Reconstrucción 3D de Imagenes RGB-D

Alexis Mendoza, Crhistian Turpo y Diego Amable

I. INTRODUCCIÓN

La reconstrucción 3D de alta calidad es uno de los objetivos clave de la computación gráfica. La capacidad de generar estos modelos de una manera sencilla y detalla en entornos físicos pueden ayudar a numerosos campos como la industria de los videojuegos, los efectos especiales utilizados en varias áreas, y proporcionar datos valiosos para los sistemas de visión artificial. Con este fin se crean cámaras especiales como el *Kinect*, que nos proporcionan información adicional como el *depth* que es un dato importante para la reconstrucción 3D.

En la actualidad para el proceso de reconstrucción 3D existen varios métodos tales como: *Stereo*, *Motion*, *Shading*, *Photometric Stereo*, *Texture*, *Contours*, *Silhouettes*. Esta tiene varios campos donde se puede aplicar, como: reconocimiento de objetos, robótica, computación gráfica, recuperación de imágenes, geolocalización, arqueología, deportes, entre otros.

En el presente trabajo se implementa un método para la reconstrucción densa usando imágenes RGBD obtenidas con *kinect*. Se genera una superficie globalmente consistente y detallado de una escena a un *stream* continuo.

II. MARCO TEÓRICO

II-A. Kinect de Microsoft

El sensor Kinect es una barra horizontal conectada a una base pequeña con un pivote motorizado y está diseñada para posicionarse longitudinalmente por encima o por debajo de la pantalla de video. El dispositivo cuenta con una cámara RGB, un sensor de profundidad y un micrófono multi-array con software patentado, que proporciona captura de movimiento 3D de cuerpo completo, reconocimiento facial y capacidades de reconocimiento de voz.

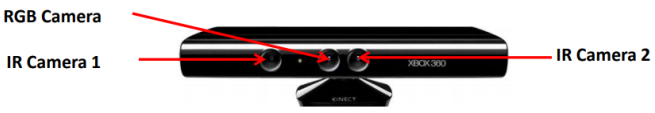

Fig. 1. Kinect y sus cámaras.

II-B. Calibración de cámara

Las cámaras estenopeicas baratas de hoy introducen mucha distorsión en las imágenes. Dos distorsiones principales son la distorsión radial y la distorsión tangencial. Debido a la distorsión radial, las líneas rectas aparecerán curvas. Su efecto es más a medida que nos alejamos del centro de la imagen. La distorsión tangencial se produce porque la lente que toma la imagen no está alineada perfectamente paralela al plano de imagen. Así que algunas áreas de la imagen pueden verse más cerca de lo esperado. Para corregir esto se tienen que hallar los coeficientes de distorsión. En adición a estas tenemos que trabajar con los parámetros intrínsecos y extrínsecos de una cámara.

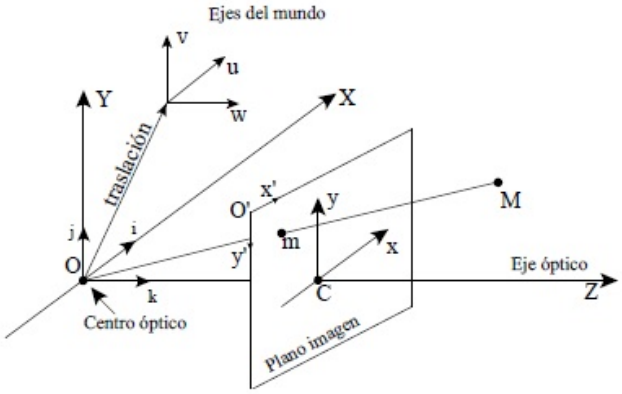

Fig. 2. Sistema Odométrico

La matriz de la cámara la usamos para ayudarnos a sacar información geométrica de las imágenes, se define:

$$\begin{pmatrix} fx & 0 & cx \\ 0 & fy & fy \\ 0 & 0 & 0 \end{pmatrix}$$

Donde (fx, fy) es la distancia focal y $(cx.cy)$ son los centros ópticos.

II-C. Geometría Epipolar

Cuando tomamos una imagen con una sola cámara se pierde información, como la profundidad de la imagen o que tan lejos esta cada punto de la imagen a la cámara, ya que se realiza una transformación 3D a 2D. Para poder encontrar la información de profundidad se utilizan dos cámaras y recibe el nombre de visión estéreo.

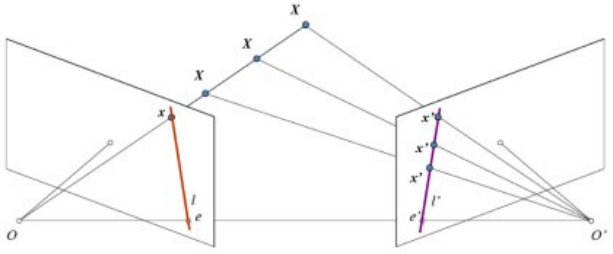

Fig. 3. Proyección de puntos 3D usando 2 cámaras.

Si estamos usando solo la cámara izquierda, no podemos encontrar el punto 3D correspondiente al punto x en la imagen porque cada punto en la línea OX proyecta en el mismo punto en el plano de la imagen. Pero considera la imagen correcta también. Ahora, diferentes puntos en la línea OX proyectan a diferentes puntos x' en el plano derecho. Así que, con estas dos imágenes, podemos triangular el punto 3D correcto.

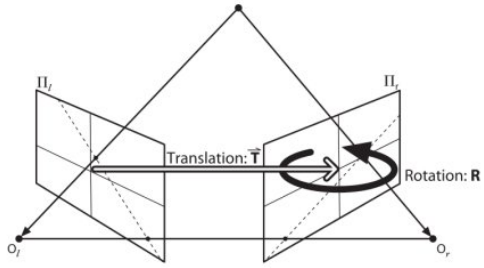

Fig. 4. Matrices de rotación y translación.

La Matriz Fundamental contiene información acerca de la rotación, traslación e información sobre los aspectos intrínsecos de ambas cámaras para que podamos relacionar las dos cámaras en coordenadas de píxeles. La Matriz Fundamental F, mapea un punto en una imagen a una línea en la otra imagen. Esto se calcula a partir de puntos coincidentes de ambas imágenes. Se requiere un mínimo de 8 puntos para encontrar la matriz fundamental, para esto se pueden el algoritmo de 8 puntos.

II-C.1. ORB: es básicamente una fusión del detector de puntos clave FAST y el descriptor BRIEF con muchas modificaciones para mejorar el rendimiento. Primero use FAST para encontrar puntos clave, luego aplique la medida de la esquina de Harris para encontrar los mejores N puntos entre ellos. También usa pirámide para producir características multiescala.

II-D. Iterative Closet Point (ICP)

ICP es un algoritmo empleado para minimizar la diferencia entre dos nubes de puntos. El ICP se usa a menudo para reconstruir superficies 2D o 3D a partir de diferentes escaneos, para localizar robots y lograr una planificación óptima de la trayectoria, para registrar de forma conjunta modelos de huesos, etc.

Dados dos nubes de puntos.

$$P = p1, ..., pn$$

$$Q = q1, ..., qn$$

Se quiere hallar la matriz de transformación que minimize el error:

$$E(x, y) = \frac{1}{n}\sum_{i=1}^{n}\|p_i - R * q_i - t\|^2 \quad (1)$$

Donde p_i y q_i son puntos correspondientes.

Para las correspondencias conocidas se calcula el centro de la masa con un promedio por cada nube de puntos.

$$\bar{p} = \frac{1}{n}\sum_{i} p_i \quad (2)$$

$$\bar{q} = \frac{1}{n}\sum_{i} q_i \quad (3)$$

Se resta el correspondiente centro de masa desde cada punto. Luego recuperamos la rotación.
Descomponemos la matriz usando SVD.

$$W = \sum_{i=1}(p_i - \bar{p})^T * (q_i - \bar{q})^T = USV^T \quad (4)$$

Si W tiene rango 3, la solución óptima de $E(R, t)$ es única y dada por:

$$W = UV^T \quad (5)$$

$$t = \bar{p} - R\bar{q} \quad (6)$$

II-E. Octree

Un Octree es una estructura de datos en la cual cada nodo interno tiene exactamente 8 nodos hijos. Las estructuras Octree se usan mayormente para particionar un espacio tridimensional, dividiéndolo recursivamente en ocho octantes. Las estructuras Octree son las análogas tridimensionales de los Quadtree bidimensionales.

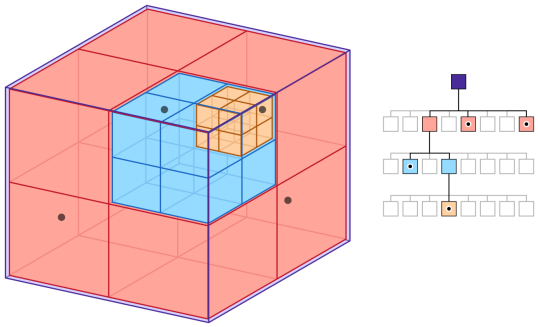

Fig. 5. Estructura Octree.

III. PROCESO

Para un mayor entendimiento siga la figura 6. La reconstrucción da inicio con la carga de la base de datos que consta de imágenes RGBD. Por cada *frame* se crea una nube de puntos. El Octree es iniciado, en él se almacenará la posición del píxel, los colores y las normales.

Se toma un *subsampling* para no trabajar con todos los *frames* de la imagen. Para el primer *frame* se crea la nube de puntos,se calcula la matriz de transformación y es almacenada en el Octree. Para los siguientes *frames* el primer *frame* sera la fuente y el segundo el objetivo. Una vez tengamos definidos estos *frames* se prosigue a calcular la Odometría. Para la reconstrucción 3D trabajamos con 2 imágenes y definimos un *match* como la presencia de un punto característico en ambas imágenes.

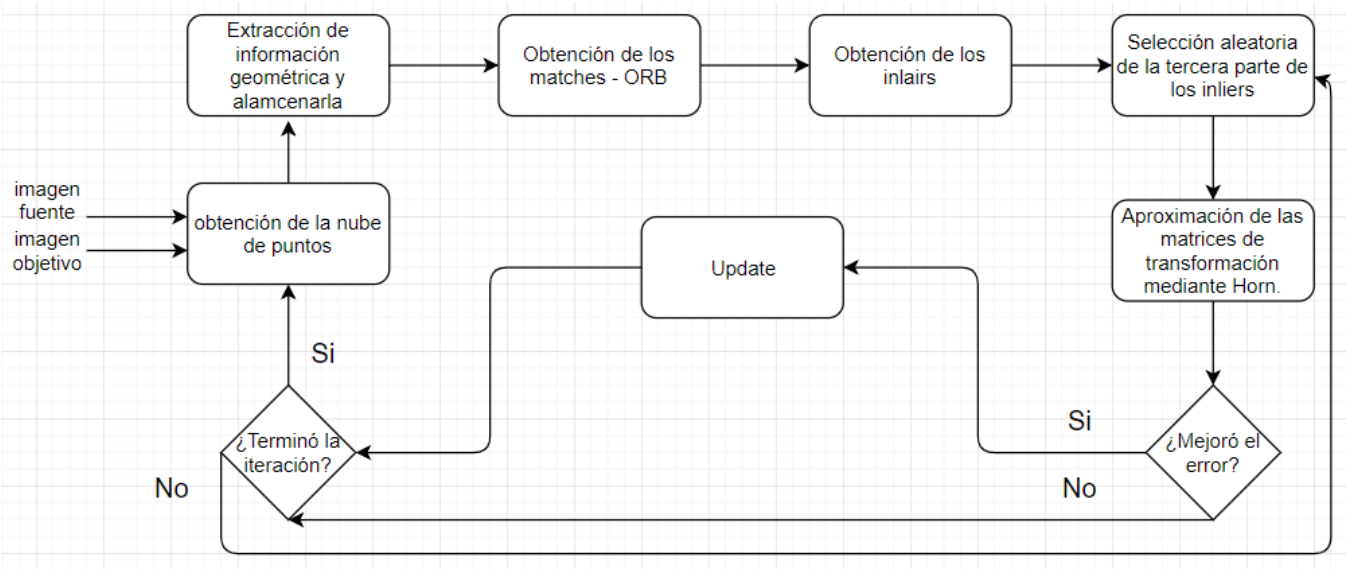

Fig. 6. Diagrama de proceso de reconstrucción 3D implementado en este trabajo.

Se calculan los *matches* entre la fuente y el objetivo, con el algoritmo ORB. De los *matches* encontrados no todos son buenos, entonces se calcula la distancia entre los matches y los que no pasen un *treshold* no serán considerados. Los *matches* que si pasan el *treshold* son llamados *inliers* y los que no pasan son llamados *outliers*.

Una vez obtenidos los *inliers*, estos son mezclados aleatoriamente y se escoge la tercera parte. Con los *inliers* escogidos usamos el algoritmo Horn (ICP) para obtener la matriz de rotación y translación. Luego de obtener la matriz de rotación y translación se procede a calcular la matriz de transformación. Esta matriz de transformación es calculada en cada iteración, donde se busca minimizar la diferencia entre la nube de puntos objetivo y la nube de puntos fuente. Para ello se calcula un error con una función distancia y la matriz de transformación se actualiza si el error es menor. La mejor transformación encontrada para un par de *frames* es almacenada en un vector en donde se irán multiplicando cada una de las matrices. La última matriz de dicho vector es utilizada para transformar la nube de puntos del *frame* actual (alineándola), cuyos puntos son almacenados en el Octree.

Una vez que todos los puntos alineados se encuentran en nuestra estructura, se procede a la recuperación de los puntos y a la triangulación para la creación de la malla. Para recuperar los puntos, se obtienen los *bounding boxes* (cajas) de cierto nivel de la estructura. Preferiblemente se selecciona el último nivel, siendo el que tiene mayor información de los puntos. Una vez obtenidas las cajas, se utiliza el centro de cada una de ellas como los vértices que formarán la malla.

Para la formación de la malla se hace uso de la librería *Point Cloud Library* (PCL) para C++. Dicha librería se basa en [6] para el cálculo de la triangulación. Los datos de la nube de puntos alineada son almacenados en un archivo PCD, y la malla generada es guardada en un archivo VTK para su posterior visualización.

IV. EXPERIMENTOS

IV-A. Base de Datos

Para las pruebas se utilizó la base de datos de Computer Vision Group [7]. Dicha base de datos proporciona *frames* RGB-D de videos de duración variable. Para las pruebas, se han considerado alrededor de los 500 primeros *frames* para ejecutar la reconstrucción 3D.

IV-B. Detalles Técnicos

Para las pruebas, usamos una tarjeta NVidia GeForce GTX 1060 con 6 GBs de memoria, Una CPU Intel Core i7-7700HQ con 2.80 GHz de velocidad de reloj y una memoria RAM de 16GB.

IV-C. Resultados

(Ver Figura 7).

V. CONCLUSIONES

El *Kinect* es una herramienta muy útil y poderosa que nos ayuda en la reconstrucción 3D con la reducción del calculo del *depth*. También nos proporciona información de los valores intrínsecos y extrínsecos que son usados para la calibración.

Nuestra implementación logra una superficie dado un *stream* continuo (en este caso, un vídeo). El algoritmo implementado puede ser mejorado en términos de uso de memoria y construcción de la superficie.

El Octree puede generar una mejor malla si se le aplican otros algoritmos (marching cubes), a pesar de ello como se muestra en los resultados, la malla generada es bastante buena.

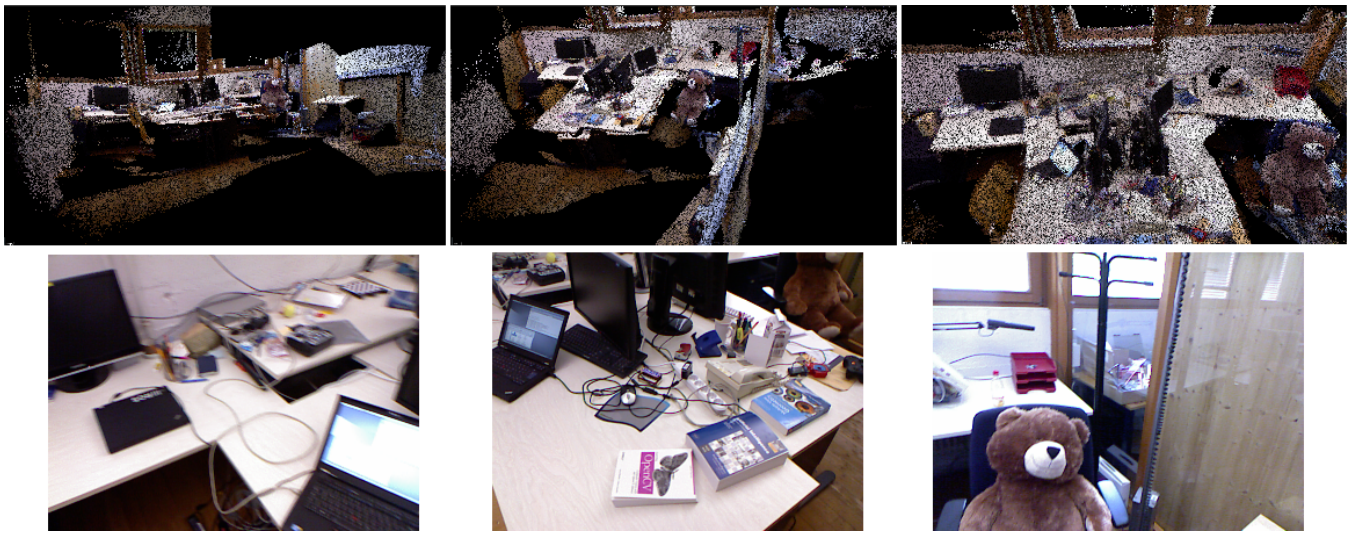

Fig. 7. Las imágenes inferiores son algunas de los frames RGB utilizados para la reconstrucción. Las superiores, son algunas perspectivas del modelo 3D obtenido.

AGRADECIMIENTOS

Agradecemos en especial a Max W. Portocarrero, estudiante del Programa de Maestría en Ciencia de la Computación de la Universidad Católica de San Pablo, por sus comentarios y guía en la implementación del presente proyecto, y al Concytec y su ente ejecutor Fondecyt por financiar dicho programa.

REFERENCES

[1] Qian-Yi Zhou, Vladlen Koltun .Dense Scene Reconstruction with Points of Interest, Journal ACM Transactions on Graphics, Volume 32 Issue 4, July 2013 Article No. 112 .

[2] S. Fuhrmann and M. Goesele, Fusion of Depth Maps with Multiple Scales. 2011

[3] Dr Mubarak Shah, UCF Computer Vision Video Lectures 2012

[4] JurgenSturm, Visual Navigation for Flying Robotos, Computer Vision Group, Technical University of Munich, Summer 2013

[5] OpenCV ORB (Orientation FAST and Rotated BRIEF)

[6] Zoltan Csaba Marton, Radu Bogdan Rusu and Michael Beet. On Fast Surface Reconstruction Methods for Large and Noisy Datasets, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2009.

[7] J. Sturm, N. Engelhard, F. Endres, W. Burgard and D. Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems, Proc. of the International Conference on Intelligent Robot Systems (IROS), 2012.