Reconocimiento de objetos tridimensionales usando el sensor de profundidad del Kinect

José L. Pérez
Universidad de Ingeniería y Tecnología (UTEC)
Jr. Medrano Silva 165, Barranco
Lima, Perú
{jose.perez}@utec.edu.pe

***Resumen*—En este trabajo se presenta el modelamiento de un programa que, a través del sensor de profundidad del Kinect, reconoce objetos tridimensionales. El objetivo es lograr que el programa presente una Nube de Puntos (Point Cloud), los cuales estén representados en un sistema de coordenadas tridimensional (x, y, z). Para esto, utilizamos el resultado que consigue el Kinect, una imagen 2D, de la cual se conoce la profundidad de todos sus puntos. Con esta información se puede llevar a cabo la reconstrucción virtual en 3D.**

I. Introducción

Para que los robots puedan trabajar en un entorno real necesitan, de una manera u otra, poder percibir el mundo. Es por eso, que, en los últimos años, se ha innovado en lo que respecta a los sensores de alcance, profundidad y ultrasonido, los cuales, a medida del tiempo, han ido proporcionando cada vez más bytes de información del mundo.

Muy recientemente, en el año 2010, apareció el Kinect como un accesorio para Consola Xbox 360. Es un dispositivo desarrollado por PrimeSense, empresa en colaboración con Microsoft [1]. Desde su presentación, Kinect se convirtió en un dispositivo ampliamente utilizado en tecnología e investigación. El Kinect contiene una cámara RGB, sensor de profundidad, fuente de luz IR, tres ejes de acelerómetro y micrófono, así como también un hardware de soporte que permite a la unidad emitir la información a un dispositivo externo. Nosotros, esencialmente, nos apoyaremos en la cámara RGB y en el sensor de profundidad para obtener la nube de puntos de diferentes objetos tridimensionales.

En nuestro caso, trabajaremos con PCL (Point Cloud Library), el cual nos sirve, esencialmente, como medio para el modelamiento de obtención de diferentes datos a través de las series de bibliotecas modulares que tiene. Este proyecto se enfoca, específicamente, en el reconocimiento y visualización de diferentes objetos tridimensionales.

Este trabajo se dividirá de la siguiente manera: en la Sección II presentaremos la metodología. La Sección III se enfocará en los resultados y las discusiones. Finalmente, presentaremos las conclusiones en la Sección IV.

II. METODOLOGÍA

A. HARDWARE

1. KINECT

Kinect es un sensor desarrollado por Primesense en colaboración con Microsoft. Originalmente conocido como parte de la consola XBOX 360, después, en junio del 2011, Microsoft decide lanzar un software de código abierto (Software development Kit – SDK) para que el sensor pueda ser utilizado con fines tecnológicos.

El Kinect cuenta con un emisor infrarrojo IR y un sensor de profundidad IR. Tiene, también, una cámara RGB con una resolución de 614x512 píxeles. Esto hace que la captura de una imagen a color sea posible. Así mismo, tiene un micrófono de múltiples matrices y un acelerómetro de tres ejes configurado en gamma 2G [2].

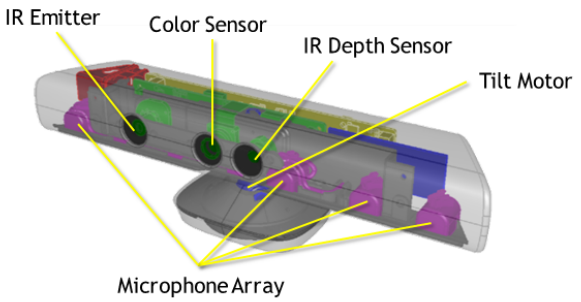


Fig 1. Partes del sensor Kinect.

A continuación, se explicará los fundamentos del sensor de para el procesamiento del reconocimiento de objetos tridimensionales:

Depth stream: Cada cuadro de la cantidad de datos almacenados en la imagen se compone de píxeles que contienen la distancia entre el objeto más cercano y el lente.

Infrared stream: El sensor de profundidad genera una luz infrarroja invisible, la cual nos sirve para determinar la distancia de un objeto con respecto al sensor. Infrared stream no es realmente un flujo de datos independiente, sino, más bien, una configuración particular de la corriente de la cámara.

Coordinate Space: El sensor de profundidad captura una imagen en escala de grises. Cada pixel contiene una distancia en milímetros que está representada en un eje cartesiano x,y.

Depth Space: Una operación común al obtener una imagen de profundidad es generar una nube de puntos (Point Cloud) en 3D. Es decir, tener un espacio proyectado desde el lente de la cámara hasta la profundidad del objeto. Para eso, se realiza un mapeo de coordenadas global de todos los píxeles de la imagen usando funciones basadas en matrices.

B. SOFTWARE

1. LIBFREENECT2

Libfreenect2 es un controlador multi-plataforma de código abierto para dispositivos Kinect v2. Esta documentación está diseñada principalmente para extraer imágenes de profundidad y color [3]. Los modos de video disponibles a través de este controlador son:

- Color: RGB888 1920x1080 30fps.
- Color: RGB888 960x540 30fps.
- Profundidad: 512x424 30fps.
- IR: 512x424 30fps.

El código del controlador está dirigido por diferentes lenguajes de programación. Entre estos encontramos Python, C++, Java y otros. Para este trabajo, se ha elegido en utilizar el lenguaje C++

2. POINT CLOUD LIBRARY(PCL)

PCL es una biblioteca de códigos para el procesamiento de nube de puntos en 2D y 3D. Contiene distintos algoritmos que tienen como función filtrar, reconstruir, registrar, visualizar y hasta modelar y segmentar las imágenes de salida. PCL ha sido compilado y desplegado en Linux, MacOS, Windows y Android [4]. Algunas de sus librerías:

- *Filters:* Contiene mecanismos atípicos y de eliminación de ruido para aplicaciones de filtrado de datos de nube de puntos 3D.
- *Features:* Contiene estructura de datos y los mecanismos para estimar la función 3D a partir de nubes de puntos.
- *Io:* Contiene clases y funciones para la escritura, lectura y captura de los datos de nube de puntos (PCD).
- *Segmentation:* Contiene algoritmos para la segmentación de una nube de puntos en grupos distintos.
- *Registration:* La combinación de varios conjuntos de datos se realiza generalmente mediante una técnica denominada registration.
- *Keypoints:* Contiene implementaciones de dos puntos algoritmos de detección de nubes de puntos significativos.
- *Visualization:* Con el fin de ser capaz de crear prototipos y visualizar los resultados de los algoritmos que operan en los datos de nubes de puntos en 3D.

3. OPEN GRAPHICS LIBRARY(OPENGL)

Es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D Y 3D [5]. La interfaz consiste en 250 funciones diferentes, las cuales sirven para redibujar imágenes tridimensionales a partir de figuras geométricas.

III. RESULTADOS Y DISCUSIONES

Para este trabajo se emplea la versión de Ubuntu 14.04 LTS debido a sus características, las cuales ayudan a procesar el trabajo de forma más sencilla, puesto que contiene el software libre para el funcionamiento del Kinect, puesto que cuenta con por defecto con la herramienta Cmake, la cual ayuda a automatizar el código. Así mismo, se utilizó lenguaje C++ para modelar el programa. Las siguientes librerías han sido utilizadas para el modelamiento del código:

<iostream>: Contiene diferentes funciones de entrada y salida;
<cstdlib>: Define varias funciones de propósito general, incluyendo la gestión dinámica de memoria, generación de números aleatorios, aritmética, clasificación y otros;
<signal.h>: Especifica como un programa maneja señales mientras se ejecuta. Una señal puede reportar un comportamiento excepcional del programa o un evento asíncrono fuera de ella.

En tanto, el proyecto se desarrolló en las dos siguientes partes:

C. Modelamiento del programa

El modelado del programa se ha desarrollado en base al código abierto del libfreenect2 y PCL. Se ha utilizado diferentes librerías, de las cuales las pertenecientes al libfreenect2 son de reconocimiento de objetos y la librería PCL es determinante para la visualización tridimensional de estos. La librería <libfreenect2/registration.h> permite trabajar sólo con la profundidad sin tener que procesar la imagen de color y poder registrarla para calcular cualquier punto 3D asociada con un valor de píxel [6]. El cálculo de la nube de puntos 3D se puede modelar de la siguiente manera:

```
for (unsigned int x=0; x<width; ++x)
{
		for (unsigned int y=0; y<height; ++y)
		{
			value = frame_data[y*width + x];
			//std::cout<<"value:"<< value << std::endl;
			//pcl::PointXYZ point(x,y,value);
			pcl::PointXYZRGBA point2;
			point2.x=x;
			point2.y=y;
			point2.z=value;
			point2.rgba=UINT32_MAX;

			cloud->points[y*width + x]=point2;
		}
}

//pcl::PointXYZ point(14,12,15);
//cloud->points[0]=point;
viewer.showCloud(cloud);
```

Fig 2. Código para calcular la nube de puntos 3D.

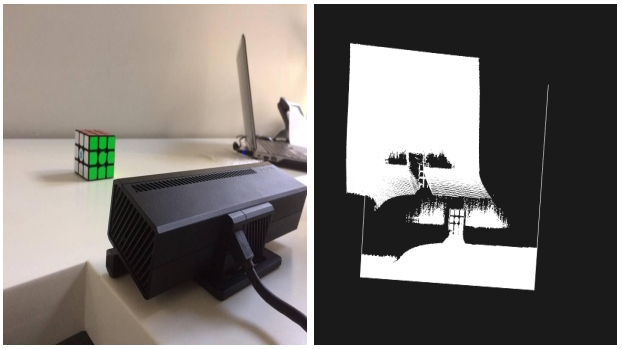
La librería <pcl/visualizatioon/cloud_viewer.h> ayuda a visualizar los algoritmos que operan en los datos de nubes de puntos 3D en forma rápida [1]. Se puede tener la visualización de la nube de puntos con pocas líneas de código usando un fragmento como este:

```
#include <pcl/visualization/cloud_viewer.h>
//...
void
foo ()
{
  pcl::PointCloud<pcl::PointXYZRGB>::Ptr cloud;
  //... populate cloud
  pcl::visualization::CloudViewer viewer ("Simple Cloud Viewer");
  viewer.showCloud (cloud);
  while (!viewer.wasStopped ())
  {
  }
}
```

Fig 3. Código para obtener la visualización de la nube de puntos.

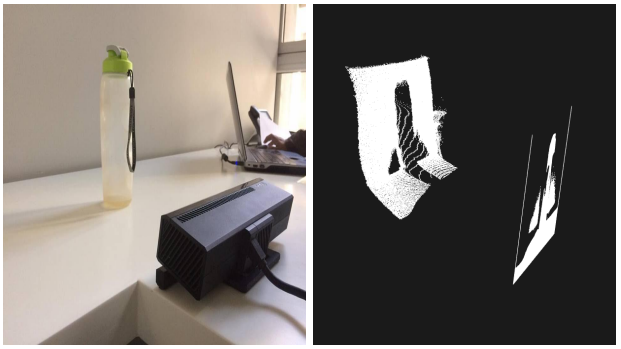
D. *Obtención de la nube de puntos por medio del sensor de Kinect*

El Kinect es un dispositivo que puede trabajar con diferentes opciones de controlador, esto dependiendo de la complejidad del proyecto. En este caso, se trabajó con libfreenect2 por su compatibilidad con Cmake y lenguaje de programación C++. Por medio de CloudViewer, aplicación de PCL, se obtuvo la nube de puntos en un campo de coordenadas tridimensional. Se obtuvo la nube de puntos de diferentes objetos como se ve a continuación:



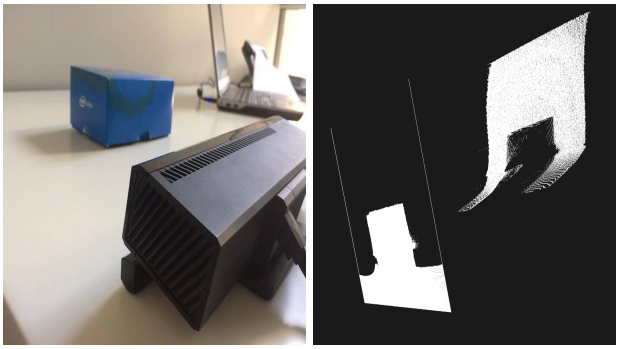
(1) (2)

Fig. 4. Reconocimiento del Objeto N°01



(3) (4)

Fig. 4. Reconocimiento del Objeto N°02



(3) (4)

Fig. 5. Reconocimiento del Objeto N°03

IV. CONCLUSIONES

Kinect es un dispositivo que es útil es áreas como la robótica, el proceso de imágenes, realidad virtual y aumentada, la reconstrucción 3D y el reconocimiento de objetos. Este proyecto se enfocó, primordialmente, en esta última área. Existen muchas plataformas de código abierto como OpenNI, libfreenect, SDK, los cuales hacen más fácil la evolución de muchas aplicaciones a través de este dispositivo.

El modelado empleado tuvo la capacidad de reconocer los objetos, calcular su nube de puntos y representarlos en un eje de coordenadas tridimensional. Así mismo, se representa, gráficamente, la profundidad y la distancia entre el objeto y el lente visor.

El reconocimiento y la visualización de nube de puntos es la base para la inclusión de otras funciones. Es por eso, que como trabajo futuro se propone introducir los demás módulos de la librería PCL, tales como Segmentation, Filters, Keypoints y demás.

Referencias

[1] R. Bogdan, S. Cousins, 3D is here: Point Cloud Library (PCL). USA, Willow Garage.

[2] Microsoft (2016). *Kinect for Windows Sensor Components and Specifications*. Recuperado el 18 de Noviembre del 2016 de: https://msdn.microsoft.com

[3] Openkinect (2016). *Libfreenect2*. Recuperado el 18 de Noviembre del 2016 de: https://openkinect.github.io/libfreenect2/

[4] OpenNI (s.f.). *PCL*. Recuperado el 19 de Noviembre del 2016 de: http://pointclouds.org/

[5] OpenGL (2016). *About OpenGL*. Recuperado el 19 de Noviembre del 2016 de: https://www.opengl.org/

[6] Openkinect (2016). *Libfreenect2. Recuperado el 19 de Noviembre del 2016 de:* https://openkinect.github.io/libfreenect2/classlibfreenect2_1_1Registration.html