RECONSTRUCCIÓN 3D BASADA EN MUESTREO COMPRIMIDO

T.G. 1419

CAMILA CORREA BALCÁZAR

Informe Final

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA ELECTRÓNICA
2014
RECONSTRUCCIÓN 3D BASADA EN MUESTREO COMPRIMIDO

T.G. 1419

CAMILA CORREA BALCÁZAR

Director
Ing. LEONARDO FLOREZ, Ph.D.
Profesor

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA ELECTRÓNICA
2014
PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA

RECTOR MAGÍFICO: RP. JOAQUÍN SÁNCHEZ GARCÍA. S.J.
DECANO ACADEMICO: ING. FRANCISCO JAVIER REBOLLEDO MUÑOZ.
DECANO MEDIO UNIVERSITARIO: RP.SERGIO BERNAL RESTREPO. S.J.
DIRECTOR DE CARRERA DE INGENIERÍA ELECTRÓNICA: ING. JAIRO HURTADO. Ph.D.
DIRECTOR DE PROYECTO: ING. LEONARDO FLOREZ. Ph.D.

ARTÍCULO 23 DE LA RESOLUCIÓN No. 13 DE JUNIO DE 1946

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque los trabajos no contengan ataques o polémicas puramente personales. Antes bien, que se vea en ellos el anhelo de buscar la verdad y la justicia".

Dedico este trabajo de grado a:

A Dios por darme fortaleza cuando la esperanza se perdía.
A mi madre por su amor y fe hacia mí.
A Daniel Mejía por su apoyo y mantenerme cuerda cuando lo necesité.
A mi director Leonardo Flórez por creer en mí y darme la esperanza que me hacía falta para lograrlo.

AGRADECIMIENTOS

Quiero agradecer especialmente a mi director, Leonardo Flórez, por acompañarme en este proceso y retomar conmigo está aventura, y por motivarme cuando lo necesité.

Darle las gracias a mi madre, Cecilia Balcázar, por sus años de esfuerzo para que pudiera sacar mi carrera adelante, soportar mis trasnochos y acompañarme con su presencia. Y a Daniel, sin su constante ánimo y palabras de fortaleza me hubiera perdido, gracias.

TABLA DE CONTENIDO

LISTA DE TABLAS ---- 8
INTRODUCCIÓN ---- 9
1. MARCO TEÓRICO ---- 10
1.1. CALIBRACIÓN ---- 10
1.2. HOMOGRAFÍA ---- 13
1.3. DESCRIPTORES ---- 14
2. ESPECIFICACIONES ---- 17
2.1 HARDWARE DEL SISTEMA ---- 17
2.1.1. Interfaz Motor-Driver ---- 18
2.1.2. Interfaz driver-Arduino ---- 21
2.1.3. Interfaz Arduino- computador ---- 22
2.2 SOFTWARE DEL SISTEMA ---- 22
2.2.1 Openni ---- 23
2.2.2 PCL ---- 23
2.2.3 Diagrama de subsistemas del aplicativo ---- 24
3. DESARROLLOS ---- 25
3.1. DESCRIPCIÓN DE SUBSISTEMAS ---- 25
3.1.1. Adquisición de Datos ---- 25
3.1.2. Estimación de Keypoints ---- 25
3.1.3. Estimación de Descriptores ---- 26
3.1.4. Estimación de Correspondencias ---- 26
3.1.5. Transformación ---- 27
3.1.6. Control Plataforma ---- 27
3.2. PROCESO DE DESARROLLO ---- 27
4. ANÁLISIS DE RESULTADOS ---- 28
4.1. RESULTADOS PRUEBAS ITERACIÓN 1 ---- 30
4.2. RESULTADO PRUEBAS ITERACIÓN 2 ---- 32
4.3. RESULTADO PRUEBAS ITERACIÓN 3 ---- 35
4.4. RESULTADO PRUEBAS ITERACIÓN 4 ---- 37
4.5. RESULTADO PRUEBAS ITERACIÓN 5 ---- 40
4.6. RESULTADO PRUEBAS ITERACIÓN 6 ---- 42
4.7. RESULTADOS PRUEBAS MÚLTIPLES IMÁGENES ---- 45
5. CONCLUSIONES ---- 53
6. BIBLIOGRAFÍA ---- 54
TABLA DE ANEXOS ---- 55

LISTA DE FIGURAS

Ilustración 1. Distribución de las cámaras en Kinecthttp://wccftech.com/microsoft-kinect-identify-africanamericans....10
Ilustración 2. Campo de visión de las cámaras RGB y de profundidad....11
Ilustración 3. Patrón tablero de ajedrez....11
Ilustración 4. Distancia entre emisor y sensor infrarrojo....12
Ilustración 5. Región de influencia, k-vecindario [20]....16
Ilustración 6. Sistema de coordenadas fijo....16
Ilustración 7. Hardware del sistema....17
Ilustración 8. Detalle Hardware del sistema....18
Ilustración 9. Funciones del driver....19
Ilustración 10. Arduino UNO....22
Ilustración 11. Vista por capas del sistema....23
Ilustración 12. Diagrama de subsistemas del aplicativo....24
Ilustración 13. Imagen de referencia....29
Ilustración 14. Resultado Iteración 1 vista frontal....31
Ilustración 15. Resultado Iteración 1 Vista superior....31
Ilustración 16. Resultado Iteración 1 Vista Lateral....32
Ilustración 17. Resultado Iteración 1 Vista Frontal....33
Ilustración 18. Resultado Iteración 2 Vista Superior....34
Ilustración 19. Resultado Iteración 2 Vista Lateral....34
Ilustración 20. Resultado Iteración 3 Vista Frontal....36
Ilustración 21. Resultado Iteración 3 Vista Superior....36
Ilustración 22. Resultado Iteración 3 Vista Lateral....37
Ilustración 23. Resultado Iteración 4 Vista Frontal....38
Ilustración 24. Resultado Iteración 4 Vista lateral....39
Ilustración 25. Resultado Iteración 4 Vista Superior....39
Ilustración 26. Resultado Iteración 5 Vista Frontal....41
Ilustración 27. Resultado Iteración 5 Vista Superior....41
Ilustración 28. Resultado Iteración 5 Vista Lateral....42
Ilustración 29. Resultado Iteración 6 Vista Frontal....43
Ilustración 30. Resultado Iteración 6 Vista Superior....44
Ilustración 31. Resultado Iteración 6 Vista Superior Lateral....44
Ilustración 32. Resultado Prueba Múltiple Vista Frontal - 3 imágenes....46
Ilustración 33. Resultado Prueba Múltiple Vista Superior - 3 Imágenes....47
Ilustración 34. Resultado Prueba Múltiple Vista Lateral – 3 Imágenes....47
Ilustración 35. Prueba Múltiple Vista Superior – 10 imágenes....50
Ilustración 36. Prueba Múltiple Vista Frontal – 10 imágenes....50
Ilustración 37. Prueba Múltiple Vista lateral 1 – 10 imágenes....51
Ilustración 38. Prueba Múltiple Vista lateral 2 – 10 imágenes....51
Ilustración 39. Imagen referencial biblioteca....52
Ilustración 40. Imagen sobre puesta de la nube de puntos y la biblioteca....52

LISTA DE TABLAS

Tabla 1. Micropasos 19
Tabla 2. Entradas y salidas del driver 20
Tabla 3. Asignación de pines en el Driver 21
Tabla 4. Correspondencia de pines entre el driver y el arduino 22
Tabla 5. Matriz de rotación sobre eje Z para 5° 29
Tabla 6. Resultado Pruebas Iteración 1 30
Tabla 7. Matriz de transformación Iteración 1 30
Tabla 8. Resultado Prueba Iteración 2 32
Tabla 9. Matriz de Transformación Iteración 2 33
Tabla 10. Resultados Iteración 3 35
Tabla 11. Matriz Transformación Iteración 3 35
Tabla 12. Resultado Iteración 4 37
Tabla 13. Matriz de Transformación Iteración 4 38
Tabla 14. Resultado Iteración 5 40
Tabla 15. Matriz de transformación Iteración 5 40
Tabla 16. Resultado Prueba Iteración 6 42
Tabla 17. Matriz de transformación Iteración 6 43
Tabla 18. Resultado Prueba Múltiple 1.1 45
Tabla 19. Matriz de transformación Múltiple 1.1 45
Tabla 20. Resultado Prueba Múltiple 1.2 45
Tabla 21. Matriz transformación Múltiple 1.2 46
Tabla 22. Resultado Prueba Múltiple 1 – 10 imágenes 48
Tabla 23. Resultado Prueba Múltiple 2 – 10 imágenes 48
Tabla 24. Resultado Prueba Múltiple 3 – 10 imágenes 48
Tabla 25. Resultado Prueba Múltiple 4 – 10 imágenes 48
Tabla 26. Resultado Prueba Múltiple 5 – 10 imágenes 49
Tabla 27. Resultado Prueba Múltiple 6 – 10 imágenes 49
Tabla 28. Resultado Prueba Múltiple 7 – 10 imágenes 49
Tabla 29. Resultado Prueba Múltiple 8 – 10 imágenes 49
Tabla 30. Resultado Prueba Múltiple 9 – 10 imágenes 49

INTRODUCCIÓN

La unión de la ingeniería electrónica y la ingeniería de sistemas ha permitido el avance de la tecnología en diversos ámbitos, específicamente, en el contexto de este proyecto, en el desarrollo de la robótica. Es por esto en conjunto con los grupos de investigación TAKINA, de ingeniería de sistemas, y SIRP, de ingeniería electrónica, nace el proyecto en caminado a la percepción en robótica.

Con el desarrollo y comercialización de cámaras RGB-D a bajo costo el campo de percepción en robótica ha estado en creciente actividad. Como parte de ese crecimiento se plantea la necesidad de reconstrucción en tres dimensiones de espacios interiores, que a futuro pueda servir como punto de partida para proyectos relacionados con SLAM, Simultaneous Localization And Mapping.

Este trabajo se apoya en herramientas recientes en desarrollo de manipulación y procesamiento de nubes de puntos que ha abierto una nueva puerta a las posibilidades, ligado a la facilidad de las nuevas tecnologías en cámaras de profundidad, permite un acercamiento natural a la percepción y por tanto a la navegación y localización en robótica.

Como objetivo principal es formular una estrategia que permita una buena aproximación a la reconstrucción 3D de espacios interiores, que permita unir hardware y software, haciendo uso de algoritmos ya existentes, haciendo uso de elementos comerciales como, Kinect, sobre una plataforma de transformaciones rígidas controladas, que permitirá controlar el ángulo de rotación. Este proyecto se desarrolló para que sirva como base a futuros proyectos en caminados a la tarea de localización y mapeo simultáneo.

El desarrollo permite a partir de un sistema de bajo costo, que sea asequible comercialmente, plantear una estrategia para la reconstrucción que permita unir los enfoques de ingeniería de sistemas e ingeniería electrónica bajo un objetivo.

Este documento se divide en 5 capítulos:

- 1 MARCO TEÓRICO: da una visión general de los conceptos básicos empleados para que se contextualice con el desarrollo del proyecto.
- 2 ESPECIFICACIONES: en las especificaciones se incluyen el diagrama general de bloques, la interfaz del motor, herramientas usadas.
- 3 DESARROLLO: Explica el desarrollo donde se encuentran los recursos usados y el proceso planteado para la codificación y pruebas.
- 4 ANÁLISIS DE RESULTADOS: En este capítulo se realiza el análisis de resultados de los subsistemas y del programa completo para obtener las transformaciones.
- 5 CONCLUSIONES: Se encuentran las conclusiones y posibles mejoras a futuros para otros alcances.

1. MARCO TEÓRICO

1.1. CALIBRACIÓN

Para trabajar con la cámara Kinect lo primero que se debe hacer es realizar una calibración para obtener buenos resultados. La calibración busca solucionar dos problemas, que se presentan a continuación, por la geometría de las cámaras.

El Kinect posee dos cámaras, una de profundidad a través de un infrarrojo y una RGB. Al ser imposible ubicar las cámaras exactamente sobre el mismo punto hay una distancia entre las cámaras que hace que la correspondencia entre imágenes no sea directa y se produzca un error que aumenta con la distancia.

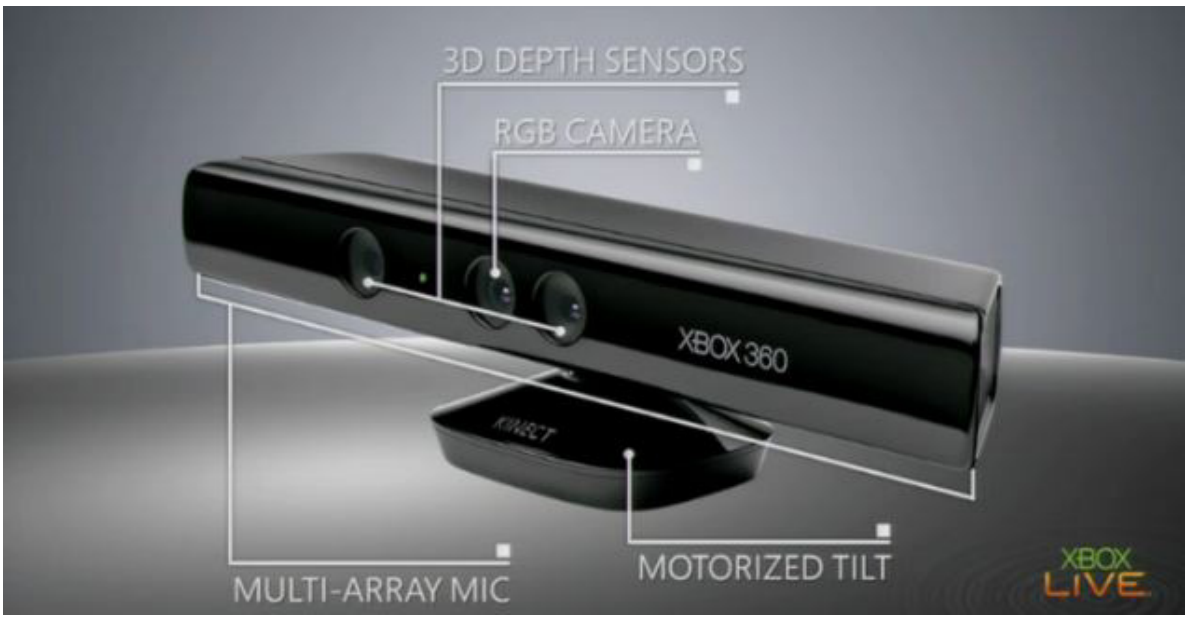


Ilustración 1. Distribución de las cámaras en Kinecthttp://wccftech.com/microsoft-kinect-identify-africanamericans

El primer problema que es necesario enfrentar es producido por la distancia entre el receptor de la cámara de profundidad y la cámara RGB. Esto produce dos campos de visión diferentes, ilustración 2, donde existen tres regiones: una solo visible para la cámara RGB, la común a las dos cámaras, y una solo visible para la cámara de profundidad.

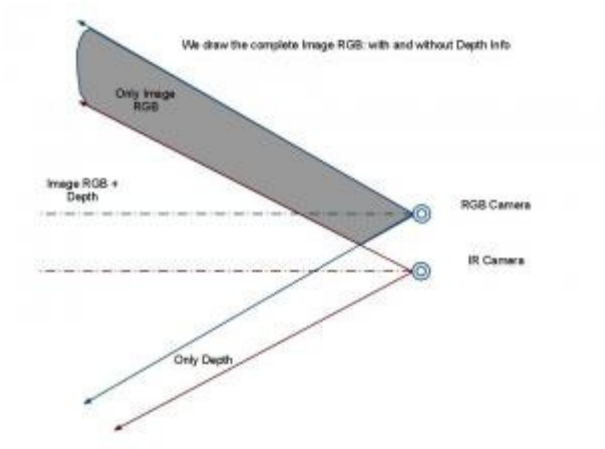


Ilustración 2. Campo de visión de las cámaras RGB y de profundidad

La distancia entre las cámaras RGB y la cámara de profundidad recuerda un problema de visión estéreo. Esta línea base es pequeña. Utilizando un patrón conocido como un tablero de ajedrez como el observado en la ilustración 4, se extrae en la cámara de profundidad las esquinas del tablero, que también se extraen de la cámara RGB. Luego se proyecta cada píxel de profundidad a un espacio 3D usando las ecuaciones 1, 2,3.

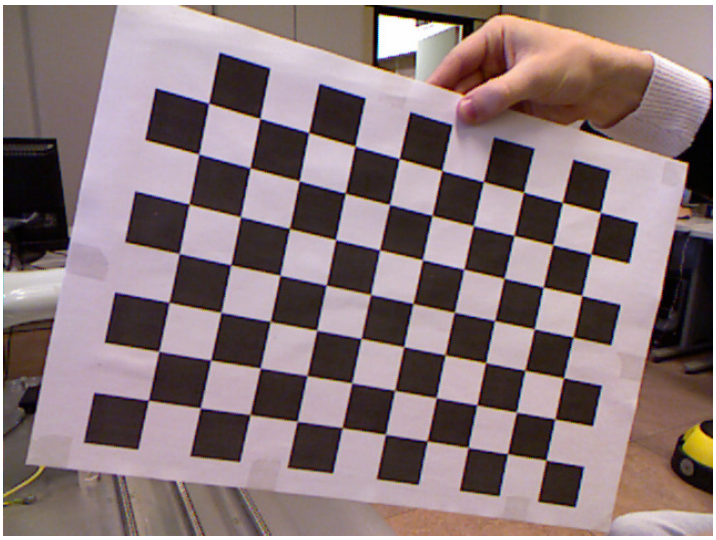


Ilustración 3. Patrón tablero de ajedrez

$$P3D.x = (x_d - cx_d) * \frac{depth(x_d,y_d)}{fx_d} \qquad \text{Ecuación 1}$$

$$P3D.y = (y_d - cy_d) * \frac{depth(x_d,y_d)}{fy_d} \qquad \text{Ecuación 2}$$

$$P3D.z = depth\,(x_d,y_d) \qquad \text{Ecuación 3}$$

Donde fx_d, fy_d, son intrínsecos de la cámara de profundidad. Ahora se puede reproyectar cada punto en 3D a su imagen de color y por tanto obtener el color por píxel.

$$P3D' = R.P3D + T \qquad \text{Ecuación 4}$$

$$P2D_{rgb}.x = \left(P3D.x * \frac{fx_{rgb}}{P3D'.z}\right) + cx_{rgb} \qquad \text{Ecuación 5}$$

$$P2D_{rgb}.y = \left(P3D.y * \frac{fy_{rgb}}{P3D'.z}\right) + cy_{rgb} \qquad \text{Ecuación 6}$$

El segundo error a solucionar solo involucra la cámara de profundidad, se tiene un emisor de infrarrojo y su respectivo sensor. El emisor proyecta una malla infrarroja y se encuentra ubicado al extremo izquierdo del Kinect, visto de frente, mientras que el sensor infrarrojo está ligeramente a la derecha, desde la misma perspectiva, con una distancia aproximada de 7.5cm.

El emisor golpea con un ángulo los objetos, el reflejo es captado por el sensor, lo que causa que el objeto en un ángulo, como se observa en la ilustración 3, sea invisible para la imagen de profundidad. Adicionalmente se creará una sombra detrás de cada objeto.

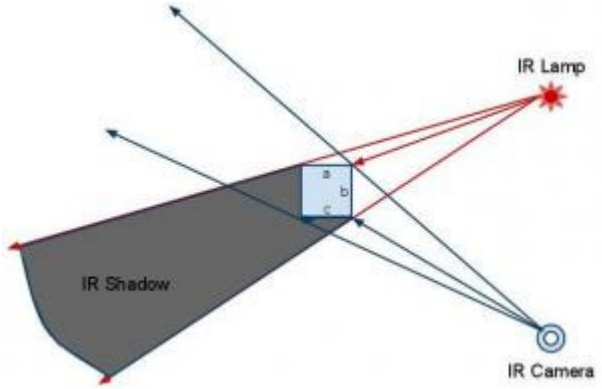


Ilustración 4. Distancia entre emisor y sensor infrarrojo

El objetivo de realizar la calibración es disminuir estos errores. Es importante anotar que las características físicas de cada Kinect son diferentes y que las medidas que se pueden obtener son solo aproximaciones, por lo cual los errores solo pueden disminuirse más no evitarse.

La profundidad se calcula por triangulación una vez proyectada la malla de infrarrojo, esto se hace con ayuda de un patrón conocido. Se utiliza una ventana de correlación de 9x9 o 9x7 para por cada píxel de la imagen comparar el patrón local de ese píxel con el que ya se memorizó y además con 64 píxeles cercanos tomados en ventana horizontal. Así se puede obtener la disparidad, y con el patrón conocido se puede calcular la profundidad por triangulación. La relación existente entre profundidad y disparidad viene dada por la Ecuación 7, donde z en metros es la profundidad, b es la línea base horizontal entre las cámaras, f la longitud focal de las cámaras en píxeles, y d la disparidad en píxeles.

$$z = b * f/d \qquad \text{Ecuación 7}$$

Cuando la disparidad es cero los rayos de las cámaras son paralelos, es decir, la profundidad es infinita. Si la disparidad es un valor alto, entonces se tiene una distancia más corta. Pero esto no es cierto en el Kinect, ya que la disparidad cero no corresponde a una distancia infinita.

Por lo anterior se debe pensar en una disparidad del Kinect, que corresponde a la disparidad normalizada como se encuentra en la Ecuación 8, donde *d* corresponde a la disparidad normalizada, *kd* a la disparidad del Kinect expresada en unidades de 1/8 de píxel, y *doff* es el valor de desplazamiento particular del Kinect.

$$d = 1/8 * (doff - kd)$$ Ecuación 8

Reemplazando la Ecuación 2 en la Ecuación 1, tenemos como resultado la Ecuación 9.

$$z = b * f / \left(\frac{1}{8} * (doff - kd)\right)$$ Ecuación 9

En la Ecuación 9, *b* corresponde a la distancia entre el emisor y el receptor de infrarrojo, como ya se mencionó, que es aproximadamente 7.5 cm, y el *doff* se asume normalmente en 1090. Con esto se soluciona el segundo problema planteado.

1.2. HOMOGRAFÍA

La relación de dos imágenes en cuanto a líneas y puntos en una escena plana se llama homografía. La relación solo se da cuando, como ya se mencionó, la escena es plana o si el desplazamiento que presenta es muy pequeño. Es una transformación proyectiva de formas geométricas planas. Existen varios tipos de transformaciones homográficas como la traslación, simetría y homología.

Las transformaciones proyectivas se definen como transformaciones invertibles de P^2 en P^2. Una homografía se puede representar en coordenadas homogéneas como una transformación lineal e invertible. Entre dos planos una homografía se da entre vectores tridimensionales homogéneos a través de una matriz de 3x3, $\vec{x'} = H\,\vec{x}$.

La isometría es una transformación que preserva la distancia euclidiana, está dada según la ecuación 10, donde $\varepsilon = \pm 1$, si es positiva se mantiene la orientación, y al ser negativo se invierte, R es la matriz de rotación de 2x2, *t* es el vector de translación.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \varepsilon\cos\theta & -\sin\theta & t_x \\ \varepsilon\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$ Ecuación 10

$$\bar{\mathrm{x}}' = \mathrm{H_E}\bar{\mathrm{x}} = \begin{bmatrix} \mathrm{R} & \mathrm{t} \\ 0^{\mathrm{T}} & 1 \end{bmatrix} \bar{\mathrm{x}}$$ Ecuación 11

Una similaridad es una isometría que tiene en cuenta una escala isotrópica, en la ecuación 12 se muestra la representación de la similaridad, donde *s* es el factor de escalamiento.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Ecuación 12

$$\tilde{\mathbf{x}}' = \mathbf{H_S}\tilde{\mathbf{x}} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^{\mathbf{T}} & 1 \end{bmatrix} \tilde{\mathbf{x}}$$

Ecuación 13

Afinidad es otra transformación, está definida como una transformación lineal no singular. Es una homografía con un centro impropio, es decir que el punto de concurrencia de los rayos de homología es un punto en el infinito. Como se muestra en la ecuación 14.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Ecuación 14

$$\tilde{\mathbf{x}}' = \mathbf{H_A}\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^{\mathbf{T}} & 1 \end{bmatrix} \tilde{\mathbf{x}}$$

Ecuación 15

La proyectividad generaliza las transformaciones proyectivas, puede expresarse como se muestra en la ecuación 16, se puede descomponer para obtener la ecuación 17, donde *A* es la matriz no singular $A = sRK + tv^T$, y *K* es una matriz triangular superior normalizada como *det(K) =1*.

$$\tilde{\mathbf{x}}' = \mathbf{H_P}\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^{\mathbf{T}} & v \end{bmatrix} \tilde{\mathbf{x}}$$
$$\mathbf{v} = [v_1, v_2]^T$$

Ecuación 16

$$\mathbf{H} = \mathbf{H_S}\mathbf{H_A}\mathbf{H_P} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^{\mathbf{T}} & 1 \end{bmatrix} \begin{bmatrix} s\mathbf{K} & \mathbf{t} \\ \mathbf{0}^{\mathbf{T}} & 1 \end{bmatrix} \begin{bmatrix} s\mathbf{I} & \mathbf{t} \\ \mathbf{v}^{\mathbf{T}} & v \end{bmatrix} = \begin{bmatrix} s\mathbf{A} & \mathbf{t} \\ \mathbf{v}^{\mathbf{T}} & v \end{bmatrix}$$

Ecuación 17

1.3. DESCRIPTORES

La representación de puntos en el espacio 3D viene dada por las coordenadas (x,y,z), con respecto a un origen determinado. Cuando se realiza adquisición de imágenes en dos tiempos diferentes, se encuentra que dos puntos que tengan exactamente las mismas coordenadas, pero la cámara entre los tiempos va a cambiar sin modificar el origen. Por lo tanto estos puntos en el mundo real no son el mismo, por más que las coordenadas en con respecto a nuestro origen sean las mismas.

Es por esto que se necesita hacer cada punto único, por lo que hay que proporcionar información adicional sobre cada punto. Es aquí donde los descriptores se hacen necesarios para que la comparación entre imágenes no sea ambigua.

Necesitamos no solo distinguir el punto sino distinguir las superficies geométricas. Se hace necesario por tanto definir o describir un punto no solo por el mismo sino también en cuanto a sus vecinos colindantes. Para que una representación pueda ser considerada como buena debe ser capaz de soportar transformaciones rígidas, variación en la densidad de muestreo, y niveles de ruido leve.

La propiedad de superficie geométrica que se utiliza para los descriptores es la normal de la superficie. Esta normal representa un vector perpendicular a la superficie en un punto. Para el caso de la nube de puntos se debe calcular las normales en cada punto de la nube.

El método de estimación de la normal se aproxima por medio de la estimación de la normal de un plano tangente a la superficie, lo que se convierte en un problema de estimación de mínimos cuadrados. La solución se reduce al análisis de los valores y los vectores propios, también conocido como PCA, Análisis de los componentes principales, de la matriz de covarianza que ha sido creada a partir de los vecinos más cercanos. Esta matriz de covarianza C está expresada en la ecuación 18, donde *k* es el número de vecinos que se tienen en cuenta para p_i, $\bar{p}$ representa el centroide 3D de los vecinos cercanos, λ_j es el valor propio *j* de la matriz de covarianza, y por último, el vector del valor propio es $\overrightarrow{v_j}$.

$$C = \frac{1}{k}\sum_{i=1}^{k}((p_i - \bar{p}) \cdot (p_i - \bar{p})^T) \qquad \text{Ecuación 18}$$

$$C \cdot \overrightarrow{v_j} = \lambda_j \cdot \overrightarrow{v_j} \qquad \text{Ecuación 19}$$

$$j \in \{0,1,2\} \qquad \text{Ecuación 20}$$

Matemáticamente no existe una forma de resolver la dirección de la normal, por lo que este signo es ambiguo al calcularse la normal. Para solucionar esto solo se necesita conocer el punto de vista, que se asume en la posición 0,0,0, y se debe satisfacer la ecuación 21 para la orientación de las normales. Se tiene que v_p corresponde al punto de vista, y $\overrightarrow{n_i}$ a las normales.

$$\overrightarrow{n_i} \cdot (v_p - p_i) > 0 \qquad \text{Ecuación 21}$$

El problema central que queda es determinar cuáles son los vecinos cercanos al punto que determinan su vecindario, valor que se necesita si se desea realizar de forma automática la estimación de cada punto. La escogencia de la vecindad depende del nivel de detalles que se desee tener, si se desea tener detalles muy finos el vecindario debe ser pequeño, mientras que si se interesa por detalles gruesos o no finos el vecindario puede ser grande.

Los descriptores *Point Feature Histograms* (PFH) tienen como objetivo codificar un punto a través de las propiedades geométricas de la k-vecindad, lo que se hace generalizando la curvatura media alrededor de un punto usando histogramas multidimensionales de valores. Esto responde bien al ruido del vecindario así como a las diferentes densidades de muestreo. La idea principal es captar las variaciones de la superficie teniendo en cuenta las interacciones entre las normales estimadas. En la ilustración 5 se observa una región de influencia, k-vecindario, para un punto específico, p_k, los puntos en el vecindario están completamente interconectados en una red. [20]

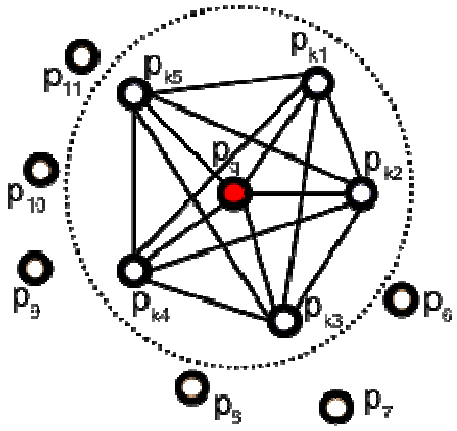


Ilustración 5. Región de influencia, k-vecindario [20]

Para calcular la diferencia relativa entre dos puntos, p_i y p_j, y sus normales asociadas, n_i y n_j, se debe definir un sistema de coordenadas fijo en uno de los puntos, como se observa en la ilustración 6.

$$\mathbf{u} = \boldsymbol{n}_s$$

$$\mathbf{v} = \mathbf{u} \times \frac{(\boldsymbol{p}_t - \boldsymbol{p}_s)}{\|\boldsymbol{p}_t - \boldsymbol{p}_s\|_2}$$

$$\mathbf{w} = \mathbf{u} \times \mathbf{v}$$

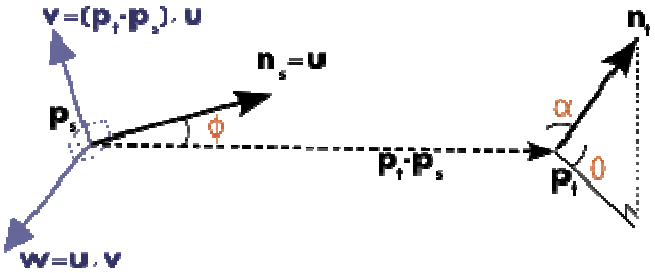


Ilustración 6. Sistema de coordenadas fijo

La diferencia entre las normales n_i y n_j, teniendo en cuenta el sistema de coordenadas elegido, puede ser expresado de forma angular como:

$$\alpha = \mathbf{v} \cdot \boldsymbol{n}_t$$

$$\dot{\phi} = \mathbf{u} \cdot \frac{(\boldsymbol{p}_t - \boldsymbol{p}_s)}{d}$$

$$\theta = \arctan(\mathbf{w} \cdot \boldsymbol{n}_t, \mathbf{u} \cdot \boldsymbol{n}_t)$$

Donde d corresponde a la distancia euclidiana entre los puntos, $d = \|\boldsymbol{p}_t - \boldsymbol{p}_s\|_2$, la cuádrupla $\langle \alpha, \dot{\phi}, \theta, d \rangle$ se calcula para cada par de puntos en el k-vecindario.

2. ESPECIFICACIONES

El sistema planteado tiene varios componentes que deben interactuar y relacionarse. Un primer componente a nivel de software, donde está el grueso del proyecto en cuanto a la reconstrucción, Y un segundo componente de hardware que permite la toma de imágenes que son la entrada para el software.

Para la parte de software se escogió para el trabajo la librería PCL, Point Cloud Library, que permite el procesamiento y manipulación de nubes de puntos, es una librería nueva y en desarrollo que fue escogida por las herramientas que ofrece y por encontrarse entre lo más reciente en el estado del arte.

En el hardware, la primera escogencia que se hizo fue el Kinect, aún antes de la presentación de la propuesta para el desarrollo de este proyecto. El Kinect permite la toma de imágenes de profundidad así como RGB, tiene un costo bajo por su característica comercial, y permite una interacción desde computador a través de varias librerías. Dentro del desarrollo, al haber escogido como herramienta principal PCL, era necesario trabajar con la librería Openni para la manipulación del Kinect.

En cuanto al control del movimiento del Kinect, se construyó una plataforma móvil para la cual se usó un motor PK299DBA que permite una gran precisión al tener la opción de micropasos para manejar este motor se usa un driver RBD245A-V, y finalmente para controlar el movimiento e interactuar con el software se usa un arduino UNO.

A continuación se ampliará la descripción de cada una de las partes ya mencionadas.

2.1 HARDWARE DEL SISTEMA

En la Ilustración 7. Hardware del sistema se observa el sistema completo a nivel del hardware usado durante el desarrollo. Esto incluye: motor PK299DBA, driver RBD245A-V, arduino UNO, computador y Kinect.

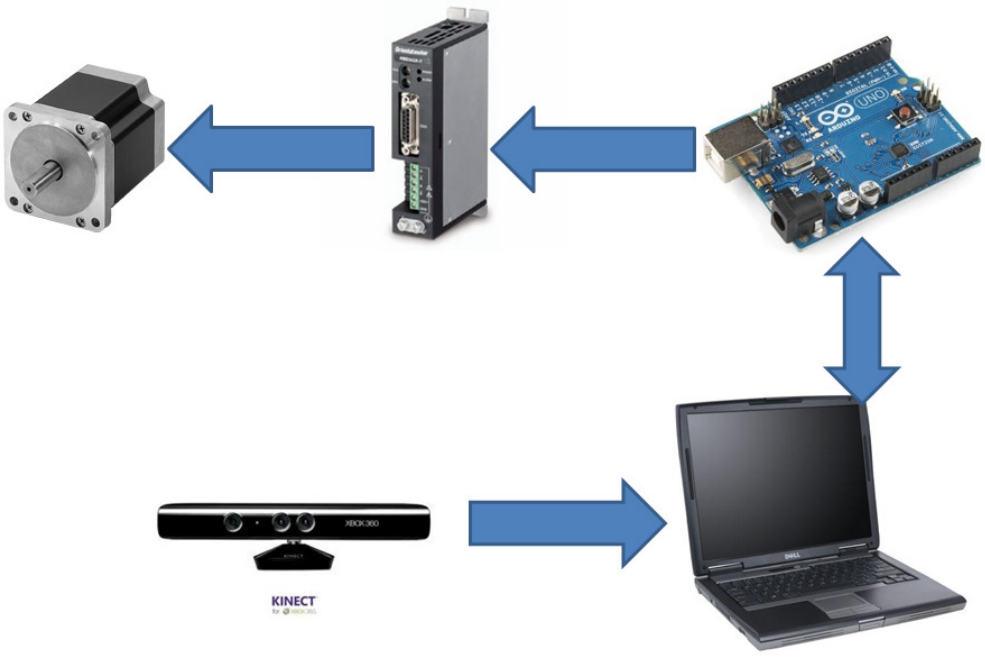


Ilustración 7. Hardware del sistema

Para poder dividir en subsistemas, se necesita conocer más del sistema mismo, es decir, las conexiones existentes, cómo se relaciona cada uno de estos componentes para poder dar una descripción más profunda.

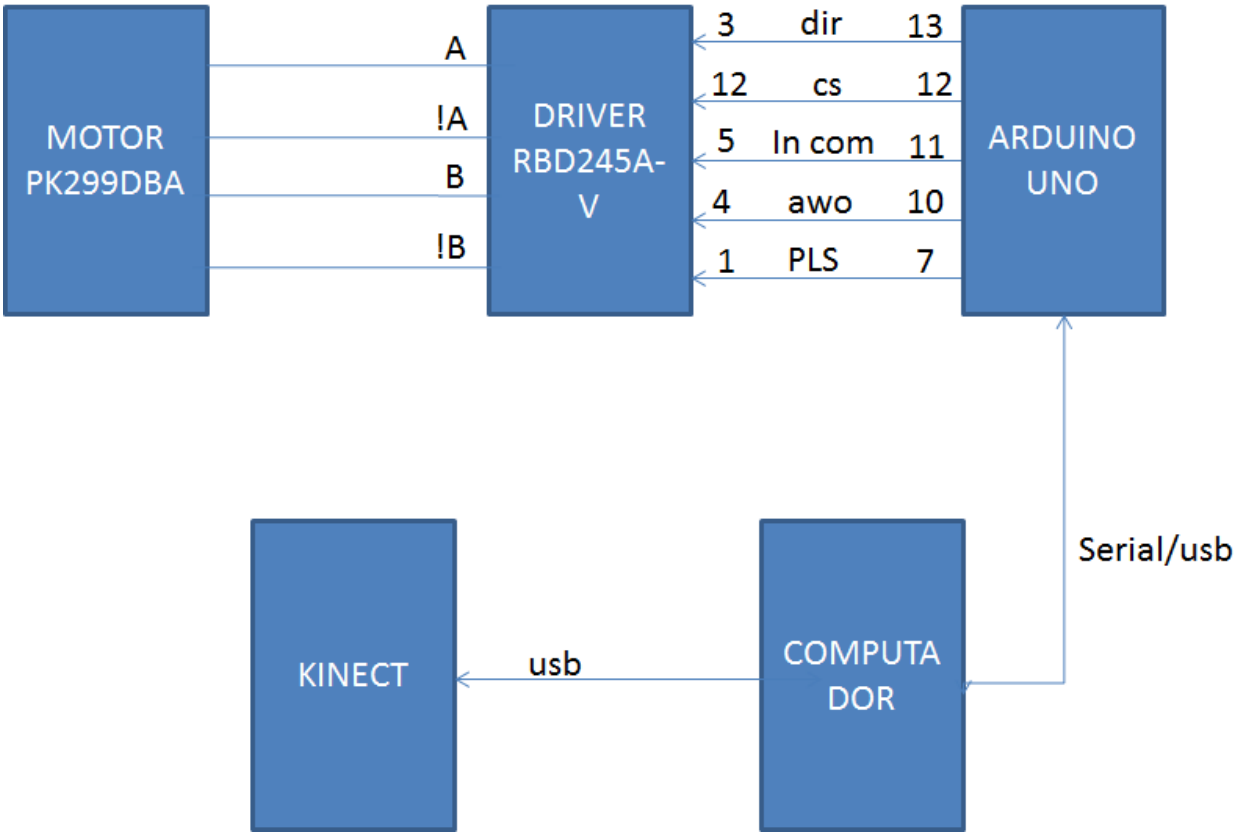


Ilustración 8. Detalle Hardware del sistema

2.1.1. Interfaz Motor-Driver

Entre el motor y el driver hay 4 fases que permiten desde el driver controlar la rotación del motor así como los pasos que este da, esto se hace a través de A, !A, B, !B.

Se puede en el driver seleccionar los micropasos que se desee usar. Los pasos son de 1.8°, lo que significa una resolución de 200 pasos, pero el micropaso más pequeño que tiene el driver es de 0.0140625° con una resolución de 25600 pasos, entre estos dos extremos hay 16 puntos medios, valores que se pueden observar en la **¡Error! No se encuentra el origen de la referencia.**.

Dial setting	Number of divisions	Resolution	Step angle
0	1	200	1.8°
1	2	400	0.9°
2	4	800	0.45°
3	5	1000	0.36°
4	8	1600	0.225°
5	9	1800	0.2°
6	10	2000	0.18°
7	16	3200	0.1125°
8	18	3600	0.1°
9	20	4000	0.09°
A	32	6400	0.05625°
B	36	7200	0.05°
C	40	8000	0.045°
D	64	12800	0.028125°
E	80	16000	0.0225°
F	128	25600	0.0140625°

Tabla 1. Micropasos

Para poder seleccionar los micropasos deseados entre los 16 niveles posibles se usa un switch giratorio conocido como DATA que permite realizar con facilidad la escogencia. En Ilustración 9. Funciones del driver se observa el switch DATA.

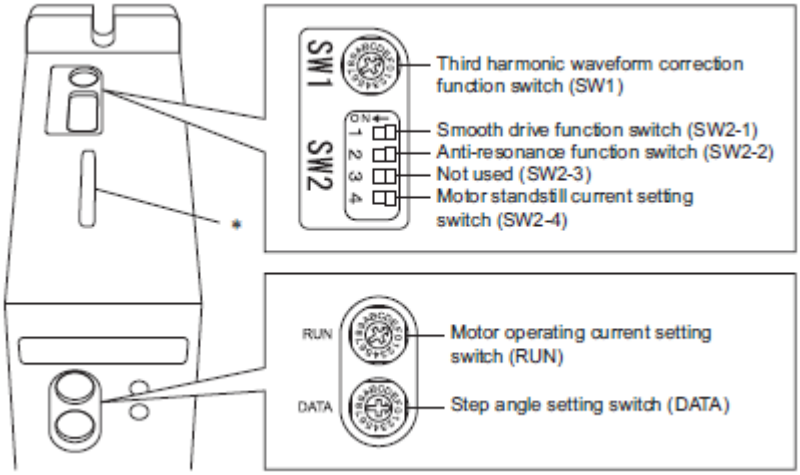
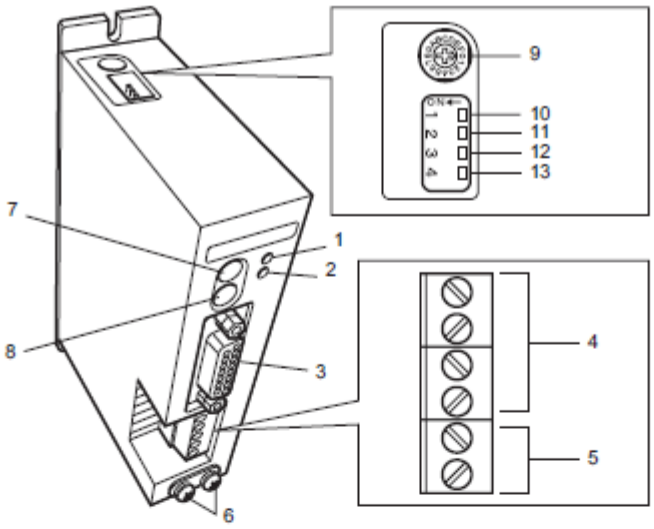


Ilustración 9. Funciones del driver

El motor recibe la alimentación de voltaje y corriente directamente del driver. El driver se alimenta a través de la conexión número 5 que se puede observar en la **¡Error! No se encuentra el origen de la referencia.**, para el correcto funcionamiento debe ser alimentado con 24V con una limitación en corriente de 5.2A.

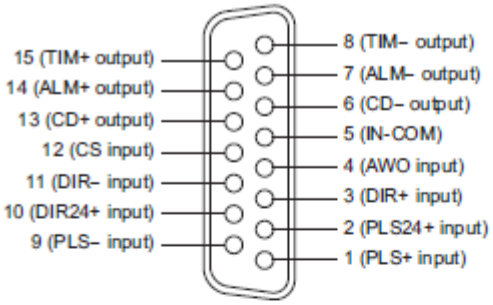


No.	Name	Description
1	POWER LED (green)	Lit when the power is on.
2	ALARM LED (red)	This LED blinks when the protective function was triggered and the ALARM output has turned OFF as a result. The triggered protective function can be checked by counting the number of times the LED blinks.
3	I/O signals connector (CN1)	Connect to I/O signals.
4	Motor terminal	Connect to motor.
5	Power supply terminal	Connect to power supply.
6	Protective Earth Terminal	Make ground connection by installing a grounding wire of AWG18 (0.75 mm^2) or larger.
7	Motor operating current setting switch (RUN)	Set the operating current of the motor.
8	Step angle setting switch (DATA)	Select a desired motor step angle from among the 16 preset levels.
9	Third harmonic waveform correction function (SW1)	This function sets a correction value to be applied to motor drive current waveforms.
10	Smooth drive function switch (SW2-1)	This function lets you reduce vibration and noise during low-speed operation.
11	Anti-resonance function (SW2-2)	This function reduces vibration during medium-speed operation.
12	SW2-3	Not used. (Keep this switch in the OFF position.)
13	Motor standstill current setting switch (SW2-4)	Set the current when the motor is at a standstill.

Tabla 2. Entradas y salidas del driver

2.1.2. Interfaz driver-Arduino

El arduino permite controlar el motor por medio de la programación del mismo, programación que se realiza sobre C con instrucciones propias del arduino. En la **¡Error! No se encuentra el origen de la referencia.** está la descripción de cada uno de los pines en el Driver.



Pin No.	Signal name	Description
1	PLS+ input	Pulse input
2	PLS24+ input	Pulse input (24 VDC)
3	DIR+ input	Rotation direction input
4	AWO input	All windings off input
5	IN-COM	Common input
6	CD− output	Current-cutback output
7	ALM− output	Alarm output
8	TIM− output	Excitation timing output
9	PLS− input	Pulse input
10	DIR24+ input	Rotation direction input (24 VDC)
11	DIR− input	Rotation direction input
12	CS input	Step angle switching input
13	CD+ output	Current-cutback output
14	ALM+ output	Alarm output
15	TIM+ output	Excitation timing output

Tabla 3. Asignación de pines en el Driver

Desde el arduino se programan las entradas:

- DIR: permite controlar la dirección del motor, para el proyecto no se contempla el cambio de dirección por lo que este valor desde el arduino se mantiene en un valor fijo.
- CS
- IN-COM: para el correcto funcionamiento la entrada común debe permanecer en valor alto.
- AWO: permanecerá en valor bajo durante el proyecto.
- PLS: la entrada de pulso debe estar de acuerdo a los valores necesarios para el movimiento, con un ciclo útil del 50%.

En la Tabla 4. Correspondencia de pines entre el driver y el arduino se puede ver las conexiones entre el driver y el arduino, en el caso del driver los pines son fijos, mientras que en el arduino puede escogerse la mejor distribución posible.

PIN DRIVER	PIN ARDUINO	NOMBRE
3	13	DIR
12	12	CS
5	11	INCOM
4	10	AWO
1	7	PLS

Tabla 4. Correspondencia de pines entre el driver y el arduino

2.1.3. Interfaz Arduino- computador

Arduino UNO es una placa electrónica basada en un microcontrolador ATmega328. Funciona con un oscilador de cristal, que trabaja como reloj interno, de 16 MHz. Este Arduino tiene una conexión USB que lo diferencia de otros Arduinos, y por lo cual se escogió trabajar con él, ya que incluye un ATmega8U2 programado como convertidor de USB a serie, por lo cual la conexión con el computador es directa.[1]

La fuente de poder del arduino UNO se selecciona automáticamente entre la conexión USB o el adaptador de poder externo, en el desarrollo de este proyecto se usa la conexión USB:

Para la programación se utiliza el software de Arduino. No se necesita un programador externo para la ATmega328 ya que viene con un código quemado previamente que le permite cargar nuevo código, es decir que trae un gestor de arranque.

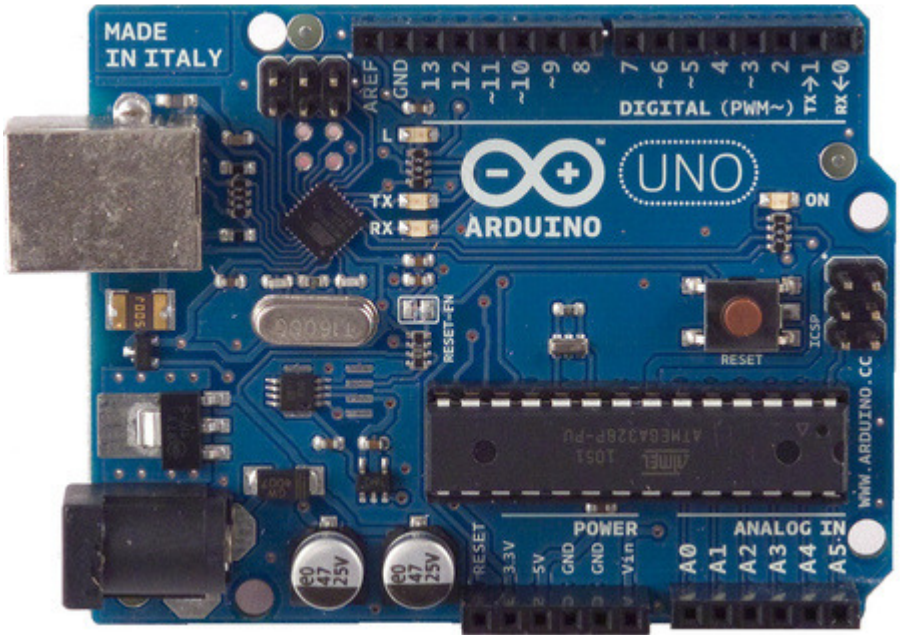


Ilustración 10. Arduino UNO

2.2 SOFTWARE DEL SISTEMA

[1] http://www.arduino.cc/en/Main/arduinoBoardUno [online] Arduino UNO. Copyright © 2014 Arduino

El sistema se puede dividir por capas para mayor claridad del desarrollo, como se observa en la Ilustración 11. Vista por capas del sistema.

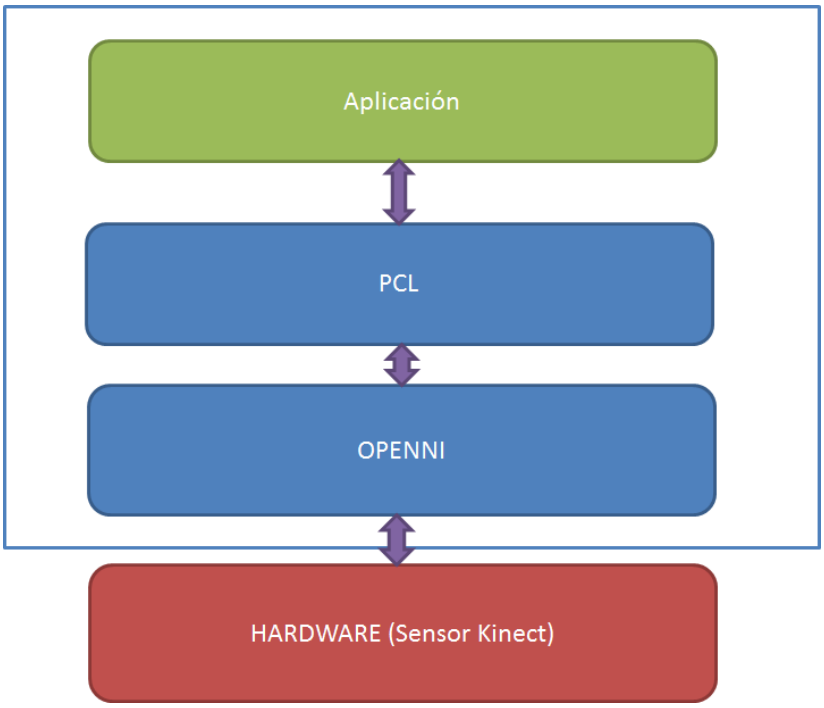


Ilustración 11. Vista por capas del sistema

El desarrollo se realizó sobre Ubuntu 11.04, y se probó igualmente sobre Ubuntu 11.10. Para el correcto funcionamiento se necesita una serie de dependencias que se explicarán en los siguientes numerales.

2.2.1 Openni

En Linux el openni tiene como prerrequisito de instalación:

- GCC 4.x
- Python 2.6+/3.x
- LibUSB 1.0.8
- FreeGLUT3
- JDK 6.0
- Mono (aunque aparece como opcional es necesario)

2.2.2 PCL

Para el correcto funcionamiento de PCL son necesarios algunos requerimientos:

- Boost
- Eigen
- FLANN
- VTK
- QHull

- OpenNI

2.2.3 Diagrama de subsistemas del aplicativo

La parte de software del aplicativo se divide en 6 módulos o subsistemas que interactúan entre ellos intentando mantener la mayor independencia posible. Estos subsistemas se muestran en la Ilustración 12. Diagrama de subsistemas del aplicativo.

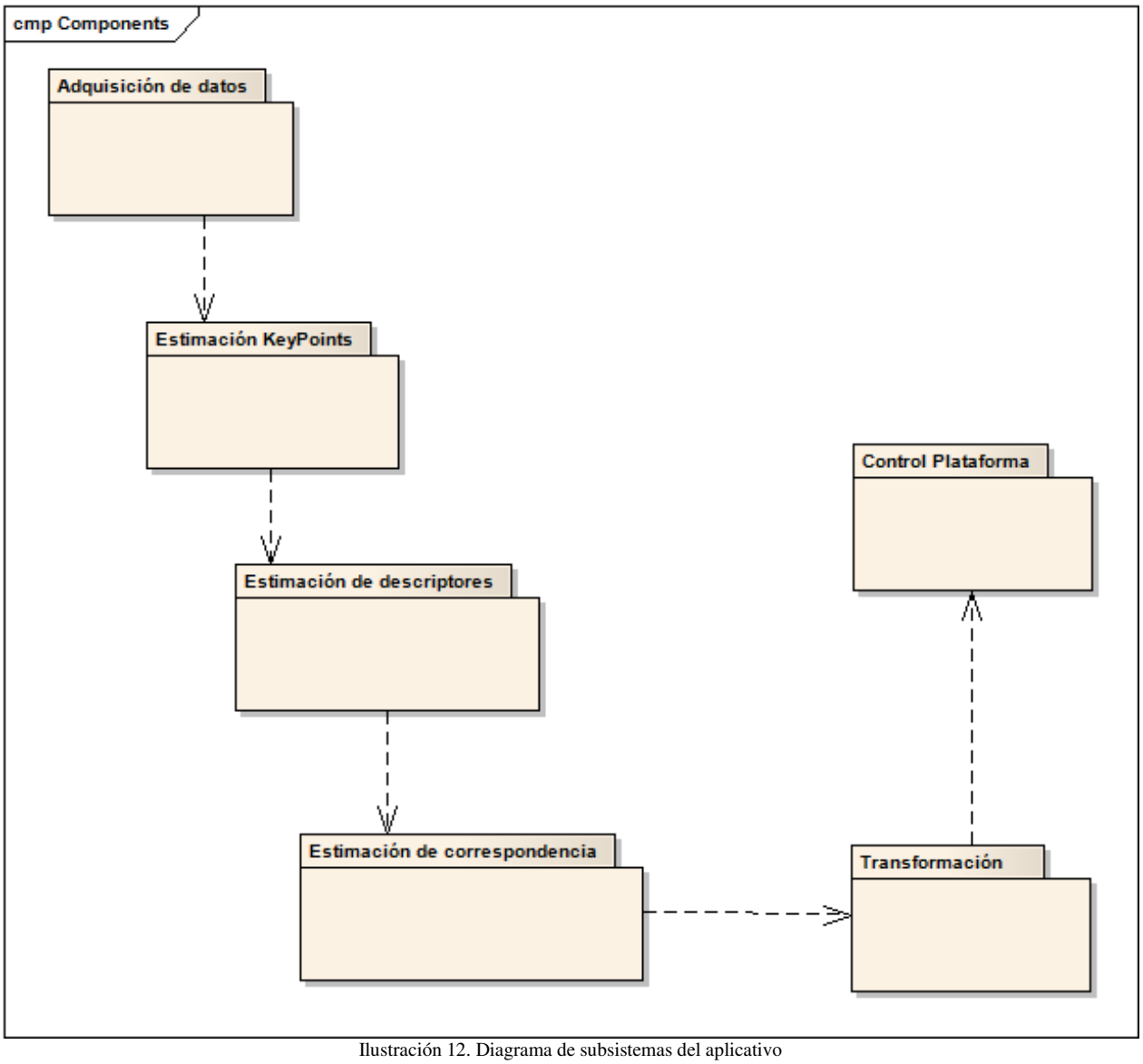


Ilustración 12. Diagrama de subsistemas del aplicativo

- Adquisición de datos: este es el módulo que interactúa con el Kinect para tomar las imágenes y pasarlas a nubes de puntos.
- Estimación *KeyPoints*: filtra la nube de puntos para encontrar los puntos de interés que sean representativos para la imagen, puntos distintivos.

- Estimación de descriptores: este módulo para cada punto de interés (*keypoint*) se obtiene un descriptor que contiene información sobre este punto en relación con los cercanos buscando hacerlo único.

- Estimación de correspondencia: el módulo busca la correspondencia entre imágenes basados en la similaridad dada por los descriptores.

- Transformación: este módulo debe obtener la transformación, minimizar el error y optimizar los puntos.

- Control plataforma: permite la interacción con la plataforma.

3. DESARROLLOS

Para el desarrollo del presente proyecto tiene aproximaciones de forma iterativa para conseguir el mejor resultado posible, modificando algunos de los subsistemas planteados en cada iteración.

De esta forma se siguió un ciclo de vida iterativo, donde se trabajó de forma modular por subsistemas. Lo anterior permitió estabilizar el proyecto por cada ciclo de vida obteniendo resultados que se fueron mejorando a lo largo de las pruebas.

En este capítulo se explicará el desarrollo paso a paso del proyecto teniendo en cuenta los ciclos que se llevaron a cabo.

Como primer paso dentro de lo propuesto en el desarrollo se quería tener un punto de control, de forma tal que los resultados fueran comparables en tiempos y exactitud. Por lo tanto se decidió realizar una primera aproximación con un algoritmo ICP, *iterative closest point*, que busca una posible correspondencia sobre todos los puntos, que para el caso es una nube de puntos de 302700, este proceso lo realiza de forma iterativa sobre todos los puntos de las nubes para encontrar la mejor transformación.

En el Anexo B se encuentra el código con comentarios de la aplicación por subsistema.

3.1. DESCRIPCIÓN DE SUBSISTEMAS

3.1.1. Adquisición de Datos

Con el Kinect podemos obtener una imagen RGB y otra de profundidad, para este proyecto estamos haciendo uso de la imagen de profundidad que la obtenemos a partir de la librería Grabber del PCL que permite la captura de las imágenes. Estas imágenes de profundidad se guardan en una nube de puntos.

3.1.2. Estimación de Keypoints

Los puntos de interés, o keypoints, se refiere a puntos de características especiales o de interés para poder procesar las imágenes de puntos. Estos se obtienen a través de fundamentos matemáticos que permiten definir la posición y las características de un punto en el espacio.

Para el desarrollo de este proyecto se usaron dos formas de obtener los puntos de interés.

NARF

Los puntos de interés obtenidos a partir de NARF se basan en "*Normal Aligned Radial Feature*". El trabajo con este tipo de descriptores tiene una particularidad y es que para poder calcular los NARF no se trabaja directamente sobre la nube de puntos, sino a través de una imagen de rango. Los puntos NARF corresponden a puntos situados cerca de esquinas o fronteras.

Para poder obtener puntos de interés adecuados es necesario trabajar sobre una imagen de rango con suficiente abertura para que cubra un espacio suficiente para encontrar puntos de interés.

SIFT

Los puntos de interés obtenidos a partir de SIFT se describen a través de una región circular que debe tener orientación, usa sus coordenadas y su ángulo, para detectar puntos de interés. SIFT es invariante ante cambios de luz o perspectiva.

3.1.3. Estimación de Descriptores

Para la estimación de descriptores se está haciendo uso de FPFH, Fast Point Feature Histograms, este tipo de estimación, descrita brevemente en el marco teórico, permite un procesamiento muy rápido, para aplicaciones, como este caso, que buscan acercarse a una buena transformación en tiempos muy cercanos al real.

3.1.4. Estimación de Correspondencias

La estimación de la correspondencia busca a través de los descriptores encontrar qué puntos se corresponden entre ellos, de la mejor forma posible, y una vez se encuentran estas correspondencias se calcula la matriz de transformación que nos permite saber cómo se movió en los tres ejes y su rotación.

RANSAC

Este algoritmo es ampliamente usado, permite a través de iteraciones buscar las mejores correspondencias. En PCL este algoritmo ya se encuentra implementado y sus librerías son de fácil acceso. Es un algoritmo determinístico, por lo que se puede conseguir muy buenas aproximaciones.

Dentro de las funciones implementadas en la librería de PCL ya se encuentra la consecución de la transformación, y permite adicionalmente obtener un valor de "*FitnessScore*" que mide qué tan buena es la transformación que se obtuvo.

REJECT BAD CORRESPONDENCES AND RIGID TRANSFORMATION

En este caso buscamos calcular las correspondencias posibles entre los descriptores que se han obtenido. Una vez se consiguen las correspondencias se aplica una función que rechaza aquellas correspondencias

que pueden afectar de forma negativa la transformación, dejando solo aquellas que tienen más posibilidades de ser vecinas las unas de las otras.

Con esta información de correspondencias, se puede calcular una transformación rígida para obtener la matriz de transformación.

3.1.5. Transformación

Una vez se obtiene la matriz de transformación, esta se debe aplicar a la matriz fuente. En este punto se debe tener especial cuidado, porque una de las matrices es aquella que queda de referencia, y la otra es sobre la cual se está buscando la transformación y a la que hay que aplicarle la transformación para poder obtener una correcta transformación.

3.1.6. Control Plataforma

El control de la plataforma se realiza desde la aplicación, este control inicializa la rotación de la plataforma, la plataforma a través de Arduino UNO informa que terminó la rotación, en el momento en que termina la rotación, la aplicación procede a tomar la imagen correspondiente.

Para el desarrollo de la plataforma y su control se adquirieron dos fuentes, con referencia "Mean well RS-75-12, que conectadas en paralelo alimenta al motor, a través del Driver se puede graduar de forma manual, esto no puede ser alterado desde la aplicación, los grados de rotación, como se explica en el capítulo Especificaciones.

Sobre el motor a través de una plataforma de madera se coloca el Kinect de forma estable para su giro. El motor está en capacidad de girar de forma precisa aún con el peso del Kinect.

En el Anexo C se puede observar la configuración usada sobre el Driver y la programación en el Arduino UNO.

3.2. PROCESO DE DESARROLLO

El proceso de desarrollo, como se mencionó anteriormente, está basado en módulos o subsistemas que pueden modificarse a necesidad, reduciendo el impacto sobre el desarrollo.

En otras palabras la descripción de este proceso sería la siguiente:

1. Definición de las necesidades: en este punto se plantearon los objetivos claros de proyecto, de forma tal que estuviera correctamente definido, lo que se logró a través del Poster y el anteproyecto presentado. Este punto no está dentro de la iteración del ciclo de vida desarrollado, y se realizó una sola vez para tener claro la hoja de ruta a seguir.

2. Diseño de la solución: Durante esta etapa, que se explica a profundidad en el capítulo de Especificaciones, se definieron los subsistemas a intervenir y la vista macro del proyecto, lo que permitió detectar los puntos con los que se podía trabajar para obtener mejores resultados. Este punto tampoco está dentro de la iteración, y se dejó definido al principio del proyecto.

(A partir de este punto está la parte iterativa)

3. Análisis: Una vez planteado el diseño, se realiza un análisis a partir de la investigación hecha que permite encontrar qué métodos para cada subsistema aplicar y determinar cuáles subsistemas son susceptibles a ser modificados. Durante el primer análisis se descartaron los siguientes subsistemas como modificables, de forma tal que estos fueron constantes a lo largo del desarrollo y las pruebas:

    - Adquisición de Datos.
    - Control de plataforma: En este punto es importante aclarar que no se modificó la forma de realizar el control de la plataforma, pero sí se modificaron los parámetros de rotación en las pruebas.

4. Codificación: Una vez realizado el análisis de las posibles mejoras o implementaciones, se deben codificar.

5. Pruebas: En este punto las pruebas se realizaron en dos niveles diferentes, unas de forma local o unitaria donde se probó el funcionamiento de los subsistemas afectados en codificación y que fueron desarrollados nuevamente, y otro integral que probó el funcionamiento del código en conjunto.

    - Pruebas Unitarias: Estás pruebas se realizaron únicamente sobre el subsistema afectado, de forma tal que se pudiera corroborar el funcionamiento del mismo de forma aislada.
    - Pruebas Integrales: el código en conjunto fue probado para revisar el funcionamiento total obtenido, y los resultados con las modificaciones o nuevo código incluido.

6. Validación: con los resultados de las pruebas obtenidos en el punto anterior, se realiza validación de los mismos para revisar a través de diferentes mecanismos si los resultados se acercan a lo esperado.

7. Evolución: En caso tal de obtener como resultado de la validación que es posible realizar alguna mejora sobre los subsistemas afectables, se toma la decisión de retomar el ciclo.

Uno de los mayores riesgos de un ciclo de vida iterativo de desarrollo es no reconocer un fin de las iteraciones para dar resultados concretos. Por este motivo, al iniciar el desarrollo se tomó la decisión de implementar máximo dos codificaciones diferentes por subsistemas, y que las iteraciones totales de acuerdo al tiempo del cronograma plateado no podrían superar las 6 iteraciones.

4. ANÁLISIS DE RESULTADOS

Como la primera prueba para conseguir un punto de partida y referencial se realizó con ICP sobre dos nubes completas, cada nube de puntos obtenida con el Kinect tiene 302700 puntos, la búsqueda de la matriz de correspondencia. Sin embargo, la prueba no pudo ser terminada debido al largo procesamiento necesario para completarse. Luego de 2 días ininterrumpidos no se consiguió transformación.

Para realizar un análisis de resultados comparativos las pruebas aquí presentadas se tomaron sobre las mismas nubes de puntos.

El código por Iteración se puede encontrar en el Anexo B.

Como referencia el poder controlar el giro del motor y por tanto la tomas de imágenes nos permite dejar como referencia una rotación de 5 grados sobre el eje Z. Lo anterior es equivalente a la matriz siguiente, que nos sirve de referencia.

0,996195	-0,08716	0	0
0,087156	0,996195	0	0
0	0	1	0
0	0	0	1

Tabla 5. Matriz de rotación sobre eje Z para 5°

Con esta matriz referencial vamos a calcular la media de calidad sobre cada una de las iteraciones, comparándola con la matriz de transformación obtenida para ellas. Esta medida se obtiene como la suma de la diferencia en valor absoluto de cada campo. Este valor debe tender a cero, es decir, que sea lo más cercano posible a la matriz referencial.

Para que las imágenes sean muchos más claras en la ilustración siguiente se observa la biblioteca sobre la que se realizó la toma de imágenes, de forma tal que sirva de referencia en la estructura que se está buscando reconstruir.

Ilustración 13. Imagen de referencia

4.1. RESULTADOS PRUEBAS ITERACIÓN 1

En la primera iteración realizada se trabajó con la obtención de *Keypoints* a través de NARF, se aplicó el filtro *Voxel Grid*, se obtuvo los descriptores con PFPH, y la correspondencia con *Bad Reject.* Esta fue el primer acercamiento para hacer reconstrucción de espacios cerrados, donde se buscó obtener resultados de correspondencias.

El filtro aplicado reduce la nube a menos de la mitad del tamaño. En cuanto al rango de para obtener los *Keypoints* a través de NARF, se usa un ángulo de 57 grados en horizontal y 43 grados en vertical.

En esta prueba se obtuvieron los valores de la tabla 6.

	IMAGEN 1	IMAGEN 2
Tamaño Pointcloud	307200	307200
Tamaño Pointcloud filtrada	124976	139037
NARF: Puntos dentro del rango	2450	2450
NARF: Keypoints	34	28
Correspondencias	10	

Tabla 6. Resultado Pruebas Iteración 1

La matriz resultante no se corresponde con la matriz referencial, por lo que denota que la transformación obtenida no es muy cercana a la esperada, como se observa en la Tabla 7.

0.983911	0.17226	0.0473909	-0.0818908
-0.164012	0.976074	-0.142757	0.310024
-0.0708486	0.132688	0.988622	0.022609
0	0	0	1

Tabla 7. Matriz de transformación Iteración 1

A nivel visual en las imágenes es evidente que la transformación realizada no corresponde con lo esperado. Se observa una rotación sobre el eje Y que hace que se pierda las líneas horizontales de la imagen, lo anterior se muestra en la Ilustración 14. En la ilustración 15 se observa una rotación a nivel del eje Z, sin embargo está rotación no es la esperada.

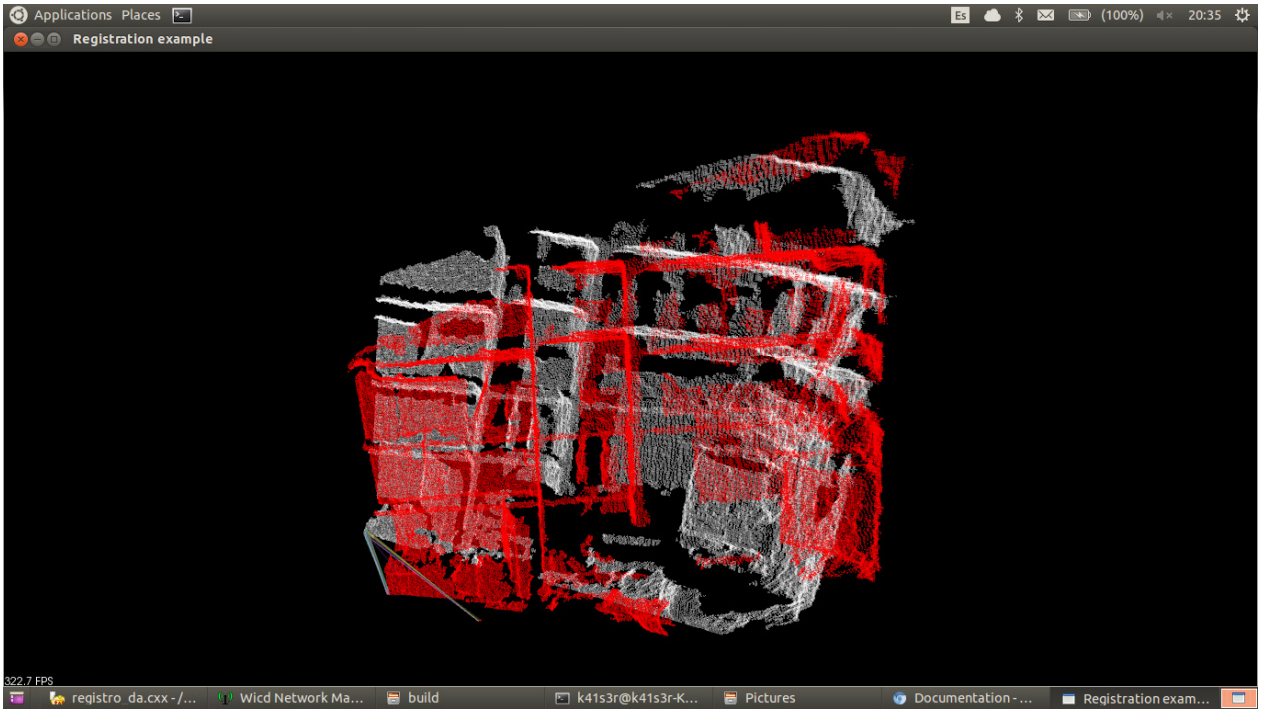


Ilustración 14. Resultado Iteración 1 vista frontal

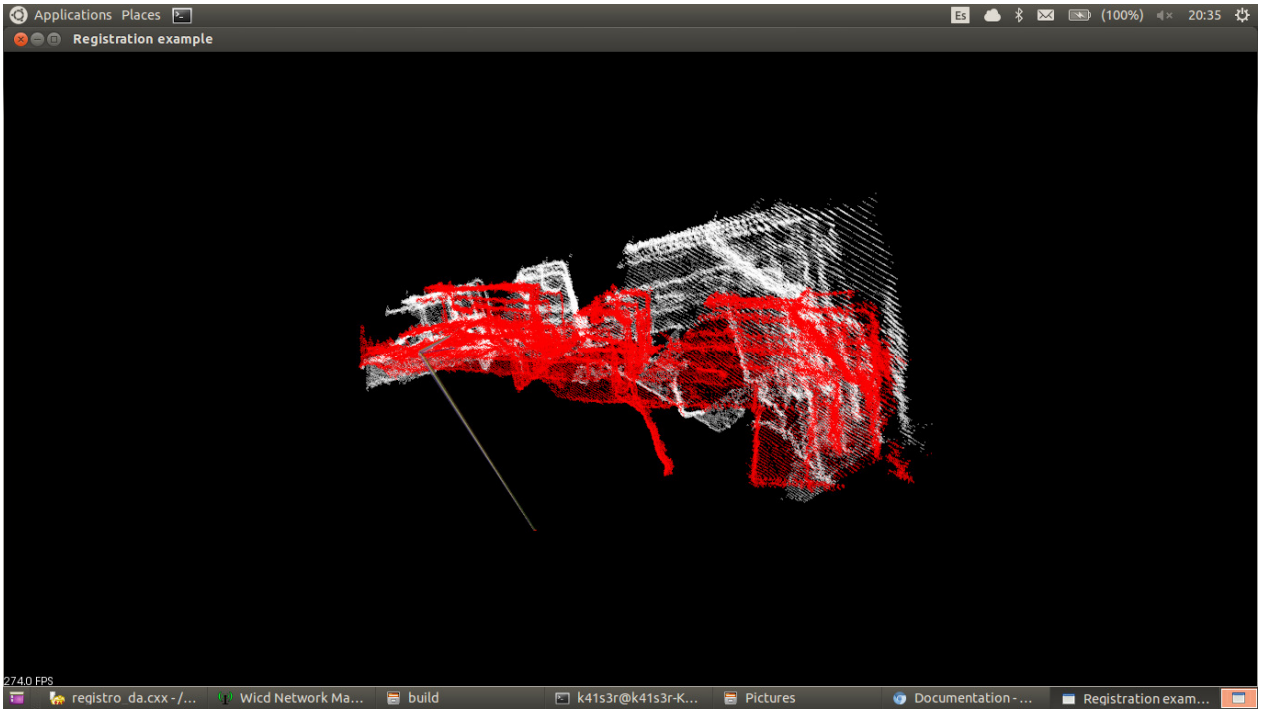


Ilustración 15. Resultado Iteración 1 Vista superior

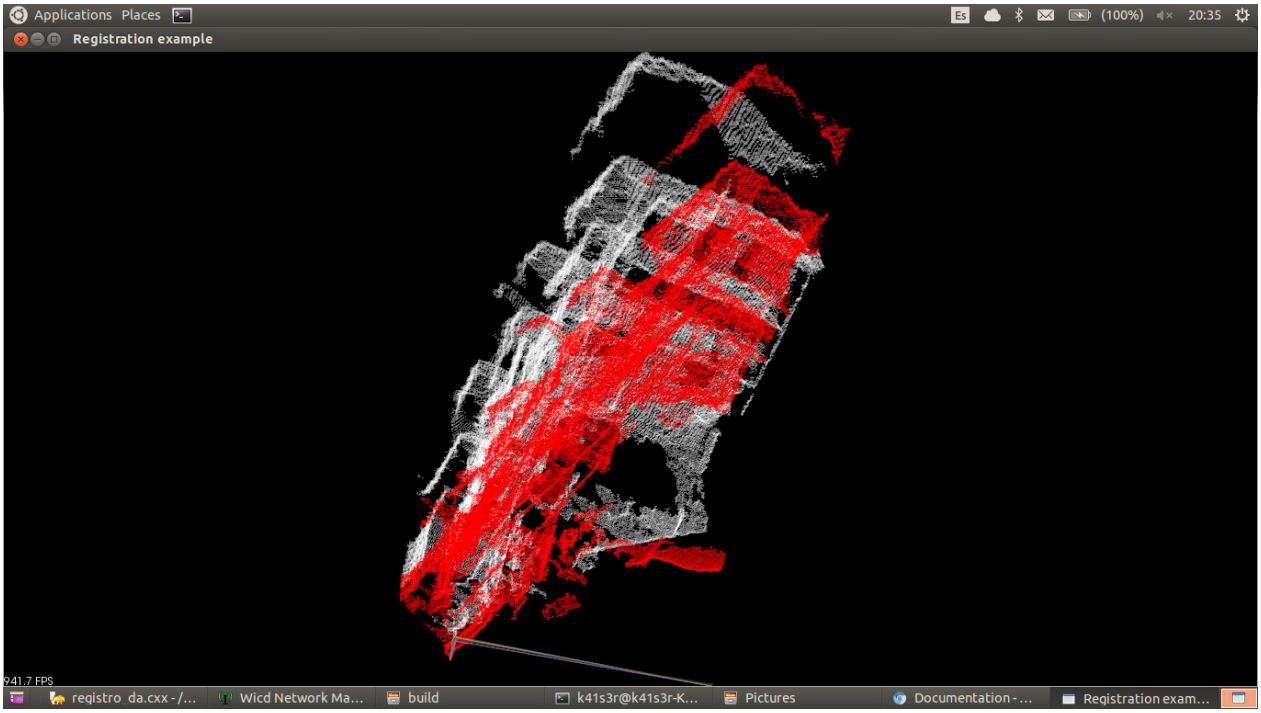


Ilustración 16. Resultado Iteración 1 Vista Lateral

La medida de calidad planteada da como valor 1.3625793.

4.2. RESULTADO PRUEBAS ITERACIÓN 2

Para la segunda iteración dejó de utilizarse filtro para obtener la información, pero se mantuvo el resto del funcionamiento. Lo anterior se realizó con el objetivo de obtener más puntos y poder conseguir una transformación más acertada. En cuanto al rango de para obtener los *Keypoints* a través de NARF, se mantuvo un ángulo de 57 grados en horizontal y 43 grados en vertical.

En esta prueba se obtuvieron los valores de la tabla 8. Con respecto a los resultados de la iteración 1, que se encuentra en la tabla 2, se observa un cambio en los KeyPoints por imagen y un aumento en las correspondencias.

	IMAGEN 1	IMAGEN 2
Tamaño Pointcloud	307200	307200
NARF: Puntos dentro del rango	2655	2655
NARF: Keypoints	28	31
Correspondencias	**12**	

Tabla 8. Resultado Prueba Iteración 2

En cuanto a la matriz de transformación para la iteración 2 se puede observar en la Tabla 9, está matriz está mucho más cercana a la esperada, y se observa una rotación que es buena aproximación a la esperada sobre el eje Z. Mejorando los resultados sobre la iteración anterior.

0.995276	0.0653291	0.071818	-0.275738
-0.0604979	0.995883	-0.0675076	0.191641
-0.0759323	0.0628437	0.995131	0.00205719
0	0	0	1

Tabla 9. Matriz de Transformación Iteración 2

Visualmente en la Ilustración 17 sobre la vista frontal se observa una mejoría con respecto a lo obtenido en la ilustración 14, las líneas horizontales y verticales que rigen la imagen están más cercanas. En la ilustración 18 se puede observar la rotación sobre el eje Z.

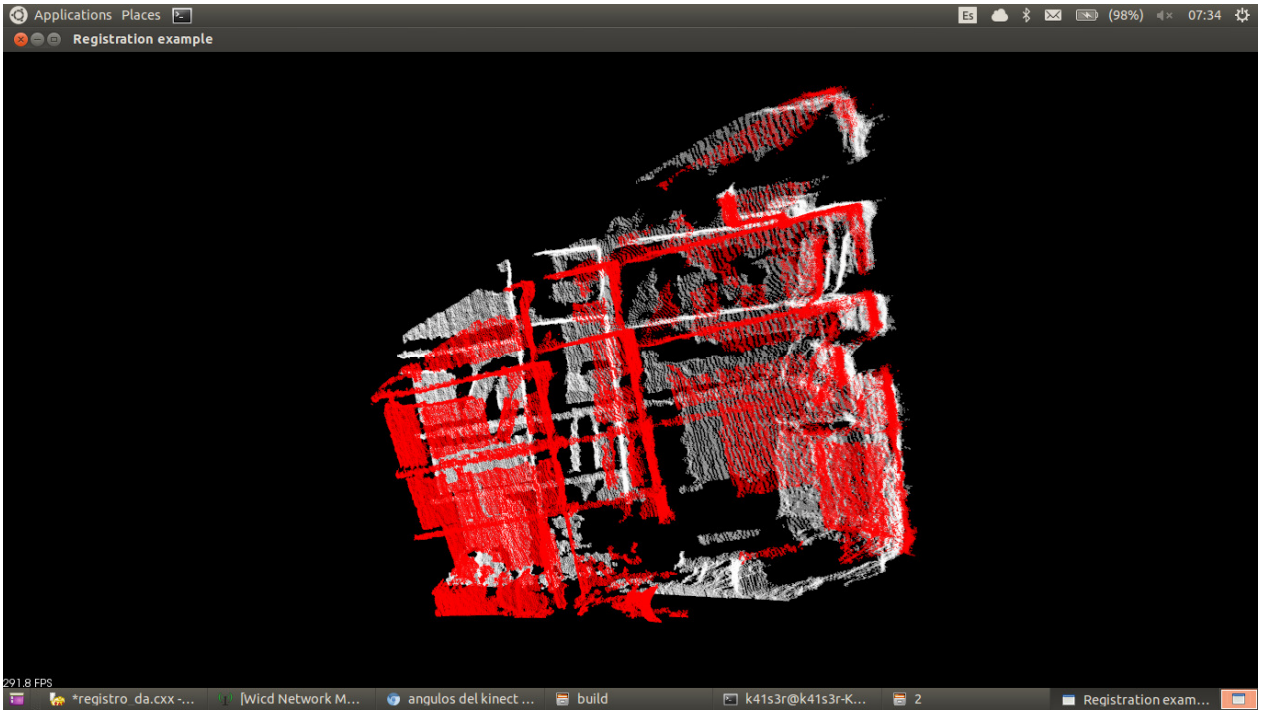


Ilustración 17. Resultado Iteración 1 Vista Frontal

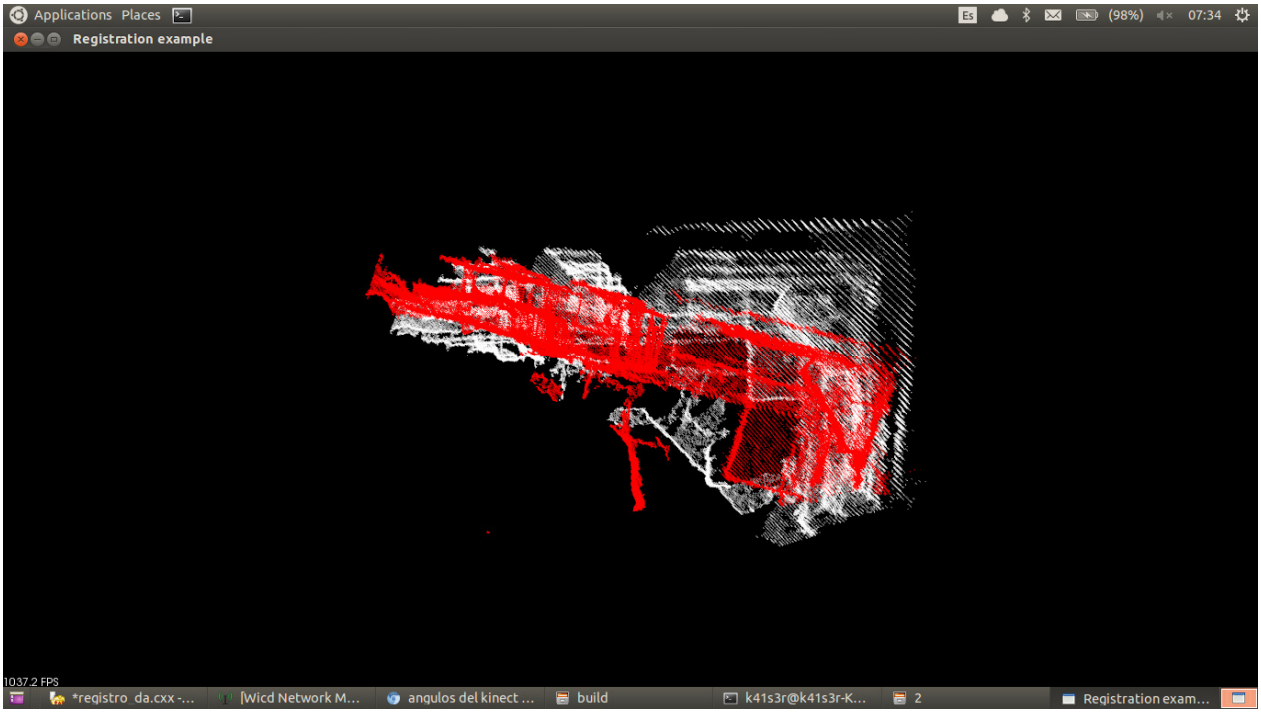


Ilustración 18. Resultado Iteración 2 Vista Superior

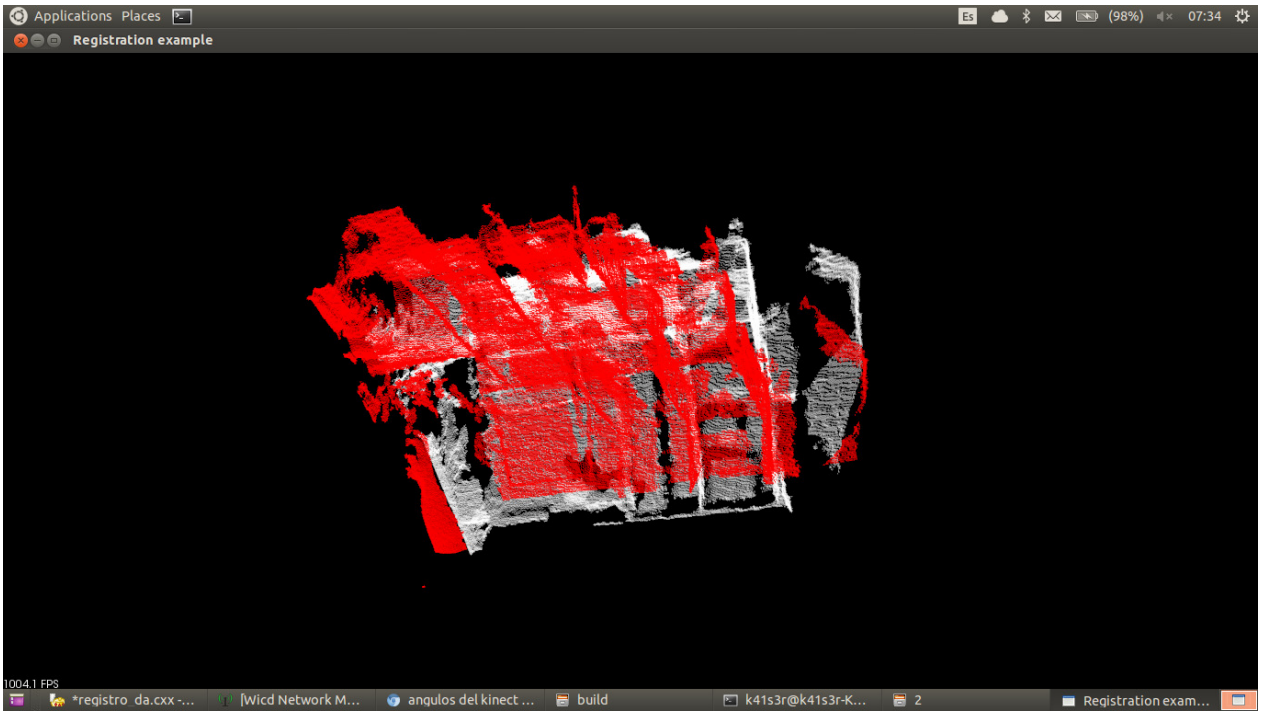


Ilustración 19. Resultado Iteración 2 Vista Lateral

La medida de calidad para esta iteración es 1,05378079 este valor es más cercano a cero que el de la iteración anterior, lo que deja al descubierto que este iteración es más cercana a la matriz referencial esperada, por lo que se puede decir que es una mejor reconstrucción.

4.3. RESULTADO PRUEBAS ITERACIÓN 3

En la tercera iteración, al obtener mejores resultados sin filtro se decidió no mantener el filtro para las pruebas posteriores. En búsqueda de obtener mejores resultados, para esta prueba se buscaron correspondencias con los *Keypoints* obtenidos con NARF, los descriptores con FPFH, pero para las correspondencias se hizo uso de RANSAC. En cuanto al rango de para obtener los *Keypoints* a través de NARF, se mantuvo un ángulo de 57 grados en horizontal y 43 grados en vertical.

Estos son los primeros resultados usando RANSAC, y están en la tabla 10. Con respecto a los resultados de la iteración 2, se observa que como no se han cambiado NARF ni su rango para los *Keypoints* estos son los mismos. Y aunque se cambió la forma de obtener la correspondencia, los puntos de correspondencia obtenidos son los mismos.

	IMAGEN 1	IMAGEN 2
Tamaño Pointcloud	307200	307200
NARF: Puntos dentro del rango	2655	2655
Keypoints	28	31
Correspondencias	**12**	

Tabla 10. Resultados Iteración 3

Al observar la matriz de transformación, Tabla 11, no hay semejanza alguna con la matriz de transformación esperada en cuanto a la rotación sobre el eje Z, Ecuación 1. En este sentido hasta el momento esta es la aproximación más lejana a la esperada obtenida.

-0.415632	-0.787468	-0.455131	0.458281
-0.677687	-0.065624	0.732416	-0.77892
-0.606622	0.612852	-0.506381	1.75085
0	0	0	1

Tabla 11. Matriz Transformación Iteración 3

En la ilustración 20 se observa cómo la segunda nube que es transformada se desvía completamente de la imagen de referencia, y cambia de plano, alejándose de realizar una correspondencia entre las dos imágenes.

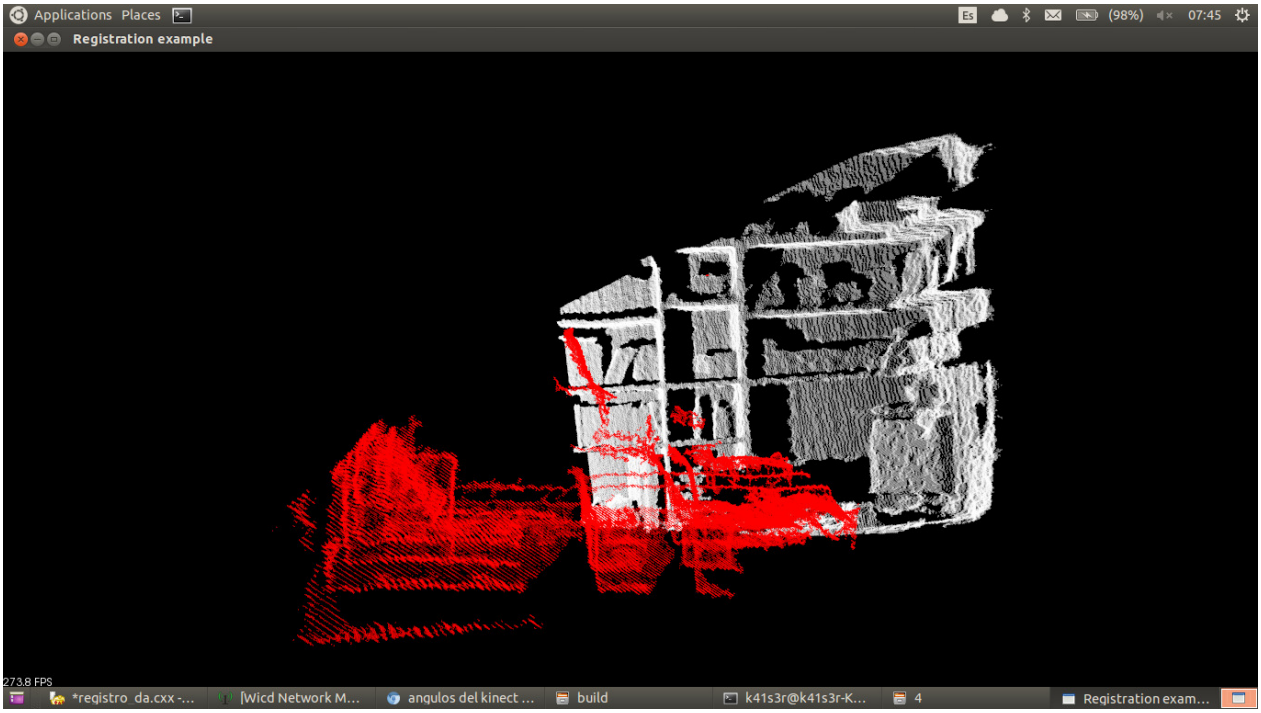


Ilustración 20. Resultado Iteración 3 Vista Frontal

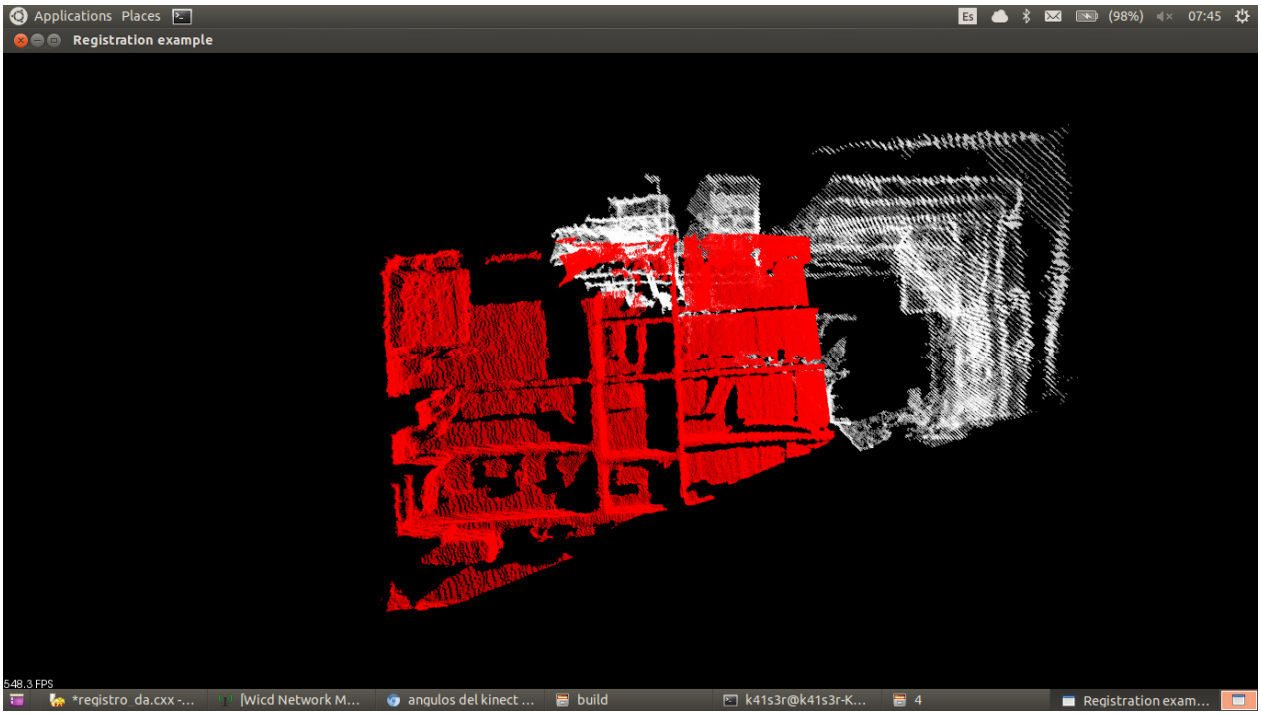


Ilustración 21. Resultado Iteración 3 Vista Superior

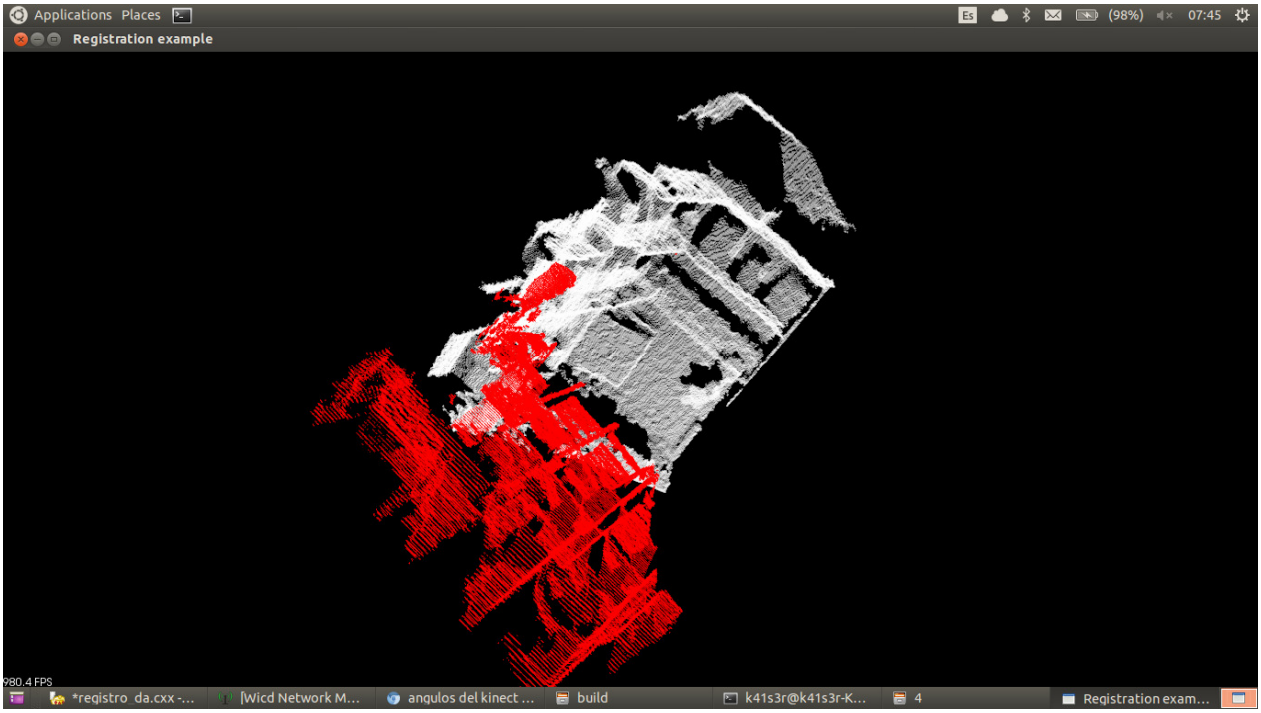


Ilustración 22. Resultado Iteración 3 Vista Lateral

La medida de calidad para esta iteración es 175,094 este valor es mucho mayor a las iteraciones anteriores, y este valor está lejos del valor de calidad esperado, que es cero, por lo cual esta iteración no nos da buena transofrmación.

4.4. RESULTADO PRUEBAS ITERACIÓN 4

Para la cuarta iteración se buscó aumentar el rango para obtener los NARF, así se aumentó a 120° por 120°, lo anterior para obtener un mayor número de *keyPoints*.

Se volvió a realizar la prueba de correspondencia con RANSAC con aumentando como se mencionó el rango de NARF. Como se observa en la tabla 12 al aumentar los ángulos para el rango de imagen de NARF los puntos dentro del rango aumenta, así mismo aumentaron los *keypoints*, aunque no significativamente.

	IMAGEN 1	IMAGEN 2
Tamaño Pointcloud	307200	307200
NARF: Puntos dentro del rango	3100	3100
NARF: Keypoints	35	34
Correspondencias	**10**	

Tabla 12. Resultado Iteración 4

La matriz de transformación para esta iteración se puede observar en la Tabla 13, y aunque esta cambió significativamente con respecto a la iteración anterior, sigue sin acercase a la transformación referencial esperada.

-0.465921	-0.317987	0.825713	-1.07944
-0.147799	0.948051	0.281702	-0.246789
-0.872396	0.00921181	-0.488715	2.12474
0	0	0	1

Tabla 13. Matriz de Transformación Iteración 4

En la ilustración 23 se observa que la segunda imagen en rojo ya no está completa mente horizontal, como aparecía en la ilustración 20, su transformación sigue siendo muy lejana de lo esperado, visualmente no está ni siquiera orientado en el mismo plano.

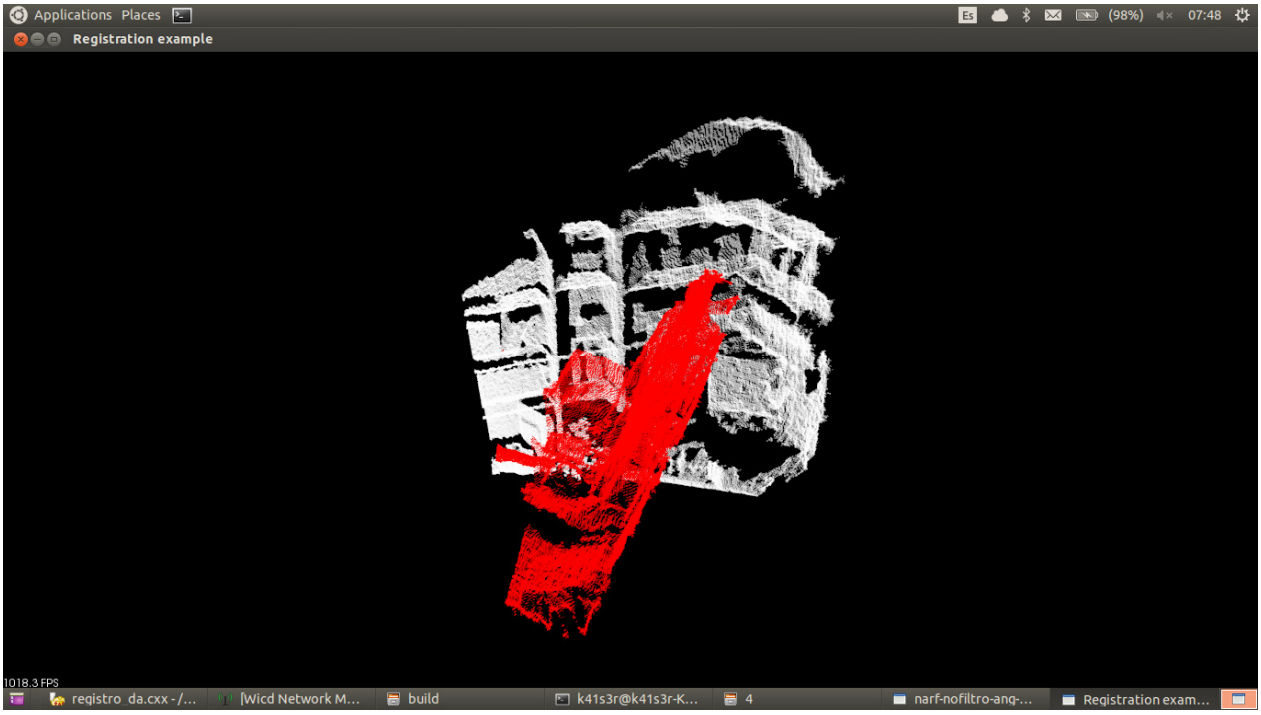


Ilustración 23. Resultado Iteración 4 Vista Frontal

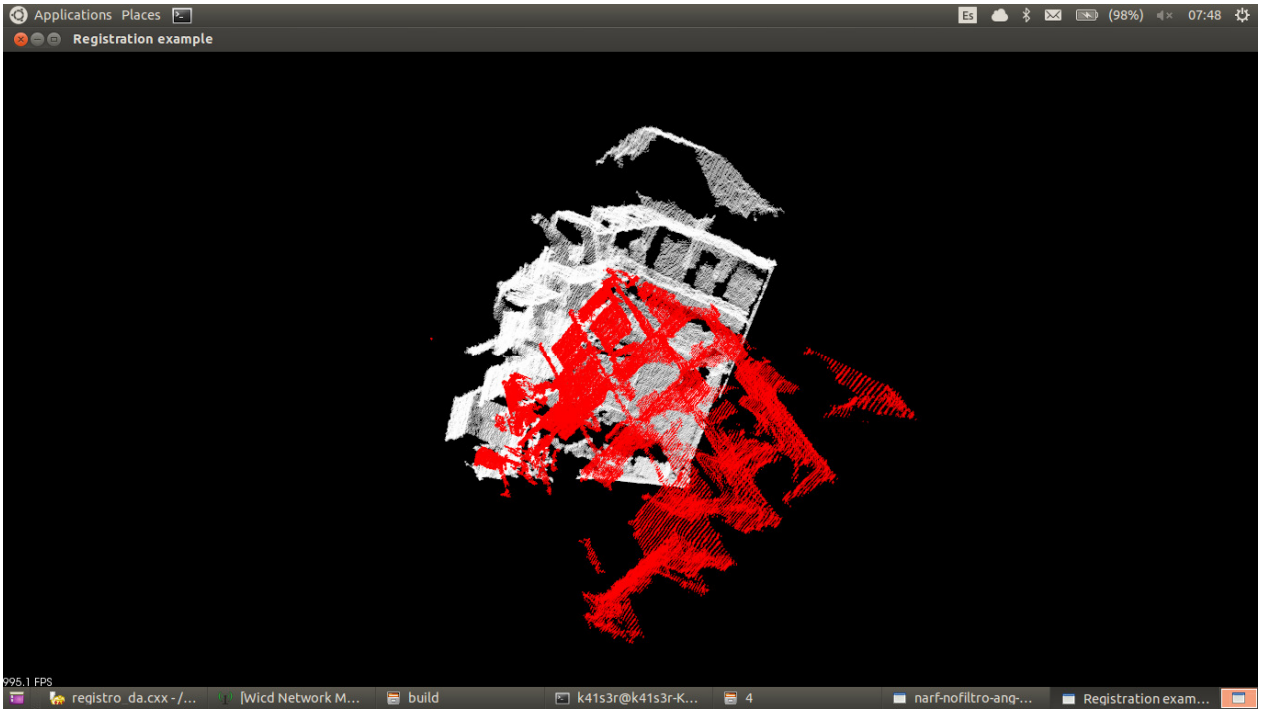


Ilustración 24. Resultado Iteración 4 Vista lateral

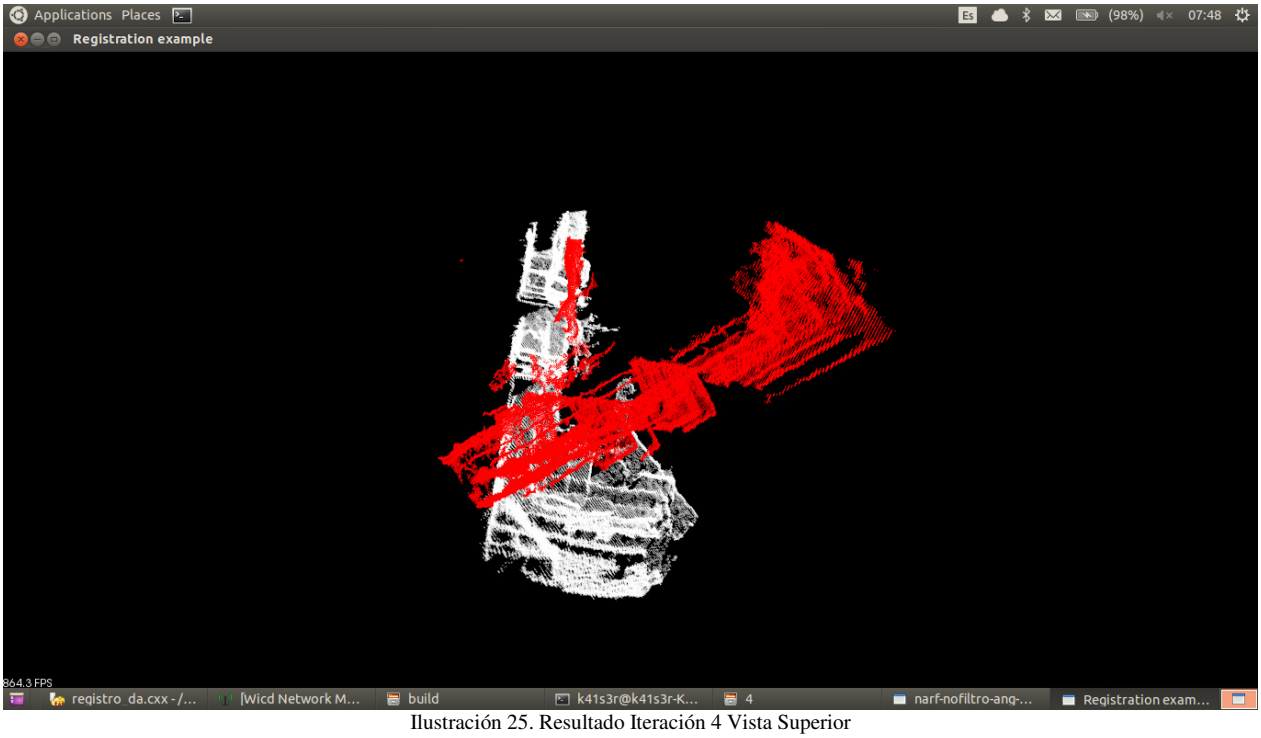


Ilustración 25. Resultado Iteración 4 Vista Superior

La medida de calidad para esta iteración es 320,424 valor que hasta el momento demuestra la transformación más alejada de lo esperado. Con este valor y el de la iteración 3, los dos con RANSAC tenemos que este algoritmo nos da una mediad de calidad muy mala que se evidencia en reconstrucciones que cualitativamente son muy lejanas a lo esperado..

4.5. RESULTADO PRUEBAS ITERACIÓN 5

Debido a que los resultados con RANSAC no consiguieron una buena correspondencia, aun aumentando el rango de imagen para los NARF momento, se aplicó con el aumento del rango de imagen y aplicando *BadRejectCorrepsondences*. Para la quinta iteración se mantiene el aumento del rango para obtener los NARF, así se aumentó a 120° por 120°, lo anterior para obtener un mayor número de *keyPoints*.

Al observar la tabla 14 los resultados obtenidos son exactamente igual a este nivel que los resultados de la iteración anterior que están en la tabla 12. Ya que hasta el punto de obtener los *Keypoints* el proceso entre esta y la anterior iteración es el mismo.

	IMAGEN 1	IMAGEN 2
Tamaño Pointcloud	307200	307200
NARF: Puntos dentro del rango	3100	3100
Keypoints	35	34
Correspondencias	**10**	

Tabla 14. Resultado Iteración 5

Sin embargo aunque los resultados hasta este punto son los mismos, al obtener la matriz de transformación por otro método se obtiene una mejor transformación, que se puede observar en la matriz de la Tabla 15.

0.999478	0.0149228	0.0286448	-0.250348
-0.0142705	0.999637	-0.0228375	0.0404942
-0.0289751	0.0224168	0.999329	0.0403267
0	0	0	1

Tabla 15. Matriz de transformación Iteración 5

Al observar la ilustración 26 se observa una muy buena alineación de las líneas horizontales, en las líneas verticales se puede observar un corrimiento que puede asociarse con la transformación rígida de la rotación sobre el eje Z. Esta misma rotación se ve de forma más clara en la ilustración 27.

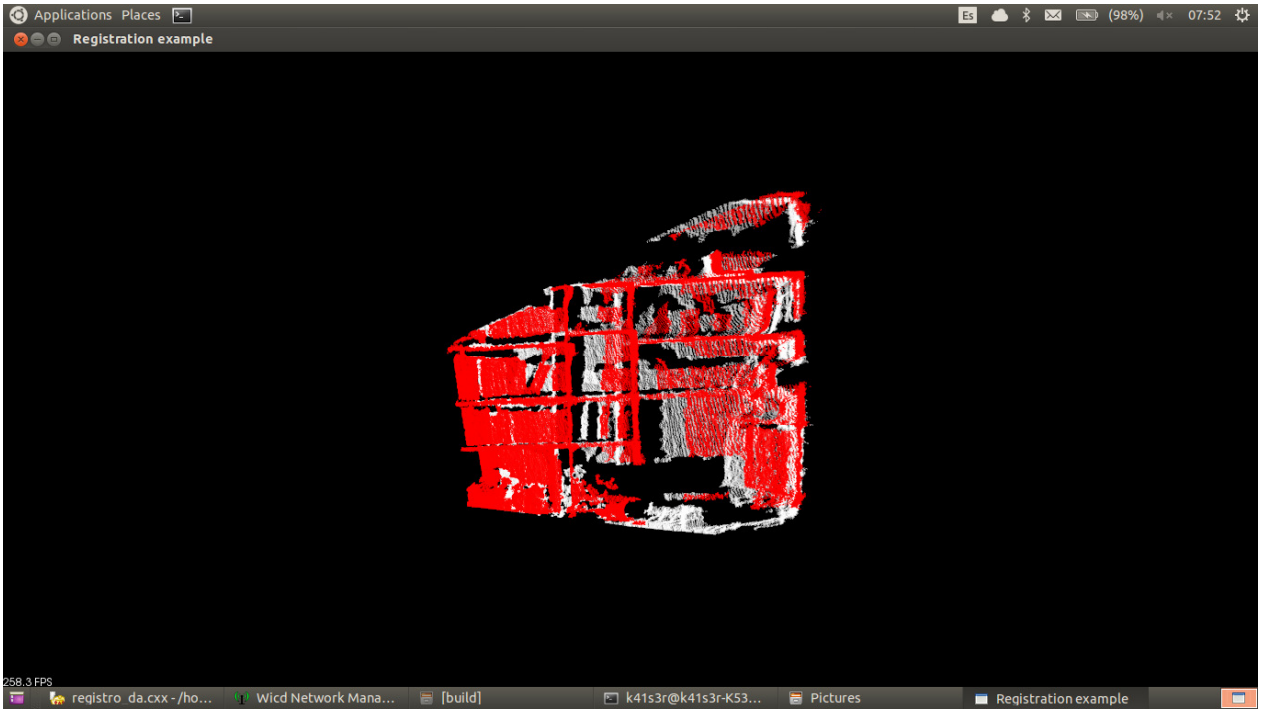


Ilustración 26. Resultado Iteración 5 Vista Frontal

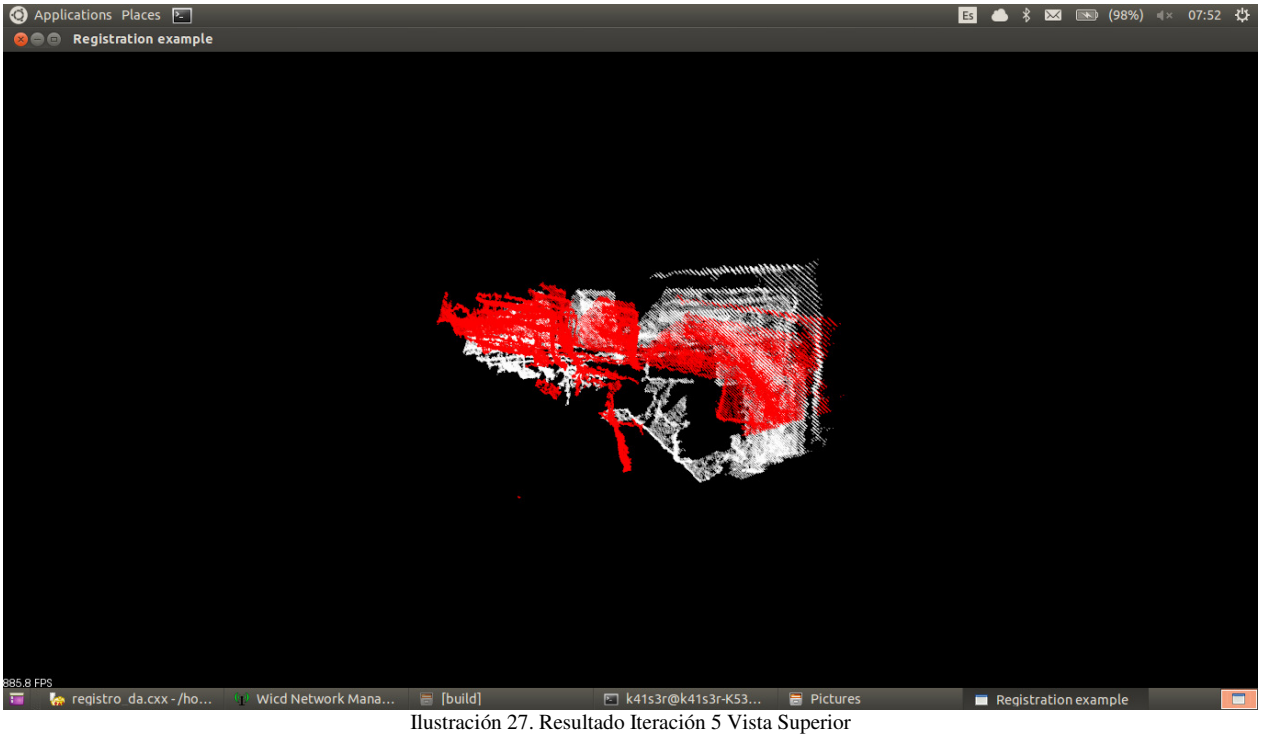


Ilustración 27. Resultado Iteración 5 Vista Superior

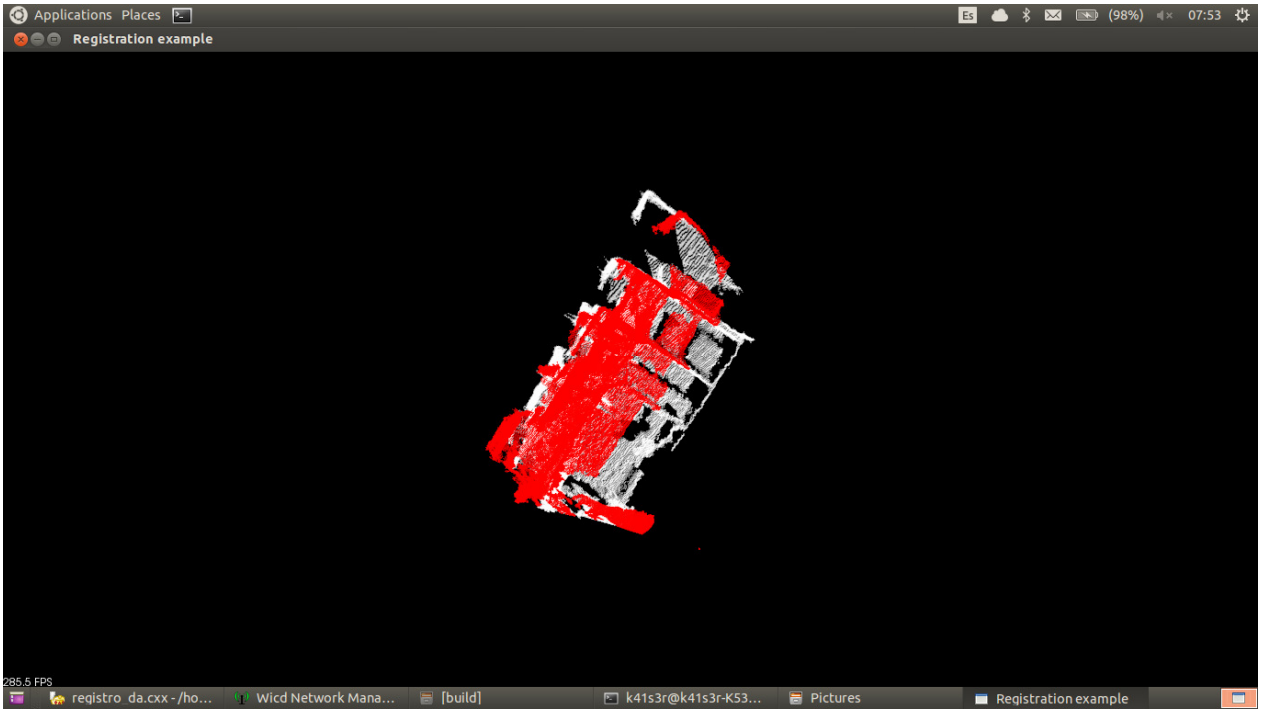


Ilustración 28. Resultado Iteración 5 Vista Lateral

La medida de calidad para esta iteración es 0,6449484 este valor es más cercano a cero que el de la iteración 2, y es el primero que se obtiene por debajo de 1, lo que deja al descubierto que este iteración es más cercana a la matriz referencial esperada, por lo que se puede decir que es una mejor reconstrucción hasta el momento.

4.6. RESULTADO PRUEBAS ITERACIÓN 6

Aunque los resultados obtenidos con NARF y usando *BadRejectCorrespondences* se acercaban a lo esperado, se decidió buscar una mejora cambiando la forma de obtención de los *keypoints* para poder calcularlos sobre toda la imagen y no sobre un rango particular. Con esto en mente se pasó a utilizar los SIFT, esperando que con mejores *keyPoints* las correspondencias obtenidas fueran mejores.

Al revisar la obtención de los *Keypoints* con SIFT, efectivamente logramos una mayor cantidad, como se observa en la tabla 16, lo que conlleva a encontrar un mayor número de correspondencias, comparativamente con las obtenidas en la iteración anterior, como está en la tabla 14.

	IMAGEN 1	IMAGEN 2
Tamaño Pointcloud	307200	307200
SIFT: Keypoints	125	166
Correspondencias	31	

Tabla 16. Resultado Prueba Iteración 6

Este aumento en el número de *keypoints* y correspondencias nos da como resultado una matriz de transformación, Tabla 17, comparativamente más cercana a la matriz referencial que estamos usando. Lo

anterior, con respecto a la matriz de transformación se obtiene una mejora en el resultado al tener una mayor cantidad de *keypoints*.

0.998493	0.0443428	0.0323553	-0.13923
-0.0446577	0.998961	0.0090782	0.0256487
-0.0319192	-0.0105093	0.999435	0.017441
0	0	0	1

Tabla 17. Matriz de transformación Iteración 6

Visualmente con la transformación de la matriz anterior observamos que las líneas horizontales se mantienen, como se observa en la Ilustración 29, las líneas verticales son paralelas entre sí por lo que la orientación de la imagen visualmente es correcta. En la ilustración 30 se puede observar la rotación sobre el eje Z que realiza la imagen de acuerdo a la matriz de transformación.

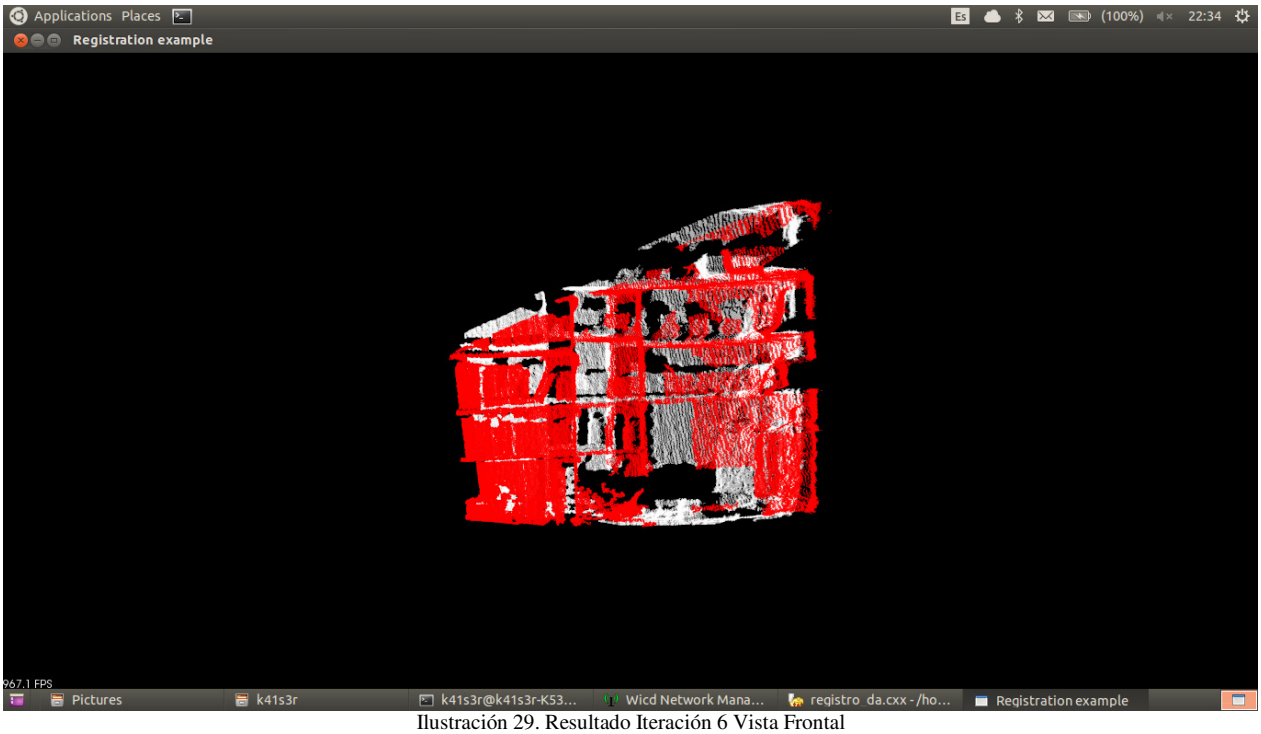


Ilustración 29. Resultado Iteración 6 Vista Frontal

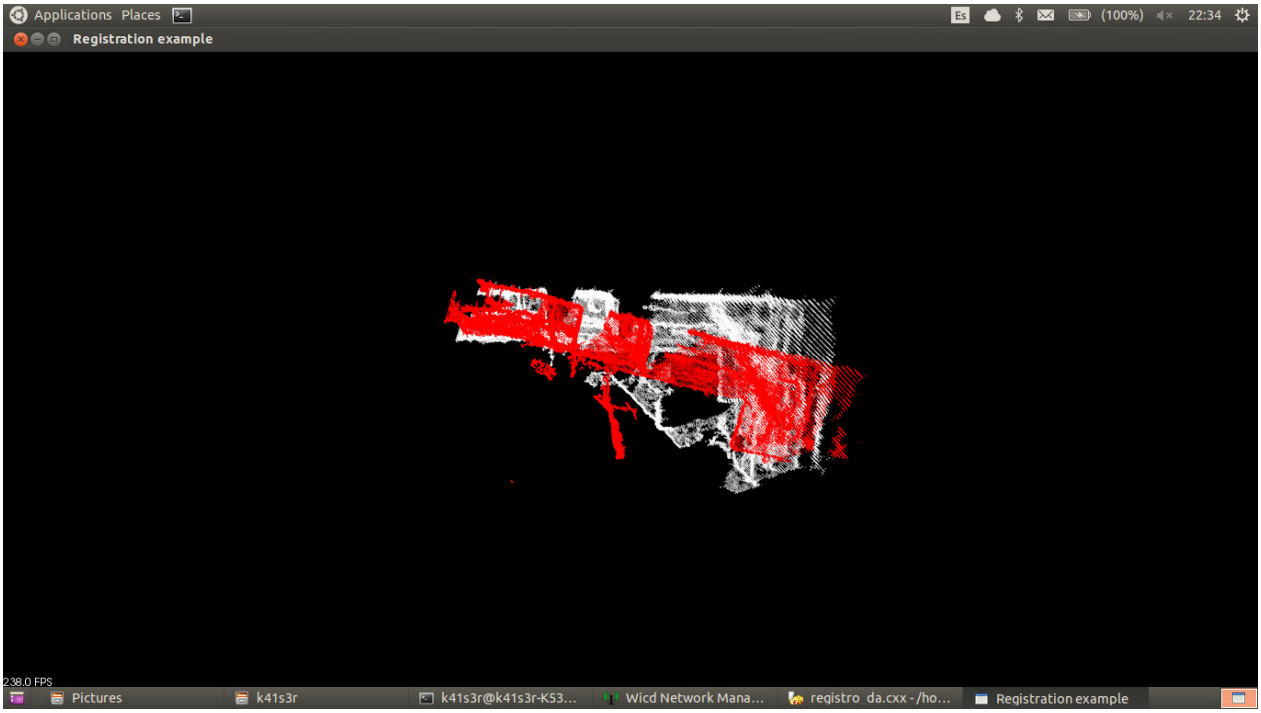


Ilustración 30. Resultado Iteración 6 Vista Superior

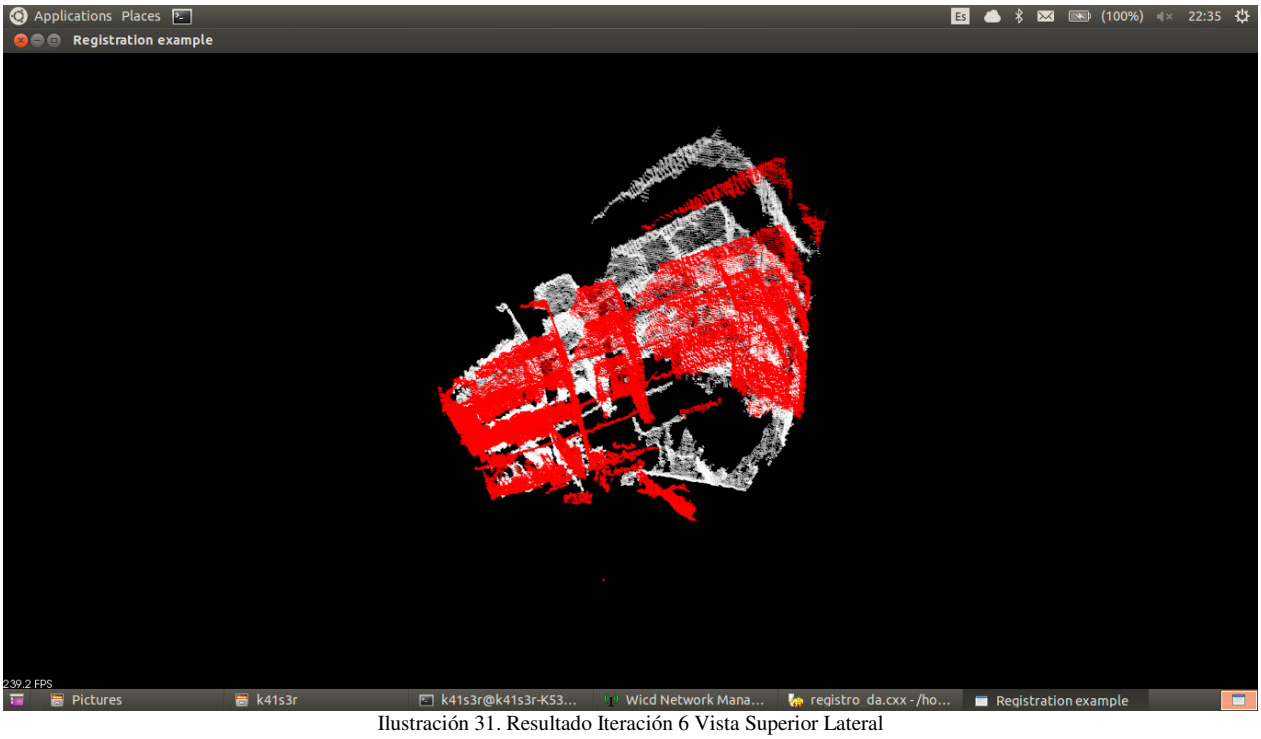


Ilustración 31. Resultado Iteración 6 Vista Superior Lateral

La medida de calidad para esta iteración es 0,53512720 este valor es más cercano a cero que el de la iteración anterior, por lo cual la iteración 6 da el valor más cercano a la matriz referencial esperada, y es la mejor reconstrucción obtenida.

4.7. RESULTADOS PRUEBAS MÚLTIPLES IMÁGENES

4.7.1. Resultados 3 imágenes

Con los resultados de la última iteración se obtuvo la mejor aproximación, por lo que se decidió usar los mismos parámetros de la iteración para realizar el emparejamiento con múltiples nubes de puntos.

Por cada imagen deben calcularse nuevamente tanto los *keypoints* como sus correspondencias y se obtiene la matriz de transformación. Se realiza el proceso de registro entre las dos primeras nubles de puntos, obteniendo los resultados observados en la tabla 18; obteniendo como resultado la matriz de transformación de la Tabla 19.

	IMAGEN 1	IMAGEN 2
Tamaño Pointcloud	307200	307200
SIFT Keypoints	125	146
Correspondencias	12	

Tabla 18. Resultado Prueba Múltiple 1.1

0.995014	-0.0167946	-0.0983175	0.0507374
0.0108464	0.998096	-0.0607271	0.0913935
0.09915	0.059358	0.993301	0.0671828
0	0	0	1

Tabla 19. Matriz de transformación Múltiple 1.1

Para realizar el registro con la nube de puntos 3, es necesario aplicar la transformación obtenida a la nube de puntos 2 y sobre esta nube transformada buscar las correspondencias.

En este proceso se obtuvieron los resultados de la tabla 20. En los dos procesos para obtener correspondencias se obtuvieron la misma cantidad de correspondencias, aunque el número de *keypoint* varía según la información contenida en cada imagen.

	IMAGEN 2 T	IMAGEN 3
Tamaño Pointcloud	307200	307200
SIFT Keypoints	162	166
Correspondencias	12	

Tabla 20. Resultado Prueba Múltiple 1.2

	-	-	
0.999504	0.0134819	0.0284603	0.024444
			-
0.0142629	0.999523	0.0274204	0.0721927
0.0280772	-	0.999219	-

	0.0278127		0.0701411
0	0	0	1

Tabla 21. Matriz transformación Múltiple 1.2

Una vez aplicadas las dos transformaciones se puede observar en la Ilustración 32 como se mantienen las líneas horizontales, como resultado de la rotación en Z, así como las líneas verticales paralelas en la transformación.

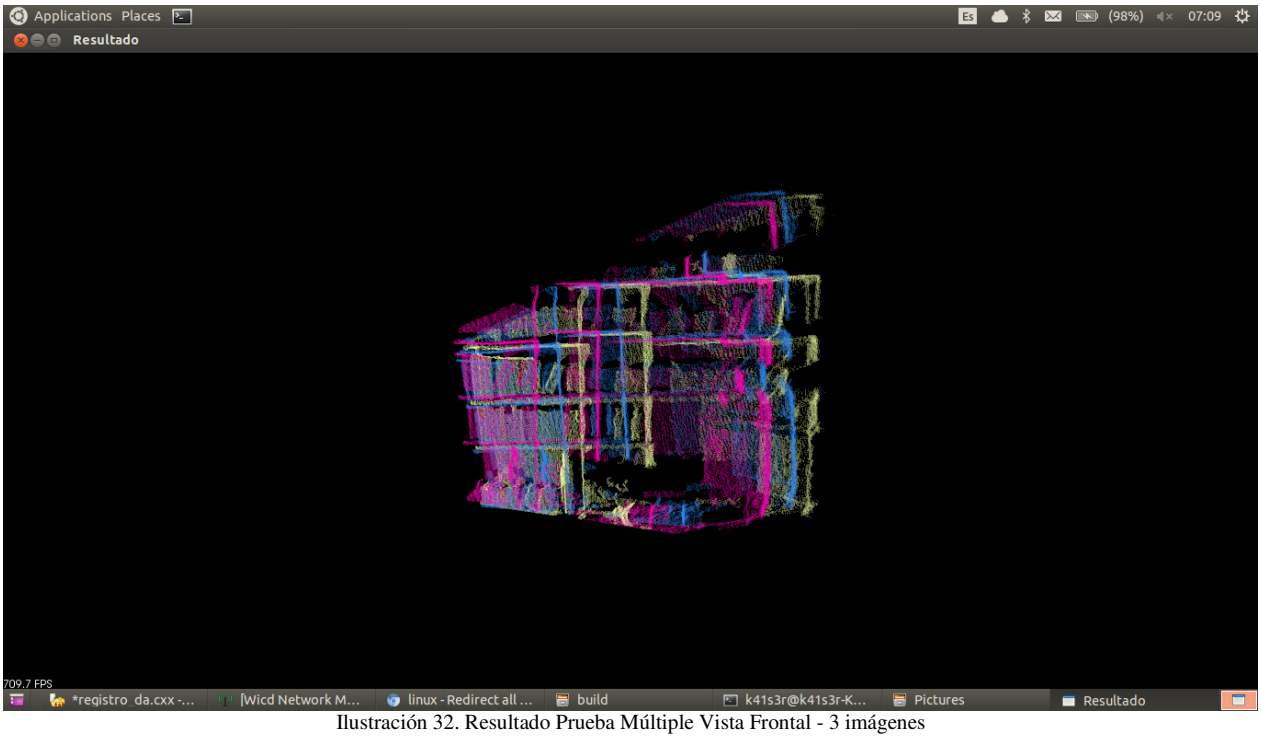


Ilustración 32. Resultado Prueba Múltiple Vista Frontal - 3 imágenes

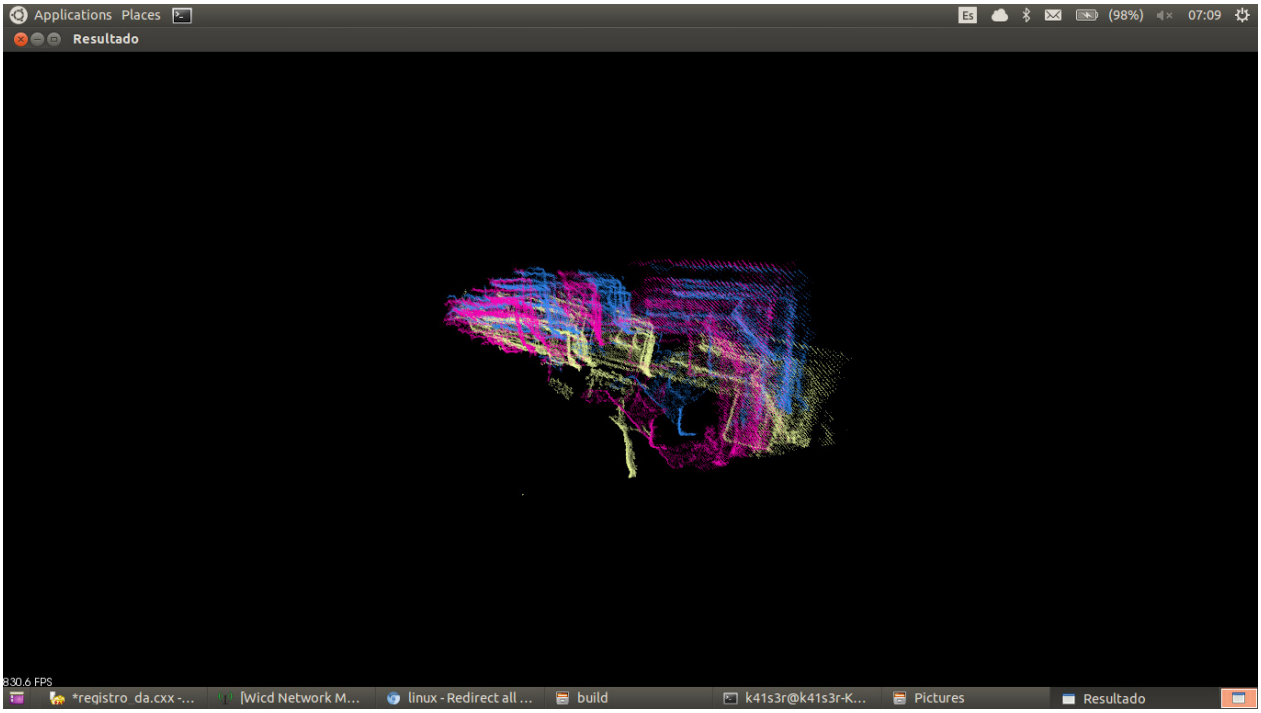


Ilustración 33. Resultado Prueba Múltiple Vista Superior - 3 Imágenes

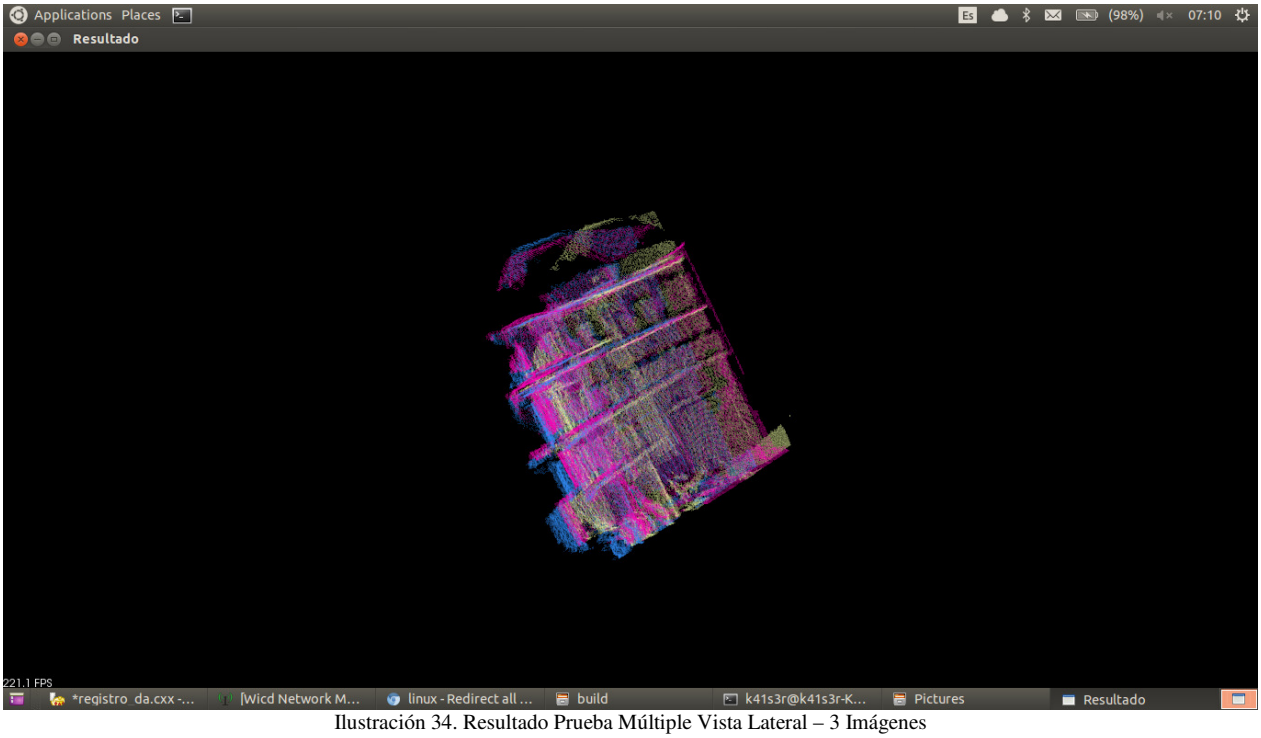


Ilustración 34. Resultado Prueba Múltiple Vista Lateral – 3 Imágenes

4.7.2. Resultados 10 imágenes

Finalmente, con el sistema completo se toma las imágenes con cada rotación del motor y se procesa de forma paralela. Los resultados se puede observar en las tablas siguientes.

	IMAGEN 1	IMAGEN 2
Tamaño Pointcloud	307200	307200
SIFT: Keypoints	125	146
Correspondencias	12	

Tabla 22. Resultado Prueba Múltiple 1 – 10 imágenes

0.995014	-0.0167946	-0.0983175	0.0507374
0.0108464	0.998096	-0.0607271	0.0913935
0.099150	0.059358	0.993301	0.0671828

IMAGEN 2 T	IMAGEN 3
307200	307200
146	166
12	

Tabla 23. Resultado Prueba Múltiple 2 – 10 imágenes

0.999504	-0.0134819	-0.0284603	0.024444
0.0142629	0.999523	0.0274204	-0.0721927
0.028077	-0.027812	0.999219	-0.070141

IMAGEN 3 T	IMAGEN 4
307200	307200
162	180
14	

Tabla 24. Resultado Prueba Múltiple 3 – 10 imágenes

0.989153	0.108443	0.0990801	-0.234765
-0.111581	0.993397	0.0266815	-0.0533119
-0.095532	-0.037447	0.994722	0.0347068

IMAGEN 4 T	IMAGEN 5
307200	307200
183	201
16	

Tabla 25. Resultado Prueba Múltiple 4 – 10 imágenes

0.993502	-0.0798198	-0.0811444	0.0959486
0.0720653	0.992925	-0.0943779	0.157186
0.088103	0.087916	0.992224	0.0106932

IMAGEN 5 T	IMAGEN 6
307200	307200
191	215
25	

0.998781	0.045	-0.0202771	-0.02154
-0.0445675	0.998779	0.0213026	-0.0146381
0.0212109	-0.020373	0.999567	0.0128626
0	0	0	1

Tabla 26. Resultado Prueba Múltiple 5 – 10 imágenes

	IMAGEN 6T	IMAGEN 7
Tamaño Pointcloud	307200	307200
SIFT: Keypoints	207	201
Correspondencias	22	

0.999649	0.00415185	-0.02619	-0.0177422
-0.00393339	0.999957	0.00838688	0.0113179
0.0262237	-0.00828075	0.999622	-0.027318
0	0	0	1

Tabla 27. Resultado Prueba Múltiple 6 – 10 imágenes

	IMAGEN 7 T	IMAGEN 8
Tamaño Pointcloud	307200	307200
SIFT: Keypoints	187	211
Correspondencias	14	

0.996965	-0.0685301	-0.0369391	-0.0057494
0.0659299	0.995537	-0.0675261	0.113333
0.0414018	0.0648858	0.997033	0.00518227
0	0	0	1

Tabla 28. Resultado Prueba Múltiple 7 – 10 imágenes

	IMAGEN 8 T	IMAGEN 9
Tamaño Pointcloud	307200	307200
SIFT: Keypoints	218	195
Correspondencias	12	

0.99957	0.000514584	-0.0293407	-0.01858
0.00319596	0.99199	0.126279	-0.24233
0.0291706	-0.126318	0.991561	-0.00758421
0	0	0	1

Tabla 29. Resultado Prueba Múltiple 8 – 10 imágenes

	IMAGEN 9 T	IMAGEN 10
Tamaño Pointcloud	307200	307200
SIFT: Keypoints	194	194
Correspondencias	24	

0.999884	0.0111623	-0.0103648	-0.0204217
-0.0126403	0.987728	-0.155672	0.314852
0.00849976	0.155785	0.987755	0.0576173
0	0	0	1

Tabla 30. Resultado Prueba Múltiple 9 – 10 imágenes

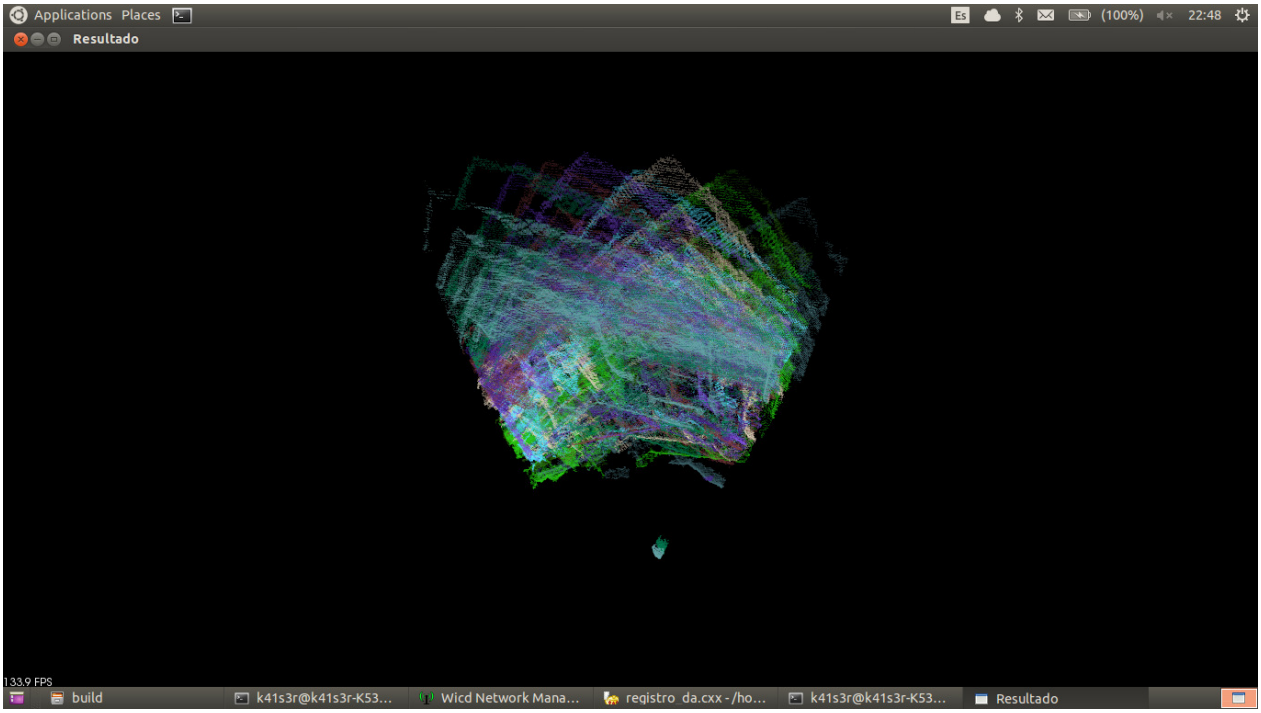


Ilustración 35. Prueba Múltiple Vista Superior – 10 imágenes

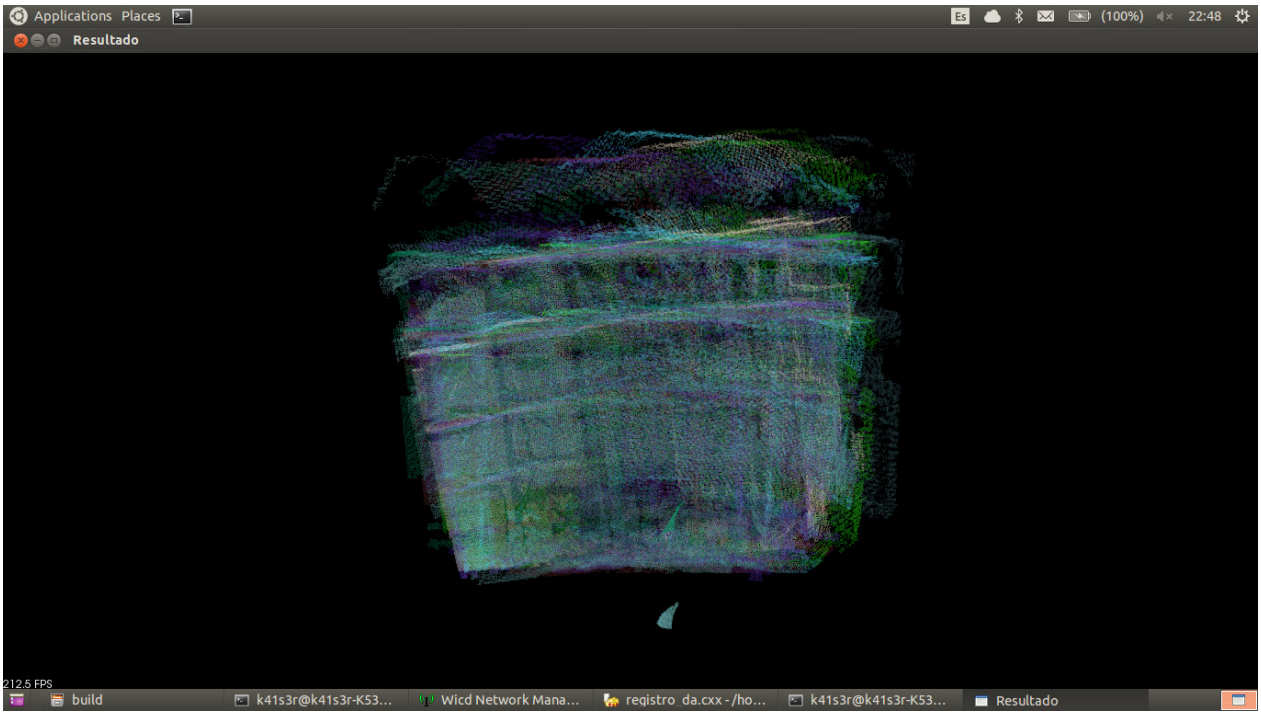


Ilustración 36. Prueba Múltiple Vista Frontal – 10 imágenes

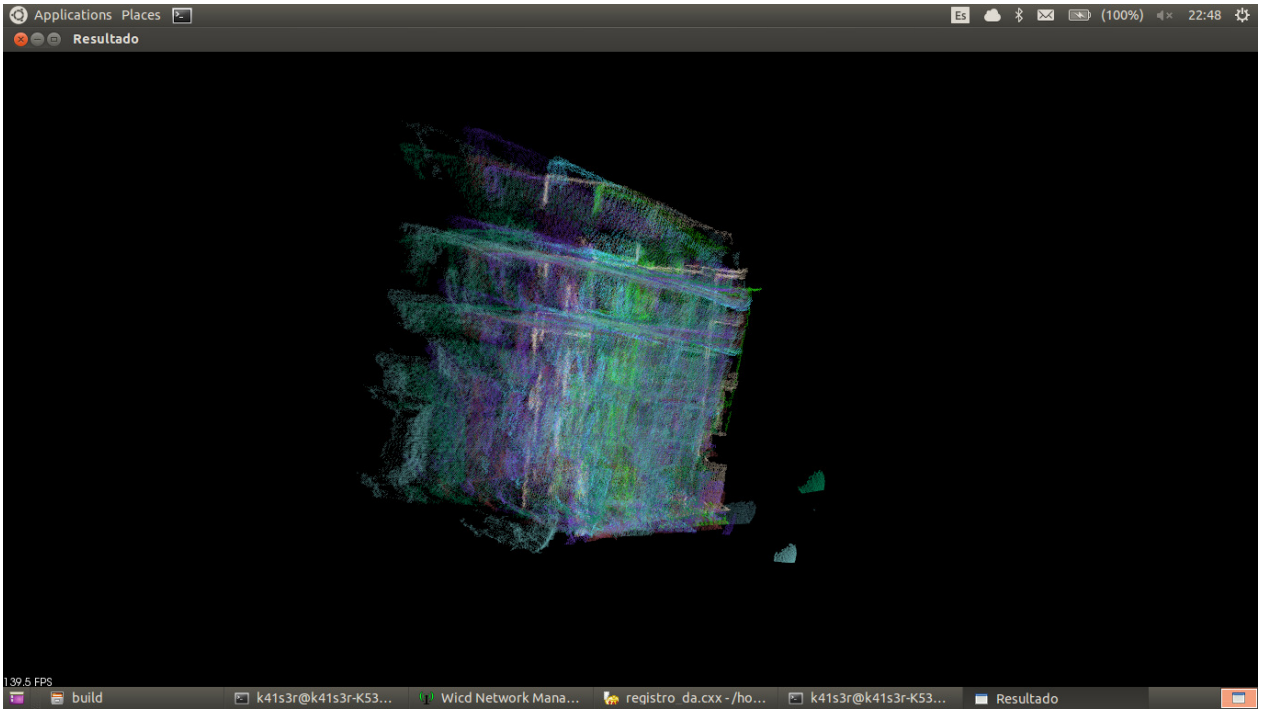


Ilustración 37. Prueba Múltiple Vista lateral 1 – 10 imágenes

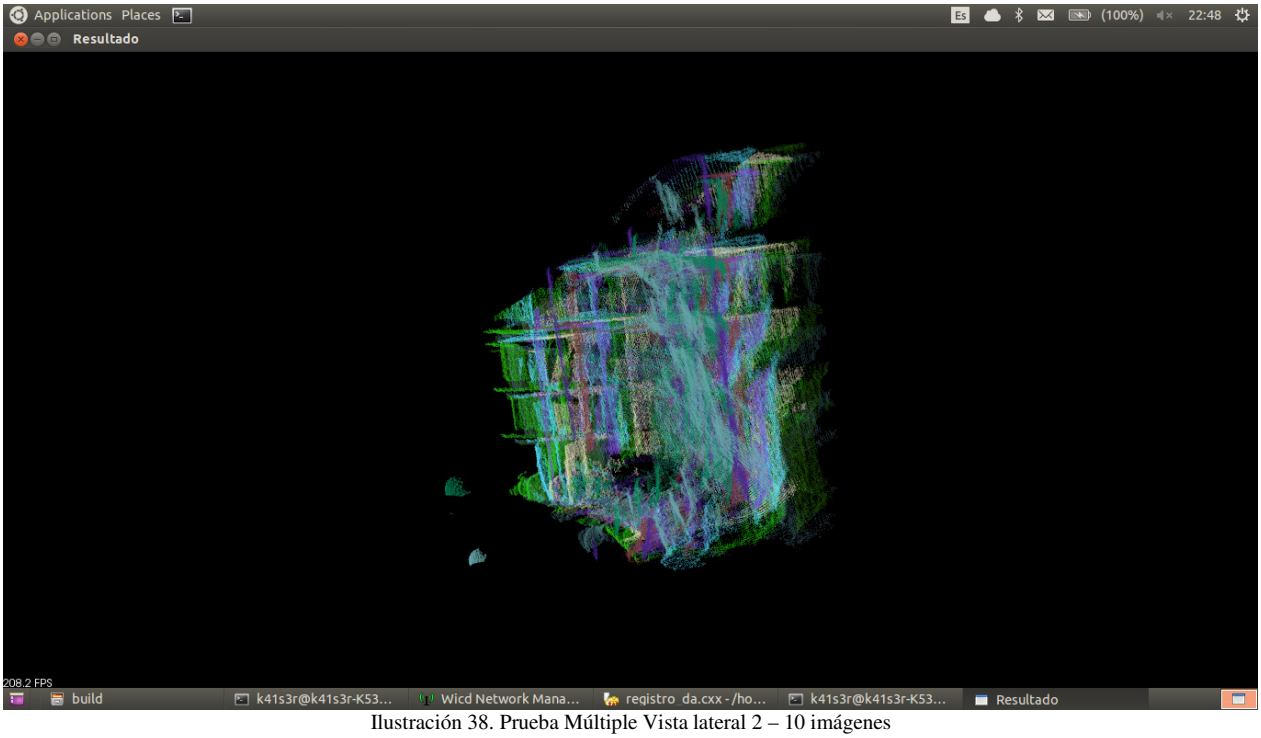


Ilustración 38. Prueba Múltiple Vista lateral 2 – 10 imágenes

En las ilustraciones 35, 36, 37 y 38 se puede observar el resultado final, donde las 10 imágenes obtenidas son procesadas y transformadas en búsqueda de una correspondencia entre ellas. Los resultados no son muy exactos pero tienen relación. Es importante notar que en cuanto a la orientación se consiguió un buen resultado, pero la transformación rígida limita hasta cierto punto que el Registro sea exacto. Corresponde a una aproximación a la solución final del problema.

Finalmente, para hacer más fácil la comparación cualitativa se muestra una imagen en la ilustración 39 de la biblioteca completa. Y a continuación una de las reconstrucciones, con solo dos imágenes, sobre la escena. Es importante notar que en la escena original existe un Angulo de 90 grados, que no se puede replicar en la imagen plana.

Ilustración 39. Imagen referencial biblioteca

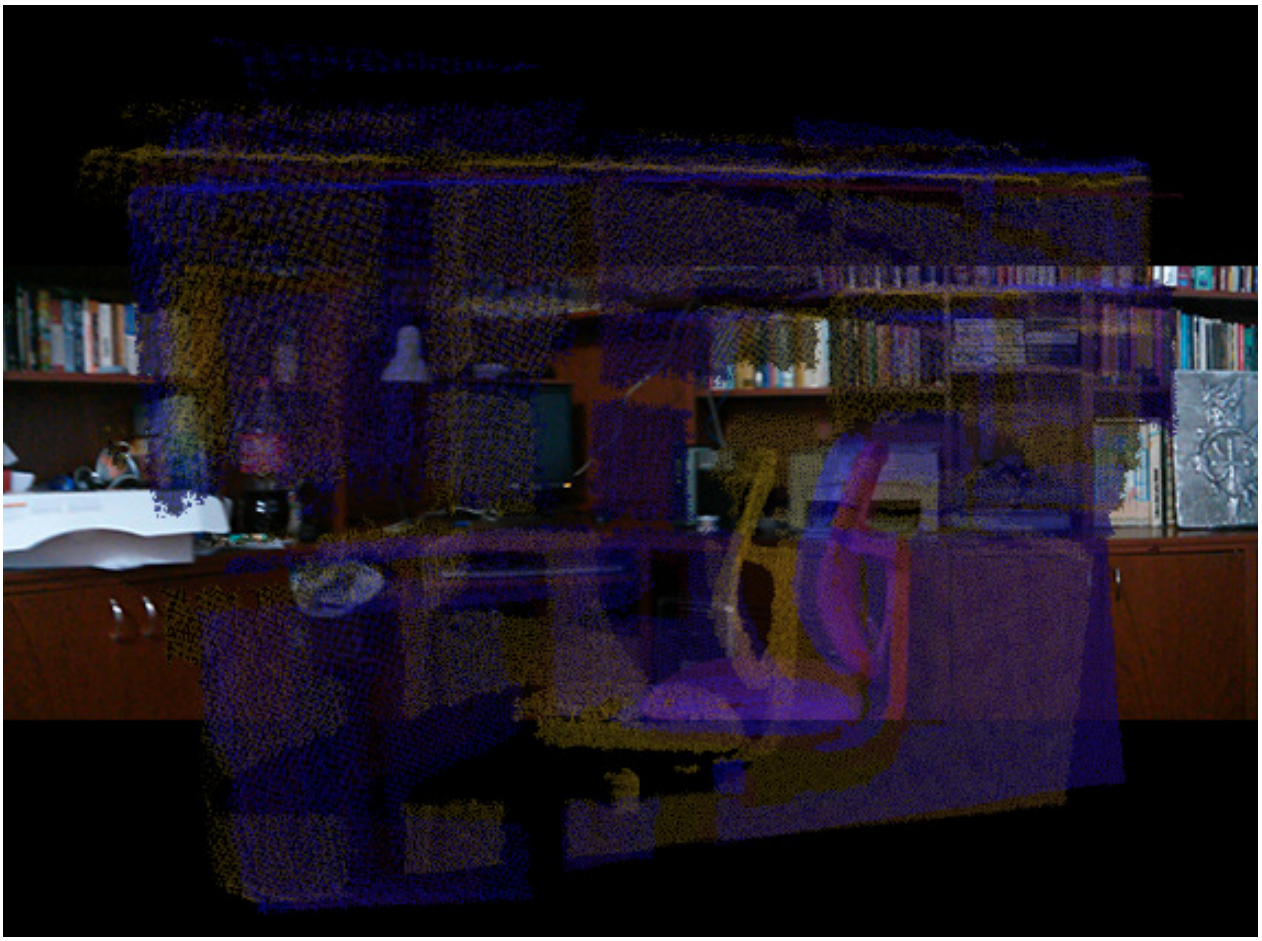


Ilustración 40. Imagen sobre puesta de la nube de puntos y la biblioteca

4.8. ESPACIOS INTERIORES TRABAJADOS

Como parte de las pruebas se realizaron pruebas sobre diferentes espacios interiores, con el objetivo de comparar resultados sobre estos. Sobre algunas características especiales de los espacios interiores se presentaron problemas para realizar las comparaciones.

Estas dificultades encontradas en cierto tipo de espacios interiores están expuestas en la siguiente lista:

- Ventanas: Cuando las ventanas no tienen cortinas que estén cerradas, el vidrio dificulta la toma de la nube de puntas. Si sobre la ventana hay una entrada de sol directo esto puede distorsionar la nube por lo que los resultados no son muy precisos.
- Paredes vacías: al realizar la prueba sobre paredes que no tuviera ninguna clase de adorno, si no que por el contrario fueran completamente homogéneas, los keypoints obtenidos no eran determinantes por lo que pocos datos sobre la pared permiten el manejo y esto puede producir puntos falsos con los que la correspondencia no es correcta.
- Puertas abiertas: En los casos donde la habitación tenga puertas abiertas los puntos en esta área de la puerta se pierden por profundidad, por lo que queda un vacío que disminuye la nube de puntos. Sin embargo, toma el marco de la puerta como referente lo que minimiza el error.
- Espacios grandes: Debido a las limitaciones propias del Kinect el rango sobre el cual se puede trabajar para encontrar puntos, no permite que los espacios muy amplios tengan resultados, ya que la nube de puntos no se puede determinar. Esta limitante propia del Kinect no permite que este esté ubicado más allá de los 3,5 metros, por lo que si se centra en la habitación el Kinect el tamaño máximo del espacio sobre el que se puede trabajar es de 7x7 aproximadamente.

5. CONCLUSIONES

Se formuló una estrategia para modelado de espacios interiores cerrados, que permite incluir un componente electrónico y un componente de procesamiento en software. En este sistema se tiene una plataforma sobre la cual reposa un Kinect que gira de forma precisa gracias a la escogencia de motores de alta precisión y potencia, este ángulo de rotación puede ser modificado según necesidad, sin embargo, para que las pruebas fueran consistentes se escogió un ángulo permanente para el manejo de las toma de imágenes. Sobre este punto es importante notar que en ángulos muy pequeños de rotación, por la disparidad propia del Kinect los resultados entre imágenes no alcanzan una correspondencia, ya que en algunos casos no se detecta la rotación.

En la unión de la plataforma con el sistema de procesamiento de las imágenes, no se tuvo inconvenientes, ya que envía y recibe a través del driver la inicialización de la rotación, de esta forma se controla desde el software la rotación y momento de la rotación. Como limitante debido a que los motores escogidos tienen de forma manual la graduación del ángulo, esto no pude ser incluido dentro del desarrollo como una variable. Sobre este punto se deja la puerta abierta hacia futuro de la manipulación del ángulo desde el programa que permite nuevas conclusiones y sea más versátil a la hora de tomar las pruebas.

Para el análisis de los espacios interiores, el desarrollo fue principalmente investigativo, sobre las limitantes propias de las tecnologías usadas, En cuanto a las pruebas estas se limitaron a los espacios interiores a los que se tenía acceso. Por lo cual cierta clase de espacios o superficies de los espacios quedaron por fuera de las pruebas. En este sentido se recomienda a futuro realizar pruebas sobre espacios más amplios, que pueda tener dificultades propias como colocación de espejos, fuentes de luz diferentes, esquinas en ángulos diferentes a 90°, entre otras.

En cuanto al algoritmo, de la parte investigativa se obtuvo un panorama amplio, que permitió retomar lo planteado dentro del Registro por la misma librería de PCL y explorar diferentes combinaciones que permitiera lograr un mejor resultado. Sobre este punto es importante mencionar que el diseño en sí mismo es el desarrollo, y que las pruebas realizadas permitieron ajustar este desarrollo, para encontrar un procedimiento, algoritmo, paso a paso que diera los mejores resultados. Los resultados finales no consiguen una reproducción exacta del espacio sobre el que se está probando, pero consigue una buena aproximación en cuanto a la orientación y dimensión de las imágenes. Debido al uso de transformaciones rígidas, la transformación encontrada se aplica sobre toda la nube de puntos lo que produce una distorsión en los resultados. Se recomendaría explorar otros tipos de transformación suave para buscar mejores resultados.

La plataforma solo permite la rotación del Kinect, más no el desplazamiento, que quedó fuera del alcance del presente proyecto. La inclusión de esta variable, así como de movimiento sobre el eje Z, puede estar dentro del alcance de proyectos a futuros que permitan entender el funcionamiento con estas nuevas variables para la toma y procesamiento de las imágenes.

Finalmente no se cumplió el objetivo de generar modelos y realizar la comparación de los mismos, debido a que los resultados obtenidos no se corresponden con una geometría en la habitación ya que las imágenes a nivel de rotación realizan la misma rotación de la cámara quedando en ángulos diferentes, la comparación con un modelo no daba información nueva. Por lo que no se incluyó en este informe.

6. BIBLIOGRAFÍA

1. GU, CHunhui. REN, Xiao. “Discriminative Mixture of Templates of viewpoint Classification”. ECCV 2010. Volume 6315, Publisher: Springer, pp 408-421. ISBN 978642155543
2. LEE, H. Grosse, R. RANGANATH, R. “Convolutional deep belief networks for scalable unsupervised learning of hierechical representations”. ICML 2009.
3. COLLET, Romea. BERENSON,D. Srinivasa, S. Ferguson, d. “Object recognition and full pose registration from a single image for robotic manipulation,” in Proc. of the IEEE International Conference on Robotics & Automation (ICRA), 2009.
4. DALAL, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR. (2005)
5. REN, Xiaofeng. “Figure-Ground Segmentation Improves Handled Object recognition”. IEEE Conferences on Computer Vision and Pattern Recognition 2008. Issue1, pp 1-8.
6. FELZENSZWALB, P. McAllester, D. Ramanan D. “A discriminatively trained, multiscale, deformable part model”. CVPR, 2008.
7. BO, Liefeng. REN, Xiaofeng. FOX, Dieter. “Kernel Descriptors for Visual Recognition”. Advances in Neural Information Processing Systems 23 (2010).
8. KRAININ, Michael. Henry, Peter. Ren, Xiaofeng. Fox, Dieter. “Manipulator and object tracking for in hand model acquisition”. Science 2010.
9. HABBECKE, M., KOBBELT, L., “A surface-growing approach to multiview stereo reconstruction,” in CVPR, 2007.
10. LOWE, D. Distinctive image features from scale-invariant keypoints. IJCV, 60:91–110, 2004.
11. BO, Liefeng, SMINCHISESCU, C. Efficient Match Kernel between Sets of Features for Visual Recognition. In NIPS, 2009.

12. WEISE, T., WISMER, T., LEIBE, B., GOOL, V., “In-hand scanning with online loop closure,” in IEEE International Workshop on 3-D Digital Imaging and Modeling, October 2009.
13. RAMANAN, D., BAKER, S.. Local Distance Functions: A Taxonomy, New Algorithms, and an Evaluation. In Proc. of the International Conference on Computer Vision (ICCV), 2009.
14. LAI, Kevin. BO, Leifeng. REN, Xiaofeng. FOX, Dieter. “Sparse Distance learning for Object Recognition Combining RGB and Depth Information”. 2010
15. KOENDERINK, J., VAN DOORN, A.: The internal representation of solid shape with respect to vision. Biological Cybernetics 32 (1979) 211–6
16. HENRY, Peter. KRAININ, Michael. HERBST, Evan. REN, Xiaofeng. FOX, Dieter. “RGB-D Mapping: Using Depth cameras for dense 3D Modeling of indoor environments”. Work 2010.
17. FURUKAWA,Y., CURLESS, B., SEINTZ, S., SZELISKI, R.. Reconstructing building interiors
18. http://www.arduino.cc/en/Main/arduinoBoardUno [online] Arduino UNO. Copyright © 2014 Arduino
19. http://pointclouds.org/ [online] Point Cloud Library. Creative Commons Attribution 3.0.
20. RUSU, Radu Bogdan, BLODOW, Nico, BEETZ, Michael. “Fast Point Feature Histograms (FPFH) for 3D Registration”
21. FITZGIBBON, Andrew. “Robust Registration of 2D and 3D Point Sets”, University of Oxford.
22. RUSU, Radu Bogdan, COUSINS, Steve. “3D is here: Point Cloud Library (PCL)”. Willow Garage

TABLA DE ANEXOS

ANEXO A CÓDIGO FUENTE FINAL DE LA SOLUCIÓN
ANEXO B CÓDIGO FUENTE PRUEBAS POR ITERACIONES
ANEXO C CÓDIGO FUENTE ARDUINO