Cloud/Edge Robotics: Navegación autónoma de Auto-Robot y Cuadricoptero

Manuel Costanzo[1], Marcos Boggia[1], Ismael Rodriguez[1], and Armando De Giusti[1,2]

[1]Instituto de Investigación en Informática LIDI, Facultad de Informática, UNLP-CIC
[2]Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
[1]{mcostanzo, mboggia, ismael, degiusti}@lidi.info.unlp.edu.ar

Abstract: Este trabajo presenta el diseño, desarrollo y despliegue de un sistema Multi-Robots, compuesto por un robot que simula un vehículo tradicional, como así también, un Dron capaz de capturar imágenes aéreas; ambos conectados al Cloud de Amazon. Se detallan los prototipos desarrollados y el protocolo de comunicación utilizado; se mencionan los algoritmos implementados para el procesamiento y reconocimiento de objetos en imágenes, la simulación y planificación de caminos óptimos, junto con los métodos de transformación de los mismos y la determinación de los movimientos, con el fin de que el vehículo no tripulado pueda alcanzar un punto de destino, siguiendo las directivas desde el Cloud.

Keywords: Cloud Robotics, Edge Robotics, Robótica, Amazon Web Services, IoT, Procesamiento de imágenes, planificación de caminos, Multi-Robot, RRT

1. Introducción

Considerando la gran influencia de la robótica en la sociedad y la industria, y la diversidad de servicios ofrecidos por robots, nos encontramos que los mismos presentan grandes limitaciones en consumo de energía, poder de cómputo, capacidad de almacenamiento, toma de decisiones, tareas cognitivas, entre otras. Desde el año 2010, comenzaron a surgir proyectos de investigación (ej: RoboEarth [1]), que integran las tecnologías de Cloud con los sistemas de robots. Es así, que James Kuffner propone el concepto de Cloud Robotics, basado en combinar las tecnologías de robots con el paradigma de Cloud Computing [2].

La idea de Cloud Robotics, es permitir por medio de aplicaciones tratar los datos de los componentes de hardware del robot (sensores, actuadores, cámaras, memoria, etc.), sin importar las limitaciones de cómputo y almacenamiento de las placas de desarrollo de los robots [3]. En otras palabras, este concepto permite a los robots obtener resultados de tareas de cómputo intensivo, tales como: procesamiento de imágenes, reconocimiento de voz, determinación de rutas, confección de mapas, acciones cognitivas, etc., sin tratamiento local, sino en el Cloud [4].

Debido al alto incremento de los robots conectados y a la gran cantidad de información generada, se han presentado limitaciones con respecto a problemas de latencia, tiempos de respuestas variables, calidad de servicio, privacidad y seguridad, que han dado origen a la extensión del paradigma, realizando todo el cómputo que se requiera y sea factible de ejecutar, en el extremo de la red; a esto se lo denomina Edge Robotics [5] [6].

Estos paradigmas brindan la capacidad de establecer escenarios para Multi-Robots, donde cada robot se integra de un hardware mínimo con conectividad inalámbrica, y los datos de los sensores y la adquisición de imágenes se procesarán de forma local, intermedia o en el Cloud, según la capacidad y conveniencia del uso de los recursos [7].

2. Conceptos elementales

- **Amazon Web Services**

  Es una plataforma de Cloud Público, que provee "Infraestructura como servicio"; a través de la tecnología de virtualización. De sus servicios más destacados, utilizamos "Elastic Compute Cloud" (EC2) y "AWS Internet of Things" (AWS IoT) [8].

- **Internet of Things (IoT)**

  Es un nuevo paradigma que considera a los dispositivos como "objetos" conectados a Internet, utilizando protocolos de comunicación estándar [9].

- **Protocolo Message Queuing Telemetry Transport (MQTT)**

  Es un protocolo de comunicación ligero [10] [11], diseñado para tolerar conexiones intermitentes y reducir los requisitos de ancho de banda de red; soporta comunicación segura con TLS y calidad de servicio (QoS).

  Está basado en el modelo publicación/suscripción, como muestra la figura siguiente:

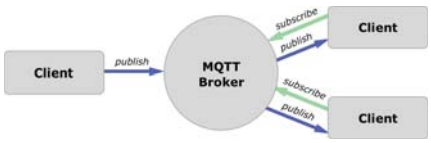

Figura 1: Comunicación MQTT

- **Amazon Web Servicies Internet of Things (AWS IoT)**

  Es un servicio que proporciona AWS con el fin de conectar, administrar y operar grandes conjuntos de dispositivos [12]. Para cada dispositivo, se emite un certificado y un par de llaves privada-pública, que junto al certificado de la Entidad Certificante (CA), permite conectar el dispositivo al servicio de AWS IoT, en forma segura.

- **Node-RED**

  Es una herramienta de programación visual WEB, orientada a flujos que son almacenados en formato JSON, y permite comunicar hardware y servicios de forma rápida y escalable, utilizando nodos MQTT sobre conexiones TLS [13]. En la figura 2 puede observarse un ejemplo de uno de los flujos Node-RED utilizado en el Auto-Robot.

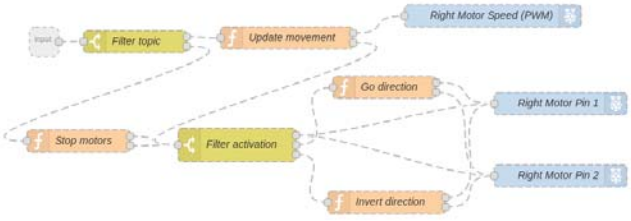

Figura 2: Ejemplo de flujo Node-RED del auto robot

3. Trabajo experimental

El problema a resolver en este trabajo consiste en guiar a un dispositivo de vuelo no tripulado (Dron) para sobrevolar un escenario compuesto con elementos que simulan objetos del mundos real. Cada objeto se identifica por medio de figuras geométricas; el Dron realizará capturas de imágenes del entorno mencionado, con el fin de detectar un vehículo de 4 ruedas no tripulado (Auto-Robot), un punto destino a alcanzar por este último y diversos obstáculos que se deberán sortear, para que el Auto-Robot llegue al destino. Las imágenes adquiridas requieren ser procesadas en tiempo real, con el objeto de reconocer cada uno de los elementos. Luego, se necesita planificar el camino óptimo que conecte el Auto-Robot con el punto de destino; y por último, se precisa determinar los desplazamientos que el Auto-Robot deberá efectuar para abordar el punto de destino final.

Además, se requiere diseñar una infraestructura que permita la interconexión de los diferentes dispositivos, en forma segura y con tiempos de respuesta aceptables.

Para el presente trabajo se utilizarán dispositivos de bajo recursos con limitaciones de cómputo. Tales limitaciones se subsanarán aprovechando los beneficios del Cloud.

Sin embargo, existen ocasiones en las cuales no será posible delegar la toma de decisión al Cloud, ya que existen acciones de máxima prioridad que el Auto-Robot debe ejecutar en forma inmediata; como son de común conocimiento las desventajas que proporciona la "dependencia de la red" (atenuación en la latencia y aún pérdida de conectividad), que puede provocar que el Auto-Robot se desvíe del camino y colisione con los obstáculos. Para evitar esta problemática es deseable aplicar los conceptos del paradigma de Edge Robotics, con el fin de procesar la información de los sensores de colisión en forma local y tomar la decisión de avanzar o frenar inmediatamente.

Asimismo, es deseable contar con un dispositivo intermedio que permita direccionar las imágenes que adquirirá el Dron hacia el Cloud, aprovechando los beneficios de la conectividad Ethernet con respecto a WiFi; como así también, controlar la navegación del Dron en forma segura.

Para alcanzar los requerimientos antes mencionados, se ha confeccionado la estructura del sistema con los siguientes cuatro componentes: el Auto-Robot, el Dron, el dispositivo intermedio (DI) y la instancia de cómputo EC2 en el Cloud de AWS (IEC2).

Sobre la componente IEC2 se ha desarrollado una interfaz WEB, que permite al usuario dar inicio al sistema y visualizar el comportamiento del mismo. Al inicio del sistema, la IEC2 notifica al componente DI, y este último es quien se encarga de controlar la navegación del Dron. Todas las imágenes capturadas, son comprimidas y posteriormente redireccionadas por el componente DI a la IEC2 vía Ethernet.

En la IEC2, cada imagen se descomprime y se procesa para detectar los diferentes objetos y características del escenario. En base a los datos obtenidos, se procede a planificar el camino óptimo, para posteriormente calcular y enviar el movimiento que el Auto-Robot deberá efectuar para desplazarse.

Por otro lado, el Auto-Robot aplica el movimiento recibido; cabe destacar aquí, que a lo largo de la vida del sistema, el Auto-Robot adquiere y procesa los datos del sensor de colisión de forma local, con el fin de evitar una incidencia.

Una vez que el Auto-Robot aborda el punto de destino, es la IEC2 quien se encarga de detectar dicho evento y notificar, tanto al Auto-Robot como al DI, que el sistema debe finalizar. Así, el Auto-Robot concluye su desplazamiento apagando los motores y el sensor de colisión; mientras que el Dron finaliza la captura de imágenes y retorna a su posición original para finalmente aterrizar.

Durante todo el tiempo de ejecución del sistema, el usuario puede visualizar en la interfaz WEB todos los eventos ocurridos en tiempo real, como así también la traza calculada junto

con las capturas de imágenes.

Las siguientes secciones detallan el desarrollo efectuado en cada uno de los componentes, como la integración de los mismos para lograr el trabajo propuesto.

3.1. Auto-Robot

Se ha confeccionado un prototipo que ejecuta los movimientos recibidos desde la IEC2, a través del protocolo MQTT. También, se ha desarrollado y provisto al Auto-Robot de toda la lógica para traccionar los motores ante las órdenes recibidas y procesar los datos extraídos del sensor de colisión, para evitar chocar con algún obstáculo.

El prototipo se abstrae de la capa lógica necesaria para la planificación del camino y el reconocimiento del escenario. El vehículo se restringe a seguir las órdenes provenientes de la IEC2; esto permite confeccionar Robots económicos, de bajo recursos y consumo energético.

3.1.1. Características y componentes

Se detallan las diferentes componentes y características pertinentes al Auto-Robot:

- **Estructura**: se ha adaptado un chasis de 4 ruedas, que simula un vehículo tradicional. Sus dimensiones son: 248mm x 146mm x 70mm. Proporciona tracción trasera (RWD) mediante un motor y dirección delantera a través de un servo que orienta las ruedas.
- **Raspberry Pi 3**: placa de bajo costo de desarrollo que brinda conexión WiFi, almacenamiento en memoria MicroSD y alimentación de energía de 5v.
- **RaspiRobot Board V3**: módulo que brinda la interfaz de conexión entre la RPi y los motores; suministra energía a estos componentes.
- **Fuente conmutada Step-Down**: regula la energía de salida de 1,25 a 36V.
- **Sensor Ultrasonido HC-SR04**: sensor de proximidad para evitar colisiones.

3.1.2. Funcionalidad

El vehículo es el actor que se encarga de cumplir con los movimientos enviados por la IEC2. Para esto, se diseñó un flujo de programación en Node-RED, el cual permite modificar el comportamiento del dispositivo, ya sea alterando la dirección, orientando sus ruedas delanteras, o el desplazamiento mediante la activación/desactivación del motor de tracción trasera. Asimismo, el flujo permite modificar la velocidad del motor.

Además, el Auto-Robot procesa los datos obtenidos de su sensor de colisión y toma la decisión de desactivar el motor cuando se detecta que el dispositivo se encuentra a menos de 5 centímetros de un obstáculo.

Figura 3: Sistema de direccionamiento del Auto-Robot

Figura 4: Auto-Robot de completo

Figura 5: Sistema de motor del Auto-Robot

3.2. Dron

Para integrar al sistema propuesto, se ha adquirido un Dron de marca Parrot, modelo Bebop 1 (figura 6), que se utiliza para capturar y enviar imágenes del escenario a la IEC2.

Figura 6: Parrot Bebop 1.

El Dron proporciona video en tiempo real comprimido en formato H.264, para posteriormente compartir las imágenes por medio de paquetes Real-Time Transfer Protocol (RTP). El DI se compone por una placa de desarrollo Raspberry Pi 3 con conexión a Internet vía Ethernet, lo que acelera la transmisión de los paquetes (figura 7).

Cuando el Dron recibe la notificación de inicio del sistema, se eleva a una altura preestablecida (1,5 metros) y se desplaza hacia el centro del escenario. El Dron provee de una cámara que tiene una visión de 90° orientada al suelo. Al recibir la notificación de finalización del sistema, el Dron retorna a su posición inicial y termina con la transmisión del video.

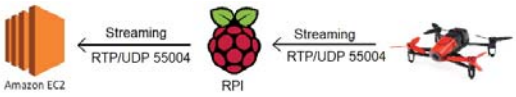

Figura 7: Camino que realiza el streaming de video.

Con el propósito de controlar los movimientos del Dron, se utilizó la librería desarrollada por la empresa Parrot, que permite ejecutar comandos en el dispositivo. Para ello, el DI recibe vía MQTT las directivas desde la nube y este los traduce en comandos que envía al Dron. En la figura 8 se puede observar la conexión del Dron con la IEC2.

Figura 8: Conexión del Dron, DI y la IEC2.

3.3. Instancia EC2 en AWS (IEC2)

Se utilizó una instancia del servicio EC2 sobre el Cloud de AWS, alojada en EE.UU. La instancia adquirida es del tipo T3.small, posee 2 CPU Core, 2GB de RAM y 8GB de almacenamiento SSD. Se desplegó la instalación del S.O. Linux Ubuntu 18.04 LTS.

Dicha instancia es utilizada para descomprimir y procesar las imágenes del entorno, como también para planificar y actualizar las rutas óptimas y calcular los movimientos a realizar por el vehículo. Además, brinda una interfaz WEB que permite gestionar y controlar el sistema. Asimismo, se desarrolló un simulador que permite generar diversos escenarios con el fin de testear la planificación.

Otro aspecto importante de la instancia, es que se encarga de la comunicación con los dispositivos mediante el protocolo de comunicación MQTT, que utiliza un canal de comunicación seguro sobre TLS, basado en certificados de seguridad.

3.3.1. Procesamiento de imágenes

El objetivo de este procesamiento es detectar los elementos del escenario, es decir, encontrar en la imagen tanto el Auto-Robot, como el punto de destino y los obstáculos; y obtener las propiedades de los mismos (coordenadas del centro, radio, orientación, etc).

Los elementos nombrados anteriormente son identificados con figuras geométricas para simplificar el proceso de detección. El punto de destino es representado con un círculo, los obstáculos con un triángulo y el Auto-Robot con un cuadrado o rectángulo, con un círculo interior, que denota su orientación. El procesamiento de las imágenes se realiza mediante el uso de la librería OpenCV v3.0.

En una primer instancia, se obtiene una imagen del streaming de video (figura 9), la misma se convierte a escala grises (figura 10) y se procede a aplicar una colección de filtros con tal de facilitar la detección de las figuras.

Figura 9: Imagen original

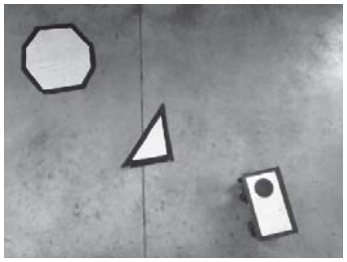

Figura 10: Imagen en escala de grises

Se aplica un primer filtro de suavizado (figura 11) y luego se procede a la aplicación de un segundo filtro que detecte bordes, como se puede observar en las figuras 12 y 13

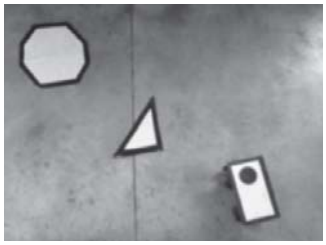

Figura 11: Imagen con suavizado Gaussiano.

Figura 12: Imagen con filtro “Canny Edge”

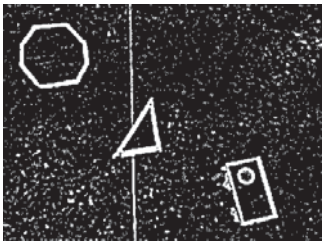

Figura 13: Imagen con filtro “Threshold”

Posteriormente, se obtienen los contornos y se itera por cada uno de ellos verificando la cantidad de vértices que posea. En caso de tener tres lados, se considera un triángulo, cuatro lados es un cuadrado o rectángulo y más de cinco es un círculo.

3.3.2. Planificación de caminos y cálculo de movimientos

La determinación de trayectorias consiste en encontrar aquella ruta que permita al Auto-Robot desplazarse desde su posición inicial hasta llegar al destino, sorteando los obstáculos presentes en el escenario. Para lograr este objetivo, es necesario utilizar un algoritmo que

detecte la ruta a realizar y un método de transformación y conversión que traduzca dicha ruta en movimientos entendibles por el Auto-Robot.

3.3.3. Planificación RRT

La técnica seleccionada es la denominada Rapidly Exploring Random Tree (RRT).

Un algoritmo RRT se basa en la construcción de un árbol de configuraciones que aumenta desde un punto origen hacia un destino preestablecido. Este método tiene como objetivo generar un árbol de exploración, que permita cubrir de manera uniforme todo el espacio de configuraciones libres de colisión.

El algoritmo recibe como entradas el punto de inicio y fin, el tiempo máximo, y la distancia entre los nodos (es decir, la longitud en píxeles de las ramas del árbol). Como resultado final, se obtendrá un árbol (conjunto de puntos), que representa el camino encontrado [14].

En las figuras 14 y 15, se puede observar el resultado de simular la ejecución del algoritmo RRT, tomando como punto de inicio la ubicación del vehículo (punto azul), el punto de destino (punto verde) y los obstáculos de por medio (puntos rojos). Como resultado de la planificación se obtiene una colección de nodos que representan una coordenada (x, y).

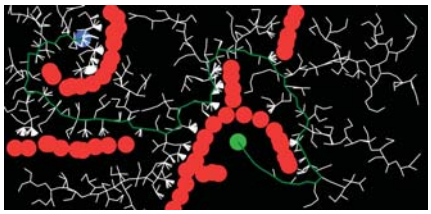

Figura 14: Ejemplo 1: de planificación RRT

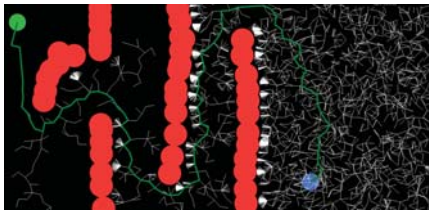

Figura 15: Ejemplo 2: de planificación RRT

3.3.4. Optimización del camino

El algoritmo RRT, como se mencionó, retorna un árbol que consiste en una lista de puntos (x, y). Como se puede apreciar en las figuras 14 y 15, el camino resultante cuenta con una gran variedad de desvíos innecesarios, lo que repercutirá en un mayor esfuerzo por parte del Auto-Robot al momento de desplazarse.

Para solucionar este problema, se propone un método de optimización del camino que consiste en crear conexiones lo más longitudinales posibles entre dos nodos extremos, removiendo como consecuencia los puntos que no forman parte de esa conectividad. El nodo más lejano se obtiene generando una búsqueda dicotómica en la lista de coordenadas.

El resultado de esta operación consiste en una reducción de los nodos del árbol RRT, como muestran las figuras 16 y 17.

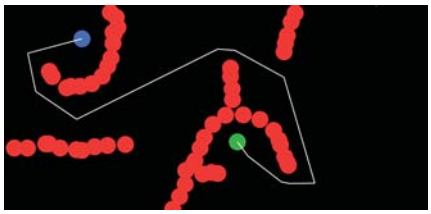

Figura 16: Ejemplo 1: camino optimizado

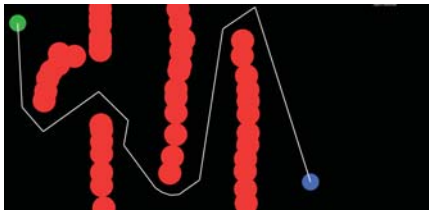

Figura 17: Ejemplo 2: camino optimizado

3.3.5. Suavizado del camino

Si bien la ruta obtenida en el paso anterior contiene menos inflexiones, es evidente que el Auto-Robot no podrá cumplir con exactitud la trayectoria, como consecuencia de los desvíos pronunciados que derivan de la unión de las diferentes partes del camino.

Para resolver este inconveniente, se aplica una función matemática de interpolación cúbica (o del inglés "cubic spline"), que permite suavizar el camino y posibilita que el vehículo se oriente en todo momento, para no tener que direccionar una gran cantidad de grados y evita realizar una curva pronunciada.

Esta función trata los puntos de la ruta anterior y retorna una nueva colección de puntos que conforman la curva suavizada. El algoritmo utilizado permite establecer la distancia en la que debe estar cada uno de los puntos de la curva, como se observa en la figura 18 y 19.

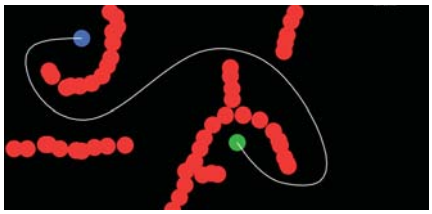

Figura 18: Ejemplo 1: trayectoria suavizada

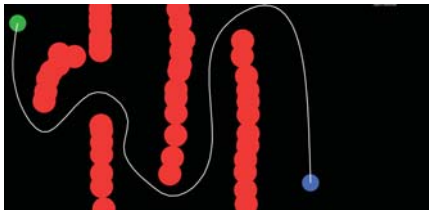

Figura 19: Ejemplo 2: trayectoria suavizada

3.3.6. Cálculo de la maniobra

Luego de obtener la ruta resultante, se procede a calcular el movimiento que el Auto-Robot deberá ejecutar. Dicho valor está conformado por el grado de giro ([-180°, 180°]) y el sentido de desplazamiento (1 hacia delante, -1 hacia atrás, 0 freno).

Se implementó una técnica que consiste en incorporar a nivel lógico un paralelogramo en los límites del Auto-Robot. Este polígono abarca el área de movimiento del vehículo, por lo que se requiere conocer el ángulo de giro máximo que posee el auto (este dato es enviado por el Auto-Robot en la interacción inicial previo al comienzo de la ejecución).

Como se puede observar en las figuras 20 y 21, si dentro del área de colisión se detecta que existe un obstáculo, se intenta revertir el sentido, siempre en función del punto de referencia calculado previamente.

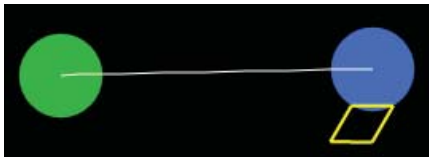

Figura 20: Sin presencia de obstáculo en el camino

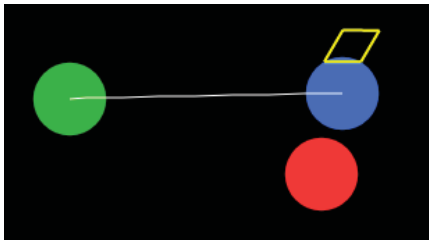

Figura 21: Con presencia de obstáculo en el camino

3.3.7. Plataforma WEB

Se desarrolló una interfaz WEB que permite al usuario interactuar con el sistema. Por un lado, se despliega una sección que posibilita simular la determinación de caminos, con

la propiedad de configurar diferentes aspectos, como por ejemplo el radio de los objetos, el grado al que está orientado el Auto-Robot, etc.

Por otro lado, se presenta una sección denominada tiempo real o "live", utilizada para dar comienzo con la ejecución del sistema y visualizar en todo momento los eventos que ocurren en el transcurso del mismo, como también el streaming de video original y luego de procesado, junto con la traza resultante.

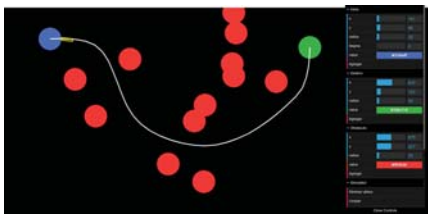

Figura 22: Interfaz de simulación

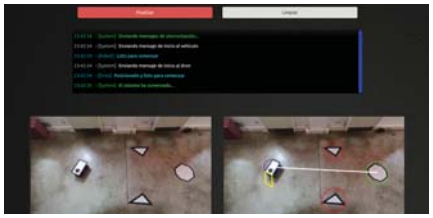

Figura 23: Interfaz live

4. Resultados obtenidos y líneas de trabajo a futuro

En este trabajo de investigación se analizó, diseñó, desarrolló e implementó un sistema de Multi-Robot conectados al Cloud público de Amazon Web Service. Para esto se aplicaron los conceptos de los paradigmas de Cloud y Edge Robotics.

Se confeccionó un Auto-Robot, un vehículo de 4 ruedas no tripulado, que simula un auto de tracción trasera tradicional. Se analizó, diseñó, desarrolló y se implementó la lógica que le otorga la capacidad de desplazarse para alcanzar un punto de destino, mediante las órdenes provenientes desde el Cloud; como así también, evitar toda posible colisión por medio del procesamiento de datos obtenidos por un sensor de colisión en forma local.

El sistema también lo integra un Dron no tripulado, que sobrevuela el escenario y envía imágenes al Cloud. Se adhirió un dispositivo intermedio, que permite gestionar y controlar el Dron bajo las decisiones y ordenes efectuadas en la instancia de cómputo del Cloud (IEC2). Con respecto al medio de conectividad a Internet, se optó por la comunicación vía Ethernet, pues permite reducir la latencia lo menor posible en el envío de las imágenes a la nube.

Se adquirió y desplegó una instancia de servidor en EC2 de AWS, con el fin de delegar la intensidad del cómputo del sistema a los recursos ofrecidos por la nube; esto ha posibilitado la confección de robots equipados con un hardware de requerimientos mínimos, lo cual permite soluciones más económicas y de menor consumo energético.

También, se desarrolló los algoritmos que permiten procesar imágenes para detectar las figuras geométricas que identifican a cada objeto del escenario. Juntamente, se ha probado y seleccionado diversos filtros de imágenes, aptos para escenarios con diferente luminosidad.

Así mismo, se diseñó y desarrolló un método de planificación de caminos que permite obtener la ruta óptima desde un punto de inicio hasta un punto de destino determinado. Para alcanzar tal objetivo, se utilizó el algoritmo de planificación RRT, el cual se adaptó a los requerimientos del sistema, y se desarrolló un método de optimización del camino, que simplifica la trayectoria a transitar. Además, se diseñó un algoritmo de suavización de ruta, que permite al Auto-Robot ejecutar un desplazamiento con la mayor precisión posible, a pesar de las limitaciones físicas que posee el vehículo.

Por otro lado, se confeccionó un algoritmo que permite calcular el movimiento a realizar por el Auto-Robot, contemplando un área de posible colisión con el fin de determinar la presencia de un obstáculo en su camino.

Por último, se implementó una interfaz WEB que brinda distintas funcionalidad, entre ellas, permite simular planificaciones de escenarios creados por el usuario o en base a imáge-

nes reales. También, permite que el usuario tenga el control de iniciar o finalizar ejecución del sistema, visualizando en tiempo real todos los eventos que se generan, como así también, las imágenes del escenario, que también incluyen la detección de los objetos junto con la traza a seguir por el Auto-Robot.

Es deseable destacar, que ante la necesidad del sistema de operar en tiempo real, se han implementado algoritmos lo más óptimos, eficientes y precisos posible.

Podemos concluir qué la aplicación de los conceptos de Cloud y Edge Robotics, colaboraron a la confección y despliegue del sistema propuesto; en base a los mismos, se ha optado por realizar el procesamiento de los datos en forma local, intermedia y la nube, conforme a la necesidad, capacidad y la conveniencia del cómputo.

Nos hemos propuesto como trabajo a futuro, dotar al sistema con la capacidad de procesar y reconocer imágenes de objetos reales, con el fin de descartar la identificación de los mismos en base a figuras geométricas. Por otro lado, adaptar el sistema para integrar más de un Auto-Robot, y que los mismos trabajen colaborativamente; para esto, es deseable incorporar los conceptos del paradigma de Fog Robotics, añadiendo una cámara LIDAR que brinde la capacidad de mapear el escenario y compartirlo a otros Auto-Robots.

Referencias

[1] *RoboEarth*. `http://www.roboearth.org`.

[2] Kuffner J. «Cloud-enabled robots». En: *IEEE-RAS International Conference on Humanoid Robot* (2010).

[3] Wang L.; Liu M.; Meng M.; Siegwart R. «Towards Real-Time Multi-Sensor Information Retrieval in Cloud Robotic System». En: *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)* (2012).

[4] Costanzo M.; Boggia M.; Rodriguez I.; De Giusti A. «CLOUD ROBOTICs: Navegación de un vehículo autónomo en un entorno con obstáculos». En: *XII Workshop de Procesamiento Distribuido y Paralelo (WPDP) – XXII Congreso Argentino de Ciencias de la Computación (CACIC2018)* (2018).

[5] Tanwani A.K.; Mor N.; Kubiatowicz J.; Gonzalez J.E. «A Fog Robotics Approach to Deep Robot Learning: Application to Object Recognition and Grasp Planning in Surface Decluttering». En: *IEEE International Conference on Robotics and Automation, ICRA, 2019* (2019).

[6] Antevski K.; Groshev M.; Cominardi L.; Bernardos C.; Mourad A.; Gazda R. «Enhancing Edge robotics through the use of context information». En: (2018).

[7] Turnbull L. «Cloud Robotics: Formation Control of a Multi Robot System Utilizing Cloud Infrastructure». En: *Proceedings of IEEE – Southeastcon* (2013).

[8] *Amazon Web Services*. `https://aws.amazon.com/es/`.

[9] Atzori L.; Iera A. «The Internet of Things: A survey». En: *ELSEVIER Journal Computer Networks* (2010).

[10] *Protocolo MQTT*. `https://www.oasis-open.org`.

[11] *OASIS MQTT*. `http://mqtt.org`.

[12] *Amazon Internet Of Things*. `https://aws.amazon.com/es/iot/`.

[13] *Node-RED*. `https://nodered.org`. Último acceso 30 de Noviembre 2018.

[14] Zulfiqar Habib Iram Noreen Amna Khan. «A Comparison of RRT, RRT* and RRT*-Smart Path Planning Algorithms». En: (2016).