Evaluación teórico-experimental de los dispositivos de sensado 3D basados en la tecnología PrimeSense para su aplicación en entornos médicos e industriales

Jose Fernando Florez Gomez

Universidad Industrial de Santander
Facultad de Ciencias
Escuela de Física
Bucaramanga
2017

Evaluación teórico-experimental de los dispositivos de sensado 3D basados en la tecnología PrimeSense para su aplicación en entornos médicos e industriales

Jose Fernando Florez Gomez

Propuesta de Trabajo de grado para optar al título de Físico

Director
Dr. Jaime Enrique Meneses Fonseca
Ph.D. Ciencias para la Ingeniería

Universidad Industrial de Santander
Facultad de Ciencias
Escuela de Física
Bucaramanga
2017

Agradecimientos

A Cristobal Flórez y Ruth Gómez, dos imprescindibles.

Al Dr. Jaime Meneses cuya pasión por la ciencia le ha permitido mostrarle un futuro prometedor a cientos de sus estudiantes.

Hay hombres que luchan un día y son buenos.
Hay otros que luchan un año y son mejores.
Hay quienes luchan muchos años, y son muy buenos.
Pero hay los que luchan toda la vida, esos son los imprescindibles.

Bertolt Brecht

Índice general

Índice de tablas

Índice de figuras

RESUMEN

TÍTULO: Evaluación teórico-experimental de los dispositivos de sensado 3D basados en la tecnología PrimeSense para su aplicación en entornos médicos e industriales[1]

AUTOR: Flórez Gómez, José Fernando[2]

Palabras claves: Kinect, RGBDemo, Reconstrucción tridimensional.

Descripción:

El propósito de este trabajo fue cuantificar el error en su primera aproximación cometido en una reconstrucción tridimensional usando el dispositivo Kinect en un ambiente controlado dentro de su campo de observación. Se tomaron datos desde diferentes perspectivas y se utilizó el código de RGBDemo para acceder a los datos arrojados por el dispositivo Kinect, así como también se utilizó el software Skanect para convalidar los datos tomados. En general, se encontró que para distancias mayores el objetivo presentaba una distribución de error mayor, mientras que para distancias cercanas las reconstrucciones se acercaban más a la realidad, dicho error puede aumentar hasta 10 veces cuando se trabaja en el límite de su rango de trabajo. También se pudo encontrar un error considerable en la distribución longitudinal de los datos tomados usando un código que nos permitió evaluar la menor distancia con sus vecinos (puntos) más cercanos. Esta distribución espacial de los puntos es debido a la no regularidad de los puntos de speckle proyectados por el sensor. También se demuestra la presencia de una discretización constante en los datos de profundidad, dicho valor de discretización fue calculado. Además se pudo combinar nubes de puntos desde diferentes perspectivas usando el Algoritmo ICP con el fin de obtener un render ideal. Bajo estas consideraciones se caracteriza al Kinect dentro de los dispositivos no aptos para analizar detalles inferiores a 2 [mm], luego este puede ser usado en ambientes dónde la resolución demandada no sea superior.

[1] Trabajo de grado

[2] Facultad de Ciencias, Escuela de Física, Jaime Enrique Meneses (Director).

ABSTRACT

TITLE: Theoretical and experimental evaluation of 3D sensing dispositives based on the technology PrimeSense for its application in medical and industrial fields.[3]

AUTHOR: Flórez Gómez, José Fernando[4]

Keywords: Kinect, RGBDemo, tridimensional reconstruction.

Description:

The purpose of this work was to quantify the error in its first approximation made in a tridimensional reconstruction using the Kinect dispositive in a controlled environment inside its observation field. The data was taken from different perspectives and the RGBDemo code was used to access to the data given by the Kinect dispositive as well as the software Skanect to validate the data taken. In general, it was found that for bigger distances the target presented a greater error distribution, whereas for nearby distances the reconstructions were closer to reality, this error may increase up to 10 times when working in the limit of its range of work. It was also possible to find a considerable error in the longitudinal distribution of the taken data using a code that allowed us to evaluate the minor distances in between nearby points. This spatial distribution of points is due to the non-regularity of the speckle points projected by the sensor. Also the presence of a constant discretization in the depth data is demonstrated, that value of discretization was calculated. It was also possible to combine point clouds from different perspectives using the ICP algorithm in order to obtain an ideal rendering.Under these considerations, the Kinect is considered as a dispositive unable to analyze details inferior to 2 [mm], then this may be used in environments where the demanded resolution is greater.

[3]Bacherlor thesis

[4]Faculty of Sciencies. School of Physics. PhD. Jaime Enrique Meneses (Advisor).

Introducción

Investigaciones recientes han mostrado la gran capacidad que tiene el dispositivo Kinect en el campo de la reconstrucción tridimensional aumentando su capacidad resolutiva hasta 1000 veces. Investigadores del Instituto Tecnológico de Massachusetts (MIT) han mostrado que tomando un sensor de baja calidad, como lo es el dispositivo Kinect, y añadiendo filtros de polarización da un resultado de calidad mejor que cientos de scaners láseres de alto costo [13]. Mejorar la calidad y ampliar los entornos de trabajo de las reconstrucciones tridimensionales son prioridades fundamentales en la sociedad moderna que cada día sufre la poca implementación de dichos elementos. Por ejemplo la industria de la construcción pierde billones de dólares cada año debido a las deficiencias de la interoperabilidad [5] entre los arquitectos, ingenieros y los especialistas en el manejo de datos 3D. El NIST [6] ha reportado que estas billonarias pérdidas no se deben tomar de forma negativa, sino como un elemento impulsador de la investigación e inversión en este campo.

Aunque incrementar la resolución del dispositivo no es el objetivo de este trabajo [1], se da por cierto que la introducción de este dispositivo puede colaborar ampliamente en la búsqueda de un sistema de reconstrucción tridimensional compacto de alta resolución y de bajo costo, objetivo que ahora no está tan lejos en el grupo de óptica. Los escenarios posibles del uso del Kinect son extensos, la medicina, arte, educación, entretenimiento y demás, el interés de usar el dispositivo Kinect no solo en el ámbito del entretenimiento es debido a su gran potencial. De esta manera el Kinect se ha convertido en un sensor 3D importante que puede ser usado en el campo industrial o médico, los cuales son de especial interés en esta tesis. El Kinect usa un patrón de luz estructurada [7] para crear mapas de profundidad en tiempo real, el cuál es usado para reproyectar una nube de puntos 3D discretos.

El 4 de noviembre de 2010 el dispositivo Kinect, usado en la industria del video juego, podía ser adquirido de manera libre por cualquier persona. Con el fin de incrementar el uso del dispositivo, Industrias Adafruit ofreció una recompensa para quién pudiera entregar un controlador de código abierto para el Kinect. El 10 de noviembre del mismo año se anunció al español Héctor Martín como el ganador, quién usando ingeniería inversa desarrolló un controlador para GNU/Linux; en Junio de 2011 Microsoft liberó su propio software (SDK)[7] [16], desde entonces la tecnología del dispositivo desarrollada por la compañia PrimeSense

[5] La **Interoperabilidad** se define como la capacidad de gestionar y comunicar productos electrónicos y datos de proyectos entre empresas colaboradoras

[6] El National Institute of Standards and Technology reportó en el 2002 pérdidas que ascienden a los 16 billones de dólares bajo el concepto de interoperabilidad ineficiente.

[7] Software Development Kit https://www.microsoft.com/en-us/download/details.aspx?id=44561

y aquirida por Apple Inc ha sido ampliamente usada en múltiples campos como lo son la visualización computacional, robótica, medicina [8], entre otros.

En el contexto de la reconstrucción tridimensional y reconocimiento de patrones, el Kinect ha tenido gran acogida por múltiple razones, entre ellas su bajo costo y su fácil manejo. Muchos sensores 3D usados para la extración de información topográfica de objetos son aparatosos y costosos (ver ejemplo, [5]) lo cuál es una gran dificultad para el campo investigativo, que en general siempre carece de recursos. A pesar de su enorme potencial, el Kinect posee varias limitantes como lo es su baja resolución, es por esto que se hace necesario la evaluación de varios factores que determinen la viabilidad de implementar tal dispositivo en algunos campos como la metrología óptica.

Determinar que factores influyen a la hora de tomar datos es necesario, puesto que de tal forma se caracteriza en que ambientes es posible usar el Kinect y que superficies son aptas para la toma de datos. Para esto se ha llevado a cabo diferentes pruebas, como lo fue en su inicio el ambicioso proyecto de reconstruir un carro en condiciones naturales, lo cuál no se pudo llevar a cabo dada la alta incidencia de la luz del sol, la cuál ingresa al sensor de manera aleatoria, la luz del sol posee suficiente potencia en el rango infrarrojo como para cegar el sensor aún a pesar de su polarizador que bloquea ciertas longitudes de onda. En pocas palabras, no es suficiente el polarizador puesto que aunque se controle el tipo de luz que atraviesa al sensor, no es posible controlar la rata con la que ésta entra; el exceso de luz en este rango cega por completo la camara IR. También se pudo observar que el dispositivo Kinect no puede adquirir datos de superficies con una reflectancia muy alta, como lo es un espejo, estos se muetran en la nube de puntos como datos ruidosos, y el objeto o espacio con estas características perderá toda la información topológica. Existen otros factores como el exceso de temperatura que pueden afectar la toma de datos, a pesar que el Kinect posee un peltier (disipador de calor) para mantener la longitud de onda del rayo constante se pudo observar algunas variaciones en la toma de datos cuando la temperatura del medio ambiente era mayor a la temperatura promedio.

Con el desarrollo de este trabajo se logró mostrar un método para el cálculo del error del dispositivo Kinect y la posibilidad de implementarlo en futuras versiones del mismo. Con los datos adquiridos por el dispositivo Kinect se comprobó como la distancia al objetivo afecta la calidad de la reconstrucción de manera exponencial se pudo observar un aumento significativo de la capacidad resolutiva del dispositivo Kinect para distancias entre 0.5 m y 1 m, mientras que para distancias superiores su capacidad resolutiva disminuía notablemente, hasta 10 veces mayor, dando errores que superan los 0.02 m cuando se acerca al límite de su campo de observación.

Este trabajo está organizado en 6 capítulos de la siguiente manera: En el capítulo 2 se da una breve descripción de las capacidades del Kinect mostrando el software y hardware disponible a la fecha para su tratamiento de datos; en el capítulo 3 se muestra la teoría de calibración de las cámaras del Kinect, anteriormente exploradas en el grupo de óptica y bien conocidas, así como los parámetros intrínsecos y extrínsecos encontrados. También se presenta el modelo matemático usado para la calibración de profundidad y los detalles de la discretización de los puntos. En el capítulo 4 se muestra la teoría detrás del algoritmo ICP el cuál se encarga de hacer un empalme de las nubes de puntos, también se muestran diferentes reconstrucciones tridimensionales desde diferentes perspectivas con sus respectivos empalmes generando así un render total de la figura; en el capítulo 5 se muestra una breve descripción de las herramientas matemáticas usadas para el cálculo del error cometido en una reconstrucción 3D. También se muestran los resultados obtenidos

para el cálculo del error, así como diferentes reconstrucciones tridimensionales de diferentes objetos; finalemente, en el capítulo 6 se muestran las conclusiones de esta tesis.

CAPÍTULO 1

Dispositivo Kinect

1.1. Software de uso

En esta sección se muestra una buena recopilación del software actual que permanece en el mercado y en la comunidad internauta para la manipulación de los datos que arroja el kinect en diferentes sistemas operativos. La compilación o instalación de algunos de estos requiere de cierta destreza computacional, como lo es para RGBDemo y Rgbdslam que son de uso libre. Sin embargo se puede adquirir software desde 130 $ USD como lo es el caso de Skanect, el cuál reune las características necesarias para llevar a cabo un reconstrucción tridimensional de casi cualquier objeto. Uno de los problemas de acceder a software no libre, a parte de su costo, es poder hacer uso del código abiertamente y explorar su funcionamiento, como también de los parámetros de calibración los cuales vienen predefinidos desde su fabricación; a esto se le puede agregar que no podrá ser usado e integrado en otro proyecto independiente, algo que no sucede con el uso de software libre. En el mapa de la Figura 1.1 se resumen algunos de los software más importantes usados en la actualidad y los sistemas operativos que permiten su ejecución.

Kinfu: PCL-KinFu hace parte de la librería PCL (Point Cloud Library)[1] y en el fondo no es mas que la implementación de código abierto de Kinect Fusion [11], [19], un poderoso algoritmo creado por Microsoft para la reconstrucción en tiempo real de escenas. KinFu ha estado cambiando constantemente y actualmente posee muchas variantes [2], la descripción de los ligeros cambios introducidos en KinFu pueden ser examinados en [22] el cuál muestra su versatilidad debido a que permite trabajar con varios sensores de profundidad a diferencia de Kinect Fusion. Es decir puede ser usado con toda la gama existe de sensores de profundidad que incorparan la tecnología PrimeSense.

ReconstrucMe: ReconstrucMe SDK [3] hace parte del software pago disponible en el mercado usado para la reconstrucción en tiempo real de escenas. Las versiones que carecen de licencia solo permiten exportar una limitada cantidad de puntos. ReconstrucMe va correlacionando cada nube de puntos adquirida, reconociendo los puntos ya tomados de la superficie y vinculando estos al mapa 3D que unificado. Este software es compatible con

[1] http://pointclouds.org/news/kinectfusion-open-source-html
[2] http://pointclouds.org/documentation/tutorials/using_kinfu_large_scale.php
[3] http://reconstructme.net/

diferentes tarjetas GPU para mejorar su rendimiento y exactitud; en el modo normal usa un metro cuadrado dividido en 256 elementos mientras que en su modo de alta resolución toma el mismo volumen y lo divide en 512 elementos. Adicionalmente soporta una alta gama de sensores [4] lo cual puede ser útil en posteriores trabajos.

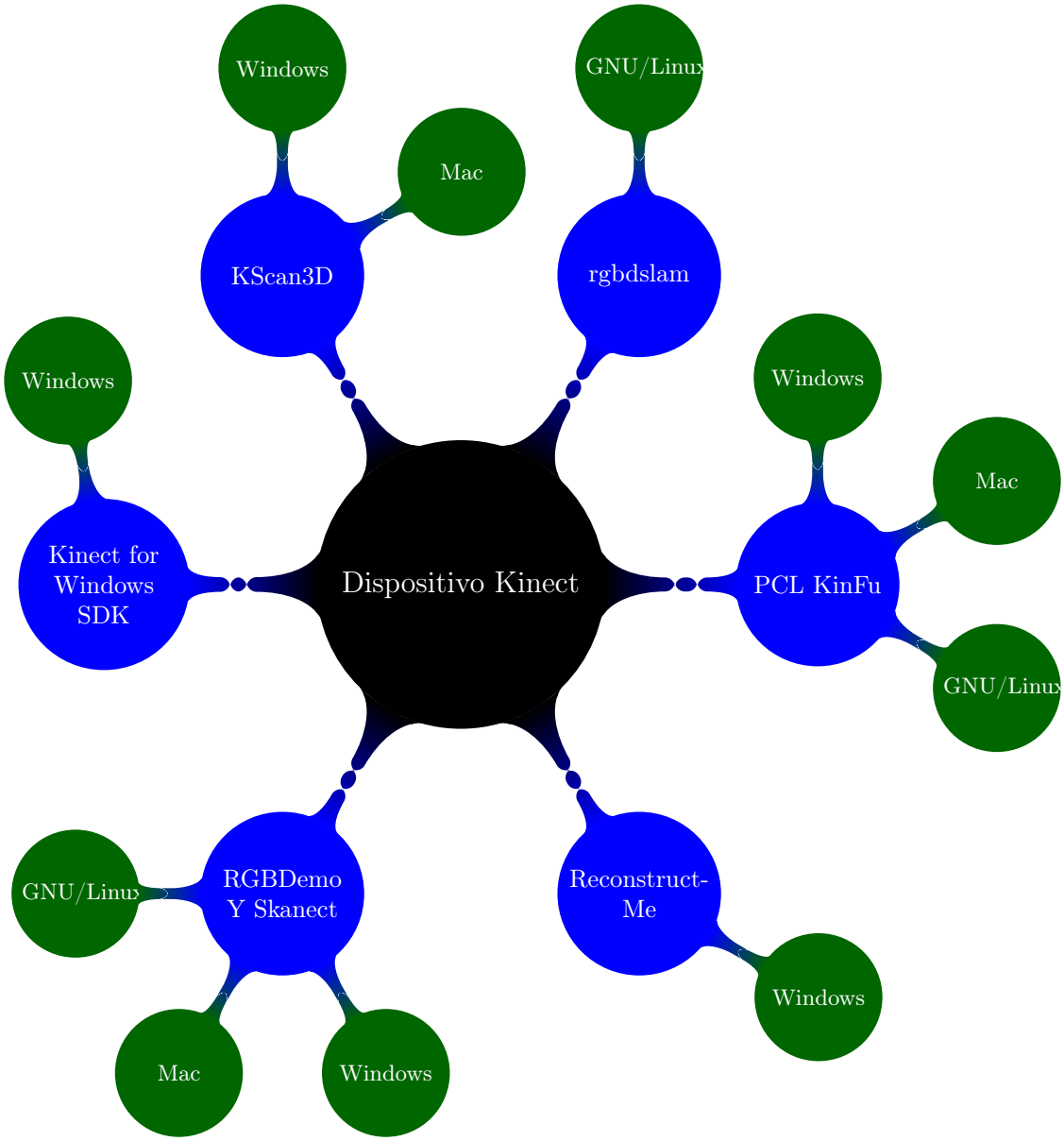


FIGURA 1.1. Software de uso para el dispositivo Kinect

Skanect: Es el método pago más fácil y eficiente para obtener una reconstrucción tridimensional por medio del Kinect o aquellos dispositivos que usan la tecnología PrimeSense. Skanect [5] permite rápidamente exportar en diferentes formatos y obtener un render 3D

[4] http://reconstructme.net/qa_faqs/which-3d-sensors-are-supported-by-reconstructme/
[5] http://skanect.occipital.com/

solo rodeando el objeto a reconstruir. Skanect fue desarrollado por ManCTL, una compañia fundada a finales de 2011 por Nicolas Tisserand y Nicolas Burrus. La compañia fue adquirida por Occipital en 2013.

RGBDemo: Pertenece al conjunto de software libre, actualmente no se encuentra en mantenimiento ni muestra mejoras. Debido a su gran acogida, su creador decidió formar lo que actualmente se conoce como Skanect. RGBDemo sin duda es la mejor opción en cuanto al tratamiento de datos, pues permite la exploración de la mayoría de características que posee el dispositivo Kinect y dispositivos semejantes, tiene compatibilidad con PCL, permite integrar y calibrar varios dispositivos Kinect a una sola unidad de procesamiento (PC), lo cuál puede llegar a ser muy útil si se quieren adquirir varias perspectivas de un mismo objetivo. Gran parte de sus cualidades son presentadas en la referencia [6], entre ellas la compilación de este que en muchos casos presenta serias dificultades.

SDK Windows: El software de desarrollo para el Kinect permite usar y desarrollar nuevas aplicaciones para el uso de los datos del Kinect y sus características. Actualmente existe una abundante información acerca de su uso [12]. Para desarrollar una aplicación se necesita instalar Kinect para Windows SDK disponible en la referencia [7], además de algunos requerimientos extras como lo son:

- Microsoft Visual Studio 2012 Express ó Visual Studio 2012 edition
- .NET Framework 4 (Instalada con Visual Studio 2012)

Kinect para Windows permite el uso de su dispositivo más reciente Kinect 2.0 el cuál posee muchas mejoras sobre la primera version, entre ellas está una mejora considerable en su resolución. Este dispositivo se deja para posible investigaciones en posteriores trabajos. El SDK está disponible para C++, aunque la mayoría de las características y códigos estan disponibles para C# lo cuál hace su programación algo difícil.

KScan3D: Este software hace parte de la gama de sotfware pago, cuenta con diferentes opciones para la manipulación y edición de nubes de puntos, esto lo diferencia de los demás y hace que su valor sea de unos 300 $ USD. Al igual que Skanect permite la adquisición de nubes de puntos en tiempo real. KScan3D [8] posee la facilidad de editar y exportar fácilmente sus adquisiciones y modificarlas con la apliación de diferentes algoritmos y filtros, soporta los sensores Xtion PRO, PrimeSense Carmine Sensor y Kinect Sensor, todos basados en la tecnología PrimeSense.

RgbdSlam: En español, Localización y Mapeado Simultáneo (SLAM), es una técnica usada por robots autónomos para construir un mapa de un entorno desconocido, a la vez que estima su trayectoria al desplazarse dentro de este. Rgbdslam [9] usa el complejo algoritmo SURF, el cuál usa las características visuales de dos imágenes para hacerlas coincidir al mismo tiempo que usa el algortimo RANSAC para estimar la transformación 3D entre las imágenes. Rgbdslam no solo permite obtener reconstrucciones tridimensionales en tiempo real, también es usado ampliamente en la robótica como método de identificación de el entorno en el cuál se mueve un robot, ver ejemplo [26].

[6] http://rgbdemo.org/
[7] https://www.microsoft.com/en-us/download/details.aspx?id=44561
[8] http://www.kscan3d.com/
[9] http://felixendres.github.io/rgbdslam_v2/

1.2. Hardware del Kinect

El dispositivo kinect es un periférico de entrada ajustado a la consola Xbox 360 y está diseñado originalmente para percibir, oir y reconocer los movimientos de los jugadores en cierto escenario. Es por esto que posee un conjunto de elementos necesarios para su funcionamiento; aunque algunos de estos no serán usados en este proyecto vale la pena mencionar sus especificaciones técnicas, entre las cuales se encuentran sus cámaras frontales y micrófonos especificados en la figura 1.2. El principio básico detrás del sensor de profundidad del kinect es emisión y captura de un patrón de speckle pseudo-aleatorio leído y procesado por el microchip PS108. En la tabla 1.1 se resumen las propiedades principales y fundamentales del dispostivo kinect.

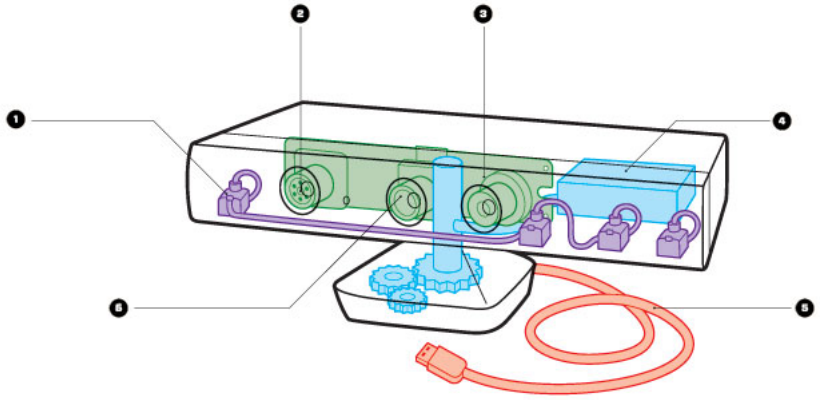


FIGURA 1.2. *Componentes internos del Kinect* Fuente: https://www.wired.com/2011/06/mfkinect/

1. **Micrófonos**
   El arreglo de 4 micrófonos, son los que permiten la interacción del usuario con el dispositivo, con fin de darle órdenes desde la distancia. Su configuración y ubicación en el sensor le permite a este conocer de dónde provienen los sonidos en una habitación, así como también están programados para tomar gran parte del ruido proveniente del exterior y poder eliminarlo.

2. **Proyector Láser**
   Consiste en un láser no modulado, que no produce pulsos en su salida, por lo tanto se mantiene constante. El láser emite luz con una longitud de onda de $\lambda = 830$ [nm], este es constituido por un sistema de difracción que subdivide el haz de luz en diferentes instancias, proyectando así un patrón pseudo-aleatorio, aunque es probable que la difracción produzca pérdidas. El sensor emite una potencia máxima de unos 60 [mW], la cuál no es nociva para la visión

3. **Sensor CMOS de infrarrojos** Se trata de un sensor CMOS monocromo de 1/2"MT9M001C12STM, con una tamaño de pixel de 5.2 [μm]. Posee un formato de video de 5:4 con una resolución de 1280x1024, es decir unos 1.3 megapixeles; posee un filtro cromático que opera a la misma frecuencia que el láser para evitar que luz en otros rangos de longitud de onda entren al sensor y lo ceguen por completo, aunque algunos experimentos realizados con otras fuentes de luz visible ($\lambda = 950$ [nm]) producen una influencia mínima en el sensor.

4. **Motor de inclinación**
   El sensor Kinect cuenta con un motor de inclinación que mueve el sensor de arriba a abajo unos $\pm 27^{\circ}$ que permite abarcar fácilmente un cuerpo situado frente al sensor, sin importar la altura a la cuál esté situado, además posee un acelerómetro que que indica la inclinación que posee en cada instante, ver 1.3.

5. **Cámara RGB**

   La cámara RGB opera a 30 Hz y toma imágenes de 640x512 pixeles; ésta puede aumentarse a un modo de alta resolución, transmitiendo aproximadamente a 10 fps a 1280x1024 pixeles. La cámara en si posee un grupo de características especiales incluidas como lo es el balance de blancos, referencia de negros, eliminación de parpadeo, saturación de color y corrección de defectos. Al igual, la cámara IR toma imágenes a 1200x960 pixeles , aunque estas imágenes son submuestreadas por el hardware ya que el USB no puede llevar tanta información junto con los datos de la cámara RGB.

6. **Cable USB** La conexión se hace por medio de un cable USB 2.0 tanto a la consola como al ordenador. El hecho de que sea un cable USB 2.0, que transmite información aproximadamente a 60 MB/s, hace que no se pueda transmitir toda la información de las imágenes a la más alta resolución y con la mayor tasa de cuadros por segundo. De esta manera la resolución que llega a los ordenadores no es la misma que posee el sensor. Con este mismo cable se alimenta de energía las cámaras. El motor de inclinación requiere de conexión a una red eléctrica diferente.

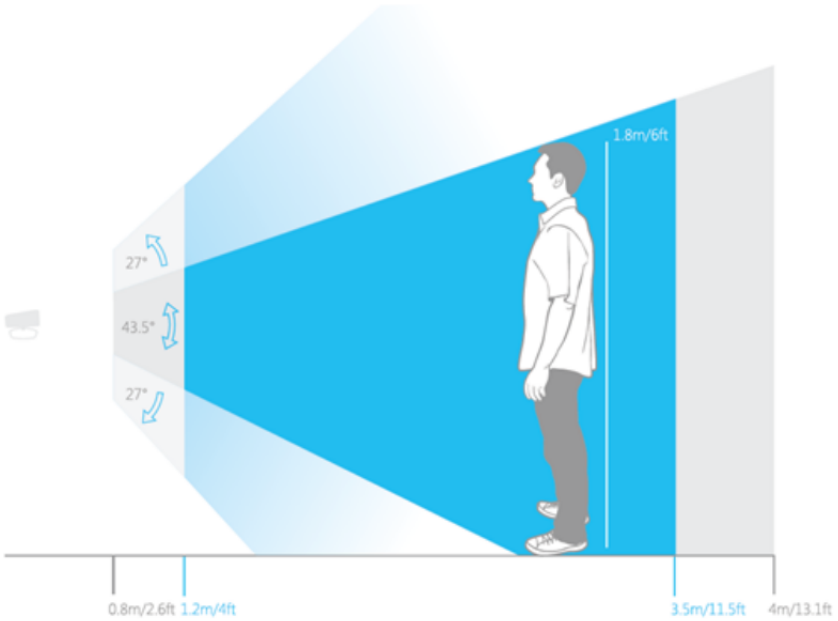
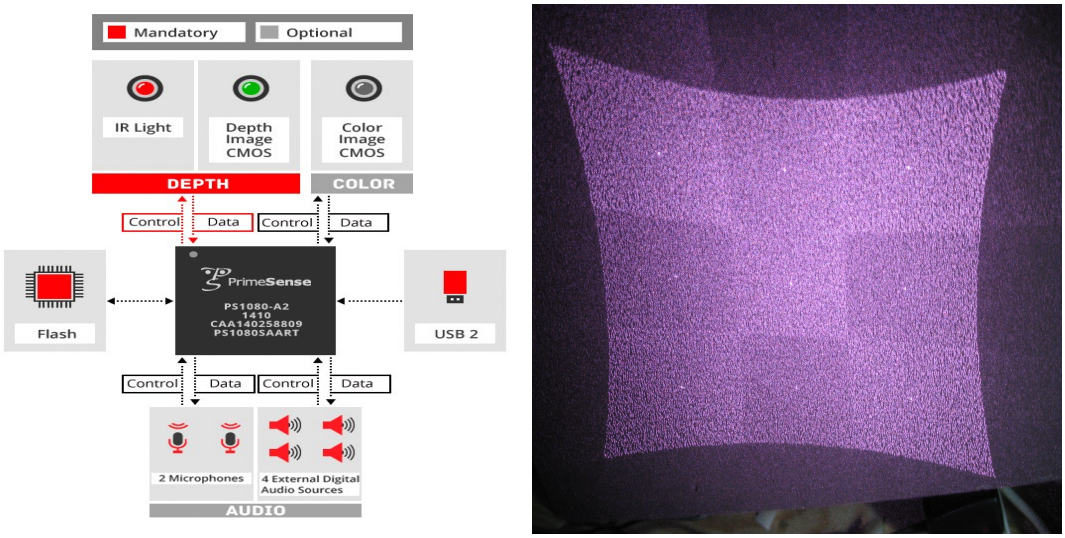


FIGURA 1.3. *Campo de observación* Fuente: https://malenyabrego.wordpress.com/category/kinect/page/6/

El **microchip** PS1080 desarrollado por compañia Israelí PrimeSense, y posteriormente adquirida por Apple Inc, se encarga aplicar los algoritmos necesarios para codificar los datos de profundidad (Ver esquema 1.4 a).

El emisor crea un conocido patrón de luz estructurada, este patrón es localmente correlacionado con un patrón fijo grabado en el chip interno con el fin de identificar cualquier desplazamiento local en el patrón debido a alguna variación en la topografía de la superficie observada, que por medio de una triangulación (ver sección 2.3) permite extraer los datos de profundidad. Dentro del sensor existe un peltier (disipador de calor) el cuál se encarga de mantener la temperatura y por ende la longitud de onda constante. En la creación de patrones de speckle es normal observar un punto muy brillante en el centro, es por esto que el patrón de speckle (ver imagen 1.4 b) muestra 9 puntos brillantes generados por la introducción de un filtro de dispersión el cuál dispersa el haz principal.

Por su parte el patrón de **speckle** mostrado en la figura 1.4 (b) es básicamente rectangular pero severamente distorsionado; este rectángulo parece encajar en sub-patrones 3x3 de diferente brillo cada uno.

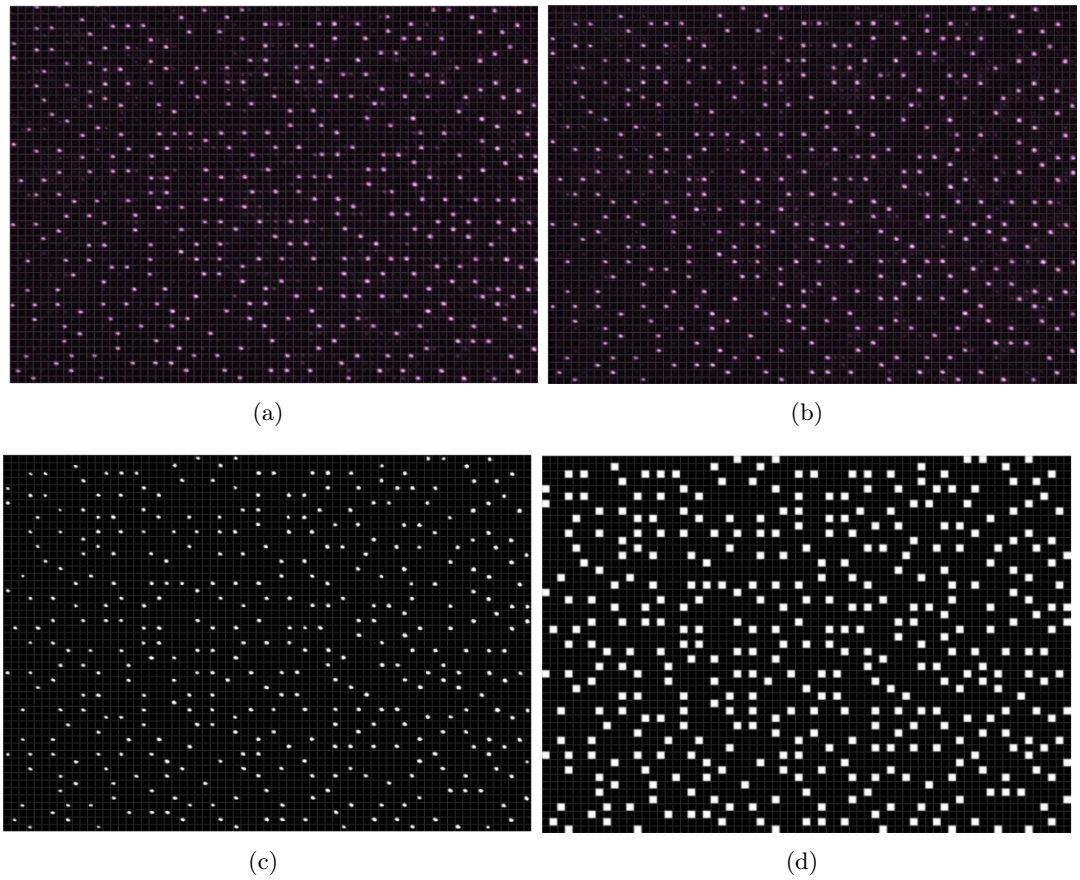


(a) Esquema interno (b) Speckle

FIGURA 1.4. *Tecnología PrimeSense* Fuente: http://www.radiolocman.com/news/new.html?di=146889

El patrón está hecho de una cuadrícula ortogonal regular, los puntos presentes son brillantes, las otras regiones se muestran oscuras. De esta manera el hecho de tener una cuadrícula ortogonal con mas o menos un mapa brillantes/oscuros simplifica a adquisición del patrón, la representación 1.5 muestra un seguimiento sobre una región del patrón. (Una completa revisión sobre el patrón se muestra en [10])

[10]https://azttm.wordpress.com/2011/04/03/kinect-pattern-uncovered/



(a) (b)

(c) (d)

FIGURA 1.5. *Representación en mapa de bits del patrón de speckle* Fuente: https://azttm.wordpress.com/2011/04/03/kinect-pattern-uncovered/

Contando los puntos en las direcciones X e Y, y aumentando la imagen se puede observar que cada punto tiene 32x32 pixeles en promedio. El patrón se divide en una matriz de 3x3 en donde cada elemento de la matriz contiene la misma distribución de puntos de aproximadamente 211x165 puntos; el patrón es invariante bajo rotación de 180° lo cuál no representa ninguna ventaja algorítmica. Es decir el patrón está compuesto por una repetición 3x3 de unos 211x165 puntos, en total 633x495, un número muy similar a la resolución VGA.

PROPIEDAD	**VALOR**
Campo de vision angular	57° horizontal., 43° vertical
Dimensiones (Ancho,Alto,espesor)	$14cm$ x $3.5cm$ x $5cm$
Tamaño de la imagen de profundidad	Máx., 1280x1024 Min. 640x480 (VGA)
Tamaño de la imagen de color	Máx., 1280x1024 Min. 640x480 (VGA)
Cuadros por segundo	Máx 60 fps., Mín 10 fps
Consumo	2.25 W
Tipo de conexión	USB (+Cable de poder)
Temperatura de operación	0° - 40°
Rango de detección en profundidad	$0{,}5m < Z < 4m$ Rango ópt. $1{,}2m < Z < 3{,}5m$

TABLA 1.1. Especificaciones técnicas del hardware del kinect

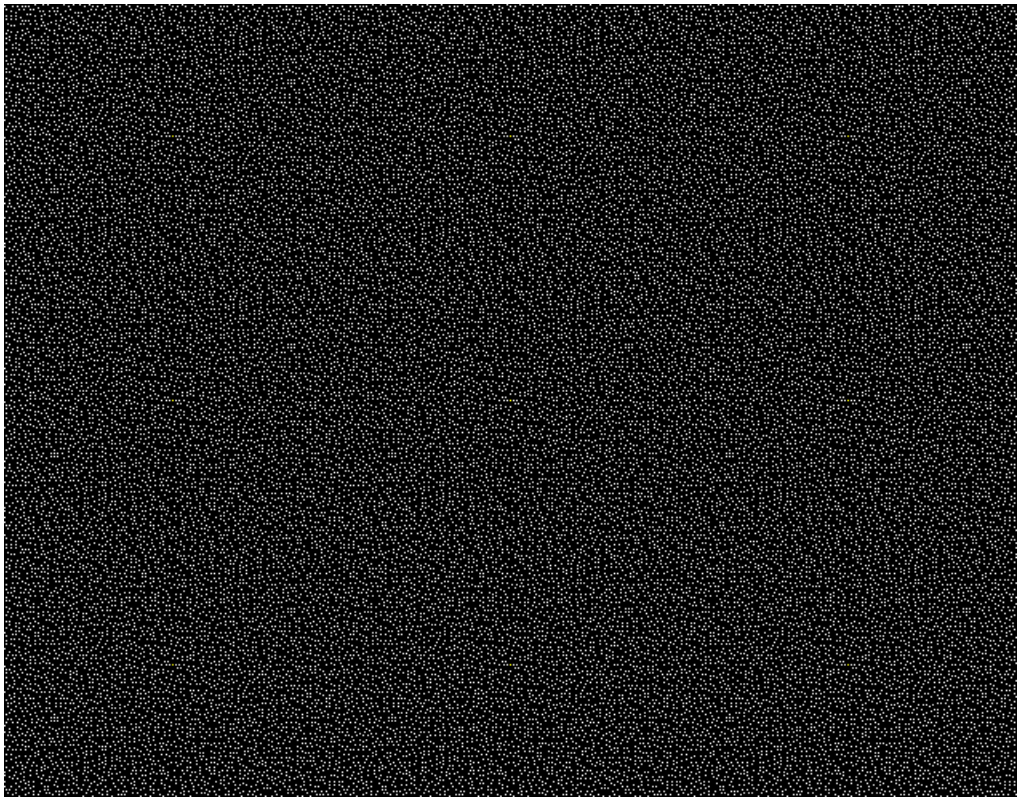


FIGURA 1.6. *Patrón completo reproducido a partir de una región arbitraria* Fuente: https://azttm.wordpress.com/2011/04/03/kinect-pattern-uncovered/

CAPÍTULO 2

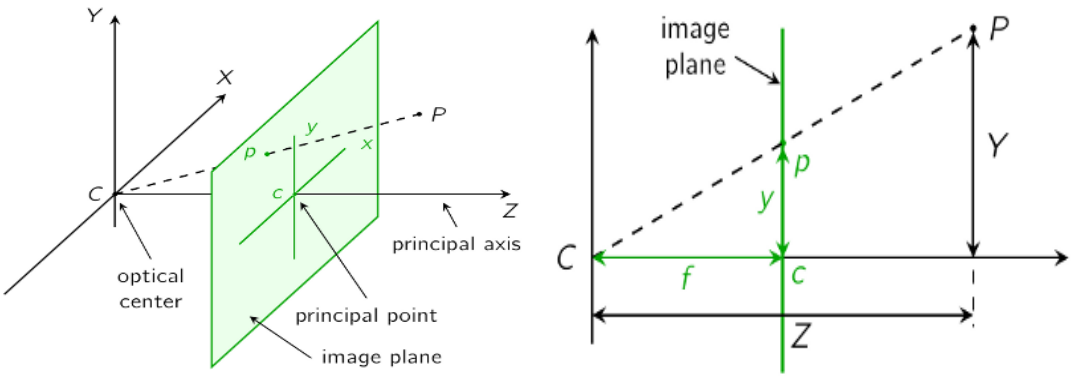
Calibración

Introducción:

En la sección anterior se presentaron los elementos que conforman el Kinect y los programas o software de uso libre o pago que permite explorar sus capacidades metrológicas. En esta sección se muestra cómo los datos arrojados por el Kinect permiten extraer información 3D del objeto. Se revisará en detalle cómo las imágenes obtenidas de las cámaras junto con los parámetros de calibración permiten obtener la nube de puntos 3D. Se presenta teórica, y se evalua experimentalmente, el modelo de calibración en profundidad empleado para modelizar en profundidad el funcionamiento del Kinect.

2.1. Modelo Pinhole

El modelo de la cámara pinhole frecuentemente es utilizado para describir de manera razonable cómo una cámara representa una escena tridimensional. El modelo pinhole describe la cámara más simple e ideal posible, así mismo sus conceptos son determinantes a la hora de implementar determinados cálculos, como lo puede llegar a ser la calibración de un sistema de visión estéreo. El modelo pinhole se puede describir como la relación matemática que existe entre las coordenadas de un punto 3D y su proyección en el plano imagen; esta modelización de una cámara consiste en una pequeña caja oscura, con un agujero (pinhole) en una de sus paredes y un material fotosensible. La luz ingresa por el agujero y golpea la pantalla, para producir una imagen nítida es necesario que el agujero sea muy pequeño, casi infinitesimal, de esta manera la cámara se usa sólo como una aproximación de primer orden en el trazo de una mapa de una escena 3D a una imagen 2D.



(a) Sistemas de referencia (b) Vista lateral

FIGURA 2.1. *Modelo de la cámara pinhole* Fuente: http://pille.iwr.uni-heidelberg.de/~kinect01/doc/reconstruction.html

Para describir el modelo se toma en consideración un punto $P = (X, Y, Z)^T \in \mathbb{R}^3$ el cuál se proyecta en el plano imagen como un punto $p = (x, y)^T \in \mathbb{R}^2$, Ver imagen 2.1 (a). Por facilidad se toma el plano imagen a lo largo el eje Z, de la imagen se observan los dos sistemas de coordenadas: el sistema de coordenadas XYZ y el sistema coordenado del plano imagen xy; una representación lateral del modelo se ve representado en la figura 2.1 (b) en el cuál por medio de triángulos semejantes se obtiene la relación 2.1 que se puede escribir según 2.2.

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

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} fX/Z \\ fY/Z \end{pmatrix} \tag{2.2}$$

En términos de las coordenadas homogéneas [27] se tiene:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = s \begin{pmatrix} fX/Z \\ fY/Z \\ 1 \end{pmatrix} = s \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = s \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{2.3}$$

En general se toma el origen como el centro de coordenadas del plano imagen. Dado que este puede no estar siempre situado en el origen se debe añadir un posible corrimiento $(\hat{c}_x, \hat{c}_y)$ en el punto central. Luego la ecuación anterior puede reescribirse incluyendo estos parámetros y en términos de las coordenadas homogéneas como:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = s \begin{pmatrix} fX/Z + \hat{c}_x \\ fY/Z + \hat{c}_y \\ 1 \end{pmatrix} = s \begin{pmatrix} fX + \hat{c}_x Z \\ fY + \hat{c}_y Z \\ 1 \end{pmatrix} = s \begin{pmatrix} f & 0 & \hat{c}_x & 0 \\ 0 & f & \hat{c}_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \tag{2.4}$$

donde s representa un factor de equivalencia en las coordenadas homogéneas. En las anteriores ecuaciones los valores $f, \hat{c}_x, \hat{c}_y$ en unidades de longitud. Los puntos sobre la superficie pueden ubicarse usando el sistema coordenado del objeto (X_w, Y_w, Z_w). Es útil

introducir la transformación rígida que relaciona (X, Y, Z) con (X_w, Y_w, Z_w), la ecuación 2.4 puede escribirse como:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = s \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \tag{2.5}$$

Donde $f_x = k_x f$ y $f_y = k_y f$ están en unidades de pixeles con k_x,k_y el número de pixeles por unidad de longitud en la dirección x y y respectivamente. De igual forma, $c_x = k_x \hat{c}_x$ y $c_y = k_y \hat{c}_y$. La ecuación 2.5 describe la relación existente entre un punto cualquiera en el espacio tridimensional del objeto y su equivalente capturado por la cámara en el plano imagen de ésta; más adelante se observará el significado de cada uno de estos valores. Es necesario mencionar que para cada cámara del Kinect se deben encontrar diferentes valores para los parámentros f_x, f_y, c_x, c_y mientras que para la matriz que se introdujo de manera Ad-hoc se usa una sola transformación para combinar los sistemas coordenados, luego entonces solo se necesita calcular una sola matriz de transformación.

2.2. Calibración común para las cámaras IR y RGB

La calibración es uno de los procesos más importantes a la hora de llevar a cabo una reconstrucción tridimensional. Una descripción detallada de este proceso para el dispositivo kinect es mostrada en [15], la calibración permite producir una correlación exacta de un objeto en el mundo real con los pixeles de la imagen. Generalmente hablando, la calibración es un proceso mediante el cuál se toma un conjunto de imágenes con el objetivo de adquirir los parámetros que describen el modelo pinhole de la cámara, éstos parámetros se clasifican de dos formas.

Parámetros intrínsecos:

Básicamente los parámetros intrínsecos describen y caracterizan la configuración interna de la cámara; éstas especificaciones internas detallan como se genera la imagen. En la ecuación 2.5 se puede observar los parámetros intrínsecos de manera directa, dada la matrix

$$A = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.6}$$

Donde $(f_x,\ f_y)$ son las distancias focales y (c_x, c_y) es el centro de la proyección del plano imagen. Dado que los pixeles en la imagen pueden ser rectangulares en vez de un cuadrados se toman dos distancias focales, lo cuál nos lleva a introducir el tamaño de cada pixel en su respectiva dirección. Así $f_x = F s_x$ donde F es la distancia focal física de la cámara y s_x es la distancia de cada pixel en la dirección x. De igual forma se debe trabajar para f_y y s_y, las unidades de f_x y f_y están en pixeles, puesto que s_x tiene unidades de pixeles por milímetro y las unidades de F son milímetros.

Los valores mostrados en la matriz A no son los únicos parámetros intrínsecos de una cámara, también lo son los **coeficientes de distorsion**, los cuales llegan a ser introducidos

dado que una cámara puede presentar fallas en su proceso de fabricación. Estos defectos se pueden mostrar de dos formas, aquellos que estan basados en la forma de los lentes y son conocidos como distorsion radial y aquellos debidos al desplazamiento del lente, conocidos como distorsion tangencial. Se dice que una imagen está afectada por la distorsion radial cuando las lineas rectas en el mundo real se muestran curvas en la imagen, la figura 2.2 evidencia los dos tipos de distorsion radial que pueden presentarse en una imagen

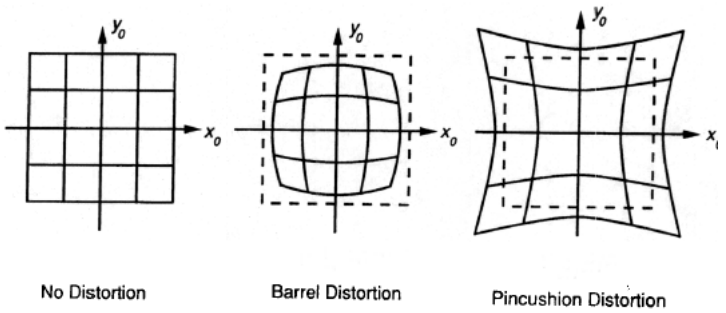


FIGURA 2.2. *Tipos de distorsion radial: (izquierda) malla sin distorsion, (medio) distorsion Barel, (derecha) distorsion Pincusion.* Fuente: http://www.faculty.virginia.edu/ASTR5110/lectures/optics2/optics2.html

Dado un punto ideal no distorsionado $q_p = (u_p, v_p)$ y el generado por la cámara $q_d = (u_d, v_d)$ se plantea la siguiente relación matemática:

$$u_p = u_d - \delta_u \tag{2.7}$$
$$v_p = v_d - \delta_v \tag{2.8}$$

Donde δ_u y δ_v son funciones de distorsion en forma de una funcion polinomial (ver fórmulas siguientes 2.9). Asumiendo que la distorsion en el centro es nula se puede aproximar la posición de un punto no distorsionado agregando componentes adicionales; como se establece en [6] bajo la expansión de taylor se puede modelizar la distorsion realice como:

$$x_{corregido} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2.9}$$
$$y_{corregido} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2.10}$$

Dónde x_c y y_c son las coordenadas del punto no distorsionado, mientras que las coordenadas de los puntos distorsionados son x y y; k_1, k_2, k_3 son los llamados coeficientes de distorsion radial, con $r = \sqrt{(x - x_p)^2 + (y - y_p)^2}$ la distancia de el punto distorsionado al punto principal (x_p, y_p).

De manera similiar se trata la distorsion tangencial, generada por el mal posicionamiento del lente, puesto que en el caso ideal el lente debe ser paralelo a los otros lentes y su punto central debe estar localizado sobre el eje óptico. Esta distorsion se puede modelizar y corregir tomando en cuenta las siguientes ecuaciones

$$x_{corregido} = x + (2p_1y + p_2(r^2 + 2x^2)) \quad (2.11)$$
$$y_{corregido} = y + (p_1(r^2 + 2y^2) + 2p_2x) \quad (2.12)$$

Donde p_1 y p_2 son coeficientes los coeficientes de distorsion tangencial.

Parámetros extrínsecos:

Se puede establecer una relación exacta para definir la transformación que me convierte un punto del espacio real en el sistema coordenado del objeto, al sistema coordenado de la cámara. Esta relación se establece con los parámetros extrínsecos. Conociendo la distancia entre un origen de un sistema coordenado a otro y la posible rotación en cierto ángulo que pueden llegar a tener sus ejes, se puede establer dicha relación. Dado que rotar un punto representado como vector es equivalente a multiplicar el punto por una matrix de rotación de 3x3, y que la traslación necesaria para llevar un punto del sistema coordenado del objeto o llamado sistema coordenado del mundo al de la cámara se puede notar como $T = p_{mundo} - p_{camara}$, se puede escribir una fórmula general para la transformación de un punto en el sistema coordenado del mundo al sistema coordenado de la cámara como:

$$p_{camara} = R(p_{mundo} - T) \quad (2.13)$$

Donde R y T son las componentes de rotación y traslación; así como p_{mundo} y p_{camara} son coordenadas en el espacio real y la cámara respectivamente.

Los algoritmos que permiten encontrar éstos parámetros son basados en los trabajos de Zhengyou Zhang [29] y Duane C.Brown [6]. Estos métodos se basan en la extracción de las esquinas de un patrón ajedrezado con dimensiones conocidas para encontrar dichas variables, y ampliamente usadas en el (GOTS) [21]. La siguiente imagen 2.3 muestra la secuencia general usada .

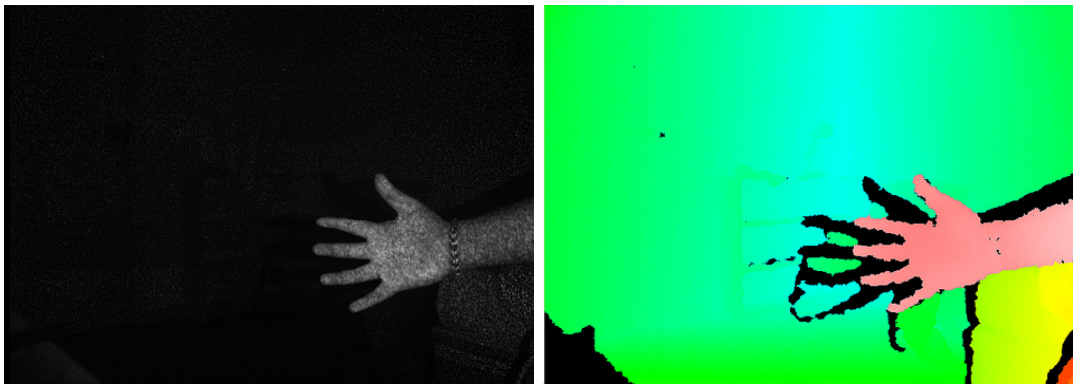
FIGURA 2.3. *Método de Zhang para la calibración*

2.3. Relación entre la disparidad y la profundidad

El sensor de profundidad emplea tres tipos de imágenes, una de ellas está almacenada en el chip PrimeSense y es invisible para los usuarios, la segunda imagen es aquella capturada por la cámara IR [2.4 (a)], y la tercera imagen es el mapa de disparidad [1] generada apartir de la comparación de las dos anteriores [2.4 (b)]. El medio por el cuál el dispositivo Kinect puede codificar la profundidad de los objetos [7] está basado en tres elementos, el primero es

[1] ***Disparidad:*** Es la diferencia relativa existente en la posición de cada imagen, la cuál tiene una relación directa con la profundidad.

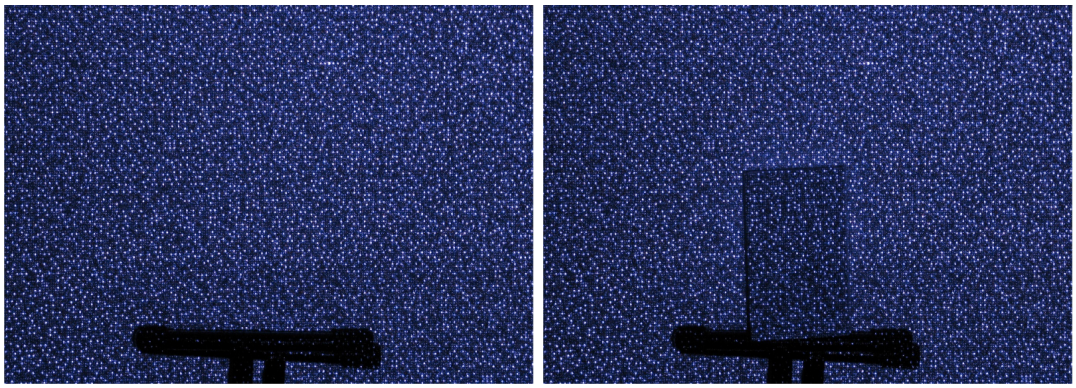
el patrón pseudo-aleatorio enviado por el emisor IR, el segundo es la cámara IR que detecta dicho patrón de luz infraroja y el tercero es una imagen del patrón a una distancia fija almacenada en el chip, estos dos últimos pueden ser vistos como una cámara estereoscópica.



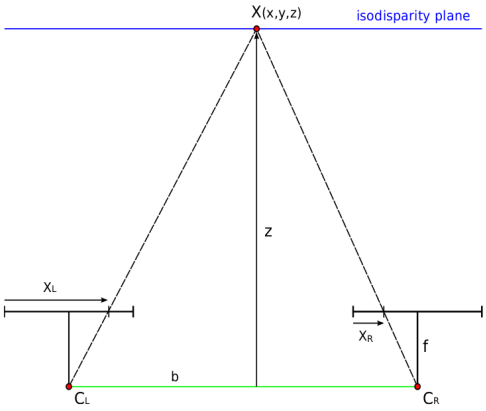
(a) Imagen IR (b) Mapa de disparidad

FIGURA 2.4. *Imágenes generadas por el sensor IR*

Los corrimientos entre la imagen fija [2.5 (a)] y el patrón proyectado sobre una escena [2.5 (b)] pueden ser calculados por una pequeña ventana de correlación que es usada para generar el mapa de disparidad entre las dos imágenes. El algoritmo de correlación local puede ser visto en [23]. A partir de la disparidad se puede calcular la profundidad de un objeto mediante un proceso de triangulación. Si la imagen de referencia del patrón speckle se adquirió a una distancia Z_0 del Kinect, el corrimiento de cada punto speckle debe ser cero, si la imagen de referencia se correlaciona con la imagen adquirida del speckle a la misma distancia ($Z = Z_0$). De esta manera la imagen de disparidad indica el corrimiento local del speckle con respecto a una distancia de referencia Z_0.



(a) Patrón proyectado sobre un plano (b) Patrón proyectado sobre un objeto



(c) Proceso de triangulación

FIGURA 2.5. *Relación entre la disparidad y la profundidad, tomada de [25]*

Modelo de distancia

Existen varios modelos para la calibración de profundidad del Kinect [25],[10], básicamente todos basados en el modelo de la cámara estereoscópica, en el cuál la triangulación hecha sobre el mapa de disparidad es la base obtener la profundidad de un objeto. La figura 2.5 (c) muestra la configuración del sistema óptico del Kinect y una ilustración de la disparidad generada a partir de la profundidad: los centros de las cámaras son denotados como C_L y C_R, b denota la línea de base entre las cámaras, mientras que f es la distancia focal. La relación entre la profundidad z y las distancias x_L y x_R, que son dadas por la intersección de los rayos que van a través del punto X y el plano imagen, puede ser encontrada usando triangulos similares:

$$\frac{b}{z} = \frac{b + (x_R - x_L)}{z - f} \tag{2.14}$$

Sustituyendo $d = x_R - x_L$ y reacomodándo términos tenemos:

$$d = \frac{bf}{z} \tag{2.15}$$

La hipótesis es que los datos de la imagen de disparidad d_{raw}, obtenidas del proceso de correlación, pueden ser relacionados a la distancia d por un polinomio de primer orden de la forma $d = c_1 d_{raw} + c_0$, luego el modelo queda escrito como:

$$z = \frac{bf}{c_1 d_{raw} + c_0} \tag{2.16}$$

donde b es la linea de base entre el proyector laser y la cámara IR, f es la distancia focal de la cámara IR y c_1, c_0 son los coeficientes del modelo.

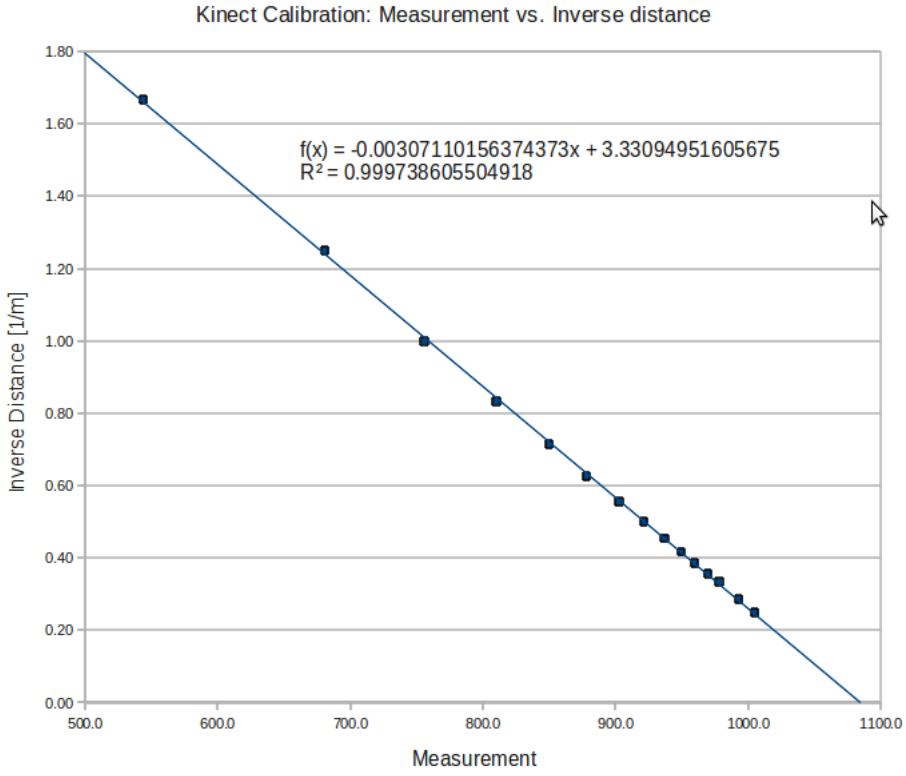


FIGURA 2.6. *Resultados experimentales. Fuente: ROS.org*

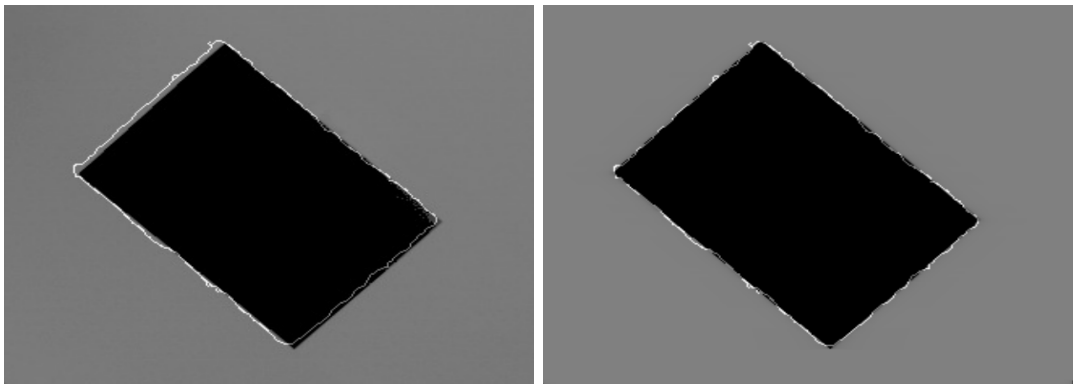
Resultados experimentales obtenidos en ROS.org [2] han demostrado la validez de la relación lineal entre d_{raw} y $1/z$ calculándose por regresión lineal el valor de las constantes del modelo. Los datos experimentales encontrados son mostrados en 2.6. La figura 2.6 indica que existe una fuerte relación lineal entre $1/Z$ y d_{raw} que corresponde a la imagen de disparidad arrojada por el Kinect. Como b y f son constantes, la calibración consiste en calcular las constantes de la relación lineal.

Corrección de imágenes IR y de profundidad

Cuando se compara la imagen de profundidad y la imágen IR se observa un pequeño desplazamieno constante de coordenadas en pixeles, este corrimiento se muestran en la figura 2.7. El valor del desplazamiento entre las imágenes es determinado en diferentes trabajos (ver [25]), los cuales proponen una corrección. Experimentalmente se encuentra que las coordenadas de la imagen de profundidad y la cámara IR se relacionen de la forma:

[2] http://wiki.ros.org/kinect_node/Calibration

$$\begin{pmatrix} x_D \\ y_D \end{pmatrix} = \begin{pmatrix} x_{IR} - 3 \\ y_{IR} - 3 \end{pmatrix} \tag{2.17}$$



(a) *Imágen de la cámara IR (Negro), imágen de profundidad (borde blanco) desalineadas*

(b) *Imágen de la cámara IR (Negro), imágen de profundidad (borde blanco) alineadas, tomada de [25]*

FIGURA 2.7. Imágenes de profundidad

Muchos autores coniciden en que el corrimiento surge del proceso de correlación.

2.4. Estero Calibración

Cuando se trabaja con un par de cámaras como lo es el caso del Kinect, (cámara IR y cámara RGB) se deben calibrar las dos cámaras al mismo tiempo, con el objetivo de que ambas cámaras observen el mismo objetivo. Para esto se usa el mismo patrón ajedrezado. Este proceso generará los parámetros necesarios para describir cómo se correlacionan las dos cámaras en el espacio, para poder transformar un sistema coordenado de una cámara al otro. Estos parámetros generan un vector de traslación y una matrix de rotación en vez de un conjunto de de traslaciones y rotaciones, ya que se está correlacionando dos cámaras para buscar correspondencias entre la posición de las cámaras y la imagen observada. Se considera que los ejes ópticos son paralelos y un punto P en el sistema coordenado del mundo, así se puede transformar en el espacio de la cámara izquierda y derecha separadamente de acuerdo a:

$$P_l = R_l P + T_l \tag{2.18}$$
$$P_r = R_r P + T_r \tag{2.19}$$

Donde la primera ecuación es válida para la cámara izquierda mientras la seguda es válida para la cámara derecha, se tiene que el acoplamiento (ver 2.8) entre los dos sistemas se da bajo la ecuación:

$$P_l = R^T (P_r - T) \tag{2.20}$$

Donde R y T detallan la cantidad de desplazamiento y rotación entre los sistemas coordenados de las cámaras

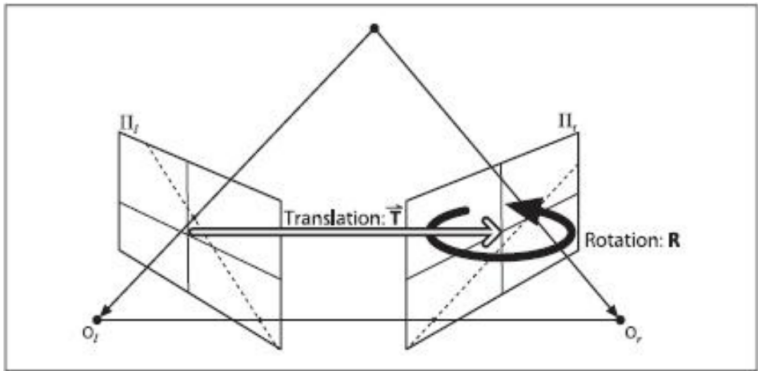


FIGURA 2.8. *Relación entre los sistemas coordenados de las cámaras, izquierda y derecha. Fuente: [4]*

Resolviendo las ecuaciones anteriores para la rotacion y traslación por separado se tiene que:

$$R = R_r(R_l)^T \quad (2.21)$$

$$T = T_r - RT_l \quad (2.22)$$

Ya que R y T serán ligeramente diferentes cada vez que se toma una imagen, los valores medios son optimizados y mejorados usando el algoritmo de Levenber-Marquardt, el cuál está incluido en las librerías de OpenCV.

2.5. Resultados de la calibración para el Kinect.

Para este proceso se usa el software RGBDemo, el cuál se encarga de tomar las respectivas imágenes de una superficie plana con un patrón ajedrezado en diferentes posiciones, como se muestra a continuación.

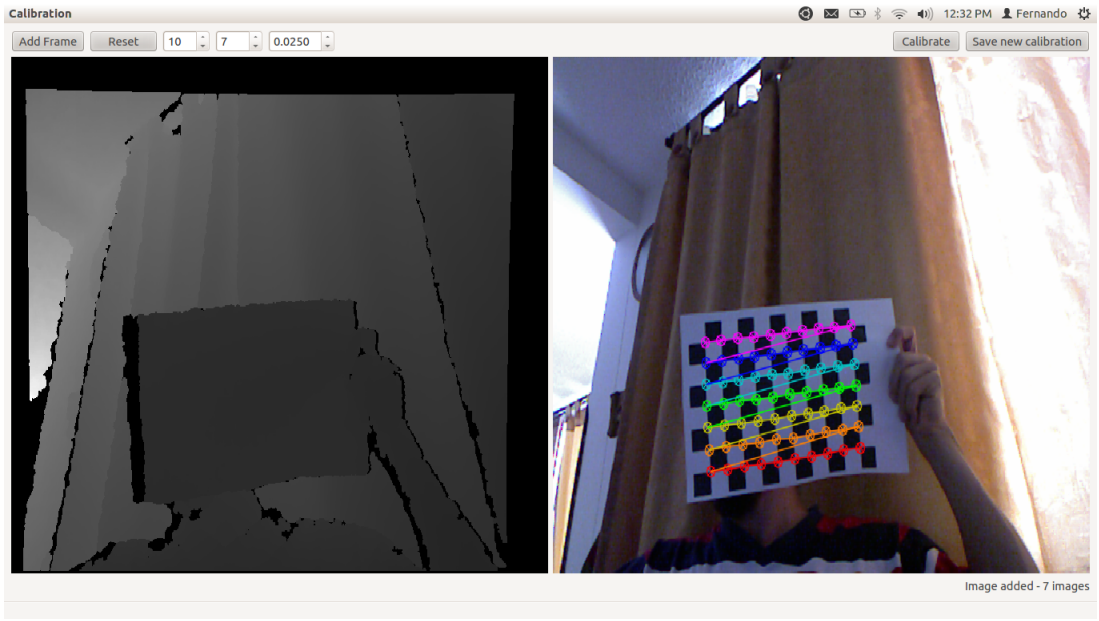


FIGURA 2.9. *RGBDemo - Calibración*

Procedimiento

- Se construye un patrón ajedrezado con cuadrados de 31 [mm]
- Se asegura que el objetivo esté bien iluminado sin reflexiones especulares
- Se capturan imágenes del patrón ajedrezado por la cámara RGB y por la cámara IR.
- Se cambia la posición y orientación del patrón ajedrezado, para repetir el procedimiento.

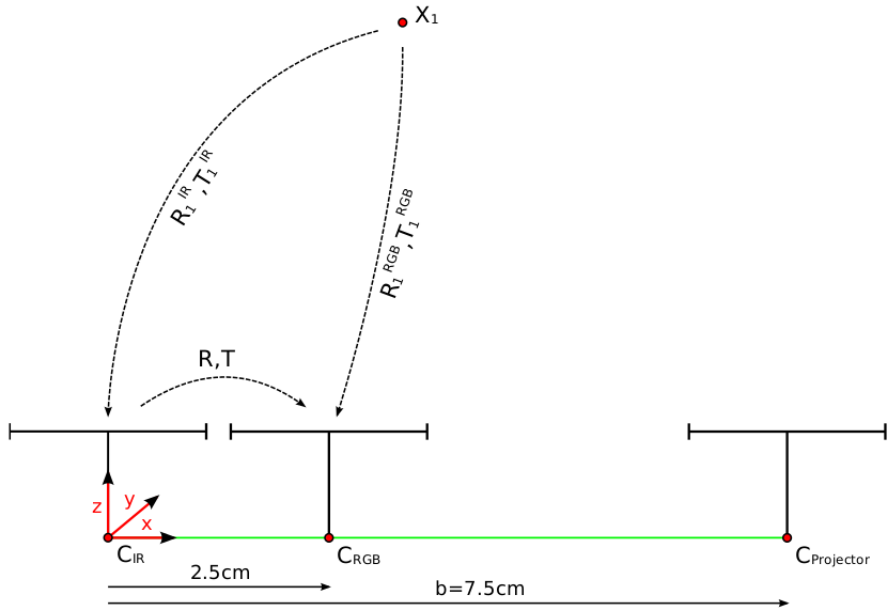
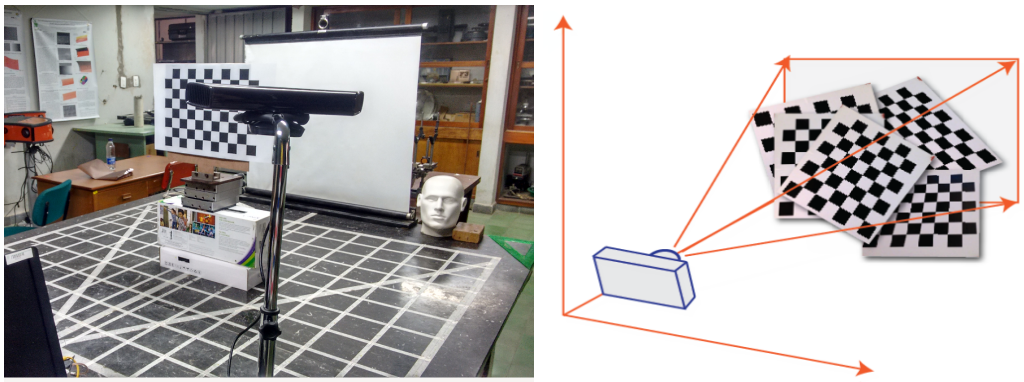


FIGURA 2.10. *Esquema de Cámaras del dispositivo Kinect. Fuente: [25]*

Un esquema de las cámaras que posee el Kinect se ve representado en la figura 2.10, en el esquema se pueden observar los diferentes parámetros que se deben conocer, así como las transformaciones que deben realizarse para llevar a cabo la calibración adecuadamente. La tabla 2.1 muestra los parámetros que fueron calculados para un Kinect determinado.



(a) *Montaje experimental para la calibración de un Kinect*

(b) *Representación general de la toma de imágenes*

FIGURA 2.11. Montaje experimental y representación de la toma de imágenes

Como se mencionó anteriormente, se empleó el modelo propuesto por Zhang para calcular los parámetros intrínsecos y extrínsecos de la cámara RGB e IR. La figura 2.12 muestra el esquema experimental. Para una posición del tablero de ajedrez se adquiere una imagen. Un punto de control (esquina de un cuadrado) se conocen sus coordenas en el sistema coordenado el objeto $P_i(x_i, y_i, 0)$ donde x_i y y_i están relacionados con el número de cuadrados y el tamaño de cada cuadrado. Por tratamiento digital de imágenes se calculan las coordenadas en pixeles sobre la imagen, que corresponden a las coordenadas $p_i(x_i, y_i)$ experimentales. Asignando valores a los parámetros intrínsecos y parámetros extrínsecos se calculan las coordenadas en pixeles teórica $p_{ti}(x_{ti}, y_{ti})$ para el punto de coordenadas $p_i(x_i, y_i, 0)$ según ecuación 2.5, teniendo en cuenta las distorsiones geométricas. Se define la función de error:

$$f = \sum_{i=1}^{N} (x_i - x_{ti})^2 - (y_i - y_{ti})^2 \tag{2.23}$$

La ecuación 2.23 define la función de error que al minimizarla por procesos de optimización, permiten calcular los parámentros intrínsecos y los parámentros extrínsecos. A continuación se muestran los P.I y P.E. de las cámaras RGB e IR.

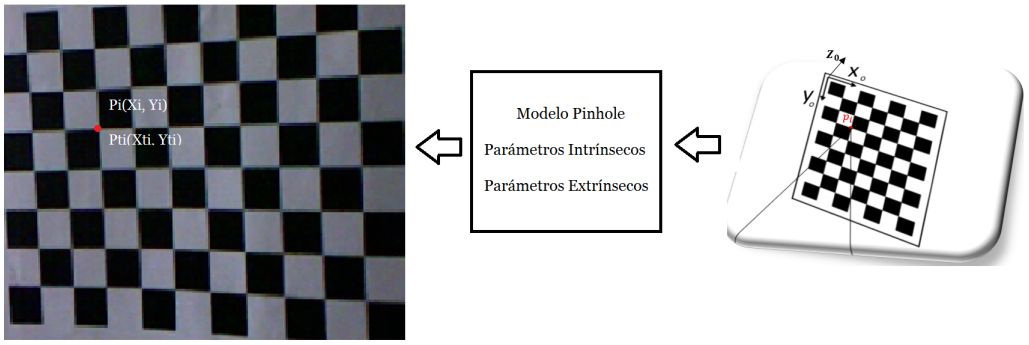


FIGURA 2.12. *Modelo de Zhang*

Cámara RGB

Parámetro	Descripción
$f_{Xrgb}, f_{Yrgb}, c_{Xrgb}, c_{Yrgb}$	Parámetros intrínsecos de la cámara RGB
$k_{1rgb}, k_{2rgb}, k_{3rgb}, p_{1rgb}, p_{2rgb}$	Coeficientes de distorsión de la cámara RGB

TABLA 2.1. Parámetros de calibración cámara RGB

Para la cámara RGB se obtuvieron los siguientes parámetros intrínsecos.

```
image_width: 640
image_height: 480
rgb_intrinsics: !!opencv-matrix
camera_matrix:
   rows: 3
   cols: 3
   data: [ 5.2921508098293293e+02, 0., 3.2894272028759258e+02, 0.,
       5.2556393630057437e+02, 2.6748068171871557e+02, 0., 0., 1. ]
distortion_coefficients:
   rows: 1
   cols: 5
   data: [ 2.6451622333009589e-01, -8.3990749424620825e-01,
       -1.9922302173693159e-03, 1.4371995932897616e-03,
       9.1192465078713847e-01 ]
```

FIGURA 2.13. *Valores obtenidos para los parámetros intrínsecos de la cámara RGB, salida de RGB-Demo*

De forma matricial las distancias focales y el centro de proyección en pixeles se muestra como:

$$I_{rgb} = \begin{pmatrix} f_{x_{rgb}} & 0 & c_{x_{rgb}} \\ 0 & f_{y_{rgb}} & c_{y_{rgb}} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 529{,}215 & 0 & 328{,}942 \\ 0 & 525{,}563 & 267{,}480 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.24}$$

Mientras que los coeficientes de distorsion mostrados en el vector

$$D_{rgb} = \begin{bmatrix} k_{1rgb} \\ k_{2rgb} \\ p_{1rgb} \\ p_{2rgb} \\ k_{3rgb} \end{bmatrix} \tag{2.25}$$

$$D_{rgb} = \begin{bmatrix} 0{,}26451 \\ -0{,}83990 \\ -0{,}00199 \\ 0{,}00143 \\ 0{,}91192 \end{bmatrix} \tag{2.26}$$

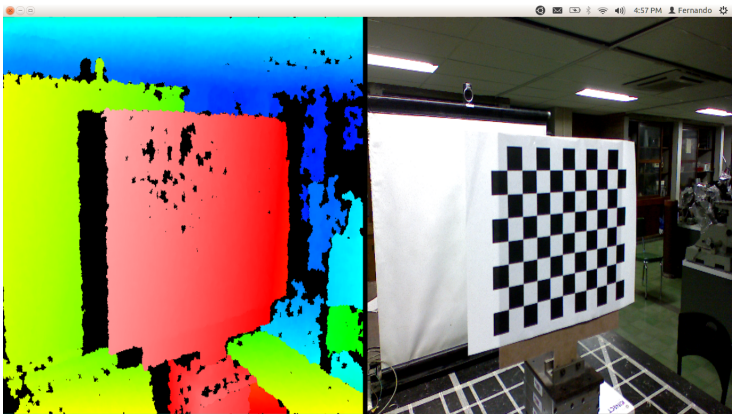


FIGURA 2.14. *Captura del patrón ajedrezado: (izquierda) mapa de profundidad (derecha) cámara RGB*

Cámara IR

Para la cámara IR se adquieren simultáneamente las mismas imágenes usadas para calibrar la cámara RGB como se mencionó anteriormente. Para que la cámara IR, pueda observar el patrón es necesario encender el laser infrarrojo, otros trabajos usan lámparas para iluminar el patrón y así poder identificar las esquinas de una manera más adecuada; para este trabajo se usó el mismo patrón de speckle para iluminar el patrón.

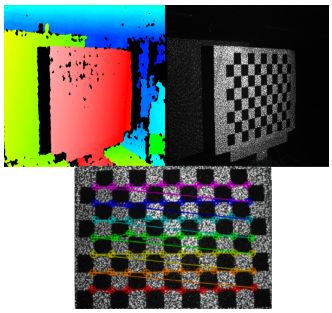
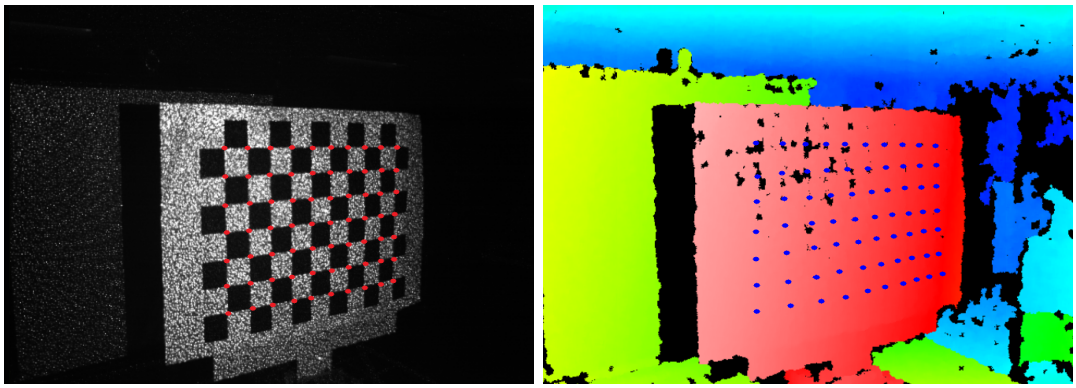


FIGURA 2.15. *Captura del patrón ajedrezado, en IR*

Una vez hecha la corrección mostrada en la ecuación 2.17, se tiene que las esquinas encontradas en el patrón ajedrezado coinciden con las mismas en la imágen de profundidad como lo muestra el ejemplo de la figura, 2.16. Realizando un procedimiento similar al anterior se define la función de error y se calculan los parámentros intrínsecos y extrínsecos de la cámara IR.



(a) *Esquinas en imagen IR (rojo)* (b) *Esquinas en imagen de profundidad (azul)*

FIGURA 2.16. Esquinas capturadas

Se tiene entonces que los parámentros a encontrar aparte de los parámentros intrínsecos son:

Parámetro	**Descripción**
$f_{Xir}, f_{Yir}, c_{Xir}, c_{Yir}$	Parámetros intrínsecos de la cámara IR
$k_{1ir}, k_{2ir}, k_{3ir}, p_{1ir}, p_{2ir}$	Coeficientes de distorsión de la cámara IR
c_1, c_0	Coeficientes de relación (IR-Z)

TABLA 2.2. Parámetros de calibración cámara IR

Para la cámara IR se obtuvieron los siguientes parámetros intrínsecos.

```
image_width: 640
image_height: 480
depth_intrinsics: !!opencv-matrix
camera_matrix:
   rows: 3
   cols: 3
   data: [ 5.9421480358642339e+02, 0., 3.3930546187516956e+02, 0.,
       5.9104092248505947e+02, 2.4273843891390746e+02, 0., 0., 1. ]
distortion_coefficients:
   rows: 1
   cols: 5
   data: [ -2.6389095690190378e-01, 9.9983033880181316e-01,
       -7.6323952014484080e-04, 5.0337278410637169e-03,
       -1.3056496956879815e+00 ]
```

FIGURA 2.17. *Valores obtenidos para los parámetros intrínsecos de la cámara IR*

De forma matricial las distancias focales y el centro de proyección en pixeles se muestra como:

$$I_{IR} = \begin{pmatrix} f_{x_{IR}} & 0 & c_{x_{IR}} \\ 0 & f_{y_{IR}} & c_{y_{IR}} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 594{,}214 & 0 & 339{,}305 \\ 0 & 591{,}040 & 242{,}738 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.27}$$

Mientras que los coeficientes de distorsion mostrados en el vector

$$D_{IR} = \begin{bmatrix} k_{1ir} \\ k_{2ir} \\ p_{1ir} \\ p_{2ir} \\ k_{3ir} \end{bmatrix} \tag{2.28}$$

$$D_{rgb} = \begin{bmatrix} -0{,}26389 \\ 0{,}99983 \\ -0{,}00076 \\ 0{,}00503 \\ -1{,}30564 \end{bmatrix} \tag{2.29}$$

Calibración en profundidad

Para cada posición de la cuadrícula tomada con la cámara IR también se calcula la disparidad correspondiente d_{raw} medida de la imágen de profundidad, es por esto que se usan las mismas imágenes para la adquisición de los coeficientes de la ecuación 2.16, escrita como:

$$\frac{fb}{z} = d_{raw}(c_1) + c_0 \tag{2.30}$$

Al realizar la corrección de posición según la ecuación 2.17, a cada punto de control de la imagen de la cuadrícula se conoce el valor de disparidad d_{raw}. Para el mismo punto, del proceso de calibración de la cámara, usando el algoritmo de Zhang, se conocen sus coordenadas (x_i, y_i, z_i). Usando z_i y d_{raw} para cada punto de control de cada imagen adquirida del tablero de ajedrez se calculan los coeficientes de la ecuación 2.30, usando regresión lineal por mínimos cuadrados entre $1/z$ vs d_{raw}

Con un total de 30 imágenes del patrón ajedrezado (con 70 posiciones de esquinas detectadas en cada uno), se tiene un total de 2100 puntos que permiten calcular por regresión lineal los parámetros de la ecuación 2.30.

Parámetro encontrado del ajuste	**Valor**
c_1	-0.0013
c_0	1.813

TABLA 2.3. Parámetros de calibración en profundidad

Finalmente, aunque la distancia entre centros ópticos b puede ser conocida de fábrica, se puede obtener del proceso de modelización; se encontró una distancia de $b = 74{,}45\ [mm]$.

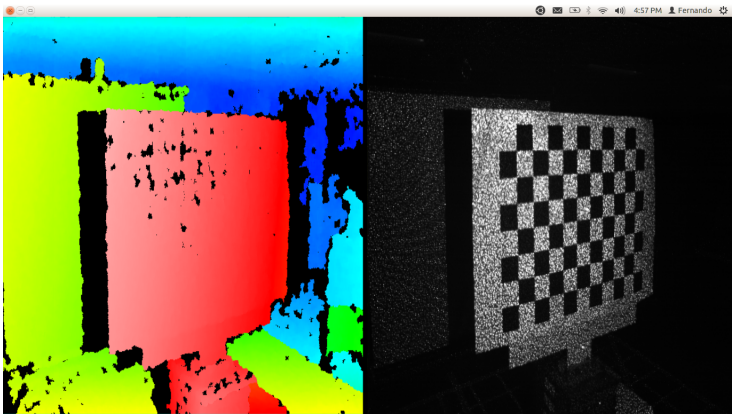


FIGURA 2.18. *Captura del patrón ajedrezado, (izquierda) mapa de profundidad (derecha) cámara IR*

Parámetros extrínsecos

Cómo se mencionó anteriormente, para recibir la información de profundidad y la de la cámara cromática es necesario encontrar la rotación y traslación del sistema que una el par estéro (cámara IR y cámara RGB) como lo muestra la siguiente figura.

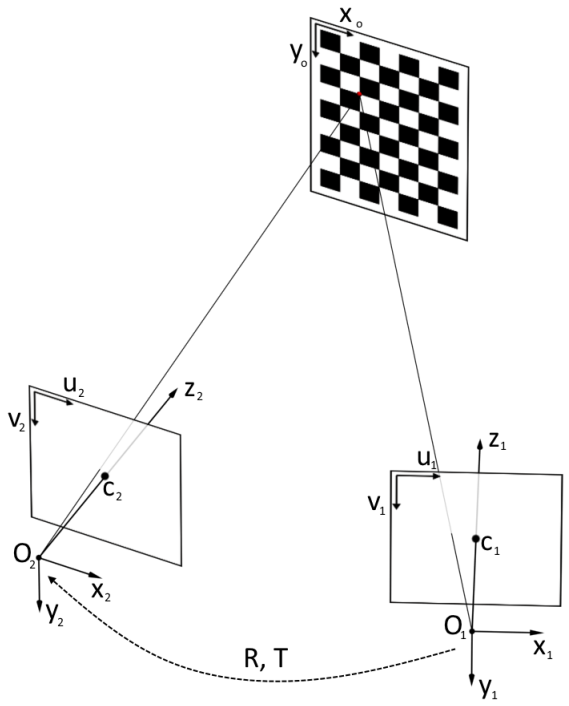


FIGURA 2.19. *Rotación y traslación entre sistemas de coordenadas* Fuente: http://developers-club.com/posts/272629/

Luego, los parámetros extrínsecos obtenidos se muestran a continuación:

```
R_extrinsics: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 5.8648559388189325e-01, 8.9531518657868153e-02,
       -8.0499612131725140e-01, -5.2326439540033264e-02,
       9.9598387093646346e-01, 7.2650344524174335e-02,
       8.0826764867466161e-01, -4.8579958191612149e-04,
       5.8881505764093700e-01 ]
T_extrinsics: !!opencv-matrix
   rows: 3
   cols: 1
   dt: d
   data: [ -6.7266696077708277e-01, 1.3881261006734612e-02,
       -1.4493200209349527e-01 ]
```

FIGURA 2.20. *Parámetros extrínsecos obtenidos*

En la representación matricial tenemos los parámetros de rotación :

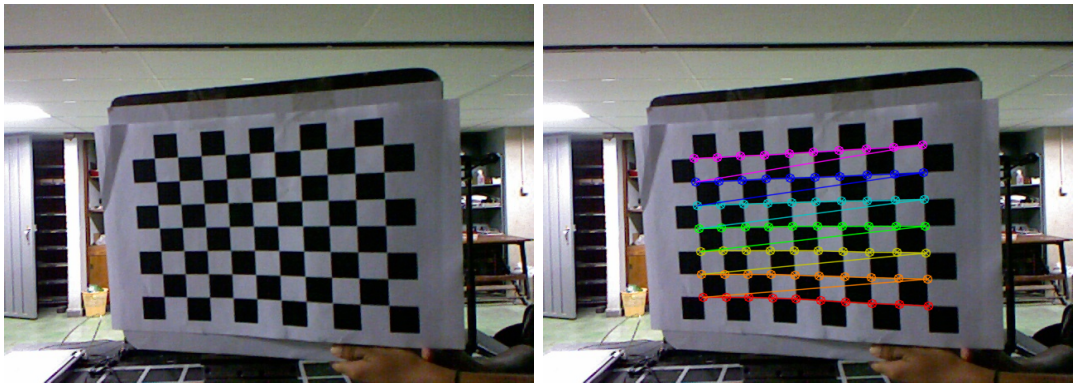
$$R_{kinect} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \tag{2.31}$$

$$= \begin{pmatrix} 0{,}58648 & 0{,}08953 & -0{,}80499 \\ -0{,}05232 & 0{,}99598 & 0{,}07265 \\ 0{,}80826 & -0{,}00048 & 0{,}58881 \end{pmatrix} \tag{2.32}$$

Igualmente para los parámetros de traslación:

$$T_{kinect} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{2.33}$$

$$T_{kinect} = \begin{bmatrix} -0{,}6726 \\ 0{,}0138 \\ -0{,}1449 \end{bmatrix} \tag{2.34}$$



(a) *Montaje experimental para la calibración de un Kinect*

(b) *Montaje experimental para la calibración de un Kinect*

FIGURA 2.21. Imágen para la stéro calibración

2.6. R3D de objetos

Una vez calibrado el sistema y almacenado sus parámetros, el kinect puede ser usado para digitalizar la superficie de objetos 3D. Los datos arrojados por el Kinect corresponden a una nube de puntos tridimensional que corresponden a los puntos vistos por la cámara IR en todo su campo de observación. Como el campo es amplio, se obtiene información tridimensional de objetos ubicados alrededor del objeto de estudio. Usando el software meshlab, los datos tridimensionales son filtrados para únicamente obtener la nube de puntos del objeto de estudio. La figura 2.22 (a) muestra la imagen de disparidad codificada cromáticamente del objeto (torso) mostrado en la figura 2.22 (b). La figura 2.23 muestra una imagen virtual (Render) obtenido de la nube de puntos con iluminación artificial. La figura 2.24 muestra una imagen virtual del cráneo humano. La ventaja del Kinect consiste en la velocidad de procesamiento: un sistema electrónico que posee un microprocesador dedicado al procesamiento de la imágen de triangulación. Del sistema se obtuvo una cadencia de nube de puntos 3D de 20-30 fps, donde cada ”frame” es una nube de puntos 3D.

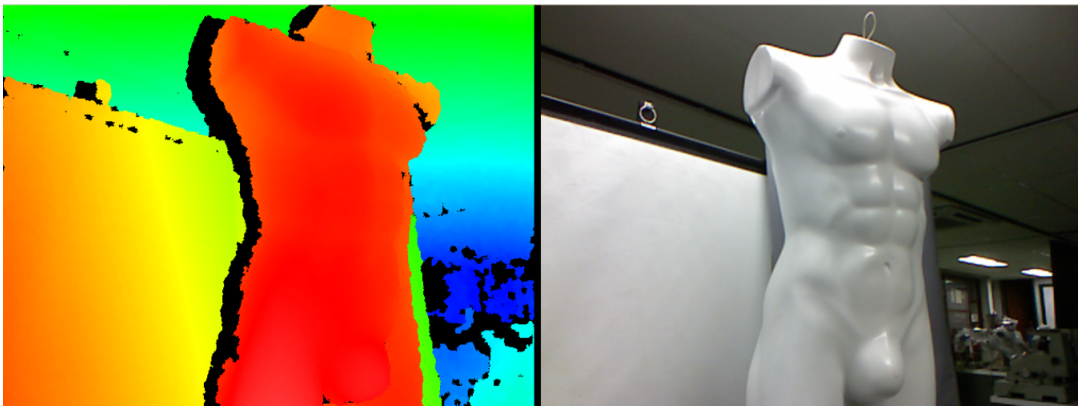


FIGURA 2.22. *(a) Vista del mapa de disparidad y el objeto dentro del volumen de calibración (b)*

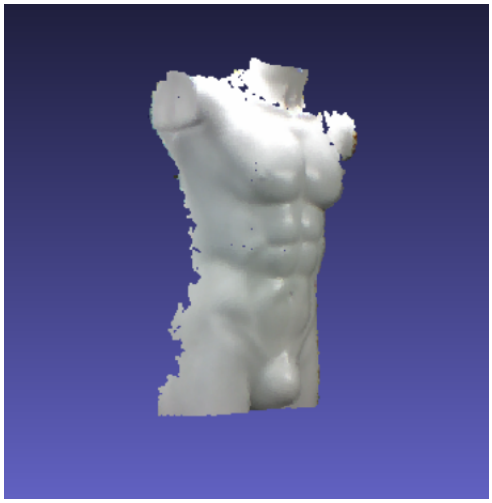


FIGURA 2.23. *Volumen de datos adquirido para una vista frontal*

FIGURA 2.24. *Vista frontal de un cráneo humano*

Exploración de la nube de puntos:

Una vez calibrado el kinect se convierte en un sistema cerrado que arroja nubes de puntos 3D de un objeto que se ubique en su campo de observación, a una cadencia de 20-30 Fps. Es importante por lo tanto estudiar en detalle la nube de puntos arrojada por el Kinect. Para la figura 2.23 se obtiene el histograma de valores Z de la nube de puntos. La figura 2.25 muestra el histograma calculado para 10^6 valores en el rango z de -1.25 a -0.95 [m].

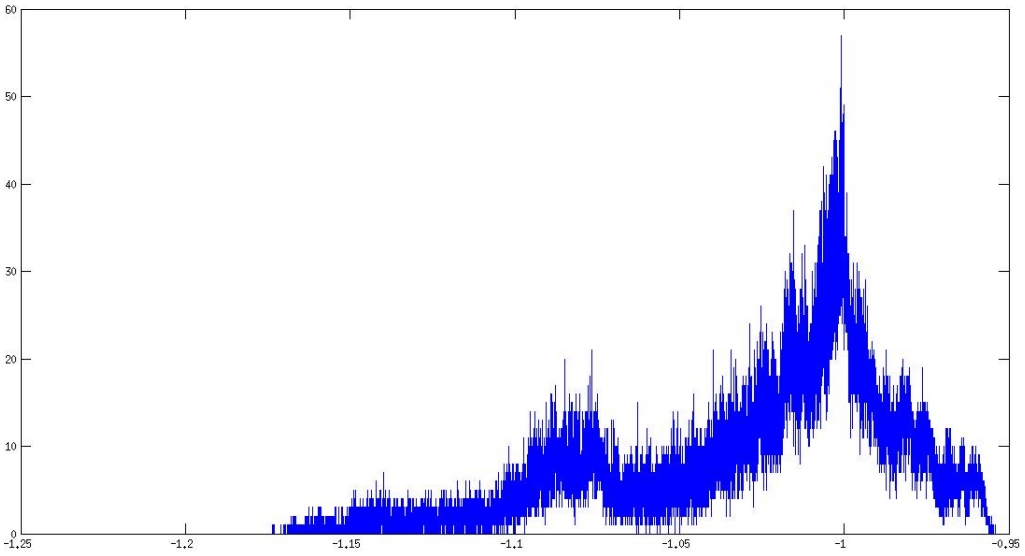


FIGURA 2.25. *Discretización observada en una figura 3D* Histograma: Distancias Z vs Conteo de datos

La figura 2.26 muestra un zoom del histograma, donde se pueden evidenciar que los valores en z están discretizados. Es decir para los 10^6 valores en z estudiados, solamente hay alrededor de 30.000 únicos valores en el rango de valores de z del objeto. El paso siguiente consiste en calcular el ΔZ de muestreo. Usando el histograma de valores z y haciendo la diferencia para los valores de z encontrados se obtiene un valor de $\Delta Z = 0{,}010$ [mm].

Indudablemente la discretización en z está asociada a la forma matemática de cómo se calcula la imagen de disparidad. El algoritmo de correlación empleado no arroja posiciones decimales en pixeles.

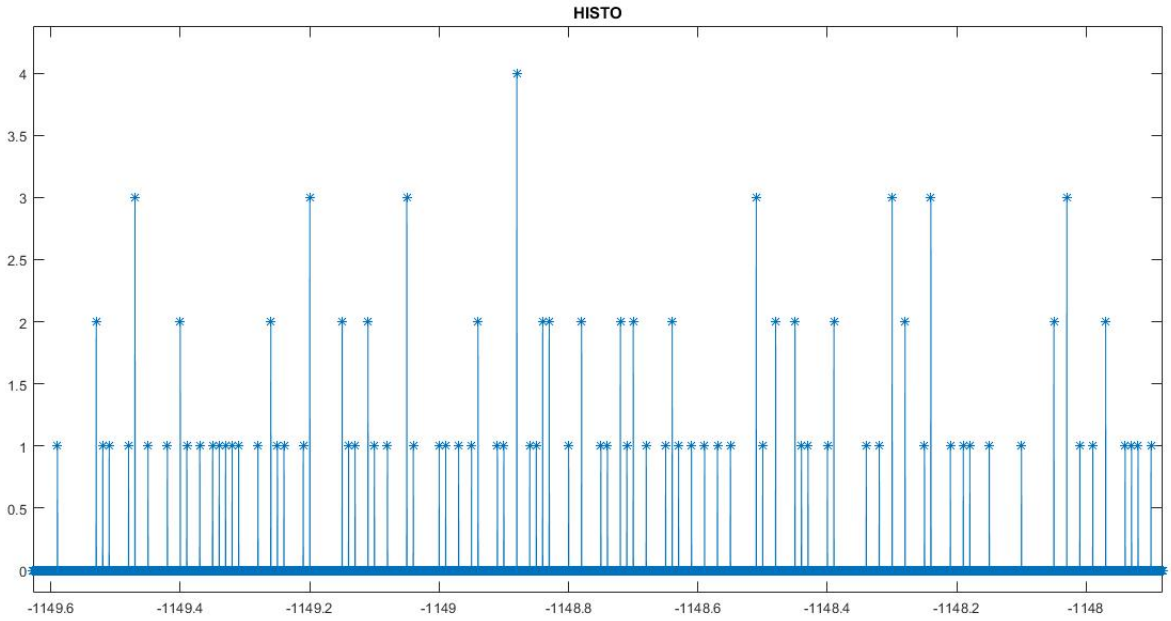


FIGURA 2.26. *Discretización para una región del torso* Histograma: Distancias Z vs Conteo de datos

CAPÍTULO 3

Registro 3D de nubes de puntos

Introducción:

Una ventaja interesante que posee el Kinect es el tiempo empleado para obtener la nube de puntos. Actualmente en el GOTS se presentan exigencias metrológicas para el sector médico en donde no interesa mucho la resolución, pero sí la obtención de información 3D del objeto completo a 360° de observación. El uso de un Kinect o múltiples con el fin de obtener información 3D de las diferentes posiciones angulares del objeto, permite obtener información completa del mismo. Para esto se requiere el uso de un algortimo que permita unificar las diferentes vistas 3D del objeto en un único sistema coordenado, este proceso se llama registro de información y bibliográficamente se usa el algoritmo ICP (Iterative Closest Points).

Las nubes de puntos que se obtienen del dispositvo Kinect pueden ser tomadas desde diferentes posiciones/ángulos y por diferentes dispositivos con el fin de obtener un modelado 3D. Cada vista del objeto debe ser comparada con la anterior para poder combinar las nubes de puntos en un mismo sistema de referencia; para esto es necesario encontrar la transformación que combine las nubes de puntos. El proceso de encontrar la transformación adecuada es conocido como registro de la nube de puntos, repetir el mismo proceso para diferentes nubes de puntos permitirá obtener una imagen virtual o render de un objeto 3D.

Dependiendo de los datos suminstrados por el sistema de adquisición, como lo puede ser el color, existen varios algoritmos para la integración de nubes de puntos. Estos usan algunos puntos característicos en la imagen y varios puntos descriptores, los más comúnmente conocidos son SIFT [28] y FAST [9]. Los puntos característicos poseen información altamente distintivas y suministran una transformación inicial bastante aceptable, pero suelen ser métodos que conllevan un costo computacional más alto que los métodos convencionales.

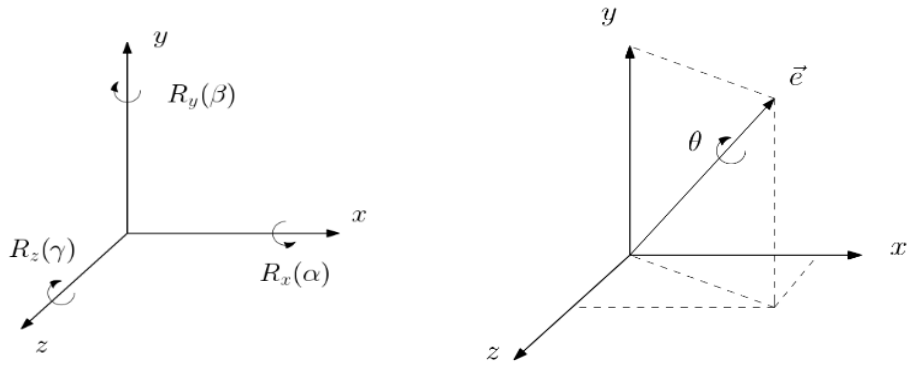
3.1. Transformaciones

Dado que el objetivo principal para unir dos nubes de puntos es necesario entender las transformaciones usadas para llevar acabo este proceso, así un punto en tres dimensiones se puede representar de la siguiente forma

$$p_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \tag{3.1}$$

Con un conjunto de puntos se tiene que

$$P = \{p_i | i = 1, \cdots N\} \tag{3.2}$$



(a) *Rotación alrededor del eje x* (α)*, eje y* (β)*, eje z* (γ)

(b) *Rotación alrededor de un eje cualquiera*

FIGURA 3.1. Sistemas coordenados. Fuente: 3D reconstruction using Kinect v2 camera - Lembit Valgma

Siendo N el número total de puntos. Dado un punto en el espacio, la rotación de este alrededor de un eje coordenado se pueden expresar independientemente usando las matrices de rotacion alrededor de cada uno de los ejes (ver figura 3.1 (c)) cartesianos:

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \tag{3.3}$$

$$R_y(\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \tag{3.4}$$

$$R_z(\gamma) = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.5}$$

Luego la matriz que representa la rotación total se puede escribir como:

$$R = R(\alpha, \beta, \gamma) = R_z(\gamma) R_y(\beta) R_x(\alpha) \tag{3.6}$$

$$R(\alpha, \beta, \gamma) = \begin{pmatrix} \cos\beta\cos\alpha & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \cos\beta\sin\gamma & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma \\ -\sin\beta & \sin\alpha\cos\beta & \cos\alpha\cos\beta \end{pmatrix} \tag{3.7}$$

Las rotaciones pueden también ser expresadas a lo largo de un eje arbitrario, este eje es definido por un vector $\vec{e}$ (ver figura 3.1 (d)) y el monto de la rotación por un ángulo θ, puede ser expresado como:

$$\langle \vec{e}, \theta \rangle = \big((e_x, e_y, e_z)^T, \theta\big) \tag{3.8}$$

Una forma más adecuada de escribir las transformaciones es através de los cuaterniones [1] el cuál es definido cómo un número complejo de cuatro dimensiones de la siguiente manera:

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{3.9}$$

Donde q_0,q_1,q_2,q_3 son reales, su cuadrado se define como:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{3.10}$$

Usualmente los cuaterniones unitarios pueden ser usado para representar una rotación. Dada la representación de la ecuación 3.8 la unidad del quaternion correspondiente es expresado como:

$$q = \cos\frac{\theta}{2} + e_x \sin\frac{\theta}{2} i + e_y \sin\frac{\theta}{2} j + e_z \sin\frac{\theta}{2} k. \tag{3.11}$$

así para un punto $p = (x, y, z)^T$ su cuaternion respectivo será de la forma:

$$p_q = 0 + xj + yj + zk \tag{3.12}$$

Y la representación de una rotación en términos del cuaternion unitario $q = q_0 + q_1 i + q_2 j + q_3 k$ puede ser representada en la siguiente matriz.

$$R(q) = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_2 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \tag{3.13}$$

No sólo es necesario rotar una nube de puntos para generar un empalme, también es preciso trasladarlas, así la traslación en el espacio tridimensional puede ser expresado como el vector:

$$\vec{t} = (t_x, t_y, t_z) \tag{3.14}$$

Donde t_x,t_y,t_z representan la traslación lo largo de los ejes x,y y z respectivamente, así las coordenadas del punto trasladado serán:

$$p_2 = p_1 + \vec{t} \tag{3.15}$$

3.2. Algoritmo ICP (Iterative Closest Point)

El método iterativo del punto más cercano ha sido usado satisfactoriamente en diferentes estudios con el fin de obtener el modelo tridimensional de escenas/personas, ver [14],[17], [3].

[1] https://es.wikipedia.org/wiki/Cuaternión

El algortimo ICP constituye una herramienta computacional óptima para la integración de diferentes nubes de puntos, éste posee distintas variantes [24]. El objetivo básico del algoritmo consiste en integrar dos nubes de puntos a un sistema coordenado determinando pares de puntos correspondientes con el fin de estimar la transformación que minimiza las distancias entre las correspondencias y finalmente aplicar la transformación que alinea la nube de puntos inicial con su objetivo.

Usualmente se tienen dos nubes de puntos, a la nube de puntos a la cuál se le aplica la transformación se le conoce como puntos de referencia, y la otra nube de puntos se le conoce como puntos modelo. Luego, los puntos de referencia son denotados por $P = \{p_i | i = 1, \cdots N\}$ mientras que los puntos modelos son denotados como $M = \{m_i | i = 1, \cdots N\}$, dado esto, matemáticamente el objetivo se traduce en minimizar la siguiente función métrica e

$$e = f(M, TP) \tag{3.16}$$

La suma de cuadrados es la métrica usada frecuentemente en la aplicación del algoritmo ICP. Así dada la correspondencia entre dos puntos p_i y m_i, e puede ser definido como:

$$e = \frac{1}{N} \sum_{i=1}^{N} \|m_i - Tp_i\|^2 \tag{3.17}$$

Dado que al tomar los datos con el dispositivo Kinect la forma de los objetos no cambia, solo la rotación y la traslación deben ser considerados, así la ecuación anterior se puede escribir

$$e = \frac{1}{N} \sum_{i=1}^{N} \left\|m_i - Rp_i - \vec{t}\right\|^2 \tag{3.18}$$

Esta aproximación es útil siempre y cuando exista una correspondencia de puntos entre las dos nubes de puntos. Es decir, para cada $i = 1 \cdots, N$ se conoce cuál p_i es transformado a algún m_i, esto es un gran problema para la toma de datos del dispositivo Kinect, ya que este no proporciona tal correspondencia, solamente arroja nube de puntos burdos.

En [2] se propone una mejora al algoritmo ICP para buscar los puntos correspondientes usando los puntos más cercanos antes de generar la transformación. Básicamene se usan puntos en común para acercar las nubes de puntos tanto como sea posible, luego se procede a generar la transformación y a aplicarla sobre los puntos de referencia.

El algoritmo trabaja de la siguiente forma:

1. Datos de referencia P y datos modelo M son suministrados.
2. Inicia la iteración estableciendo $P_0 = P, R = I, \vec{t} = (0, 0, 0)$ y $k = 0$. Si existe alguna estimación inicial para la rotación y traslación, se usan estos para iniciar. Los pasos 3-6 se repiten hasta que la convergencia tenga una toleracia τ
3. Computa los puntos más cercanos Y_k para P_k en M
4. Encuentra la transformación óptima T_k entre P_0 y Y_k

5. Aplica la transformación $P_{k+1} = T_k P_0$

6. Encuentra el error e_k entre Y_k y M. Termina la iteración si el error desciende por debajo del umbral. Es decir si $e_k - e_{k+1} < \tau$

Dadas las nubes de puntos correspondientes $P = p_i$ y $M = m_i$ el objetivo es encontrar la rotación R y la traslación $\vec{t}$ tal que la ecuación 3.18 sea minimizada. Se denota el centro de masa μ_p para los datos de referencia P y μ_m para los datos modelo M, los cuales pueden ser calculados de la siguiente forma:

$$\mu_p = \frac{1}{N}\sum_{i=1}^{N} p_i, \qquad \mu_m = \frac{1}{N}\sum_{i=1}^{N} m_i \tag{3.19}$$

La matriz de covariancia $\sum pm$ de los conjuntos P y M se define como

$$\sum pm = \frac{1}{N}\sum_{i=1}^{N}\left[(p_i - \mu_p)(m_i - \mu_m)^T\right] = \frac{1}{N}\sum_{i=1}^{N}\left[p_i m_i^T\right] - \mu_p \mu_m^T \tag{3.20}$$

Los componentes cíclicos de la matriz antisimétrica $A_{ij} = (\sum pm - \sum pm^T)_{ij}$ son usados para formar el vector columna $\Delta = [A_{23} A_{31} A_{12}]^T$. Este vector es usado para formar la matriz simétrica 4x4 $Q\left(\sum pm\right)$.

$$\mathcal{Q}\left(\sum pm\right) = \begin{pmatrix} traza\left(\sum_{pm}\right) & \Delta^T \\ \Delta & \sum_{pm} + \sum_{pm}^T - traza\left(\sum_{pm}\right) I_3 \end{pmatrix}, \tag{3.21}$$

donde I_3 es la matriz identidad 3x3. El eigenvector $q = (q_0, q_1, q_2, q_3)^T$ correspondiente con el máximo eigenvalor en la matriz anterior es seleccionado como la representación del cuaternion de la rotación óptima. La rotación $R(q)$ puede ser construida usando 3.13, y el vector de translación óptimo puede ser hayado usando la matriz de rotación y los centros de masa como

$$\vec{t} = \mu_m - R(q)\mu_p \tag{3.22}$$

Existen varios algoritmos libres que implementan el algoritmo ICP para el registro de datos 3D, En este trabajo se usó Meshlab [2], el cuál permite integrar dichas nubes de puntos. El software Meshlab toma como referencia mínimo 4 puntos correspondientes en las nubes de puntos, con el fin de acercarlas lo mejor posible para poder aplicar la transformación.

Otra manera usada fue a través de las librerías de PCL [3] proporcionan una manera fácil y adecuada de unir las nubes de puntos, a continuación se muestra el código básico para el registro de dos nubes de puntos.

[2] https://en.wikipedia.org/wiki/MeshLab
[3] http://pointclouds.org/

```
// ...
pcl::IterativeClosestPoint<pcl::PointXYZ, pcl::PointXYZ> icp;
icp.setInputCloud (cloud1);
icp.setInputTarget (cloud2);
icp.setMaximumIterations (20);
icp.setMaxCorrespondenceDistance (0.1);
icp.align (*cloud2);
// ...
```

3.3. Variantes al algoritmo ICP

La gran utilidad que muestra el algoritmo ICP a la hora de unir diferentes nubes de puntos ha aumentado su popularidad en diferentes escenarios, debido a su utilidad muchas modificaciones a los pasos de los algoritmos han sido propuestas, aquí se resumen dichas modificaciones.

1. Selección de puntos de los datos de entrada
Cuando los datos de entrada son voluminosos se aumenta el costo computacional, así, en vez de usar todoso los datos de referencia y los datos modelo, sub-seccionar los datos podría ser útil, reducir la nube de puntos tomando en cuenta diversas características como lo puede ser la selección de puntos con altos gradientes.

2. Haciendo coincidir los puntos muestreados en la otra nube de puntos

Diferentes métricas e pueden ser usadas para encontrar los puntos correspondientes en la otra nube de puntos, estas posibles variantes incluyen la proyección de los datos de referencia sobre los datos modelo para luego buscar en un rango más pequeño, ésta operación es una de las que más consume costo computacional a lo largo del proceso ICP.

3. Ponderar adecuadamente los pares correspondientes

Cuando el grupo de parejas es encontrado, se procede a asignar los pesos a cada una de ellas, para esto algunas mejoras proponen pesos constantes, así como lo pueden ser pesos bajos con alta distancia de punto a punto. También la asignación de un peso que pueda depender de la compatibilidad de las normales.

4. Rechazar algunos pares de parejas basados en la observación de cada par individual

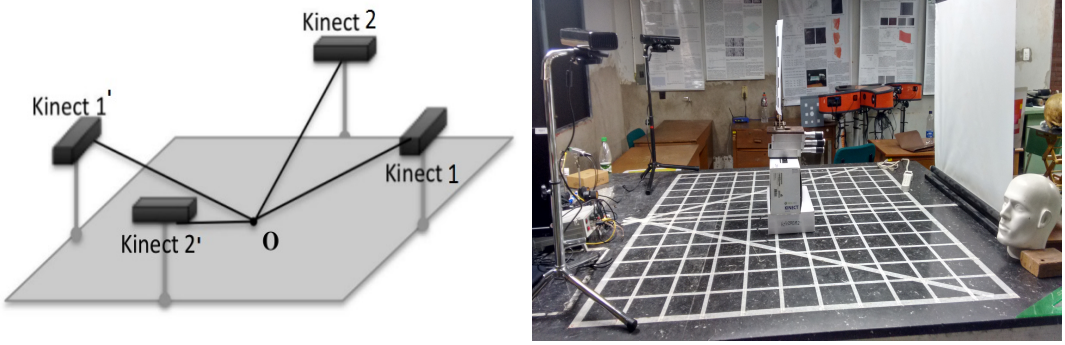
Esta mejora ayuda a reducir efectivamente los valores atípicos, para poder usar este criterio es necesario la asignación de pesos de la etapa anterior. Así rechazar un porcentaje dependiendo de las peores parejas, como también puede ser rechazar parejas que coincidan con parejas vecinas.

5. Asignación y minimización de la métrica del error

Para la determinación de la métrica del error se ha propuesto la suma del cuadrado de las distancias entre parejas de puntos; así como también la suma del cuadrado de las distancias desde cada punto origen, hasta el plano que contiene el punto destino y que está orientado perpendicularmente a la normal del destino.

3.4. Resultados

Para el montaje experimental se usaron las instalaciones del laboratorio de óptica y tratamiento de señales, GOTS, dado que las condiciones de iluminación afectan el Kinect, el lugar es ideal para la toma de datos. Se hicieron reconstrucciones de diferentes objetos, como lo es el rostro humano, una figura geométrica cúbica;4 para el rostro se tomaron 4 diferentes perspectivas (tomas 1',2',3',4') como se muestra a continuación



(a) *Montaje experimental elaborado, tomado de [18]*

(b) *Montaje experimental GOTS, dos Kinects*

FIGURA 3.2. Montaje experimental

El montaje de la figura 3.2 (d) es rotado para conseguir las dos tomas de datos restantes, para el caso del rostro se observa el mapa de disparidad generado en la figura 3.3

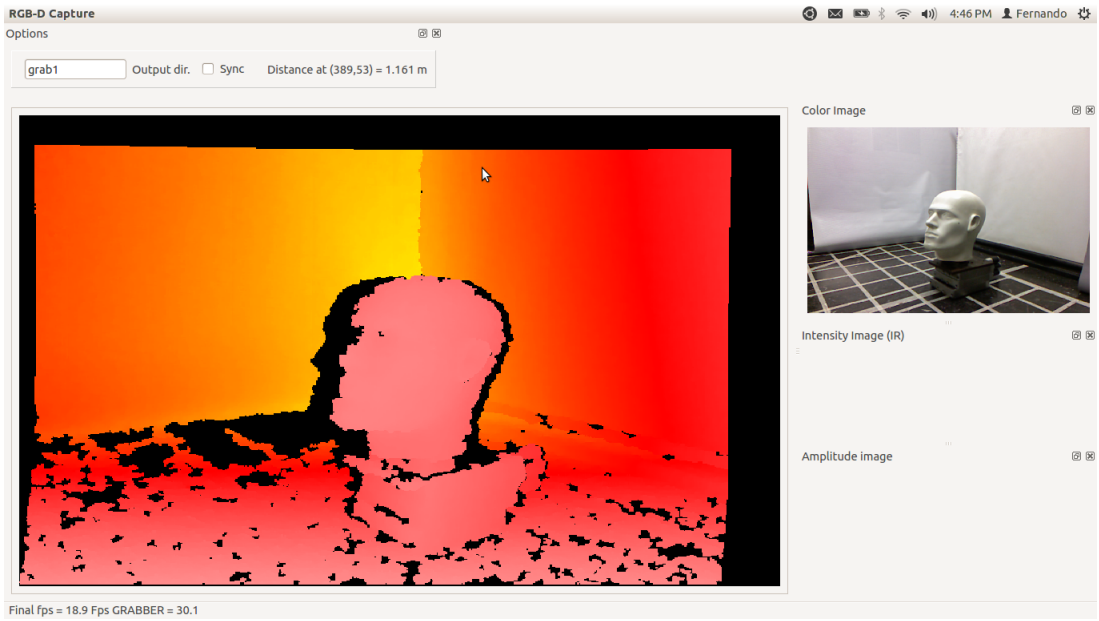


FIGURA 3.3. *Vista lateral del rostro desde un Kinect*

Las vistas pueden ser fácilmente procesadas para cada dispositivo desde el software RGBDemo, a continuación se muestran los datos extraídos desde diferentes perspectivas.

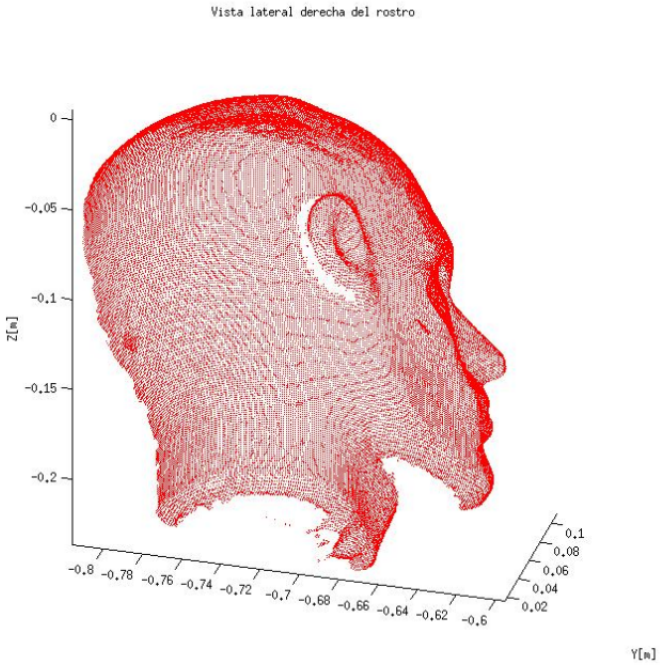


FIGURA 3.4. *Vista lateral derecha del rostro*

Los datos fueron exportados a Matlab usando un script que pudiese convertir los datos arrojados por el dispositivo kinect en una matriz de 3xN, lo cuál para Matlab es de fácil manejo.

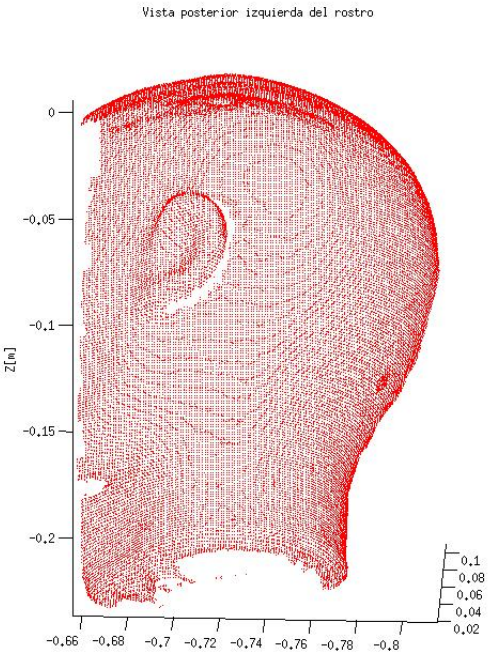


FIGURA 3.5. *Vista posterior derecha del rostro*

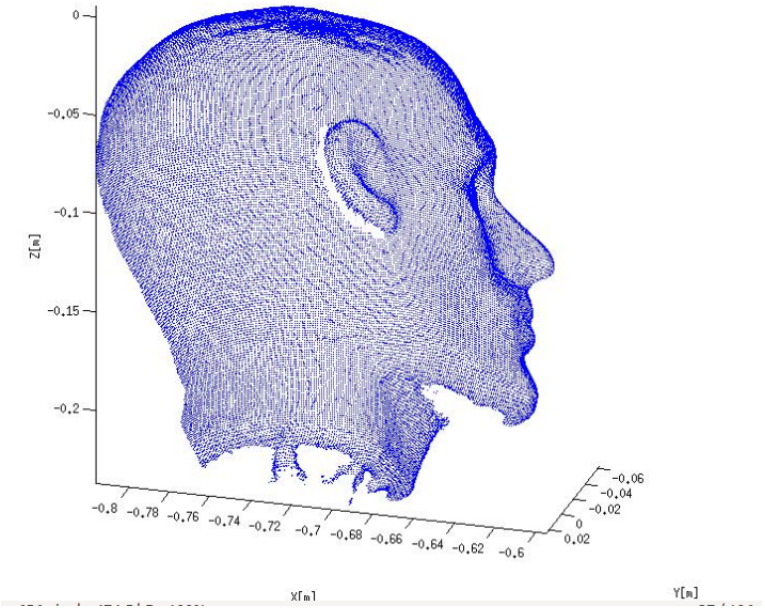


FIGURA 3.6. *Vista lateral izquierda del rostro*

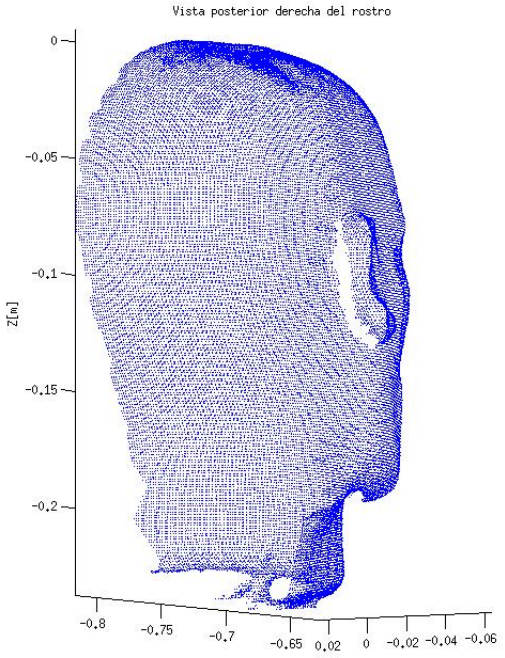


FIGURA 3.7. *Vista posterior derecha del rostro*

El software Meshlab tiene implementado el algoritmo ICP. Permite cargar las dos nubes de puntos y requiere de 4 puntos correspondientes mínimos seleccionados a mano en las 2 nubes de puntos 3D. Las figuras 3.8 y 3.9 muestra el proceso y los resultados obtenidos.

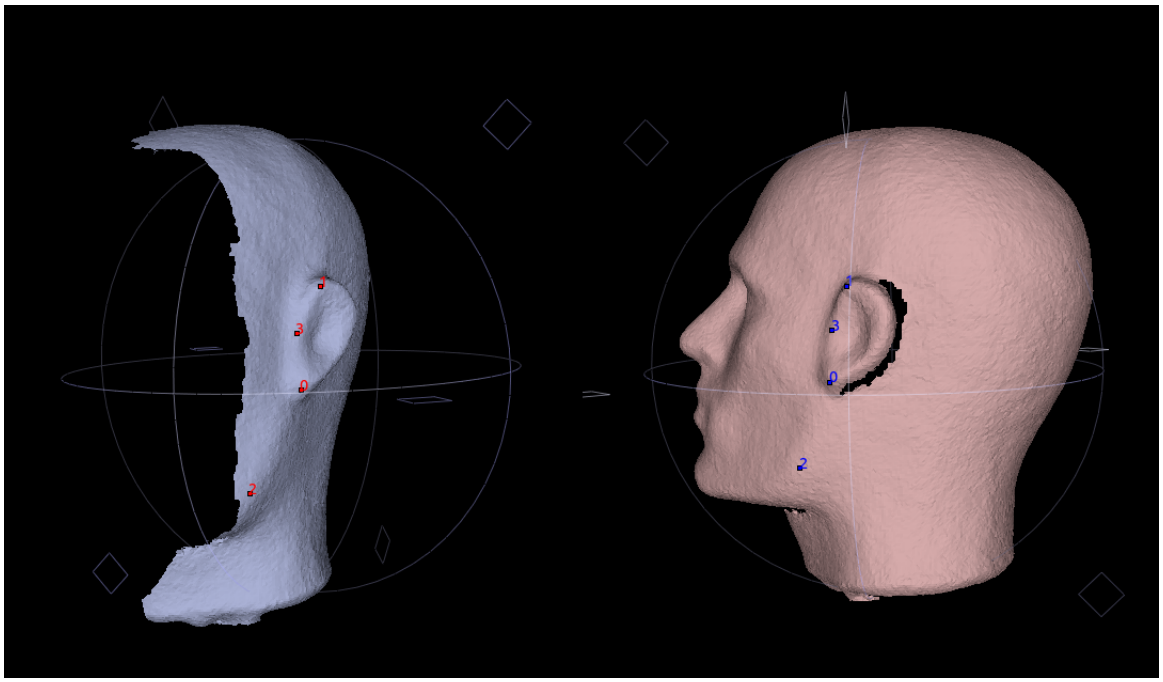


FIGURA 3.8. *Vista de Meshlab de dos nubes de puntos*

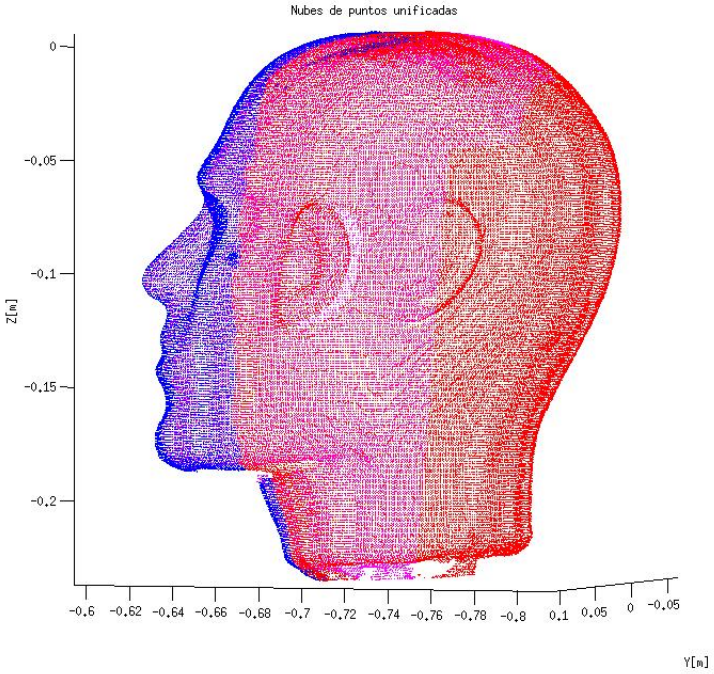
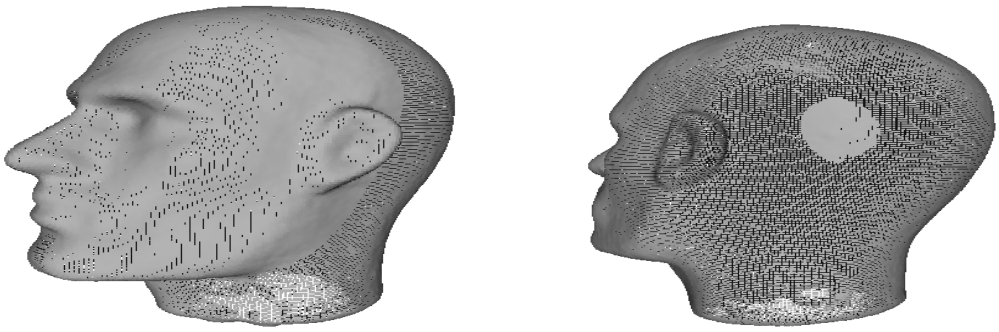
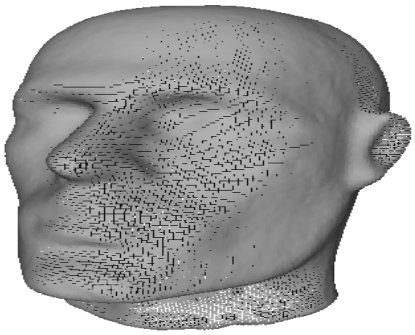


FIGURA 3.9. *Vista de las diferentes nubes de puntos unificadas*



(a) *Vista del rostro lateral*

(b) *Vista del rostro posterior*



(c) *Vista del rostro frontal*

FIGURA 3.10. Diferentes vistas del rostro

Una forma de analizar el error cometido en una reconstrucción tridimensional tomada con el Kinect a posteriori puede ser llevada a cabo reconstruyendo alguna figura geométrica con lados de distancias fijas. Las figuras siguientes muestran la reconstrucción tridimensional de un cubo de 260 [mm] por arista.

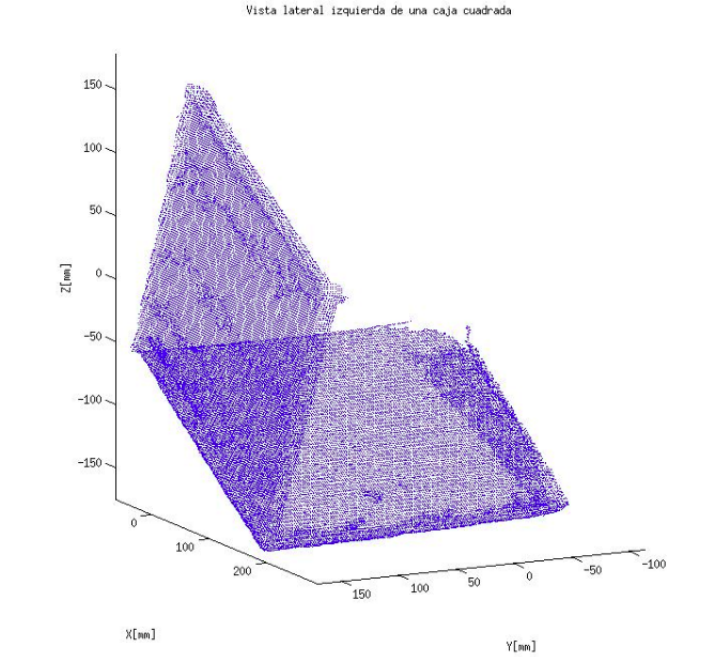


FIGURA 3.11. *Vista lateral izquierda de un cubo*

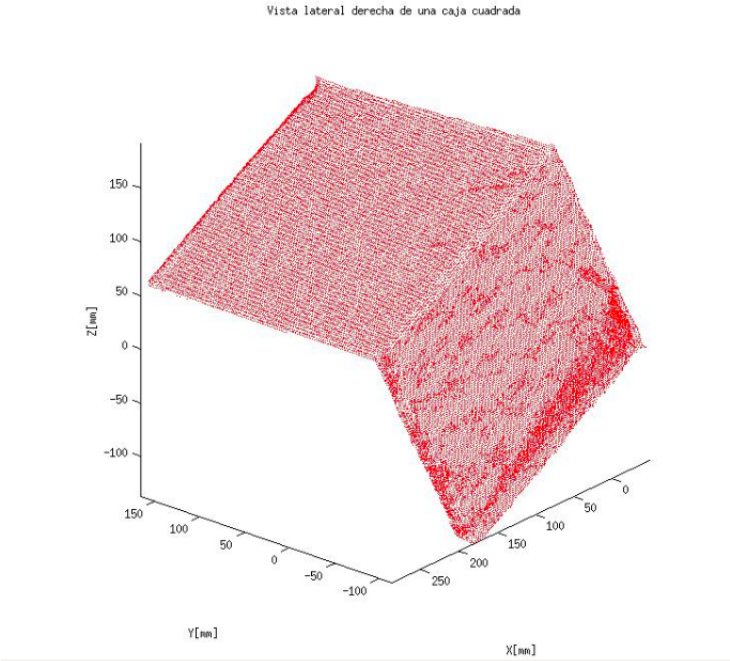


FIGURA 3.12. *Vista lateral derecha de un cubo*

Los datos unificados se muestran en la figura 3.13, como puntos coincidentes para la traslación y rotación incial se tomaron las aristas de el cubo.

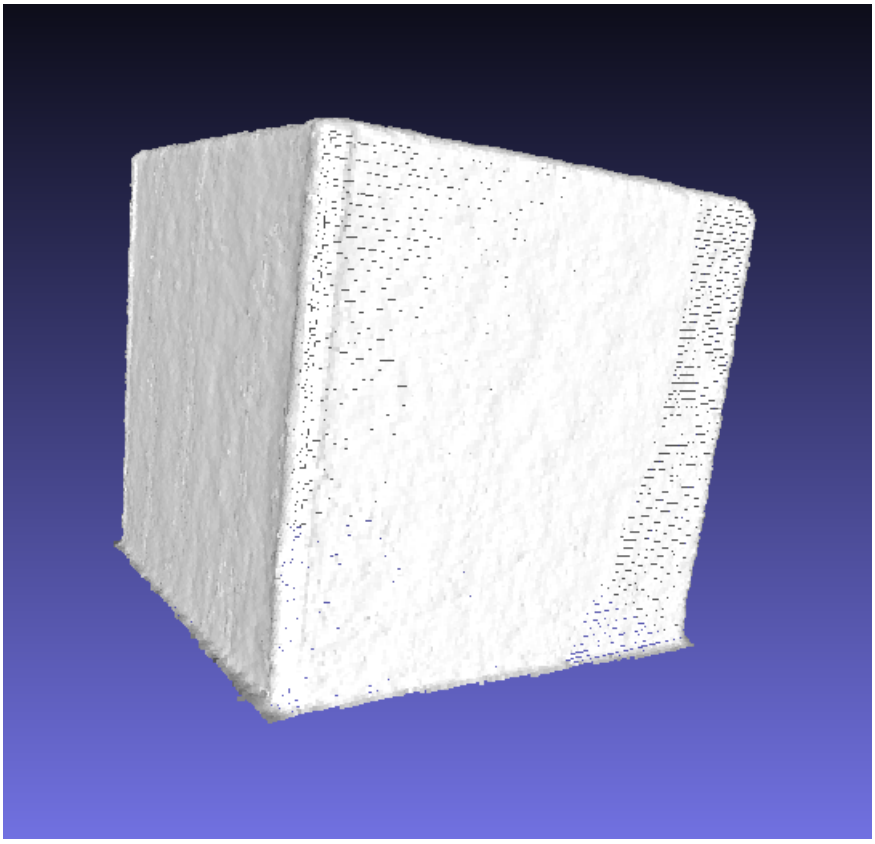


FIGURA 3.13. *Vista de la reconstrucción 3D del cubo desde Meshlab*

De la imagen 3D final se obtiene que el lado del cubo calculado fue de 261,1204$\pm$1,1810 [mm]. Esto indica que se debe analizar en detalle el error que comete el Kinect cuando se obtiene una nube de puntos.

CAPÍTULO 4

Evaluación metrológica del Kinect

Introducción

Una vez calibrado el sensor 3D, el dispositivo permite digitalizar en 3D puntos de la superficie del objeto, ubicados dentro del campo de observación calibrado. La pregunta normal que surge es cuál es la distancia más pequeña en el plano $(x - y)$ del sensor que puede muestrearse? y cuál es el error en profundidad que introduce el sistema de medida?. En este capítulo se describen los experimentos realizados para obtener la resolución en el plano transversal xy y el error en z. Estos parámetros, en especial el error en profundidad, son determinantes en el momento de valorar el uso del Kinect como dispositivo de medida 3D.

4.1. Evaluación transversal promedio mínima

En esta sección se desea calcular cuál es la distancia en el plano $(x - y)$ mínima y su distribución que se obtiene de la nube de puntos 3D que arroja el Kinect. Recordando que el eje z es el eje óptico de la cámara IR y el plano $(x - y)$ es el plano de su captor CCD, se pretende por lo tanto calcular cuál es la distancia más pequeña que puede muestrearse en $(x - y)$.

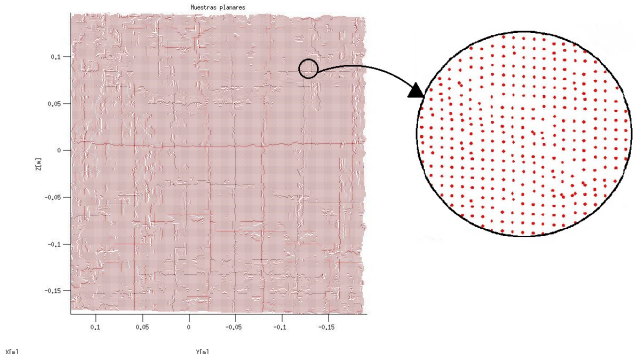


FIGURA 4.1. *Vista frontal de un plano reconstruido*

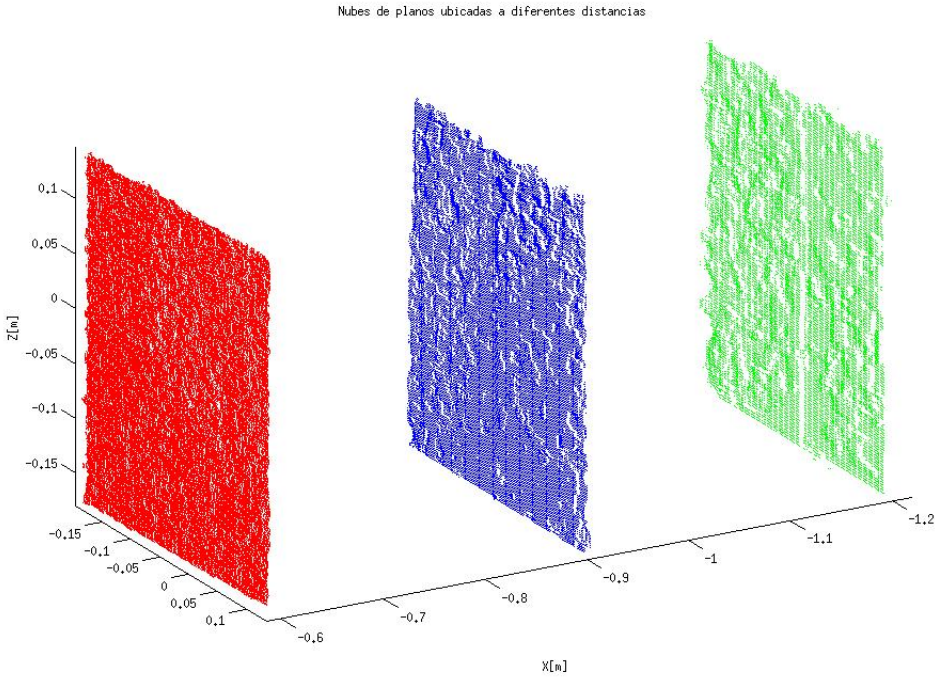
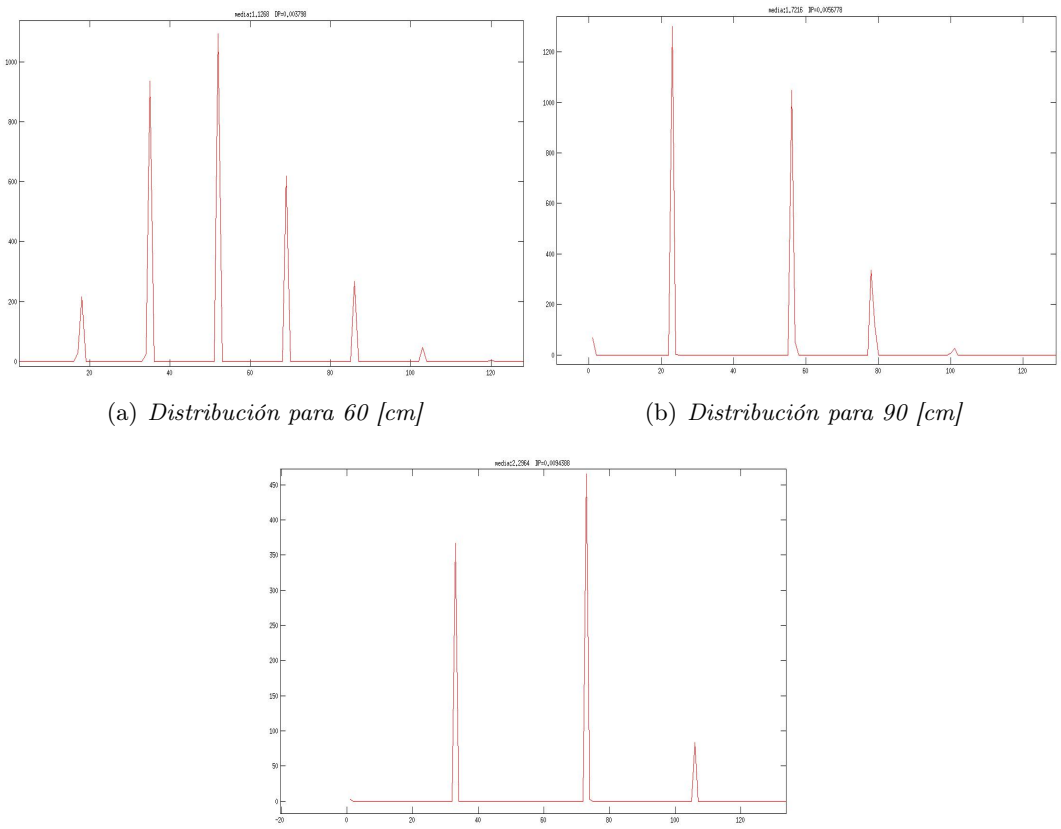


FIGURA 4.2. *Planos a diferentes distancias*

Con el fin de resolver este inconveniente se utilizó una superficie plana de referencia ubicada a diferentes distancias z. Para cada posición se determinaron la nube de puntos 3D y se emplearon 3 distancias z. La figura 4.2 muestra una nube de puntos del plano para las distancias 60 [cm], 90 [cm] y 120 [cm]. En la figura 4.1 se muestran los puntos muestreados y un zoom dónde se evidencia la no regularidad en el muestreo de la superficie.

Las figuras 4.3 muestran el histograma de distancias mínimas en el plano (x-y) para las nubes de puntos muestreados a los 3 valores de z. Es decir, cada imagen tenía alrededor de 700000 puntos 3D. Para cada punto se calculó su coordenada (x-y) y se calculó la distancia del vecino más cercano. De esta manera se obtuvo un vector con 700000 puntos de distancias más cercanas. El histograma de valores de distancia muestra que hay valores discretos de distancias mínimas. La tabla 4.1 muestra el valor medio de distancias mínimas, su desviación estándar y el error máximo, entendido como la distancia entre el más alejado y el valor medio.



(a) *Distribución para 60 [cm]*

(b) *Distribución para 90 [cm]*

(c) *Distribución para 120 [cm]*

FIGURA 4.3. *Distribuciones de probabilidades para las distancias 60 [cm], 90 [cm], 120 [cm]*

Propiedad \ Medida	60 $[cm]$	90 $[cm]$	120 $[cm]$
a	1,1269 $[mm]$	1,7224 $[mm]$	2,2979 $[mm]$
σ	$2{,}95 \times 10^{-4}$ $[mm]$	4×10^{-4} $[mm]$	6×10^{-4} $[mm]$
$Error_{max}$	$6{,}47 \times 10^{-4}$	1×10^{-3}	$2{,}7 \times 10^{-2}$

TABLA 4.1. Valores promedios encontrados para cada distancia

La causa de la discretización en (x-y) se origina de la distribución espacial de los granos del patrón de speckle. Según los resultados obtenidos, el patrón posee direcciones de distancias mínimas. Debido a la no-telecentricidad, estas distancias mínimas se incrementan con z. Observando el valor del error máximo se concluye que las distancias mínimas no cambian mucho en un z dado, se puede asumir constante. Estos valores indican que si la distancia mínima es a se puede muestrear un objeto con detalles en (x-y) de hasta $2a$ de separación en el plano.

4.2. Capacidad resolutiva de un Kinect en profundidad

Esta sección del trabajo estudia en su primera aproximación la resolución en profundidad del dispositivo Kinect, para esto se dispuso de tres diferentes distancias, 60 cm, 90 cm

y 120 *cm*. Se optó por tomar distancias relativamente pequeñas, sin desplazarnos al límite del rango de alcance de un Kinect, donde es posible encontrar errores en la medida muy grandes, los cuales no merecen ser calculados.

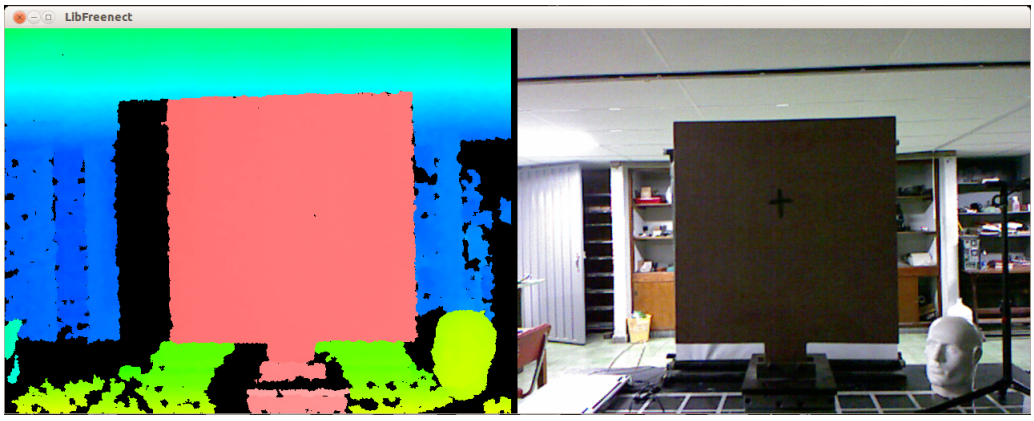


FIGURA 4.4. *Objetivo de captura*

Para esto se dispuso de una superficie plana de 30×30 [cm] como lo muestra la figura 4.4 ubicada frontalmente al dispositivo Kinect; Así se ubica perpendicularmente al eje óptico del sensor. Con el fin de reducir el error humano y del ambiente la toma de datos se da en completa ausencia de luz visible en el laboratorio de óptica, con el fin evitar que luz de otras longitudes de onda entren en el sensor y afecten la medida.

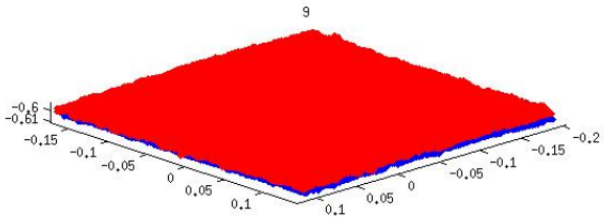


FIGURA 4.5. *Pareja de planos desplazados una distancia de 5 [mm]*

Para una distancia de aproximandamente 60 *cm* se adquieren 2 nubes de puntos 3D del plano con desplazamientos de 5 [mm], el proceso se repite 10 veces. El proceso anterior se repite para $z = 90$ [cm] y $z = 120$

La figura 4.5 muestra las nubes de puntos para una pareja de planos trasladados 5 [mm]. Con el fin de eliminar la influencia de la discretización en z, se realizó un proceso

Valor / Distancia	$Valor_{medio} \pm \sigma$ [mm]	E_{max}
60 $[cm]$	4.909 ± 0.1018	0.4869
90 $[cm]$	5.0878 ± 0.3694	0.99796
120 $[cm]$	6.621 ± 0.87949	2.312

TABLA 4.2. Tabla de valores medios y errores máximos para cada distancia

de regresión lineal para obtener el plano más óptimo por mínimos cuadrados. Luego se determinaron las distancias entre planos, para los puntos muestreados en (x-y); De esta manera se obtuvieron 70000 puntos por nube de puntos 3D que arroja 700000 puntos de distancias entre coordenadas z. Para cada posición en z estudiados se calculó el histograma de valores en dirección z. Con cada histograma normalizado se obtuvieron los valores medios, la desviación estándar y el error máximo, entendido como el valor de distancia en z máximo encontrado. La tabla 4.2 muestra los valores encontrados.

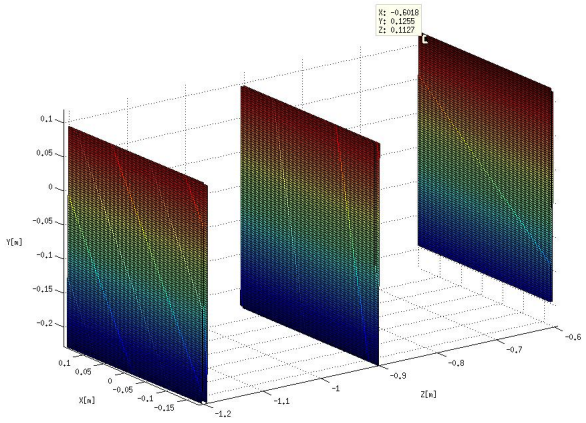


FIGURA 4.6. *Linealización de los planos para cada distancia*

De la tabla se observa que comparado con el valor esperado de 5 [mm] el error se incrementa en z, siendo muy fuerte para distancias superiores a 120 [cm], fuera de la zona de calibración. El proceso de linealización reduce la influencia de la discretización en z y del ruido, obteniendose unos errores en z de alrededor del 2 % .

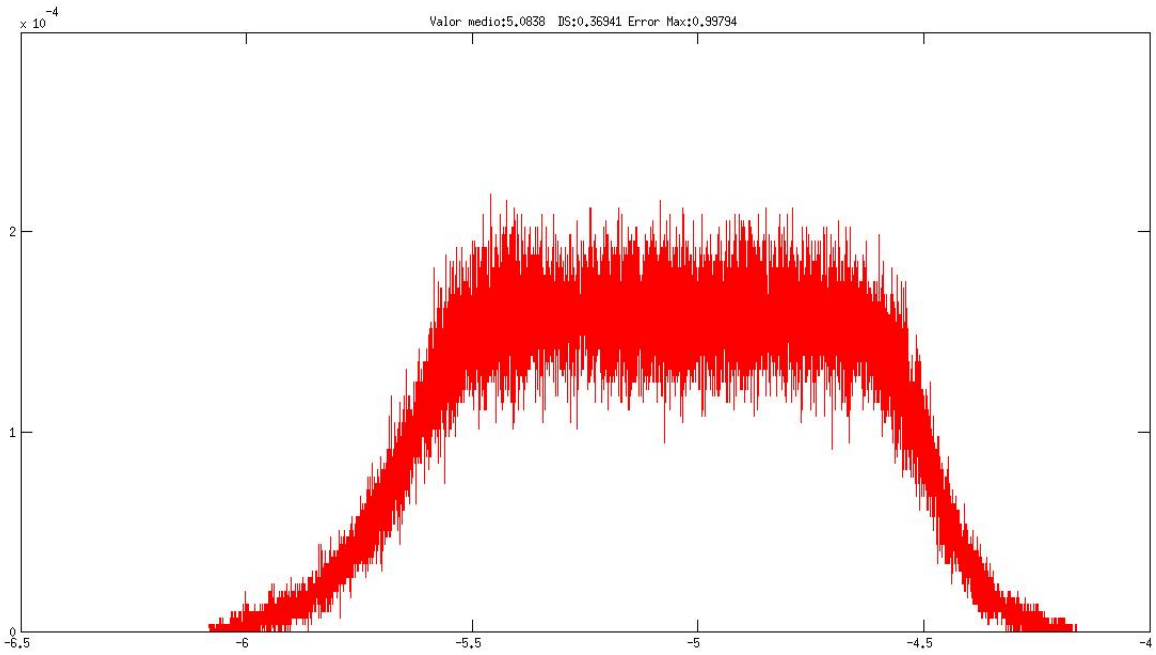


FIGURA 4.7. *Distribución de probabilidad de distancias calculadas entre planos*

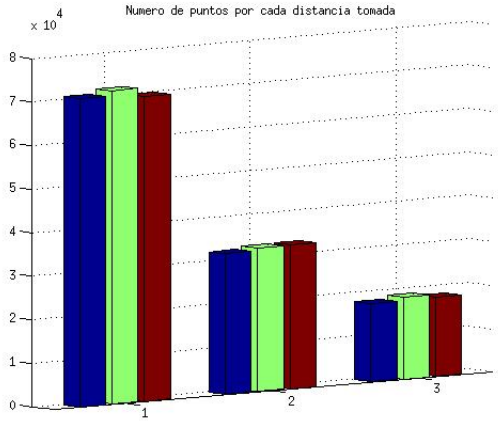


FIGURA 4.8. *Histograma: Número de puntos para cada distancia adquirida*

La capacidad resolutiva del dispositivo Kinect se reduce notoriamente a medida que aumenta la distancia al objetivo de captura, para una captura inicial a una distancia cercana al límite inferior de un plano de 30×30 *cm* se obtienen nubes de puntos, que oscilan los 70000 puntos, y éstas disminuyen drásticamente a medida que aumenta la distancia como lo muestra el digrama 4.8.

CAPÍTULO 5

Conclusiones

- Se investigó diferentes sensores de bajo costo con diferentes métodos para la obtención de datos 3D, de los cuales se usaron el dispositivo Kinect y las librerías de acceso libre para el desarollo del trabajo.
- Basados en las librerías de OpenCv que vienen enlazadas al software RGBDemo se pudo calibrar exitosamente un sensor Kinect, comprendiendo así sus propiedades internas de funcionamiento.
- Se adquiere información topográfica de diferentes objetos variando sus orientaciones. El software Meshlab fue adecuado para unificar las diferentes nubes de puntos, ya que la cantidad de datos adquiridos no sobrepasan la capacidad de procesamiento de datos del software.
- Se demostró la existencia de discretización de los datos en profundidad (DZ=0.010 mm) y transversales (1.12 mm) en función de la posición en Z para diferentes nubes de puntos. Esto implica que tanto en Z como en X,Y existen valores límites de tamaño mínimos que se pueden resolver, según el critero de Nyquist. Para el caso el muestreo en X,Y a Z=60 cm se pueden resolver detallas hasta de 2.24 mm, y en Z, detalles hasta 0.020 mm.
- Se estudió el error que comete el dispositivo para medir distancias en Z, encontrándose errores no superiores al 2 %, para desplazamientos de 5 mm, a Z=60 cm. Esto fue posible debido a que se usó una superficie plana y las nubes de datos 3D se linealizaron por mínimos cuadrados. Esto implica que existe una fuerte influencia del ruido en el proceso de determinación de la imagen de disparidad. Para objetos de forma 3D se puede hacer un proceso de aproximación polinomial local para la nube de puntos 3D que arroja el dispositivo, de tal manera que se reduzca la influencia del ruido. Esta idea ya fue implementada en algunos softwares de uso comercial para el kinect.
- La revisión bibliográfica hecha sobre el Kinect da cuenta de las capacidades de este dispositivo no solo en el campo de la metrología óptica sino también en el campo de la robótica, medicina, realidad virtual.
- Se comprobó que en un rango corto de trabajo el Kinect puede presentar grandes diferencias con respecto al valor real, disminuyendo así su capacidad resolutiva a medida que la distancia al objetivo aumenta.

- Se pudo comprobar como el exceso de luz en otras longitudes de onda en el rango IR pueden llegar a cegar el Kinect completamente, lo que lo hace poco útil en ambientes externos no controlados.

- De los resultados obtenidos al caracterizar el error que comete el dispositivo en Z y los valores de discretización en (X,Y) y Z se puede concluir que corresponden a valores que estan dentro de lo exigido en situaciones de metrología aplicada en ambientes médicos e indutriales, donde se exija alta cadencia de adquisición de datos 3D con bajas resoluciones. De igual forma, existe la posibilidad de mejorar las características metrolǵicas del dispositivo realizando ajustes a los elementos ópticos que conforman el dispositivo, como al procesamiento posterior de los datos 3D.

CAPÍTULO 6

Perspectivas

- Según los resultados obtenidos, se puede concluir que es posible mejorar las características metrológicas del dispositivo realizando ajustes en su óptica y realizando procesamiento de imágenes. De manera puntual, se puede reducir el campo de observación para incrementar la resolución transversal y axial. De igual forma, se puede realizar un procesamiento digital que reduzca el ruido localmente.

- Debido a la facilidad y alta velocidad para extraer datos de un objeto 3D, se plantea el uso no gratuito del software SKANECT, que posee procesamiento para reducir ruido. Al final de la realización del presente trabajo de caracterización, se utilizó un Kinect para reconstruir el Muñon por corte transtibial [20], con el objeto de construir la prótesis adecuada. Resultado del trabajo se pudo demostrar que reduciendo el ruido de salida y advirtiendo que la resolución requerida para construir la prótesis debía ser superior a 1 mm, se pudo utilizar la nube de datos del sofware para diseñar y construir las prótesis adecuadas. Como continuidad de este trabajo, se planteó un proyecto de investigación con financiación externa para realizar las prótesis de personas afectada por minas antipersonales; el proyecto aprobado y con presupeusto asignado es: MODELO DE INNOVACIÓN SOCIAL ORIENTADO A VÍCTIMAS DE MINAS ANTIPERSONAL (MAP) CON AMPUTACIÓN DE MIEMBRO INFERIOR, INTEGRANDO LA ESTRATEGIA REHABILITACIÓN BASADA EN COMUNIDAD (RBC) Y EL MODELO DE SERVICIO FUNDAMENTADO EN TECNOLOGÍAS VIRTUALES PARA EL DESARROLLO DE SOCKETS.

- En conclusión, este estudio permitió entender el funcionamiento del dispositivo Kinect y se pudo definir una serie variaciones, tanto ópticas como digitales que permitirán mejorar fuertemente sus propiedades metrológicas, que permitirán su fácil desempeño en ambientes médicos e industrales exigentes metrológicamente.

Bibliografía

[1] Stefano Berretti, Pietro Pala, and Alberto Del Bimbo. Increasing 3d resolution of kinect faces. In *Workshop at the European Conference on Computer Vision*, pages 639–653. Springer, 2014.

[2] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[3] Aymeric Bethencourt and Luc Jaulin. 3d reconstruction using interval methods on the kinect device coupled with an imu. *International Journal of Advanced Robotic Systems*, page xxx, 2012.

[4] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. .ºReilly Media, Inc.", 2008.

[5] Hein AM Daanen and Frank B Ter Haar. 3d whole body scanners revisited. *Displays*, 34(4):270–275, 2013.

[6] C BROWN Duane. Close-range camera calibration. *Photogrammetric engineering*, 37(8):855–866, 1971.

[7] Barak Freedman, Alexander Shpunt, Meir Machline, and Yoel Arieli. Depth mapping using projected patterns, April 3 2012. US Patent 8,150,142.

[8] Luigi Gallo, Alessio Pierluigi Placitelli, and Mario Ciampi. Controller-free exploration of medical image data: Experiencing the kinect. In *Computer-based medical systems (CBMS), 2011 24th international symposium on*, pages 1–6. IEEE, 2011.

[9] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663, 2012.

[10] C Herrera, Juho Kannala, Janne Heikkilä, et al. Joint depth and color camera calibration with distortion correction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(10):2058–2064, 2012.

[11] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[12] Abhijit Jana. *Kinect for Windows SDK Programming Guide*. Packt Publishing Ltd, 2012.

[13] Achuta Kadambi, Vage Taamazyan, Boxin Shi, and Ramesh Raskar. Polarized 3d: High-quality depth sensing with polarization cues. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3370–3378, 2015.

[14] Rizwan Macknojia, Alberto Chávez-Aragón, Pierre Payeur, and Robert Laganiere. Calibration of a network of kinect sensors for robotic inspection over a large workspace. In *Robot Vision (WORV), 2013 IEEE Workshop on*, pages 184–190. IEEE, 2013.

[15] Ekaterina Manuylova. Investigations of stereo setup for kinect. 2012.

[16] Microsoft. Software development kit by microsoft. https://dev.windows.com/en-us/kinect. Accessed: 11-04-2016.

[17] Masataka Nakazawa, Ikuhisa Mitsugami, Yasushi Makihara, Hiromasa Nakajima, Hitoshi Habe, Hirotake Yamazoe, and Yasushi Yagi. Dynamic scene reconstruction using asynchronous multiple kinects. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 469–472. IEEE, 2012.

[18] Mitsuru Nakazawa, Ikuhisa Mitsugami, Yasushi Makihara, Hozuma Nakajima, Hitoshi Habe, Hirotake Yamazoe, and Yasushi Yagi. Dynamic scene reconstruction using asynchronous multiple kinects. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 469–472. IEEE, 2012.

[19] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[20] Astrid Rocio Moreno Pardo. Desarrollo de un socket para amputacion transtibial adaptable a los cambios de volumen del muñon.

[21] Carlos Ricardo Contreras Pico. Análisis tridimensional de objetos a 360 grados de observación y campo amplio con múltiple configuración proyector-cámara a partir de iluminación estructurada. *Recurso electrónico UIS*, page xxx, 2014.

[22] Michele Pirovano. Kinfu–an open source implementation of kinect fusion+ case study: implementing a 3d scanner with pcl. *Project Assignment, 3D structure from visual motion, University of Milan*, 2011.

[23] ROS. Calibración ros.org.

[24] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152. IEEE, 2001.

[25] Jan Smisek, Michal Jancosek, and Tomas Pajdla. 3d with kinect. In *Consumer Depth Cameras for Computer Vision*, pages 3–25. Springer, 2013.

[26] JOAQUIN VIÑALS PONS. *Localización y generación de mapas del entorno (SLAM) de un robot por medio de una Kinect*. PhD thesis, 2012.

[27] Wikipedia. Homogeneous coordinates — wikipedia, the free encyclopedia, 2016. [Online; accessed 1-April-2016].

[28] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1625–1632, 2013.

[29] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.