Artículo Científico / Scientific Paper

UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR
INGENIUS
Revista de Ciencia y Tecnología
https://doi.org/10.17163/ings.n32.2024.08
pISSN: 1390-650X / eISSN: 1390-860X

Localización de robots basada en red neural utilizando características visuales

Neural network-based robot localization using visual features

Felipe Trujillo-Romero[1]

Recibido: 16-11-2023, Recibido tras revisión: 29-05-2024, Aceptado: 12-06-2024, Publicado: 01-07-2024

Resumen

Este artículo presenta el desarrollo de un módulo que puede desarrollar un algoritmo de construcción de mapas mediante odometría inercial y características visuales. Utiliza un módulo de reconocimiento de objetos basado en características locales y redes neuronales artificiales no supervisadas para conocer elementos no dinámicos en una habitación y asignarles una posición. El mapa está representado por una red neuronal donde cada neurona corresponde a una posición absoluta en la habitación. Una vez construido el mapa, basta con capturar un par de imágenes del entorno para estimar la ubicación del robot. Los experimentos se realizaron mediante simulación y utilizando un robot real. Se utilizó el entorno Webots con el robot humanoide virtual NAO para realizar las simulaciones. Al mismo tiempo, se obtuvieron resultados utilizando un robot NAO real en un escenario con diversos objetos. Los resultados muestran una buena precisión en la localización dentro de los mapas bidimensionales de ±(0,06, 0,1)m en simulación en contraste con el entorno natural; el mejor valor obtenido fue ±(0,25, 0,16)m.

Palabras clave: características visuales, mapas bidimensionales, odometría inercial, robot humanoide NAO, descriptor A-KAZE, estructura de crecimiento celular

Abstract

This paper outlines the development of a module capable of constructing a map-building algorithm using inertial odometry and visual features. It incorporates an object recognition module that leverages local features and unsupervised artificial neural networks to identify non-dynamic elements in a room and assign them positions. The map is modeled using a neural network, where each neuron corresponds to an absolute position in the room. Once the map is constructed, capturing just a couple of images of the environment is sufficient to estimate the robot's location. The experiments were conducted using both simulation and a real robot. The Webots environment with the virtual humanoid robot NAO was used for the simulations. Concurrently, results were obtained using a real NAO robot in a setting with various objects. The results demonstrate notable precision in localization within the two-dimensional maps, achieving an accuracy of ± (0.06, 0.1) m in simulations contrasted with the natural environment, where the best value achieved was ± (0.25, 0.16) m.

Keywords: Visual Features, Bidimensional Maps, Inertial Odometry, Humanoid Robot NAO, A-KAZE descriptor, Growing Cell Structure

[1,*]Departamento de Ingeniería Electrónica, DICIS, Universidad de Guanajuato, México.
Autor para correspondencia ✉: fdj.trujillo@ugto.mx.

Forma sugerida de citación: Trujillo-Romero, F. "Localización de robots basada en red neural utilizando características visuales," *Ingenius, Revista de Ciencia y Tecnología*, N.° 32, pp. 77-89, 2024. DOI: https://doi.org/10.17163/ings.n32.2024.08.

1. Introducción

Según la Federación Internacional de Robótica (IFR) [1], un robot de servicio es un sistema robótico que opera total o parcialmente de forma autónoma para realizar servicios valiosos para el bienestar de los humanos y el equipo, excluyendo las operaciones de manufactura. Los robots de servicio están diseñados específicamente para entornos humanos, como hogares, hospitales y restaurantes, lo que les exige tomar decisiones complejas. Esto incluye identificar, detectar, reconocer y manipular diversos objetos dentro de su entorno.

Para que un robot de servicio opere de manera autónoma, debe estar equipado con un sistema de control que le permita interactuar con su entorno para tomar las decisiones correctas y lograr objetivos específicos. Un componente crítico de este sistema de control para los robots de servicio implica aprender sobre el entorno en el que operarán. Inicialmente, el robot debe familiarizarse con la ubicación y los elementos no dinámicos con los que interactuará. Por ejemplo, en algunas competiciones, se da a los participantes un período para familiarizarse con las interacciones del escenario y realizar las calibraciones necesarias para completar las tareas.

La capacidad de ver mejora su interacción con las personas y el entorno; por ejemplo, se utilizan sensores de visión para la localización y el mapeo [2]. Se desarrolla un sistema de estereovisión para detectar objetivos a partir del mapa de profundidad generado [3]. Además, Scona et al. [4] utilizaron un sensor de estereovisión para explorar desafíos como el desenfoque por movimiento, la falta de características visuales, los cambios de iluminación y el movimiento rápido.

En la localización ambiental por robots móviles, se implementó un sistema de visión para desarrollar algoritmos de localización y mapeo simultáneos (SLAM) [5]. En [6] se desarrolló una aplicación para mapeo topológico y navegación utilizando SLAM visual. Ovalle-Magallanes et al. [7] utilizaron información visual para crear un sistema de localización basado en la apariencia para un robot humanoide. Lasguignes et al. [8] implementaron un sistema de localización apoyado en ICP, utilizando información visual en el robot humanoide TALOS. Por el contrario, Wozniak et al. [9] propusieron un algoritmo para el reconocimiento visual de lugares utilizando imágenes adquiridas por un robot humanoide, con una red neuronal como reconocedor. También se desarrolló un SLAM elipsoidal basado en visión de hitos aumentados en un robot humanoide NAO para escenarios interiores [10]. Además, se implementó un método para SLAM eficiente utilizando un sensor de visión monocular con vista frontal [11].

Además de las cámaras RGB, se emplean otros sensores; en [12], se utilizó un sensor IMU para localizar un robot humanoide en el entorno. En [13] se implementó una combinación de LiDAR 2D y odometría para permitir que un robot navegue y se localice. Wen *et al.* [14] presentaron un EKF-SLAM, utilizando sensores de cámara y láser para la localización y el mapeo en interiores. Un SLAM, basado en visión, permite a un robot móvil navegar en entornos desconocidos [15]. En [16] se propone un sistema SLAM para estimar las poses del robot y construir un mapa tridimensional del entorno. Además, se combinó el seguimiento basado en características de un sensor de visión estereoscópica para obtener un SLAM híbrido [17].

Mientras tanto, Cheng *et al.* [18] utilizaron puntos característicos para desarrollar un método que integra el flujo óptico con ORB-SLAM para diferenciar entre elementos dinámicos y estáticos. Ganesan et al. [19] propusieron un método para reducir el espacio de búsqueda para el algoritmo RRT* en tareas de planificación de rutas. La coincidencia de características para algoritmos de construcción de mapas se exploró utilizando la distancia de una nube de puntos obtenida de un sensor de alcance [20]. Se construye un mapa del entorno utilizando una fusión de sensores de odometría, láser 2D y RGB-D [21]. Una propuesta donde el entorno se representa mediante polígonos 3D que permiten a un robot localizarse se presenta en [22]. En contraste, se propuso un sistema de navegación topológica basado en representación simbólica en [23] para un robot humanoide.

Todos los trabajos mencionados anteriormente emplean técnicas para mejorar la localización, el mapeo o la búsqueda de objetos dentro de un entorno humano, llevadas a cabo por un robot móvil. Por esta razón, se utiliza el robot humanoide NAO [24] como plataforma para implementar la localización y el mapeo en este estudio.

En la sección 2 se describe los diversos métodos y materiales utilizados en este estudio. Posteriormente, en la sección 2.5 se presenta la implementación del sistema propuesto. Los resultados obtenidos con ambas plataformas se detallan en la sección 3. Finalmente, la sección 4 se discute las conclusiones y las direcciones para futuras investigaciones.

2. Materiales y métodos

2.1. Robot NAO

El robot NAO, como se muestra en la Figura 1(a), es la plataforma robótica principal elegida para implementar el sistema desarrollado. NAO, un robot autónomo y programable de altura media [24], es ampliamente reconocido como uno de los robots más sofisticados y completos del mercado. A lo largo de los años, se han introducido cinco versiones, cada una incorporando mejoras específicas, mientras que el concepto fundamental permanece inalterado.

La Figura 1(b) presenta un esquema del robot, indicando sus dimensiones, incluyendo altura, ancho y

longitud de los brazos. El robot NAO está equipado con el software integrado NAOqi, que le proporciona autonomía. NAOqi está integrado en el sistema operativo del robot, OpenNAO, una distribución GNU/Linux embebida basada en Gentoo. Este sistema incluye numerosas bibliotecas y programas esenciales para NAOqi. Una característica notable es la capacidad de ejecutar copias de NAOqi en una computadora, lo que facilita el uso de robots virtuales.

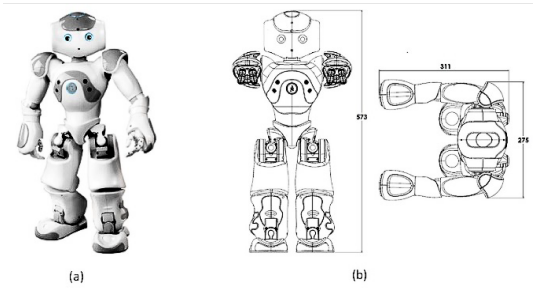

Figura 1. NAO (a) Robot, y (b) dimensiones en mm [25]

2.2. A-KAZE descriptor

El método A-KAZE [26], representado en la Figura 2, se divide en tres tareas principales: (1) construcción de un espacio de escala no lineal, (2) detección de características y (3) descripción de características. La construcción del espacio de escala no lineal implica procesar una imagen de entrada utilizando el método numérico de difusión rápida explícita (FED) [25], aplicado con un enfoque piramidal.

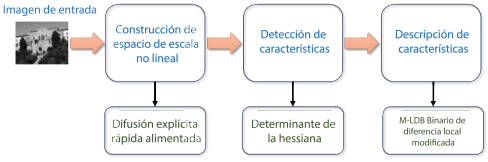

Figura 2. Descripción general del algoritmo A-KAZE

Inicialmente, el espacio de escala se discretiza en una serie de O octavas y S subniveles, identificados por índices discretos (o y s, respectivamente). Posteriormente, se mapean a su escala correspondiente, σ, utilizando la ecuación (1).

$$\sigma(o,s) = 2^{\frac{o+s}{s}} \tag{1}$$

La imagen de entrada se convoluciona con una desviación estándar gaussiana $\sigma 0$ para reducir el ruido y los posibles artefactos, considerando tanto la imagen de entrada como un factor de contraste λ, que es calculado automáticamente por el algoritmo. Posteriormente, se detectan características 2D de interés que exhiben un determinante de escala normalizada de la respuesta hessiana a través del espacio de escala no lineal para cada imagen filtrada. La normalización se realiza utilizando un factor que tiene en cuenta la escala de cada imagen en el espacio de escala no lineal, como se ilustra en la ecuación (2).

$$L^i_{Hessiana} = \sigma^2_{i,norm}(L^i_{xx}L^i_{yy} - L^i_{xy}L^i_{yx}) \tag{2}$$

El filtro concatenado de Scharr [27] calcula derivadas de segundo orden para aproximar la invariancia rotacional. Inicialmente, se obtiene la respuesta máxima del detector en una ubicación espacial específica para estimar la posición 2D del punto clave. Esto se logra ajustando una función cuadrática a la respuesta máxima del determinante de Hessian dentro de un vecindario de 3×3.

Finalmente, la orientación principal del punto clave se calcula utilizando el descriptor modificado-binario de diferencia local (M-LDB) [28]. Este método utiliza información sobre gradientes e intensidad del espacio de escala no lineal para generar un vector descriptor de longitud 64.

En el caso del descriptor utilizado, su principal ventaja es su rendimiento superior en la obtención de información visual al implementar el sistema de mapeo, debido a su invariancia a los cambios de escala y rotación. Además, opera más rápido que otros descriptores y el autor del algoritmo proporciona el código. Entre las desventajas, es necesario mencionar que se requiere una afinación precisa del umbral utilizado para identificar los puntos característicos, junto con el ajuste del número de niveles y subniveles dentro del espacio de escala no lineal.

2.3. Estructura de células en crecimiento

Las estructuras de células en crecimiento (GCS) [29] están disponibles en variantes supervisadas y no supervisadas. La variante de interés en este contexto es el modelo no supervisado, que ofrece la ventaja significativa de determinar automáticamente una estructura y tamaño de red adecuados. Esta capacidad se facilita mediante un crecimiento controlado, que incluye la eliminación periódica de unidades. Este modelo se basa en el trabajo de Kohonen [30] sobre mapas autoorganizativos. El pseudocódigo para GCS se presenta en el algoritmo 1 (Figura 3).

El algoritmo de estructura de células en crecimiento (GCS) ofrece varias ventajas clave. Puede ajustar autónomamente el número de neuronas, añadiéndolas o eliminándolas según sea necesario. Funciona como una red no supervisada, lo que le permite formar asociaciones de vectores de entrada de manera independiente de la entrada externa. Su simplicidad de implementación es también una característica destacable. Sin embargo, una desventaja notable es que la red puede fallar si los vectores a asociar están muy cerca unos de otros.

Algorithm 1: Growing Cell Structure [30].

Data: ϵ_b Best matching, ϵ_n neighboring and λ steps
Start: k-dimensional simplex $V = R^b$
while *(≠ desired network size)* **do**
  for *adaptationsteps* $= 0 \rightarrow \lambda$ **do**
    Choose an input signal ξ according to $P(\xi)$
    Locate the best matching unit $s = \phi_w(\xi)$.
    Increase matching:
    $\Delta w_s = \epsilon_b(\xi - w_s)$
    $\Delta w_c = \epsilon_n(\xi - w_c)(\forall c \in N_s)$
    Increment the signal counter of s:
    $\Delta\tau_s = 1$
    Decrease all signal counters by a fraction α in the network A:
    $\Delta\tau_c = -\alpha\tau_c(\forall c \in A)$
Determine $q : h_q \geqslant h_c(\forall c \in A)$
Look q largest distance neighbor $f : \| w_f - w_q \| \geqslant \| w_c - w_q \| (\forall c \in N_q)$
Insert cell r between q ans f.
Initialize $r : w_r = 0 : 5(w_q + w_f)$
Redistribute counter:

$$\Delta\tau_c = \frac{|F_c^{new}| - |F_c^{old}|}{|F_c^{old}|}\tau_c$$

Initialize new cell:

$$\tau_r = -\sum_{c \in N_r} \Delta\tau_c$$

After insertion, check $\hat{p}_i < \eta$
Cells remove:

$$\hat{p} = \tilde{p} - \sum_{c \in A} \tilde{f}_c$$

Figura 3. Estructura celular en crecimiento [30]

2.4. Simulador robótico WEBOTS

Webots [31] es una aplicación de escritorio de código abierto y multiplataforma para la simulación de robots. Por esta razón, se utilizará Webots para simular el sistema y facilitar su respectiva validación.

Este simulador de software permite probar aplicaciones y algoritmos para el robot NAO dentro de un entorno virtual. La Figura 4 ilustra el entorno del software, un mundo virtual que simula los movimientos de NAO mientras se adhiere a las leyes físicas. Este entorno ofrece un entorno seguro para probar comportamientos antes de que se implementen en un robot real.

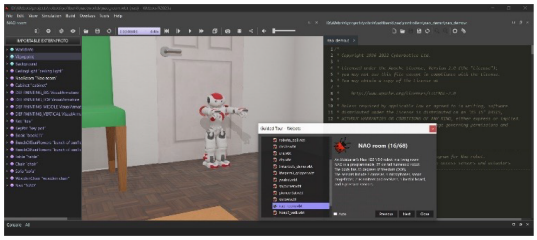

Figura 4. Entorno de Webots [32]

2.5. Construcción de mapas bidimensionales

Como se mencionó en la Introducción, la autonomía se logra a través de un sistema de planificación y control de actividades, diseñado para asegurar el cumplimiento de sus objetivos. Una característica clave de estos sistemas es la navegación espacial, que permite el cálculo de la pose del robot (posición y orientación) basada en mediciones incrementales, inerciales y visuales. Esta sección presenta los conceptos de odometría y características visuales empleados en el módulo de navegación espacial. Estas herramientas permiten al robot construir un mapa bidimensional y localizarse mientras navega.

2.5.1. Odometría

La odometría facilita la estimación de la posición relativa de un robot móvil dentro de un entorno durante la navegación, partiendo de su ubicación inicial. Además, registra y rastrea el movimiento del robot dentro de un espacio para construir un mapa bidimensional. El robot NAO tiene funciones que abordan varios desafíos, incluida la odometría. El algoritmo 2 (ver Figura 5) muestra el pseudocódigo donde se utilizan funciones de la odometría inercial de Aldebaran [24].

Algorithm 2: Pseudo code to store inertial odometry using Aldebaran functions [25].

```
//Store the initial position
AL::Math::Pose2D worldToRobotInit= Pose2D(getRobotPosition())
//Wait until it finishes scrolling
//Storing the final position
AL::Math::Pose2D worldToRobotAfter= Pose2D(getRobotPosition())
Pose2D robotMove = pose2DInverse(worldToRobotInit)+worldToRobotAfter //Movement
theta = modulo2PI(robotMove.theta)//Angle
```

Figura 5. Pseudocódigo para almacenar odometría inercial utilizando funciones de Aldebaran [26]

En esta implementación, la posición bidimensional del robot se inicializa con valores explícitos obtenidos de los valores de pose inicializados, que se recuperan de los encoders rotatorios magnéticos (MRE) de las articulaciones. Cada vez que el robot se activa, registra una posición absoluta dentro del mundo del escenario.

Al construir el mapa bidimensional, el robot primero guarda su posición inicial. Luego, se le instruye para que siga una trayectoria de circuito cerrado predeterminada dentro de la habitación, avanzando una distancia especificada mientras camina. A medida que se mueve, la posición bidimensional del robot se registra periódicamente. Posteriormente, se calculan el desplazamiento y el ángulo recorridos por el robot. La posición bidimensional entre puntos consecutivos se calcula entonces para reflejar con precisión el movimiento del robot.

Así, la implementación general de la odometría puede establecerse de la siguiente manera:

1. Capturar la posición del robot relativa al entorno antes de caminar.
2. Detectar cuando el robot comienza a caminar.
3. Simultáneamente, comenzar a recopilar datos odométricos

4. Procesar y acumular los datos odométricos.
5. Detectar la finalización del recorrido del robot. Si el recorrido no está completado, repetir los pasos 3 y 4.
6. Calcular la distancia recorrida por el robot.
7. Almacenar los datos de distancia y posición del robot para construir el mapa bidimensional.

La Figura 6 presenta el diagrama de flujo general para generar el mapa bidimensional. En este diagrama, el algoritmo comienza con la captura de una imagen. Posteriormente, se extraen características visuales de esta imagen; estas características sirven como entradas para el sistema de aprendizaje, es decir, la red neuronal. A continuación, el robot aprende y registra la localización espacial correspondiente a su posición. Si el recorrido designado se completa, el algoritmo concluye. Si no, el robot se mueve a la siguiente posición y el algoritmo continúa hasta que se alcance el final del recorrido.

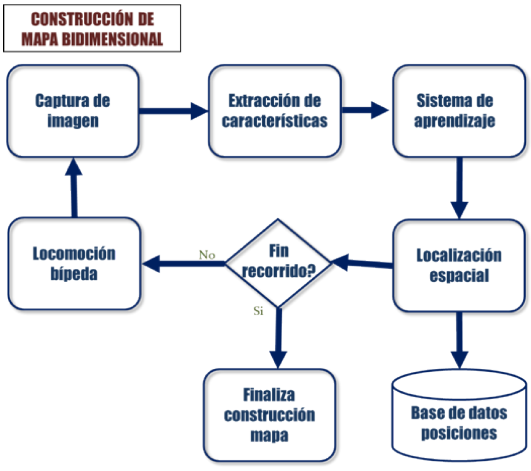

Figura 6. Diagrama de flujo de datos para la reconstrucción del mapa

2.6. Características visuales

Los elementos visuales se identifican analizando y catalogando los detalles existentes en el entorno, considerando la posición del robot cuando se captura la imagen. Generalmente, se considera que los elementos visuales más significativos para representar el entorno son aquellos ubicados en o cerca de las paredes. Un mapa bidimensional de la habitación recorrida por el robot puede construirse utilizando estos elementos visuales y la ubicación estimada derivada de la odometría. Por ejemplo, la Figura 7 ilustra una habitación virtual con varios objetos cotidianos típicamente encontrados en un hogar. Estos objetos suelen permanecer estacionarios. Por lo tanto, el robot debe navegar por esta habitación siguiendo un circuito cerrado, preferiblemente cuadrangular, capturar imágenes y registrar la posición estimada desde donde se tomó cada imagen.

Figura 7. Habitación virtual en Webots

Además, el robot debe centrarse en capturar imágenes de la pared más cercana a su recorrido. La Figura 8 ilustra tres capturas realizadas por el robot en diferentes puntos. Durante la navegación, tomará una captura de pantalla en cada paso basado en el número de imágenes especificadas por el usuario para la habitación. Por ejemplo, si se requieren veinte imágenes en una habitación donde cada pared mide 4 metros de largo, se tomará una imagen cada 20 cm. Además del número de capturas y las dimensiones de las paredes de la habitación, también se puede determinar la frecuencia del recorrido. Cuantos más circuitos se completen, más detallada será la construcción del mapa de la habitación y más fácil será localizar al robot.

Una vez completados los circuitos, el robot utiliza la información almacenada para construir el mapa bidimensional. Las imágenes capturadas contendrán objetos de los cuales se deben extraer detalles específicos. El módulo de reconocimiento de objetos [32] procesa las imágenes para obtener descriptores, que luego se aprenden y se vinculan a la pose del robot durante la captura. Esta información se integra en una representación bidimensional, formando el mapa de la habitación. Antes de iniciar cualquier recorrido por la habitación, el robot debe identificar la pared más cercana para determinar dónde enfocará sus capturas de imagen, simplemente girando su cabeza hacia la pared visible. Esta detección de paredes se logra estimando visualmente las distancias. Antes de la navegación, el robot debe estar posicionado en paralelo a la pared seleccionada y colocado en una esquina de la habitación. Luego captura una imagen, que se analiza posteriormente dividiéndola en dos partes.

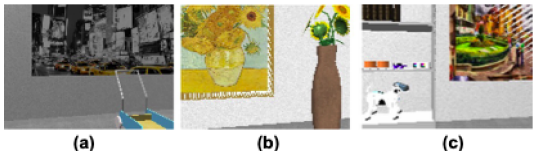

Figura 8. Imágenes adquiridas por el robot NAO en una habitación simulada

Por ejemplo, la Figura 9 muestra tres capturas diferentes de una habitación tomadas desde varias posiciones del robot. En la imagen 9(a), la pared más cercana está a la izquierda, mientras que en las imágenes 9(b) y 9(c), está a la derecha. Para cada imagen, se identifican puntos de interés en cada lado, utilizando el algoritmo A-KAZE [26]. La imagen con más puntos salientes indica la ubicación de la pared más cercana, asumiendo que la habitación está libre de obstáculos.

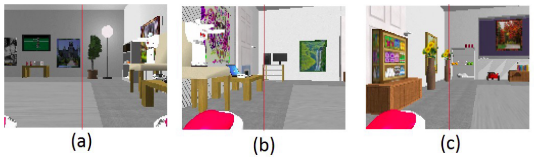

Figura 9. Imágenes tomadas por el robot NAO anteriormente

La Figura 10 muestra los resultados de la evaluación para cada imagen, con los puntos salientes indicados por pequeños círculos de colores. En la imagen 10(a), el lado derecho contiene la mayoría de los puntos salientes, con 108 en comparación con 36 en el lado izquierdo; en la 10(b), predomina el lado izquierdo con 128 puntos frente a 50 en el derecho; y en la 10(c), nuevamente el lado izquierdo lidera con 119 puntos comparados con 53 en el derecho. Basándose en estas observaciones, el robot luego gira hacia el lado con más puntos salientes para continuar su exploración de la habitación.

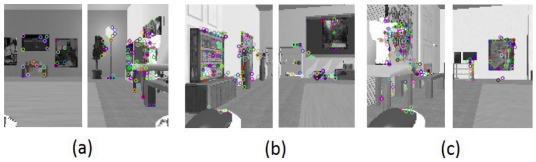

Figura 10. Detección de paredes utilizando descriptores

2.7. Algoritmo para la construcción del mapa

La construcción del mapa bidimensional procede de la siguiente manera: inicialmente, el robot realiza un circuito cerrado alrededor de una habitación cuadrada, capturando y registrando imágenes junto con sus respectivas poses. Es esencial conocer las dimensiones de las paredes, el tamaño del paso durante el movimiento y el número de iteraciones. El robot mejora su comprensión de la habitación con cada circuito adicional completado. Al inicio del algoritmo 3 (Figura 11), el robot realiza una captura inicial para detectar la pared más cercana y determinar el ángulo para su siguiente giro.

Antes de comenzar su movimiento, el robot registra su posición actual mediante odometría como el punto de referencia global para la habitación. Posteriormente, se registra la distancia recorrida, indicando tanto la longitud del trayecto del robot como la distancia total que necesita navegar dentro de la habitación. Esta medida se monitorea continuamente durante un ciclo de trabajo, que persiste hasta que la distancia recorrida sea igual a la longitud combinada de las cuatro paredes de la habitación.

Algorithm 3: Navigation Module. *Execution of a closed lap.*
Data: d dimensions of the room, p step size of the robot when walking, n number of paths
Result: *data* images and poses
picture =TakePicture()
AngleYaw=DetectNearestWall(**picture**)
TurnHead(**AngleYaw**)
O_w=CurrentPose()
TotalDistanceWalked=0
for $j = 1$ *a* **n do**
  while $TotalDistanceWalked \neq d \times 4$ **do**
    DistanceWalked=0
    while $DistanceWalked \neq d$ **do**
      Walk(*p*)
      RP=CurrentPose()
      picture =TakePicture()
      data = (**P**, **I**)
      DistanceWalked=DistanceWalked+*p*
    TurnBody(-**AngleYaw**)
    TotalDistanceWalked=TotalDistanceWalked+DistanceWalked

Figura 11. Módulo de navegación. Ejecución de una vuelta cerrada

Después de completar el circuito y almacenar la base de datos de la habitación, se inicia el algoritmo 4 (Figura 12) para aprender de una nueva base de datos que incluye información de captura y pose. Se extraen todos los puntos de interés, se construyen histogramas y se entrena una red neuronal utilizando el método de Growing Cell Structures (GCS) [32].

Algorithm 4: Navigation Module. *Construction of two-dimensional map*
Data: I images, P poses.
Result: ***classes(N,P)*** classes of the two-dimensional map.
for $i \leftarrow 1$ *to* I **do**
  keypoints =AKAZE(***I(i)***)
  H(i)=BuildHistos(keypoint)
ANN_trained=GCS(***H***)
classes(***N***)=ANN_trained
for $i \leftarrow 1$ *to* N **do**
  NewPose(i) $= \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{P}(classes(i))$

Figura 12. Módulo de navegación. Construcción de mapa bidimensional

El algoritmo 5 (Figura 13) se utiliza para evaluar el mapa. Este módulo procesa las imágenes, extrayendo puntos salientes y construyendo histogramas. Estos histogramas se utilizan luego para evaluar la red neuronal entrenada. Este proceso ayuda a identificar las neuronas correspondientes. Una vez determinadas las clases, se recuperan las poses asociadas. Las posiciones

bidimensionales en el mapa se calculan luego y se devuelven.

El algoritmo opera dentro de ciertas restricciones, que incluyen conocer las dimensiones de la habitación para calcular la distancia total que el robot recorrerá alrededor de ella. Además, el entorno debe estar libre de obstáculos, ya que este trabajo no incorpora estrategias de evasión de los mismos.

Finalmente, si algún elemento dentro de la habitación ha sido movido, el robot debe reconstruir su mapa de navegación para reflejar estos cambios.

Algorithm 5: Navigation Module. *Using the two-dimensional map*

Data: $\boldsymbol{I}$ images
Result: ***class(I)*** object classes, ***Pose*** pose
1 keypoints =AKAZE($\boldsymbol{I}$)
2 H($\boldsymbol{I}$)=BuildHistos(keypoint)
3 classes($\boldsymbol{I}$)=ANN_trained
4 Pose=$\frac{1}{n}\sum_{j=1}^{n}$ ***NewPose(I)***$(classes(i))$

Figura 13. Módulo de navegación. Uso del mapa bidimensional

3. Resultados y discusión

Los experimentos descritos en esta sección se dividen en dos partes: (1) construcción de un mapa bidimensional y (2) localización dentro del mapa. Estos experimentos se han llevado a cabo utilizando tanto robots NAO virtuales como reales.

3.1. Entorno virtual

3.1.1. Construcción del mapa

La habitación simulada representada en la Figura 7, con dimensiones de 6×6 metros, fue creada utilizando Webots. Esta habitación estaba amueblada con varios objetos, como sillas, mesas y retratos. Se empleó un robot NAO virtual para construir el mapa bidimensional de la habitación. El robot inició su recorrido desde la esquina inferior izquierda de la habitación, navegando en un circuito cerrado cuadrangular. El robot giró la cabeza hacia las paredes durante todo su recorrido para capturar imágenes. Las paredes se numeraron del 1 al 4 en sentido antihorario para ilustrar los resultados de la construcción del mapa bidimensional.

El robot completó dos circuitos en sentido antihorario alrededor de la habitación, capturando imágenes y registrando sus relaciones espaciales. El número de imágenes tomadas por pared se detalla en la Tabla 1. El término 'vuelta' se refiere al número de circuitos que completa el robot. 'Imágenes' indica el número total de imágenes guardadas durante cada vuelta. 'Pared 1', 'Pared 2', 'Pared 3' y 'Pared 4' indican el número de imágenes almacenadas para cada pared. En total, se registraron 164 imágenes y sus poses asociadas, las cuales se utilizaron para construir el mapa bidimensional.

Tabla 1. Principales parámetros de los experimentos: Construcción del mapa bidimensional

Vuelta	Imágenes	Pared 1	Pared 2	Pared 3	Pared 4
1	89	20	21	31	17
2	75	18	20	24	13

Los parámetros correspondientes al reconocimiento de objetos se detallan en la Tabla 2. Como se señala, se realizaron tres iteraciones de construcción de mapas bidimensionales utilizando 100, 200 y 300 neuronas, respectivamente. El objetivo fue evaluar la efectividad del módulo en construir un mapa que refleje con precisión las observaciones del robot dentro de la habitación.

Tabla 2. Parámetros del módulo de reconocimiento de objetos para la construcción del mapa bidimensional

Experimento	Entrenamiento	Neuronas	Época	Tiempo
1	164	100	100	4,063
2	164	200	200	14,287
3	164	300	300	33,347

Después del entrenamiento, se generaron mapas bidimensionales que contenían 72, 111 y 132 poses, respectivamente. Los puntos marcados en cada mapa en la Figura 14 representan una pose asociada con una neurona. Es evidente que a medida que aumenta el número de neuronas, la distribución de las poses se vuelve más refinada. Es importante destacar que las poses se homogeneizaron en las coordenadas y se mantuvieron constantes durante el recorrido para asegurar que se muestre una distribución precisa.

La distribución en el mapa construido con cien neuronas es subóptima, ya que incluye algunas poses en áreas donde el robot no ha viajado, junto con agrupamientos de poses en ciertas secciones. La distribución mejora significativamente en el mapa construido con doscientas neuronas, aunque aún se observa cierto apilamiento de poses. El mapa con trescientas neuronas muestra la mejor distribución, cubriendo más áreas de manera más completa. Aunque aún existen algunas poses erróneas, son mínimas.

Gracias al mapa bidimensional, el robot puede identificar las ubicaciones de las paredes, lo que le permite evitarlas mientras ejecuta sus tareas.

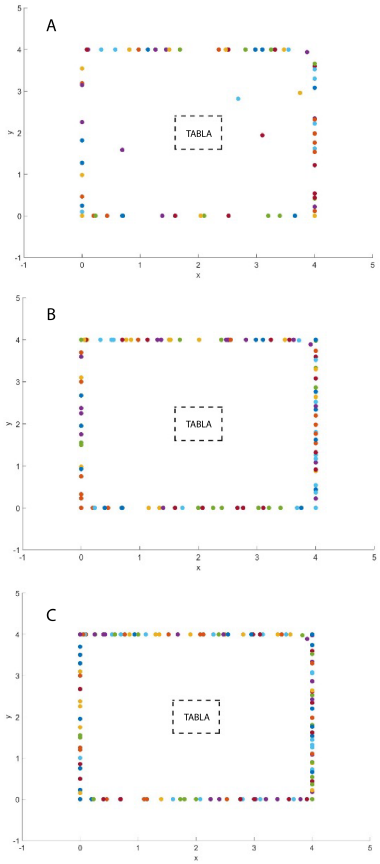

Figura 14. Distribución de neuronas por poses en la habitación de los experimentos: (a) 1, (b) 2 y (c) 3.

3.1.2. Ubicación en el mapa

El propósito del mapa bidimensional es permitir que el robot regrese a la posición global 0 en el mapa, una vez que haya completado sus tareas. Con el mapa construido, el robot puede determinar su ubicación dentro de la habitación utilizando una o dos imágenes de las paredes más cercanas. Se realizaron cuatro experimentos para validar esta funcionalidad.

La Tabla 3 enumera los parámetros, que incluyen el número de experimento, el mapa bidimensional construido en la sección anterior (1, 2 y 3), y la posición real a calcular (x, y) en metros.

La construcción del mapa bidimensional se evalúa de la siguiente manera: el robot virtual captura dos imágenes desde diferentes perspectivas en cada una de las cuatro posiciones más cercanas a las paredes bajo evaluación. Ejemplos de estas capturas realizadas por el robot se ilustran en la Figura 15. En cada posición, se capturan dos imágenes de las paredes más cercanas.

Las dos primeras imágenes superiores corresponden a la posición (0, 0), mientras que las dos siguientes corresponden a la posición (4, 4) dentro de la habitación. Los resultados se presentan en la Tabla 3, que detalla cinco evaluaciones con dos imágenes para cada experimento. El módulo registra las poses individuales capturadas en cada columna para los cuatro experimentos, con un par de imágenes por evaluación.

Figura 15. Ejemplos de capturas realizadas por el robot NAO virtual

Tabla 3. Resultados de las evaluaciones de poses y ubicaciones en el mapa

N.°	Mapa	(x,y) m	1	2	3	4	5
1	1	(3.5,0.5)	(3.7,1.7)	(3.6,1.6)	(3.7,1.7)	(3.8,1.8)	(3.3,1.7)
2	1	(0.0)	(0.1,0.2)	(0.2.0.2)	(0.3.0.1)	(0.3,0.0)	(0.3.0.2)
3	2	(0.5,3.5)	(0.0,2.0)	(0.0,2.0)	(0.2,2.2)	(0.0,2.0)	(0.2,2.2)
4	3	(4,4)	(4.0,3.8)	(3.9,3.9)	(4,4)	(3.9,3.9)	(3.9,3.9)

3.2. Escenario real

3.2.1. Construcción del mapa

El mapa bidimensional se construyó en una habitación de 4×3 metros, dentro de la cual el robot desarrolló un mapa de 3×3 metros. La habitación contiene varios elementos, incluidos carteles con información diversa. La Figura 16 muestra las cuatro paredes de la habitación, ilustrando los elementos utilizados para el aprendizaje. Además, hay una plataforma de 30 cm ubicada en el centro de la habitación, como se muestra en la Figura 17. Esta plataforma sostiene 20 objetos distribuidos a lo largo de los bordes, mejorando la visibilidad para el robot y asegurando que los objetos permanezcan dentro del área de trabajo de los manipuladores para facilitar su recuperación.

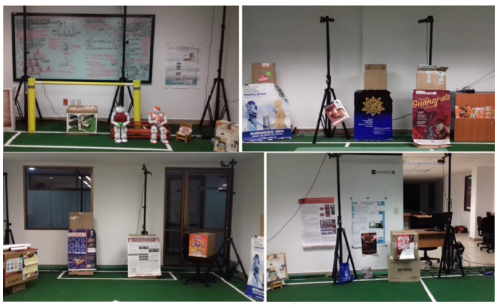

Figura 16. Las paredes del escenario real para la construcción del mapa

Esta evaluación completó tres circuitos para construir un mapa más preciso. El robot inició su ruta desde las coordenadas globales de la habitación (0, 0), ubicadas en la esquina derecha de la pared 1.

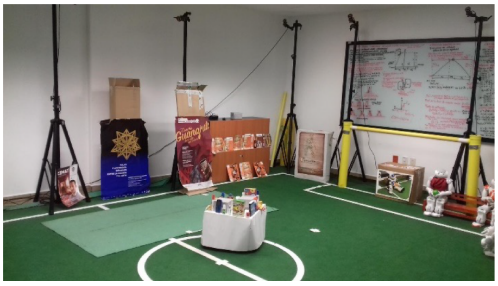

Figura 17. Plataforma con objetos colocados en el centro de la habitación

Durante los recorridos, el robot gira la cabeza hacia la pared para capturar imágenes mientras avanza y mantiene su posición relativa (Figura 18).

Figura 18. El robot sigue su trayectoria dirigiéndose hacia la pared para capturar imágenes

La Tabla 4 detalla las imágenes capturadas durante cada circuito a lo largo de las paredes. Esta tabla especifica el número de circuitos completados, el total de imágenes tomadas y las imágenes capturadas correspondientes a cada pared. Se registraron un total de sesenta y ocho imágenes y poses utilizadas para construir el mapa bidimensional.

Al igual que en el escenario virtual, las imágenes capturadas se introducen en el módulo de reconocimiento de objetos, encargado del aprendizaje de características y la generación del mapa de la habitación.

Tabla 4. Principales parámetros de los experimentos: Construcción del mapa bidimensional

Vuelta	Imágenes	Pared 1	Pared 2	Pared 3	Pared 4
1	25	7	7	5	6
2	26	5	8	5	8
3	17	5	5	4	3

La Tabla 5 presenta los parámetros utilizados para generar el mapa de la habitación, incluyendo el número de experimentos realizados, las imágenes utilizadas para el entrenamiento, el número de neuronas y las épocas involucradas.

Tabla 5. Parámetreos para la construcción del mapa

Experimento	Entrenamiento	Neuronas	Veces
1	68	400	100
2	68	500	200

Después del entrenamiento, los mapas bidimensionales contenían 27 y 36 poses, respectivamente. La Figura 19 muestra estos mapas, donde los puntos azules indican las poses en las que el robot capturó imágenes. La Figura 19(a) ilustra el mapa ideal, mostrando las poses objetivo para la captura de imágenes durante los experimentos. El mapa construido para el experimento 1 corresponde a la Figura 19(b), mientras que el mapa para el segundo experimento se representa en la Figura 19(c). Estos mapas reflejan la distribución de las neuronas asociadas con cada pose. Se observa que al aumentar el número de neuronas de 400 a 500 mejora ligeramente la distribución. Sin embargo, el tamaño del mapa construido se redujo de 3×3 metros a 1,5×1,5 metros.

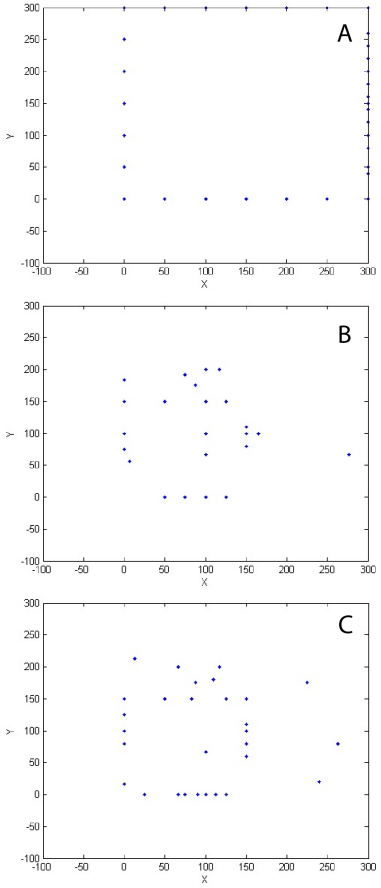

Figura 19. Distribución de neuronas por poses en la habitación

Desde el análisis anterior, se puede inferir que la reducción en el tamaño del mapa se debió a numerosos falsos positivos y a la interasociación de poses, lo que condujo a su consolidación.

3.2.2. Localización en el mapa bidimensional

La tarea de localización en el mapa bidimensional sirve varios propósitos. Un objetivo clave es que el robot regrese al punto de partida para entregar un objeto solicitado por el usuario. Además, el robot utiliza el mapa para localizar las paredes de la habitación, lo que ayuda a evitarlas durante las tareas de búsqueda de objetos.

Por lo tanto, con el mapa construido, el robot puede determinar su ubicación dentro de la habitación utilizando una o dos imágenes de las paredes más cercanas. Esta capacidad se evaluó a través de diez experimentos que se llevaron a cabo.

La Tabla 6 enumera los números de experimento y las posiciones correspondientes a calcular (x, y) en metros.

Tabla 6. Parámetros principales de los experimentos para la ubicación del mapa

N°	1	2	3	4	5	6	7	8	9	10
(x,y)m	(0,0)	(0,1)	(0,2)	(0,3)	(1,0)	(2,0)	(3,0)	(3,3)	(1.5,2)	(2,1.5)

La precisión de la ubicación del robot en el mapa bidimensional se determina utilizando dos imágenes capturadas desde las posiciones potenciales más cercanas a la pared adyacente a ese punto.

Los resultados se presentan en la Tabla 7, donde se muestran las diez ubicaciones asociadas con dos poses derivadas de la evaluación de las dos imágenes tomadas desde cada posición. Debido a las imprecisiones en la construcción del mapa bidimensional, las poses obtenidas no coinciden estrechamente con las posiciones reales.

La mayor precisión se logró con la pose número (1), mostrando una precisión de ±(0.25, 0.16) cercana a la pose esperada. La siguiente mejor precisión se obtuvo con la pose número (9), mostrando una precisión de ±(0.34, 0.62). Las poses menos precisas fueron (2) y (8), con precisiones de ±(1.50, 0.50) y ±(1.75, 2.00), respectivamente. Aunque el mapa construido fue inexacto, la evaluación arroja resultados favorables dados el mapa entrenado. La Figura 20 proporciona una representación gráfica de las diez ubicaciones determinadas, utilizando el módulo en el mapa bidimensional previamente entrenado. Se observa que la mayoría de las poses están muy cerca de las posiciones entrenadas, excepto 2, 7 y 8, que estaban significativamente desalineadas.

Tabla 7. Resultados de las evaluaciones de poses para cada experimento

N°	1	2	3	4	5	6	7	8	9	10
1	(0,0.16)	(1.50,1.50)	(0.75,0)	(1.00,0)	(0,0.80)	(0,1.0)	(0.25,0)	(1.25,0)	(1.06,1.75)	(1.50,0.80)
2	(0.25,0)	(1.50,0.60)	(1.25,0)	(0,0.80)	(0.25,0)	(0.25,0)	(0.66,0)	(0,1.00)	(1.25,1.00)	(1.25,1.00)

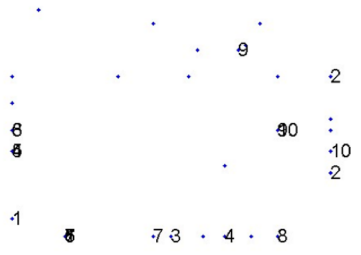

Figura 20. Gráfico con las diez ubicaciones determinadas en el mapa entrenado

4. Conclusiones

Este estudio presenta el desarrollo de un algoritmo para la construcción de mapas bidimensionales utilizando odometría inercial y elementos visuales. El mapa bidimensional se crea utilizando un módulo de reconocimiento de objetos basado en características locales y redes neuronales artificiales no supervisadas. Este módulo se utiliza para aprender el diseño de la habitación y asociar una pose con cada neurona en la red, que se entrena para representar el mapa bidimensional.

Se realizaron experimentos utilizando (1) un robot NAO virtual y (2) un robot NAO real en un escenario auténtico. Los resultados son prometedores, ya que fue posible construir un mapa bidimensional de la habitación y localizar con precisión el robot móvil con una precisión de hasta ±(0.06, 0.1) en simulación y ±(0.25, 0.16) en el entorno natural. Estos resultados pueden mejorarse aún más mejorando la calidad de las imágenes capturadas.

El enfoque para generar mapas a partir de información visual presenta varias limitaciones, incluyendo las siguientes:

1. Las cámaras del robot NAO no son óptimas para capturar imágenes de alta calidad, lo que conduce a errores tanto en las fases de aprendizaje como de reconocimiento.

2. El entorno debe estar estructurado para incluir suficientes referencias visuales en las paredes de la habitación para mejorar la precisión de localización del robot.

3. Esta implementación no tiene en cuenta elementos dinámicos; por lo tanto, la escena solo contiene al robot, la mesa y los objetos circundantes.

4. El camino del robot debe ser recto, lo que requiere un camino despejado libre de objetos para permitir un posicionamiento adecuado con respecto a la pared y sus marcadores visuales.

5. El enfoque depende en gran medida de la información visual, por lo que la ausencia de esta información causaría confusión y obstaculizaría significativamente la capacidad del robot para navegar por la habitación.

4.1. Trabajo futuro

1. **Fusión avanzada de sensores.** El trabajo futuro se centrará en mejorar la integración de datos de odometría inercial y elementos visuales. Este enfoque tiene como objetivo reducir la dependencia únicamente de características visuales, mejorando así la robustez y precisión del sistema.

2. **Evaluación de arquitecturas de redes neuronales.** Se evaluarán diversas arquitecturas de redes neuronales para determinar cuál es la más adecuada para la tarea de construcción de mapas. La arquitectura que muestre el mejor rendimiento será seleccionada para un desarrollo e implementación adicionales.

3. **Pruebas de detectores avanzados de puntos característicos.** Para mejorar el rendimiento del sistema, se probarán detectores de puntos característicos de última generación. Se espera que estos detectores ofrezcan mejoras significativas en la detección y procesamiento de puntos característicos, contribuyendo a la eficiencia general del sistema.

Referencias

[1] IFR. (2024) Homepage. International Federation of Robotics. International Federation of Robotics. [Online]. Available: https://ifr.org/

[2] Y. Omori, T. Furukawa, T. Ishikawa, and M. Inaba, "Humanoid vision design for object detection, localization and mapping in indoor environments," in *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1109/SSRR.2018.8468604

[3] X. Cui, M. Wang, B. Fan, and J. Yi, "Target detection based on binocular stereo vision," in *2017 International Conference on Computer Technology, Electronics and Communication (ICCTEC)*, 2017, pp. 1093–1097. [Online]. Available: https://doi.org/10.1109/ICCTEC.2017.00239

[4] R. Scona, S. Nobili, Y. R. Petillot, and M. Fallon, "Direct visual SLAM fusing proprioception for a humanoid robot," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1419–1426. [Online]. Available: https://doi.org/10.1109/IROS.2017.8205943

[5] L. K. Garzón Obregón, L. A. Forero Rincón, and O. M. Duque Suárez, "Diseño e implementación de un sistema de visión artificial usando una técnica de mapeo y localización simultánea (SLAM) sobre una plataforma robótica móvil," *Mundo FESC*, vol. 8, no. 16, pp. 8–17, 2018. [Online]. Available: https://is.gd/pqjvTy

[6] F. Blochliger, M. Fehr, M. Dymczyk, T. Schneider, and R. Siegwart, "Topomap: Topological mapping and navigation based on visual SLAM maps," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3818–3825. [Online]. Available: https://doi.org/10.1109/ICRA.2018.8460641

[7] E. Ovalle-Magallanes, N. G. Aldana-Murillo, J. G. Avina-Cervantes, J. Ruiz-Pinales, J. Cepeda-Negrete, and S. Ledesma, "Transfer learning for humanoid robot appearance-based localization in a visual map," *IEEE Access*, vol. 9, pp. 6868–6877, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3048936

[8] T. Lasguignes, I. Maroger, M. Fallon, M. Ramezani, L. Marchionni, O. Stasse, N. Mansard, and B. Watier, "ICP Localization and Walking Experiments on a TALOS humanoid robot," in *2021 20th International Conference on Advanced Robotics (ICAR)*, 2021, pp. 800–805. [Online]. Available: https://doi.org/10.1109/ICAR53236.2021.9659474

[9] P. Wozniak, H. Afrisal, R. G. Esparza, and B. Kwolek, "Scene recognition for indoor localization of mobile robots using deep CNN," in *Computer Vision and Graphics*, L. J. Chmielewski, R. Kozera, A. Orłowski, K. Wojciechowski, A. M. Bruckstein, and N. Petkov, Eds. Cham: Springer International Publishing, 2018, pp. 137–147. [Online]. Available: https://doi.org/10.1007/978-3-030-00692-1_13

[10] E. S. Lahemer and A. Rad, "An adaptive augmented vision-based ellipsoidal slam

for indoor environments," *Sensors*, vol. 19, no. 12, 2019. [Online]. Available: https://doi.org/10.3390/s19122795

[11] T.-J. Lee, C.-H. Kim, and D.-I. D. Cho, "A monocular vision sensor-based efficient slam method for indoor service robots," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 1, pp. 318–328, 2019. [Online]. Available: https://doi.org/10.1109/TIE.2018.2826471

[12] M. Fourmy, D. Atchuthan, N. Mansard, J. Solà, and T. Flayols, "Absolute humanoid localization and mapping based on IMU Lie group and fiducial markers," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, 2019, pp. 237–243. [Online]. Available: https://doi.org/10.1109/Humanoids43949.2019.9035005

[13] S. J. Dignadice, J. R. Red, A. J. Bautista, A. Perol, A. Ollanda, and R. Santos, "Application of simultaneous localization and mapping in the development of an autonomous robot," in *2022 8th International Conference on Control, Automation and Robotics (ICCAR)*, 2022, pp. 77–80. [Online]. Available: https://doi.org/10.1109/ICCAR55106.2022.9782658

[14] S. Wen, M. Sheng, C. Ma, Z. Li, H. K. Lam, Y. Zhao, and J. Ma, "Camera recognition and laser detection based on EKF-SLAM in the autonomous navigation of humanoid robot," *Journal of Intelligent & Robotic Systems*, vol. 92, no. 2, pp. 265–277, Oct 2018. [Online]. Available: https://doi.org/10.1007/s10846-017-0712-5

[15] X. Deng, Z. Zhang, A. Sintov, J. Huang, and T. Bretl, "Feature-constrained active visual SLAM for mobile robot navigation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7233–7238. [Online]. Available: https://doi.org/10.1109/ICRA.2018.8460721

[16] A. Li, J. Wang, M. Xu, and Z. Chen, "DP-SLAM: A visual SLAM with moving probability towards dynamic environments," *Information Sciences*, vol. 556, pp. 128–142, 2021. [Online]. Available: https://doi.org/10.1016/j.ins.2020.12.019

[17] N. Krombach, D. Droeschel, S. Houben, and S. Behnke, "Feature-based visual odometry prior for real-time semi-dense stereo SLAM," *Robotics and Autonomous Systems*, vol. 109, pp. 38–58, 2018. [Online]. Available: https://doi.org/10.1016/j.robot.2018.08.002

[18] Y. S. Jiyu Cheng and M. Q.-H. Meng, "Improving monocular visual slam in dynamic environments: an optical-flow-based approach," *Advanced Robotics*, vol. 33, no. 12, pp. 576–589, 2019. [Online]. Available: https://doi.org/10.1080/01691864.2019.1610060

[19] S. Ganesan and S. K. Natarajan, "A novel directional sampling-based path planning algorithm for ambient intelligence navigation scheme in autonomous mobile robots," *Journal of Ambient Intelligence and Smart Environments*, vol. 15, pp. 269–284, 2023, 3. [Online]. Available: https://doi.org/10.3233/AIS-220292

[20] K. Zhang, H. Gui, Z. Luo, and D. Li, "Matching for navigation map building for automated guided robot based on laser navigation without a reflector," *Industrial Robot: the international journal of robotics research and application*, vol. 46, no. 1, pp. 17–30, Jan 2019. [Online]. Available: https://doi.org/10.1108/IR-05-2018-0096

[21] C. Wang, J. Wang, C. Li, D. Ho, J. Cheng, T. Yan, L. Meng, and M. Q.-H. Meng, "Safe and robust mobile robot navigation in uneven indoor environments," *Sensors*, vol. 19, no. 13, 2019. [Online]. Available: https://doi.org/10.3390/s19132993

[22] A. Roychoudhury, M. Missura, and M. Bennewitz, "3D polygonal mapping for humanoid robot navigation," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 2022, pp. 171–177. [Online]. Available: https://doi.org/10.1109/Humanoids53995.2022.10000101

[23] F. Martín, J. Ginés, D. Vargas, F. J. Rodríguez-Lera, and V. Matellán, "Planning topological navigation for complex indoor environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9. [Online]. Available: https://doi.org/10.1109/IROS.2018.8594038

[24] Aldebaran. NAO Documentation. Aldebaran NAO Documentation. Aldebaran NAO Documentation. [Online]. Available: https://is.gd/eSNPWH

[25] MIA. (2023) Mathematical image analysis group. MIA Group. [Online]. Available: https://is.gd/69mEso

[26] P. Fernández Alcantarilla, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," in *British Machine Vision Conference (BMVC)*, 09 2013. [Online]. Available: http://dx.doi.org/10.5244/C.27.13

[27] H. Scharr, *Optimale Operatoren in der Digitalen Bildverarbeitung*. University of Heidelberg, Germany, 2000. [Online]. Available: https://doi.org/10.11588/heidok.00000962

[28] X. Yang and K. Cheng, "LDB: an ultra-fast feature for scalable augmented reality on mobile devices," *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 49–57, 2012. [Online]. Available: https://doi.org/10.1109/ISMAR.2012.6402537

[29] B. Fritzke, "Growing cell structures—a self-organizing network for unsupervised and supervised learning," *Neural Networks*, vol. 7, no. 9, pp. 1441–1460, 1994. [Online]. Available: https://doi.org/10.1016/0893-6080(94)90091-4

[30] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990. [Online]. Available: https://doi.org/10.1109/5.58325

[31] Cyberbotics. (2023) Simulating your robots with webots. Cyberbotics - Robotics simulation services. Cyberbotics - Robotics simulation services. [Online]. Available: https://is.gd/Q31yau

[32] K. L. Flores-Rodríguez, F. Trujillo-Romero, and W. Suleiman, "Object recognition modular system implementation in a service robotics context," in *2017 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CONIELECOMP.2017.7891833