Visual SLAM con cámara estéreo y RBG-D para exploración*

*Proyecto Final para el curso de Robótica Autónoma

Cabeza José
Dep. Ingeniería Mecatrónica
UTEC
jose.cabeza@utec.edu.pe

Carita Manuel
Dep. Ingeniería Mecatrónica
UTEC
manuel.carita@utec.edu.pe

Fabián Jim
Dep. Ingeniería Mecatrónica
UTEC
jim.fabian@utec.edu.pe

Del Río Alejandro
Dep. Ingeniería Mecatrónica
UTEC
alejandro.delrio@utec.edu.pe

***Abstract*—En este paper se presenta el uso del robot diferencial Turtlebot 3 para la exploración del laboratorio L201 de la UTEC. El objetivo es implementar un algoritmo de Visual SLAM para obtener información del entorno y conocer la posición y orientación del robot. Como resultados que *la queso*.**

***Index Terms*—Turtlebot 3, ROS, Visual SLAM, Python, cámara stereo, RGB-D, RTAB-map, LiDAR, navegación autónoma**

I. Introducción

Hoy en día está aumentando el uso de robots para exploraciones en zonas de alto riesgo para una persona. Drones o robots móviles como cuadrúpedos o diferenciales con ruedas son mayormente usados en la minería subterránea o en desastres naturales. Asimismo, se ha incrementado métodos donde no solo se explora de forma visual, sino una reconstrucción tridimensional del entorno. Aún más, se busca una autonomía que permite al robot movilizarse sin la necesidad de intervención humana. En este sentido, uno de los algoritmos más usados es Visual SLAM (Simultaneous Localization and Mapping), donde se conoce dónde está el robot y cómo se observa el entorno. Como hardware se tienen sensores exteroceptivos como cámaras, LiDAR, sonar, entre otros.

En la búsqueda de un método que pueda ser usado en distintos robots, se eligió el robot diferencial de ruedas Turtlebot 3. En esta oportunidad se comparará el uso de una cámara stereo y una cámara RGB-D usando visual SLAM. Del mismo modo, para la autonomía de este se usará un LiDAR 2D para evitar choques del robot.

II. Metodología

La metodología del proyecto consiste en la comparación de mapas obtenidos por el método de SLAM del RTAB-map utilizando diferentes sensores de entrada para obtener la información del entorno. Estos sensores son en un primer caso dos cámaras en configuración estereo y para el segundo caso un cámara de profundidad RGB-D Kinect con un sensor IMU. La explicación detallada de cada paso del proyecto se explica en los siguientes puntos.

A. Algoritmo de navegación con LIDAR 2D

El robot que se utiliza en este proyecto es el Turtlebot3 Waffle Pi. El primer paso de la metodología es establecer un algoritmo de navegación basado en las mediciones que realiza el LiDAR 2D. El diagrama de flujo de este algoritmo se detalla mejor en la figura 1,. La esencia es evitar colisionarse con obstáculos mientras navega por un entorno desconocido realizando el SLAM.

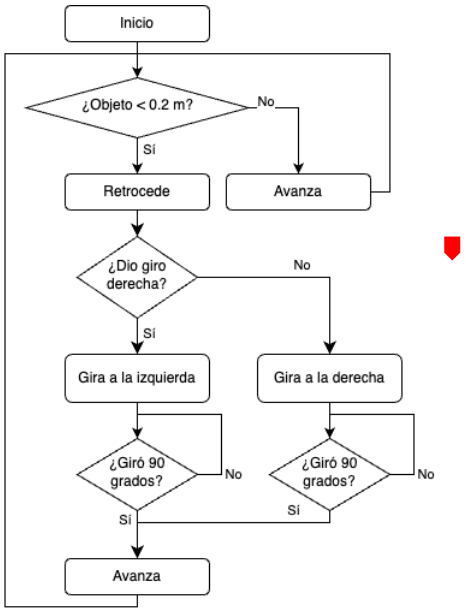


Fig. 1: Algoritmo de navegación autónoma con LIDAR 2D

B. Visual SLAM - Cámara Stereo

Para realizar la localización y el mapeo del robot en simultáneo (SLAM), el primer método de obtención del mapa del entorno es utilizando dos cámaras en configuración estéreo. Para ello se realiza inicialmente la calibración de las cámaras con una tabla de ajedrez en Gazebo. Seguidamente, con una librería, se toma como entradas la imagen de la cámara izquierda y derecha para poder obtener tanto la odometría de robot, es decir, su posición (x,y) y su orientación θ, como

una nube de puntos RGB-D del entorno del robot. Tomando la información del RGB-D generado por la cámara estéreo y el cálculo de la odometría, se usa la librería RTAB-map. Esta librería permite realizar el SLAM. Lo que se espera es poder tener un mapa en 3D del entorno del robot. Para su ejecución, el robot debe recorrer la mayor cantidad de espacio posible.

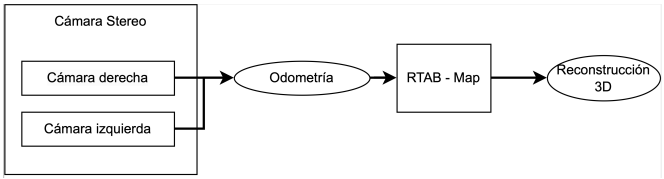


Fig. 2: Método de Visual SLAM con cámara stereo

C. Visual SLAM - Kinect+IMU

El segundo método para la obtención del mapa consiste en utilizar una cámara Kinect para obtener información RGB-D y un sensor IMU para estimar el *pose* (posición y orientación) del robot. El RTAB-map como se ha estado discutiendo necesita de la información del odom y del RGB-D. En el caso de la odometría se aplica el filtro de Kalman Extendido (EKF) para estimar el pose de robot teniendo como mediciones las aceleraciones lineales y la velocidad angular. Asimismo, la señal de control es la velocidad lineal y la angular, con ello y un modelo de movimiento basado en velocidad se aplica EKF. Este nodo publica en el tópico */odom* la información del pose. Con ello se aplica el RTAB-map para la obtención del mapa del entorno, movilizando al robot por el mismo escenario que en el caso anterior.

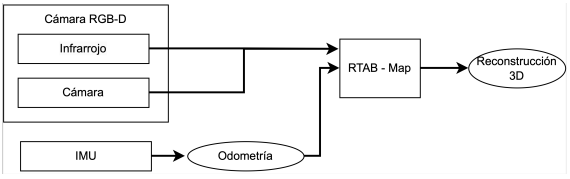


Fig. 3: Método de Visual SLAM con cámara RGB-D

El filtro de Kalman Extendido utiliza un modelo no lineal para realizar la estimación. Las ecuaciones que relacionan a la posición y orientación con la velocidad lineal y angular de manera no lineal se muestra a conitnuación.

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{v}{\omega}\left(-\sin\theta_{t-1} + \sin\left(\theta_{t-1} + \omega\Delta t\right)\right) \\ \frac{v}{\omega}\left(\cos\theta_{t-1} - \cos\left(\theta_{t-1} + \omega\Delta t\right)\right) \\ \omega\Delta t + \gamma\Delta t \end{bmatrix}$$

Fig. 4: Algoritmo del Filtro de Kalman Extendido

El algoritmo para realizar este filtro se muestra a continuación.

Algorithm Extended_Kalman_filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$**:**

$\bar{\mu}_t = g(u_t, \mu_{t-1})$

$\bar{\Sigma}_t = G_t\, \Sigma_{t-1}\, G_t^T + R_t$

$K_t = \bar{\Sigma}_t\, H_t^T (H_t\, \bar{\Sigma}_t\, H_t^T + Q_t)^{-1}$

$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$

$\Sigma_t = (I - K_t\, H_t)\, \bar{\Sigma}_t$

return μ_t, Σ_t

Fig. 5: Algoritmo del Filtro de Kalman Extendido

D. Método de comparación de mapas

Por último los mapas obtenidos se deben comparar o medir qué tan preciso fueron los mapas generados. Para ello, la idea consiste en comparar el mapa obtenido por alguno de los métodos con un mapa patrón, que sería el ideal o real. Para esta comparación, se trata a los mapas como imágenes 2D, donde un píxel de intensidad 255 (blanco), significa espacio libre y 0 (negro), obstáculo. En el caso de los mapas generados por lo métodos se debe realizar un procesamiento de imágenes para obtener esto. Estas imágenes se normalizan para que 255 sea 1. Estas imágenes se vectorizan para que no estén en forma de matrix. y lo que se compara es la correlación de estos vectores. Idealmente si un mapa es idéntico al patrón esta correlación debe ser 1. En la figura 6 se puede visualizar mejor este algoritmo de comparación.

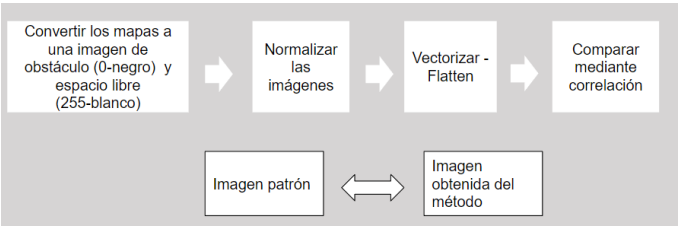


Fig. 6: Método de comparación de mapas

III. Resultados

Se realizó la calibración de las cámaras stereo simuladas en Gazebo a través del paquete *stereo_image_proc* como se muestra en 7. Esta calibración arroja una configuración que es usada por los tópicos de estimación de odometría visual del *stereo_image_proc*.

Se realizó el mapeo del mundo maze.world en simulación de Gazebo. En la figura 8 se observan representaciones en voxels de los mapas obtenidos con cada método. Para este caso el algoritmo de SLAM RGB-D y EKF tuvo resultados mucho más estables que el algoritmo de visión estéreo.

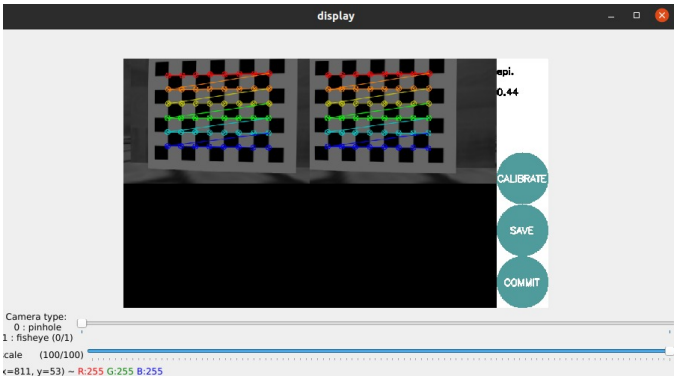
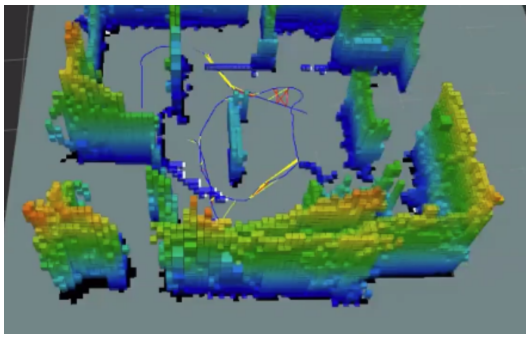
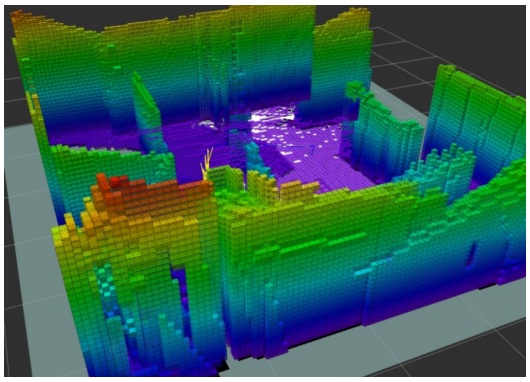


Fig. 7: Calibración de ambas cámaras del turtlebot3 en Gazebo



(a) Visión estéreo



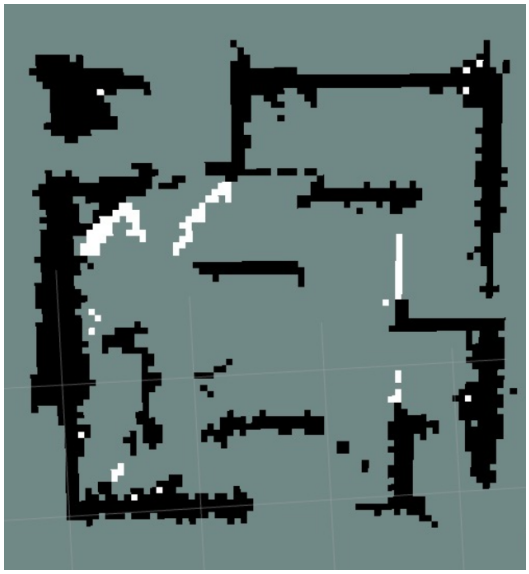
(b) RGB-D y EKF

Fig. 8: Comparación de mapa de voxels de ambos métodos.

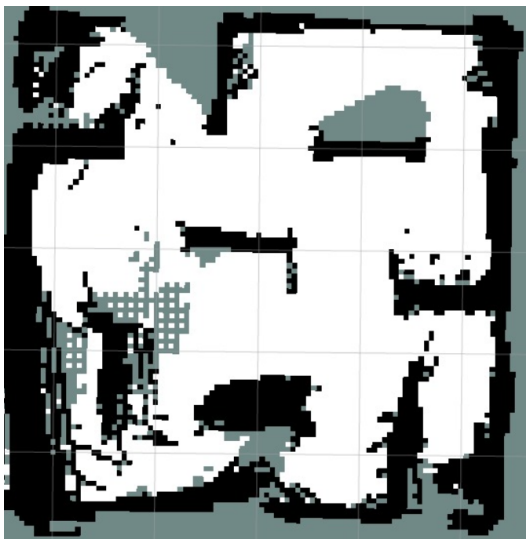
En la representación de voxels también se puede ver que la cámara RGB-D está detectando puntos en el piso. Esto no ocurre con visión estéreo, probablemente porque la textura del piso no es suficiente para aparecer en el *disparity map*. Se puede apreciar más claramente esta diferencia en la figura 9 que muestra ambos *occupancy grid* obtenidos con cada algoritmo.

Del método que se obtiene el mapa con el Kinect más el IMU, se realizó el desplzamiento del robot de manera lenta, de esta manera se lograba realizar mejor los matches de los features detectados por el algoritmo de SLAM con el Kinect. Se realizarón pruebas donde el Turtlebot3 se movía rapidamente por el mapa y lo que sucedió es que los matches no se realizaban correctamente y el mapa que se estaba obteniendo se terminaba desfasando. Por ello, se sugiere hacer sensor funsión para obtener la odometría. Por ejemplo se podría añadir encoders y con la cinemática directa se conocería la posición y orientación del robot.

Estos problemas que surgieron con la odometría obtenida del IMU, no sucedieron cuando se trabajo unicamente con las cámaras en configuración estéreo. La odometría obtenida era más precisa, pero le desventja radicaba a que de manera visual se podía distinguir que la nube de puntos obtenidas de la fusión de las dos cámaras tenía menor densidad y un ruido mayor en comparación al obtenido por el Kinect.



(a) Visión estéreo



(b) RGB-D y EKF

Fig. 9: Comparación de occupancy grid de ambos métodos.

Finalmente, para poder medir el rendimiento de cada al-

goritmo se usó el método descrito en la metodología para comparar el mapa *groud truth* mostraod en la figura 10 con los mapas obtenidos, ya normalizados y en 2 dimensiones solamente. Se aplicó correlación y *Mean Square Error* (MSE), cuyos resultados se muestran en la tabla.

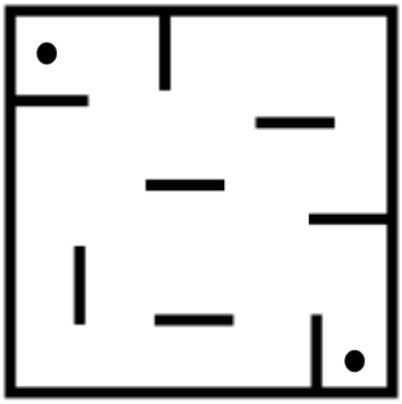


Fig. 10: Ground truth del mapa de maze world.

	RGB-D y EKF	Visión Estéreo
Correlación	0.1089	0.2157
MSE	0.7104	0.808

Es muy notorio que los coeficientes no son altos, lo que quiere decir que los mapas obtenidos por los métodos propuestos no son exactos y presentan mucho error. Sin embargo, la metodología para medir el error entre los mapas puede mejorar como el propuesto por [3] a través de una media de distancias de Nearest Neighbors y una alineación mediante Iterative Closest Point (ICP).

IV. Conclusiones

Ambos métodos de SLAM, utilizando cámara RGB-D y visión estéreo, lograron completar exitosamente la tarea de mapeo del mapa propuesto. Sin embargo, cada método tiene sus limitaciones. El método RGB-D tiene restricciones en cuanto a la distancia, ya que su rango de detección es limitado. Por otro lado, el método de visión estéreo requiere de superficies con texturas y puede ser afectado por las condiciones de iluminación. Como recomendaciones, se sugiere mejorar el algoritmo de navegación para evitar colisiones y optimizar los filtros utilizados para el IMU, con el fin de obtener una estimación más precisa del pose del robot. Estas mejoras contribuirán a obtener resultados más robustos y confiables en futuras aplicaciones de mapeo 3D con Visual SLAM.

V. Propuestas de mejora

Se recomienda realizar sensor fusión para la estimación de la odometría en el caso de se utilizaba el Kinect para obtener la nube de puntos. Los sensores que se usarían para fusionar los datos serían el IMU con encoders.

Otra recomendación para este mismo caso de la odometría, es probar la implementación de un filtro de partículas y compararlo con la odometría obtenida aplicando el filtro extendido de Kalman.

Por último, se sugiere realizar la implementación en un ambiente interior planificado, para que el método de comparación que se utilizó para los mapas, también pueda se utilizado. Es decir, replicar un entorno físico con obstáculo fáciles de representar.

VI. Anexos

El código utilizado se subió a un repositorio en GitHub. El link de acceso es el siguiente: https://github.com/Jimi1811/Visual_SLAM_in_turtlebot3.
En el mismo se detalla los procesos a seguir para reproducir lo presentado en este documento, así como también las evidencias de la simulación presentada.

References

[1] Filatov, A., Filatov, A., Krinkin, K., Chen, B., y Molodan, D. (2017). 2D SLAM Quality Evaluation Methods. https://doi.org/10.48550/arXiv.1708.02354

[2] J. Funke and T. Pietzsch, “A framework for evaluating visual slam,” in Proceedings of the British Machine Vision Conference (BMVC), vol. 6, 2009.

[3] N. Otsu, “A threshold selection method from gray-level histograms,”IEEE transactions on systems, man, and cybernetics, vol. 9, no. 1, pp. 62–66, 1979.

[4] Siegwart, R., Nourbakhsh, I. R., y Scaramuzza, D. (2011). Introduction to Autonomous Mobile Robots, second edition. MIT Press.

[5] F. Anton, F. Ayrtom, K. Kirill, C. Baian, D. Molodan. “2D SLAM Quality Evaluation Methods”. (2017). From https://arxiv.org/abs/1708.02354