Una propuesta de arquitectura para el control de un robot guía

Karla Lourdes Luna Gallegos
CIEP FI - UASLP, Av. Manuel Nava s/n, C.P. 78290, San Luis Potosí, SLP, México
karla.luna@alumnos.uaslp.edu.mx

Elvia Ruth Palacios Hernández
FC – UASLP, Av. Salvador Nava s/n, C.P. 78290, San Luis Potosí, SLP, México
epalacios@fciencias.uaslp.mx

Antonio Marín Hernández
CIIA - UV, Sebastián Camacho No. 5, C.P. 91000, Xalapa, Veracruz, México
anmarin@uv.mx

Resumen

En este artículo se propone una arquitectura de software para controlar un robot móvil en la tarea de robot guía. La arquitectura se divide en tres módulos generales de acción dirigidos por un supervisor. La toma de decisiones se basa en el comportamiento que presente el usuario con la finalidad de ofrecer un mejor servicio a las personas. En el primer módulo, el robot está a la espera de un usuario, en el segundo módulo el robot realiza la acción de guía en la cual traza una trayectoria al destino deseado, además controla la velocidad e identifica al usuario para mantenerlo en el campo de visión del robot, una vez que llego a su destino el robot regresa a su origen. Cada actividad se comunica mediante el middleware ROS pudiendo operar de manera separada y en grupo. La arquitectura propuesta se implementa en la plataforma Tbot, éste es capaz de trasladarse del origen a su destino modulando su velocidad sin perder de vista al usuario, la navegación se considera sin obstáculos.

Palabra(s) Clave(s): arquitectura de software, control automático, control de sistemas, robots móviles.

1. Introducción

Hoy en día las aplicaciones de la robótica se han extendido a lugares cotidianos para las personas como sus hogares, lugares públicos, zonas de desastres, etc. Esto involucra una interacción más cercana del robot con las personas, donde para ofrecer un servicio seguro, confiable y eficiente debe considerar normas y comportamientos sociales. Se requiere entonces que el robot realice tareas complejas y toma de decisiones para adaptarse a un ambiente humano.

En este artículo el objetivo principal del robot móvil es la tarea de guía de personas, para realizar esta tarea se implementará una arquitectura de software para controlar al robot. Existen muchas formas de arquitectura de software para el control de un sistema, éstas permiten controlar los comportamientos del robot en distintos niveles y de forma paralela, lo que permitirá realizar una tarea compleja. A continuación se detallan algunas de las propuestas que existen en la literatura, no se tiene preferencia por algún diseño en particular.

En [1] se propone una arquitectura de control que consiste en 4 capas: manejo de dispositivos, control, coordinación y organización, cada capa se controla de manera autónoma y su procesamiento es de forma paralela, realiza 4 módulos generales: navegación con control, planeación de trayectoria global, operaciones de control y comunicación. La arquitectura en [2] se compone de 6 capas: hardware, dispositivos, middleware, componentes, servicios y aplicaciones. Una arquitectura diferente se propone en [3] basada en toma de decisiones por control, que consiste en un sistema de gestión de tareas, un sistema de procesamiento de información y un sistema de navegación, cada uno independiente y paralelo de los otros. En [4] se presenta una arquitectura jerárquica para el control de un robot humanoide, consta de cinco capas de implementación: dos capas de comportamiento, de tareas y de estado del robot. El controlador del robot juzga el estado del robot y proporciona al robot comportamientos basados en la percepción y el control autónomo de los actuadores.

Una arquitectura de estructura modular se presenta en [5], aquí los módulos están separados por las características y responsabilidades, primero se encuentra el nivel básico donde están los dispositivos y sensores, después un nivel reactivo para mantener la integridad del robot, los

siguientes módulos controlan las funciones de movimiento de bajo nivel (velocidad, aceleración, posición) y alto nivel (navegación, teleoperación, manipulación).

En este artículo se describe el diseño de una arquitectura jerárquica para controlar un robot móvil en la aplicación de robot guía. Se propone una arquitectura de 6 capas: el nivel de dispositivos, una capa de comunicación entre el hardware y el software, una capa de habilidades, de ejecución, de comportamiento, hasta el nivel supervisor.

El artículo está organizado de la siguiente manera: en la sección 2 se describen las capas que integran la arquitectura jerárquica propuesta. En la sección 3 se muestran los experimentos realizados y los resultados obtenidos. Finalmente en la sección 4 se muestran las conclusiones del trabajo.

2. Arquitectura jerárquica para un robot móvil

La arquitectura se compone de 6 capas (Figura 1), en la primera capa se encuentran los dispositivos de entrada y salida, la segunda capa es la información que se extrae de los sensores, en la tercera capa se encuentran las habilidades que tiene el robot, en la capa siguiente se encuentra la operación a ejecutar del robot, después una capa de comportamiento y en la última capa está el supervisor. Todas las capas están conectadas al middleware ROS para la transferencia de información entre procesos.

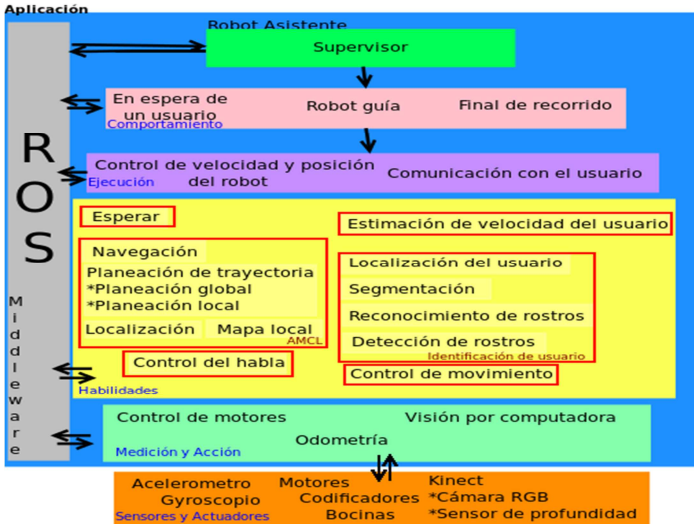

Fig. 1. Arquitectura del software.

2.1. Primera capa: Dispositivos

La plataforma móvil que se utilizará en este artículo es el robot llamado Tbot (Figura 2), el cual consiste de una plataforma Turtlebot como base con 2 pisos extras y mide en total 1.2 metros, tiene dos sensores Microsoft Kinect, una laptop para el control de navegación y otra para el control por visión, unas bocinas estéreo, una batería para la alimentación del iRobotCreate y una para los dos Kinect.

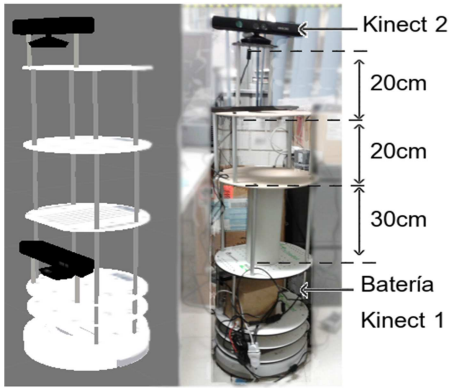

Fig. 2. Robot Tbot.

La base móvil es un iRobotCreate (Figura 3) la cual está equipada con sensores que perciben el entorno y proveen una retroalimentación de su posición y orientación para un algoritmo de control. Tiene un codificador en cada rueda con lo que podemos calcular la odometría, un giroscópio, cuatro sensores cliff para detectar si se está cerca de un escalón, dos sensores de contacto para detectar si el robot choca con algún objeto, un botón de encendido. Los actuadores son dos motores de las ruedas izquierda y derecha que proporcionan un sistema de acción diferencial, tres luces de diodo (LED), un altavoz capaz de producir diferentes tonos, tiene además una rueda de soporte [6].

Fig. 3. iRobotCreate, [7].

Los sensores Microsoft Kinect, cuentan con una cámara RGB, un emisor/receptor de luz infrarroja (IR), un arreglo múltiple de micrófonos y un acelerómetro de tres ejes en un motor de inclinación (Figura 4). La cámara tiene un ángulo de apertura de 43° vertical y 57° horizontal, la cual obtiene imágenes de 640x480 pixeles a una velocidad de 30 frames/sec.

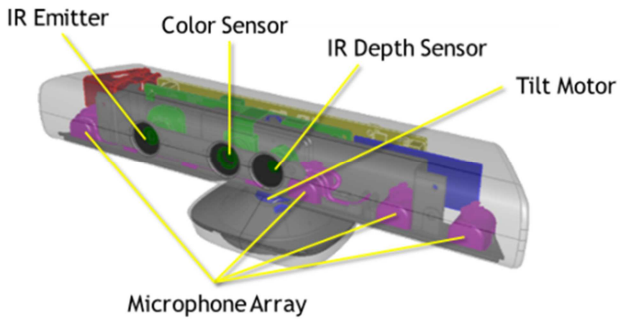

Fig. 4. Sensor Microsoft Kinect, [8].

2.2. Segunda capa: Percepción y acción. ROS

El Middleware ROS (Robot OperatingSysyem) es un conjunto de software de código abierto que incluye librerías y herramientas para los desarrolladores de software para robots. Éste funciona como un sistema operativo, proporciona controladores de dispositivos y acceso al hardware con la que podemos obtener la información de los sensores y controlar los actuadores. Por medio de paso de mensajes entre procesos podemos manipular la información obtenida y accionar al robot por medio de velocidad lineal y angular, además ofrece algoritmos de aplicación para múltiples robots, visualizadores, simuladores, entre otras cosas. ROS es un software distribuido de procesos (también conocidos como nodos) que permite a los ejecutables ser diseñados de forma individual y acoplados en tiempo real. Estos procesos se pueden agrupar en paquetes y pilas, que pueden ser fácilmente compartidos y distribuidos. ROS también es compatible con un sistema de repositorios que permiten la colaboración y distribución de código [9].

2.3. Capa de habilidades

Una vez que se obtiene la información de los dispositivos pasamos a la capa de habilidades. En esta capa se presentan las capacidades que tiene el robot, son acciones que puede realizar de manera conjunta con otras acciones, de manera separada o de forma paralela.

2.3.1. Estado de espera

Inicialmente, el robot se encuentra en un estado de espera, en este momento el robot debe conocer el mapa en el que se encuentra y conocer su posición actual, esperando por un posible usuario. La acción del movimiento del robot es nula mientras que el algoritmo de visión es el que se encuentra en activo para detectar si una persona se encuentra a una distancia cercana del robot. Una vez que se identificó un posible usuario, se espera la instrucción para continuar con la actividad.

2.3.2. Algoritmo de identificación

La primera actividad que realiza el robot es con el algoritmo de identificación de usuario el cual con los datos que proporciona el sensor Kinect superior (RGB y depth) por medio de control visual, es posible determinar cuando alguien se encuentra dentro del campo de visión de la cámara (2D) donde serán detectados y localizados en un espacio de tres dimensiones. Se realizó un algoritmo para este propósito el cual se detalla en [10], el usuario es encontrado por su rostro, la clasificación entra como auxiliar para la ubicación de la persona en caso de perder la detección del rostro y después se realiza una segmentación para la localización del usuario en el espacio de trabajo. El esquema final del algoritmo de identificación de personas se muestra en la Figura 5.

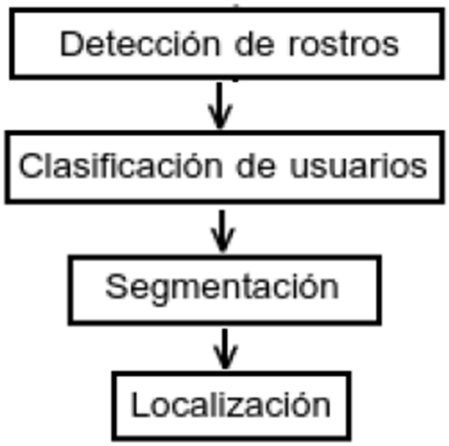

Fig. 5. Algoritmo de Identificación de personas.

2.3.3. Estimación de velocidad

Después de localizar al usuario se estima la trayectoria que este realiza y la velocidad. Para calcular la trayectoria utilizamos los splines cúbicos ya que éste elimina dos problemas de la interpolación lineal: su lentitud de convergencia y el que presenta ángulos en los nodos, lo cual es inadecuado en ciertas aplicaciones. Un spline cúbico es una función polinómica segmentaria de grado 3 con primera y segunda derivada continuas [11].

Dados $n+1$ puntos (x_k, y_k) para $k = 1,2,\dots,n+1$ con $x_1 < x_2 < \cdots < x_{n-1}$ tales que:

$$s(x) = q_k(x), \qquad x \in [x_k, x_{k+1}] \quad para \;\; k = 1,2,\dots,n \tag{1}$$

que verifiquen las siguientes condiciones de interpolación:

$$\begin{aligned} q_k(x_k) &= y_k \qquad para \;\; k = 1,2,\dots,n \\ q_k(x_{k+1}) &= y_{k+1} \;\; para \; k = 1,2,\dots,n \end{aligned} \tag{2}$$

y también en las condiciones de conexión:

$$\begin{aligned} {q'}_k(x_{k+1}) &= {q'}_{k+1}(x_{k+1}) \;\; para \; k = 1,2,\dots,n-1 \\ {q''}_k(x_{k+1}) &= {q''}_{k+1}(x_{k+1}) \;\; para \; k = 1,2,\dots,n-1 \end{aligned} \tag{3}$$

Las primeras condiciones (ec. 2) establecen que s es continua e interpola los puntos (x_k, y_k). Las segundas (ec. 3) implican que s tiene primera y segunda derivadas continuas [11]. Se dice entonces que $s(x)$ es spline interpolador para $P_0, P_1, \dots, P_n$. Denotando con $h_k = x_{k+1} - x_k$ para $k = 0,1,\dots,n-1$ y $\sigma_k = s''(x_k)$ para $k = 0,1,\dots,n-1$. Se tiene que:

$$\begin{aligned} q_k(x) = {} & \frac{\sigma_k}{6}\left[\frac{(x_{k+1}-x)^3}{h_k} - h_k(x_{k+1}-x)\right] + \frac{\sigma_{k+1}}{6}\left[\frac{(x-x_k)^3}{h_k} - h_k(x-x_k)\right] + y_k\left[\frac{x_{k+1}-x}{h_k}\right] \\ & + y_{k+1}\left[\frac{x-x_k}{h_k}\right] \;\; para \; k = 0,1,\dots,n-1. \end{aligned} \tag{4}$$

La cual es la ecuación del spline $q_k(x)$.

Para obtener $\sigma_0, \sigma_1, \dots, \sigma_n$ debemos resolver la siguiente ecuación:

$$h_{k+1}\sigma_{k+1} + 2(h_{k-1} + h_k)\sigma_k + h_k\sigma_{k+1} = 6\left(\frac{y_{k+1}-y_k}{h_k} - \frac{y_k - y_{k+1}}{h_{k+1}}\right) \;\; para \; k = 1,\dots,n-1 \tag{5}$$

Se realizaron pruebas fuera de línea en las que se tomaron los datos de algunas personas al caminar por un pasillo para obtener su trayectoria, los datos obtenidos fueron la posición del

usuario en un espacio tridimensional y el tiempo a lo largo del recorrido. En la gráfica de la Figura 6 se muestra el recorrido de las personas y la trayectoria estimada con los splines.

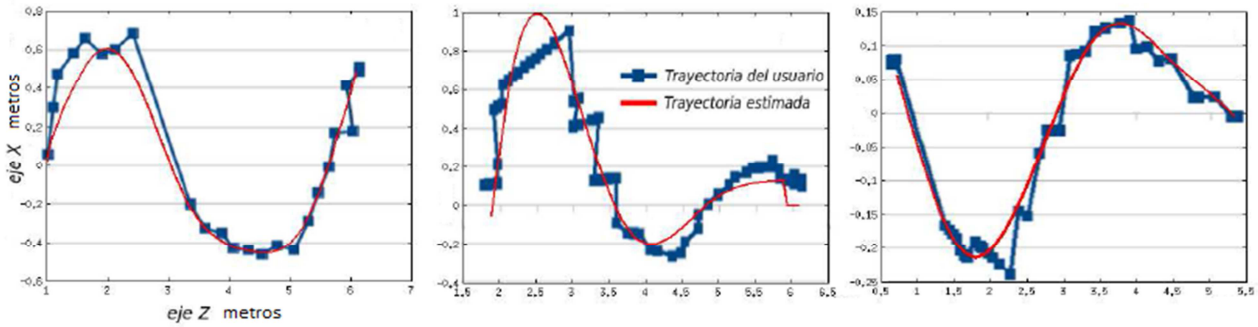

Fig. 6. Trayectoria estimada con algoritmo de visión.

Para la estimación de velocidad se utilizó el método de longitud del arco de punto a punto sobre la trayectoria obtenida con el spline y el tiempo del segmento. La longitud del arco es:

$$s = \int_a^b \sqrt{1 + [f'(x)]^2}\,dx \qquad (6)$$

donde $f(x)$ es el spline y la velocidad se determina con la fórmula: $v = {}^{s}/_{t}$

Las velocidades resultantes están dadas en metros por segundo y se muestran en la Tabla 1. La velocidad promedio de una persona al caminar es de 1.2 m/s [12], como se muestra en la Tabla 1, el comportamiento de las velocidades al iniciar el recorrido es menor a la velocidad promedio, sin embargo conforme avanza, ésta se nivela a una velocidad aproximada de 1 m/s, disminuyendo nuevamente mientras se acerca al final del trayecto.

Velocidad del usuario (punto a punto) m/s		
Persona 1	Persona 2	Persona 3
0.7205	1.0239	0.4478
1.0046	0.6853	1.0449
0.9800	0.8303	0.9742
0.9666	0.7345	1.1266
0.5776	0.5693	1.1060

Tabla 1. Velocidad estimada del usuario.

2.3.4. AMCL

AMCL es un sistema de localización probabilística para mover un robot en un espacio de 2D. Éste implementa el enfoque Adaptativo de la Localización de Monte Carlo, el cual utiliza un filtro de partículas para rastrear la posición del robot contra un mapa conocido [13].

Utiliza algoritmos para localizarse en el mapa [14], tales como*, simple motion model odometry* el cual acepta una posición inicial y la información de odometría como entrada y proporciona una salida aleatoria de la siguiente posición, este algoritmo es más sencillo de implementar que otros que calculan la probabilidad de una nueva posición. Otro algoritmo utilizado es el *beam range finder model*, para este modelo se incorporan cuatro tipos de mediciones del error: pequeños ruidos de medición, errores debido a objetos inesperados, a fallos en la detección de objetos y ruido inexplicable. La entrada del algoritmo es una exploración completa de rango (utilizando las mediciones anteriores), la posición del robot y un mapa. El algoritmo devuelve una probabilidad de los objetos que se encuentran en el campo de visión de la cámara. El algoritmo *likelihood field range finder model*, entra como refuerzo del anterior, pues, mientras más se vincule la geometría y la física de los telémetros se tendrán dos inconvenientes: la falta de suavidad y la complejidad computacional. Otro de los algoritmos que utiliza es el *augmented MCL*, este algoritmo es parecido al MCL (Monte Carlo Localization) pero a diferencia del original este es capaz de relocalizarse en caso de que el robot sea secuestrado o que tenga fallas de localización global por medio de añadir partículas al azar a los conjuntos de partículas ya existentes suponiendo una pequeña probabilidad de que el robot sea secuestrado.

2.3.5. Control de movimiento

Se utilizó la ley de control del artículo [15] en el cual se propone un controlador adaptativo para guiar a un robot móvil uniciclo durante el seguimiento de trayectoria. El control consiste en generar velocidades lineales y angulares teniendo en cuenta soló el modelo cinemático y compensando la dinámica del robot. Se analizó la estabilidad del sistema utilizando la teoría de Lyapunov.

La ley de control cinemática propuesta aplicada al robot está dada por:

$$\begin{bmatrix} u_{ref}^{c} \\ w_{ref}^{c} \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi \\ -\frac{1}{a}\sin\psi & \frac{1}{a}\cos\psi \end{bmatrix} \begin{bmatrix} \dot{x_d} + I_x \tanh\left(\frac{k_x}{I_x}\tilde{x}\right) \\ \dot{y_d} + I_y \tanh\left(\frac{k_y}{I_y}\tilde{y}\right) \end{bmatrix} \quad \textbf{(7)}$$

Donde u_{ref}^{c} es la velocidad lineal, w_{ref}^{c} es la velocidad angular, ψ es la orientación del robot,x y y son la posición actual,x_d y $y_d$son la posición deseada, $\tilde{x} = x_d - x$ y $\tilde{y} = y_d - y$son los errores en la posición actual, $k_x > 0$ y $k_y > 0$ son las ganancias del controlador,$I_x \in \Re$ y $I_y \in \Re$ son las constantes de saturación.

2.4. Capa de ejecución

Algunas de las capacidades mencionadas se realizan como un proceso interno (por ejemplo el control visual), los datos manejados se manipulan a nivel del software. Otras habilidades vinculan la información procesada con los dispositivos (hardware), permitiendo accionar al robot.

En la capa de ejecución se activan dos estados de manera paralela, el control de velocidad y posición del robot y la comunicación con el usuario.

Las habilidades que involucran el control de la base iRobot Create son: navegación y control de movimiento. Mientras que la comunicación con el usuario está conectado directamente al control del habla. Estos operan en conjunto con las demás actividades del robot, esto explicara de manera clara en la capa de supervisor.

2.5. Capa de comportamiento

Para la tarea de robot asistente se consideran tres comportamientos básicos en su funcionamiento. Como se explicó anteriormente, el robot inicia en una etapa de espera en la que los actuadores de la base están inactivos hasta que encuentre un usuario, aquí el robot puede tratar de establecer una comunicación con un posible usuario, invitándolo a utilizar sus servicios con el algoritmo de control de voz, si la persona se encuentra a una distancia cercana.

El comportamiento como robot guía inicia al momento en que un usuario requiere la asistencia del robot para llegar a algún punto en específico, el robot reconoce a su usuario y una vez conocido el destino al que se desea llegar traza una trayectoria y modula su velocidad por

medio de un sistema de control con la información estimada de la velocidad del usuario. Sin perder la comunicación con el usuario para ofrecer un recorrido eficiente y confortable.

Al final del recorrido, el robot se despide del usuario y traza una trayectoria para volver a su origen.

2.6. Supervisor

La capa del supervisor se encarga de dirigir todas las capas anteriores, en que momento debe accionarse o desactivarse cada una de ellas. En la Figura 7 se muestra la operación del supervisor en un diagrama de flujo.

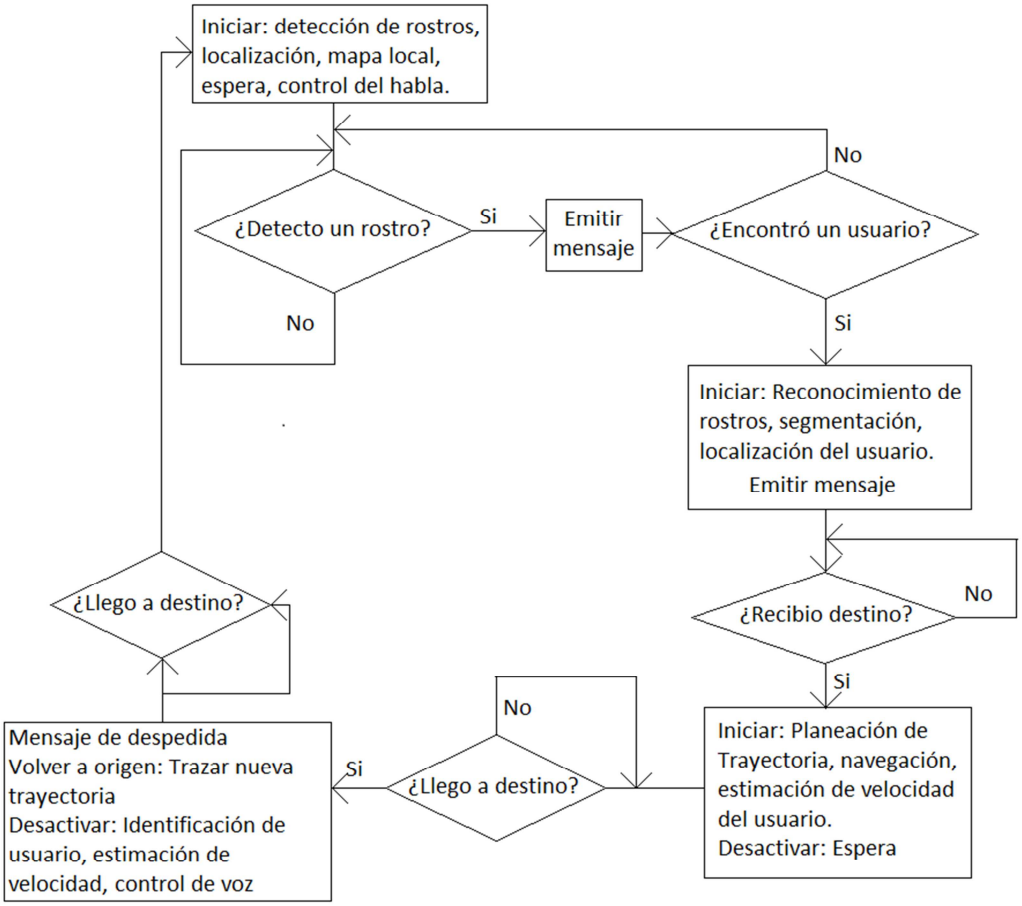

Fig. 7. Supervisor.

El funcionamiento del robot se inicia con la localización del robot en un mapa local, mientras que se encuentra inmóvil en la recepción (estado de espera). El robot inicia la búsqueda de rostros en su campo de visión, una vez que detectó un rostro se establece una comunicación con el posible usuario, cuando la persona solicita los servicios del robot se identifica y localiza al usuario. Luego de recibir el destino deseado se ejecutan los algoritmos de navegación, planeación de trayectorias y estimación de velocidad del usuario además se mantiene una comunicación con el usuario a lo largo del recorrido por medio del algoritmo de control del habla. Finalmente, cuando se llega al destino se emite un mensaje de despedida y se traza una nueva trayectoria para volver al origen.

3. Experimentos y resultados

Para obtener el mapa local se utiliza un paquete de navegación que proporciona ROS llamado GMapping, este algoritmo puede generar mapas por medio de un láser, el cual realiza una técnica llamada SLAM (Simultaneous Localization And Mapping) que consiste en utilizar un filtro de partículas en la que cada partícula lleva un mapa individual del entorno. Se calcula una distribución teniendo en cuenta no solo el movimiento del robot sino también la observación más reciente. La Figura 8 muestra el entorno donde opera el Tbot obtenido por GMapping.

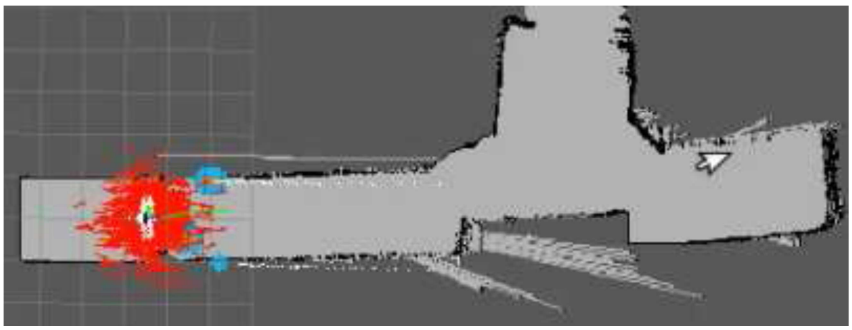

Fig. 8. Mapa local.

El robot es localizado en la recepción para iniciar con la actividad de robot guía, la señal de los actuadores se encuentra inactiva. Los resultados del algoritmo de detección de rostros correspondiente a la capa de habilidades se muestran en la Figura 9, el algoritmo es capaz de ubicar la cara de una persona con un recuadro verde en diferentes de posiciones.

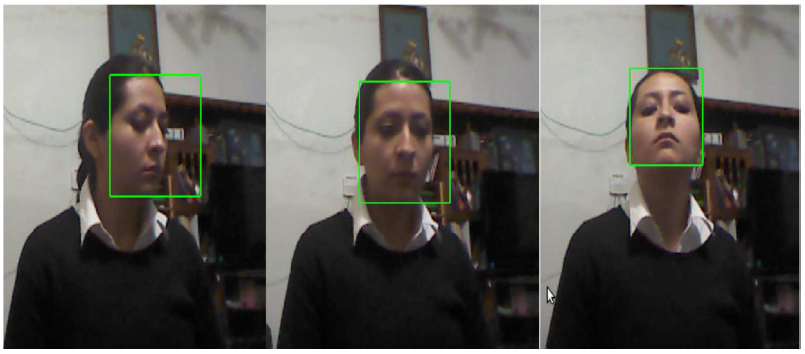

Fig. 9. Identificación de rostros.

El robot inicia el control del habla que se maneja con un paquete de ROS, éste se acciona invitando al usuario a disponer de sus servicios con la frase “Hola, soy Tbot ¿Puedo ayudarte?” y se mantiene a lo largo del recorrido, para mantener una interacción con el usuario con frases como “sígueme”, “vas muy rápido”, etc. ofreciendo así un servicio más amigable.

Cuando el usuario es identificado, la fase de segmentación toma la posición del rostro del usuario obtenida con el algoritmo de detección de rostros y por medio de segmentación por color (Hue, Saturation, Value) y con el método de región de crecimiento (con la información de profundidad como criterio de pertenencia) es posible separar a la persona del resto de la imagen incluso si el rostro se pierde. Una vez que se conoce la ubicación del usuario respecto a la cámara, es posible localizarlo en un espacio de 3D como se muestra en la Figura 10. Con los datos obtenidos se puede calcular la velocidad del usuario como se muestra en la sección 2.3.3.

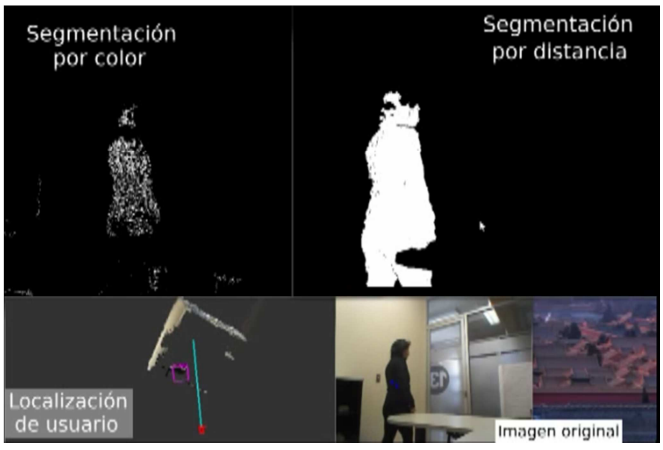

Fig. 10. Segmentación y localización del usuario.

El control del movimiento de las ruedas del robot se realiza con la ley de control de la sección 2.3.5 el cual se muestra su comportamiento en la Figura 11, donde la línea verde es la trayectoria real que sigue el robot y la línea roja es la trayectoria deseada.

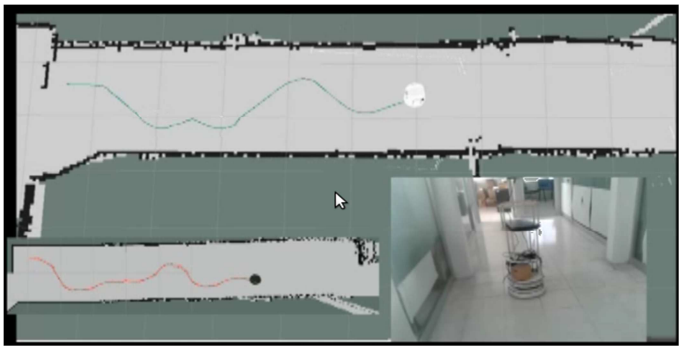

Fig. 8. Control de Trayectoria.

Se envían mensajes de velocidad tanto angular como lineal para accionar los motores del robot, al mismo tiempo se retroalimenta la posición real del robot que se realiza haciendo una transformación de la posición inicial del robot dentro del mapa con la odometría de éste.

4. Conclusiones

Las arquitecturas de software que manejan los robots móviles se han basado principalmente en la navegación, teniendo en segundo plano las demás actividades. Para una interacción con los humanos, se propuso una arquitectura de organización jerárquica centrada en el comportamiento de las personas. En la arquitectura propuesta, cada capa se comunica con la capa superior inmediata y con su subordinada inmediata, además cuenta con la ventaja de que cada capa se comunica por medio del middleware ROS el cual permite operar de manera paralela y conjunta cada actividad, pues facilita la comunicación entre procesos y el paso de mensajes de información. Se ordenaron las capas de acuerdo a su valor, de manera descendente, se encuentra primero el supervisor, el cual se encarga de tomar la decisión final de la actividad que realizará el robot, esto depende en su mayoría de la acción del usuario. La capa de ejecución contiene dos funciones generales, el movimiento del robot y la comunicación con el usuario, el supervisor ordena que función debe ejecutarse iniciando así la capa de habilidades. Esta capa requiere conocer las mediciones de los sensores y envía información para el movimiento de los actuadores. La capa de medición y acción funciona como enlace

entre los sensores y los algoritmos de habilidades. Finalmente en la última capa de la arquitectura se encuentran los sensores y actuadores, que son los dispositivos con los que cuenta el robot y proporcionan la información del entorno.

Con los resultados obtenidos podemos observar que es posible identificar y localizar al usuario, además el robot es capaz de seguir una trayectoria, logrando con esto llevar al usuario de una posición inicial (recepción) a una posición final (destino) con un control de velocidad. Todas las actividades se administran por medio del supervisor proporcionando así la acción que tomara el robot.

La tarea de un robot guía consiste en dirigir a una persona a un destino deseado. La arquitectura de software permite realizar ajustes de forma sencilla a esta actividad, logrando adaptar el servicio del robot guía de acuerdo a las necesidades del entorno y de los usuarios. Algunas aplicaciones para el servicio de este robot son por ejemplo: guía en museos o áreas de entretenimiento, hospitales, asistencia para adultos mayores, guía y transporte de objetos, entre otros.

5. Referencias

[1] Behavior-based autonomous cooperative control of intelligent mobile robot systems with embedded Petri nets. 15th International Symposium on Soft Computing and Intelligent Systems (SCIS) Joint 7th International Conference on and Advanced Intelligent Systems (ISIS). 3-6 Diciembre 2014.

[2] Robot Software Architecture based on IPv6. 6th IEEE Conference on Industrial Electronics and Applications (ICIEA). 21-23 June 2011.

[3] Research on open control architecture of autonomous mobile robot with multi-layer and modularization. 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR). 6-7 Marzo 2010.

[4] SSVEP-based hierarchical architecture for control of a humanoid robot with mind. 11th World Congress on Intelligent Control and Automation (WCICA). 29 Junio – 4 Julio 2014.

[5] A Generic Embedded Robot Platform for Real Time Navigation and Telepresence Abilities. 2nd Eastern European Regional Conference on the Engineering of Computer Based Systems (ECBS-EERC). 5-6 Septiembre 2011.

[6] A Guided Internship For High School Students Using iRobot Create. Proceedings of the 18th World Congress of International Federation of Automatic Control. Agosto 2011.

[7] MATLAB – based Simulator for the iRobot Create. En línea en: http://verifiablerobotics.com/CreateMATLABsimulator/createsimulator.html. Acceso: Mayo 2015.

[8] Kinect for Windows Sensor Components and Specifications. En línea en: http::/msdn.microsoft.com/en-us/library/jj131033.aspx. Acceso: Mayo 2015.

[9] ROS Introduction. En línea en: http://wiki.ros.org/ros/introduction. Acceso: Mayo 2015.

[10] Detección y Seguimiento de Personas con Análisis de Color en Datos RGB-D. XVI Congreso Latinoamericano de Control Automático. 14-17 Octubre 2014.

[11] A. Cordero Barbero, J.L. Hueso Pagoaga, E. Martinez Molada, J.R. Torregrosa Sánchez, Métodos Numéricos con Matlab. 2005. Ed. Univ. Politéc. Valencia. 496 pp.

[12] M. D. Latt, H. B. Menz, V. S. Fung, S. R. Lord, "Walking Speed, Cadence and Step Length are Selected to Optimize the Stability of Head and Pelvis Accelerations", Experimental Brain Research. Vol. 184. No. 2. Enero, 2008. 201-209 pp.

[13] AMCL. Acceso: http://wiki.ros.org/amcl. Mayo 2015.

[14] Probabilistic Robotics. Early Draft. 1999-2000.

[15] An Adaptive Dynamic Controller for Autonomous Mobile Robot Trajectory Tracking. XVI Control Engineering Practice. Mayo 2008.

6. Autores

M. en I. Karla Lourdes Luna Gallegos recibió el título de Ingeniero en Electrónica en 2011. En 2012 obtuvo el grado de Maestro en Ingeniería con especialidad en Control

Automático. Actualmente estudia el Doctorado en Ingeniería Eléctrica en la Universidad Autónoma de San Luis Potosí.

Dr. Elvia Ruth Palacios Hernández recibió el título de Ingeniero en Comunicaciones y Electrónica de la Universidad de Guadalajara en 1994. En 1999 obtuvo el grado de Maestro en Ciencias en Ingeniería Eléctrica del CINVESTAV Unidad Guadalajara. En 1999 realizó una estancia en el Laboratorio de Análisis y Arquitectura de Sistemas (LAAS-CNRS), obteniendo el Diploma de Estudios A profundidad (DEA) y el grado de doctor en sistemas automáticos del Instituto Nacional de Ciencias Aplicadas (INSA) de Toulouse en 2000 y 2004, respectivamente.

Dr. Antonio Marín Hernández recibió el título de Licenciado en Física de la Universidad Veracruzana en 1995. En 1998 obtuvo el grado de Maestro en Inteligencia Artificial de la Universidad Veracruzana y en el 2000 obtuvo el título de Maestría en Informática y Telecomunicaciones (DEA) así como el grado de Doctor en Informática y Telecomunicaciones del Institut National Polytechnique de Toulouse en 2004.

Pistas Educativas Año XXXVI - ISSN 1405-1249

Certificado de Licitud de Título 6216; Certificado de Licitud de Contenido 4777; Expediente de Reserva 6 98 92