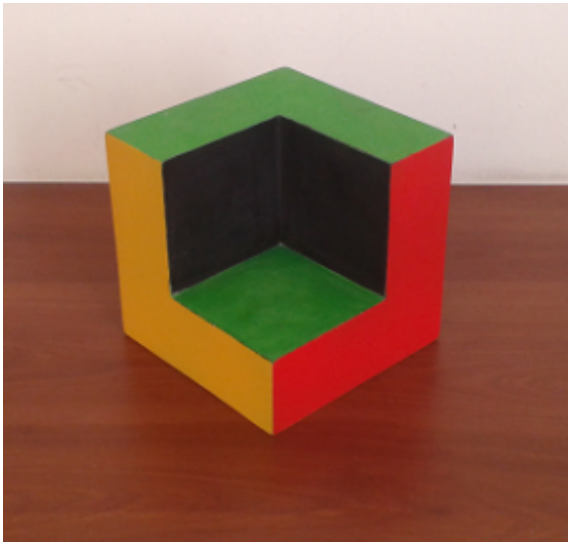

Fig. 2.7: Objeto de calibración utilizado por Luis P. Fuentes W.[23].

La idea de utilizar este patrón de calibración en particular, es que permite encontrar fácilmente las esquinas en la cámara de color, debido al alto contraste del color entre sus caras, además permite ajustar planos a sus caras en la imagen de profundidad, de modo de encontrar esas mismas esquinas en las intersecciones de los planos.

3. ANÁLISIS DEL PROBLEMA Y SOLUCIÓN PROPUESTA

El problema que se enfrenta en esta memoria es el de sintetizar la vista de una cámara virtual, el cual nace como una potencial idea de negocio, debido a la paulatina, pero constante expansión del mercado de dispositivos de realidad virtual, y del software disponible para estos.

Solucionar el problema permitiría posicionar a los dispositivos de realidad virtual, como los dispositivos predilectos para visualizar el contenido multimedia que consumimos cotidianamente, como por ejemplo los noticiarios, eventos deportivos, producciones dramáticas, entre otras.

3.1. Definición de requerimientos

Para realizar el desarrollo de un software es necesario comenzar evaluando las características que se necesitan que este cumpla. Con el objetivo en mente de lograr una vista virtual en un espacio real, en el cual existan varias cámaras captando lo que ocurre en ella, es necesario que el software sea capaz de componer imágenes utilizando la información proveniente de más de una cámara. Esta idea entrega una buena base para comenzar, pero es necesario desagregar esta idea para comenzar a llevarlo a un software realizable.

Como se desea generar una vista virtual utilizando información proveniente de múltiples cámaras, el problema se puede acotar a solo dos cámaras, y conseguir realizar un travelling entre estas dos imágenes reales. Evaluando este problema de manera técnica es posible proponer utilizar geometría epipolar para obtener información tridimensional de la escena, realizar un modelado 3D de esta y finalmente realizar proyec-

ciones en diversos ángulos para obtener una solución al problema. Para lograr que esta solución funcione es necesario conocer la posición relativa entre ambas cámaras, por lo que sería necesario que el software sea capaz de obtener esta información mediante algún proceso de calibración.

En la universidad se cuenta con la posibilidad de utilizar cámaras Kinect, las cuales proveen una imagen de color y otra de profundidad, la cual nos entrega una información tridimensional de la escena, lo que simplifica el modelar la escena en 3D. Debido al uso de estas cámaras es necesario que el software sea capaz de configurar las cámaras Kinect para obtener las imágenes de color y profundidad en las resoluciones deseadas.

Adicionalmente se desea que este software genere estas imágenes a tiempo real, de modo de presenciar eventos deportivos en vivo, por lo que se decidió no realizar levantamientos 3D, los cuales requieren grandes estructuras de datos, y trabajar exclusivamente con imágenes. Es necesario que el software sea capaz de componer imágenes sin recurrir a API's que trabajen los datos en 3D, como OpenGL.

Para evaluar la solución, se desea que el software permita visualizar los resultados obtenidos, por lo que deberá tener una interfaz gráfica que permita ver las imágenes obtenidas desde las cámaras, las imágenes generadas, además de permitirnos 'desplazarnos' entre estas dos cámaras. Adicionalmente se deseará poder medir parámetros de desempeño, tales como tasas de error o cuadros por segundo generados.

Para realizar una evaluación más extensiva del resultado se decide probar en algún dispositivo de realidad virtual las imágenes obtenidas. En la universidad se dispone de las gafas de realidad virtual Oculus Rift, por lo que se desea que el software pueda desplegar las imágenes obtenidas en estas gafas de realidad virtual.

3.2. Solución Propuesta

Para este trabajo se dispuso de dos cámaras Kinect, en su primera versión para el sistema operativo Windows, por lo que el software desarrollado solo permite el uso de

dos cámaras, pero la mayoría de los algoritmos utilizados son fácilmente extensibles a más cámaras. Para la proyección de las imágenes generadas se utilizaron las gafas de realidad virtual Oculus Rift, las cuales son de primera generación.

En la solución propuesta, el primer paso realizado es el de calcular las posiciones relativas entre dos cámaras Kinect utilizando un patrón de calibración con forma de tablero de ajedrez. El uso del patrón de tablero de ajedrez se debe al alto contraste entre los cuadrados que lo componen, lo que facilita la detección de las esquinas. Para realizar esta detección se utiliza la biblioteca OpenCV, la cual implementa un algoritmo que explora los bordes, uniendo los contornos y descartando figuras que no tengan cuatro bordes o posean un perímetro muy pequeño. Posteriormente se realizan conexiones, se buscan cuadriláteros que estén adyacentes, y se ordenan de modo de empezar a formar el patrón de calibración. La información de los bloques que componen el patrón se almacena en forma de grafos, para posteriormente calcular el grafo dual y obtener las esquinas.

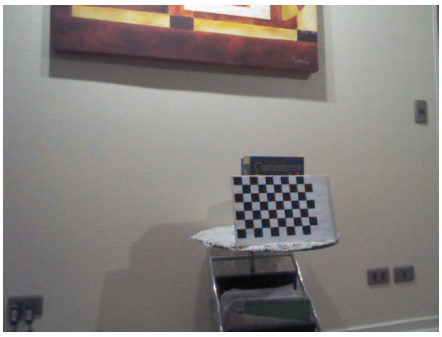

Fig. 3.1: Vista desde la izquierda del patron de calibración.

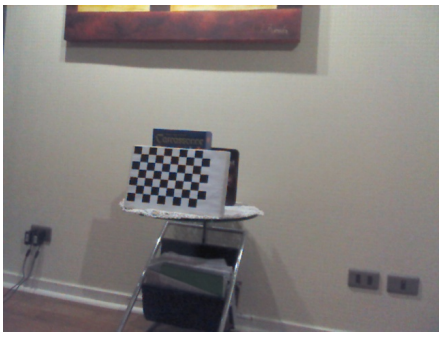

Fig. 3.2: Vista desde la derecha del patron de calibración.

El número de imágenes a utilizar para realizar la calibración depende de diversos factores, tales como la resolución de las cámaras, la distribución del patrón en las imágenes, la inicialización de parámetros, entre otros. Es por esto que para cada situación es mejor realizar pruebas y determinar de manera empírica el número de imágenes a utilizar, y verificar en base al error de reproyección el número de pixeles de error aceptable, siempre siendo cuidadoso de no sobre entrenar. Una vez encontrados

los puntos en ambas imágenes es posible realizar una calibración estéreo, utilizando como base el modelo ‘pinhole’ explicado previamente.

Con este modelo en mente sabemos que piezas son las que debemos encontrar para lograr establecer la posición relativa entre las cámaras y lograr nuestro par estéreo. La estrategia es usar los puntos del patrón de calibración en la imagen, asignarles un sistema de coordenadas en el mundo real, y usar estos para generar el sistema de ecuaciones que permite resolver la proyección mediante alguno de los métodos mencionados en el estado del arte. Finalmente se realiza una descomposición para obtener los parámetros intrínsecos y extrínsecos. Los parámetros intrínsecos corresponden a las características propias de la cámara, estos son las distancias focales expresadas en pixeles, las coordenadas del punto principal, generalmente el centro de la imagen, y el factor de skew, que representa que tan cuadrado es un pixel. Los parámetros extrínsecos corresponden a una matriz que describe la posición de la cámara respecto a una escena estática.

Una vez calibradas las cámaras de color, aún nos falta realizar una corrección a las imágenes de profundidad. Pero el API de Microsoft Kinect provee un método para realizar esta corrección, la cual corresponde a una trasformación proyectiva, por lo que se decidió utilizar esta vía por simplicidad. Tal transformación depende exclusivamente de los parámetros intrínsecos de la cámara y no de la escena, por lo que la librería provee una tabla con los valores de las nuevas coordenadas de los pixeles de color en base a los valores obtenidos en la imagen de profundidad.

En este punto es posible realizar las proyecciones de las imágenes provenientes de ambas cámaras a una nueva vista intermedia. Al realizarse está proyección existen regiones en donde ambas cámaras tienen información y su proyección se traslapará, por lo que es necesario determinar un modo de decidir qué información es la que se utilizará. Pensando en una extensión del trabajo, utilizando más de dos cámaras, se utilizaron dos criterios para determinar qué información utilizar. El primero es distancia, en el sentido de que tan lejana está la cámara real de la cámara virtual, y el segundo es que tan ‘frontal’ la cámara real se encuentra de la cámara virtual. El criterio de

distancia está pensado en términos de resolución, de que tanto detalle puede proveer esa cámara, y el criterio de orientación es para conocer qué tan útil es la información que dispone la cámara.

Para obtener la distancia se utiliza el concepto de distancia euclidiana en base a las coordenadas obtenidas en la calibración estéreo.

En cuanto a determinar qué tan frontal es una cámara con respecto a otra, se utiliza una medida de distancia, calculando la norma de la multiplicación entre la matriz de rotación de la cámara virtual transpuesta y cada una de las matrices de rotación de las cámaras reales, luego mientras más pequeña la norma resultante más 'frontal' será la cámara.

Con estos dos parámetros de combinación en mente es posible realizar varios algoritmos de combinación, tales como combinación por promedios ponderados, combinación binaria por distancia o la cámara más frontal, etc. Se decide utilizar promedios ponderados con el objetivo de mantener la esencia de la escena en términos de color, ya que puede ser que una luz esté frontal hacia una cámara, saturando la imagen, y se desea que cuando se acerque virtualmente a esta cámara se empiece a ver gradualmente esta misma saturación.

A modo de ejemplo se muestra a continuación el proceso más simple de composición de imágenes, utilizando un promedio simple entre las imágenes reproyectadas.

Fig. 3.3: Imágenes obtenidas desde las cámaras estáticas.

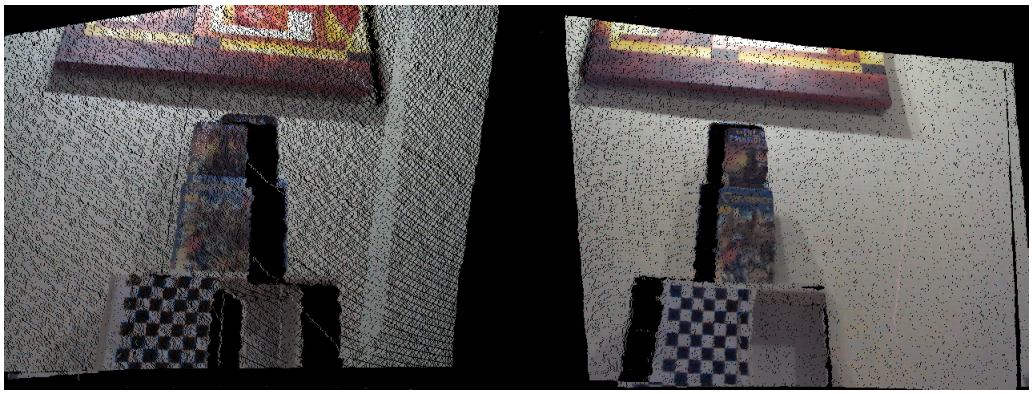

Fig. 3.4: Imágenes reproyectadas a una posición intermedia entre las cámaras estáticas.

Las imágenes reproyectas fueron obtenidas ocluyendo el sensor de profundidad de la cámara opuesta, ya que cualquier información inválida de la imagen de profundidad es descartada por el software, permitiendo ver el aporte de cada cámara de manera independiente.

Fig. 3.5: Composición por promedio simple de imágenes reproyectadas.

El resultado mostrado en la figura 3.5, permite comprender el proceso de composición de imágenes de manera sencilla, ya que se alejó la cámara virtual para mostrar de mejor manera el aporte proveniente de cada una de las cámaras, pero no representa de manera consistente los resultados obtenidos ya que la calibración fue realiza de manera simple y el método de composición no fue el más refinado. Posteriormente se

evaluara de manera más extensa los resultados obtenidos.

Realizados los procedimientos anteriores es posible obtener dos imágenes, que posean la separación intrapupilar de un ser humano. Esta separación es un valor bastante arbitrario debido a la gran varianza que esta variable posee[22], y debe ser modificada para cada usuario para obtener mejores resultados. Finalmente se despliegan estas imágenes en el dispositivo Oculus Rift, para lo cual se creó una escena en openGL con una muralla que ocupa todo el campo visual, y en esta muralla se coloca la textura de la imagen correspondiente al renderizar la escena para cada ojo.

Fig. 3.6: Demostración del funcionamiento del software que despliega las imágenes

En la figura anterior3.6 se muestra el funcionamiento del software que despliega las imágenes en las gafas de realidad virtual, en la cual de manera de ejemplo se muestran dos imágenes completamente distintas.

En cuanto a la interfaz gráfica desarrollada para desplegar resultados intermedios en el proceso de generación de las vistas de las cámaras virtuales, se desarrolló una interfaz sencilla, la cual despliega las imágenes de color provenientes de las cámaras Kinect, informa acerca del número de pares de frame obtenidos (color y profundidad) provenientes desde las cámaras kinect, permite iniciar el proceso de calibración, desplegar una vista de las cámaras virtuales y modificar las posiciones de estas.

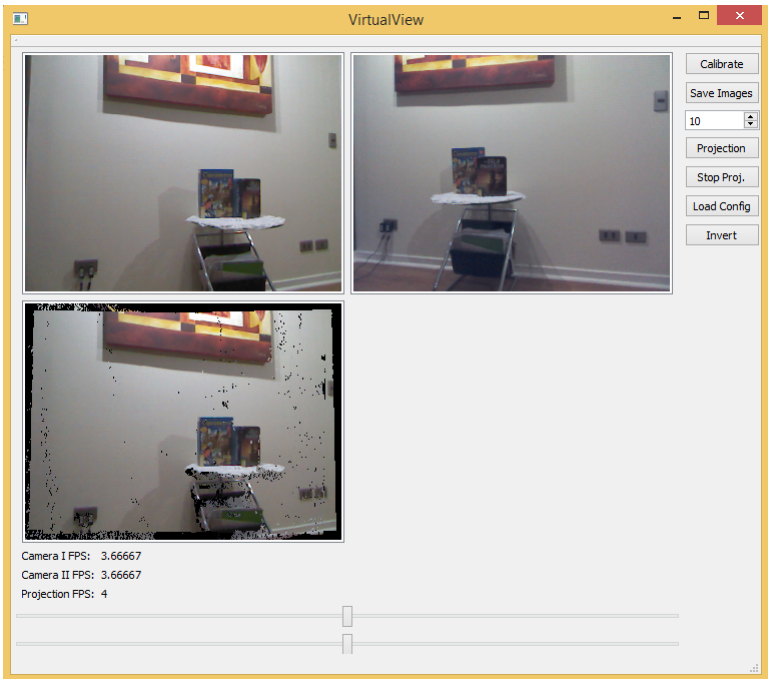

Fig. 3.7: Interfaz gráfica desarrollada

4. EVALUACIONES DE VIRTUAL VIEW

Para evaluar la solución propuesta se analiza el desempeño de cada una de las partes que lo componen. En general no fue sencillo encontrar métodos cuantitativos para evaluar el desempeño de la aplicación, pero en base a imágenes y observaciones objetivas se procede a mostrar su rendimiento.

4.1. Calibración de parámetros de las cámaras Kinect

Para la calibración de parámetros intrínsecos y extrínsecos es posible calcular el error de reproyección, el cual corresponde a la raíz cuadrada media de las diferencias entre los pixeles obtenidos en las imágenes reales y la reproyección con los puntos del objeto real. Hay dos variables importantes que hace que los resultados obtenidos varíen de manera considerable, el primero de ellos es el número de imágenes a utilizar para la calibración, el segundo factor relevante es el tamaño del patrón de calibración, el cual tiene un efecto similar al de variar el número de imágenes ya que modifica el número de puntos utilizados para la calibración, pero además fuerza a utilizar patrones de mayor tamaño, lo que produce mejores resultados ya que los puntos están mejor distribuidos en cada imagen. Esta variable no fue evaluada experimentalmente debido a la complejidad de elaborar diferentes patrones de calibración. A continuación se presentan algunos valores experimentales del error de reproyección al variar el número de imágenes con las que se calibraron las cámaras.

Nº de imágenes	Error de reproyección (pixeles)
1	0.199
3	0.504
5	0.582
7	0.380
10	0.687

Tab. 4.1: Error de reproyección para calibraciones con distintos número de imágenes.

Estos resultados nos muestran que la calibración es suficientemente buena en la vecindad del patrón de calibración, independiente del número de imágenes utilizadas para este proceso, y a medida que se utilizan más imágenes el error tiende a aumentar, lo cual es coherente con la dificultad de ajustar el modelo con puntos más distribuidos en las imágenes. En cuanto al resultado obtenido con 7 imágenes, este puede indicar que en las imágenes utilizadas eran muy similares, es decir que el patrón de calibración se encontraba en posiciones muy cercanas entre ellas.

4.2. Mejoras a las imágenes de profundidad

La evaluación de los resultados de mapear las imágenes de color a las de profundidad es difícil de realizar de manera cuantitativa, por lo que para su evaluación se decidió utilizar una segmentación en la imagen de color en base a la imagen de profundidad, en donde se vuelve evidente los resultados.

Fig. 4.1: Segmentación realizada con la imagen de profundidad sin corrección.

Fig. 4.2: Segmentación realizada con la imagen de profundidad corregida.

Como se observa al comparar las imágenes 4.1 y 4.2, es evidente la mejora, ya que al menos el contorno del lado derecho de la segmentación se ajusta bastante bien a la mesa y los objetos sobre esta, no así el lado izquierdo, lo cual es un tanto más difuso debido a la posición de los sensores en la cámara. Es posible observar adicionalmente una serie de manchas al lado izquierdo de ambas imágenes, que corresponde a la interferencia que produce en el sensor se profundidad al utilizar más de una cámara Kinect.

4.3. Resultados de las vistas generadas por las cámaras virtuales

Para evaluar de mejor manera el resultado de la composición de imágenes, es necesario comenzar evaluando el resultado de las reproyecciones individuales de cada cámara. Un ejemplo de estas imágenes se muestran en la figura 3.4, en las cuales es posible notar una gran cantidad de pixeles en negro, correspondientes al estiramiento de las superficies. Esta característica genera que se difuminen los objetos más cercanos a las cámaras, ya que los pixeles del fondo se cuelan entre estos espacios. En general cuando la distancia entre la cámara estática y la cámara virtual aumenta, el número de pixeles en negro también lo hace. Cabe destacar que el resultado de estas reproyecciones individuales es mejores que las que se usan durante la ejecución del software VirtualView, ya que fueron obtenidas utilizando una sola cámara Kinect a la vez, por lo que no presentan secciones con interferencia de los emisores infrarrojos.

En cuanto a la etapa de composición de imágenes, se evaluó inicialmente el uso de 'la cámara más frontal' para determinar la cámara que posee más información de lo que ve la cámara virtual. Esta información se utilizó de manera binaria, de modo que la cámara más frontal aporta toda la información que posee a la imagen generada, para posteriormente permitir a la segunda cámara real rellenar los espacios donde la primera cámara no tuvo nada que aportar. Un ejemplo de esto se presenta en las figuras 7.3, 7.4, 7.5, disponibles en los anexos, las cuales corresponden a una secuencia de imágenes que simulan un travelling entre las cámaras estáticas, siendo la primera y la última imágenes de la serie las reproyecciones de los lugares que se encuentran las cámaras estáticas. Para estos ejemplos las vistas de las cámaras estáticas son las mostradas en la figura 7.1. Al observar en detalle la secuencia de imágenes es posible notar lo susceptible que es a los errores o vacíos en la imagen de profundidad, ya que sobresalen los pixeles de fondo, y terminan desapareciendo las superficies más cercanas a la cámara. Es posible notar que el algoritmo para determinar la cámara más frontal funciona, ya que es posible notar el salto que se produce entre la quinta y la sexta imagen de la secuencia7.3, 7.4, 7.5, en términos de la información de los elementos sobre la mesa. En general los resultados al utilizar este método no son buenos ya que se desperdicia información al permitir a la segunda cámara solo actualizar pixeles que no fueron tocados por la primera cámara.

El segundo método evaluado en la composición de imágenes, y el que dio mejores resultados con el procesamiento realizado, es el de combinación mediante promedios ponderados, dependiente de qué tan similar sea el ángulo entre la cámara estática y la virtual. Un ejemplo de esto se presenta en la secuencia de imágenes de las figuras 7.6, 7.7y 7.8, disponible en los anexos, las vistas de las cámaras reales para esta secuencia se pueden observar en la figura 7.2. En esta segunda secuencia es posible notar una mejora con respecto a la primera ya que se ve mucho más fluida en términos de como ocurre la transición, y a pesar de que la calibración es defectuosa como en la primera secuencia, es menos notoria su influencia debido a la ponderación del color por sobre la pared que es más clara. Esta ponderación permite además eliminar parcialmente las

manchas producidas por las discrepancias entre las imágenes de color y profundidad, ya que atenúa los pixeles mal 'segmentados' al darle más peso a la cámara que los ve mejor.

4.4. Rendimientos a 'tiempo real'

Un aspecto importante para el éxito de esta tecnología, es el hecho de que pueda ser utilizada a tiempo real, por lo que es necesario evaluar qué tan bien se desempeña en este aspecto. El computador utilizado para realizar las pruebas de rendimiento posee un procesador Intel i5-3337U y posee 8[GB] de memoria RAM. Debido a que no se utilizaron hebras de procesamiento en la tarjeta de vídeo, sus especificaciones no son relevantes. El principal modo de evaluar el desempeño de la aplicación se basa en el número de frames obtenidos después de cada etapa de procesamiento. Las cámaras Kinect entregan 30 fps tanto de sus imágenes de color y profundidad. Tras aplicar la corrección a las imágenes de profundidad para que se ajusten a las imágenes de color se obtienen 8 fps, y el uso de cpu alcanza el 80 %. Una vez que se comienzan a realizar las proyecciones de las dos imágenes de las cámaras virtuales que se componen, el uso de cpu no aumenta considerablemente, pero se obtienen solo 4 fps en promedio. Cabe destacar que a pesar de que se diseñó una interfaz gráfica para desplegar las imágenes obtenidas, no es posible desplegar las imágenes provenientes de las cámaras y mantener la interfaz funcional, por lo que cuando la cpu alcanza niveles de alto uso se prioriza el procesamiento de las imágenes a su despliegue en la interfaz, por lo que algunas imágenes dejan de desplegarse. Si se fuerza a mostrar todas las imágenes el uso de cpu alcanza 100 % y la interfaz gráfica deja de responder.

4.5. Despliegue de imágenes en el Oculus Rift

Con respecto a la proyección final en los lentes de realidad virtual, es difícil demostrar su funcionamiento sin utilizar los lentes, pero es posible mostrar cómo se

proyectan las imágenes, y comentar los resultados obtenidos.

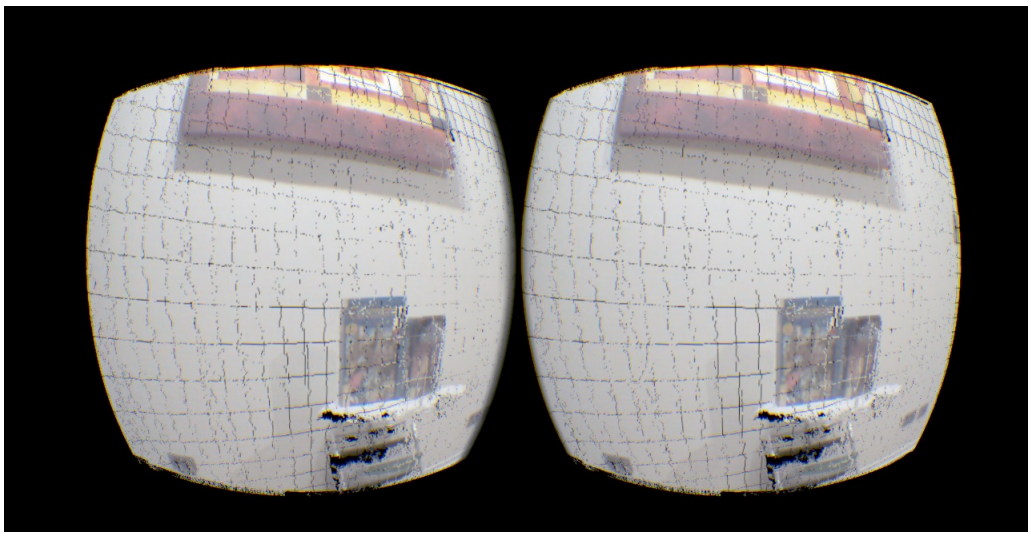

Fig. 4.3: Proyección de imágenes en el Oculus Rift.

En esta imagen se muestra que se consigue desplegar las imágenes generadas en las gafas de realidad virtual, pero debido a la baja tasa de refresco se decidió no implementar el uso de los sensores de posición de las gafas de realidad virtual para modificar las imágenes, es decir, variar la posición de la cámara virtual. Por lo que se lograr observar la escena, tener una sensación de tridimensionalidad debido a las dos proyecciones, pero se siente como observar a través de una ventana, debido a que se observa una pared estática con una 'pintura' que cambia.

5. REFLEXIONES FINALES

5.1. Conclusiones

Durante el desarrollo de este trabajo, y a medida de que se obtenían los primeros resultados, fue posible notar una serie de problemas del ámbito de la ejecución, que no fueron previstos cuando se analizó el problema inicialmente, o se desconocía su magnitud e importancia que tendrían durante la implementación de la solución. Uno de los mayores problemas enfrentados fue el funcionamiento del sensor de profundidad de las cámaras Kinect, los cuales tienen problemas con la luz natural, generando zonas sin información, adicionalmente la disposición espacial entre el emisor y receptor infrarrojo genera sombras en un costado de los objetos presentes en las imágenes de color. Estos sensores poseen un rango acotado de trabajo para las imágenes de profundidad, lo que dificultó generar escenas con las que trabajar, que compatibilizaran de buena forma la información espacial y la cromática. Estos problemas son los que presentan cada sensor, pero además, al utilizar más de una cámara Kinect estas interfieren entre sí, generando manchas relativamente aleatorias en la imagen donde no existe información de profundidad. Los problemas de fallan en las imágenes de profundidad pueden ser solucionados parcialmente mediante la implementación de un buffer para recuperar valores de profundidad cuando se produzcan interferencias, pero debido al alto costo en tiempo de procesamiento que la aplicación VirtualView ya posee, agregar un ciclo de revisión en busca de valores erróneos no aporta demasiado, y es preferible esperar a que sea resuelto mediante la mejora del hardware. Algunas soluciones, como la posibilidad de apagar el sensor infrarrojo de la cámara que no está capturando, ya han sido implementadas en versiones posteriores de las cámaras

Kinect.

La solución implementada trabaja las imágenes de manera muy similar a lo que sería realizar un levantamiento 3D, por lo que no utilizar una librería gráfica como lo sería OpenGl terminó complicando la solución, ya que finalmente igual se requirió para desplegar las imágenes en los lentes de realidad virtual.

Los resultados obtenidos en la composición de imágenes no fueron los mejores debido a que a pesar de que la calibración es buena en las secciones donde las vistas de las cámaras se interceptan, hacia los costados es solo extrapolación, ya que no existieron datos para realizar la calibración.

Hubiese sido recomendable realizar calibraciones independientes para cada cámara, obtener sus parámetros, y posteriormente realizar la calibración que permitiese determinar las posiciones relativas entre las cámaras.

A pesar de las falencias en el trabajo, se logró concretar gran parte de los objetivos inicialmente planteado, y aun que el resultado final está lejos de la idea inicial, fue posible entender los desafíos en términos tecnológicos necesarios para poder conseguir los resultados deseados.

5.2. Trabajos futuros

Un punto débil de este trabajo fue la calidad de las imágenes compuestas, las cuales pueden ser mejoradas realizando algunas sencillas mejoras, como lo sería realizar modelos de las superficies, como por ejemplo paredes, para poder interpolar la información de color de manera más precisa. Existen muchas posibilidades de mejora en términos de la composición final, debido a que no se realizó ningún post-procesamiento para mejorar las imágenes, por lo que es muy común ver líneas negras en las vistas de las cámaras virtuales, debido a que se producen vacíos de información al re-proyectar las imagen en vistas más cercanas que las cámaras reales, lo que se podría solucionar repitiendo información cercana, o interpolando pixeles de color promediando vecindades con filtros gaussianos. Dado que este problema se parece

bastante al ruido ‘sal y pimienta’ se puede aplicar un filtro de medianas para rellenar los espacios sin información.

Debida a la alta demanda del uso del procesador, sería recomendable delegar procesamiento a la tarjeta de vídeo, por lo que llevar las hebras encargadas de las reproyecciones a esta mejoraría el rendimiento del software. Adicionalmente sería recomendable eliminar la interfaz gráfica, ya que su función era evaluar las etapas intermedias, pero una vez que se lograron los resultados deseados, solo genera una recarga adicional al procesador.

Para lograr una aplicación más refinada sería recomendable realizar una etapa intermedia de pruebas con imágenes sintéticas, que permitiese evaluar los mejores métodos de composición de imágenes sin tener que lidiar con todos los problemas que acarrea trabajar con las imágenes capturas, como lo son las inconsistencias en las imágenes de profundidad, distorsiones en base a los lentes, entre otros.

Bibliografía

[1] Joaquim Salvi, Xavier Armangue, Joan Batlle. *A comparative review of camera calibrating methods with accuracy evaluation*. Pattern Recognition journal Vol.35 Pag.1617-1635, 2002.

[2] E.L. Hall, J.B.K. Tio, C.A. McPherson, F.A. Sadjadi. *Measuring curved surfaces for robot vision*.Comput. J. Vol.15 Pag.42-54, 1982.

[3] O.D. Faugeras, G. Toscani. *The calibration problem for stereo*.Proceedings of the IEEE Computer Vision and Pattern Recognition, pp.15-20, 1986.

[4] J. Salvi, J. Battle, E. Mouaddib. *A robust-coded pattern projection for dynamic 3D scene measurement*.Int. J. Pattern Recognition Lett. 19, 1055-1065, 1998.

[5] R.Y. Tsai. *A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the shelf TV cameras and lenses*.IEEE Int. J. Robot .Automat. RA-3, 323-344, 1987.

[6] Yan Cui, Sebastian Schuon, Derek Chan, Sebastian Thrun, Christian Theobalt. *3D Shape Scanning with a Time-of-Flight Camera*.Computer Vision and Pattern Recognition (CVPR), 1173 - 1180, 2010.

[7] J. Weng, P. Cohen, M. Herniou. *Camera calibration with distortion models and accuracy evaluation*.IEEE Transactions on pattern analysis and machine intelligence. Vol.14, NO. 10, 1992.

[8] C. Zhang and Z. Zhang. *Calibration between Depth and Color Sensors for Com-*

modity Depth Cameras.In Intl. Workshop on Hot Topics in 3D, in conjunction with ICME, Barcelona, Spain, 2011.

[9] A. Staranowicz, F. Morbidi, and G.L. Mariottini. *Depth-camera calibration toolbox (dcct): accurate, robust, and practical calibration of depth cameras*.In Proc. of the Brit. Mach. Vision Conf., 2012.

[10] Richard Hartley, Andrew Zisserman. *Multiple View Geometry in Computer Vision*.Cambridge University Press, 2000.

[11] Everyone has a BLOG: Kinect Color - Detph Camera Calibration,
`http://cv4mar.blogspot.cl/2011/03/kinect-color-...`
`...detph-camera-calibration.html`

[12] Open Source Computer Vision,
`http://opencv.org/`

[13] MathWorks: Computer Vision System Toolbox,
`http://www.mathworks.com/help/vision/index.html`

[14] IVT: Integrating Vision Toolkit,
`http://ivt.sourceforge.net/`

[15] Holoportation,
`https://www.microsoft.com/en-us/research/...`
`...project/holoportation-3/`

[16] Hololens,
`https://www.microsoft.com/microsoft-hololens/en-us`

[17] Google Cardboard,
`https://vr.google.com/cardboard/`

[18] Vive,
`https://www.htcvive.com/us/`

[19] Playstation VR,
https://www.playstation.com/es-es/explore/playstation-vr/

[20] Oculus Rift,
https://www3.oculus.com/en-us/rift/

[21] FOVE VR,
http://www.getfove.com/

[22] Neil A. Dodgson. *Variation and extrema of human interpupillary distance*. SPIE Proceedings Vol. 5291, pp. 36-46, 2004.

[23] Luis P. Fuentes W. *Virtualización tridimensional de un ambiente en tiempo real*. UTFSM, 2014.

[24] ROS: Robot Operating System,
http://www.ros.org/

7. ANEXOS

Fig. 7.1: Vistas de las cámaras estáticas con las que se evalúa la combinación de imágenes con el método binario dependiente de la cámara más frontal.

Fig. 7.2: Vistas de las cámaras estáticas con las que se evalúa la combinación de imágenes con el método de promedio ponderado dependiendo del ángulo de visión.

Fig. 7.3: Secuencia de imágenes perteneciente al recorrido entre las cámaras estáticas, método binario - parte 1.

Fig. 7.4: Secuencia de imágenes perteneciente al recorrido entre las cámaras estáticas, método binario - parte 2.

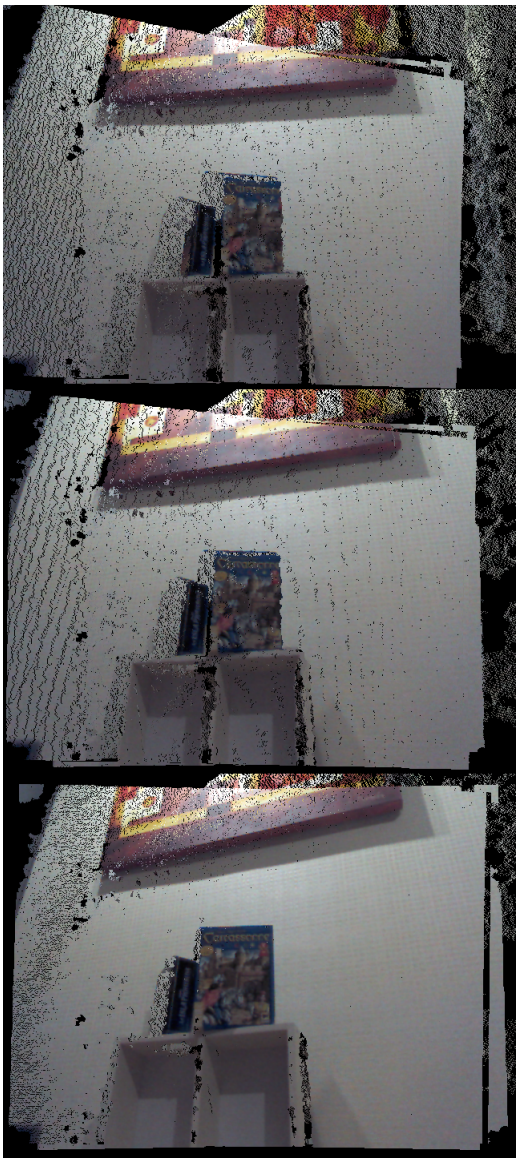

Fig. 7.5: Secuencia de imágenes perteneciente al recorrido entre las cámaras estáticas, método binario - parte 3.

Fig. 7.6: Secuencia de imágenes perteneciente al recorrido entre las cámaras estáticas, método promedio ponderado - parte 1.

Fig. 7.7: Secuencia de imágenes perteneciente al recorrido entre las cámaras estáticas, método promedio ponderado - parte 2.

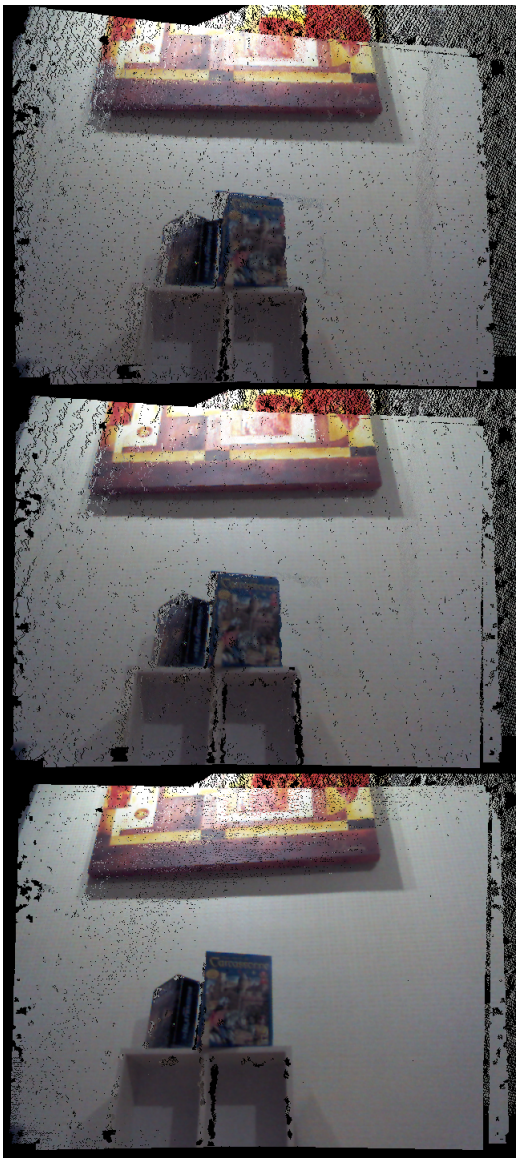

Fig. 7.8: Secuencia de imágenes perteneciente al recorrido entre las cámaras estáticas, método promedio ponderado - parte 3.