Coloquio de Investigación Multidisciplinaria

VOLÚMEN 7 Núm. 1 OCTUBRE 2019

Revista Periódica

latindex

JOURNAL CIM - REVISTA DIGITAL

ISSN: 2007 - 8102

DIFUSIÓN VÍA RED DE CÓMPUTO
https://www.cim-tecnm.com/articulos

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

 ISSN 2007-8102

Algoritmo de reconstrucción 3D implementado en Raspberry-Pi 3

E. A. Cuellar Cortés, Y. A. Vásquez Beltrán, J. R. Hernández Gutiérrez, J. Posada Gómez
Tecnológico Nacional de México – Instituto Tecnológico de Orizaba, Oriente 9 No. 852, C.P. 94320, Orizaba Veracruz, México
acuellarc@orizaba.tecnm.mx, yael.a.vasquez.b@outlook.com
***Área de participación**: Ingeniería Mecánica y Mecatrónica*

Resumen

Se presenta en este trabajo la implementación de un algoritmo en el lenguaje Python orientado a la reconstrucción tridimensional de entornos y objetos utilizando dos imágenes capturadas por medio de dos cámaras idénticas con eje óptico paralelo y haciendo uso de la tarjeta Raspberry-PI 3. Se explica el principio de funcionamiento que se utilizó en este trabajo, así como también las características esenciales que se deben de cumplir para obtener resultados válidos. Se describe cada una de las partes del código programado, explicando de manera precisa los parámetros modificables que ayudan a obtener una reconstrucción adecuada. Se muestra la estructura empleada para la colocación de las cámaras y, finalmente, se presentan los resultados obtenidos en las diferentes pruebas, especificando las características aplicadas para la generación de los mapas de disparidad. De esta manera, se demuestra que mediante dos cámaras idénticas de cualquier tipo y una tarjeta Raspberry-PI 3 se puede lograr una reconstrucción 3D.

***Palabras clave**: Cámaras idénticas, reconstrucción tridimensional, mapas de disparidad.*

Abstract

The implementation of an algorithm in the Python language aimed at the three-dimensional reconstruction of environments and objects using two images captured by means of two identical cameras with parallel optical axis and using a Raspberry-PI 3 is presented in this paper. The operating principle used in this work is explained, as well as the essential characteristics that must be met to obtain valid results. Each part of the programmed code is described, explaining in a precise way the modifiable parameters that help to obtain an adequate reconstruction. The structure used for the placement of the cameras is shown and, finally, the results obtained in the different tests are presented, specifying the characteristics applied for the generation of the disparity maps. In this way, it is demonstrated that by means of two identical cameras of any type and a Raspberry-PI 3 a 3D reconstruction can be achieved.

***Key words**: Identical cameras, three-dimensional reconstruction, disparity maps.*

Introducción

La reconstrucción 3D es el proceso mediante el cual objetos reales son reproducidos en la memoria de una computadora, manteniendo sus características físicas (dimensiones, volumen y forma) [1]. Existen dos tipos de métodos utilizados para la reconstrucción tridimensional. Los métodos activos, es decir, los métodos de datos de rango, dado el mapa de profundidad, reconstruyen el perfil 3D por aproximación numérica y construyen el objeto en un escenario basado en el modelo. Los métodos pasivos no interfieren con el objeto reconstruido, solamente utilizan un sensor para medir la luminosidad reflejada o emitida por la superficie del objeto para inferir su estructura 3D a través de la comprensión de la imagen. Normalmente, el sensor es un sensor de imagen en una cámara sensible a la luz visible y la entrada al método es un conjunto de imágenes digitales (una, dos o más) o video. En este caso, hablamos de reconstrucción basada en imágenes y la salida es un modelo 3D. En comparación con los métodos activos, los métodos pasivos pueden aplicarse a una gama más amplia de situaciones [2].

La visión estéreo es un método pasivo que extrae la información 3D a partir de dos imágenes digitales. En la figura 1 se puede observar el funcionamiento del método de visión estéreo binocular Esta técnica requiere de dos cámaras idénticas con eje óptico paralelo para observar un mismo objeto, adquiriendo dos imágenes desde diferentes puntos de vista. En términos de relaciones de trigonometría, la información de profundidad se puede

rec 06/ago/2019, acc. 13/sep /2019

 ISSN 2007-8102

calcular a partir de la disparidad. El método de visión estéreo binocular está bien desarrollado y contribuye de manera estable a la reconstrucción 3D favorable, lo que lleva a un mejor rendimiento en comparación con otras técnicas de reconstrucción 3D [3].

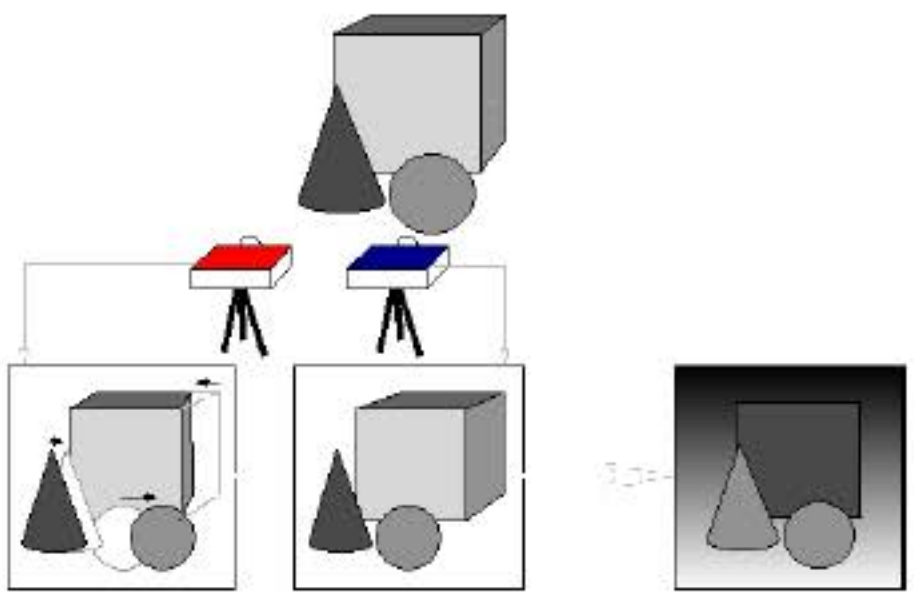

Figura 1. Funcionamiento del método de visión estéreo binocular

El desarrollo para la obtención de una reconstrucción tridimensional consta de diferentes etapas. Éstas son la manufactura de la base, la calibración de las cámaras, la adquisición de las imágenes, la extracción de información, la correspondencia estéreo y, finalmente, la restauración 3D del entorno. Las características de cada una son de gran importancia puesto que impactan en la calidad del resultado final que se obtendrá en la representación tridimensional de los objetos. Para visualizar las reconstrucciones que se generan en la visión estéreo existen diferentes softwares que permiten leer las nubes de puntos producidas en el algoritmo, por ejemplo, MeshLab [4-6]. En la figura 2 se puede observar una reconstrucción en el software de MeshLab. Sin embargo, requerir de un programa externo resulta inconveniente para los usuarios puesto que requiere de espacio de almacenamiento adicional y obtención de resultados parciales.

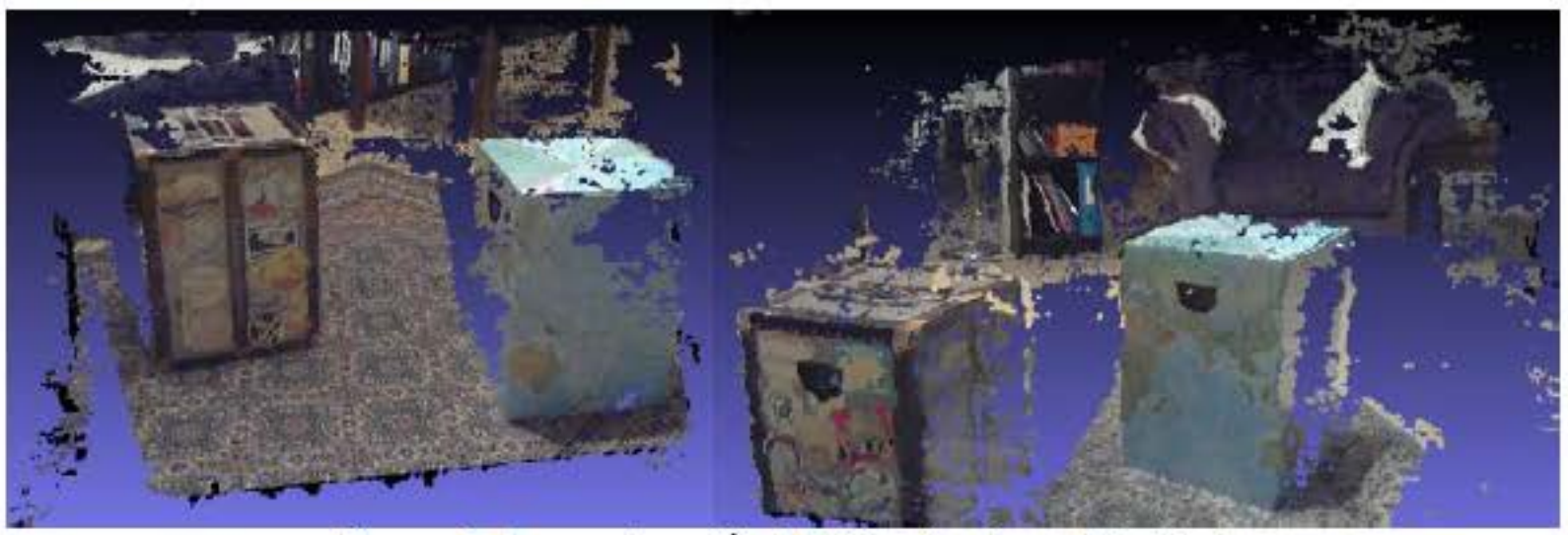

Figura 2. Reconstrucción 3D visualizada en MeshLab

Se decidió emplear una tarjeta Raspberry-PI 3, que se puede observar en la figura 3, la cual es una placa computadora (SBC) de bajo coste, para el procesamiento de las imágenes capturadas y la recreación del entorno tridimensional. El concepto de esta tarjeta es el de un ordenador al cual le eliminan todos los accesorios que no afecten al funcionamiento básico. Está formada por una placa que soporta varios componentes necesarios en un ordenador común y es capaz de comportarse como tal.

La principal limitación del microordenador era la conectividad inalámbrica. Por suerte, con la llegada del Raspberry Pi 3 se logró resolver esta característica. Esta versión del microordenador mantiene 1 GB de memoria RAM, pero utiliza una revisión del procesador al BCM2837 ARMv8, de 4 núcleos, pero que funciona a 1.2 Ghz. Esta versión es la primera en utilizar instrucciones de 64 bits. Además, la mayor novedad de esta versión es que finalmente incluyó Wi-Fi y Bluetooth 4.1 de base, sin necesidad de adaptadores, ampliando los horizontes de su usabilidad mucho más allá. [7]

 ISSN 2007-8102

Figura 3. Raspberry-PI 3

En el presente trabajo se empleó el método de visión estéreo binocular por medio de dos cámaras web para computadora idénticas, utilizando el lenguaje Python para diseñar un algoritmo que produzca la reconstrucción tridimensional. Se hizo uso de las librerías Open Source con las que dispone Python enfocadas a la creación de mapas de disparidad y obtención de las nubes de puntos que permiten visualizar los objetos en 3D. Entre algunas de las librerías necesarias para realizar esta tarea, las más importantes son PyntCloud, Open3D, Pillow y Pandas. La facilidad que permite este algoritmo para la obtención de una reconstrucción 3D es una de las grandes ventajas que ofrece, además del poco espacio que emplean las librerías necesarias y los archivos generados por el programa, haciendo posible su uso en tarjetas de programación Raspberry-Pi 3 o similares.

Metodología

Material, equipo y software

Para el diseño de este prototipo fue necesaria una Raspberry-Pi 3 de 1.4Ghz a 64bits con un procesador Quad-Core, banda dual de red y Bluetooth 4.2. Otra característica de la tarjeta es el puerto CSI diseñado para conectar la RaspiCam, sin embargo, para este proyecto fue necesario incluir dos cámaras que capturen dos frames al instante, debido a esto se decidió utilizar dos cámaras tipo USB, con las cuales ya se contaban puesto que la tarjeta posee 4 puertos USB que no estaban siendo utilizados. Estas cámaras tienen las siguientes características: marca Gear Head, modelo WC755SLV, resolución de 1.3 MP y una resolución máxima de 1280x1024. También se utilizó una base de madera y un tubo de acero, por otro lado, se requirió de una micro SD de 16 GB para almacenar el sistema operativo de la tarjeta que incluye las librerías de Python 3.6, PyntCloud, OpenCV, Numpy, Matplotlib, Pandas, OS, Open3D y Pillow.

Manufactura de la base

Se diseñó una base de madera que permitiera el desplazamiento horizontal de las cámaras para la realización de las pruebas a diferentes distancias entre las cámaras. En la figura 4 se muestra la estructura utilizada en el presente trabajo.

Figura 4. Base de soporte para las cámaras

 ISSN 2007-8102

Máquina virtual en Raspberry-PI 3

Se instaló el sistema operativo de Lubuntu en la micro SD de la tarjeta Raspberry-PI 3 puesto que utiliza menos espacio de memoria y, posteriormente, se instaló Python 3.6 con las librerías antes mencionadas. Además, se instalaron los drivers necesarios para la vinculación con las cámaras web.

Descripción del algoritmo

En la figura 5 se describe en un diagrama de flujo la programación con Python del algoritmo implementado. Cada etapa es descrita con mayor precisión a continuación.

Figura 5. Diagrama de flujo del algoritmo

Primeramente, se activan las cámaras web y se confirma la captura de las imágenes con la creación de las imágenes en la memoria. También, se verifica que las imágenes mantengan el mismo tamaño en pixeles para que sea posible aplicarles los pasos siguientes. Si no se cumple esta característica, el mapa de disparidad no puede ser generado.

Una vez capturadas las imágenes, para generar el mapa de disparidad se hace uso de la geometría epipolar, la cual es aquella dada por la intersección del plano de cada imagen con una familia de planos, donde todos los planos de esta familia contienen la línea base que une los centros ópticos de las cámaras. La importancia de esto es que somos capaces de calcular exactamente dónde está el punto del epípolo en nuestra imagen al encontrar el punto de la imagen donde se cruzan múltiples epilíneas.

Además, son necesarias las matrices esencial y fundamental. La matriz esencial contiene información sobre la traslación y rotación que describe la posición de la cámara con respecto a su contraparte. Pero para nuestra solución queremos hacer nuestros cálculos en términos de coordenadas de píxeles de la cámara, y es aquí donde entra en juego la matriz fundamental, ya que contiene la misma información que la matriz esencial, pero también incluye información intrínseca sobre ambas cámaras para que podamos relacionar las dos en tales términos.

Afortunadamente, podemos usar OpenCV para derivar la matriz de un conjunto de puntos coincidentes conocidos en cada imagen. Este es esencialmente un proceso de calibración por el cual se recomienda encontrar y utilizar un mínimo de 8 puntos de coincidencia de las imágenes para lograr la precisión necesaria.

Por lo tanto, el mapa de disparidad es generado con el apoyo de la librería de OpenCV. La función StereoSGBM_create permite la obtención del mapa por medio de las imágenes, sin embargo, requiere de un gran número de parámetros para que el resultado obtenido tenga la precisión adecuada para posteriormente crear la nube de puntos de la reconstrucción tridimensional. Estos parámetros son los siguientes: windowSize, minDisparity, numDisparities, blockSize, P1, P2, preFilterCap, uniquenessRatio, speckleWindowSize y speckleRange. Cada una de estas propiedades se debe modificar de acuerdo a las características del entorno donde se capturen las imágenes para obtener la óptima calidad de los resultados.

Posteriormente, se extrae la profundidad del mapa de disparidad obtenido y por medio de la librería de Pandas se genera y se guarda la nube de puntos con la información de la profundidad de cada píxel, conservando el color de la imagen y obteniendo un archivo con la extensión ".ply" para ejecutar con un visualizador de objetos 3D.

Por último, empleando la librería de Open3D se puede abrir el archivo ".ply" generado y nos permite visualizar la reconstrucción tridimensional de los objetos que capturamos en las imágenes. En la ventana de Open3D podemos configurar el grosor de los puntos de la reconstrucción, así como trasladar y rotar la recreación del entorno para mejorar la observación del resultado obtenido.

 ISSN 2007-8102

Resultados y discusión

Primera prueba

Los resultados obtenidos durante la primera prueba no permitían observar realmente la profundidad que existía de los objetos, puesto que los parámetros que se colocaron para diseñar el mapa de disparidad fueron los valores estándares que proponía el ejemplo de Visión estéreo de OpenCV. En las figuras 6 y 7 se pueden observar las vistas frontal y lateral, respectivamente, de la explanada del Instituto Tecnológico de Orizaba, donde se tomaron las primeras capturas. Los parámetros utilizados en esta prueba se muestran en la tabla 1.

Figura 6. Vista frontal de la explanada

Figura 7. Vista lateral de la explanada

Tabla 1. Parámetros del mapa de disparidad de la primera prueba

Parámetro	Valor
windowSize	3
minDisparity	0
numDisparities	16
blockSize	5
P1	8*3*windowSize**4
P2	32*3*windowSize**4
preFilterCap	20
uniquenessRatio	10
speckleWindowSize	100
speckleRange	16

 ISSN 2007-8102

Reconstrucción 3D con parámetros modificados

Se decidió modificar los valores de los parámetros para mejorar la calidad de las reconstrucciones 3D y los resultados obtenidos mejoraron considerablemente, puesto que ya se lograban observar las diferentes profundidades de los objetos en las imágenes. En las figuras 8 y 9 se pueden observar las vistas frontal y lateral, respectivamente, del laboratorio de investigación. Los parámetros utilizados en esta prueba se muestran en la tabla 2.

Figura 8. Vista frontal del laboratorio

Figura 9. Vista lateral del laboratorio

Tabla 2. Parámetros del mapa de disparidad de la segunda prueba

Parámetro	Valor
windowSize	11
minDisparity	0
numDisparities	16
blockSize	5
P1	8*3*windowSize**4
P2	32*3*windowSize**4
preFilterCap	50
uniquenessRatio	20
speckleWindowSize	150
speckleRange	32

 ISSN 2007-8102

Por último, se realizaron capturas en el cubículo de profesores alterando nuevamente el valor de los parámetros. En las figuras 10 y 11 se pueden observar los resultados obtenidos de las vistas frontal y lateral del cubículo. Los parámetros utilizados en esta prueba se muestran en la tabla 3.

Figura 10. Vista frontal del cubículo

Figura 11. Vista lateral del cubículo

Tabla 3. Parámetros del mapa de disparidad de la segunda prueba

Parámetro	Valor
windowSize	7
minDisparity	0
numDisparities	16
blockSize	1
P1	8*3*windowSize**4
P2	32*3*windowSize**4
preFilterCap	100
uniquenessRatio	20
speckleWindowSize	150
speckleRange	32

 ISSN 2007-8102

Se logró notar que para las capturas en las que los objetos no estuvieran tan alejados de las cámaras, el parámetro windowSize tenía que reducirse, además que, dependiendo de la iluminación del medio, los valores de los parámetros preFilterCap y blockSize permitían mejorar la calidad de la reconstrucción al incrementar su valor. A mayor iluminación, mayor el valor que debían tener estas propiedades. Por otra parte, la distancia que se mantuvo entre las cámaras se decidió establecer de 15cm, puesto que dio los mejores resultados con esta separación.

Además de los resultados anteriores podemos hablar del tiempo de ejecución, el cual, según las pruebas realizadas, cuando se iniciaron eran de 15 segundos; esto bajo condiciones no ideales, como una mayor distancia entre las cámaras o una disparidad muy grande. Se mencionó anteriormente que la distancia optima es de 15cm, en esta condición los frames tienen una disparidad ideal, bajando el tiempo de ejecución hasta 10 segundos.

Trabajo a futuro

Se planea mejorar la calidad de las reconstrucciones tridimensionales al obtener cámaras de mayor resolución, además, se busca incorporar el algoritmo presentado en un campo más amplio de aplicaciones, por ejemplo, la reconstrucción 3D de entornos y objetos mediante el desplazamiento de robots móviles en tiempo real y disminuyendo el tiempo de ejecución.

Conclusiones

Se obtuvieron reconstrucciones tridimensionales de buena calidad por medio del método pasivo de reconstrucción 3D denominado visión estéreo binocular. Es muy importante analizar y comprender cada una de las funciones que las librerías de Python nos otorgan, así como el significado de los parámetros que sus funciones utilicen, puesto que de estos parámetros depende la calidad de la reconstrucción obtenida. Para que los resultados sean óptimos, las cámaras deben mantener una relación de distancia en pixeles muy próxima entre ellas, así como también tener una estructura rígida que permita capturas estables. Las reconstrucciones 3D que se obtuvieron mejoraron de acuerdo a la configuración de los parámetros para la creación del mapa de disparidad. Además, se permitió observar la importancia de la iluminación del medio en que se hayan capturado las imágenes y la distancia que exista entre las cámaras y los objetos de la imagen, puesto que dichas características también es necesario tomarlas en consideración a la hora de modificar los parámetros del algoritmo. Finalmente, al emplear la tarjeta Raspberry-PI 3 se disminuyó el espacio requerido y se facilitó su adaptación para futuras aplicaciones.

Referencias

[1] Grandón, N.; Aracena, D.; Luis, Clésio. (2007). 3D Object Reconstruction with calibrated images.
[2] Theo, M.; Vergauwen, M. "3D reconstruction from multiple images part 1: Principles." Foundations and Trends in Computer Graphics and Vision. 287-404.
[3] Fernández, J. (2017). Study and implementation of depth map from Stereo Matching algorithm.
[4] Brahmbhatt, S. (2013). Practical OpenCV. Ed. Technology in action. 173-200.
[5] Howse, J.; Joshi, P.; Beyeler, M. (2016). OpenCV: Computer Vision Projects with Python. Learning Path. Ed. Packt. 357-385.
[6] Joshi, P. (2015). OpenCV with Python By Example. Ed. Packt. 388-420.
[7] Heath, N. (2017). Raspberry-PI 3 [Online]. Available: https://www.zdnet.com/article/what-is-the-raspberry-pi-3-everything-you-need-to-know-about-the-tiny-low-cost-computer/