DESDE 2013
https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI
ISSN: 2007-6363

Publicación Semestral Pädi Vol. 10 No. Especial Especial 4 (2022) 128–135

Reconstrucción 3D Monocular de objetos con cámara montada sobre un dron
Monocular 3D reconstruction of objects, using a camera on a drone

A. Juárez-Terrazas [a,*], A. Cureño-Ramírez [c], J.M. Ibarra-Zannatha [a]

[a] *Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas del IPN Av IPN 2580, La Laguna Ticomán, Alcaldía GAM, 07340 Ciudad de México, CDMX.*
[b] *Área Académica de Matemáticas y Física, Universidad Autónoma del Estado de Hidalgo, 42184, Pachuca, Hidalgo, México.*
[c] *Centro de Investigación y de Estudios Avanzados del IPN, Av. IPN 2508, San Pedro Zacatenco Alcaldía GAM, 07360 Ciudad de México, CDMX.*

Resumen

En este articulo se presenta la metodología utilizada para obtener una reconstrucción en 3D de uno o varios objetos de interés en un ambiente desconocido por medio de un sistema de videoSLAM monocular (ORBSLAM2) que procesa el video generado por la cámara monocular montada en un dron. La metodología implementada para ello se basa en el aprovechamiento de software de uso libre enfocado en este tipo de cámaras. El resultado de este trabajo será utilizado para resolver una de las pruebas del Torneo Mexicano de Robótica en su categoría de Drones Autónomos

Palabras Clave: Reconstrucción 3D, Visión por computadora, Powercrust, Screened Poisson, ORBSLAM2

Abstract

This article presents the methodology used to obtain a 3D reconstruction of one or several objects of interest in an unknown environment by means of a monocular videoSLAM system processing the video signal produced by a monocular camera mounted on a drone. The implemented metodology is based on the use of free software focused on this type of cameras. Our results will be used to solve one of the challenges included in the Autonomous Drones Category of the Mexicain Robotic Tournement.

Keywords: 3D Reconstruction, Computer vision, Powercrust, Screened Poisson, ORBSLAM2

1. Introducción

En los últimos años se ha realizado bastante investigación relativa a los drones multirotor. La versatilidad que tienen estos robots móviles los hacen atractivos para el ambiente académico a la hora de proponer soluciones para diversos problemas. Es por esto que también han proliferado diversas competiciones con drones tanto en México como a nivel internacional. Estas suelen incluir pruebas de navegación, pruebas de percepción visual, pruebas de rescate y muchas más. En el contexto del presente trabajo nos interesamos en una de estas aplicaciones: La reconstrucción 3D de objetos presentes en un entorno desconocido.

La herramienta por excelencia para este tipo de tareas es la técnica de ORBSLAM. En su versión más simple, se utilizan las imágenes capturadas por una cámara a bordo de un robót móvil para extraer puntos conocidos como "*keypoints*". Estos puntos se pueden etiquetar de manera única en una sucesión de imágenes y usando el cambio de posición de estos *keypoints* es posible extraer tanto la pose y trayectoria de la cámara; como una nube de puntos que sirve de mapa. Por ejemplo en el trabajo de Yusefi et al. (2020), se utiliza un dron con una cámara a bordo para hacer un mapa 2D un entorno desconocido. Una vez terminado dicho mapa, el sistema diseñado fue capaz de usar el algoritmo A^* para planear una trayectoria dentro de dicho mapa. A pesar del buen desempeño del sistema de SLAM, Los resultados expuestos son en simulador y su sistema también dependía de la información de la IMU del dron para hacer el mapa.

Otro trabajo reseñable es el de Li et al. (2022), quienes montaron una cámara RGBD a bordo de un dron para crear un mapa del entorno. La versión modificada de ORBSLAM que proponen les permitió no nada más obtener una nube de puntos

*Autor para correspondencia: aaronjt@outlook.com
Correo electrónico: aaronjt@outlook.com (Aarón Juárez-Terrazas), andres.cureno@cinvestav.mx (Andres Cureño-Ramírez), jibarra@cinvestav.mx (Juan Manuel Ibarra-Zannatha)

Historial del manuscrito: recibido el 16/05/2022, última versión-revisada recibida el 24/07/2022, aceptado el 09/08/2022, publicado el 05/10/2022
DOI: https://doi.org/10.29057/icbi.v10iEspecial4.9272

sino también una reconstrucción 3D del entorno usando redes neuronales convolucionales. A pesar de haber obtenido resultados experimentales favorables, fue necesario añadir una cámara de profundidad al proyecto. También se puede resaltar que la frecuencia de adquisición de datos resultó de 12 *hz* lo cual se puede considerar como un sistema de tiempo real; pero que no puede ser utilizado para tareas que necesiten de rapidez.

Así, en este artículo se presentan los resultados en simulador obtenidos por nuestro equipo de Robótica de Competición, en la preparación de la competición correspondiente a la Categoría de Drones Autónomos del Torneo Mexicano de Robótica (TMR), específicamente para la prueba de reconstrucción en 3D. Cabe mencionar que dicho equipo está formado exclusivamente por estudiantes de pregrado realizando su Servicio Social o su tesis de licenciatura.

1.1. Problema a Resolver

El problema de interés es uno de los propuestos en la competición del Torneo Mexicano de Robótica (TMR), en la categoría de drones autónomos; cuyo reglamento dice (sic): "*El dron deberá de volar de manera autónoma alrededor de una estructura texturizada ensamblada con cubos con volumen de 1 x 1 x 1 metros cúbicos como se aprecia en la Figura 1. La estructura no superará un volumen de 4 x 4 x 4 metros cúbicos y será ensamblada de manera aleatoria antes del inicio de la misión. El dron deberá generar un mapa 3D de dicha estructura y mostrar su construcción en tiempo real en un monitor sobre la estación de control. Un conjunto de jueces otorgará el puntaje de acuerdo a la calidad del mapa 3D construido*".

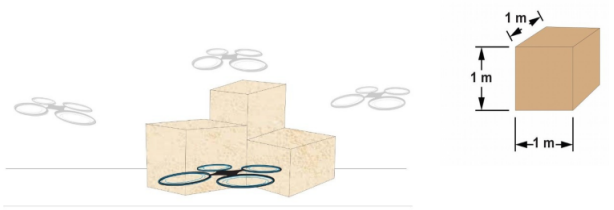

Figura 1: Prueba del TMR, reconstrucción en 3D de cajas texturizadas. Reglamento Drones Autónomos TMR.

1.2. Solución Propuesta

La propuesta de solución a este problema consta de un dron (simulado en el ambiente de Gazebo) que tendrá que volar a una cierta altitud y seguir una trayectoria circular alrededor del entorno conteniendo los objetos de los que se debe obtener un modelo 3D. Para lograrlo el dron debe apuntar su cámara al suelo justo abajo del centro de su trayectoria circular con el fin de obtener un barrido de 360° del conjunto de objetos de interés. De manera simultánea, el video obtenido por la cámara queda disponible como un tópico de ROS y, entonces, ya se le puede aplicar la técnica de videoSLAM monocular denominada ORBSLAM2.

Como resultado de la aplicación de ORBSLAM2 se obtiene una nube de puntos en 3D representando la superficie de los objetos presentes en el espacio de trabajo del dron. Dicha nube de puntos suele contener puntos que no pertenecen a los objetos de interés, ya sea por defectos de iluminación o por la presencia de sombras o de texturas ajenas a los objetos de interés. Estos puntos espurios, denominados *Outlayers*, son considerados como ruido y deben ser eliminados antes de convertir la nube de puntos en una superficie. de entre la cantidad de técnicas disponibles para la eliminación de *Outlayers* se eligieron métodos estadísticos con versiones en librerías de software libre.

Finalmente, para convertir estas nubes de puntos debidamente depuradas y ya sin *Outlayers* se emplearon un par de algoritmos de reconstrucción 3D que convierten la nube de puntos en una malla de polígonos modelando la superficie que dichos puntos representan. Estos algoritmos son: Powercrust y Screened Poisson, los cuales permiten reconstrucciones de nubes en 3D a una malla de polígonos en 3D. Finalmente, se analizan los resultados obtenidos con ambos algoritmos.

Si bien este problema es meramente de tipo académico, cabe mencionar que este tipo de reconstrucción 3D se puede utilizar para realizar mapas de utilidad para la agricultura, para el reconocimiento de zonas de desastre como terremotos o inundaciones o hasta en aplicaciones médicas como se presenta en el trabajo *SLAM-based dense surface reconstruction in monocular Minimally Invasive Surgery and its application to Augmented Reality* Chen et al. (2018).

2. Metodología

Una vez definido el problema a resolver se optó por obtener la nube de puntos de los objetos de interés con la técnica de ORBSLAM2 Mur-Artal and Tardos (2017) debido a que se puede utilizar en tiempo real con ROS y es de uso libre, este realiza la localización y mapeo de manera simultanea por medio de puntos clave en las imágenes con el método de detección ORB que se puede utilizar como una función de OpenCV. Una vez obtenida la nube de puntos, ésta tendrá que filtrarse con el fin de eliminar aquellos puntos que no pertenecen a los objetos que se quieren reconstruir. Finalmente, teniendo ya la nube lo mas limpia posible, se procede a realizar la reconstrucción, para lo cual se utilizaron dos métodos de reconstrucción 3D: Powercrust Amenta et al. (2004) y Screened Poisson Kazhdan and Hoppe (2013), más adelante se presentan estos métodos con más detalle .

En la Figura 2 se muestra la nube de puntos correspondiente a un objeto de prueba denominado *nudo*, la cual será utilizada como conjunto de prueba para realizar más adelante su reconstrucción 3D con los métodos antes mencionados.

2.1. Algoritmos de reconstrucción

Los algoritmos que se escogieron para realizar la tarea de realizar una reconstrucción en 3D de una nube de puntos, tal como se comentó más arriba, son Powercrust y Screened Poisson, los cuales han demostrado un buen funcionamiento en muchos casos y, además, estos algoritmos son de uso libre, cuentan con su respectiva documentación y son compatibles

con Linux (Ubuntu 20), que es el sistema operativo en el que corren ORBSLAM2 y ROS Noetic.

A continuación se aplicarán estos dos algoritmos de generación del mallado que convierte nubes de puntos en una superficie representando objetos volumétricos, tarea a la cual se le denomina Reconstrucción 3D, utilizando para ello la nube de puntos de prueba *nudo*. Es importante mencionar que las reconstrucciones de ejemplo con los dos algoritmos permitirán observar los resultados obtenidos con cada uno de ellos, además, debemos decir que esta comparación es meramente subjetiva, cualitativa y visual. Finalmente, mencionaremos que el software utilizado para visualizar tanto las nubes de puntos como las reconstrucciones se llama MeshLab, herramienta que también es de uso libre y es compatible con Linux .

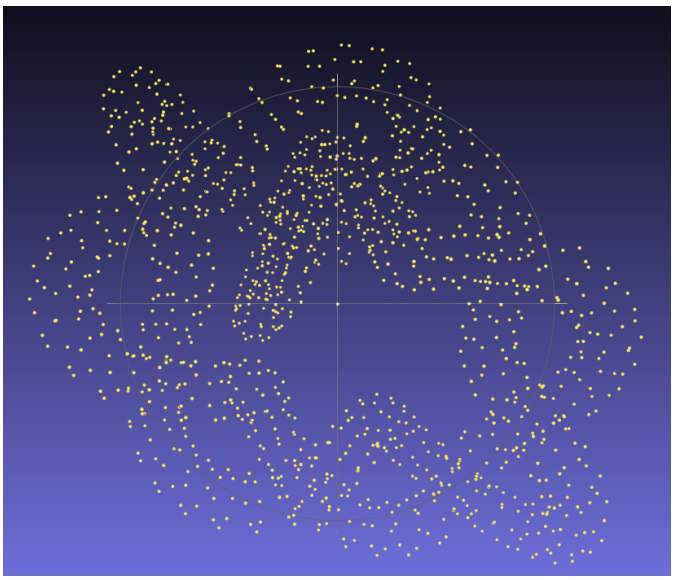

Figura 2: Nube de ejemplo “nudo”.

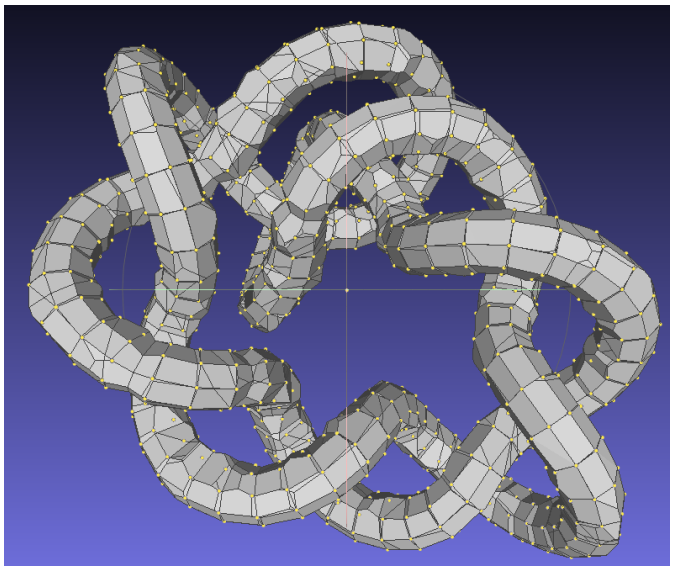

Figura 3: Reconstrucción de ejemplo “nudo” con Powercrust.

2.1.1. *Powercrust*

Este método consiste en: *(i)* Calcular el diagrama de Voronoid de la nube de puntos. *(ii)* Usar dicho diagrama para aproximar el eje medio de la superficie como un conjunto finito de bolas (Otra forma de llamar al eje medio de una superficie es esqueleto debido a que simplifica a una superficie como una sucesión de ramas y conexiones), y *(iii)* usar una transformación inversa para obtener la superficie a partir del eje medio.

Para utilizar Powercrust se utilizó el software provisto en Kona (2015). Los parámetros configurables necesarios se explican en el trabajo de Amenta et al. (2004). En la Figura 3 se muestra el resultado de aplicar este algoritmo de mallado sobre la nube de puntos de prueba mostrada en la Figura 2.

Este procedimiento es una versión mejorada del algoritmo original llamado Crust Amenta et al. (1998), solo que en esta versión se mejora la relación de puntos para la reconstrucción de objetos con el fin de que se respete una mejor interacción entre los puntos y que estos se mantengan aislados el uno del otro para que no exista una unión no deseada, por ejemplo el espacio entre los dedos de una mano.

2.1.2. *Screened Poisson*

Este método parte de la idea de que, dado un conjunto de puntos P, los cuales son una muestra de la superficie S que se desea aproximar; a cada punto de P se le puede asociar un vector normal a la superficie S y construir un campo vectorial $\vec{V}$. Así pues, el método de *Screened Poisson* busca un campo escalar $\mathcal{X}$ el cual minimice la función de costo:

$$E(\mathcal{X}) = \int \|\nabla\mathcal{X}(p) - \vec{V}(p)\|^2 dp$$

Se puede demostrar que, dicha solución cumple que:

$$\Delta\mathcal{X}(p) = \nabla \cdot \vec{V}$$

A diferencia de Powercrust, la reconstrucción con Screened Poisson puede obtener superficies más suaves, sobre todo si cuenta con una buena información acerca de los vectores normales sobre cada uno de los puntos de la nube puntos de interés. En Figura 4 se muestra la reconstrucción con Screened Poisson de la nube de puntos de prueba mostrada en la Figura 2.

Una vez comprobado que ambos métodos funcionan correctamente para la reconstrucción 3D a partir de una nube de puntos, se prosiguió a obtener la nube puntos en tiempo real de algún objeto para poder realizar las pruebas con diferentes objetos.

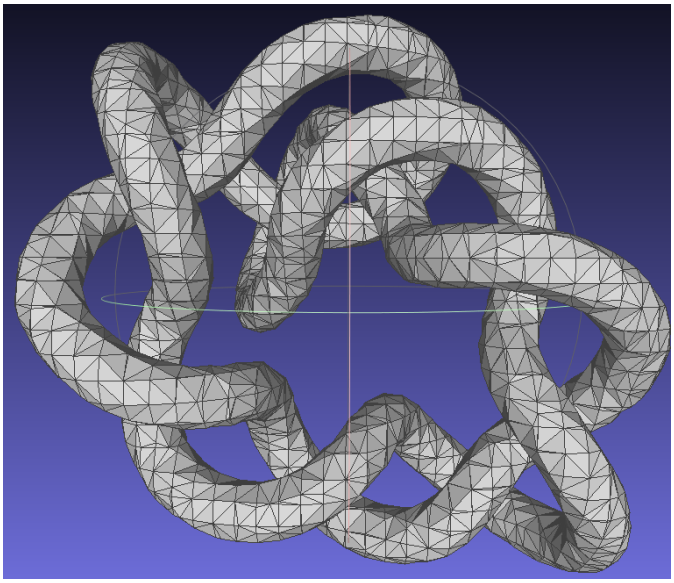

Figura 4: Reconstrucción de ejemplo “nudo” con Screened Poisson.

2.2. *Obtención de nubes puntos con ORBSLAM2*

ORBSLAM2 cuenta con la posibilidad de ejecutarse en tiempo real a través de ROS, concretamente se suscribe al tópico **/camera/image_raw** que es donde se tendrá que publicar la imagen en tiempo real del dron, por lo que se desarrolló un nodo que se encargara de publicar la imagen de la cámara de un celular por medio de RTSP a través una red local.

Cabe mencionar que se modificó el código fuente de ORBSLAM2 para poder cambiar los colores de la presentación de los puntos, se agregó una función que se encargará de guardar las nubes al finalizar la ejecución y además se adaptó para poder ser ejecutado sobre Ubuntu 20, el repositorio de esta modificación puede consultarse en Mur-Artal and Tardos (2022).

En la Figura 5 se muestra un ejemplo sencillo en el que se obtuvo en tiempo real la nube de puntos de un cojín texturizado, como es evidente, el color del renderizado es distinto al original, es decir, ya cuenta con las modificaciones anteriormente mencionadas.

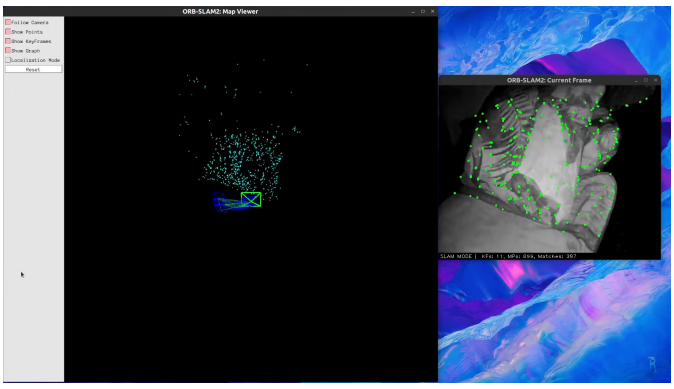

Figura 5: Obtención de nube en tiempo real con un cojín texturizado.

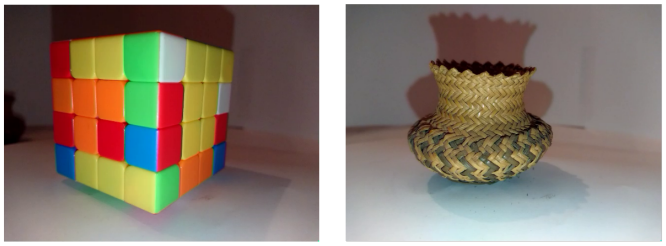

Figura 6: Objetos de prueba: un cubo de Rubik y una canasta.

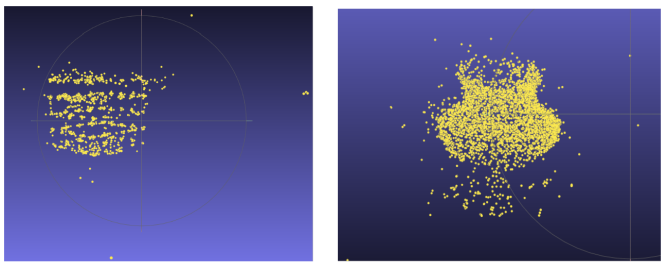

Figura 7: Nubes obtenidas de, Izquierda: Cubo de Rubik, Derecha: Canasta.

2.2.1. *Reconstrucción de nubes de puntos sin filtrar*

Para comenzar con estas pruebas se seleccionaron dos objetos: un cubo de Rubik 4x4 y una pequeña canasta, objetos que se ilustran en la Figura 6. Se adecuó un ambiente de pruebas que aislara el sistema lo mejor posible de objetos que pudieran generar ruido en la nube, por lo que se adaptó un fondo blanco, aunque no se hizo un control estricto de la iluminación, es decir que no se evitaron sombras ni brillos excesivos.

Cabe mencionar que en estos casos la obtención de nubes de puntos se hizo en tiempo real con objetos reales, para lo cual la cámara se mantuvo fija mientras que el objeto bajo observación se hizo girar sobre su propio eje vertical. Una vez listo el ambiente de pruebas se prosiguió a obtener las nubes de puntos cada uno de los objetos, habiendo obtenido el resultado se muestra en la Figura 7.

En la Figura 7 se observa claramente que se obtuvieron nubes puntos bastantes cercanas a lo que es la descripción real del objeto. Sin embargo, también puede verse que se obtuvieron puntos que se encuentran bastante alejados del objeto de interés, estos puntos son denominados Outlayers y se les considera ruido pues al momento de realizar las reconstrucciones con cualquiera de los métodos antes mencionados estos puntos outlayers pueden producir deformaciones.

En las figuras 8 y 9 se muestran los resultados de reconstrucción de los objetos de prueba con los dos diferentes métodos de reconstrucción seleccionados. En estas figuras es posible observar que ambas reconstrucciones resultaron afectadas por los outlayers, produciendo deformaciones hacia dichos puntos. Por lo tanto es necesario realizar algún tipo de filtrado de esas nubes que permita eliminar o por lo menos disminuir la cantidad de puntos espurios o ruido.

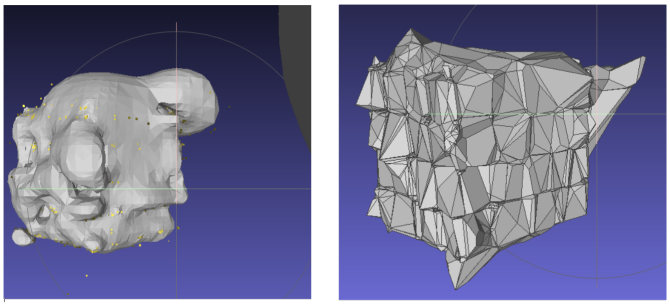

Figura 8: Reconstrucción de cubo de Rubik. A la izquierda el resultado de Screened Poisson y la derecha el resultado de Powercrust.

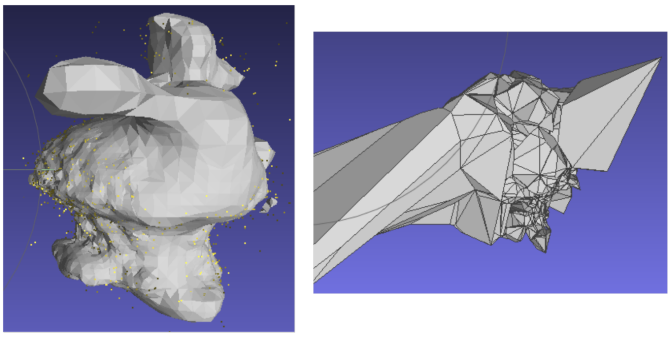

Figura 9: Reconstrucción de la canasta. A la izquierda el resultado de Screened Poisson y la derecha el resultado de Powercrust.

2.3. *Filtrado de nubes*

Para poder dar solución al problema de eliminación de puntos espurios se utilizó una librería de código abierto llamada Open3D Zhou et al. (2018) la cual permite manipular nubes de puntos. El método que se utilizó dentro de Open3D tiene el nombre de *remove_statistical_outlier* el cual permite remover aquellos puntos que se pueden considerar como outlayers, como ruido.

Este algoritmo utiliza métodos estadísticos y al momento de utilizarse deben proporcionarse dos parámetros: el primero es el **radio** que permite establecer el nivel de umbral en función de la desviación estándar de las distancias promedio a través de la nube de puntos. Cuanto menor sea este número, más agresivo será el filtro pues sólo conservará puntos que estén muy cercanos entre si. Y **el número de vecinos** que especifica cuántos vecinos se deben tomar en cuenta para calcular la distancia media de un punto dado.

Figura 10: Resultado obtenido del método de filtrado con la nube de puntos del cubo de Rubik.

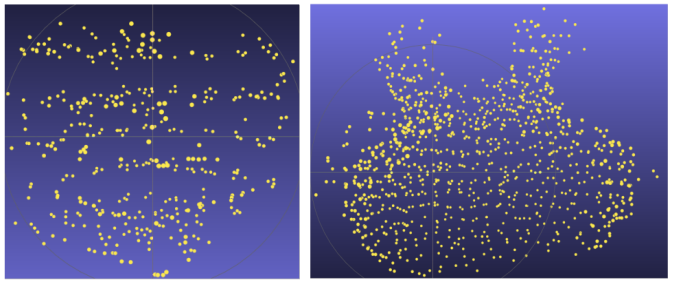

Figura 11: Nubes de puntos filtradas. En la izquierda la nube de puntos del cubo de Rubik y en la derecha la nube de puntos de la canasta.

Para la realización de estas pruebas se variaron ambos parámetros de manera independiente para cada nube de puntos que se quería filtrar. De este modo, en la Figura 10 se muestra un ejemplo con el cubo de Rubik. Los puntos que se muestran en rojo son aquellos que serán eliminados y los grises los que permanecerán. Las nubes de puntos filtradas de los dos objetos de interés se muestran en la Figura 11 .

2.3.1. *Reconstrucción de nubes filtradas*

Para finalizar la propuesta de cómo es que se llevará a cabo la reconstrucción, se realizaron las reconstrucciones de las nubes filtradas de las Figuras 12 y 13 con los dos métodos de reconstrucción propuestos. Estas reconstrucciones se notan mas uniformes y cercanas a los objetos que que originaron la nube de puntos mediante ORBSLAM2, por lo que el algoritmo utilizado para realizar el filtrado resultó ser positivo para la reconstrucción.

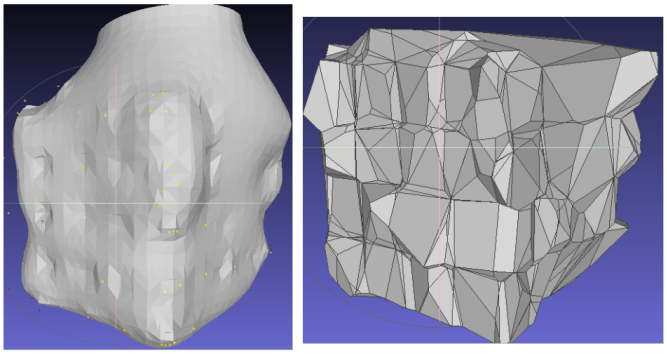

Figura 12: Reconstrucción de cubo de Rubik con nube de puntos filtrada, en la izquierda con Screened Poisson y en la derecha con Powercrust.

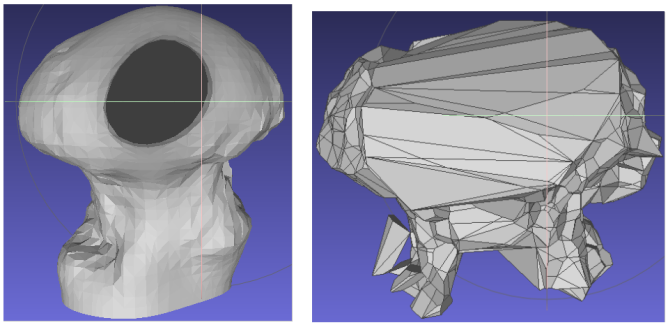

Figura 13: Reconstrucción de canasta con nube de puntos filtrada, en la izquierda con Screened Poisson y en la derecha con Powercrust.

3. Resultados de la simulación

Para llevar a cabo la reconstrucción 3D de un objeto usando una cámara monocular en un ambiente simulado se utilizó el software de ROS Noetic y el simulador Gazebo. Parte del software utilizado en la implementación se puede encontrar en Johannes Meyer (2022). Así pues, con el entorno listo se comenzó a programar la estrategia de vuelo del dron para que se pudiera realizar el mapeo lo mejor posible. La propuesta es la siguiente: *(i)* el dron comenzará en el piso, despegará y deberá alcanzar una altitud previamente definida, *(ii)* luego comenzará a seguir una trayectoria circular con radio constante alrededor del objeto que se desea reconstruir. *(iii)* Al mismo tiempo, se controlará la orientación del dron de modo que la cámara siempre apunte hacia el centro de dicho círculo. *(iv)* Cuando haya recorrido la trayectoria una o más veces, el dron se detendrá y con las imágenes capturadas se obtendrá una nube de puntos mediante la técnica de ORBSLAM2. *(v)* Finalmente se realizará el filtrado de dicha nube y su correspondiente mallado. La figura 14 ilustra la maniobra diseñada.

El dron simulado tiene como entradas de control al ángulo de guiñada (*yaw*) y las velocidades lineales en las direcciones x, y y z. Así que, como ley de control se utiliza:

$$\dot{x_{dron}} = k_p * e_r \tag{1}$$
$$\dot{y_{dron}} = -\text{cte.} \tag{2}$$
$$\dot{z_{dron}} = 0 \tag{3}$$
$$\dot{Yaw}_{dron} = k_p * e_{yaw} \tag{4}$$

Siendo:

$$e_r = r_{dron} - r_{circulo} \tag{5}$$
$$e_{yaw} = \arctan\left(\frac{y_{dron} - y_{circ}}{x_{dron} - x_{circ}}\right) + \pi - Yaw_{dron} \tag{6}$$

Para mantener al dron a la misma altura simplemente se usa como señal de control a (3). Por otro lado, la señal de control (2) mueve lateralmente al dron a una velocidad constante mientras que la señal (1) lo mueve longitudinalmente. Esta última está definida en términos del radio de la circunferencia que se desea seguir (r_{circ}) y se asegura de mantener al dron a una distancia constante del centro. Finalmente la señal (4) mantiene orientado al dron en la dirección deseada. En la figura 15 se muestran algunos de los parámetros utilizados.

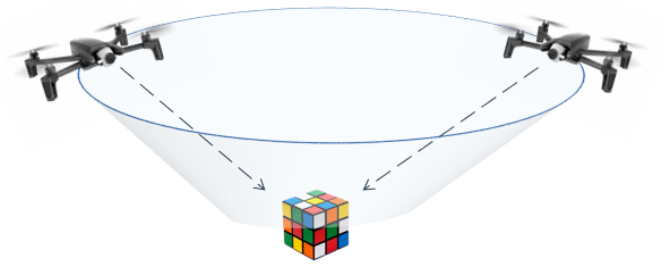

Figura 14: Estrategia de vuelo para la reconstrucción en 3D.

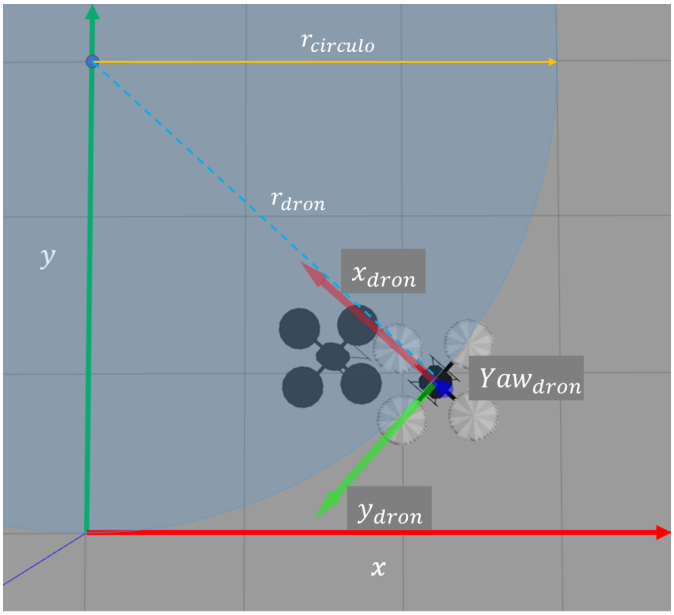

Figura 15: Parámetros utilizados en la estrategia de control.

Para la prueba de la reconstrucción dentro de la simulación se utilizó el modelo de un automóvil llamado *Hatchback* incluido en el simulador, al cual se le modificó la textura de la carrocería con el fin de tener una superficie con más puntos clave, en la Figura 16 se puede observar dicho modelo.

Figura 16: Hatchback modificado utilizado en la simulación.

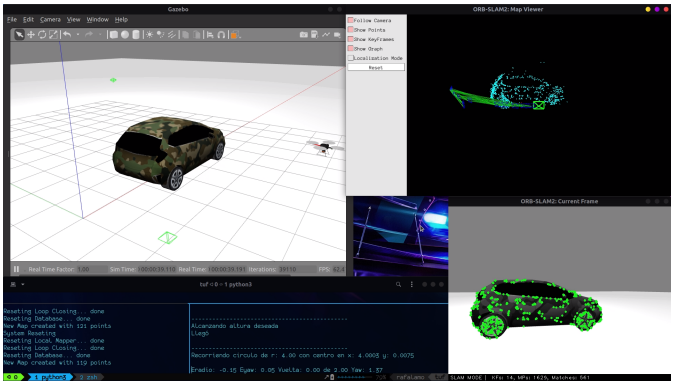

Figura 17: Mapeo de Hatchback, inicialización y trayectoria.

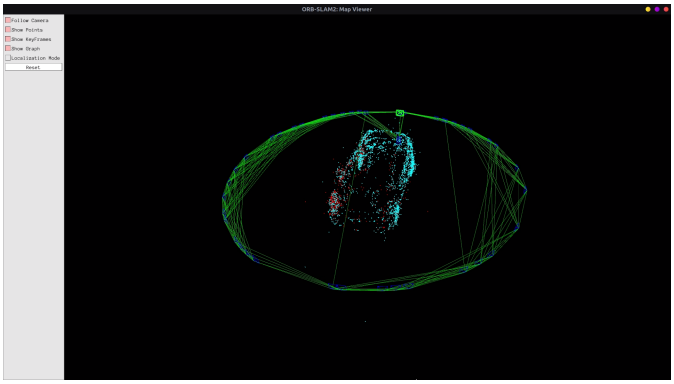

Figura 18: Mapeo final de Hatchback.

3.1. *Pruebas en la simulación*

Teniendo definido como se llevará a cabo el seguimiento de la trayectoria, se integró todo lo antes mencionado y se realizó la simulación del dron con su cámara conectada a ORBSLAM2 por medio de ROS para obtener la nube de puntos del objeto de interés, que en este caso es el automóvil.

En las Figuras 17 y 18 se muestra como es que se llevó a cabo la obtención de la nube de puntos en tiempo real (dentro del simulador). En Juárez (2022) se puede ver un vídeo de esta prueba .

Terminada la simulación es posible visualizar la nube puntos fuera de línea como se muestra en la Figura 19, en donde se observa que existen varios puntos outlayers que se consideran como ruido, pues es muy posible que generen deformaciones en las reconstrucciones posteriores. En las Figuras 20 y 21se muestran las gráficas tanto del error del radio durante el seguimiento de la trayectoria, así como una gráfica de la trayectoria ideal contra la resultante del dron.

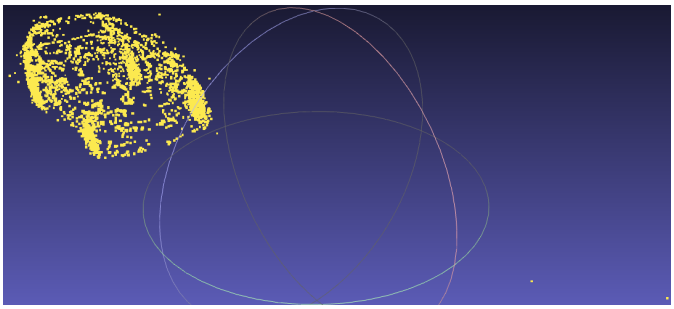

Figura 19: Nube de puntos sin filtrar de Hatchback.

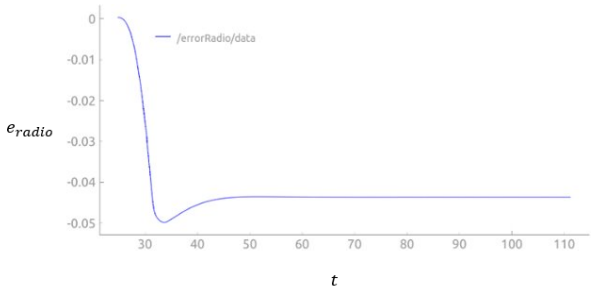

Figura 20: Gráfica de error contra el tiempo del radio del circulo.

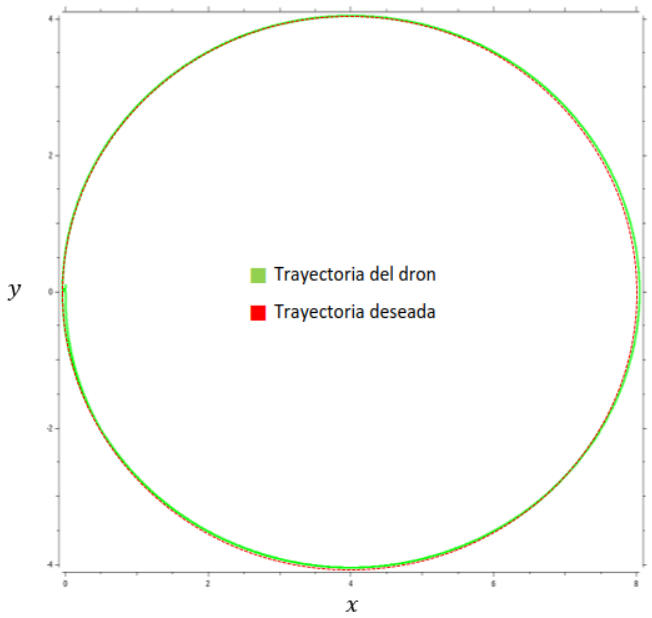

Figura 21: Gráfica de trayectoria deseada contra la trayectoria obtenida por el dron.

3.1.1. *Reconstrucción de nube de puntos sin filtrar*

Terminada la obtención de la nube de puntos, se realizaron las reconstrucciones utilizando los métodos ya mencionados y en la Figura 22 se muestran los resultados.

Es posible que las reconstrucciones obtenidas sean bastante malas, sobretodo en el caso de Screened Poisson, ya que para poder utilizar este algoritmo es necesario conocer los vectores normales de cada punto y con una nube sin filtrar es demasiado complicado realizar este cálculo. Con Powercrust se obtiene un modelo un poco mas cercano al objeto real, pero aún así cuenta con muchas deformaciones.

3.2. *Filtrado de nube*

A fin de evitar la aparición de estas deformaciones causadas por los outlayers se realizó el filtrado de la nube de puntos para eliminar aquellos puntos que generan ruido en las reconstrucciones en la Figura 23 se muestra la nube de puntos filtrada.

3.2.1. *Reconstrucción de nube filtrada*

Finalmente se realizó la reconstrucción de la nube filtrada con ambos métodos como se muestra en la Figura 24

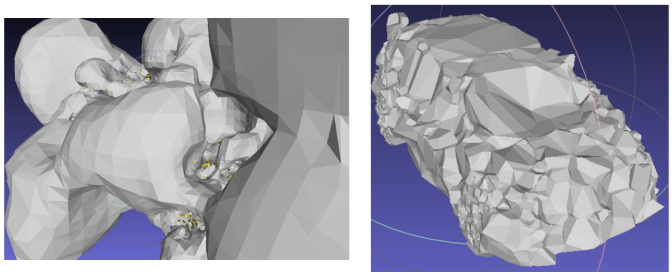

Figura 22: Reconstrucción de Hatchback con nube de puntos sin filtrar, en la izquierda con Screened Poisson y en la derecha con Powercrust.

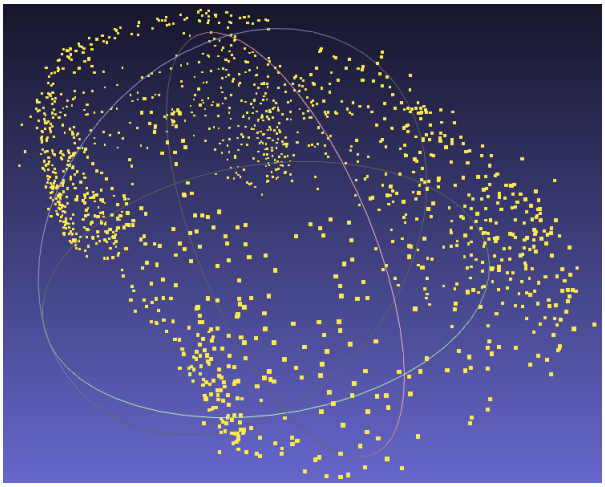

Figura 23: Nube de puntos filtrada de Hatchback.

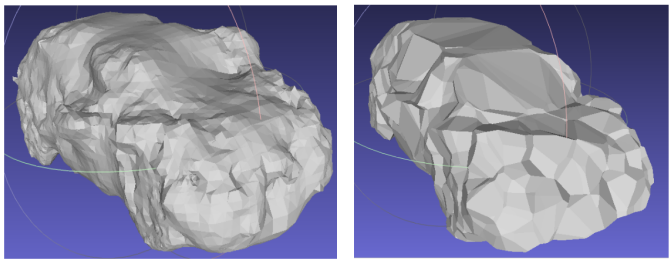

Figura 24: Reconstrucción de Hatchback con nube de puntos filtrada, en la izquierda con Screened Poisson y en la derecha con Powercrust.

Aunque en ambas reconstrucciones el resultado aún tiene deformaciones, el filtrado mejora bastante la reconstrucción y se considera que de se obtiene un modelo lo mínimamente descriptivo para poder ser relacionado con el modelo 3D original.

4. Conclusiones

Primeramente se concluye que en cualquier algoritmo de reconstrucción siempre será necesario realizar un preprocesado de los datos para eliminar lo mayor cantidad posible de puntos outlayers, es decir de aquellos puntos espurios que se consideren como ruido y que puedan afectar al resultado final de la reconstrucción. En este trabajo el filtro utilizado ayudó bastante para estas pruebas.

En la estrategia de vuelo realizada por el dron, es decir, el seguimiento de la trayectoria circular siempre mirando hacía el centro se realizó bastante bien, con errores de seguimiento tolerables de acuerdo a la prueba, además el seguimiento es bastante suave e incremental para que el dron no pierda el mapa actual generado por ORBSLAM2 en tiempo real. Se comenta que en esta versión el dron no sabe cuando se ha perdido o no ve el mapa y se piensa implementar a futuro con el fin de que en el momento de que el dron detecte que se ha perdido del mapa y regrese hasta reubicarse y así pueda continuar con su recorrido.

Las reconstrucciones realizadas en estas pruebas se consideran buenas, ya que en la prueba del TMR solo se indica que la reconstrucción será sobre cajas con textura, lo cual es un elemento mucho mas fácil de describir que el automóvil utilizado en estas pruebas.

Referencias

Amenta, N., Bern, M. W., and Kamvysselis, M. (1998). A new voronoi-based surface reconstruction algorithm. *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*.

Amenta, N., Choi, S., and Kolluri, R. (2004). The power crust. *Computatinal Geometry*, 19.

Chen, L., Tang, W., John, N. W., Wan, T. R., and Zhang, J. J. (2018). Slam-based dense surface reconstruction in monocular minimally invasive surgery and its application to augmented reality. *Computer Methods and Programs in Biomedicine*, 158:135–146.

Johannes Meyer, S. K. (2022). Hector quadrotor. `https://github.com/RAFALAMAO/hector-quadrotor-noetic`.

Juárez, A. (2022). Hatchback mapping - drone (hector-quadrotor), orbslam2, ros (noetic). `https://youtu.be/7g_r4uIvRdc`.

Kazhdan, M. M. and Hoppe, H. (2013). Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3):29:1–29:13.

Kona, J. (2015). powercrust. `https://github.com/kubkon/powercrust.git`.

Li, Z., Zhao, J., Zhou, X., Wei, S., Li, P., and Shuang, F. (2022). Rtsdm: A real-time semantic dense mapping system for uavs. *Machines*, 10(4):285.

Mur-Artal, R. and Tardos, Juan D., A. J. (2022). Orbslam2-noetic. `https://github.com/RAFALAMAO/ORB_SLAM2_NOETIC`.

Mur-Artal, R. and Tardos, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262.

Yusefi, A., Durdu, A., and Sungur, C. (2020). Orb-slam-based 2d reconstruction of environment for indoor autonomous navigation of uavs. *Avrupa Bilim ve Teknoloji Dergisi*, pages 466–472.

Zhou, Q.-Y., Park, J., and Koltun, V. (2018). Open3D: A modern library for 3D data processing. *arXiv:1801.09847*.