Universidad de Ingeniería Y Tecnología

Departamento de Ingeniería Mecatrónica

Robótica Autónoma

Proyecto Final

Integrantes

Álvarez Concha, Manuel Álvaro

Escalante Ccoyllo, Sebastian Amaru

Palma Rodríguez, Diego Alonso V.

Profesor: Ph.D. Oscar E. Ramos

Lima - Perú

2021-1

1. Introducción

De acuerdo con la Federación Internacional de Robótica (IFR) [1], un robot de servicio es un robot que opera de forma semiautónoma o totalmente autónoma para realizar servicios útiles para el bienestar de las personas, excluyendo las operaciones de fabricación. En los últimos años, la robótica ha experimentado un gran crecimiento en el sector de servicio, y se espera que crezca a tasas mucho más altas que los robots industriales [2]. En base a ello, resulta de nuestro interés proponer un robot de servicio personal (*home robots*). Algunos ejemplos de este tipo de robots son Ubtech Lynx, Asus Zenbo or Aido.

Con ello en mente, el presente trabajo presenta las bases del diseño y programación de un robot de servicio, llamado ARCA, el cual es un robot diferencial que cuenta con los siguientes sensores: un LiDAR, una Cámara RGB, un IMU y encoders en sus ruedas. En base a la data recolectada por sus sensores, ARCA es capaz de generar el mapa 2D de un entorno simulado en Gazebo, localizarse y desplazarse de manera autónoma a un punto deseado indicado por el usuario. Asimismo, ARCA puede reconocer objetos y personas por medio de las imágenes proporcionadas por su cámara. Y, en caso de reconocer a una persona, es capaz de realizar un saludo. Es importante mencionar que para realizar todas las tareas se utilizaron las plataformas de ROS, Gazebo, Rviz y OpenCV.

Este informe se encuentra dividido en tres secciones. En la sección 2 se presenta el desarrollo del proyecto, el cual se subdivide en los pasos más importantes que se llevaron acabo en el presente proyecto; asimismo, se explica la metodología empleada en cada uno de ellos, los resultados y discusiones del mismo. Finalmente, en la sección 3, se realizan algunas conclusiones y los siguientes pasos del proyecto.

2. Desarrollo del proyecto

En la presente sección se evidencian los pasos que se siguieron para implementar cada una de las tareas del robot. Esta sección se subdivide principalmente en 6 subsecciones: URDF, Entorno del robot, Localización y mapeo simultáneo (SLAM), Navegación Autónoma, Reconocimiento de objetos y Movimiento de brazos.

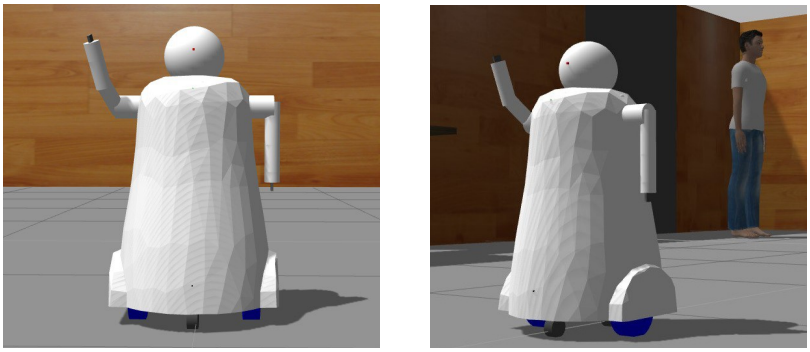
2.1. URDF

Si bien es cierto el presente proyecto se enfoca más en la navegación autónoma de ARCA, es importante mencionar cuáles son sus componentes y describir, de modo general, su estructura mecánica. ARCA es un robot diferencial que cuenta con dos ruedas motrices y dos ruedas del tipo caster, las cuales forman la base del robot. Sobre esta se encuentra la estructura mecánica del robot la cual representa el cuerpo del mismo. En una posible implementación esta estructura contiene los microcontroladores, *drivers*, sensores del robot y el cableado. Además de ello, ARCA cuenta con dos brazos de 5 gdl que se ubican a los costados de su estructura mecánica. Finalmente, se ubica la cabeza en la parte superior, la cual cuenta con 2 gdl (cuello y cabeza).

En base a ello, se elabora el URDF de arca con el fin de poder visualizar y simular al robot

en Rviz y Gazebo, respectivamente. Para ello, se utilizaron las figuras geométricas básicas que proporciona Gazebo, tales como cilindros y esferas. Asimismo, para la estructura mecánica se utiliza un *mesh* con el fin de que se obtenga una apariencia más real. Además de ello, por simplicidad, se modelaron los sensores del robot como cubos. Por otro lado, es importante mencionar que para este proyecto no se modeló el gripper (efector final) de ARCA, por lo que tareas como *pick-and-place* no son posibles de realizarse con el URDF desarrollado.

En la figura 1 se puede observar el URDF de ARCA en el entorno de Gazebo y tiene una altura de 70 cm, 30 cm de ancho y 25 cm de largo (aprox.).



(a) Vista frontal de ARCA (b) Vista lateral de ARCA

Figura 1: URDF de ARCA en Gazebo

2.2. Entorno del robot

Para el presente proyecto se realizó el diseño de un *world* en Gazebo que contiene los modelos básicos para simular un entorno de casa o interiores. En base a ello, se utilizaron como modelos dos personas, una mesa de comedor, una mesa de estar y latas de Coca-Cola, además de muros que simulan el entorno de una casa. Es importante mencionar que no se utilizaron modelos adicionales, debido a la poca capacidad de procesamiento de los dispositivos utilizados y, dado que se utilizaran algoritmos que requieren un alto costo computacional, se decidió no sobrecargar el entorno con modelos adicionales. El entorno se visualiza en la Figura 2, donde se puede observar los modelos mencionados anteriormente.

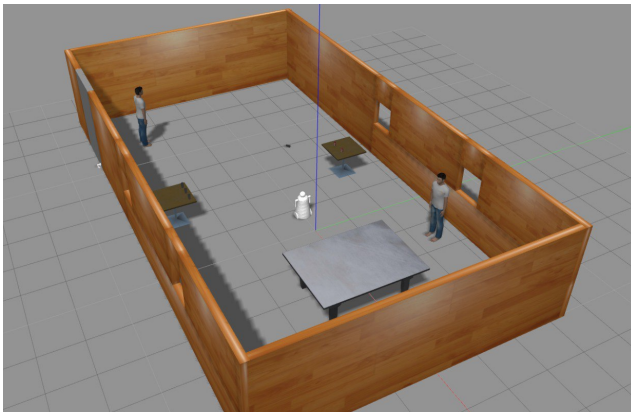
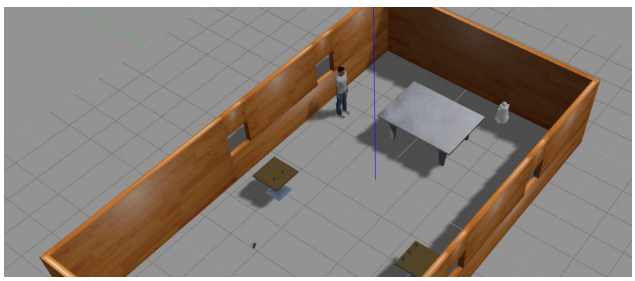


Figura 2: Entorno de ARCA (*world* de Gazebo)

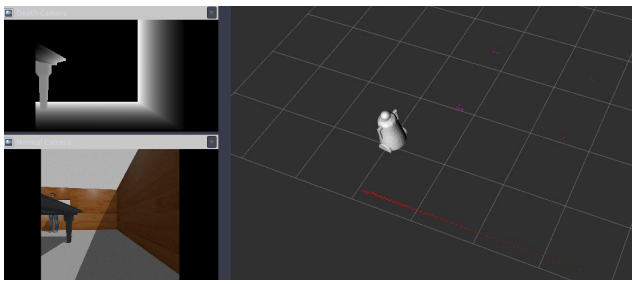
2.3. Localización y mapeo simultáneo (SLAM)

Uno de los pasos más importantes para que ARCA pueda desarrollarse de forma autónoma es que sea capaz de localizarse en su entorno y también obtener un mapa del mismo, por lo que es necesario realizar un algoritmo de SLAM. Debido a que ARCA es un robot diferencial y cuenta con un LiDAR, es posible utilizar el paquete `gmapping` de ROS. Este paquete proporciona un algoritmo de SLAM basado en los datos de un LiDAR (láser) y la *pose* del robot, con lo que se puede crear un mapa de celdas de ocupación 2D a partir de dichos datos, mientras se localiza simultáneamnte al robot móvil dentro del mapa generado.

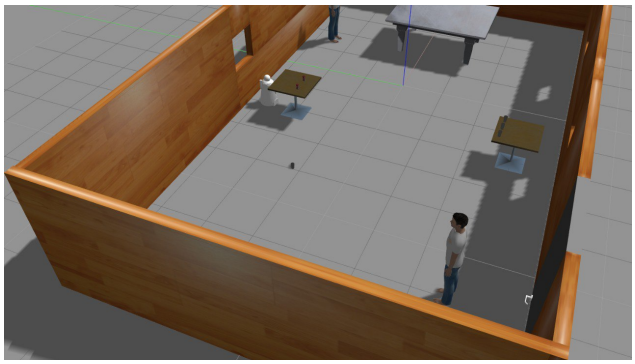
En la Figura 3 se muestran algunas capturas de pantalla de ARCA mientras realiza el mapeo del entorno. Es importante mencionar que para realizar el mapeo del entorno se teleoperó a ARCA enviando las velocidades globales deseadas. Dicha teleoperación se realizó de manera lenta y continua ($v_{max} = 0{,}1$ y $w_{max} = 0{,}1$), pues se requería reducir los errores de odometría al mínimo, de lo contrario se obtendría un mapa completamente erróneo. Teniendo en cuenta ello es posible generar el mapa del entorno, el cual se visualiza en la figura 4.



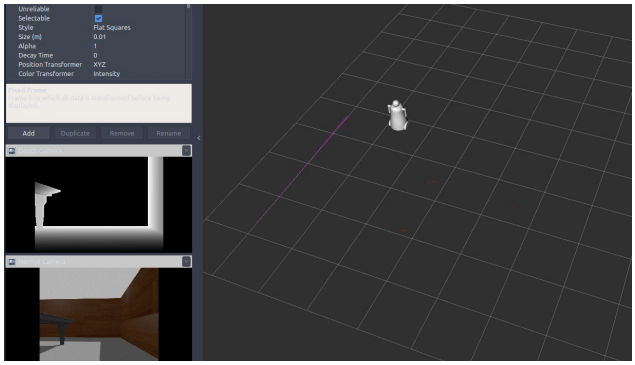
(a) Mapeo del ancho del mapa en Gazebo



(b) Mapeo del ancho del mapa en Rviz



(c) Mapeo del largo del mapa en Gazebo



(d) Mapeo del largo del mapa en Rviz

Figura 3: Mapeo del entorno de ARCA

2.4. Navegación autónoma

Una vez obtenido el mapa del entorno es posible realizar la navegación del robot de un punto deseado a otro. Para ello es necesario combinar el mapa obtenido con la odometría del robot y las mediciones en tiempo real para evadir nuevos posibles obstáculos de su entorno. En ese sentido, se utilizó el Algoritmo de Localización adaptativo de Monte Carlo (AMCL), el cual se implementa mediante el paquete `amcl` de ROS. Este paquete es un sistema de

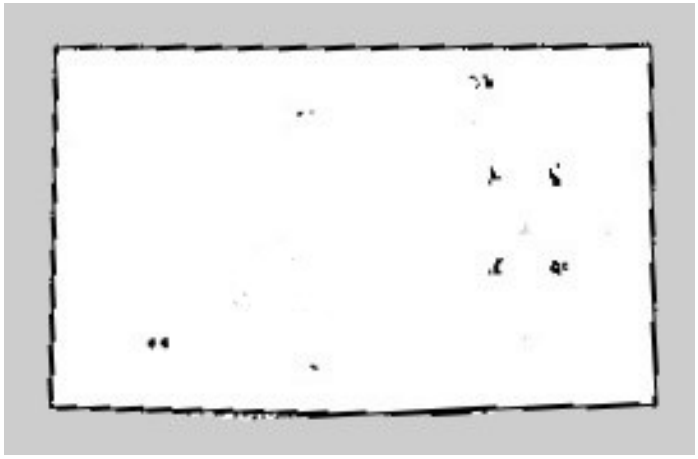
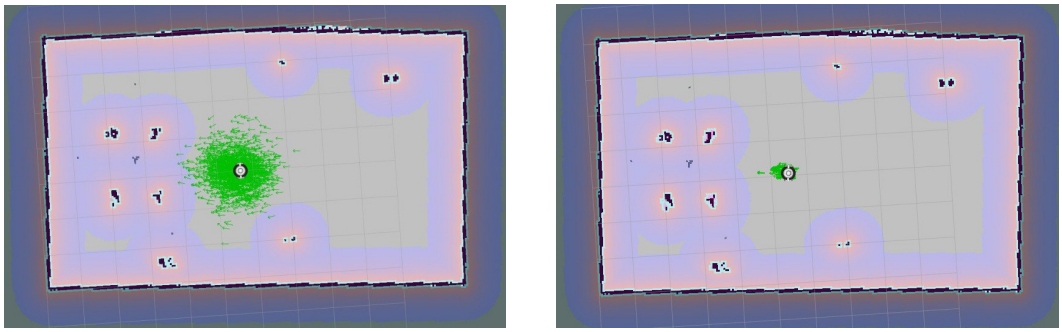


Figura 4: Mapa del entorno de Gazebo

localización probabilística que utiliza un un filtro de partículas para rastrear la *pose* de un robot en base a un mapa ya conocido.

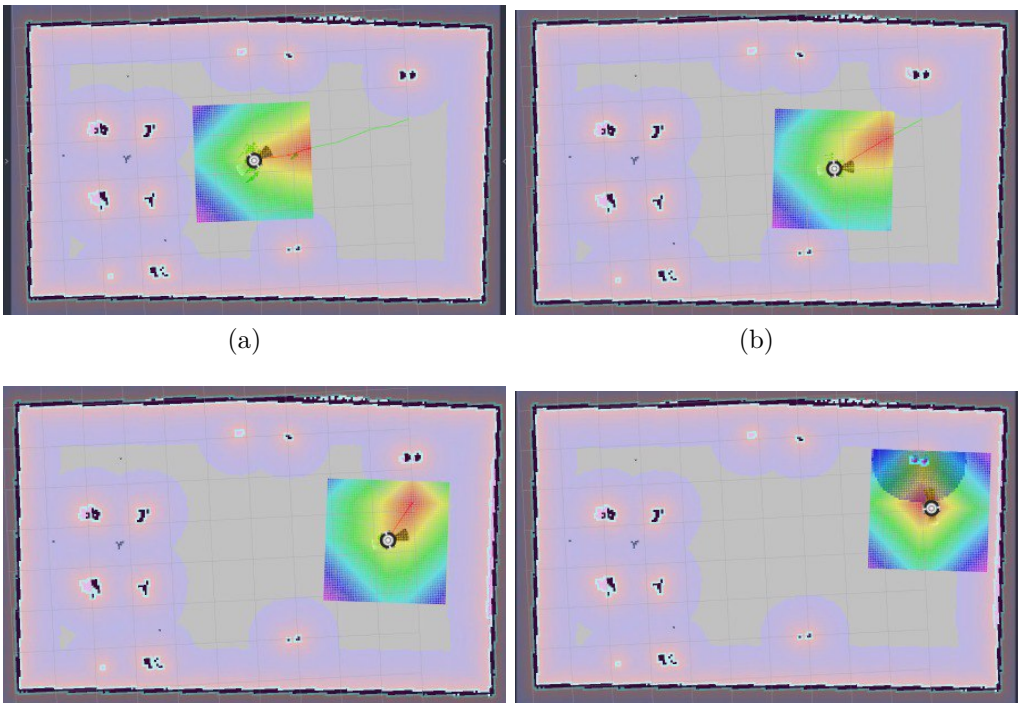
En la Figura 5 se puede ver la implementación del paquete `amcl` en Rviz. Como se puede observar, al inicio la cantidad de particulas es muy grande y dispersa (Figura 5a), pues no se conoce con certeza la posición del robot. Por ello, es necesario en un inicio que el robot se desplace por un pequeño periodo de tiempo para que la cantidad de partículas se reduzca (Figura 5b).



(a) Inicio del AMCL (b) AMCL después de la localización

Figura 5: AMCL

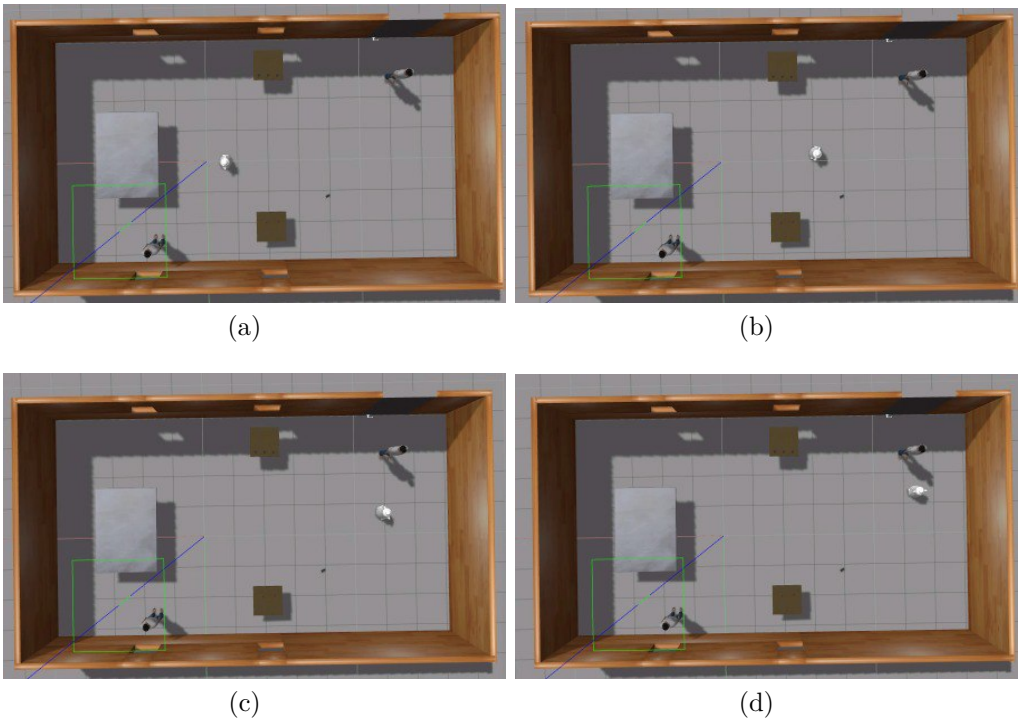
Luego de ello, se define el punto y orientación deseado del robot y el algoritmo nos brinda una trayectoria hacia el mismo. Para ello, brinda las velocidades articulares de los motores de las ruedas para llegar al objetivo deseado. Entonces, el robot realiza su recorrido tanto en Rviz como en Gazebo. En las figuras 6a-6d se muestra la visualización en Rviz de la navegación autónoma de ARCA. Asimismmo, en las figuras 7a-7d se muestra su simulación en Gazebo.



(a) (b)

(c) (d)

Figura 6: Visualización de la navegación autónoma de ARCA (a-b-c-d)



(a) (b)

(c) (d)

Figura 7: Simulación de la navegación autónoma de ARCA (a-b-c-d)

2.5. Reconocimiento de objetos y personas

El reconocimiento de objetos y personas es crucial para que un robot de servicio como ARCA pueda hacer efectivo su propósito en el entorno al que se le asigne y que pueda interactuar con los objetos y personas que deba asistir. Por ello, es necesario utilizar un algoritmo de detección y reconocimiento de objetos. El algoritmo YOLO [3], muestra una gran facilidad para realizar esta tarea, además de ser uno de los métodos más usados en la detección de objetos en tiempo real en la actualidad. Además de ello, se utilizó un filtro de color para identificar objetos que no se entrenaron en el algoritmo YOLO, como las latas de gaseosa. A continuación, se detallan algunas especificaciones de dichos algoritmos y se expone su uso en este proyecto.

2.5.1. YOLO

YOLO es una propuesta de solución al problema de detección de objetos publicada el 2015. Básicamente propone un pipeline de procesos para detección de objetos en una sola red, que puede ser optimizada de inicio a fin directamente en el rendimiento de detección. La arquitectura presentada es altamente rápida, con una capacidad de procesar 45 cuadros por segundo en tiempo real por medio del modelo base, y hasta 155 cuadros por segundo con una versión más simple (Fast YOLO). El algoritmo sacrifica precisión en localización para reducir la probabilidad de predecir falsos positivos en el fondo de la imagen. Una nueva versión de YOLO, YOLOv3 [4], muestra mejoras en el algoritmo. La red es un poco más grande que en anteriores versiones pero tiene una mayor precisión. Dado que lo que se desea de la visión implementada en un robot de servicio es que sea capaz de diferenciar correctamente los objetos y las personas con las que deba interactuar, la precisión y facilidad de implementación de YOLOv3 nos permite tener una buena identificación de los objetos que se desean evaluar en el entorno y circunstancias propuestas.

La implementación de YOLOv3 para la identificación de objetos del ARCA fue hecha en python y está basada en la arquitectura de red de Darknet, la cual puede ser importada al instalar la librería Darknet. También se pueden cargar inmediatamente las clases de objetos y los pesos obtenidos en el preentrenamiento de la red. Para obtener las imágenes es necesario utilizar un nodo de cámara y a su vez suscribirse a un tópico de la cámara RGB. También se utiliza un tópico para publicar las imágenes de salida y otro para publicar si se detectó a una persona. Un resultado de la simulación usando estas configuraciones con el Turtlebot en Gazebo se puede ver en la Figura 8 y la implementación en el entorno creado y usando el robot ARCA se puede ver en las Figuras 9a y 9b. Como resultado, ARCA puede distinguir objetos y personas con facilidad en el entorno simulado.

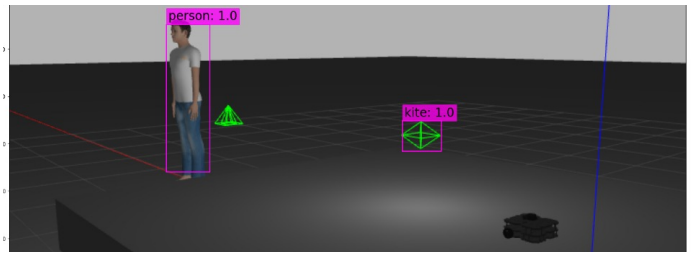


Figura 8: Test de YOLO en Gazebo usando el Turtlebot

(a) Reconocimiento de mesa y persona (b) Reconocimiento de persona

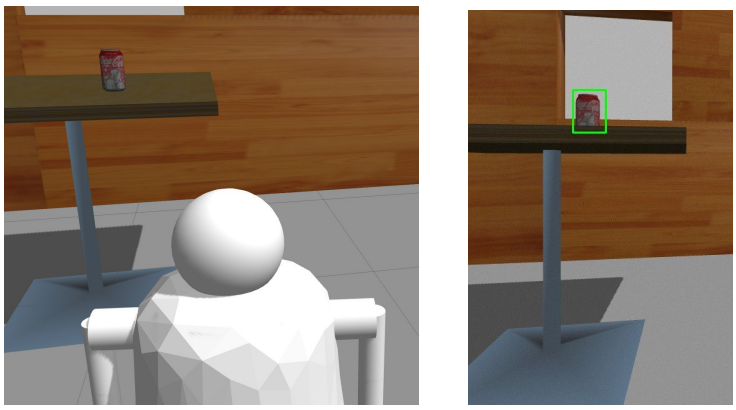
Figura 9: Aplicación de YOLO en ARCA dentro del entorno simulado

2.5.2. Segmentación de color

Para detectar objetos como las latas de gaseosa se aprovecha el hecho de que estas son, en su mayoría, de color rojo a lo largo de su forma cilíndrica. En ese sentido, se emplea un filtro de color para poder identificarlas en base a la información proporcionada por la cámara.

Este algoritmo fue realizado con OpenCV y Python, y consiste básicamente en convertir la imagen BGR a HSV. Luego, se hace uso del espacio Hue (H) para definir los umbrales de filtraje, pues se sabe que este espacio está asociado con el color dominante que una persona percibe con sus ojos. Además de ello, se sabe que se quiere filtrar el color rojo, por lo que los umbrales utilizados son: $(170, 100, 100)$ y $(179, 255, 255)$. Luego de ello, se requiere realizar la detección del contorno de los objetos que tengan el color rojo como dominante (latas). Sin embargo, el resultado obtenido no siempre mostraba un contorno cerrado, por lo que fue necesario realizar una opeación morfológica de dilatación. Para ello, se utilizó un kernel rectangular de 20 x 20 px. Finalmente, se determinan los contornos resultantes y se dibujan los rectángulos delimitadores solo si la altura del objeto identificado es mayor a su ancho. Dicha condición es algo que siempre se cumple en las latas, por lo qeu si se detecta lo contrario probablemente sea ruido del entorno.

En la Figura 10 se muestra el resultado de implementar el filtro de color en la simulación en Gazebo.



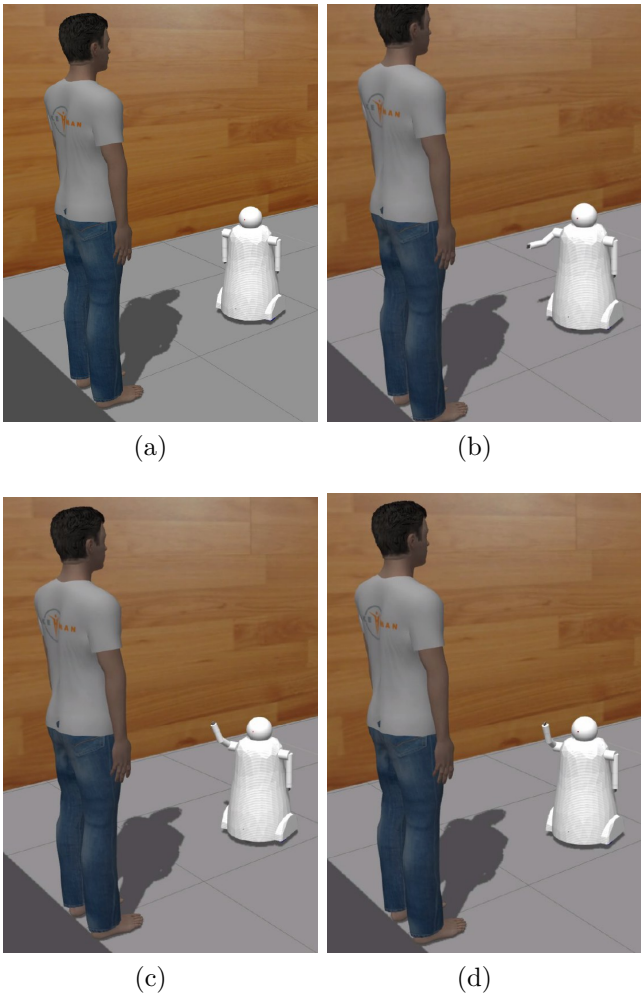
(a) Vista del usuario (b) Vista del robot

Figura 10: Implementación del filtro de color en simulación

2.6. Movimiento de brazos

Como se mencionó anteriormente, el URDF de ARCA utilizado en el presente proyecto no contempla el efector final de los brazos de ARCA, por lo que no esposible realizar tareas como *pick-and-place* por el momento. Sin embargo, es posible accionar los brazos del robot de tal manera que realice algún movimiento que se requiera. En este caso en particular se realiza el movimiento de los brazos de ARCA para realizar un saludo al momento de estar frente a una persona.

Para lograr el movimiento de los brazos se utilizaron controladores PID para la posición angular de cada una de sus articulaciones. Asimismo, se utilizó el paquete `controller_manager` para realizar la simulación en Gazebo. En las figuras 11a-11d se muestran capturas de pantalla del saludo. A continuación se describe cada una de estas etapas.



(a) (b)

(c) (d)

Figura 11: Movimiento de brazos: Saludo de ARCA (a-b-c-d)

3. Conclusiones

1. En el presente proyecto se han desarrollado las bases de la autonomía de un robot de servicio: ARCA. Para lograr ello se elaboró el URDF del robot (desde cero), se diseño un entorno de interiores en Gazebo, se realizó la localización del robot y el mapeo simultáneo de su entorno (SLAM), así como la navegación autónoma hacia cualquier punto deseado. Además de ello, ARCA es capaz de detectar objetos y personas mediante el algoritmo YOLOv3 y filtros de color, lo cual es una tarea fundamental para un robot de servicio. Finalmente, ARCA es capaz de realizar movimientos con sus brazos, como el de un saludo al reconocer una persona.

2. Al momento de realizar el mapeo del entorno del robot la velocidad lineal y angular del robot deben ser limitadas a $0{,}1\frac{m}{s}$ y $0{,}1\frac{rad}{s}$, respectivamente, con el fin de obtener un mapa 2D preciso del entorno del robot; de lo contrario, se obtendrá un mapa de diferentes dimensiones o con un giro no deseado. En base al mapa 2D es posible realizar la navegación autónoma del robot. Esto nos indica que se debe realizar un mapeo de forma constante en caso el entorno del robot presente cambios o se agreguen obstáculos. Sin embargo, el algoritmo AMCL puede prevenir esto, es decir, evita que el robot choque con obstáculos no identificados en el mapa, pero no los actualiza para realizar un nuevo planeamiento de la trayectoria.

3. El algoritmo YOLOv3 para el reconocimiento de objetos y personas es una buena alternativa para un robot de servicio ya que este ha sido entrenado en el COCO Dataset [5] una librería de imágenes de objetos comunes, la cual va de la mano con la aplicación de un robot de servicio, el cual tiene como objetivo interactuar con humanos en un entorno que debe ser familiar e interpretable por el. Sin embargo, se debe tomar en cuenta que este modelo es bastante pesado, y dependiendo de la tarjeta que se use para su implementación se deben realizar optimizaciones, como reducir la cantidades de inferencia por segundo y solo utilizarlo despues de cierto intervalo de tiempo para actualizar la posición del objeto.

4. El siguiente paso del proyecto es implementar las simulaciones realizadas en el robot real y verificar que todo se cumpla a cabalidad, o en todo caso realizar las correcciones correspondientes. Además de ello, se debe actualizar el URDF de Arca con un gripper en su efector final, de tal modo que ARCA pueda realizar tareas adicionales como *pick-and-place* de objetos.

Referencias

[1] M. Ceccarelli, *Service Robots and Robotics: Design and Application*. Premier Reference Source, Italy: IGI Global, 1 ed., 2012.

[2] A. Resources and I. Insights, "Industry trends and market potential. what's next?," 2021.

[3] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[4] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.

[5] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.