*Ciencia, Ingenierías y Aplicaciones,* Vol. 3, No. 2, julio-diciembre, 2020
ISSN (impreso): 2636-218X • ISSN (en línea): 2636-2171
DOI: https://doi.org/10.22206/cyap.2020.v3i2.pp91-126


# Diseño y simulación de un robot seguidor de personas para el transporte de suministros en hospitales

## *Design and Simulation of a People Following Robot to Transport Supplies in Hospitals*

Darian Marlis Rodríguez Vásquez[a], Luis Reyes Herrera[b], Amín Deschamps López[c], Yobany Díaz Roque[d]

Recibido: 8/9/2020 • Aprobado: 19/10/2020

**Cómo citar:** Rodríguez Vásquez, D. M., Reyes Herrera, L., Deschamps López, A., & Díaz Roque, Y. (2020). Diseño y simulación de un robot seguidor de personas para el transporte de suministros en hospitales. *Ciencia, Ingenierías y Aplicaciones, 3*(2), 91-126. Doi: https://doi.org/10.22206/cyap.2020.v3i2.pp91-126

**Resumen**

*Las enfermeras suelen realizar múltiples tareas de manera simultánea en hospi-tales; las de medicación son las más frecuentes y exigen el transporte de equi-pos pesados, lo cual genera agotamiento y provoca una disminución en la cali-dad de su trabajo . El objetivo de este artículo es desarrollar un prototipo simulado de un robot capaz de seguir enfermeras durante su re-corrido en una superficie plana, aligerando su carga de trabajo. Este robot debe ser capaz no solo de seguir a una persona en particular, identificada por medio de un mar-cador aruco en la espalda, sino tener la capacidad de obtener información del ambiente que le rodea mientras sigue a la persona para actuar ante los cam-bios ambientales y seguirla, incluso cuando des-aparece de su zona de visión momentáneamente al doblar en esquinas. La entrada del sistema es la imagen del marcador, que mediante el algoritmo de identificación computa las coorde-nadas del centro en pixeles y distancia al robot; si la persona está presente se utiliza un controlador P y PD para*

[a] Estudiante de Ingeniería Electrónica y de Comunicaciones, Instituto Tecnológico de Santo Domingo (INTEC), República Dominicana. Correo-e: darianrodriguezv@gmail.com ORCID: 0000-0002-2651-2405

[b] Estudiante de Ingeniería Electrónica y de Comunicaciones, (INTEC), República Dominicana. Correo-e: luismreyes62@gmail.com, ORCID: 0000-0002-7688-9577

[c] Docente de Ingeniería Electrónica y de Comunicaciones, (INTEC), República Dominicana. Correo-e: amin.deschamps@intec.edu.do, ORCID: 0000-0001-7164-7563

[d] Autor de Correspondencia. Coordinador y docente de Ingeniería Electrónica y de Comunicaciones, (INTEC), República Dominicana. Correo-e: yobany.diaz@intec.edu.do, ORCID: 0000-0002-3646-2755

*el seguimiento. En caso de que el usuario desaparezca de la cámara se realiza un planeamiento de la ruta local hacia la última coordenada global extraída de la persona, mediante la cámara monocu-lar y luego un sistema de búsqueda. Los parámetros de planeamiento se modifi-can para pasar a través de una puerta; dicho reconocimiento se logra mediante la red convolucional YOLO v3. La localización del robot se determina con la odometría de rueda y al doblar el usuario en una esquina se determina la di-rección en la cual se estaba moviendo para localizarlo nuevamente. Para anali-zar el desempeño del sistema propuesto se utilizó el software CoppeliaSim con el modelo de robot Pioneer 3-DX, donde se llevan a cabo diversas simulaciones bajo distintos escenarios a los cuales estaría expuesto el robot.*

**Palabras clave**: odometría; planeamiento de la ruta local; marcador aruco; coordenadas globales; robot seguidor de persona.

**Abstract**

*Nurses tend to perform multiple tasks simultaneously, the medication tasks being the most frequent where they transport heavy carts, which usually generates ex-haustion and causes a decrease in the quality of their work. The objective of this article is to develop a simulated prototype of a robot capable of following nurses during their journey on a flat surface, lightening their workload. This robot must be capable not only of following a particular person identified by means of an aruco marker on the back, but also have the ability to obtain information from the environment around them while following the person to act on environmental changes, and following the target even when disappearing from his vision zone momentarily when turning at different corners. The input of the system is the im-age of the marker, the coordinates of the center in pixels and distance to the ro-bot are computed by the identification algorithm. If the person is present, a P and PD controller is used for following. In case the user disappears for some reason from the camera, the robot makes a local path planning to the last global coordinate of the person extracted using the monocular camera and then a search system. The planning parameters are modified to pass through a door, this recognition is achieved through the YOLO v3 convolutional network. The localization of the robot is determined with the wheel odometry and when the target user gets around corners, the direction in which it was moving is deter-mined in order to follow it again. To analyze the performance of the proposed system, the CoppeliaSim software was used with the Pioneer 3-DX robot model, where various simulations are carried out under different scenarios to which the robot would actually be exposed.*

**Keywords**: odometry; local path planning; aruco marker; global coordinates; people follow-ing robot.

## 1. Introducción

La robótica es una de las tecnologías que ha crecido considerablemente en los últimos años, muestra de esto es la necesidad de robots seguidores de personas que puedan interactuar y coexistir con las mismas (Morioka, Lee, & Hashimoto, 2004). Los escenarios de seguimiento de personas surgen cuando un humano y un robot autónomo colaboran en una tarea común que requiere que el robot siga al humano. Por lo general, el humano dirige la tarea y coopera con el robot durante la ejecución (Islam, Hong, & Sattar, 2019). Esta necesidad de brindar asistencia constantemente a las personas para facilitar sus vidas hace que se lleven a cabo implementaciones de técnicas de seguimiento de personas, pero para realizar esta tarea con precisión el robot necesita mecanismos que le permitan visualizar a la persona y tomar decisiones para realizar su tarea.

De acuerdo con un estudio realizado por BMC *Health Services Research* (Westbrook, Duffield, Li, & Creswick, 2011) el incremento en la carga de trabajo de las enfermeras aumenta sus probabilidades de enfermarse en un 7 %. Por lo general, las enfermeras realizan múltiples actividades de manera simultánea, por lo cual es importante aligerar su carga de trabajo y que no dediquen todos sus esfuerzos en transportar los suministros médicos. Por estas razones en este artículo se propone un sistema automatizado de seguimiento de personas mediante robots.

La mayoría de los robots que siguen a las personas están equipados con cámaras, y la percepción se realiza mediante detección visual. Se utilizan otros sensores para medir con precisión la distancia y las actividades (caminar, agitar las manos, etc.) de la persona para una navegación e interacción segura, como el telémetro láser, sensores infrarrojos, ultrasónicos y cámaras estéreo. La elección de los sensores a menudo está determinada por el entorno operativo, es decir, interior o exterior. Por ejemplo, los sensores RGBD son muy efectivos en ambientes interiores (Islam, Hong, & Sattar, 2019). Cabe destacar que el procesamiento de imágenes involucra considerables recursos computacionales para lograr que se realice en tiempo real. Dicho factor hace compleja una implementación sencilla que logre los objetivos.

Este documento presenta el desarrollo de un robot con un sistema de seguimiento de una persona, especialmente las enfermeras en los hospitales. El mismo es capaz de distinguir el objetivo humano de otros mediante los marcadores aruco. El seguimiento de la persona se realiza bajo condiciones de presencia de obstáculos tanto estáticos como dinámicos y pérdida del objetivo rastreado, al moverse alrededor de las esquinas haciendo que desaparezca del campo de visión del robot, el cual fue probado en una simulación.

El sistema propuesto emplea la odometría de rueda para la localización del robot. El usuario o cliente es identificado a través de un marcador aruco vestible en la espalda el cual tiene un identificador asociado a la persona. Con el controlador P y PD se logra el seguimiento de la persona manteniéndolo en el centro de la imagen capturada por una cámara monocular. Se emplea un algoritmo de planeamiento de la ruta local adaptativo para computar la ruta a seguir por el robot ante escenarios de presencias de obstáculos o al perder a la persona objetivo. De igual manera, se recurre a las técnicas de proyección de puntos en una cámara, desde el sistema de coordenadas del objeto al plano de la imagen para la extracción de las coordenadas de la persona con respecto al sistema de coordenadas del robot.

El artículo está organizado de la siguiente manera. En la siguiente sección la metodología empleada para el sistema propuesto es discutida en detalle, explicando cada uno de los algoritmos y tecnologías utilizadas para lograr un sistema eficiente. Luego, se presenta la simulación diseñada y los resultados obtenidos evaluando el desempeño de cada sistema por separado. Finalmente, se plantean las conclusiones del artículo.

## 2. Metodología

La estrategia de trabajo utilizada para este artículo se basó en el diseño de cada uno de los módulos o algoritmos necesarios para conseguir un robot seguidor de personas completamente funcional de manera separada y luego fueron integrados acorde con el diseño planteado, tomando en cuenta las características de los dispositivos seleccionados para la realización del prototipo. La evaluación de desempeño de cada uno de los módulos se realizó por medio de la simulación en CoppeliaSim haciendo

uso del modelo de robot Pioneer 3DX. El diseño del ambiente de simulación se realizó considerando los elementos más comunes que pueden encontrarse en áreas hospitalarias, debido a que la solución propuesta por este artículo ha sido diseñada para este tipo de escenarios.

El desarrollo de esta sección se divide en seis partes: arquitectura del sistema propuesto, identificación de marcadores aruco, seguimiento de la persona con controlador P y PD, localización del robot, planeamiento de la ruta local y extracción de las coordenadas 3D de la persona. En las siguientes secciones se detalla cada parte.

## 2.1. Arquitectura del sistema propuesto

El sistema propuesto está descrito en la figura 1, donde la entrada al sistema es una imagen RGB proveniente de una cámara monocular. Esta imagen es procesada por un algoritmo de lectura de marcadores aruco del cual se computan: los identificadores presentes, coordenadas en pixeles de las esquinas del marcador y del centro, y un "flag" que indica la presencia o ausencia de la persona de interés. Si el usuario está presente en la escena un controlador P y PD se usa para doblar de tal manera que se mantenga al usuario en el centro de la imagen. Si en la escena el robot encuentra obstáculos a una distancia menor a la que está de la persona se usa el planeamiento de la ruta local para evadirlos mientras sigue a la persona objetivo.

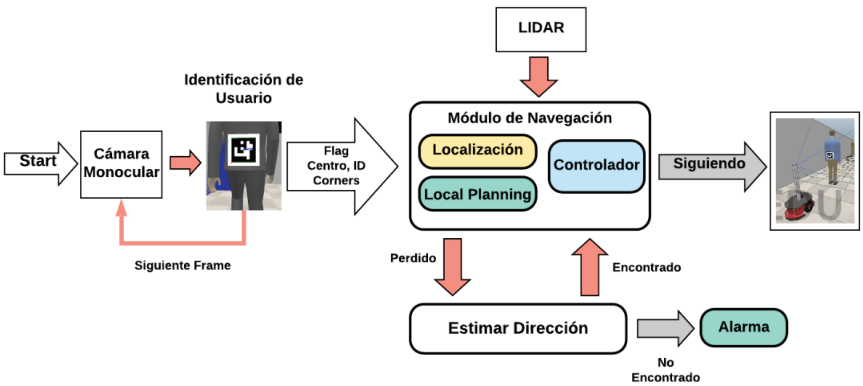



**Figura 1.** Esquema de diseño del sistema propuesto

En el módulo de navegación es importante extraer las coordenadas 3D de la persona y la localización del robot de manera recurrente para hacer uso del *local path planning* o "planeamiento de la ruta local". Cuando el robot no puede ver a la persona, se mueve hasta la última posición registrada y luego estima la posible dirección a la cual se dirigió el usuario para doblar en esa dirección; si vuelve a ver al usuario retorna al módulo de navegación, de lo contrario, si por alguna razón luego de haber hecho esto aún no puede ubicar al usuario en la escena, se declara la persona como perdida. Algunos de los casos en los que es posible que no se encuentre el objetivo incluyen cuando el objetivo se escapa después de doblar o gira inesperadamente en otro lugar. En todos estos casos es razonable suponer que el robot no podrá encontrar el objetivo; un comportamiento similar se esperaría si un humano sigue a otro humano.

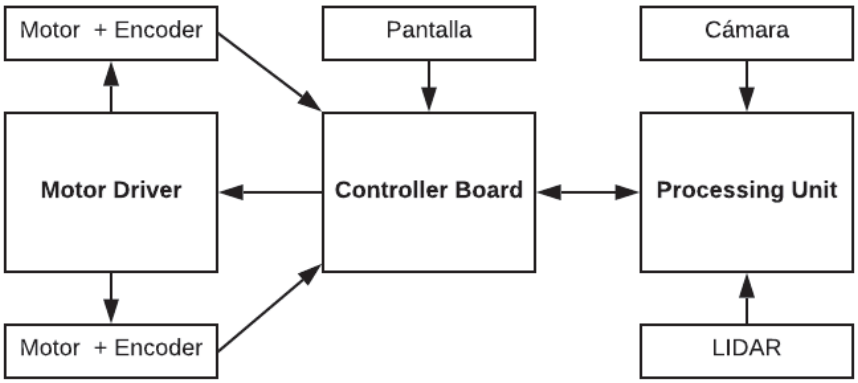



**Figura 2.** Diagrama general del hardware

En la figura 2 se puede observar el diagrama general correspondiente a la construcción física del sistema. El movimiento del robot está controlado por dos motores de engranajes de corriente continua (CC) con un codificador. Los dos motores se accionan con un controlador de motor o "Motor Driver" conectado a una placa controladora o "Controller Board" que envía señales de voltaje al controlador del motor para controlar la velocidad lineal y angular del robot. El codificador del motor está conectado a la placa controladora para contar el número de rotaciones del eje del motor, y con base en estos datos poder estimar la posición y velocidad actual del robot. Por otro lado, la pantalla muestra informaciones del estado de la batería y estado actual del robot.

El LIDAR (*Laser Imaging Detection and Ranging*) se conecta por el puerto USB a la unidad de procesamiento para obtener información de los obstáculos en el entorno, de manera que pueda procesar la ruta necesaria para evadir los obstáculos presentes. La placa controladora se mantiene en comunicación con la unidad de procesamiento donde se realiza todo el procesamiento de alta gama en el robot, que incluye el procesamiento de las imágenes obtenidas por la cámara y el LIDAR.

## 2.2. Identificación de marcadores aruco

Un marcador o *Aruco Marker* es un marcador cuadrado compuesto por un borde negro ancho y una matriz binaria interna que determina su identificador (id). El borde negro facilita su detección rápida en la imagen y la codificación binaria permite su identificación y la aplicación de técnicas de detección y corrección de errores (Garrido-Jurado, Muñoz-Salinas, Madrid-Cuevas, & Marín-Jiménez., 2014). El tamaño del marcador determina el tamaño de la matriz interna. Por ejemplo, un tamaño de marcador de 6x6 está compuesto por 36 bits como se muestra en la figura 3 (OpenCV team, s.f.). Para la identificación y lectura de los marcadores se usa la librería Aruco creada por Rafael Muñoz y Sergio Garrido, la cual retorna la posición de las cuatro esquinas en la imagen de los marcadores detectados para luego computar el centro en pixeles del marcador.

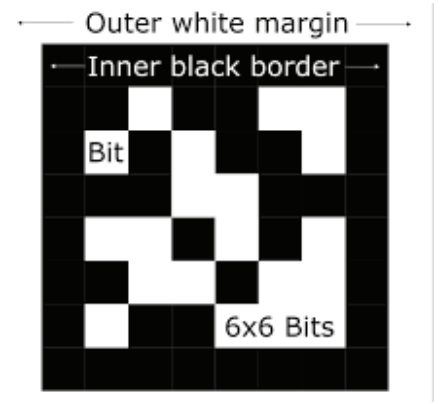



**Figura 3.** Estructura de un Marcador Fiduciario
**Fuente:** (Kunz, Hein, Genten, & Meibner, 2019)

## 2.3. Localización del robot

La localización del robot requiere estimar su posición con respecto a un marco de coordenadas global. En el caso 2D, se trata de las coordenadas X, Y y la orientación, θ del robot. El robot debe mantener una estimación de su posición a medida que se mueve en el entorno. En este artículo la localización se resuelve mediante odometría de rueda. La odometría en un vehículo con ruedas como un automóvil se puede mejorar midiendo la rotación de las ruedas con un encoder. La circunferencia de una rueda es 2πr, donde r es el radio de la rueda en cm, por lo que, si se cuentan n rotaciones, se calcula que el robot se ha movido 2πnr cm.

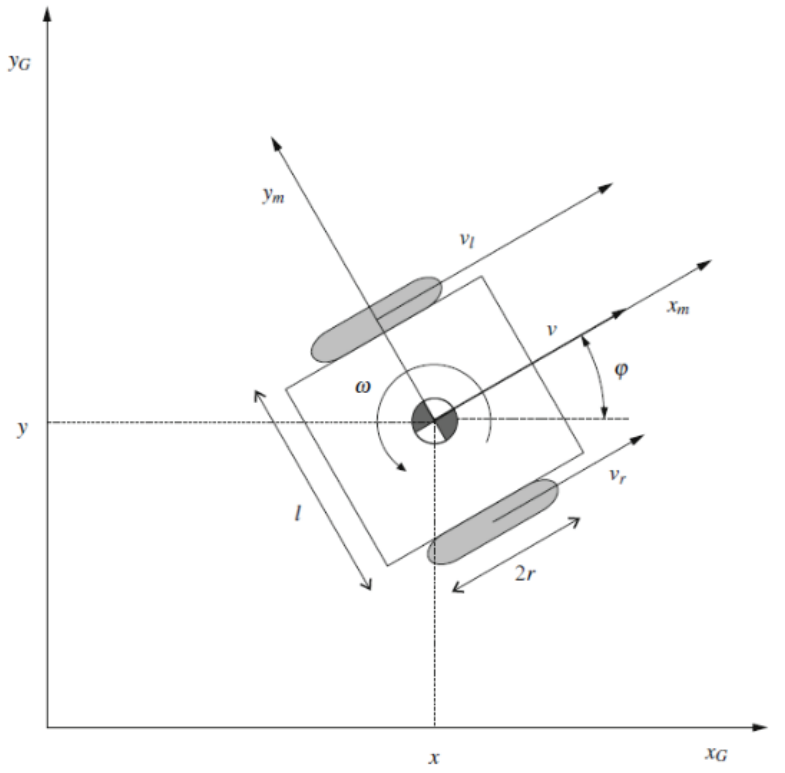



**Figura 4**. Cinemática del robot de accionamiento diferencial
**Fuente:** (Mihelj, Bajd, & Ude, 2019)

Es importante tomar en consideración las relaciones entre las velocidades angulares de las ruedas y la velocidad lineal del robot móvil de accionamiento diferencial, de acuerdo con las ecuaciones que representan la cinemática del vehículo, donde entran en juego cada una de las variables de la figura 4. (Mihelj, Bajd, & Ude, 2019).

La odometría de las ruedas es confiable para distancias cortas con un error de menos del 4 % para entornos con una superficie lisa (por ejemplo, pisos interiores, pavimentos exteriores, aceras, etc.). En el contexto de este artículo solo se considera que la posición del robot sea precisa durante un corto período de tiempo. Este es el momento en que se requiere la información de localización del robot para calcular la ruta local del objetivo, entonces los errores previamente acumulados debido al efecto *dead reckoning* son despreciables, los detalles de la precisión se especifican en la sección de resultados.

## 2.4. Seguimiento de la persona con controlador P y PD

Para simular el movimiento del robot cuando la persona está en el campo de visión, se asumió un modelo cinemático, teniendo como variable de control la velocidad de cada rueda de manera independiente, como se muestra en las ecuaciones 1 y 2. De esta manera, cuando la persona está en el campo de visión de la cámara se estimó que el control pudiera lograr una velocidad angular proporcional al error en la orientación y su tasa de cambio, e igual lograra una rapidez de traslación proporcional al error de distancia con la persona (esta última negativa, debido a que la distancia medida desde la persona tiene sentido contrario al desplazamiento del robot).

$$v\,(k+1) = -Ka * (D_{ref} - d(k)) \tag{1}$$

$$\omega(k+1) = Kb * \hat{\theta}\,(k) + Kc(\hat{\theta}(k) - \hat{\theta}(k-1)) \tag{2}$$

Donde:

**d**: es la distancia en metros medida por el sensor.

$\mathbf{D_{ref}}$: es la distancia deseada respecto a la persona (0.5 m en las simulaciones).

ω: es la velocidad angular del robot, accionada a partir de las velocidades de las ruedas.

**v**: es la velocidad lineal del robot.

$\hat{\theta}$: es la estimación del ángulo del objetivo respecto a la orientación del robot, con base en cantidad de pixeles desde el centro de la imagen.

***Ka, Kb y Kc***: son parámetros sintonizados a partir de pruebas sucesivas en simulación.

Sin embargo, para poder accionar el robot se transformaron estas variables (velocidad traslacional y angular del robot) en variables controladas que son: la velocidad de la rueda izquierda  y de la rueda derecha . Para ello se utilizan las ecuaciones de transformación (Mihelj, Bajd, & Ude, 2019) .

Para un correcto funcionamiento del robot solo se debe mover en el eje horizontal cuando la persona se haya movido lo suficiente, de lo contrario se mantendría oscilando, por eso se obtuvo una relación entre la variación máxima de pixeles que se requiere antes de aplicar el controlador de la velocidad angular y la distancia entre el robot y la persona.

## 2.5. Planeamiento de la ruta local

El planeamiento de la ruta global o *global path planning* requiere que el entorno sea completamente conocido y el terreno debe ser estático. En este enfoque, el algoritmo genera una ruta completa desde el punto de inicio hasta el punto de destino antes de que el robot comience su movimiento. Por otro lado, el *local path planning* o planificación de ruta local significa que la planificación de ruta se realiza mientras el robot está en movimiento; en otras palabras, el algoritmo es capaz de producir un nuevo camino en respuesta a los cambios en el entorno. Suponiendo que no hay obstáculos en el área de navegación, el camino más corto entre el punto inicial y el punto final es una línea recta (Sedighi, Ashenayi, & Manikas, 2004).

El algoritmo propuesto para este artículo trata el problema de desempeño en el algoritmo de BUG, pues este busca seguir la frontera del obstáculo encontrado hasta que una condición de abandono se cumpla, generando largas trayectorias, al tiempo que evita las incertidumbres en la construcción del mapa local presentado en el algoritmo VFH (Histograma de Campos Vectoriales). Este método o algoritmo es el *Heading Weight Function* (HWF), el cual toma en consideración la suma de los rayos láser que incluyen el subconjunto más cercano de obstáculos presentes dentro de un radio predefinido, lo que lo hace robusto a los errores en las lecturas del sensor e independiente de la forma, ubicación y naturaleza de un obstáculo (estático o dinámico). Para mantener el rumbo global del robot hacia el objetivo, el HWF también se combina con el controlador del robot teniendo en cuenta la posición del objetivo, asegurando que el robot llegue a su destino. El objetivo del *Heading Weight Function* (HWF) es determinar en qué dirección debe navegar el robot cuando hay un obstáculo. Se utilizó un LIDAR como sensor activo para la detección de obstáculos (Korsaeth, 2019). El sensor tiene un campo de visión (FOV) de 180 grados en la simulación, pues solo es importante tener la visión del entorno en el frente del robot para estimar una ruta.

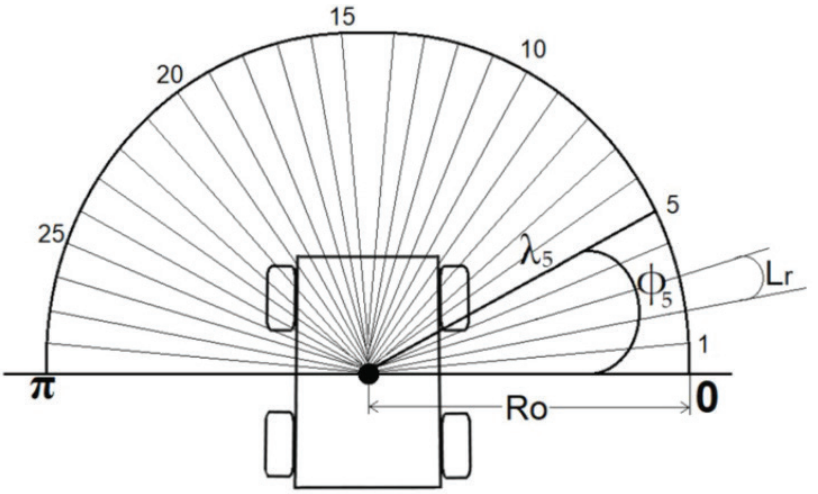



**Figura 5.** Representación de los haces de LIDAR en el robot móvil
**Fuente:** (Korsaeth, 2019).

En la figura 5 se observa la representación de los haces del LIDAR que cubren 180 grados, cabe destacar que la cantidad de haces es solo ilustrativa porque la cantidad real es mucho mayor, en el caso de la simulación son 684 haces. El ángulo que separa cada haz se llama resolución del LIDAR y se denota en este documento como Lr. Las lecturas de LIDAR se asignan a un sistema de coordenadas polares. Cada haz se caracteriza por las variables Φi y λi, que son el ángulo y la distancia medida del número de haz i, respectivamente. Φi se calcula según la ecuación 3.

$$\Phi i = i * Lr \tag{3}$$

En caso de la presencia de un obstáculo solo se toman en consideración los haces del LIDAR que son menores a un radio Ro, este radio va a depender de la distancia entre la persona y el robot siendo esta distancia Ro, es decir, solo se desean obtener los obstáculos que están detrás de la persona para no confundirla con un obstáculo. En este mismo orden, el HWF incluye los haces del LIDAR que cumplan la condición de la ecuación 4.

$$\boldsymbol{\lambda i} = \begin{cases} \boldsymbol{\lambda i}, & \boldsymbol{\lambda i} < \mathbf{Ro} \\ \infty, & \boldsymbol{\lambda i} >\geq \mathbf{Ro} \end{cases} \tag{4}$$

El HWF tiene una función de pesos, donde M es el número de haces del LIDAR que satisfacen la ecuación 5:

$$\mathbf{H}(\boldsymbol{\Phi}, \boldsymbol{\lambda}) = \sum_{i=0}^{M} \mathbf{sin}(\boldsymbol{\Phi i}) * \mathbf{sgn}(\mathbf{cos}(\boldsymbol{\Phi i})) / \boldsymbol{\lambda i} \tag{5}$$

donde sgn (x) es la función signum.

Primero, el resultado sinusoidal de cada ángulo del haz se explica por el hecho de que cuanto más cerca esté el obstáculo del centro del robot, mayor será el resultado sinusoidal. Esto significa un mayor impacto del HWF inicial en los controladores del robot. En segundo lugar, el resultado del término sinusoidal se multiplica por el término , simplemente

para saber si el haz está en el lado derecho o izquierdo del robot. Finalmente, la división entre , que es la longitud del número de haz i, significa que cuanto más cerca esté el obstáculo del robot (la longitud del haz es pequeña), mayor será el HWF (Korsaeth, 2019) de la ecuación 5, lo que significa un mayor impacto en los controladores del robot de la ecuación 6. Cabe destacar que en este artículo, para el planeamiento de la ruta local, el robot se mueve a velocidad constante de 0.6 m/s.

$$\omega=kr^{*}\alpha+k1^{*}\,H(\Phi,\lambda) \tag{6}$$

Donde:

$$\alpha = \beta - \theta \tag{7}$$

$$\beta = \operatorname{atan}\left(\frac{Yp - Yr}{Xp - Xr}\right) \tag{8}$$

El ángulo es la orientación del robot cuya posición es (,) y el ángulo es la orientación que tendría que tener el robot para alcanzar a la persona en línea recta en la posición (,). Se define una distancia predeterminada de 1.2 metros en la cual se considera que el robot ya llegó a la meta, es decir, que alcanzó a la persona. En la ecuación 6, y son ganancias positivas iguales a 1.1 y 0.01 respectivamente, dichos valores fueron obtenidos luego de varias pruebas de desempeño ante diversas situaciones.

Existen situaciones en las cuales la persona se sale de la zona de visión del robot por un lapso de aproximadamente 1 segundo, esto implica que ha doblado por una esquina o ha sido bloqueado por otra cosa. Por dicha razón, el robot siempre mantiene un historial de las coordenadas en pixeles de la persona, específicamente 50 posiciones, y luego en función de la dirección descrita, el robot gira por 5 segundos hasta encontrar a la persona; si se cumple el lapso y aún no la encuentra se declara perdida y se envía una alarma.

En las ocasiones en las cuales la persona ingresa a una habitación, el robot debe tener mayor cuidado para no colisionar al entrar, pues las

puertas suelen ser estrechas. Debido a esto, se adapta el algoritmo de planeamiento de la ruta local al ambiente,  y  son iguales a 1.1 y 0.006 respectivamente cuando se pierde la persona cerca de una puerta. El reconocimiento de las puertas se logra mediante YOLO (*You Only Look Once*, «sólo se mira una vez») que es un algoritmo de visión artificial que detecta y clasifica objetos en tiempo real. Se empleó una red pre-entrenada, para correr dicho algoritmo (Redmon & Farhadi, 2018).

Cada imagen que se pasa al algoritmo se reduce a una imagen de 416 x 416 pixeles, y luego se procesa la salida de cada una de las tres capas del algoritmo. El número de cuadros delimitadores o *bounding boxes* predichos por YOLO v3 es de 10,647 en total (Redmon & Farhadi, 2018), pero se toman como valores de interés aquellos donde la probabilidad de que el objeto pertenezca a una clase sea mayor a 0.5. Cuando se realiza la detección, sucede que se tienen varios *bounding boxes* del mismo objeto, producidos por las celdas adyacentes, por esto se emplea el *Non-maximum Suppression* que extrae las propuestas con el puntaje de confianza más alto que el umbral (0.5), luego se calcula el IOU (Intersección sobre Unión) de esta propuesta con cualquier otra propuesta permaneciendo aquellos menores a 0.4.

### 2.6. Extracción de las coordenadas 3D de la persona

La calibración de la cámara es importante para relacionar las mediciones de la cámara con las mediciones en el mundo real tridimensional. Esto permite determinar la relación entre las unidades naturales de la cámara (píxeles) y las unidades del mundo real (centímetros).

Los parámetros de la cámara incluyen intrínsecos, extrínsecos y coeficientes de distorsión. Para estimar los parámetros de la cámara, se debe tener puntos mundiales 3-D y sus puntos de imagen 2-D correspondientes. Estas correspondencias se obtienen utilizando múltiples imágenes de un patrón de calibración, como un tablero de ajedrez, que fue el patrón utilizado para el artículo. Usando las correspondencias, se resuelven los parámetros de la cámara. Después de calibrar la cámara, se evalúa la precisión de los parámetros estimados con el error de reproyección.

En *OpenCv* se utiliza el modelo de la cámara de *pinhole* o cámara estenopeica, que es una cámara simple sin lente y con una sola abertura pequeña. Los rayos de luz atraviesan la abertura y proyectan una imagen invertida en el lado opuesto de la cámara, conocido como *image plane* (Hata & Savarese, 2015). Los puntos mundiales o *world coordinates* se transforman en coordenadas de cámara utilizando los parámetros intrínsecos. Las coordenadas de la cámara se mapean en el plano de la imagen utilizando los parámetros intrínsecos, como se muestra en la figura 6.

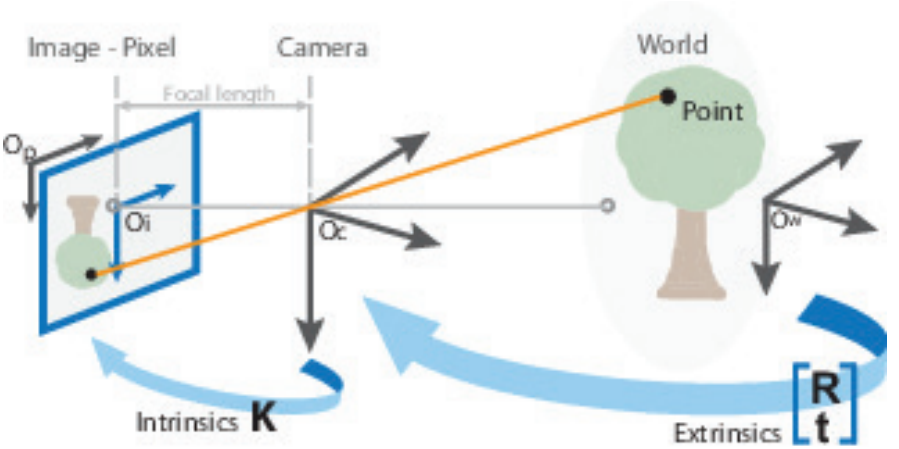



**Figura 6.** Parámetros de la cámara
**Fuente:** (mathworks.com)

Los parámetros intrínsecos incluyen la distancia focal expresada en unidades de pixeles y el centro óptico. En el caso de *OpenCV*, tiene cuatro parámetros intrínsecos (fx, fy, cx, cy,) y cinco parámetros de distorsión: tres radiales (k1, k2, k3) y dos tangenciales (p1, p2). La matriz intrínseca de la cámara involucra las ecuaciones 9 y 10 *(OpenCV.com).*

$$matriz\ de\ cámara = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

$$\text{coeficientes de distorsión} = (k1\ \ k2\ \ p1\ \ p2\ \ k3) \tag{10}$$

La posición de la persona en coordenadas globales con respecto al sistema de coordenadas de la cámara se estima teniendo en cuenta los parámetros intrínsecos de la cámara. Para calcular la posición 3D de un objeto en una imagen se requieren tres partes esenciales: las coordenadas 2D de algunos puntos, localización 3D de esos puntos en el sistema de coordenada del objeto y los parámetros intrínsecos de la cámara antes mencionados, dicha representación se muestra en la figura 7.

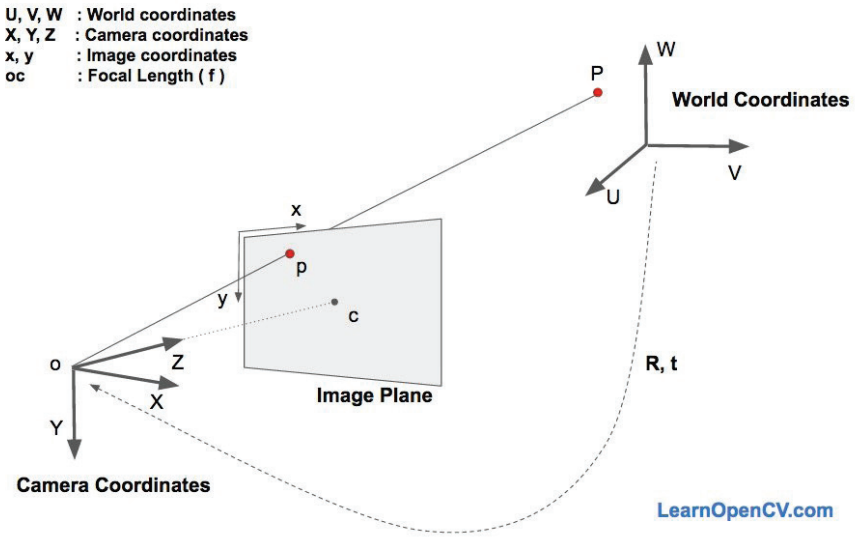



**Figura 7.** Formación de una imagen
**Fuente:** (Bradski & Kaehler, 2016)

En las relaciones de las ecuaciones 11 y 12 se observa que al conocer la matriz de rotación (R) y el vector de traslación (t) se puede conocer cualquier punto de las coordenadas mundiales con respecto a las coordenadas de la cámara.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = [R \mid t] \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix} \tag{11}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} r00 & r01 & r02 & tx \\ r10 & r11 & r12 & ty \\ r20 & r21 & r22 & tz \end{bmatrix} \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix} \quad (12)$$

Para este caso se conocen algunos puntos en el modelo 3D (es decir, (U, V, W)) mostrados en la ecuación 11, pero no se tiene conocimiento de (X, Y, Z). En ausencia de distorsión radial, las coordenadas (*x, y*) del punto p en las coordenadas de la imagen se determina con la ecuación 13.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (13)$$

Debido a que hay que tomar en cuenta la distorsión del lente y el modelo se vuelve más complejo, se decidió usar cv2.solvePnP con el *flag solvepnp_iterative* que es un modelo iterativo para resolver las ecuaciones, basándose en la optimización de Levenberg-Marquardt. En este caso, la función encuentra una posición que minimiza el error de reproyección, es decir, la suma de las distancias al cuadrado entre los puntos de imagen proyectados observados y los puntos de objeto proyectados (usando projectPoints) (OpenCV team, s.f.).

Para llevar estas coordenadas al sistema de referencia del robot, hay que considerar el eje de las x y de z, (u,v) para obtener el ángulo de la figura 8 con la tangente inversa. Cabe destacar que cuando la suma de los ángulos sea mayor a 90 grados se toma de referencia para los cálculos el ángulo suplementario entonces según lo mostrado en las ecuaciones 14 y 15.

$$X = x - d^{*}\cos(180 - (\theta + \varphi)) \quad (14)$$

$$Y = y + d^{*}\sin(180 - (\theta + \varphi)) \quad (15)$$

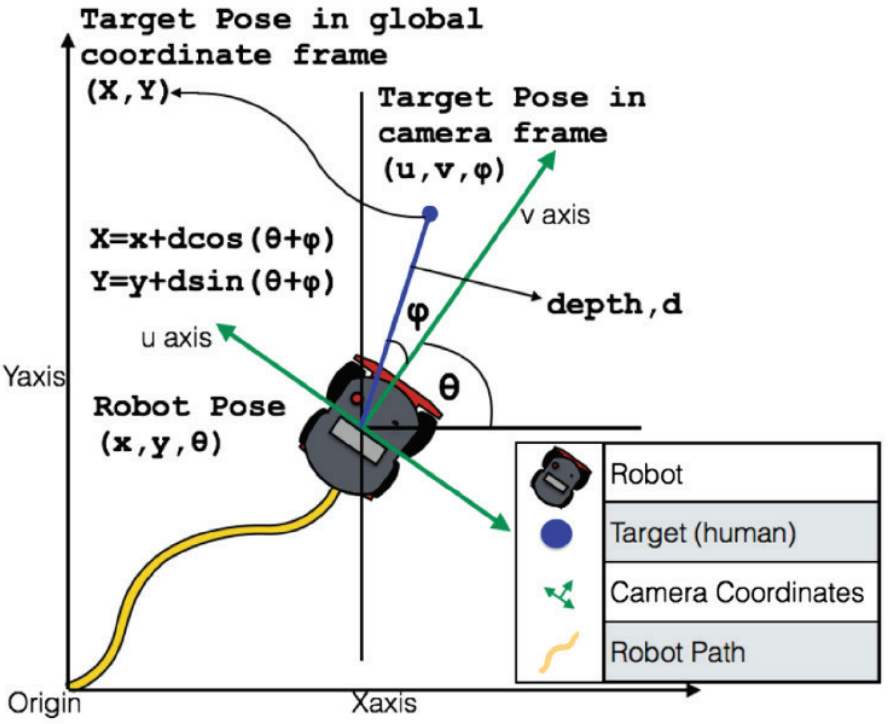



**Figura 8.** Estimación de la posición de la persona en referencia al robot
**Fuente:** (ResearchGate.com)

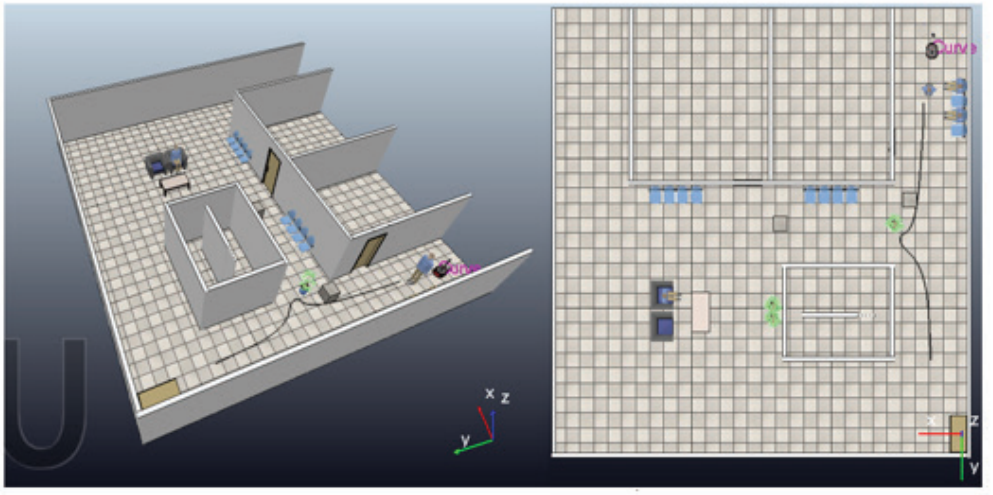
## 3. Simulación y resultados

El simulador de robot CoppeliaSim, con un entorno de desarrollo integrado, se basa en una arquitectura de control distribuido: cada objeto/modelo puede controlarse individualmente mediante un *script* incrustado, un complemento, un nodo ROS o BlueZero, un cliente API remoto o una solución personalizada (Coppelia Robotics, n.d.).

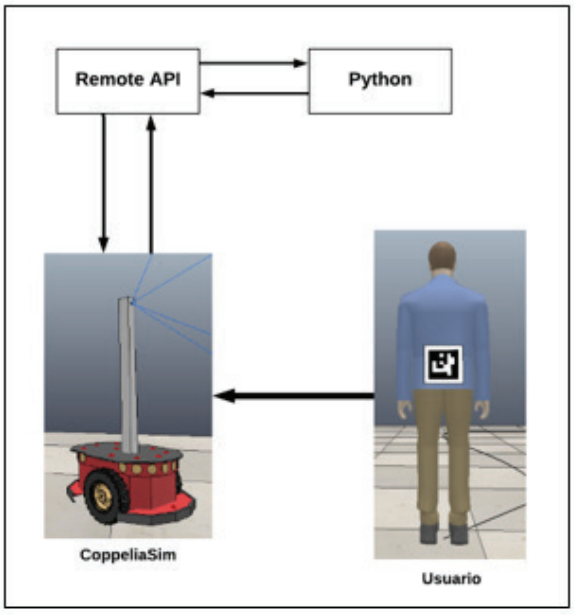
Los elementos representados en la simulación son sillas en los pasillos, sillones y mesa en un área de espera, plantas, un armario, personas detenidas y en movimiento y pequeños bloques de concreto en representación de obstáculos que puedan estar presentes. El ancho de los pasillos es de 2.20 m según las exigencias de las normativas CTE.SI y CTE.SUA, los otros elementos representados en la simulación han sido tomados de los modelos ofrecidos por el software de simulación. En la figura 9-A se muestra el entorno descrito anteriormente.

En la figura 9-B se muestra el esquema empleado para la simulación del artículo donde la API remota permite la comunicación entre CoppeliaSim y una aplicación externa (Python), pues todo el procesamiento se lleva a cabo en un *script* de Python. Al modelo de robot escogido, por ser muy pequeño, se le colocó una pieza para aumentar la altura a la cual se instala la cámara, de tal forma que pueda visualizar el marcador situado en la espalda del usuario. El modelo de robot seleccionado para la simulación es el Pioneer 3-DX es un pequeño robot con dos motores acoplados con sus respectivas ruedas, y una rueda loca de soporte. Este modelo posee 16 sensores ultrasónicos (8 orientados hacia adelante, 8 orientados hacia atrás) para mediciones de proximidad, tiene una velocidad de rotación de 300°/s y la máxima velocidad de avance y retroceso es de 1.2 m/s.

Los sensores ultrasónicos situados en el robot no se estarán utilizando, sino que se agregó un LIDAR con un modelo distinto al seleccionado en la arquitectura, el cual corresponde al modelo Hokuyo LiDAR con una resolución angular de 0.35 grados. El sensor de visión tiene una resolución de imagen de 512x512MP y un ángulo de perspectiva de 62.2 grados.



**(A)**




**(B)**

**Figura 9.** Simulación del sistema. En (A), entorno de simulación en CoppeliaSim y en (B) Diagrama del prototipo simulado

### 3.1. Seguimiento de la persona mediante un controlador P y PD

En la figura 10 se observa el comportamiento del controlador de la velocidad angular para lograr centralizar a la persona en la zona de visión del robot, siendo el centro en pixeles igual a 256 (*setpoint*), pues en la simulación se utiliza una cámara de 512 x 512 pixeles. El sistema responde rápidamente, le tarda menos de un segundo llegar a su *setpoint* o estabilizarse y posee pocas oscilaciones en comparación con las demás gráficas que se obtuvieron en las simulaciones.

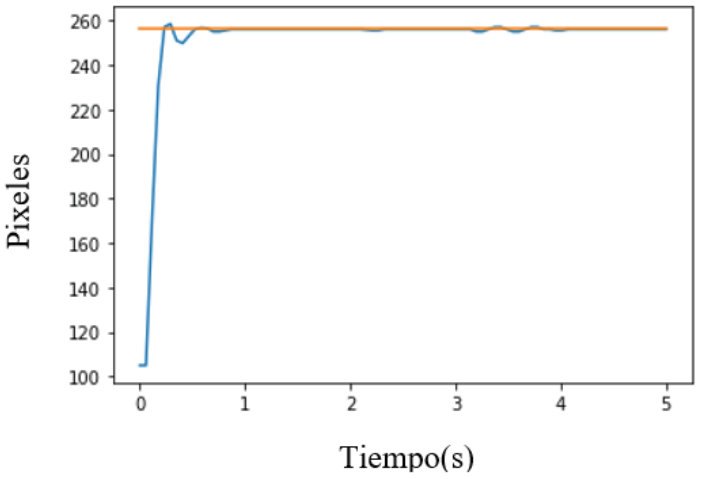
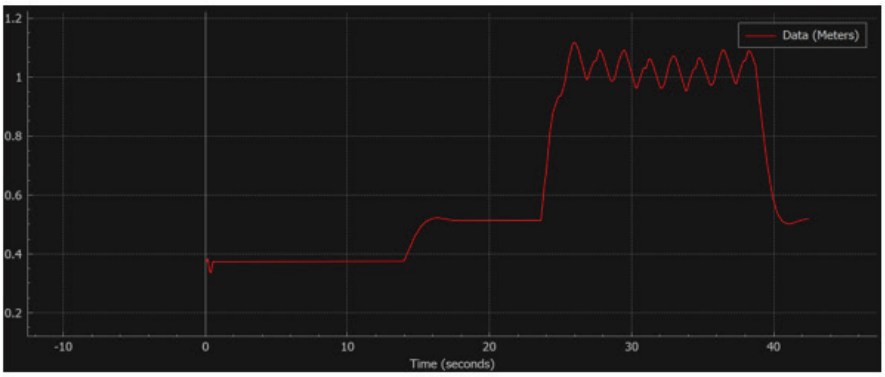


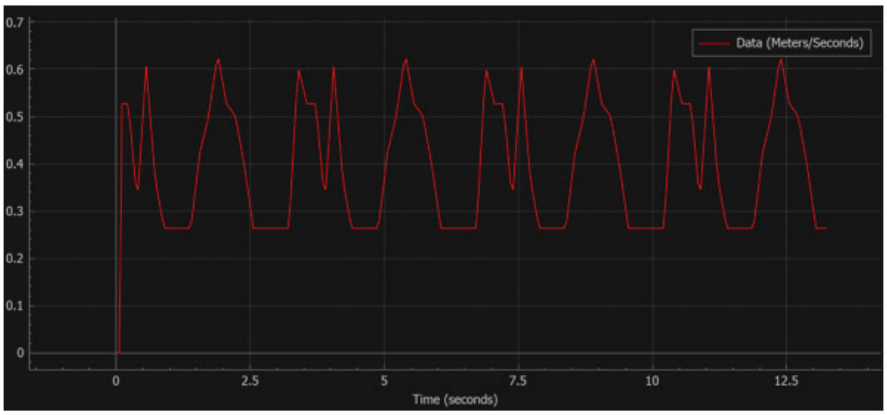

**Figura 10.** Gráfico de la respuesta del controlador de la velocidad angular

Por otro lado, en la figura 11-A se observa el comportamiento del controlador durante una ruta de seguimiento, los primeros segundos representa el inicio de la simulación, y luego se muestra que el sistema con el controlador P, es estable y no tiene prácticamente *overshoot*, lo cual permite que al detenerse de repente la persona, el vehículo no se acerque mucho como para colisionar y luego retroceder. En la figura 11-B se encuentra la velocidad de la persona durante el trayecto, es importante observar que para ese escenario el robot se mantiene aproximadamente a 1 metro de distancia de la persona, debido a que durante el movimiento de la persona el *setpoint* del controlador cambia constantemente. Esto significa que mientras más rápido camine la persona mayor será la distancia que se mantiene con el robot.

La elección de los diferentes parámetros o constantes del controlador se llevaron a cabo luego de varias pruebas y al graficar la respuesta del robot. Las pruebas consistieron en graficar las variaciones en la velocidad angular y la velocidad lineal del robot con el transcurso del tiempo hasta obtener una respuesta rápida y con pocas oscilaciones. Los valores resultantes son los siguientes:  = 0.8 para la velocidad lineal y para la velocidad angular  = 2/1000 y  = 1/1000.




(A)




(B)

**Figura 11.** Resultados de la simulación del seguimiento con el controlador. En (A), la respuesta del controlador de la velocidad lineal para el seguimiento y en (B), velocidad de la persona

En la figura 12 se observa el *path* seguido por el robot en color rosado, y el *path* negro es el correspondiente a la persona, lo cual significa que el robot sigue de manera correcta y con giros suaves, la trayectoria trazada por la persona con velocidades desde 0.5 hasta 1.1 m/s.

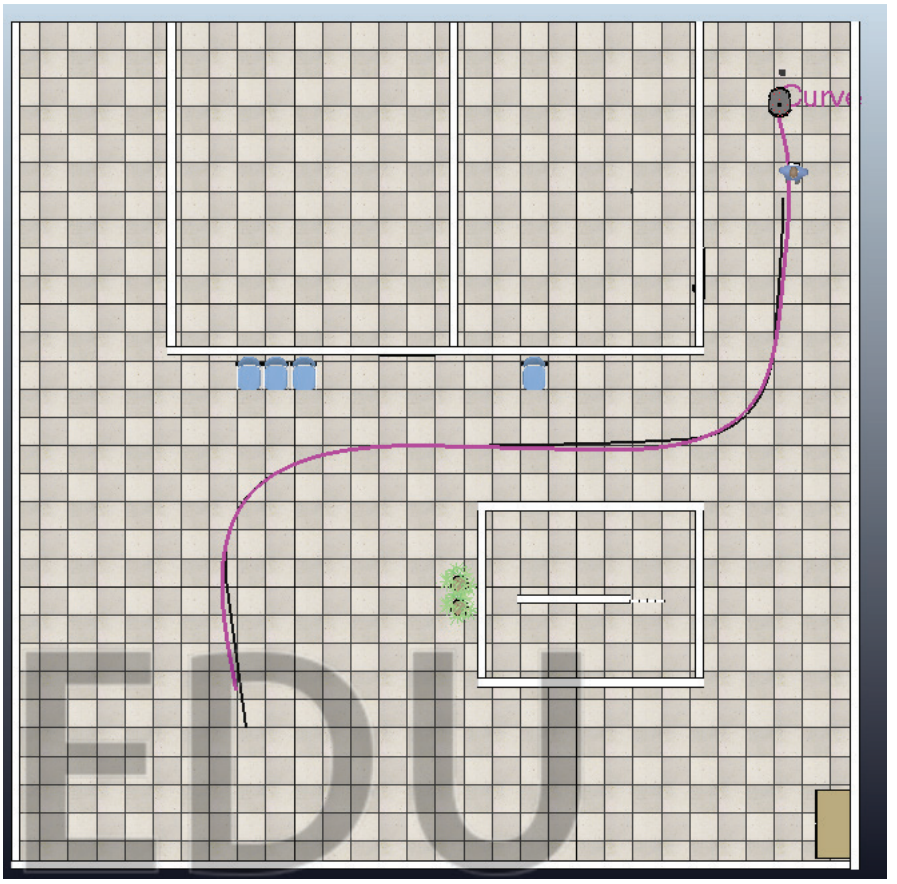


**Figura 12.** Resultados del seguimiento en la zona de visión con el controlador

### 3.2. Localización del robot

El sistema de odometría desarrollado se encuentra en un ambiente de simulación, por lo tanto, se les ha agregado ruido a las mediciones con el objetivo de crear un ambiente más cercano a la realidad. Se realizaron múltiples pruebas sometiendo el robot a distintos escenarios durante

un período aproximado de 20 segundos cada uno. Los resultados de las pruebas muestran que el sistema de odometría cuenta con un error de un 8 % en sus mediciones, siendo este error más propenso a suceder cuando se realizan trayectos con curvas muy pronunciadas o durante un período de tiempo prologado. En la figura 13 se muestra una gráfica que describe el comportamiento de la coordenada calculada con respecto a su coordenada real, se puede observar cómo la diferencia entre ambas coordenadas muestra una tendencia exponencial a medida que avanza el tiempo, siendo el color naranja la ubicación real y el color azul la ubicación calculada.

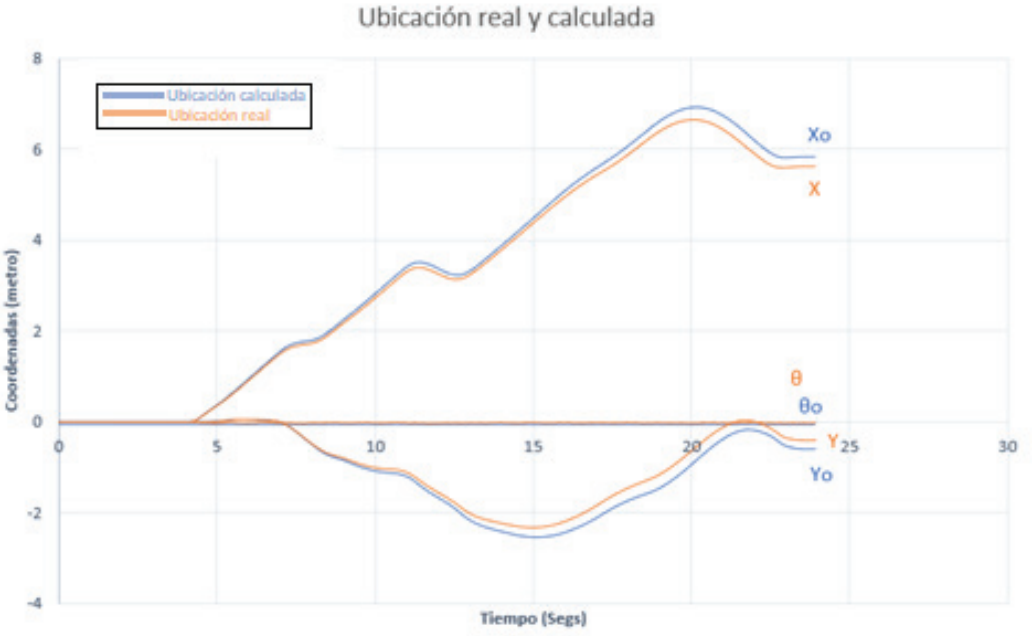



**Figura 13.** Figura 13. Comparación entre la ubicación real del robot y la calculada por el sistema de odometría

La trayectoria recorrida por el robot sobre la cual se calculó la diferencia entre las coordenadas se muestra en la figura 14 donde se observa que en trayectorias curvas la diferencia entre la posición calculada y la estimada es mayor.

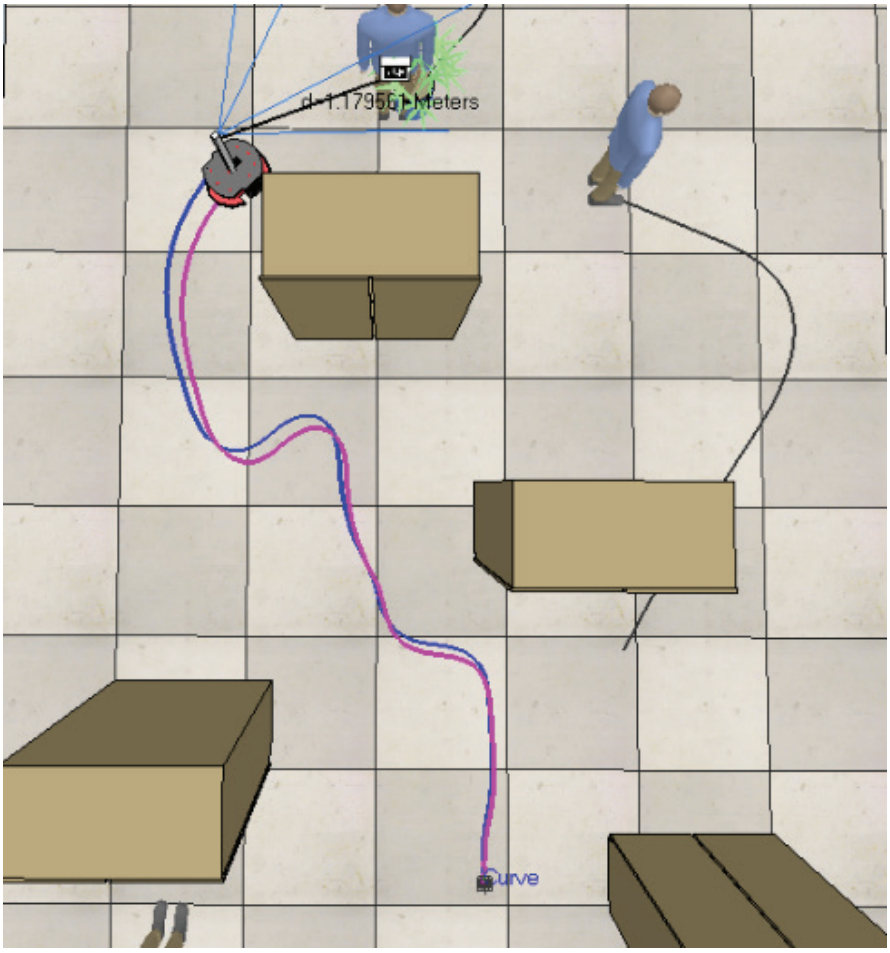



**Figura 14.** Trayectoria recorrida por el robot para el cálculo de la diferencia entre coordenadas

En el caso de que la trayectoria que el robot transite sea una línea recta o libre de curvas pronunciadas, los errores desarrollados en el proceso del cálculo de la odometría son básicamente imperceptibles. En la figura 15 se puede observar cómo las coordenadas reales y las coordenadas calculadas muestran pequeñas diferencias y las mismas van aumentando a medida que transcurre el tiempo, siendo el color naranja la ubicación real y el color azul la ubicación calculada.

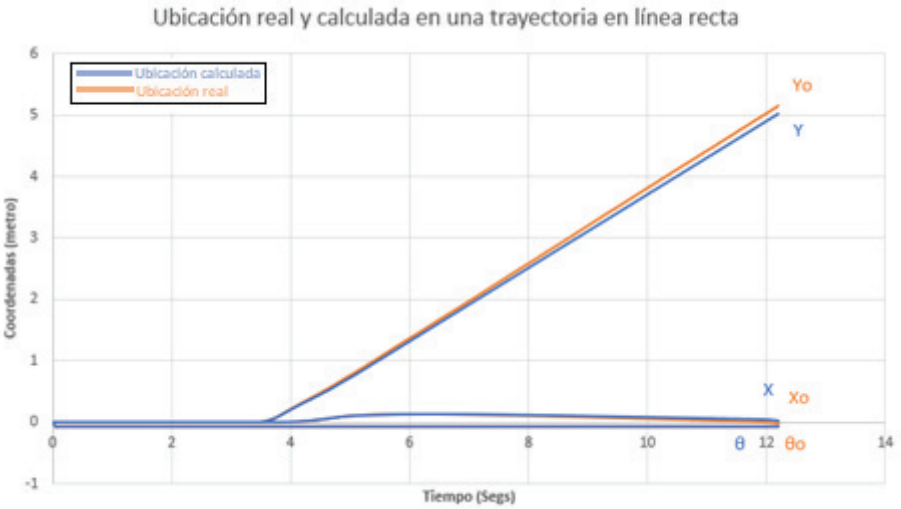



**Figura 15**. Diferencia entre las coordenadas reales y las calculadas en una trayectoria de línea recta

Con los resultados presentados anteriormente es posible afirmar que el sistema de odometría cumple con las condiciones necesarias para desempeñar sus funciones correctamente en los ambientes físicos para los cuales el robot ha sido diseñado.

### 3.3. Planeamiento de la ruta local

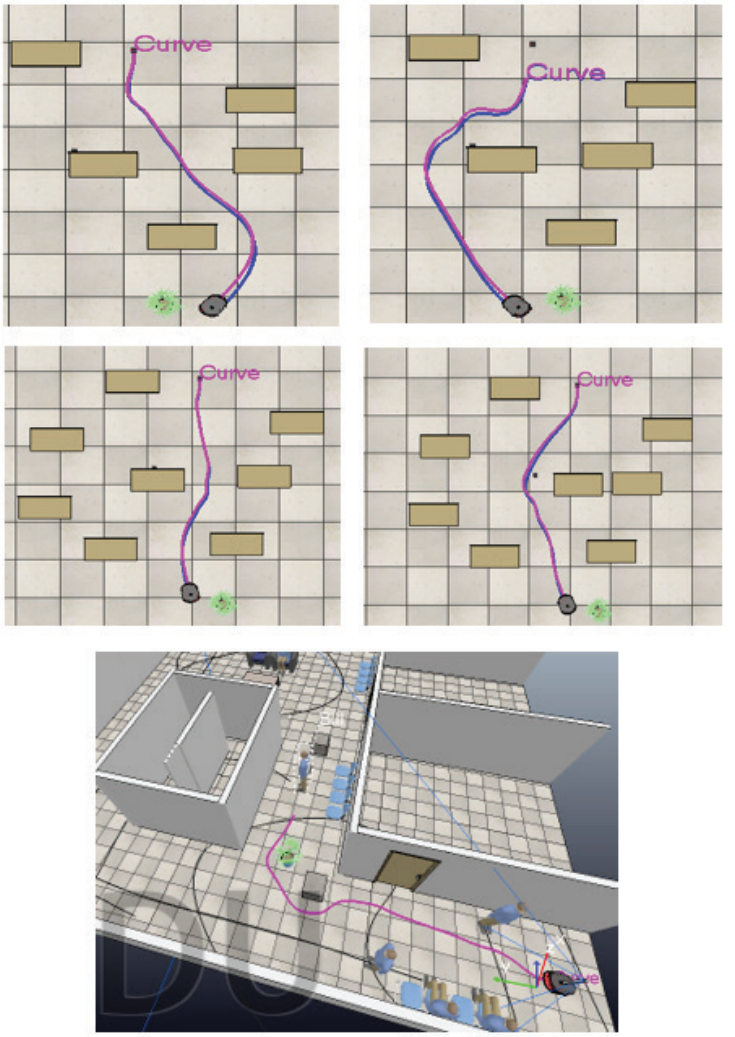



**Figura 16.** Resultados de simulación del planeamiento de la ruta local con objetos estáticos

El *local planner* fue probado a una velocidad de 0.6 m/s y en el escenario de prueba se colocaron varias cajas como objetos estáticos. Luego se le asignó al robot la tarea de conducir desde la posición inicial hasta el destino dado (punto de referencia) y, por lo tanto, evadir obstáculos en

el camino. Como primer paso, el robot recibió la tarea de navegar en un entorno con obstáculos estáticos solamente. La figura 16 muestra la ruta en color rosado del robot que se mueve desde una posición inicial hasta su destino (la planta situada en el suelo y la persona). Se puede ver que el robot pudo navegar con éxito alrededor de los obstáculos y llegar hacia su posición destino.

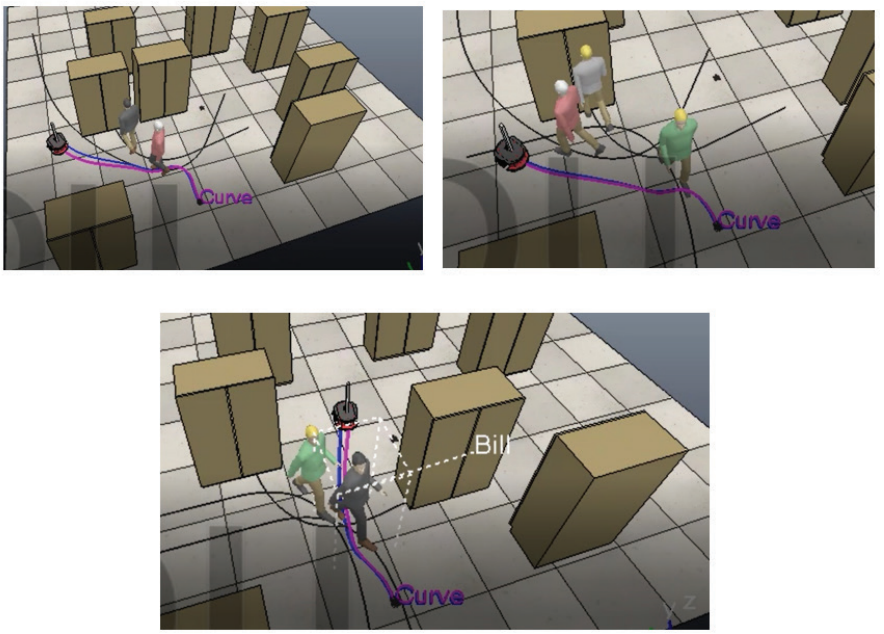



**Figura 17.** Resultados de simulación del local *path planning* con objetos dinámicos

El segundo paso fue probar la eficiencia del HWF en presencia de obstáculos dinámicos. Para hacerlo, se mantuvo la misma configuración que en el primer paso (mismo entorno y configuración de los obstáculos estáticos). Luego, durante la navegación del robot desde la posición inicial hasta el punto de referencia, varios humanos ingresaron a la escena y obstruyeron el camino del robot como se muestra en la figura 17.

En el momento en que los humanos ingresan en la escena obstruyendo el paso del robot, este se desvía y selecciona otro camino para llegar

al destino. Los resultados de las simulaciones confirmaron que el HWF propuesto proporciona un método eficiente y robusto para navegar en un ambiente interior con obstáculos estáticos. Además, las pruebas realizadas en un entorno simulado mostraron que el enfoque podría manejar situaciones con interferencia humana como obstáculos dinámicos.

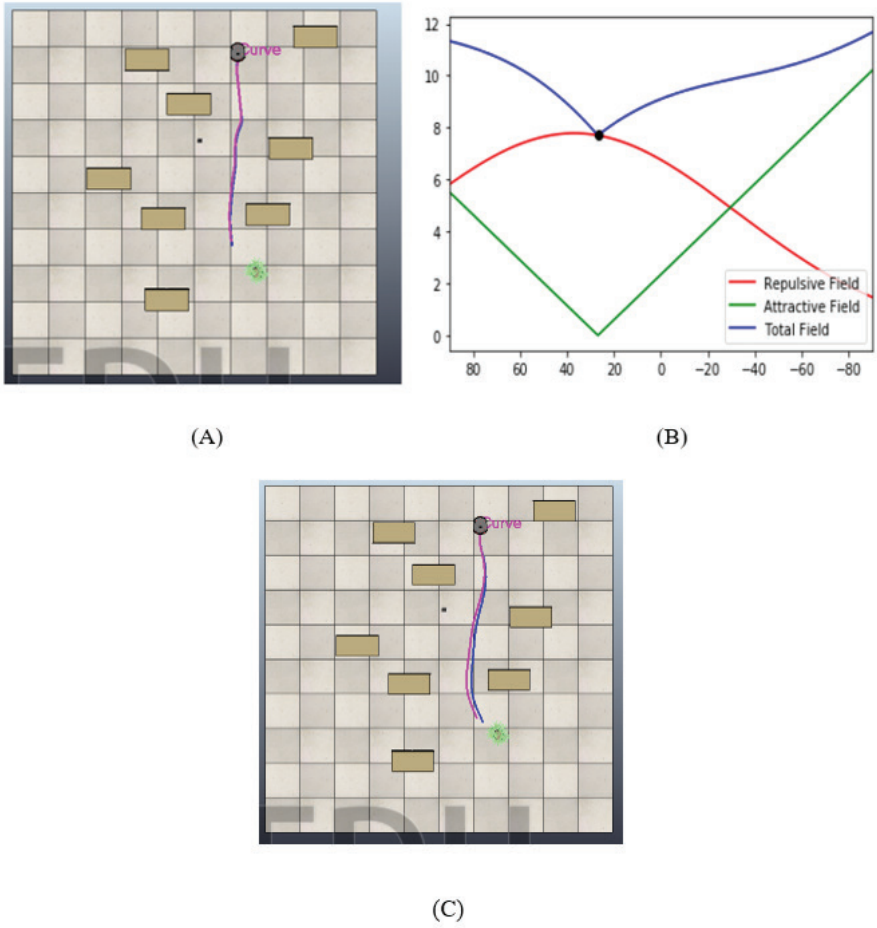



**Figura 18:** (A) Resultados de la simulación en entorno estático mediante ODGPF, (B) Campos potenciales de ODG-PF y (C) Resultados de la simulación en entorno estático mediante HWF

*The Obstacle-Dependent Gaussian Potential Field* (ODG-PF) es el otro método de *local planner* que se utilizó antes de seleccionar el HWF.

La idea principal detrás de este método es que, después de recibir datos de distancia del LIDAR, se considera solo los objetos que están dentro del rango de umbral (2 m, por ejemplo), aumentan los obstáculos con respecto al ancho del vehículo, y se construye un campo potencial gaussiano (repulsivo) a partir de ellos. A continuación, se calcula el campo de atracción a partir de la información extraída de la odometría, el campo total está formado por estos dos campos y, a partir de él, se elige el ángulo con el valor del campo total mínimo (Cho & Dong-Sung , 2018).

En la figura 18-A se muestra que con el método ODG-PG se obtuvo una buena ruta que a diferencia de HWF en figura 18-C es menos curva. ODG-PG se descartó porque habían situaciones en las cuales era muy propenso a colisionar, pues al estar muy cerca de la meta si se presenta un obstáculo justo detrás de la meta, el campo repulsivo era máximo pero el campo atractivo era mínimo, entonces el ángulo mínimo del campo total correspondiente al ángulo seleccionado por el algoritmo, provocaba que el vehículo colisionara como se observa en la gráfica de la figura 18-B, siendo el eje x la medida en ángulos y el eje y, el valor del campo potencial.

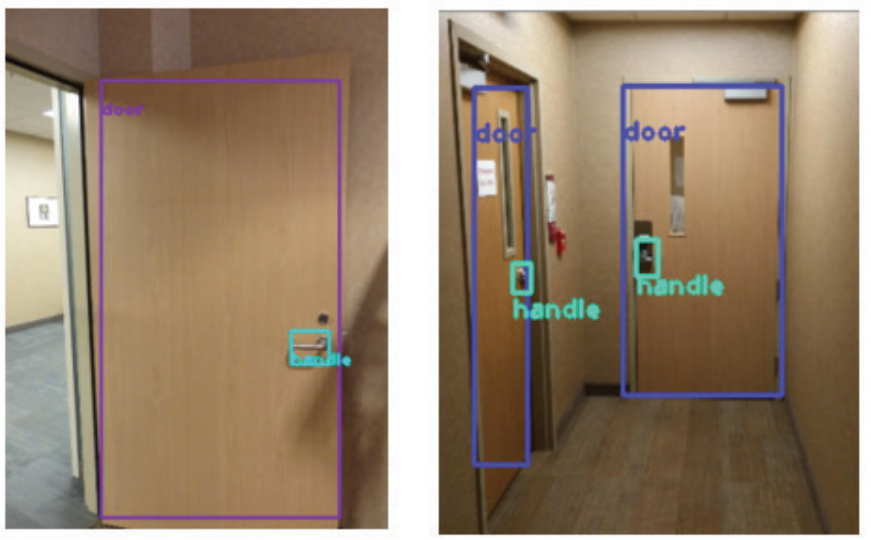



**Figura 19.** Detección de puertas con YOLO v3

**Fuente: (S.Bashiri, Rose, & Peissig, 2018)**

En la figura 19 se observan los resultados de la detección de las puertas, utilizando el *dataset* MCIndoor20000 proporcionado por Marshfield Clinic, es decir, se corrió el algoritmo sobre una *dataset* que no era el de entrenamiento con 754 fotos de prueba, proporcionando un porcentaje de precisión promedio de 97.5 %. En la figura 20 se observa la trayectoria dibujada por el robot (color rosado) al seguir a la persona a través de la puerta, sin generar colisiones.

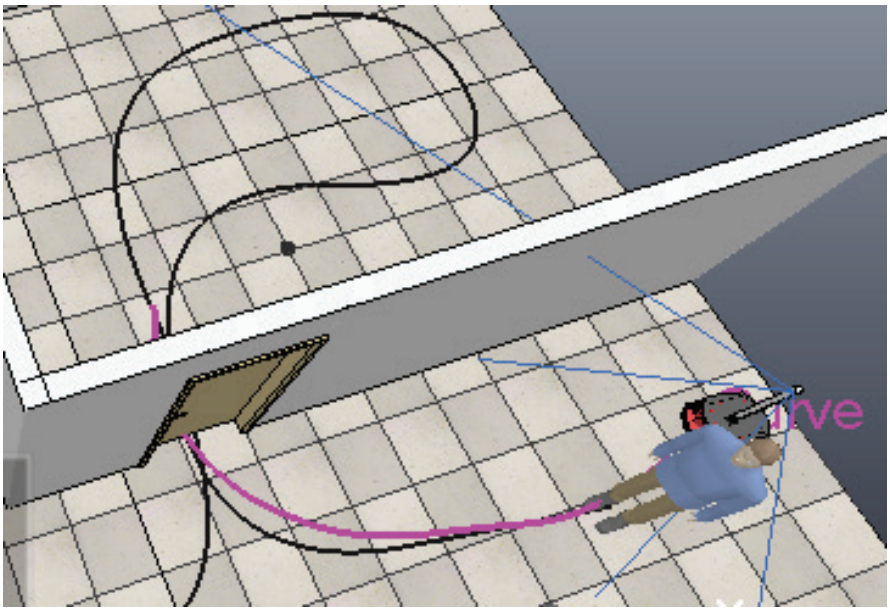


**Figura 20.** Trayectoria del robot por la puerta

### 3.4. Extracción de las coordenadas 3D de la persona

Para la calibración de la cámara web de 0.3 megapíxeles se utilizó la función de OpenCV cv2.calibrateCamera, con un tablero de ajedrez 9x6 de la figura 21, siendo esto el número de esquinas interiores. Se tomaron fotos desde diferentes perspectivas, distintas rotaciones y traslaciones para lograr una buena calibración, 33 fotos en total. El resultado logrado fue un error de reproyección de 0.0924 pixeles y se almacenan los valores obtenidos en un archivo de extensión .txt

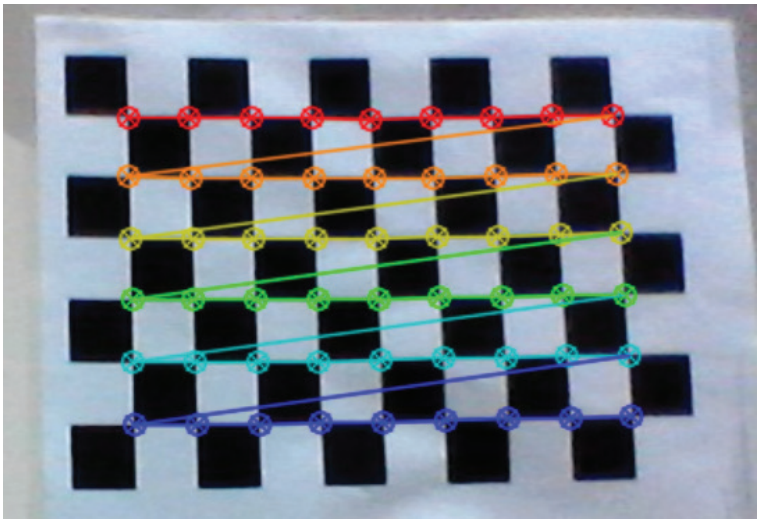


**Figura 21.** Patrón de calibración utilizado

Al obtener el vector de rotación se emplea la transformada de Rodrigues (Piña, 2011) para convertir el vector de rotación en una matriz de rotación 3x3, de manera que se pueda usar la función para convertir esta matriz a los ángulos de Euler (*roll, pitch, yaw*), dicha matriz es la rotación del marcador hacia la cámara. En la figura 22 se observan los resultados de las estimaciones, la simulación se hizo con un marcador de 4x4 de id = 4 y dimensiones de 17.3 centímetros.

Las pruebas realizadas para calcular la precisión de la estimación de la coordenada Z y X, se llevaron a cabo con una cámara web con una resolución de 0.3 MP (640x480). La simulación consistía en medir con una cinta métrica la diferencia entre el centro del marcador Aruco y el centro de la cámara en el eje horizontal, y la distancia entre la cámara y dicho marcador. El proceso se realizó con distancias desde 59 hasta 220 centímetros.

El resultado obtenido de las simulaciones arroja un error relativo para la medición de distancia de 1.44 %, produciendo diferencias desde 0.003 cm hasta 6.3 centímetros a distancias mayores a 2.2 metros, cabe destacar que estos resultados se ven afectados por los errores en las mediciones y la baja resolución de la cámara.

Con respecto a la coordenada X, resultaron diferencias desde 0.1 hasta 8.9 centímetros, cuando el marcador Aruco se encontraba a 60 centímetros del eje horizontal y una distancia de 220 cm, pero a distancias menores a 2.2 m la diferencia máxima en X fue de 0.7 cm.

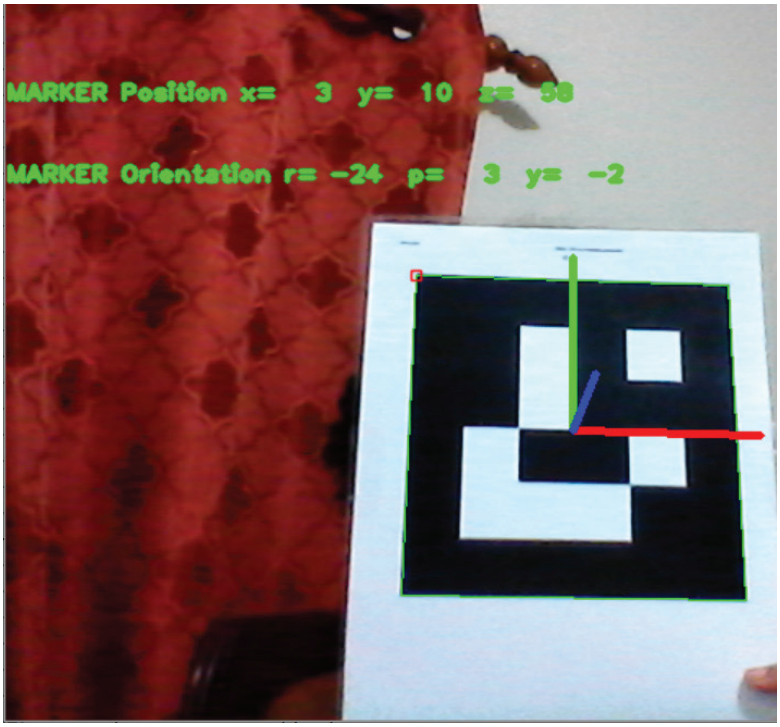



**Figura 22**. Estimación de la posición del marcador Aruco

## 4. Conclusiones

Este artículo presentó los resultados del diseño y simulación de un robot seguidor de personas utilizando marcadores aruco para el transporte de suministros en los hospitales, aplicando técnicas de procesamiento de imágenes, sistema de control y planeamiento de la ruta local en el software CoppeliaSim. La estructura de este consta de un sistema de seguimiento directo mediante un controlador P y PD para determinar la velocidad lineal y angular del vehículo, que luego es mapeado al vehículo al tomar en consideración la cinemática del robot diferencial. Se emplea un diccionario de marcadores aruco para identificar a

los usuarios y tomando en consideración los parámetros intrínsecos y extrínsecos de la cámara se extraen las coordenadas de la persona en el sistema de coordenadas globales, y la localización del vehículo se logra por la odometría de rueda con un margen de error despreciable a distancias cortas.

El sistema propuesto fue simulado en entornos dinámicos en situaciones con varios obstáculos. El enfoque presentado de acuerdo con las simulaciones realizadas fue capaz de encontrar a la persona, incluso cuando el robot la pierde momentáneamente, planificando una ruta local por medio de la data recibida del LIDAR y estimando la dirección de movimiento de la persona. Se empleó una red neuronal convolucional conocida como YOLO v3 para identificar las puertas con una precisión promedio de 97.5 %, permitiendo crear un planeamiento local adaptativo.

Esta aproximación no necesita llevar a cabo actividades que consuman mucho tiempo, como el procesamiento de imágenes o el procesamiento de visión por computadora para el planeamiento de la ruta u obtención de información mediante visión estéreo, que suelen ser utilizados en sistemas propuestos por otros autores. HWF se puede implementar fácilmente en un sistema en tiempo real porque la sobrecarga al calcularlo es relativamente ligera. El éxito de evadir obstáculos dinámicos de parte de este algoritmo reactivo depende de la velocidad relativa entre el obstáculo dinámico y el vehículo. Este algoritmo proporcionó mejores resultados en las simulaciones en comparación a otros métodos como el *Obstacle-Dependent Gaussian Potential Field* (ODG-PF), descartado porque había situaciones en las cuales era muy propenso a colisionar, pues al estar muy cerca de la meta si se presenta un obstáculo justo detrás de esta, el campo repulsivo era máximo pero el campo atractivo era mínimo, entonces el ángulo mínimo del campo total correspondiente al ángulo seleccionado por el algoritmo, provocaba que el vehículo colisionara.

Al analizar los resultados finales obtenidos queda demostrado que el diseño propuesto cumple con los objetivos establecidos, evidenciando funcionalidad en cada una de sus etapas de simulación, aplicando la red neuronal de YOLO, Heading Weight Function, odometría de rueda y sistemas de control.

## 5. Referencias

Ben-Ari, M., & Mondada, F. (2018). *Elements of Robotics.* Springer, Cham. doi:https://doi.org/10.1007/978-3-319-62533-1_5

Bradski, G., & Kaehler, A. (2016). *Learning OpenCV 3*. California: O'Reilly.

Cho, J.-H., & Dong-Sung , P. (23 de Agosto de 2018). A Real-Time Obstacle Avoidance Method for Autonomous Vehicles Using an Obstacle-Dependent Gaussian Potential Field. *Journal of Advanced Transportation, 2018*(1), 1-15. doi:10.1155/2018/5041401

Coppelia Robotics. (s.f.). *Coppelia Robotics AG*. Recuperado el 20 de Agosto de 2020, de Coppelia Robotics: https://www.coppelia-robotics.com/

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Marín-Jiménez., M. J. (1 de Junio de 2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition, 47*(6), 2280–2292. doi:10.1016/j.patcog.2014.01.005

Hata, K., & Savarese, S. (2015). *CS231A Course Notes 1: Camera Models.* Stanford University. Stanford, California, USA: Stanford University. Obtenido de https://web.stanford.edu/class/cs231a/course_notes/01-camera-models.pdf

Islam, M. J., Hong, J., & Sattar, J. (21 de Marzo de 2018). Person Following by Autonomous Robots: A Categorical Overview. *The International Journal of Robotics Research, 38*(12), 32. doi:10.1177/0278364919881683

Korsaeth, A. (Agosto de 2019). The Heading Weight Function: A Novel LiDAR-Based Local Planner for Nonholonomic Mobile Robots. *Sensors , 19*(16), 3606. doi:10.3390/s19163606

Kunz, C., Hein, B., Genten, V., & Meibner, P. (2019). Metric-based evaluation of fiducial markers for medical procedures. *Medical Imaging 2019: Image-Guided Procedures, Robotic Interventions, and Modeling*, *10951*, pág. 97. San Diego, California. doi:https://doi.org/10.1117/12.2511720

Mihelj, M., Bajd, T., & Ude, A. (2019). *Robotics* (Segunda ed.). Suiza: Springer.

Morioka, K., Lee, J.-H., & Hashimoto, H. (Febrero de 2004). Human-following mobile robot in a distributed intelligent sensor network. *IEEE Transactions on Industrial Electronics, 51*(1), 229-237. doi:10.1109/TIE.2003.821894

OpenCV team. (s.f.). *OpenCV*. Recuperado el 20 de Agosto de 2020, de OpenCV team: https://docs.opencv.org/trunk/d5/dae/tutorial_aruco_detection.html

OpenCV team. (s.f.). *OpenCV*. Recuperado el 20 de Agosto de 2020, de OpenCV team: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

Piña, E. (27 de Junio de 2011). Rotations with Rodrigues' vector. *European Journal of Physics, 32*(5), 1171. doi:10.1088/0143-0807/32/5/005

Redmon, J., & Farhadi, A. (8 de Abril de 2018). YOLOv3: An Incremental Improvement. (arXiv, Ed.) *1804.02767*. Obtenido de https://pjreddie.com/media/files/papers/YOLOv3.pdf

S.Bashiri, F., Rose, E. L., & Peissig, P. (Abril de 2018). MCIndoor20000: A fully-labeled image dataset to advance indoor objects detection. *Data in Brief, 17*, 71-75. doi:https://doi.org/10.1016/j.dib.2017.12.047

Sedighi, K., Ashenayi, K., & Manikas, T. (5 de Octubre de 2004). Autonomous Local Path Planning for a Mobile Robot Using a Genetic Algorithm. *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753). 2*, págs. 1338-1345. Portland, Estados Unidos: IEEE . doi:10.1109/CEC.2004.1331052

Westbrook, J. I., Duffield, C., Li, L., & Creswick, N. J. (11 de Noviembre de 2011). How much time do nurses have for patients? a longitudinal study quantifying hospital nurses' patterns of task time distribution and interactions with health professionals. *BMC health services research, 11*(1), 319. doi:10.1186/1472-6963-11-319