UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA ELECTRÓNICA

DESARROLLO DE UNA PLATAFORMA DE CONTROL PARA EL ROBOT MÓVIL ROOMBA UTILIZANDO ROS E INTEGRACIÓN DE UN SENSOR KINECT

Por:
Elvis Emmanuel Ruiz Álvarez

PROYECTO DE GRADO
Presentado ante la Ilustre Universidad Simón Bolívar
como requisito parcial para optar al título de
Ingeniero Electrónico

Sartenejas, Marzo de 2012

UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA ELECTRÓNICA

DESARROLLO DE UNA PLATAFORMA DE CONTROL PARA EL ROBOT MÓVIL ROOMBA UTILIZANDO ROS E INTEGRACIÓN DE UN SENSOR KINECT

Por:
Elvis Emmanuel Ruiz Álvarez

Realizado con la asesoría de:
Prof. Raúl Eduardo Acuña Godoy

PROYECTO DE GRADO
Presentado ante la Ilustre Universidad Simón Bolívar
como requisito parcial para optar al título de
Ingeniero Electrónico

Sartenejas, Marzo de 2012

UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA ELECTRÓNICA

DESARROLLO DE UNA PLATAFORMA DE CONTROL PARA EL ROBOT MÓVIL ROOMBA UTILIZANDO ROS E INTEGRACIÓN DE UN SENSOR KINECT

Proyecto de grado por: Elvis Emmanuel Ruiz Álvarez. Carnet: 06-40267

Realizado con la asesoría de: Prof. Raúl Eduardo Acuña Godoy

RESUMEN

En el Grupo de Mecatrónica de la Universidad Simón Bolívar, se dispone de una variedad de plataformas robóticas sobre las que se aplican esquemas de operación y control que requieren altas capacidades de procesamiento. En este sentido, se desarrolló un controlador para el robot móvil Roomba utilizando ROS (*robot operating system*), el cual permitió el control del móvil utilizando referencias vectoriales de velocidad. A dicho sistema se le integró el sensor Kinect y se le proveyó de comunicación con procesos en otras instancias de red para monitoreo y posible control remoto. Se aseguró que el sistema fuera lo suficientemente estándar y flexible para ser replicado, en un futuro, en otros robots Roomba. En relación a las aplicaciones que se desarrollaron para establecer la conectividad del sistema, a nivel de programación, se alcanzó controlar el móvil a partir de referencias vectoriales, suministradas por el procesamiento de información del entorno. Adicionalmente, se consiguió que el Roomba pudiera navegar en un pasillo. Por otra parte, se exportó la información para realizar una detección de bordes estimada para el pasillo, esto con procesamientos externos al desarrollo de este trabajo.

Palabras claves: controlador, Kinect, navegación, referencias, ROS, vectores.

DEDICATORIA

A un ciego en el mundo del rey tuerto.

AGRADECIMIENTOS

Por la oportunidad.

Por el apoyo.

Por la confianza.

Gracias a todos ustedes.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

LISTA DE SÍMBOLOS

Unidades

m	Metro
g	Gramo
rad	Radianes
°	Grados
V	Voltio
A	Ampere
s	Segundo
Hz	Hertz.

Cantidades físicas y otros símbolos

V	Velocidad	[m/s]
φ	Angulo	[rad]
t	Tiempo	[s]
f	Frecuencia	[Hz]
r	Radio.	[m]
Vcc	Voltaje de corriente continua	[V]

LISTA DE ABREVIATURAS

BSD: Berkeley Software Distribution (distribución de software Berkeley)

DC: Direct Current (corriente directa).

Fps: Frames per Second (cuadros por segundo).

GPL: General Public License (licencia pública general).

ICR: Irobot Create Roomba.

IDL: Interface Definition Language (lenguaje de especificación de interfaces).

IR: Infrarrojo.

LAN: Local Area Network (red de área local).

OS: Operating System (sistema operativo).

PCA: Point Cloud Library (librería de nube de puntos).

QVGA: Quarter Video Graphics Array (cuarto de la matriz de gráfico de video).

RGB: Red – Green – Blue (rojo – verde - azul).

ROS: Robot Operating System (sistema operativo de robots).

RMR: Robot Móviles con Ruedas

SDK: Software Development Kit (kit de desarrollo de software).

SLAM: Simultaneous Localization and Mapping (localización y mapeado simultáneo).

USB: Universal Serial Bus (bus universal en serie).

VGA: Video Graphics Array (matriz de gráfico de video).

INTRODUCCIÓN

La robótica móvil ha tenido un gran auge desde la década de los 70, desde la llegada de tecnologías de planificación y razonamiento automático, por lo que se ha dado pie a las investigaciones y al diseño de plataformas móviles instrumentadas, con miras al refinamiento del funcionamiento mecánico y electrónico, lo que ha permitido aumentar la complejidad de los sistemas, mejorando las prestaciones de los mismos, posibilitando el planteamiento y búsqueda de nuevas y más ambiciosas metas. De esta manera, lograr autonomía y empatía de los procesos de toma de decisiones, de forma efectiva, segura y confiable, todo esto son condiciones que se le espera dar a los sistemas robóticos.

A partir de la versatilidad de los robots móviles, se cuenta con la oportunidad de navegar en distintos terrenos y tener aplicaciones como: exploración minera, exploración planetaria, misiones de búsqueda y rescate de personas, limpieza de desechos peligrosos, automatización de procesos, vigilancia, reconocimiento de terreno, entre otras [1].

En la actualidad, se busca la mejora de los sistemas robóticos por medio del aumento de la eficiencia de la programación de múltiples tareas y de la compatibilidad e integración de nuevos equipos, sensores y aplicaciones. Esto, unido con la elaboración de sistemas autónomos, semi-autónomos y de operatividad remota desliga la necesidad de contar con un usuario presente en el área de actividad y en un constante monitoreo.

En el Grupo de Mecatrónica de la Universidad Simón Bolívar, dentro del cual se desarrolla el presente trabajo, se dispone de una variedad de plataformas robóticas móviles que van desde carros con ruedas diferenciales hasta robots caminantes, helicópteros y submarinos; sobre las que se aplican esquemas de

operación y control que requieren altas capacidades de procesamiento, como son el control por redes neuronales, lógica difusa y visión artificial.

En este sentido, se desea el desarrollo de una plataforma de control para el robot móvil Roomba, utilizando referencias vectoriales de velocidad, en el que, el sistema debe proveer comunicación con procesos en otras instancias de red para monitoreo y posible control remoto. Se requiere además la incorporación de un sensor Kinect y el acceso a los datos de dicho sensor en plataformas externas al sistema. El sistema debe ser lo suficientemente estándar y flexible para ser replicado en otros robots Roomba en el futuro.

OBJETIVO GENERAL

El objetivo principal del presente trabajo es desarrollar un controlador utilizando ROS (*robot operating system*) el cual permita el control del robot Roomba utilizando referencias vectoriales de velocidad, dicho sistema debe integrar el sensor Kinect para su posterior utilización en trabajos futuros.

OBJETIVOS ESPECÍFICOS

Puntualizando, el proyecto puede ser descompuesto en los siguientes objetivos:

a) Estudio del estado del arte en control de robots con ruedas diferenciales.

b) Estudio del robot Roomba, sensores, limitaciones, otras aplicaciones.

c) Investigar la eficiencia de ROS para control de robots móviles.

d) Utilizar ROS como plataforma de control del Roomba.

e) Integrar el sensor Kinect al robot Roomba.

f) Generar un servicio de los datos del Kinect a una plataforma externa al robot.

g) Evaluar el desempeño del sistema.

CAPÍTULO 1

MARCO TEÓRICO

1.1. ROBOTS MÓVILES

Los robots móviles se pueden clasificar por el tipo de locomoción utilizado. En general, los tres medios de movimiento son: por ruedas, por patas y orugas [1]. Cabe señalar que aunque la locomoción por patas y orugas han sido ampliamente estudiadas, el mayor desarrollo se presenta en los Robots Móviles con Ruedas (RMR). Dentro de los atributos más relevantes de los RMR, destacan su eficiencia en superficies lisas y firmes, a la vez que no causan desgaste en la superficie donde se mueven y requieren un número menor de partes y menos complejas.

Se puede definir un robot móvil de ruedas como un sistema electromecánico controlado, que utiliza como locomoción ruedas de algún tipo, y que es capaz de trasladarse de forma autónoma a una meta preestablecida en un determinado o indeterminado espacio de trabajo.

Se entiende como autonomía (reactiva) de un robot móvil, al dominio que tiene éste para determinar su curso de acción, mediante un proceso propio de razonamiento en base a sensores, en lugar de seguir una secuencia fija de instrucciones.

Existen diferentes configuraciones cinemáticas para los RMR, estas dependen principalmente de la aplicación hacia dónde va enfocado, no obstante, de manera general se tienen las siguientes configuraciones: Ackerman, triciclo clásico, tracción

diferencial, *skid steer*, síncrona y tracción omnidireccional, ver Figura 1.1. Dependiendo de la configuración cinemática que lo conforme, los RMR utilizan cuatro tipos de ruedas para su locomoción, estas son: convencionales, tipo castor, ruedas de bolas y omnidireccionales, ver Figura 1.2. En el marco de las configuraciones cinemáticas posibles y las ruedas que estas utilizan, los RMR documentados en la literatura utilizan comúnmente la configuración de tracción diferencial, donde se utilizan ruedas convencionales, como ruedas motrices y una o dos ruedas tipo castor, de bola u omnidireccionales, respectivamente, para proveer de estabilidad al móvil.

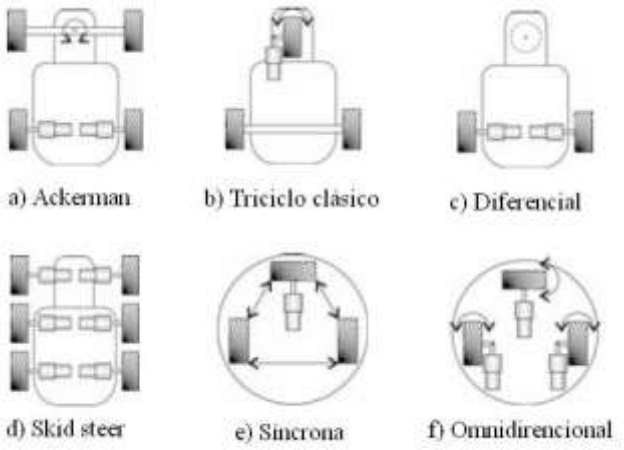

Figura 1. 1. Configuraciones de los RMR [1].

Figura 1. 2. Tipos de Ruedas [1].

Con el objeto de hacer más tratable el problema del modelado en las configuraciones cinemáticas, se suelen establecer algunas suposiciones de diseño y de operación [2]. Por una parte, dentro de las suposiciones de diseño generalmente se toman tres. La primera va dirigida a considerar que las partes dinámicas del RMR son insignificantes, que no contiene partes flexibles, de esta manera pueden aplicarse mecanismos de cuerpo rígido para el modelado cinemático. La segunda limita que la rueda tenga a lo más un eslabón de dirección, con la finalidad de reducir la complejidad del modelado. La tercera es suponer que todos los ejes de dirección son perpendiculares a la superficie, de esta manera se reducen todos los movimientos a un solo plano.

Por otra parte, respecto a las suposiciones de operación, al igual que en las de diseño, se toman tres. Una de ellas descarta toda irregularidad de la superficie donde se mueve el RMR. Otra, considera que la fricción de traslación en el punto de contacto de la rueda con la superficie donde se mueve, es lo suficientemente grande para que no exista un desplazamiento de traslación del móvil. Como complemento a lo anterior, una tercera suposición de operación establece que la fricción rotacional en el punto de contacto de la rueda con la superficie donde se mueve, es lo suficientemente pequeña para que exista un desplazamiento rotatorio. Aunque las suposiciones mencionadas son realistas, el deslizamiento que ocurre en el punto de contacto de las ruedas con la superficie se ha convertido en un tópico importante debido a las repercusiones que tiene sobre el móvil.

1.2. CONTROL DE RMR

Desde el punto de vista de la teoría de control, estos se encuentran en el área que se conoce como control de sistemas no-holonómicos, estos sistemas se caracterizan por tener un número menor de grados de libertad controlables respecto al número de grados de libertad totales, en el caso de un RMR de tracción diferencial, el número total de grados de libertad son 3 (posición X, Y y su

orientación φ) sin embargo únicamente se puede controlar el desplazamiento hacia adelante y hacia atrás así como su orientación, quedando como incontrolable el desplazamiento transversal.

Por lo tanto, matemáticamente se dice que el sistema está sujeto a restricciones no integrables en las velocidades, es decir, su plano de velocidades se encuentra restringido. El control del movimiento de los RMR, a grosso modo, se puede clasificar en cuatro tareas fundamentales; localización, planificación de trayectoria, seguimiento de la misma y evasión de obstáculos.

1.3. SISTEMAS ODOMÉTRICOS

La odometría es una técnica que tiene por objeto estimar la posición y orientación de un vehículo a partir del número de vueltas dadas por sus ruedas [3]. La idea fundamental de la odometría es la integración temporal del movimiento, lo cual lleva inevitablemente a la acumulación de errores. La ventaja de la odometría reside en su simplicidad, bajo costo y en que permite tasas muy altas de muestreo. Sin embargo, además de necesitar una calibración debido al desgaste de las ruedas, ésta técnica es vulnerable a las imprecisiones originadas por el deslizamiento de las ruedas, las irregularidades del terreno y las variaciones en la carga transportada.

La odometría es una parte importante de los sistemas de navegación en robots. En la actualidad es una técnica ampliamente usada en robots móviles y para ello se emplean codificadores ópticos de elevada precisión montados sobre los ejes de las ruedas que permiten llevar una cuenta bastante precisa del número de vueltas (y fracción) que estas realizan. Para la estimación se requiere el registro odométrico de al menos dos ruedas del vehículo.

1.4. SEGUIMIENTO DE TRAYECTORIA Y EVASIÓN DE OBSTÁCULOS

En relación puntual al tópico de seguimiento de trayectoria, existen diferentes métodos para lograr esta tarea, una de ellas es el control mediante campos adaptativos de velocidad y controles diferenciales, esta técnica se basa en el control de campos de velocidad (VFC, por sus siglas en inglés) junto con un control adaptativo, aquí la trayectoria deseada se describe por un vector de velocidad tangente, la principal ventaja del VFC reside en el hecho que el error en el control del campo de velocidad impide eficientemente que el robot deje la trayectoria deseada [1].

Métodos basados en control predictivo para solucionar el problema del control no-holonómico, se emplea un algoritmo de Gauss-Newton para el control predictivo lo que permite minimizar el error de posición [1]. Por otro lado, se presentan controles por modelos predictivos o *Receding Horizon* (RH). Este tipo de control es frecuentemente utilizado en la optimización de técnicas de control en la industria, se diseña principalmente para tratar el problema de restricciones y se trata de un algoritmo de mejora que predice las salidas del sistema en base a los estados que en ese momento estén presentes y al modelo propio del sistema [1].

Uno de los métodos más empleados en el seguimiento de trayectoria es el de persecución pura (del inglés *pure pursuit*), que es un método que se basa en la geometría del robot [4], éste genera arcos entre el punto de desplazamiento del móvil y los puntos de la trayectoria a seguir, los arcos se generan de manera que resulte un seguimiento suave.

Otra técnica recientemente empleada es mediante linealización de entrada-salida la cual impone que las variables de estado tiendan asintóticamente a la trayectoria deseada, generando una dinámica remanente asociada a una variable de estado, la cual resulta ser estable [1].

En lo que respecta a la evasión de obstáculos, existen distintos métodos para llevar a cabo esta tarea. Los más relevantes son: por detección de bordes, por descomposición en celdas, por construcción de mapas y por campos potenciales artificiales [1]. En el método por detección de bordes, el algoritmo implementado, permite que el robot detecte los bordes verticales de un posible obstáculo; sin embargo, dependiendo de los sensores que hagan esta detección, es necesario que el robot se detenga cuando detecte dicho borde y compute la información necesaria para confirmar la presencia del obstáculo.

El método por descomposición en celdas, divide en celdas el espacio de trabajo del robot en un plano bidimensional, a cada celda se le asigna un valor que permite saber si existe dentro de la misma algún obstáculo. La desventaja de este algoritmo es el tiempo computacional requerido para dividir el espacio de trabajo en celdas, aunado al requerimiento excesivo de sensores para brindar una considerable precisión al dividir el espacio de trabajo. En el método por construcción de mapas, el algoritmo implementado permite crear un grafo que conecta cada punto del espacio libre de trabajo, facilitando la generación de un camino que permita al robot navegar desde su punto inicial a su punto final eliminando la incertidumbre de posibles obstáculos.

Finalmente, respecto al método de campos potenciales artificiales se supone al móvil como una partícula, de igual forma los obstáculos que lo rodean se consideran como campos que ejercen una fuerza de repulsión sobre el móvil, mientras que, su punto de arribo, es una fuerza atractiva, consiguiéndose de esta forma establecer un campo potencial que representa el ambiente en el que debe de seguir su trayectoria el robot. Esta trayectoria es generada a partir del mismo método, donde los obstáculos pueden previamente considerarse en el algoritmo o bien detectarse mediante algún tipo de sensor.

1.5. DESCRIPCION GENERAL DEL ROBOT ROOMBA

El robot "Roomba 4400", Irobot Create® (ICR), es una plataforma móvil instrumentada que se puede describir a partir de su estructura, hardware e interfaz abierta.

Posee forma de disco, con unas dimensiones de 0.330m de diámetro, 0.085m de altura máxima en estado de reposo y un peso no mayor de 2.5Kg sin carga. Además tiene a su disposición una bahía de carga de aproximadamente 1150 cm3 y se encuentra equipada con aberturas para la instalación de estructuras adicionales, ver figura 1.3.a y 1.3.b.

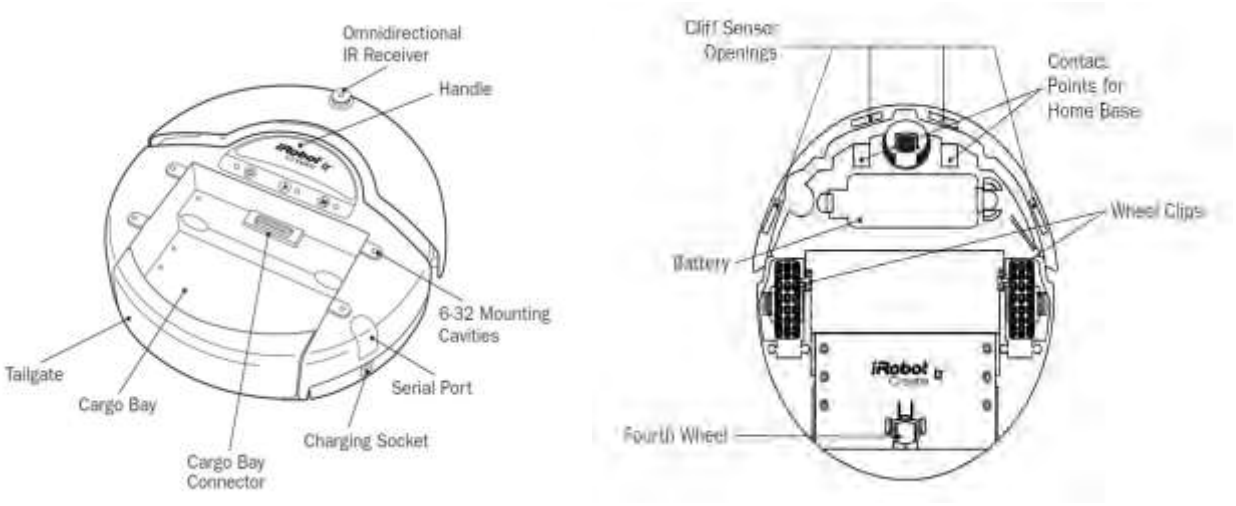

Figura 1. 3.a. Vista superior. 1. 3.b. Vista inferior.

El Irobot Create® se encuentra equipado con una gran variedad de piezas y sensores que pueden llegar a ser útiles en tareas de navegación [5], y se encuentran descritos a continuación:

a) Sensor de impacto: La plataforma tiene dos sensores de choque que le brindan la capacidad de detectar obstáculos en su desplazamiento por contacto directo. Los sensores de choque son conmutadores de dos posiciones

con muelle de retorno a la posición de reposo y se accionan por medio de una palanca. Se ubican en la parte interna del robot y la detección es producida por el impacto del parachoques (figura 1.4).

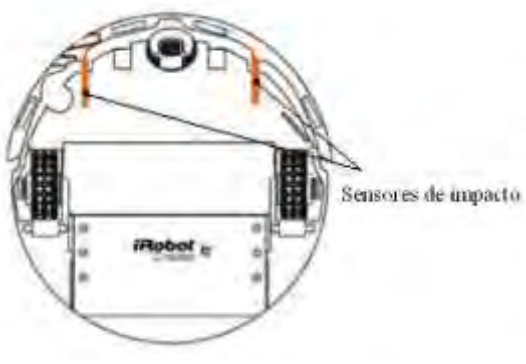

Figura 1. 4. Localización de los sensores de impacto en el IRobot Create® [5].

b) Encoders: El ICR tienes dos sensores de rotación que le permiten conocer la distancia recorrida, desplazamiento angular, ángulo de giro y sentido de giro. Estos actúan en cada una de las ruedas de forma independiente. Esto hace posible el control de velocidad, se encuentran ubicados al lado de cada rueda convencional, ver figura 1.5. La distancia es expresada en milímetros y su valor es la distancia recorrida desde su última petición hasta el momento que se hace el nuevo requerimiento, es enviada como un valor de 16 bits. Por otra parte, el ángulo es expresado en grados, su valor se calcula de igual forma que la distancia y es enviado como un valor de 16 bits.

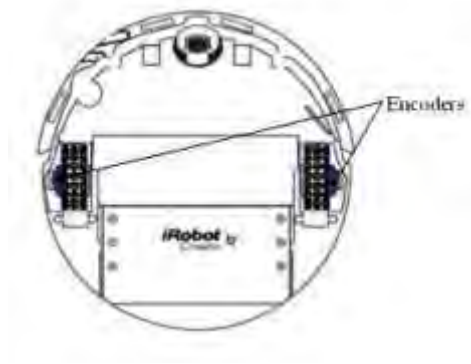

Figura 1. 5. Localización de los encoders en el IRobot Create® [5].

c) Indicadores y Botones: Los indicadores se encuentran formados por tres LEDs (ver figura 1.6 y 1.7) y un parlante (ver figura 1.6). Estos funcionan como salidas del ICR mostrando los estados del robot o cualquier información que se desee señalizar. Los botones se pueden emplear como entradas digitales para cualquier función. El botón de inicio se encarga de energizar y apagar la plataforma del ICR.

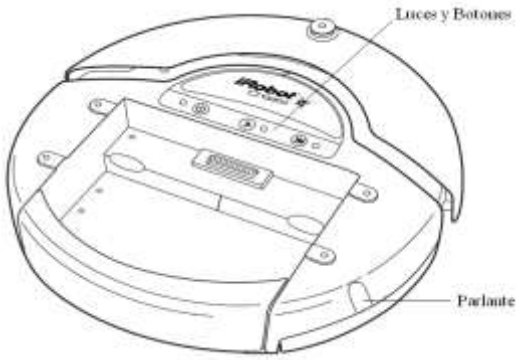

Figura 1. 6. Parlante, luces y botones.

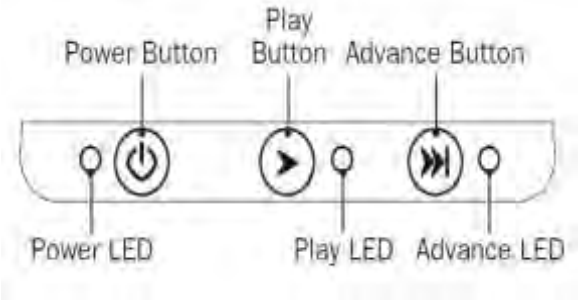

Figura 1. 7. Luces y botones.

d) Actuadores: Los encargados de la movilidad de la plataforma son dos motores DC (ver figura 1.8) que trasmiten el movimiento a las ruedas traseras a través de una trasmisión por correa, ver figura 1.8. La velocidad máxima del robot es de 0.5m/s tanto hacia delante como en reversa, puede realizar giros circulares de hasta 0.2m de radio. Con una masa menor a 2.5Kg puede acelerar a cualquier velocidad en menos de 2s [6].

Figura 1. 8.a. Posición de los motores. 1. 8.b. Tipo de Transmisión de los motores [5].

e) Ruedas: El ICR cuenta con tres ruedas en su estructura, dos ruedas traseras y una rueda libre. Las traseras comparten un mismo eje y cada una de ellas es controlada de forma independiente. El arreglo de ruedas de tipo diferencial lo que permite la ejecución de tres movimientos: Avance en línea recta, en arco y la rotación sobre su propio eje. Se puede añadir una cuarta rueda en la parte trasera del ICR para mejorar la estabilidad del robot, ver figura 1.3.b, principalmente útil cuando se añade carga a la plataforma.

f) Microcontrolador: Posee un MC9S12E128 de 16 bits, de la empresa Motorola. Es el responsable de la lectura de los sensores, manejo de actuadores, conectores físicos y permite la interpretación de los comandos de la Interfaz Abierta del ICR.

g) Baterías: Se puede emplear un paquete de 12 pilas AA de 2450 mAh y 1.5V, como fuente de energía para el ICR, o una batería recargable de níquel-hidruro metálico (Ni-MH) de 14.4V, de diseño estándar para el ICR, insertadas en su parte inferior como se muestra a continuación, ver figura 1.9.

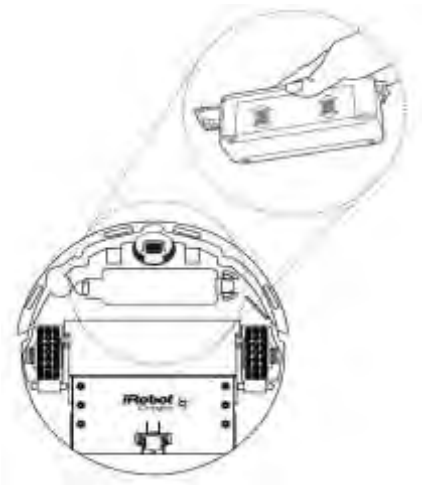

Figura 1. 9. Posicionamiento de las baterías en el IRobot Create®.

Mediante la Interfaz Abierta del ICR se puede tanto suministrar órdenes al robot como extraerle información. Para que esto se pueda llevar a cabo el ICR necesita establecer una comunicación serial con una plataforma con poder de cómputo. De esta manera se pueden emplear los comandos de la Interfaz Abierta.

Para establecer la conexión física con los controladores externos se emplea la plataforma electrónica del robot, proporcionando una comunicación serial full dúplex a niveles TTL. Consiste en: un conector DB– 25 y un conector Mini Din de 7 pines.

a) Conector DB–25: El Conector DB–25 se localiza en la parte frontal de la bahía de carga y es por donde, se puede añadir sensores adicionales al ICR, ver figura 1.3.a. Dentro de los 25 pines que contiene este conector se tienen: cuatro entradas digitales, una entrada analógica, tres salidas digitales, tres controladores de corriente útiles para el manejo de motores, un indicador del estado de carga de la batería, un trasmisor y un receptor de comunicación serial, y fuentes de voltaje regulada de 5V y no regulada conectada directamente a la batería, y una fuente conmutada de 12V.

b) BAM: De sus siglas en ingles *Bluetooth Adapter Module*, es una herramienta que permite el control inalámbrico del ICR desde una plataforma de cómputo

provista con *Bluetooth-Host*, esto debido a la creación de una conexión por un puerto serial virtual. Posee las siguientes características: Baud de 57600bps, no paridad, 8 bits de dato y 1 bit de parada. Este dispositivo se conecta en el DB-25.

c) Conector Mini Din: El Conector Mini Din se localiza en la parte lateral inferior derecha del ICR (ver figura 1.3.a), está formado por 7 pines, dos de los cuales están designados para la comunicación serial.

El software de la interfaz abierta consiste en un conjunto de comandos que permiten tener el control del robot. Estos comandos engloban el manejo de los modos, demos, actuadores, sensores, indicadores, entradas y salidas del ICR. Todos los comandos empiezan con un código de operación del tamaño de un byte, a este se le pueden añadir uno o varios bytes de datos dependiendo del comando, ver figura 1.10. De esta manera, el formato de escritura de cualquier comando se puede describir de la siguiente manera:

[Código de operación] [Dato 1 del Comando] [Dato 2 del Comando]. . .

Figura 1. 10. Formato de comandos en el IRobot Create®.

Todo código de operación o dato del comando tienen un tamaño de un byte y se emplea el espacio como separador, el formato de los datos del comando es decimal.

Se poseen los siguientes tipos de comandos:

a) Comandos de Inicialización: Se arranca y se deja lista para su uso a la Interfaz Abierta.

b) Comandos de Modo: Se selecciona el modo de operación del ICR. El robot posee cuatro modos de funcionamiento que limitan la acción de control sobre éste, y a su vez, determinan cuales comandos pueden ser ejecutados.

c) Modo Apagado: Es el estado que se presenta cuando al robot se lo enciende por primera vez o después de una carga de batería.

d) Modo Pasivo: este estado se caracteriza porque el robot permite la lectura de sus sensores pero no el accionar sobre alguno de sus actuadores. En dicho estado se encuentra el robot cuando se envía el Comando de Inicio.

e) Modo Seguro: este modo admite el control sobre los actuadores y lectura de sus sensores, con la excepción que, no identifica si una de las ruedas se atasca o si el existe un abismo mientras el robot se dirige hacia adelante. Tampoco admite el control del robot mientras se carga su batería.

f) Modo Completo: es el modo en el que se tiene el control absoluto sobre el robot.

g) Comandos de Entradas: Los comandos de entrada son utilizados para la lectura de los sensores ensamblados, de las entradas digital y analógica, y del estado de algunas variables internas del ICR. Existen 43 paquetes de datos que entregan información sobre los sensores o grupo de sensores cuyos valores son actualizados cada 15 ms.

h) Comandos de Actuadores: Los comandos de actuadores permiten realizar la acción de control sobre los motores DC, LEDs, parlante, salidas digitales y salidas utilizadas para controlar motores externos.

i) Comandos de Script: La Interfaz Abierta brinda la opción de crear comportamientos parecidos a los que se pueden visualizar en la ejecución de los demos mediante la creación de scripts. Los scripts pueden contener varios comandos y tienen un tamaño máximo de 100 bytes.

j) Comandos de Espera: Los comandos de espera son utilizados para generar un estado de espera de tiempo, distancia recorrida, ángulo de rotación o evento, durante el cual, la plataforma del ICR no detecta ni reacciona ante cambios en sus entradas. Estos comandos son útiles únicamente dentro de un script.

1.6. DESCRIPCION GENERAL DEL SENSOR KINECT

Es un dispositivo desarrollado por la empresa PrimeSense®, que luego se convirtió en un complemento para la consola de juego Xbox 360 de Microsoft [7], ver figura 1.11. Este permite el control y una interacción con la consola por parte del usuario sin necesidad de otros mecanismos, el cuerpo es el control, mediante una interfaz que reconoce gestos, comandos de voz y objetos.

Figura 1. 11. Dispositivo comercial Kinect [8].

Dispone de una cámara de video RGB (*red-green-blue*, rojo-verde-azul), un sensor de profundidad, compuesto por un emisor IR (infrarrojo) y un receptor IR, que con la combinación de información recibida mediante estos sensores es capaz de generar una imagen 3D de lo percibido, ver figura 1.12. Las condiciones en las que se realizó esta secuencia de imágenes son las siguientes: la base del Kinect se encontraba a un metro de altura (respecto al piso), la silla posee una altura de 0.7m, un área de aproximadamente 0.09 m^2 y estaba separada del Kinect por 2m, el espacio que deja la puerta abierta tiene un área de 1.8 m^2 y por último, la pared de fondo tiene una separación respecto al sensor de 6 m.

Figura 1. 12. Imagen RGB y secuencia de Imágenes RGBXYZ.

1.6.1. ESPECIFICACIONES TÉCNICAS

El sensor Kinect es una barra horizontal de aproximadamente 0.23m conectada a una pequeña base circular con un eje de articulación de rótula [8]. Se compone principalmente de:

a) Una cámara RGB (Resolución 640x480 VGA -*Video Graphics Adapter*- a 32 bits de color -RGB- @ 30fps), ver figura 1.13.

b) Un sensor de profundidad compuesto por un emisor de infrarrojos (proyector), un sensor CMOS monocromático de infrarrojos (resolución de 320x240 QVGA -*Quarter Video Graphics Array*- a 16 bits -65,536 niveles- de profundidad @ 30fps), ver figura 1.13.

c) Un sistema de micrófonos multi-arreglo (4 unidades) que opera con cada canal de audio a 16 bits con una frecuencia de muestreo de 16 Khz.

d) Un eje motor capaz de inclinar el sistema 27° hacia arriba o hacia abajo.

El sistema cuenta con un campo angular de visión de 57° en horizontal y 43° en vertical. Además, posee un sistema de cancelación de eco y reconocimiento de múltiples voces.

Figura 1. 13. Ubicación del proyector, del sensor IR y de la cámara RGB [13].

1.6.2. FUNCIONAMIENTO

La unión de los sistemas ópticos permite generar una imagen tridimensional, ver figura 1.13, de lo que tiene delante. Para ello, se calcula el tiempo que tarde, una vez que sale la luz IR, en rebotar y volver la luz IR, gracias a esto se puede saber la distancia.

Para conocer la distancia a la que se encuentra cada píxel de la imagen de profundidad se emite una constelación de puntos con el emisor IR. Entonces, la cámara infrarroja detecta esta constelación y se calcula la disparidad para cada píxel (la diferencia entre donde estaba el punto al proyectarlo a donde está en la proyección). Esencialmente, todos los píxeles que se reciben como IR son convertidos

en una escala de colores, haciendo que los cuerpos, dependiendo de la distancia, se capturen como rojos, verdes, azules hasta llegar a tonos grises, que representan a objetos muy lejanos, a esto se le llama cámara de luz estructurada [8], ver figura 1.14. Por lo que, se puede tener precisión en la detección de profundidad sin depender tanto de la luz ambiental. Las condiciones en las que se realizó esta captura son similares a la de la figura 1.12, pero adicional a la silla, hay una persona de 1.6 m de altura ubicada en la puerta con los brazos extendidos.

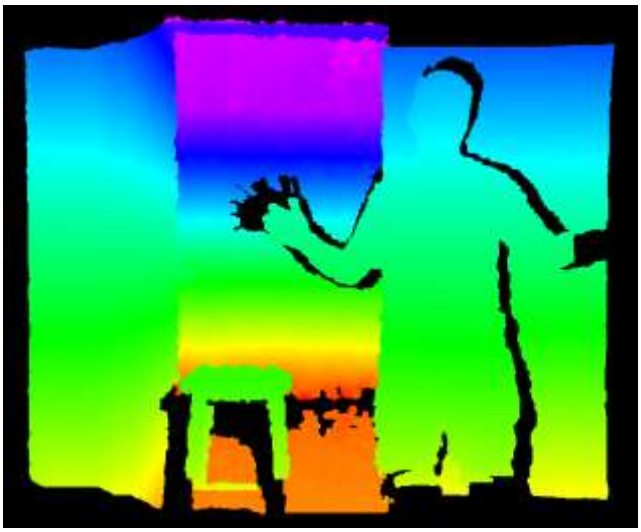

Figura 1. 14. Imagen de profundidad en escala de colores.

Por otra parte, debido a que la cámara IR y la cámara RGB están desplazadas en el eje X, es importante establecer el sistema de referencia por el que se van a regir las mediciones. Para esto, se toman los valores de las matrices de traslación y rotación genéricos aportados por ROS en su paquete de Openni Kinect, ver sección 1.7 y 2.1.2.

Una aplicación, y la que usa el Kinect en compatibilidad con la consola Xbox, es el programa que toma las imágenes 3D y las hace pasar por una serie de filtros para determinar qué es una persona y que no lo es. Para esto se siguen una serie de parámetros, en los cuales se define la estructura del cuerpo humano, para impedir que los obstáculos presentes en la escena sean reconocidos como otros jugadores.

Por otra parte, Microsoft ha hecho público el kit de desarrollo para programar bajo la plataforma de Kinect (el SDK -*Software Development Kit*- por sus siglas en ingles), con lo que cualquiera puede utilizar el kit de desarrollo para programar su propia aplicación para utilizar el dispositivo Kinect. Con el SDK oficial se pueden realizar dos seguimientos de esqueletos simultáneos y se pueden llevar a cabo reconocimientos de voz [6]. Por otra parte, con la utilización de software libre, se pueden realizar reconocimientos de caras y de dedos.

El campo de acción del sensor está limitado entre 1,2 y 3,5 metros para la aplicación de reconocimiento y seguimiento de personas, pero es capaz de pasar estos límites dependiendo del uso que se le vaya a dar, pudiendo estar en rangos de entre 0,7 y 6 metros aproximadamente.

1.7. ROBOT OPERATING SYSTEM

Desarrollado en 2007 por el Laboratorio de Inteligencia Artificial de Stanford (*Stanford Artificial Intelligence Laboratory*), de su traducción literal del inglés resultaría en "sistema operativo de robots", aunque no es propiamente un sistema operativo (OS -*operating system*- por sus siglas en inglés) desde el punto de vista de una definición tradicional, un OS es “un programa o conjunto de programas encargados de gestionar procesos, procedimientos y recursos para compartir una instancia de computación” [9]. En cambio, ROS se focaliza en proveer una estructura de comunicación y de servicios para programas y procesos usando como base un sistema operativo anfitrión [10].

Actualmente la escritura de software para robots resulta particularmente difícil, debido al constante cambio y crecimiento que se presentan en estos tópicos, haciendo que cada vez sea más compleja la tarea de generación de código.

En este sentido, la creación y desarrollo de *Frameworks* o RDEs (*Robotic Development Environments* – Ambientes de Desarrollo para Robótica) son una opción útil al momento de manejar la complejidad de los objetos de estudio. Sin embargo, esto resulta en una gran variedad software que generalmente atienden a necesidades específicas [10], y son más robustos para casos particulares ya que fueron concebidos para eso. De esta manera, ROS busca integrar a gran escala una gran diversidad de sistemas robóticos.

Así es que, ROS provee librerías y herramientas para ayudar a los desarrolladores de software en la creación de aplicaciones de robots. Está provisto de abstracción de hardware, controladores de diversos dispositivos, visualizadores, pase de mensajes, manejo de paquetes, entre otras características. Uno de sus principales rasgos distintivos es el hecho que es completamente "*open source*", es código abierto bajo el estilo de licencia BSD, es libre de usarse, cambiarse y comercializarse. El objetivo principal de ROS es permitir, o facilitar propiamente, a los desarrolladores el diseño, construcción y generación de robots cada vez más capaces, consiguiendo aplicaciones de forma rápida y sencilla.

1.7.1. CARACTERISTICAS DE DISEÑO

"Los objetivos filosóficos de ROS [10], pueden ser resumidos en:"

a) Red punto a punto: Un sistema basado en ROS consiste en un número procesos, con posibilidades de diferentes anfitriones conectados en tiempo real en una topología de red punto a punto.

b) Multilenguaje: ROS ha sido diseñado para soportar diferentes lenguajes de programación (C++, Python, Octave y LISP principalmente). Para poder realizar esta tarea, se emplea un lenguaje neutral y se hace uso de una interfaz de definición de lenguaje (IDL - *Interface Definition Language*) para

establecer los mensajes que serán empleados para la comunicación entre módulos, ver figura 1.15. De esta manera, a partir del lenguaje neutral se produce una generación de código, de forma automática, en los otros lenguajes y de esta manera poder enviar y recibir mensajes, información, entre módulos escritos en el mismo estilo de lenguaje como los que no lo están.

Header header
Float32 A
Float32 B

Figura 1. 15. Estructura estándar de mensaje de ROS.

c) Herramientas: Para mejorar el manejo de la complejidad de ROS se cuenta con una gran variedad de mini herramientas, utilizadas para generar los diversos componentes de ROS. Estas ejecutan varias tareas como: configuración de parámetros, visualización y monitoreo de la conexión punto a punto, graficación de la información de los mensajes, entre otras. La separación de las tareas en módulos puede acarrear en una pérdida de la eficiencia pero se gana un manejo más estable.

d) Ligero: Los drivers o algoritmos generados para la operatividad de los múltiples robots existentes podrían ser reutilizados o reciclados fuera del proyecto en el que se encuentran. En este sentido, para asegurar esta filosofía el sistema de construcción de ROS se produce en el código fuente y se emplea CMake, para lograr que la complejidad caiga en las librerías y que se creen ejecutables asociados a esas librerías funcionales de ROS.

e) Software libre: El código fuente de ROS es público y se encuentra disponible. Es distribuido bajo la licencia BSD lo que permite el desarrollado de proyectos comerciales y no. Por otra parte, el uso de ínter procesos de comunicación y el no requerir que los módulos enlazados estén contenidos en el mismo

ejecutable, hace que se pueden realizar pases de información entre módulos. De esta manera, todos los sistemas construidos alrededor de ROS pueden hacer uso de varios tipos de licencia, como GLP (*General Public License*) y BSD.

1.7.2. FUNDAMENTOS

Las bases operacionales en las que se encuentra implementado ROS son: gerencia de nodos, mensajes, tópicos y relaciones cliente-servidor.

a) Nodos: La premisa en la que es desarrollado ROS es en que sea modular. En este sentido, el término nodo se encuentra referenciado a un módulo de software, un programa. Las relaciones que se establecen entre los nodos pueden ser visualizados con la ayuda de la herramienta gráfica RXGRAPH, en ella los nodos aparecen encerrados en círculos, ver figura 1.16.

b) Mensajes: La comunicación entre los nodos se realiza por medio de mensajes. Estos son un tipo estructurado de datos, en varios formatos, como: enteros, punto flotante, booleano. Por otra parte, un mensaje puede estar definido por otro mensaje o conjunto de mensajes. La conexión entre los nodos en la herramienta RXGRAPH aparece representa por una flecha, y su sentido indica que nodo está enviando el mensaje y que nodo lo está recibiendo, ver figura 1.16.

c) Tópico: La comunicación de un nodo puede ser de escuchar o de hablar, y esto es posible por medio de los tópicos, en que los mensajes son publicados para que cualquier nodo pueda extraer la información por medio de una suscripción, en la herramienta RXGRAPH los tópicos son colocados encima de las flechas que representa la conexión entre nodos, ver figura 1.16.

d) Cliente – Servidor: Es un sistema de comunicación definida por una llamada a servicio, en la que se realiza una petición (cliente) y se da una respuesta (servidor). En este modelo también se encuentra definido un mensaje.

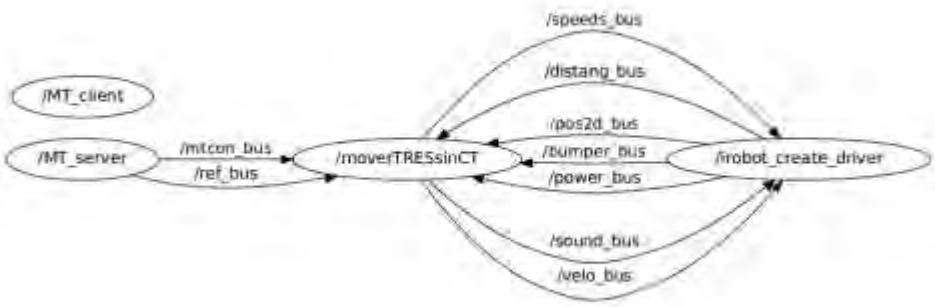

Figura 1. 16. RXGRAPH de un sistema en ROS.

1.7.3. ORGANIZACIÓN

De modo de proveer y promover una plataforma para el desarrollo colaborativo, el sistema organizativo de ROS está basado en "paquetes", en la que su definición resultaría como: un directorio que contiene un archivo XML con la descripción de su contenido y dependencias. Además de, la habilitación de mensajes, nodos, servicios y clientes.

Para finalizar, si se poseen dos o más paquetes se pueden forma una pila (*Stack*), de características similares a los paquetes pero en un orden jerárquico superior.

1.8. POINT CLOUD LIBRARY – PCL

Está enmarcado como un proyecto para el procesamiento de nubes de puntos, en 3D, y se encuentra bajo la licencia BSD, lo que la hace de libre uso para comercializar y realizar investigaciones [11].

Entre los diversos contenidos que ofrece la librería se seleccionaron cuatro, para realizar procesamiento de la data del Kinect. Estas son:

a) Point Type: Permite definir nuevos formatos o utilizar formatos existentes para la manipulación de la información. De esta manera, se pueden llegar a tener estructuras de datos con toda la información de la nube de puntos o solo aquella que interesa, como por ejemplo: Point-XYZ y Point-XYZRGB, el primero ofrece la posición (X, Y, Z) de cada punto en la nube, mientras que el segundo hace eso mas otorgar el color de cada punto.

b) Voxel-Grid: Es un proceso de filtrado de la nube de puntos. En este, se definen redes o pequeñas cajas 3D de igual tamaño en toda la nube de puntos, para luego hacer una aproximación de la imagen con el centroide de cada caja, así se tiene una data menos densa. Este proceso otorga una imagen más precisa que si se usara únicamente el centro de cada caja.

c) Passthrough Filter: Es un proceso de filtrado de dimensiones. Dada una nube de puntos, se especifica la dimensión que se quiere observar y el rango de su existencia. De esta manera, se puede tener acceso a una data en una región específica de la nube de puntos.

d) Visualizador: A partir de una nube de puntos se puede realizar una visualización de sus componentes.

1.9. REGRESIÓN LINEAL

Dada una secuencia de puntos (X, Y), con dos o más elementos, se procede a calcular la recta de la forma [12]:

$$Y = mX + b \tag{1.1}$$

En este sentido, los parámetros a determinar son la pendiente (m) y la intercepción con el eje de las ordenadas (b). Es Así que:

$$X' = \sum_{i=0}^{n} \frac{Xi}{i}$$

$$Y' = \sum_{i=0}^{n} \frac{Yi}{i}$$

$$Sxx = \sum_{i=0}^{n} (Xi - X')^2$$

$$Syy = \sum_{i=0}^{n} (Yi - Y')^2$$

$$Sxy = \sum_{i=0}^{n} (Xi - X')(Yi - Y')$$

$$m = \frac{Sxy}{Sxx} \tag{1.2}$$

$$b = Y' - mX' \tag{1.3}$$

En caso que la pendiente sea infinita, se usara la siguiente expresión para evaluar la recta:

$$X = b \tag{1.4}$$

1.10. RECTA PARALELA A DISTANCIA FIJA

Para determinar una recta paralela a otra, donde la distancia entre ellas, siempre sea la misma (SEP) sin importar la pendiente, se hace uso de la siguiente:

$$Yp = mX + bp \tag{1.5}$$

$$bp = b + d \tag{1.6}$$

$$d = SEP \sqrt{\frac{1}{(1-((\frac{1}{m+\frac{1}{m}})^2)(1+m^2))}} \tag{1.7}$$

1.11. ANTECEDENTES DE ROBOTS CON RUEDAS BASADOS EN ICR

Los robots Bilibot y TurtleBot son plataformas de ruedas diferenciales que se integran con un sensor Kinect y ROS como software.

Por una parte, el Bilibot es un proyecto de software libre que usa como plataforma móvil al ICR, como sensor al Kinect, como software ROS y además posee un brazo robótico. Este puede realizar seguimiento de personas y manipulación de objetos. El Bilibot es desarrollado por Garrat Gallagher en el MIT (del inglés *Massachusetts Institute of Technology*) [13], ver figura 1.17.

El TurtleBot, es un proyecto similar desarrollado por Willow Garage, que entre sus atributos cuenta con: teleoperación por teclado o joystick, seguimiento de personas por medio del reconocimiento que realiza el Kinect, navegación y SLAM (*Simultaneous Localization And Mapping* – Localización y Mapeo Simultáneos), ver figura 1.17.

Figura 1. 17. Bilibot a la izquierda [13] y TurtleBot a la derecha [14].

Por otra parte, sistemas desarrollados en ROS con aplicaciones directas en el IC se encuentran en la Universidad Brown con el paquete "Brown Ros Package" [15] y la Universidad de Colorado con el paquete "prairiedog-ros-pkg" que es una versión modificada y actualizada del anterior. Este último lo hace compatible con ROS V1.0 [14], y en ambos la funcionalidad radica en el establecimiento de la comunicación serial entre una computadora y el móvil.

En el Grupo de Mecatrónica de la Universidad Simón Bolívar se ha llegado a desarrollar el control de ángulo de un robot móvil desplazándose a velocidad constante [16], que sirvió de puente para desarrollar el control del módulo de la velocidad para un sistema de control por campos de velocidad [17]. Por otra parte, se realizo control por vectores de referencia de velocidad en módulo y ángulo de forma independiente, como entrada de un modelo cinemático de un vehículo diferencial para generación de campos de velocidad para navegación coordinada en un entorno de simulación [18]. De esto, se tomará el control de vectores de referencia para ejecutarlo en un robot físico y así dar continuidad al proceso de desarrollo.

CAPÍTULO 2

METODOLOGÍA

2.1. DISEÑO EXPERIMENTAL

En concordancia a lo planteado en la introducción del presente trabajo, para el cumplimiento del objetivo general y de los específicos, se empleará ROS como base del sistema integrado Roomba-Kinect.

En primer lugar, se desarrollará una plataforma de control para el robot Roomba utilizando referencias vectoriales de velocidad, ya establecidos los antecedentes, las características y limitaciones del móvil (secciones 1.1. – 1.5. Y 1.8).

En segundo lugar, se realizará la adquisición de datos del Kinect, con lo que se tendrá acceso a la información de manera local o remota y en estado natural o con algún tipo de procesamiento. Respecto a lo último, se tendrán en cuenta dos grupos o tipos de datos, el primero enfocado para propósitos generales, comprobar el servicio de datos, y el segundo fue añadido para realizar navegación en interiores, específicamente en un pasillo, de condiciones establecidas explicadas a continuación, por medio de la creación de rectas de aproximación a la nube de puntos y de la obtención de vectores de referencia, a partir de las rectas generadas. De todo esto se puede dar una evaluación general del desempeño del sistema, ver figura 2.1.

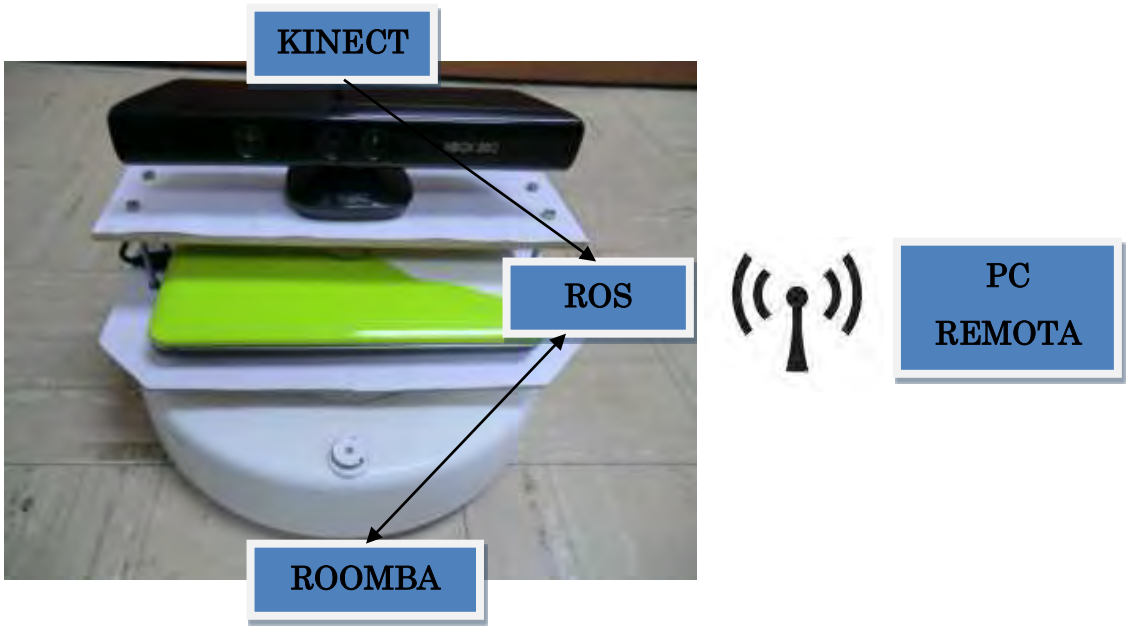

Figura 2. 1. Sistema integrado Roomba-Kinect.

Se establecen las siguientes etapas, ver figura 2.2, o módulos principales en los que se ha dividido el trabajo:

a) Control del ICR.

b) Integración del Kinect: servicio de datos de propósitos generales y para navegación.

c) Exportación de información.

Por otra parte, en un esquema clásico de control los parámetros de referencia, controlador y realimentación quedan definidos como:

a) La referencia: Se toman como referencia vectores de velocidad, en módulo y ángulo. Esta es la entrada principal del módulo de control del ICR.

b) El controlador: Se emplea control en dos niveles, el de orden superior se encuentra encargado de la navegación por el seguimiento de vectores de velocidad y el inferior de que la velocidad del móvil sea la adecuada.

c) La realimentación: Consiste en la generación de los vectores de velocidad. Estos, pueden ser creados a partir de los servicios de datos del Kinect, por cálculos de odometría del ICR y por la lectura instantánea de los encoders ubicados en las ruedas del ICR.

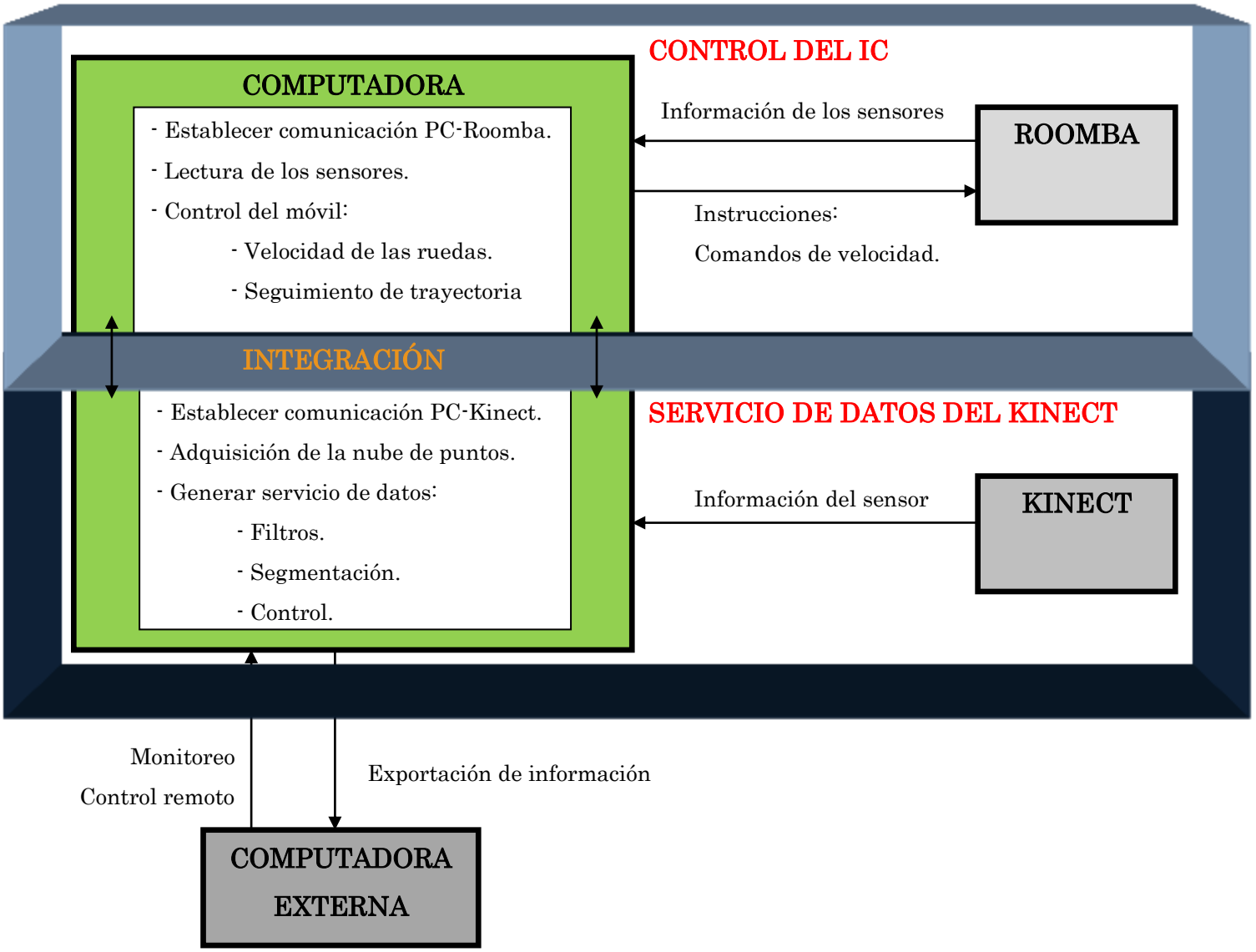

Figura 2. 2. Esquema de conexión de módulos.

Al robot se le han definido dos espacios de trabajo, que comparten la similitud de ser superficies rectangulares. El primero de ellos es un área libre de obstáculos de 2,5m de ancho por 3,5m de largo, ver figura 2.3. El segundo, es un pasillo de paredes no uniformes, con una separación entre ellas de aproximadamente 2,8m, ver figura 2.4.

Figura 2. 3. Primer espacio de trabajo, área libre de obstáculos.

Figura 2. 4. Segundo espacio de trabajo, pasillo.

2.1.1. CONTROL DEL ICR

2.1.1.1 CINEMÁTICA

A continuación se presenta la cinemática necesaria para establecer la relación entre las señales de referencia (V, φ), vector de velocidad expresado en módulo y ángulo, y las señales de control de ambas ruedas (V_{der}, V_{izq}), velocidad rueda derecha y velocidad rueda izquierda.

Considerando al robot como un cuerpo rígido, la velocidad lineal del centro de masa se obtiene a partir del promedio de las velocidades lineales de cada una de sus ruedas [19]. La velocidad lineal de cada rueda se obtiene como el producto de la velocidad angular (velocidad de giro θ_{der} y θ_{izq}) y el radio de ellas (r), ecuación 2.1. La velocidad del centro de masa queda definida como, ecuación 2.2:

$$V = \frac{r(\theta der + \theta izq)}{2} \tag{2.1}$$

$$V = \frac{Vder + Vizq}{2} \tag{2.2}$$

El ángulo de giro del robot se determina a partir de las relaciones geométricas entre el movimiento de cada una de las ruedas del robot, ve figura 2.5.

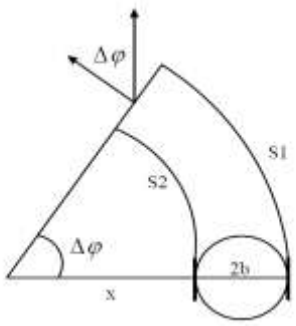

Figura 2. 5. Relaciones geométricas para el ángulo de giro [19].

De esta manera, el ángulo de giro del robot es igual al arco sostenido por la trayectoria. Dado que por definición el ángulo de dirección del robot, φ, aumenta en sentido opuesto a las manecillas del reloj, y considerando que la rueda derecha gira a una mayor velocidad que la izquierda, el ángulo de dirección debe aumentar Δφ, según la figura 2.5, la rueda izquierda sostiene un arco de radio x, por lo que la distancia recorrida por esa rueda está dada por:

$$S_2 = r\Delta\theta_{izq} = x\Delta\varphi \tag{2.3}$$

La rueda derecha se encuentra más lejos del centro de la circunferencia que determina la trayectoria, y recorre una distancia mayor en el mismo tiempo, se le añade el segmento correspondiente al diámetro del móvil (2b), y viene dada por:

$$S_1 = r\Delta\theta_{der} = (x + 2b)\Delta\varphi \tag{2.4}$$

Calculando el valor de la diferencia S1 – S2 y dividiendo por el tiempo transcurrido Δt, se obtiene la relación entre la velocidad de giro del robot, y la velocidad de cada una de sus ruedas, como se indica a continuación:

$$\varphi' = \lim_{\Delta t \to 0}\left(\frac{\Delta\varphi}{\Delta t}\right) = \frac{(Vder - Vizq)}{2b} \tag{2.5}$$

De lo anterior, se puede establecer una relación entre las velocidades de las ruedas (V_{der}, V_{izq}) y la señal de referencia (V, φ'), como se muestra a continuación:

$$V_{der} = V + b\varphi' \qquad y \qquad V_{izq} = V - b\varphi' \tag{2.6}$$

Dado que se requiere una relación con una señal de referencia de la forma (V, φ) se plantea la siguiente expresión:

$$V_{der} = V + K\varphi \qquad y \qquad V_{izq} = V - K\varphi \tag{2.7}$$

Donde K es un constante de ajuste, que permite representar la señal de control de ambas ruedas por medio del módulo y el ángulo de un vector de velocidad, fue sintonizada de forma empírica.

2.1.1.2. SINTONIZACIÓN DE CONTROLADORES

Los lazos de control se definen utilizando como variables manipuladas el módulo (V) y el ángulo (φ) de un vector de velocidad. Estas componentes fueron elegidas debido a que son características que definen a cualquier vector en el plano, por lo que en trabajos fututos se puede remplazar la velocidad, como cantidad física medida, por la aceleración u otra.

Ambas referencias se siguen con lazos de control, ver figura 2.6, sintonizados empíricamente por Ziegler-Nichols [20]. Luego el par (V, φ) es introducido en la ecuación (2.7) para obtener las ordenes de cada motor del IC.

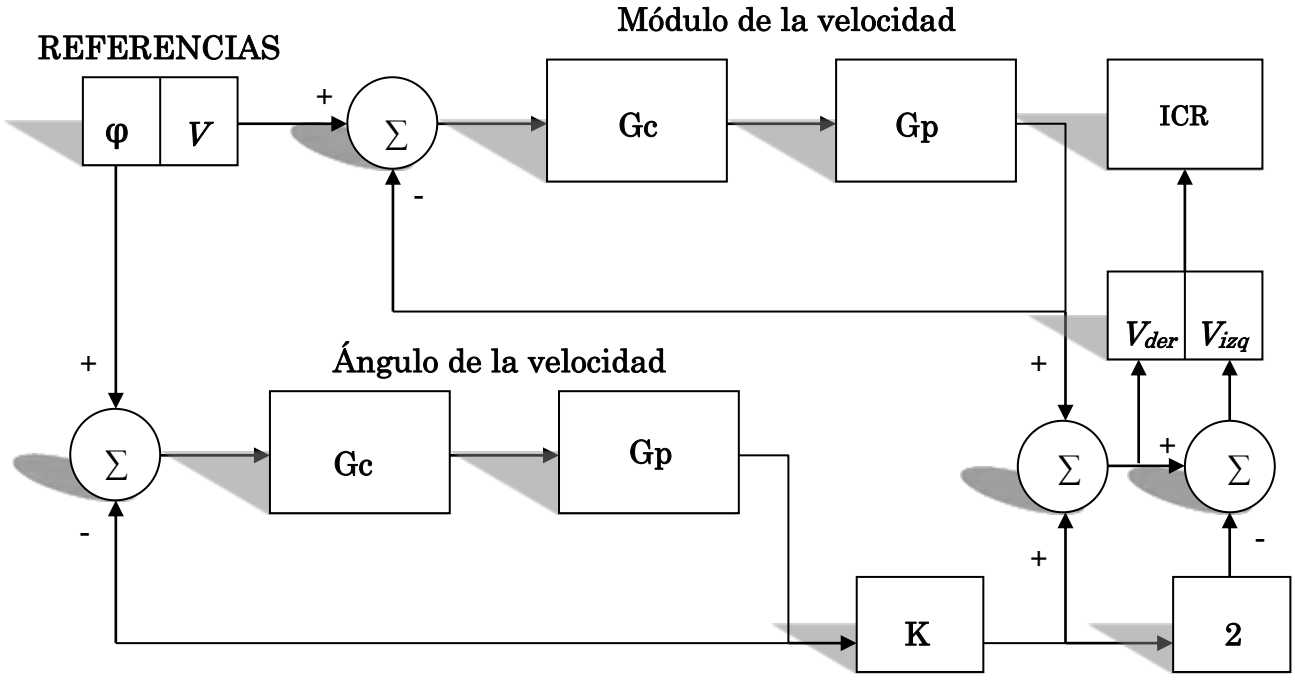

Figura 2. 6. Esquema de Control.

El primer método, o método de lazo abierto – curva de reacción, consiste en que la respuesta a una entrada escalón unitario se obtiene de manera experimental. Si la planta no contiene integradores, ni polos complejos conjugados, la curva de respuesta a un escalón puede tener forma de S, como la mostrada en la gráfica siguiente, figura 2.7.

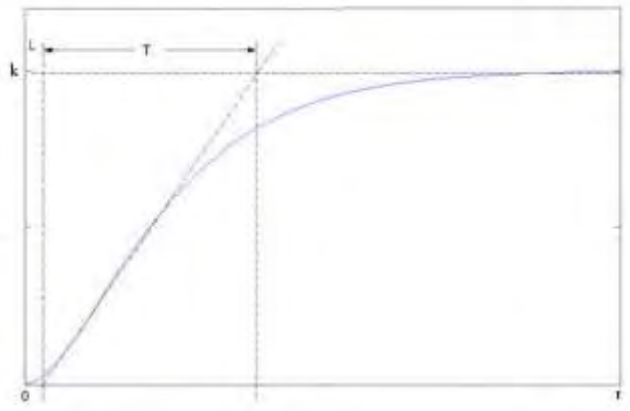

Figura 2. 7. Primer método de sintonización Ziegler-Nichols.

Se expresan como parámetros L (retardo) y T (constante de tiempo) y se debe cumplir que:

$$0.1 < \frac{L}{T} < 1 \tag{2.8}$$

Por otra parte T y L también se pueden expresar como:

$$T = 1{,}5\ (T_{0,63} - T_{0,28}) \tag{2.9}$$

$$L = 1{,}5\ (T_{0,28} - \frac{1}{3}T_{0,63}) \tag{2.10}$$

En la tabla 2.1 se muestran los valores de ganancia proporcional, integrales y derivativos para cada tipo de controlador.

Tabla 2. 1. Valores de ganancias para el primer método de sintonización Ziegler-Nichols [20].

Control	Kp	Ti	Td
P	T/L	infinito	0
PI	0,9T/L	L/0,3	0
PID	1,2T/L	2L	0,5L

El segundo método, o método a lazo cerrado – oscilaciones amortiguadas, consiste en que el sistema presenta una respuesta sub-amortiguada, por lo que se establece un modelo de control proporcional, y se realizan pruebas de ensayo y error, para determinar la ganancia critica (Kcr), y el periodo crítico (Pcr), que hace que el sistema se comporte como oscilatorio puro, ver figura 2.8. Y así, determinar los parámetros de control.

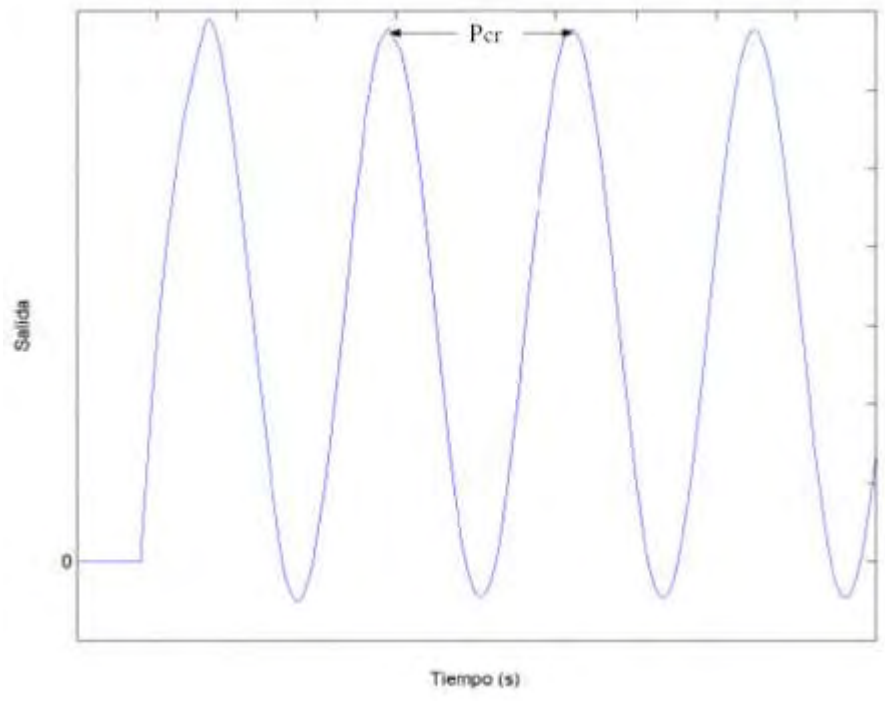

Figura 2. 8. Segundo método de sintonización Ziegler-Nichols.

Se expresan como parámetros la ganancia crítica Kcr y el periodo crítico Pcr, para establecer los parámetros de los controladores, ver tabla 2.2.

Tabla 2. 2. Valores de ganancias para el segundo método de sintonización Ziegler-Nichols [20].

Control	Kp	Ti	Td
P	0,5Kcr	infinito	0
PI	0,45Kcr	(1/1,2)Pcr	0
PID	0,6Kcr	0,5Pcr	0,125Pcr

Una vez logrado el seguimiento independiente de las referencias, se presenta el problema de lograr el módulo adecuado cuando aun no se ha logrado el ángulo, generando movimiento en la dirección equivocada. Esto puede resultar un grave problema en enfoques de campo debido que la mayor información que el mismo entrega es la dirección de movimiento, llegando algunas aplicaciones incluso a prescindir del módulo del mismo. Para evitar este problema se escala la referencia de módulo de velocidad con el coseno del error en ángulo. De esta forma la plataforma avanza con la proyección del vector velocidad sobre su dirección actual, aprovechando el retroceso y en algún nivel el avance durante el giro, y evitando a la vez avanzar en direcciones inapropiadas [18].

2.1.1.3. DETERMINACIÓN DE LA POSICIÓN

Para conocer el la posición (x, y, φ), donde se encuentra el ICR, primero hay que establecer un sistema de referencia. Este se creará como la posición inicial en la que se encuentre el ICR al arrancar, ver figura 2.9, y las demás mediciones que se realicen quedan sujetas a este primer punto.

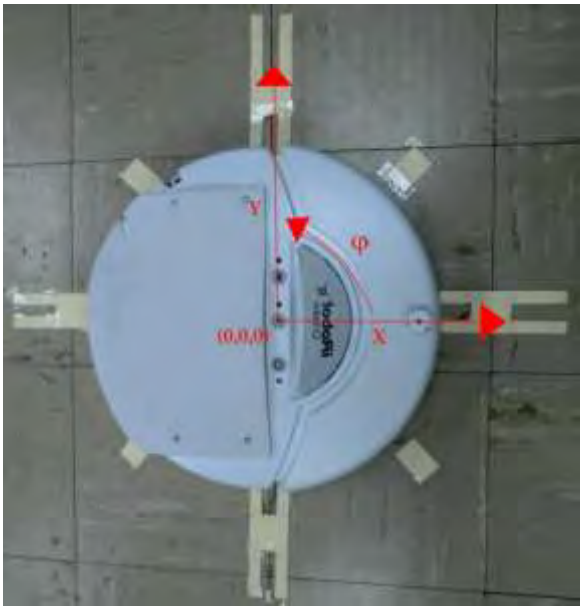

Figura 2. 9. Sistema de referencia del ICR.

Una vez que se posee el sistema de referencia, se pueden realizar los cálculos de odometría correspondientes para estimar los valores (x, y. φ), que permiten conocer la posición y orientación del ICR.

2.1.1.4. SEGUIMINENTO DE UNA TRAYECTORIA CONOCIDA

A partir del control por vectores de referencia se puede hacer seguimiento de una trayectoria ya planeada, por medio del cálculo de vectores de aproximación (v_{aprox}), desde el móvil al punto más cercano (P_{prox}) a la curva establecida. Para luego, determinar un vector tangente (v_{tan}) a ese punto [21]. De esta manera, al sumar v_{aprox} y v_{tan} se obtiene un vector de referencia (v_{ref}), que se puede descomponer en módulo y ángulo, para ser luego sometido a control, ver figura 2.10. Cabe destacar que el valor del módulo es directamente proporcional a la distancia, por lo que se escala de manera que este entre los límites permitidos de velocidades aceptadas por el móvil (sección 1.5). Se establece de manera arbitraria que para valores mayores a 2m de distancia la velocidad es de 0.11 veces el valor calculado, para menores es de 0.08 veces el valor, y para aquellos que pases los 4m se estable

una velocidad fija de 0.45 m/s, la década con que se atenúa en los 3 casos es para establecer los rangos de velocidades en cantidades inferiores a 0.5 m/s.

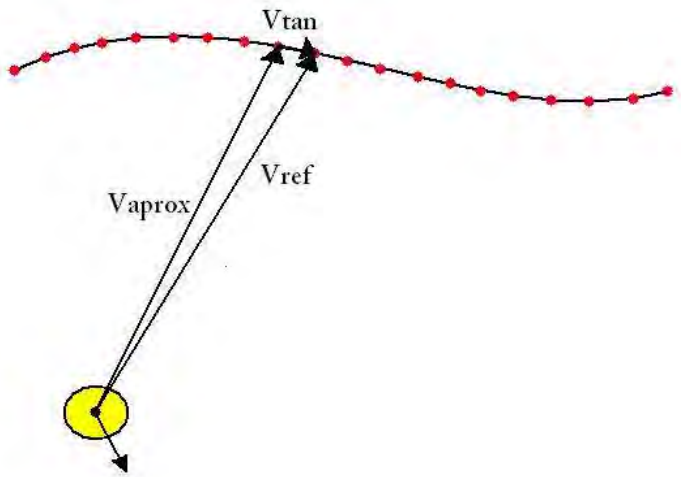

Figura 2. 10. Seguimiento de Trayectoria Planificada.

2.1.1.5. CONFIGURACIÓN DE ROS

En el presente trabajo se utilizó la versión DiamondBack-Desktop-Full de ROS, que funciona de manera estable en el sistema operativo Ubuntu 10.10 (Maverick Meerkat). Para esto, se siguieron las instrucciones planteadas en el sitio oficial de Ubuntu [22] y ROS [14].

Se hace uso del repositorio “Prairiedog-ros-pkg” de la Universidad de Colorado y específicamente del paquete Irobot_Create_Rustic, que es una versión modificada del paquete "Brown´s IRobot Create package" de la Universidad de Brown.

Ambos paquetes se caracterizan por la manipulación del ICR a partir de los comandos de interfaz abierta, la diferencia radica en que el primero establece los protocolos de comunicación por medio de sistemas de cliente-servidor, mientras que el segundo por la publicación y suscripción de mensajes, además de poseer un nodo “Driver” que se encarga de crear tópicos con la información de los sensores y para la

resección de instrucciones, además realiza los cálculos necesarios para la estimación de la posición del móvil.

Otra característica del nodo de la Universidad de Colorado, es el poder establecer una comunicación serial de 57600 Baud entre la PC y el ICR. Ahora, para el envío de las ordenes de velocidad, se le realizó una modificación, se le añadió la posibilidad de enviar las instrucciones de velocidad con el formato (V_{der},V_{izq}), que se corresponde a las necesidades del presente trabajo, inicialmente el comando usado era velocidad lineal y radio del arco de giro.

A partir de esto, se pueden desarrollar otros programas, nodos, que permitan mover al ICR. En este sentido se plantea poseer nodos dedicados a:

a) Mover el carro, en un lazo cerrado con un valor fijo de velocidad.

b) Mover el carro por medio del teclado de una computadora.

c) Enviar instrucciones de velocidad por sistema de cliente-servidor.

d) Realizar control, dado un vector de referencia de velocidad.

e) Calcular vectores de referencia a partir de una trayectoria planificada libre de obstáculos y con acceso a la posición del robot.

2.1.1.5.1 NODOS

A continuación se da una descripción de los nodos empleados en el ICR.

a) Driver: Su principal función es habilitar la comunicación serial entre la PC y el ICR, a 57600 Baud. En este sentido, posee estructuras de mensajes

diseñadas para que otros nodos puedan tener acceso a los sensores y poder recibir instrucciones que le permitan al ICR desplazarse y hasta hacer sonar una melodía. Cabe destacar, que este programa fue modificado para que tuviese habilitado el envío de velocidad por comandos de velocidad de rueda derecha y velocidad de rueda izquierda. Además, de realizar el cálculo de la velocidad instantánea medida, a partir de los datos de distancia y ángulo recorrido en un intervalo de tiempo, este se determino como el tiempo que había entre una medición y otra. Por otra parte, en el nodo driver se fija el sistema de referencia, ver figura 2.8, y en este sentido, se llevan las mediciones de la posición del robot de acuerdo a su movimiento. Este nodo se configura con el nombre de "irobot_create_driver".

b) Move_Teleop y Move_Teleop_Ref: Su característica común es el envío de comandos de velocidad por teclado. La diferencia radica en que move_teleop se comunica directamente con el driver, ver figura 2.11, mientras que move_teleop_ref crea un vector de referencia para ser controlado y luego enviado al driver, ver figura 2.12.

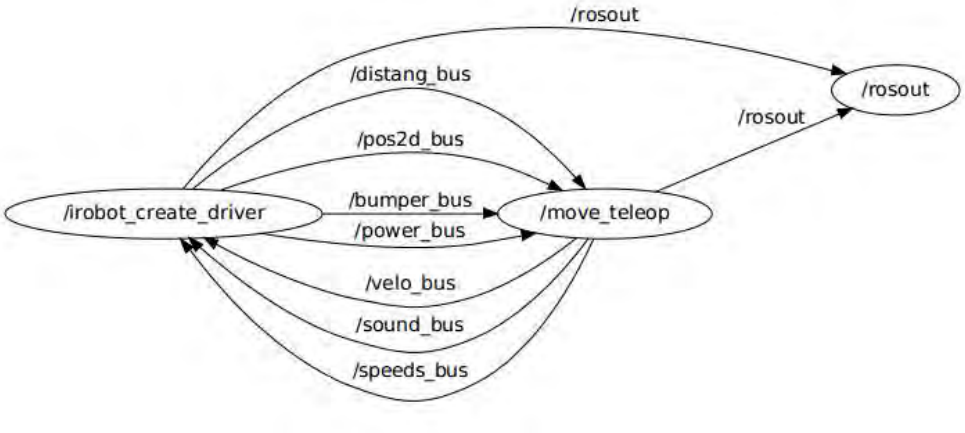

Figura 2. 11. Sistema Move_Teleop y Driver.

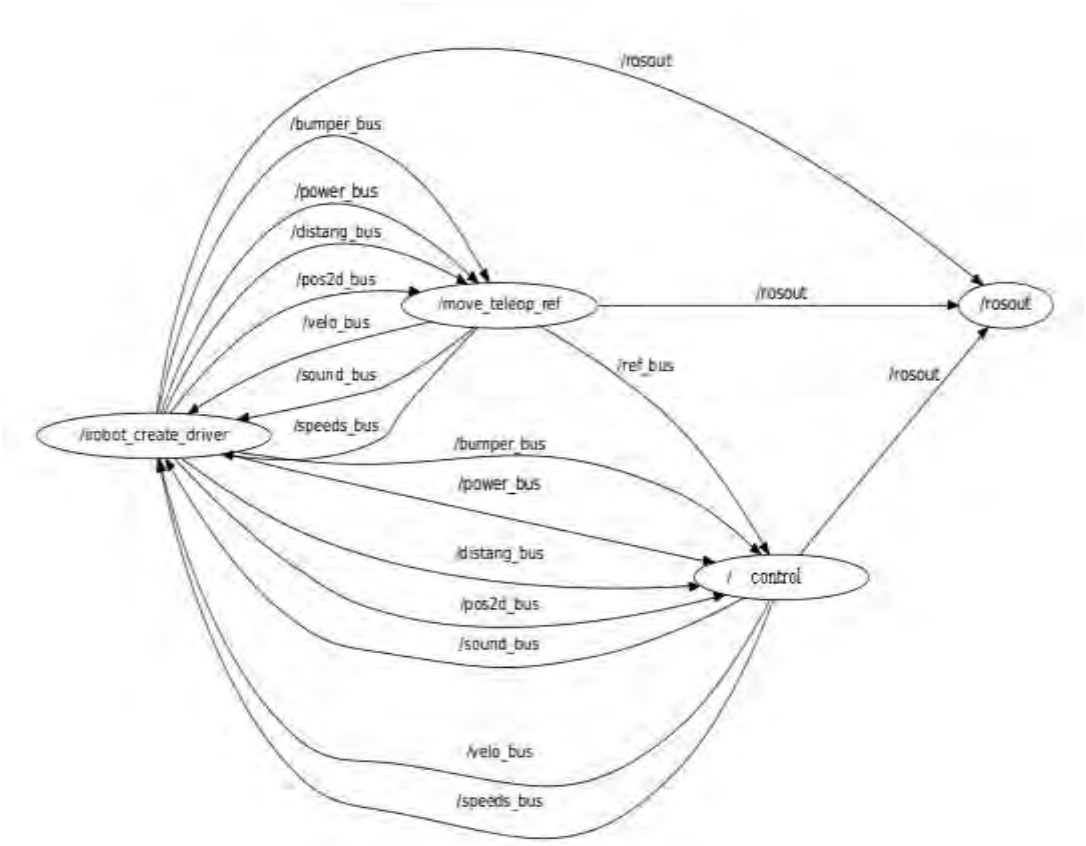

Figura 2. 12. Sistema Move_Teleop_Ref, Control y Driver.

c) MT_server, MT_client y Mover: Son una estructura de cliente-servidor que permite el envío de comandos de velocidad a un tercer nodo (Mover), que se encuentre suscrito al mensaje que publica el servicio, al igual que move_telop_ref el servicio crea un vector de referencia, módulo y ángulo de velocidad , ver figura 2.13. El nodo Mover se encarga de recibirlo para luego transformarlo en un comando tipo (V_{der},V_{izq}) para ser enviado al ICR. Se emplea con propósitos generales, uno de ellos es evaluar el comportamiento natural del sistema, al enviar comandos de velocidad en módulo y ángulo por teclado.

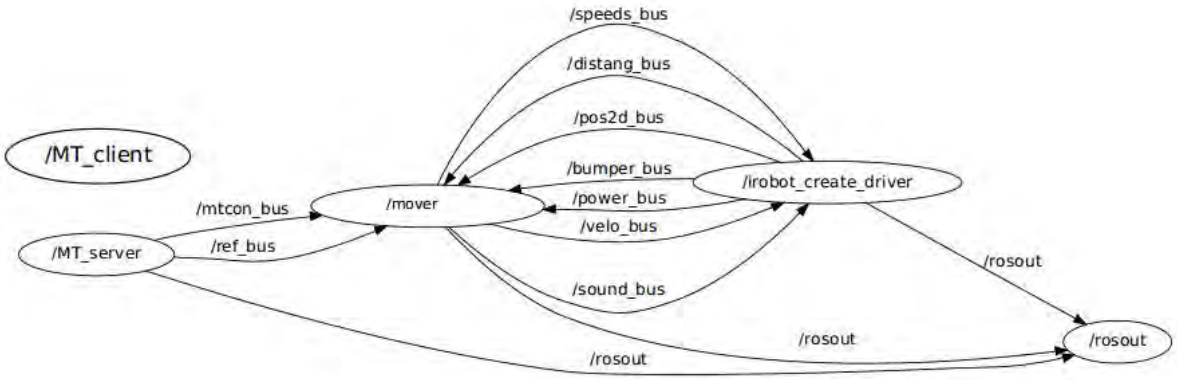

Figura 2. 13. Sistema Cliente-Servidor y nodo Mover.

d) Control: Esta encargado de recibir un mensaje de referencia en formato (V, φ), y realizar operaciones de control sobre él, para luego transformarlo en formato (V_{der},V_{izq}) y enviarlo al driver. Una característica importante de este programa, dado que el ángulo está definido de 0°-360°, es asegurar la existencia de continuidad entre 360°- 0° y viceversa. ver figura 2.12 y 2.14.

e) Mover_Ref: En el programa se encuentra fija una trayectoria y según la posición en la que se encuentre el móvil respecto a esta, se calcula un vector de referencia en el formato (V, φ), ver sección 2.1.1.4. y figura 2.14.

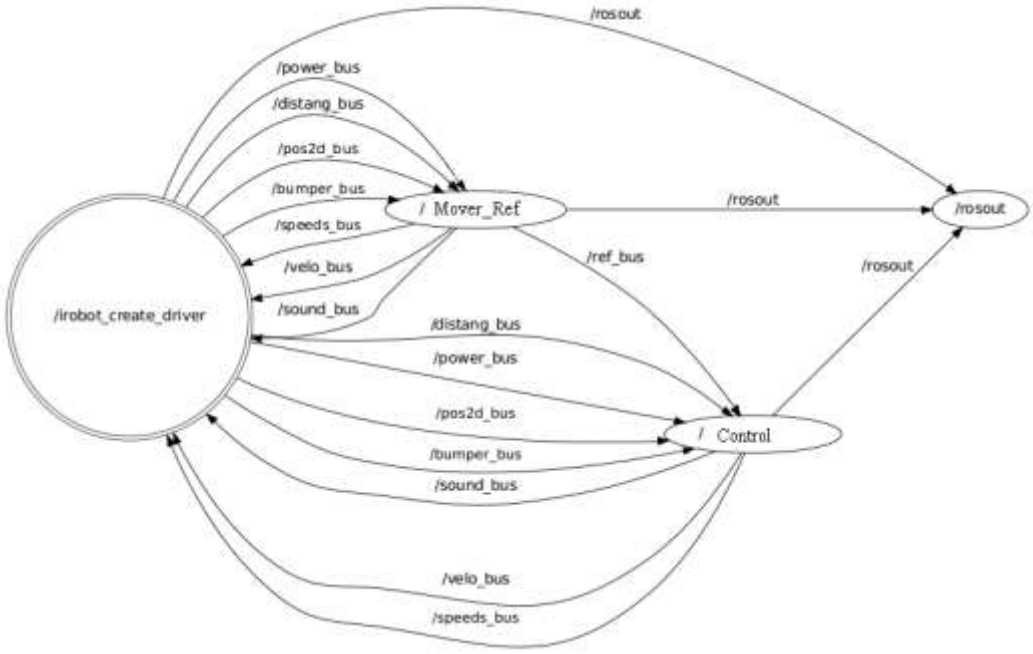

Figura 2. 14. Sistema seguimiento de trayectoria planificada y libre de obstáculos.

El mensaje de vector de referencia (V, φ) y el de posición (X, Y, φ) se trasmiten en formato estándar de ROS, todos los demás son de acuerdo a las necesidades intrínsecas del problema y en formato local.

Los esquemas de operatividad que se pueden usar con los nodos presentados, son los siguientes:

a) Driver - Move_Teleop: Realizar teleopración.

b) Driver - Move_Teleop_Ref - Control: Al realizar la teleoperación los valores de los comandos de velocidad son tonados como referencias para la entrada del control.

c) Driver - MT_server - MT_client - Mover: Se emplea para observar la respuesta del sistema a partir de una entrada tipo escalón. Esto, se logra gracias a la configuración cliente-servidor.

d) Driver - Mover_Ref - Control: Seguimiento de una trayectoria ya planificada.

e) Driver - Control: Configuración de control a la espera de un mensaje de referencia de la forma (V, φ).

Para finalizar, los tópicos principales para la suscripción y publicación de mensajes son “velo_bus”, “ref_bus” y “pos2d_bus”. El primero, es el receptor del nodo driver, empleado para enviar mensajes de velocidad de la forma (V_{der}, V_{izq}). El segundo, es la entrada del nodo de control, mensajes de la forma (V, φ). Y el ultimo, revela la posición del móvil, la estructura del mensaje es (X, Y, φ).

2.1.2. INTEGRACION DEL KINECT

2.1.2.1. MATERIALES Y EQUIPOS UTILIZADOS

A continuación se presenta una lista de los materiales y equipos empleados:

a) Computadora: DELL Inspiron mini -Procesador Intel Atom CPU N270 @ 1.60 Ghz-.

b) Sistema Operativo: Ubuntu NetBook Edition 10.10 (Maverick). Núcleo Linux 2.6.35-28-generic. GNOME 2.32.0.

c) Arquitectura de software: ROS DiamondBack desktop full.

d) Plataforma robótica móvil: Robot Roomba 4400 Irobot Create®.

e) Estructura de Apoyo: Tablero de fibra de densidad media (MDF) de 3mm de espesor, tuercas y tornillos de 1/16´´, sus dimensiones se pueden apreciar en la siguiente imagen, ver figura 2.15.

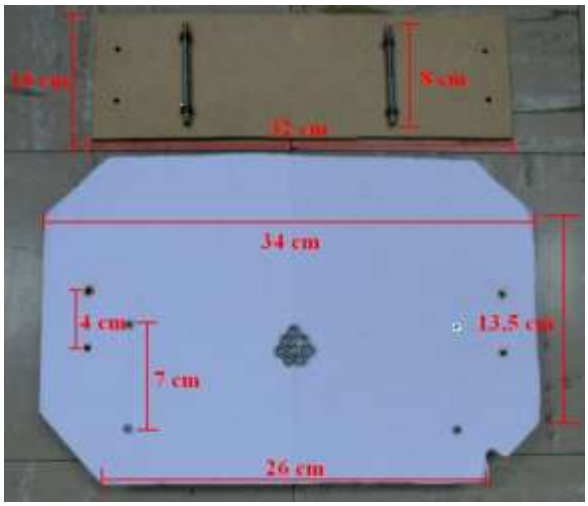

Figura 2. 15. Dimensiones de la estructura de apoyo del ICR.

f) Sensor Kinect.

g) Cable adaptador: Conector USB tipo A macho, Conector USB tipo A hembra, cable de extensión USB reciclable, ver figura 2.17.

h) Elementos varios: Cautín, estaño, silicona liquida, cinta adhesiva, multímetro, fuente de voltaje DC.

2.1.2.2. ALIMENTACIÓN DESDE EL ICR-PC AL KINECT

Para lograr que el sistema ICR-Kinect-PC sea independiente, es necesaria que la alimentación de 12V del Kinect, sea proporcionada por el mismo sistema y no por una fuente de alimentación externa. En este sentido, el ICR cuenta con una fuente de voltaje no regulada, por lo que se puede solventar esta necesidad con la implementación de un regulador *low dropout* de voltaje de 12V a1A.

Para ello, se usan los pines 10 (Vcc) y 14 (GND) del conector DB-25 del ICR. Por otra parte, hay que crear una conexión que una a la fuente regulada y sirva de enlace entre el cable principal del Kinect, configuración tipo hembra de 9 pines, ver figura 2.16, y la entrada, puerto USB, de la computadora, ver figura 2.17.

A continuación se presenta la descripción de los pines, del cable principal del Kinect, tabla 2.3:

Tabla 2. 3. Descripción de Pines del Kinect.

PIN	Descripción
1	Vcc
2	D – USB
3	D + USB
4	Tierra USB
5	+ 5V USB
6	Vcc
7	Tierra
8	Tierra
9	Sin Uso

Figura 2. 16. Disposición de los pines del Kinect.

Figura 2. 17. Cable para conexión ICR-Kinect-PC.

2.1.2.3. ESTRUCTURA DE SOPORTE

De modo que el ICR pueda trasladarse con una mini laptop y un Kinect a bordo, es necesario que cuente con una estructura de apoyo, que sea de fácil acople y que no posea mucha altura, para evitar efectos de inestabilidad, ver figura 2.18, y 2.15 para sus dimensiones.

Figura 2. 18. Estructura de apoyo del ICR.

2.1.2.4. SERVICIO DE DATOS DEL KINECT

Se establecen las configuraciones necesarias que permitan la adquisición de datos del Kinect, y en relación al inicio del presente capitulo se divide el procesamiento de la información en: propósitos generales y navegación.

2.1.2.4.1. CONFIGURACIÓN DE ROS

Se hace uso del paquete Openni Kinect. Perception Pcl Addons y de Turtlebot. El primero de ellos es la base para establecer la comunicación entre el Kinect y la computadora, el segundo para poder realizar operaciones con la librería de PCL en ROS. Finalmente el tercero, se toma como plataforma para operar con otro tipo de mensaje, el Laser Scan.

Con esto se pueden desarrollar nuevos nodos y usar o modificar los ya existentes para la generación de un servicio de datos con el Kinect. En este sentido se puede:

a) Tener acceso a la cámara RGB, a imágenes de profundidad y a la nube de puntos como un mensaje estándar de ROS tipo PointCloud2.

b) Filtrar la nube de puntos con Voxel-Grid (tener una nube de puntos menos densa), o con Passthrough Filter y tener acceso a un plano de la imagen.

c) Realizar una estratificación de la nube de puntos, es decir, crear un mensaje con sólo una fila de la matriz de 640x480. Y hacer que la selección de la fila pueda realizarse por medio de un par cliente-servidor.

d) Realizar una conversión del mensaje en formato PointCloud2 a LaserScan. Este se caracteriza por tener una representación de la nube de puntos en un

solo plano (como si fuera un láser) y cada punto es representado por su radio y Angulo (r, φ).

e) Hacer segmentación de la data en formato estándar LaserScan.

f) Generación de rectas de aproximación, de la forma de la ecuación 1.1, 1.4 y 1.5, a partir de la nube de puntos de un plano.

g) Generar vectores de velocidad de referencia, a partir de las rectas de aproximación.

Adicionalmente, se hace uso de la herramienta gráfica Rviz, que permite la visualización de la nube de puntos del Kinect y de otras estructuras de mensajes de ROS, como por ejemplo el formato LaserScan.

2.1.2.4.1.1. NODOS DE PROPOSITOS GENERALES

A continuación se da una descripción de los nodos empleados en el Kinect con propósitos generales.

a) Openni_Camera: Pertenece al paquete de Openni Kinect, es el encargado de establecer la comunicación entre el Kinect y la computadora. Sus principales características son aportar la nube de puntos en formato estándar PointCloud2 (tópico /camera/rgb/points) y realizar la calibración correspondiente entre el sensor de profundidad y la cámara RGB, ver figura 2.19, se denomina como openni_node1.

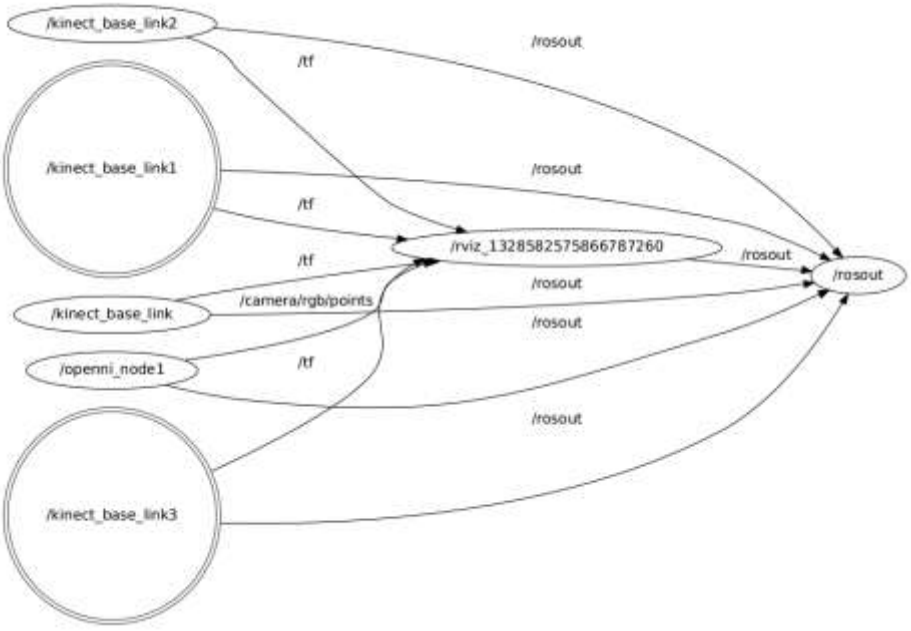

Figura 2. 19. Conexión entre el Kinect y la computadora por medio de Openni Kinect y visualización con Rviz.

b) Ver: Es un programa encargado de tomar la nube de puntos y visualizarla por medio de PCL. Este se toma como base para representar la nube de puntos con o sin los filtros plantados, ver sección 1.8. En este sentido, VerXY es un programa que utiliza Passthrough Filter para generar el plano XZ, desde la perspectiva del Kinect, pero que corresponde al plano XY desde la referencia del IC, ver figura 2.20 en representación de la conectividad con Openni Kinect.

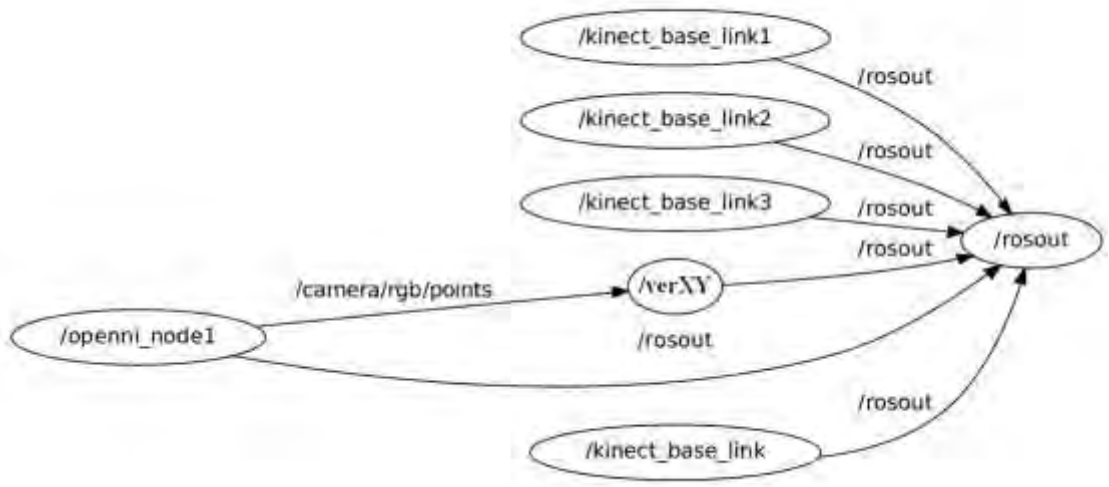

Figura 2. 20. Generación del plano XY a partir de la nube de puntos.

c) Kinav3-Indice: Es un par cliente servidor que proporciona una fila, de la matriz de 640x480, con los datos (X, Y, Z) de cada punto. Por defecto, el programa (kinav3) se encuentra en el centro del Kinect, es decir en la fina 240, pero se puede hacer un requerimiento de cambio de fila por medio del nodo Índice, ver figura 2.21.

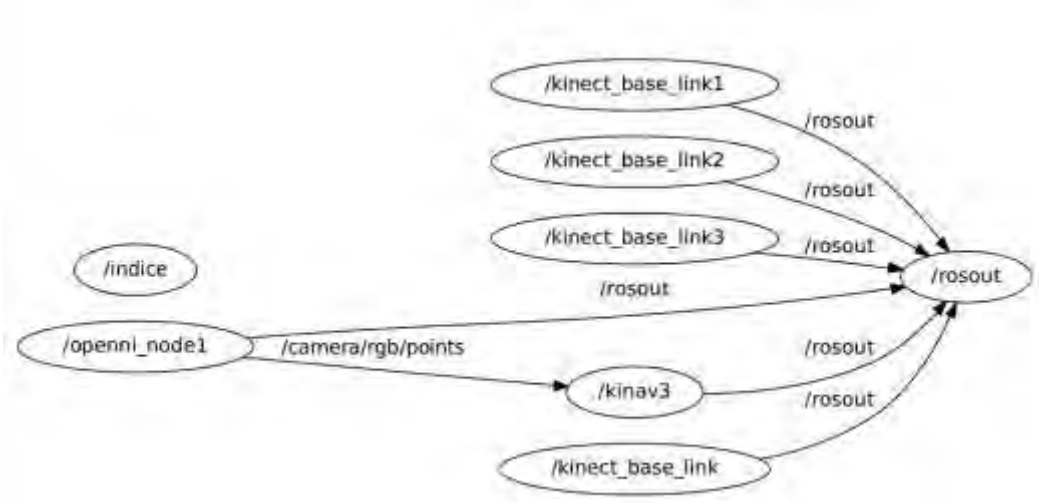

Figura 2. 21. Generación de un plano XY a partir de la obtención de una fila de la nube de puntos.

Los nodos kinect_base_link están asociados a la posición de la base del sensor, de la cámara RGB y del sensor de profundidad con fines de calibración.

2.1.2.4.1.2. NODOS PARA NAVEGACIÓN

A continuación se da una descripción de los nodos empleados en el Kinect y el Roomba, con motivo de establecer las bases para realizar navegación en un ambiente definido, pasillo.

a) To_Laser: Es un archivo perteneciente al paquete Turtlebot, se encarga de transformas la nube de puntos de PointCloud2 a formato estándar LaserScan, Además crea un plano, seleccionando los puntos que se encuentren en un rango de altura (entre 0,1m y 0,15m) y descartando los demás. El nuevo mensaje posee las siguientes características (tabla 2.4):

Tabla 2. 4. Características de mensaje LaserScan.

Angle_min	-1,57079637051 rad
Angle_max	1,57079637051 rad
Angle_increment	0,00872664619237 rad
Scan_time	0,0333333350718 s
Range_min	0,44999998079 m
Range_max	10 m
Topic	Scan (Sensro_msgs/laserScan)

b) To_splitter: es un archivo perteneciente al paquete Turtlebot, y se caracteriza por tener como parámetros el número de divisiones que se le desea realizar a un mensaje tipo LaserScan y cuantos puntos le corresponde a cada uno. En este sentido, se hace una división por la mitad de los puntos, obteniendo una

vista a la izquierda y a la derecha del centro del Kinect., la conectividad se puede apreciar en la siguiente imagen (figura 2.22.). Los dos nuevos mensajes heredan las características generales del mensaje padre.

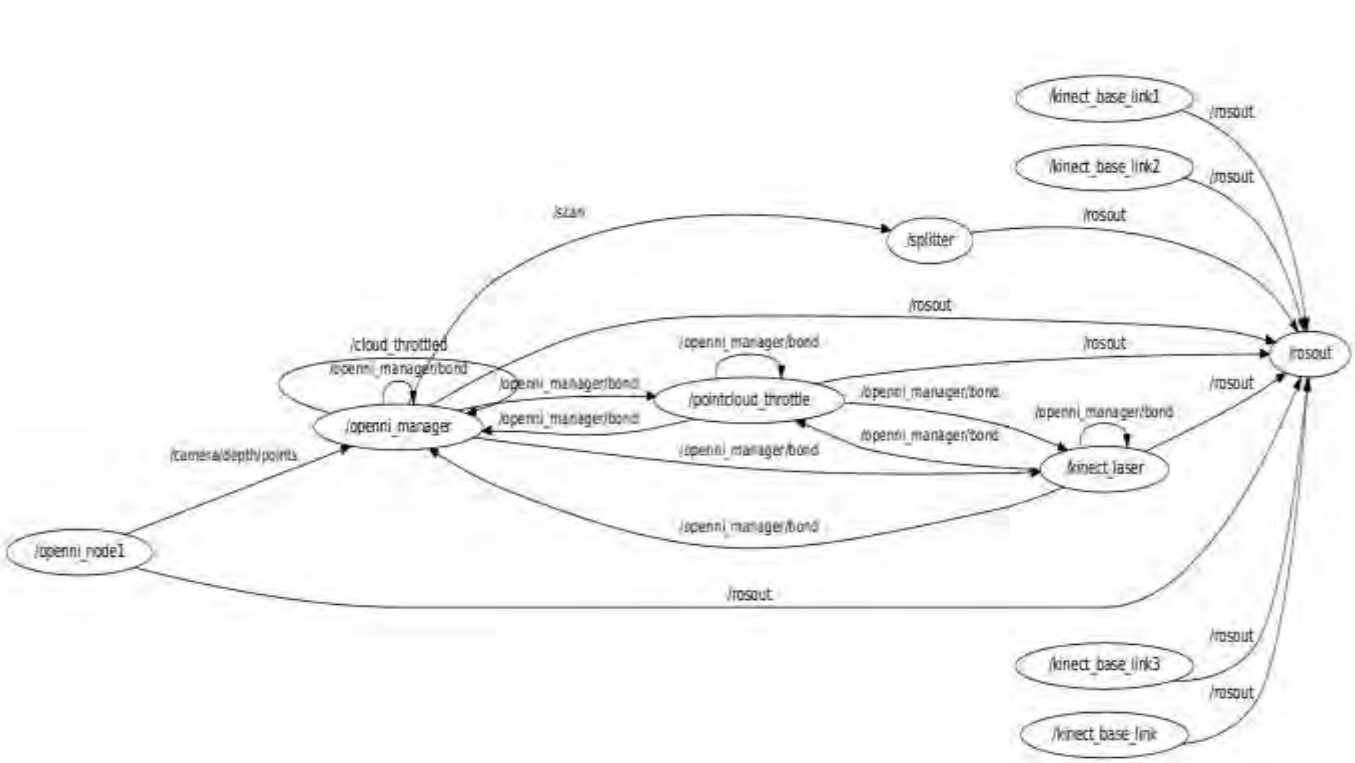

Figura 2. 22. To_splitter a partir de un mensaje LaserScan.

c) NovelScan: A partir de los mensajes que genera To_splitter, se hace la construcción de rectas que mejor representen la nube de puntos proporcionada, por medio de regresión lineal. Y se genera un mensaje en formato estándar de ROS, para publicar las características más relevantes de la recta, como: pendiente, ordenada, puntos inicial y final donde excite y cuantas muestras se usaron para generarla, se crea al tópico "recta_bus" para estos propósitos. Por otra parte, existe NovelScan2 y NovelScan, el primero genera la recta a partir de los puntos dados y el segundo hace esto, pero el mensaje final es una recta paralela con una separación ya fijada, además por

medio del conocimiento de la posición (X, Y, φ) en la que se encuentra hace que la recta que se creó este referenciada a esa coordenada.

d) Vecref: Generación de vectores de referencia en formato (V, φ) para ser enviados al control del ICR, a partir del mensaje de NovelScan. Este nodo se asemeja a Mover_Ref, pero en vez de poseer una trayectoria ya planificada, esta se determina por la recta que recibe como parámetro. Emplea el tópico "ref_bus" para publicar.

e) Vecref2: Genera un vector de referencia en formato (V, φ) para ser enviado al control del ICR, a partir del mensaje de NovelScan2. Este vector es de módulo fijo y es paralelo a la recta obtenida por mensaje. Emplea el tópico "ref_bus" para publicar.

f) Ubicación: Es un nodo que emite la ubicación del móvil (X, Y, φ) en un mensaje estándar de ROS. Emplea el tópico "ubicación" para publicar, este se asemeja a "pos2d_bus", su diferencia radica en que se emplea una estructura estándar de mensajes de ROS y no una configuración local.

g) Robot_TF: Es un programa encargado de realizar la transformación de la base del Kinect al centro del ICR, dado que estos dos coinciden en (X, Y) solo hay que hacer el ajuste en el eje Z. De este modo, el vector de transformación resultante es un desplazamiento de 0.12m solo en el eje Z. Se toma como referencia el eje de coordenadas del ICR.

Por último, para la ejecución de los nodos se utilizan archivos de ejecución (launch files), que no son mas, que un modo presentado en ROS para la ejecución de programas. De esta manera, se hace una diferenciación entre los nodos presentes en el ICR con los del Kinect, con lo que resultan los siguientes archivos, ver tabla 2.5:

Tabla 2. 5. Launch files y los nodos que ejecuta.

.Launch	Contenido
Create	Driver
	Ubicación
	Control
Kinect	to_laser
	to_splitter
	NovelScan2
	Vecref
Create2	Driver
	Ubicación
	vecref2
Kinect2	to_laser
	to_splitter
	NovelScan

Es así, que en el sistema ICR-PC-Kinect, queda integrado de forma física, como se muestra en la figura 2.23, y con una conectividad de programas en base a ROS, como se muestra en la figura 2.24.

Figura 2. 23. Sistema de integración del ICR, Kinect y una computadora utilizando ROS.

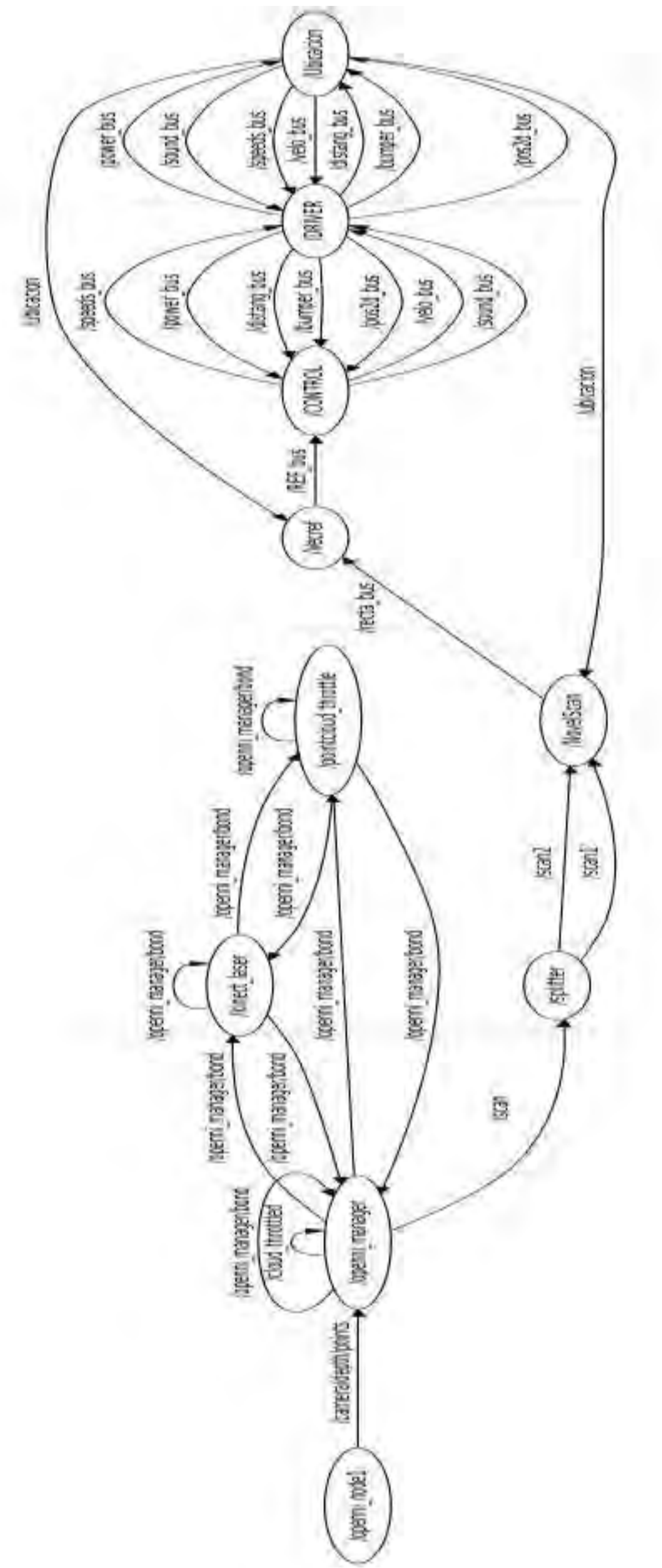
/openni_node1
/camera/depth/points
/openni_manager
/openni_manager/bond
/kinect_laser
/pointcloud_throttle
/scan
/splitter
/scan2
/recta_bus
/REF_bus
/CONTROL
/DRIVER
/speeds_bus
/power_bus
/bumper_bus
/pos2d_bus
/velo_bus
/sound_bus

Figura 2. 24. Sistema de integración del ICR, Kinect y una computadora utilizando ROS.

2.1.3. EXPORTACIÓN DE LA INFORMACIÓN

Con el propósito de poder realizar monitoreo en línea o fuera de línea (del inglés: *on line, off line*), se debe de contar con la posibilidad de compartir o almacenar la información, esto puede ser de manera local o externa.

Por una parte, una herramienta útil en ROS respecto a grabar todo lo relacionado a los tópicos, generados y asociados en un momento y en una instancia de cómputo dada, es el paquete rosbag. Este, permite grabar todos los mensajes que se estén publicando en un momento ó solo aquellos que interesen, lo que resulta es un archivo .bag que puede ser reproducible, con la misma herramienta.

Dado que, los archivos generados por el paquete rosbag son útiles para reproducir información en una instancia de ROS, es de provecho almacenar los datos en otro formato. En este sentido, los nodos que poseen mensajes relevantes como señales de control o parámetros para generación de vectores de referencia o ubicación del móvil, generan archivos de formato .txt ya que es una manera genérica de almacenaje y de fácil reproducción.

Estas dos modalidades permiten realizar operaciones y estudios fuera de línea, por plataformas externas al sistema. Es por ello, que con el uso de ROS y su capacidad de múltiples maquinas se pueden llevar a cabo tareas de monitoreo y de control remoto.

Para hacer esto posible, se selecciona una computadora que ejecute el nodo maestro (roscore), mientras que a la otra se le indica la dirección donde se encuentra el nodo maestro, por medio de la instrucción ROS_MASTER_URI. Cabe destacar,

que si se encuentra en una red *wireless* se deben habilitar el reenvió de puertos, para la computadora que tenga el roscore.

CAPÍTULO 3

ANÁLISIS DE RESULTADOS

A fin de dar continuidad a la metodología planteada, esta sección se presenta en tres módulos principales, de la misma forma en que se ha dividido el trabajo. Es este sentido se tiene: Control del robot Roomba, integración y exportación de la información.

3.1. CONTROL DEL ROBOT MÓVIL ROOMBA

Para la ejecución de estas pruebas se ha seleccionado la primera área de trabajo, figura 2.3. Con esto, se puede estudiar el comportamiento del módulo y el ángulo de la velocidad a partir de una entrada tipo escalón.

Respecto al módulo, se obtiene una respuesta que presenta un comportamiento de carácter sub-amortiguado, ver figura 3.1.

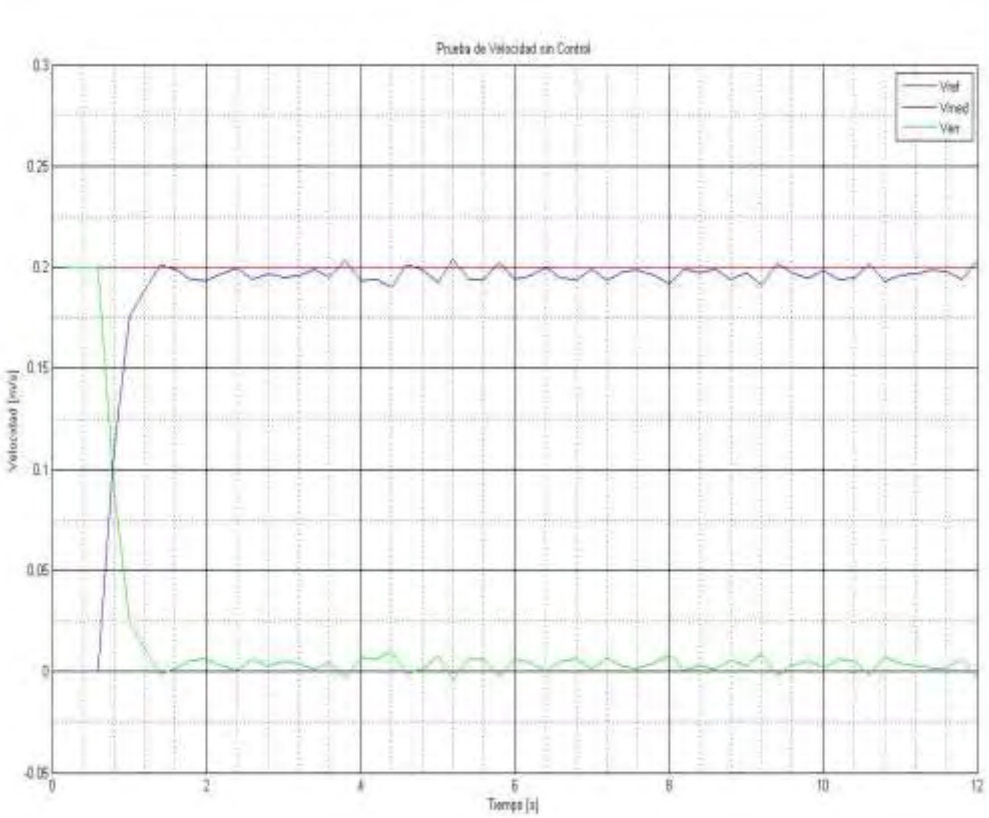

Figura 3. 1. Respuesta a una entrada escalón, del módulo de la velocidad, a lazo abierto.

En este sentido, se usó del segundo método de Ziegler-Nichols, para determinar los parámetros de un sistema de control. Para esto, hay que hallar la ganancia crítica que hace que el sistema se vuelva inestable y que provoca que el sistema oscile de forma constante. En este sentido se obtiene la siguiente respuesta, figura 3.2.

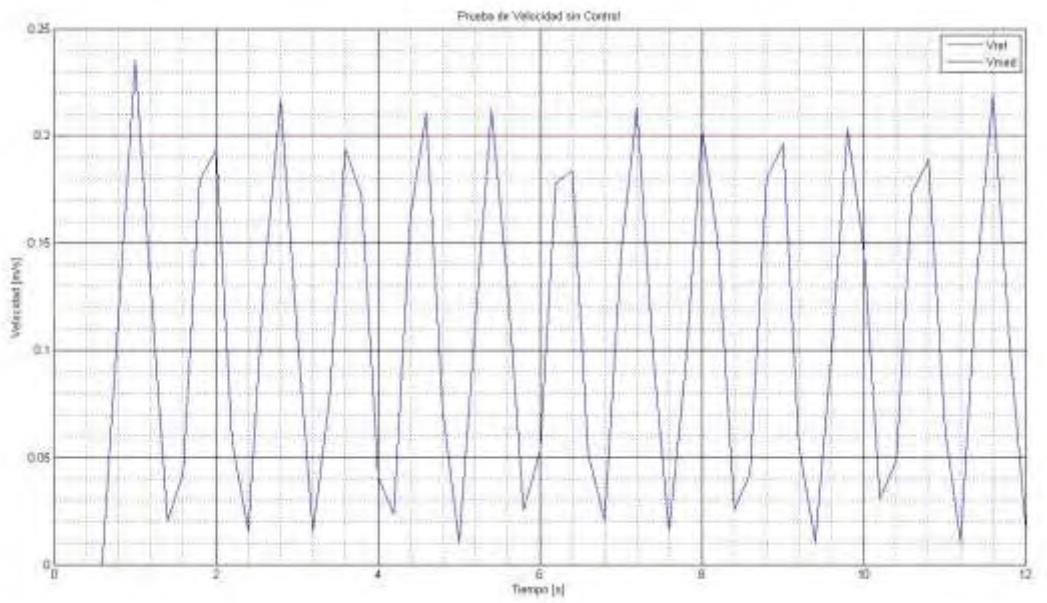

Figura 3. 2. Sistema inestable en oscilaciones constantes.

Con esto, se pueden determinar los siguientes parámetros, ver tabla 3.1:

Tabla 3. 1. Valores de ganancias para el módulo de la velocidad, dado una Kcr = 1,24 y Pcr = 0,8.

Control	Kp	Ti	Td
P	0,620	infinito	0
PI	0,558	0,925	0

Respecto al control tipo P, se logra conseguir un error de alrededor del 37,67%, de la señal de referencia, para cualquier valor de la entrada, ver figura 3.3.

Figura 3. 3. Control P en módulo de la velocidad, a diferentes entradas tipo escalón.

Por otra parte, las pruebas con el control PI muestran un tiempo de establecimiento, a un 95% de la señal de referencia, de aproximadamente cinco segundos (figura 3.4), y un error en estado estacionario que tiende a cero, para cualquier valor en la entrada (figura 3.5).

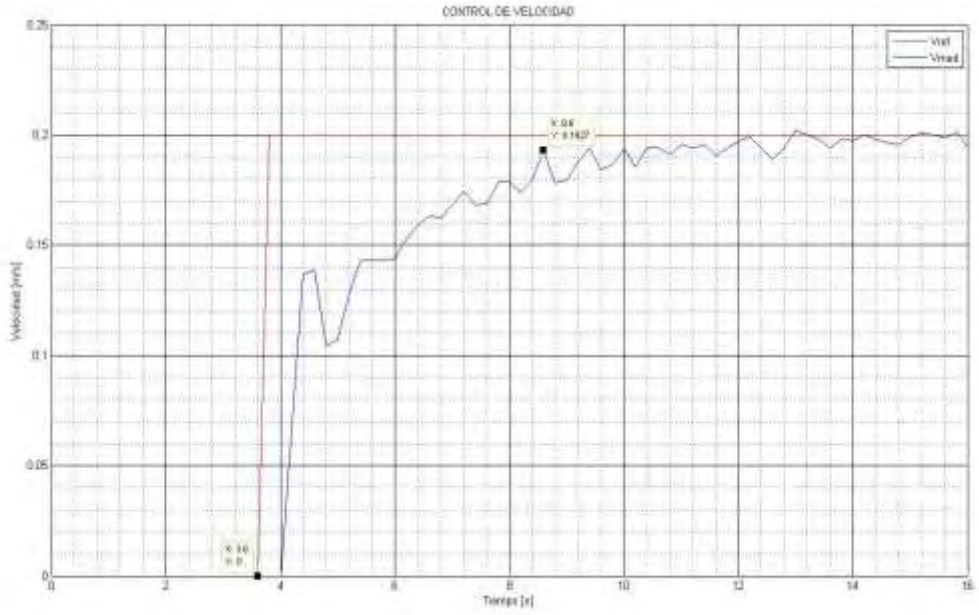

Figura 3. 4. Control PI en módulo de la velocidad, entrada escalón de 0,2 m/s.

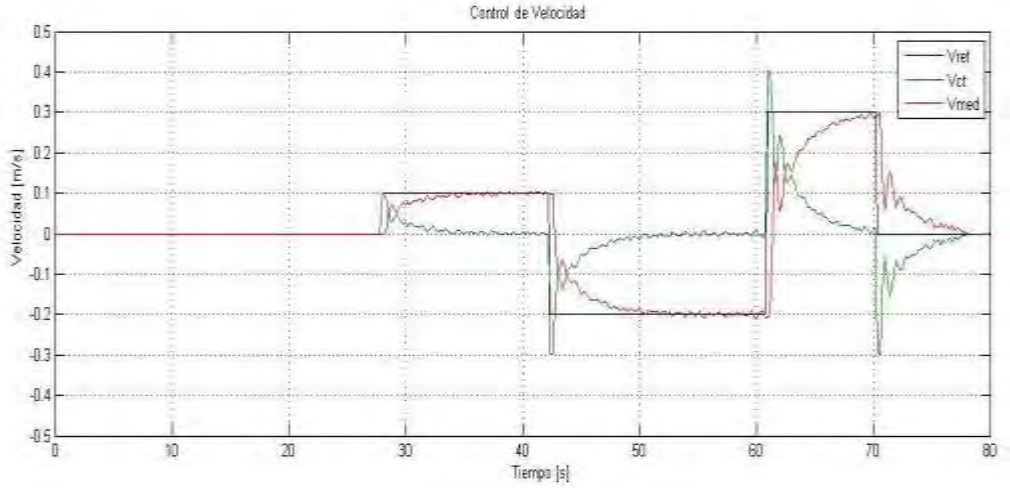

Figura 3. 5. Control PI en módulo de la velocidad, a diferentes entradas tipo escalón.

Ahora bien, respecto al ángulo de la velocidad se presenta una salida en forma de S, ver sección 2.1.1.2. y la siguiente imagen (figura 3.6), con un error de 22,23%.

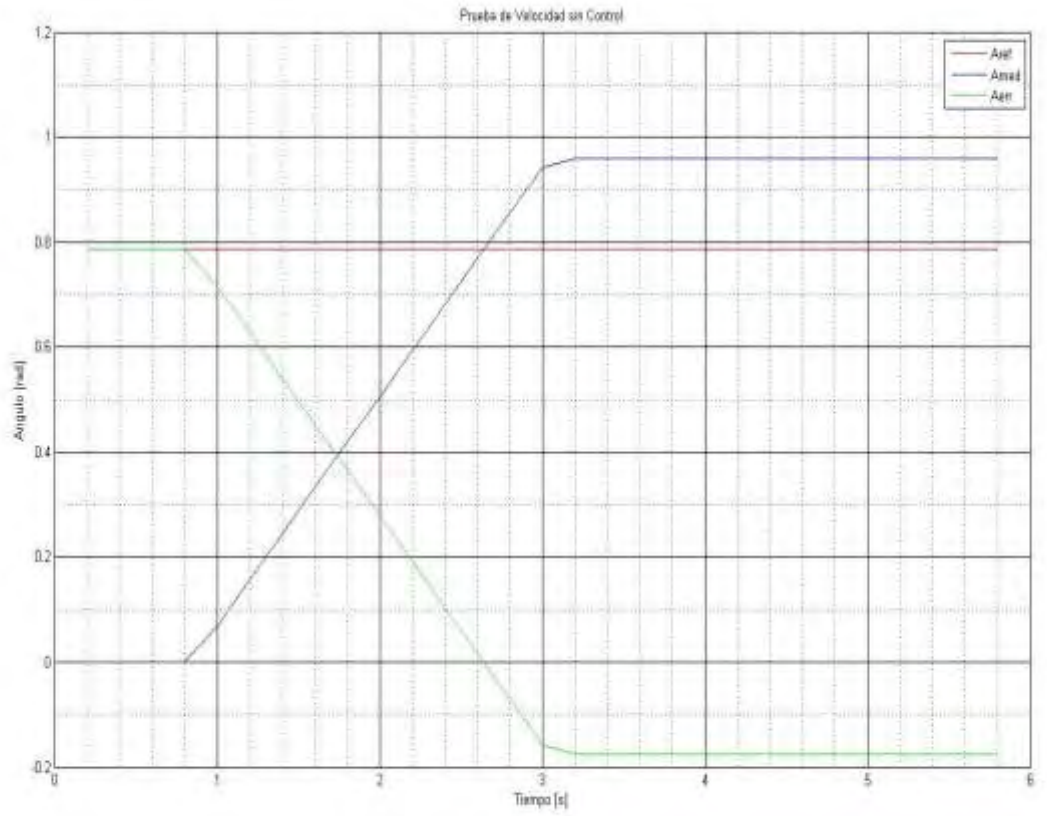

Figura 3. 6. Respuesta tipo S del ángulo de la velocidad.

Del resultado, se puede establecer el valor que debe poseer una constante para realizar un control tipo P, con las expresiones 2.9 y 2.10. Es así, que T = 1,05 y L = 0,95 y cumplen la relación 2.8. De modo que, la constante Kp para un control P es: Kp = 1,1052.

En este sentido se tiene un control con el siguiente comportamiento (figura 3.7).

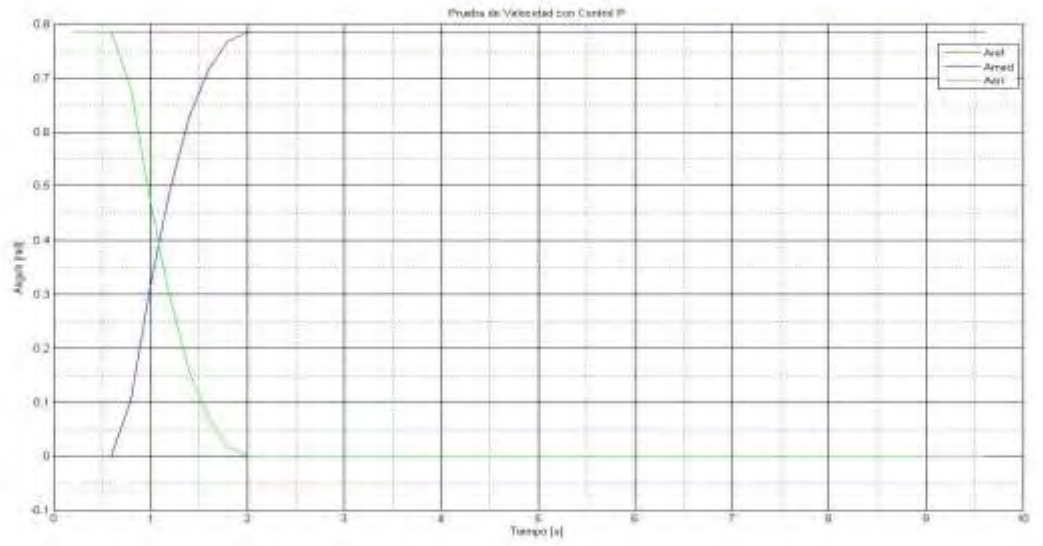

Figura 3. 7. Control P en ángulo para una entrada de 0,7853 radianes (45°).

De la gráfica anterior, se puede observar que a la señal medida le toma dos segundos alcanzar el valor de referencia y mantenerse sin cambios en el tiempo. A continuación, en la figura 3.8 se muestra la salida del control ante cambios de referencia, que corrobora la tendencia a cero del error en estado estacionario.

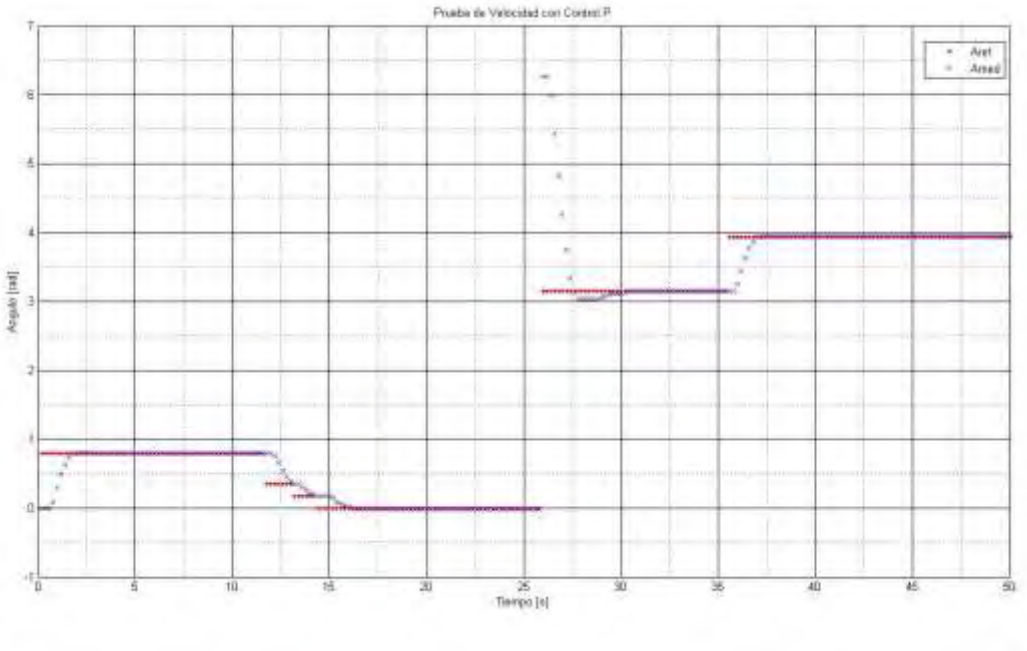

Figura 3. 8. Control P en ángulo, para varias entradas.

El salto presente en la figura anterior (al tiempo de 26 segundos), se debe a la continuidad de 0° a 360° en el cálculo del ángulo medido, ver sección 2.1.1.5.1.

Ahora bien, se cuenta con un sistema capaz de proveer control a partir de señales de referencia en módulo y ángulo, (V, φ), con lo que se procederá a comprobar el nodo de seguimiento de una trayectoria planificada y libre de obstáculos (Mover_Ref), se plantean dos escenarios: una recta paralela al eje X y otra paralela al eje Y, además el móvil puede iniciar el recorrido en su posición inicial o desplazado. Por otra parte, se destacarán los efectos de usar el coseno del ángulo del error para escalar el módulo de la velocidad [16].

En primer lugar, se tiene que el móvil parte del origen, ver figuras 3.9 y 3.10, pero la diferencia de distancia de la recta al punto inicial es mayor en la primera imagen que en la segunda, 1,5m y 1m respectivamente. Esta diferencia hace que el comienzo de la aproximación (caso recta Y = 1,5) tienda a buscar una forma perpendicular a la recta planteada, pero una vez superado un tramo del recorrido la trayectoria presenta un comportamiento asintótico, el cambio de tendencia se puede apreciar en el acercamiento del recorrido, figura 3.9. Esto se debe a que hasta el momento de cambio de tendencia el punto más próximo de la recta al móvil era aquel que determinaba un vector de velocidad perpendicular a la trayectoria.

Respecto a la otra recta planificada (X = 1), se presenta un comportamiento asintótico desde el inicio hasta el final como se muestra en la figura 3.10.

Ahora, para poder apreciar el comportamiento del seguimiento de una trayectoria con el uso del módulo de la velocidad escalado con el coseno del ángulo del error, se procedió a desplazar al ICR del origen de coordenadas, por medio de teleoperación, y ubicarlo en una posición en sentido opuesta a la trayectoria, ver figuras 3.11, 3.12, 3.13, 3.14. En las dos primeras, se usa el módulo sin escalarlo, y

la característica dominante es que el móvil ejecuta un giro abierto para orientarse hacia la trayectoria. En cambio, en las pruebas siguientes la orientación se corrige con el uso del retroceso, ver los acercamientos en las figuras 3.13 y 3.14.

Esto se comprueba observando el comportamiento de la velocidad. Para esto se puede observar la figuras 3.15, en donde las referencias de velocidad siempre son positivas al igual que las mediciones (caso módulo no escalado). En cambio, en la figura 3.16, se puede observar que a pesar de que la referencia de velocidad es positiva la medición arroja valores negativos (en el intervalo de tiempo de 0 a 5 segundos). Este comportamiento se debe a que el ICR puede acceder a comandos de velocidad menores a cero y por ende retroceder. Todo esto se traduce a que el móvil puede orientarse hacia su objetivo de una mejor manera y no invierte desplazamiento en esta tarea, además usa un menor espacio para ubicarse.

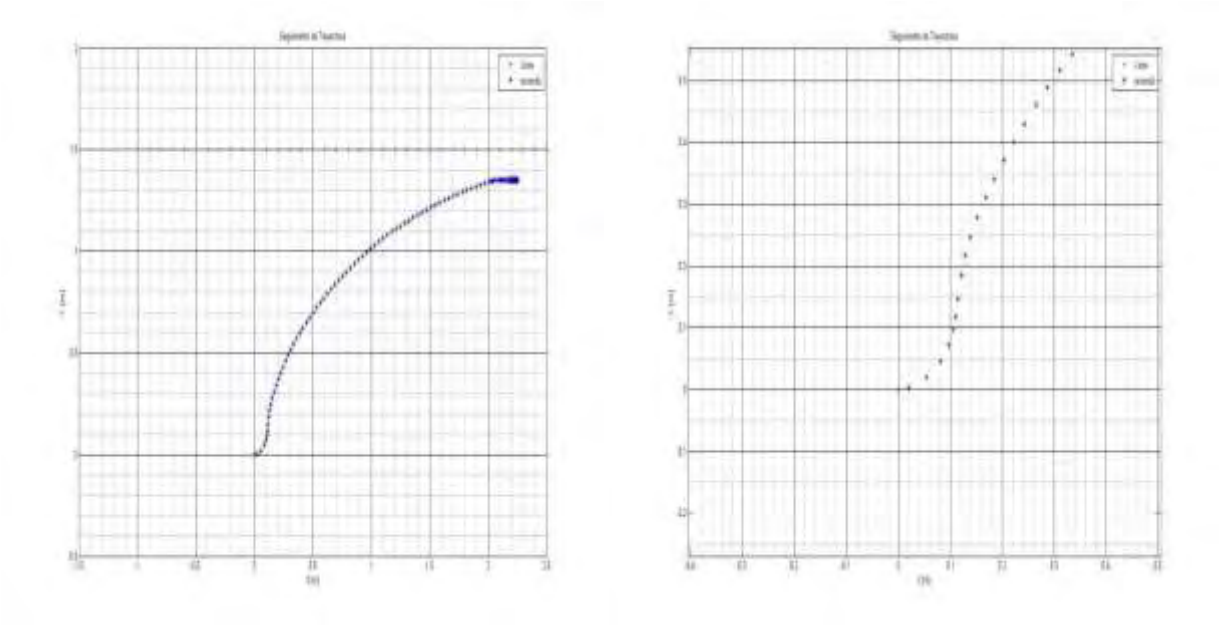

Figura 3. 9. Seguimiento de Trayectoria paralela al eje X, partiendo del origen. Izquierda: recorrido total. Derecha: el arranque, entre -0.4m y 0.5m en horizontal y 0.2m y 0.5m en vertical.

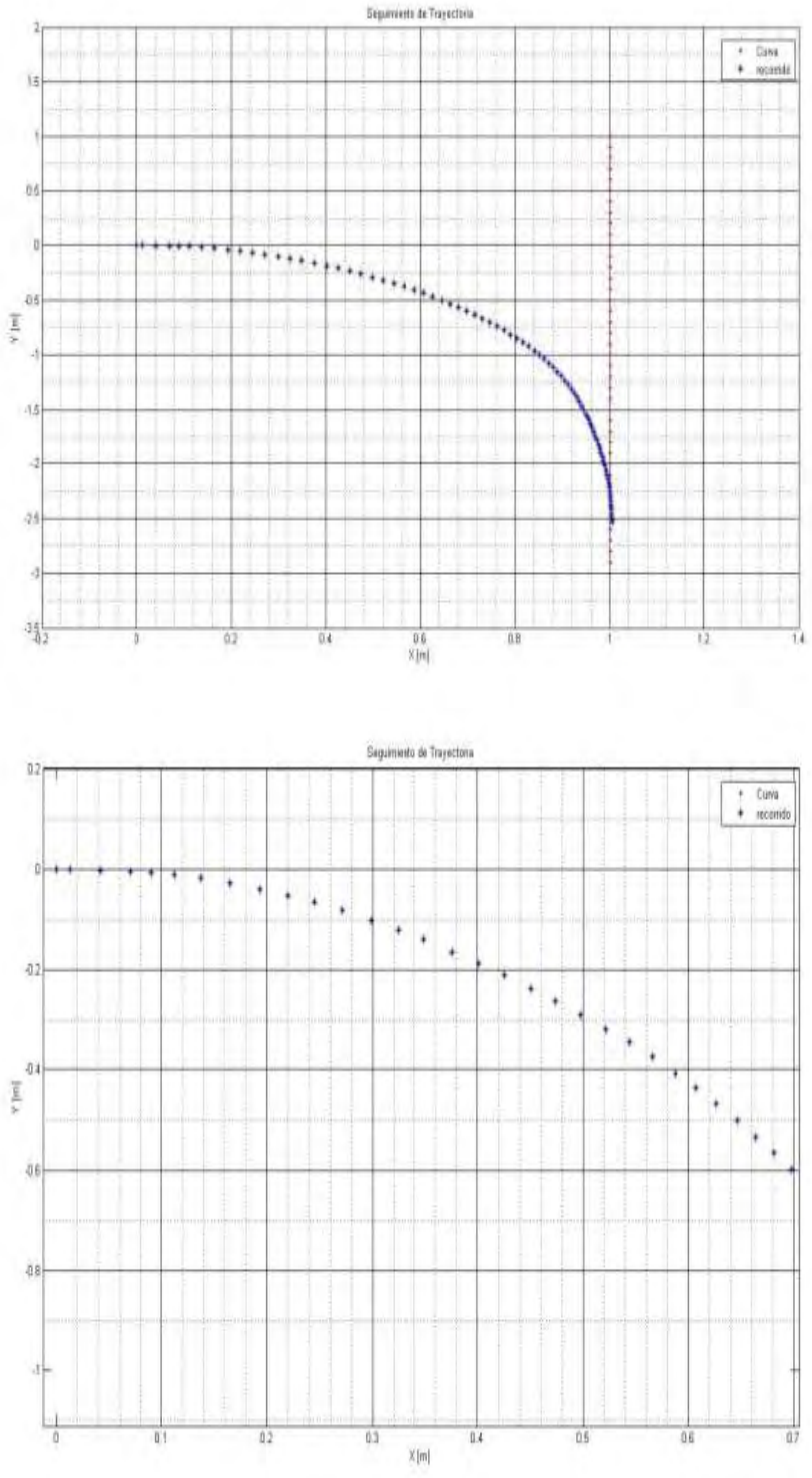

Figura 3. 10. Seguimiento de Trayectoria paralela al eje Y, partiendo del origen. A la Superior: recorrido total. Inferior: el arranque, entre 0m y 0.7m en horizontal y -1m y 0.2 en vertical.

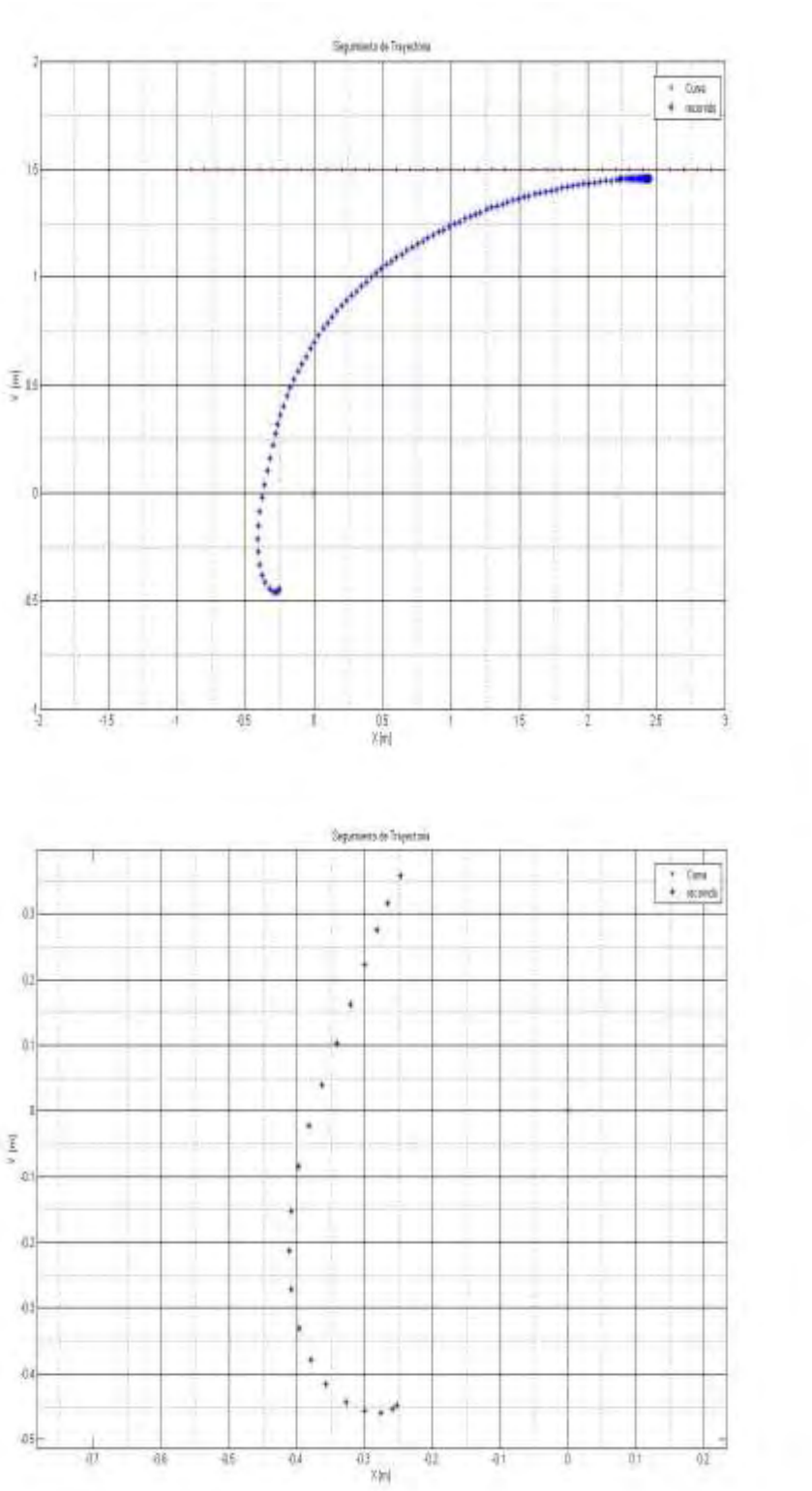

Figura 3. 11. Seguimiento de Trayectoria paralela al eje X, posición inicial desplazada del origen. Superior: recorrido total. Inferior: el arranque, entre -0.7m y 0.2m en horizontal y -0.5m y 0.3 en vertical.

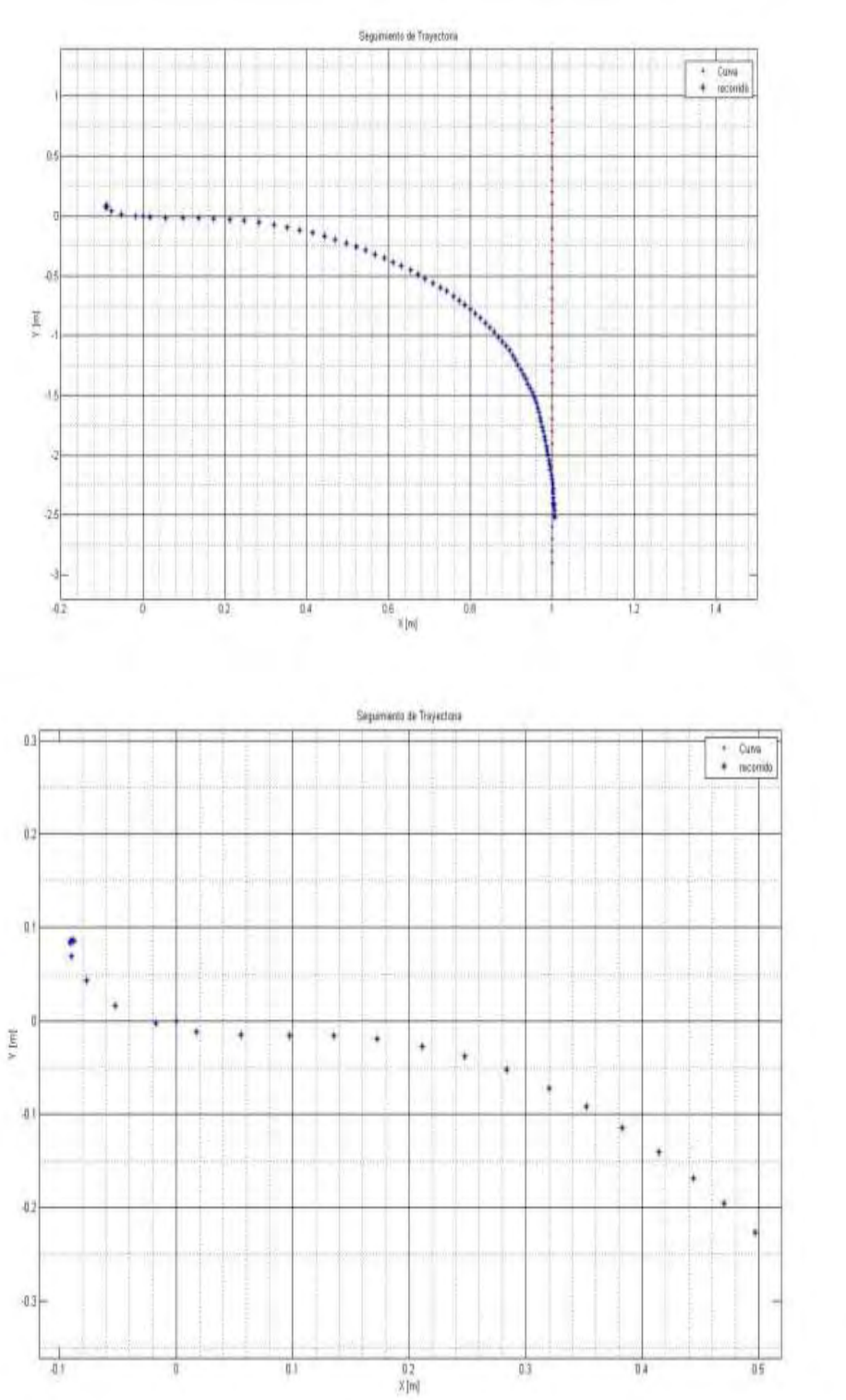

Figura 3. 12. Seguimiento de Trayectoria paralela al eje Y, posición inicial desplazada del origen. Superior: recorrido total. Inferior: el arranque, entre -0.1m y 0.5m en horizontal y -0.3m y 0.3 en vertical.

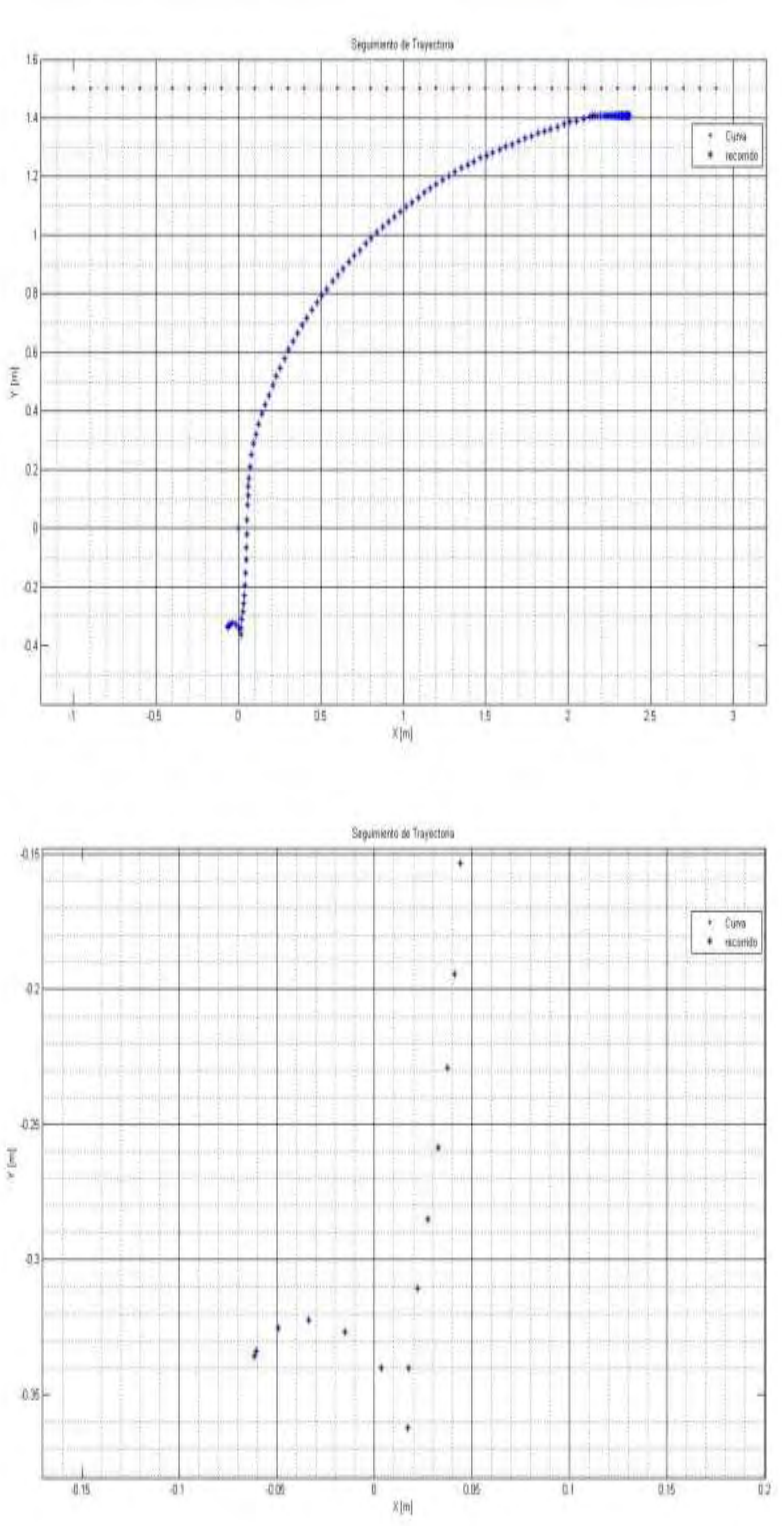

Figura 3. 13. Seguimiento de Trayectoria paralela al eje X, posición inicial desplazada del origen, módulo de la velocidad escalado. Superior: recorrido total. Inferior: el arranque, entre -0.15m y 0.2m en horizontal y -0.35m y -0.15 en vertical.

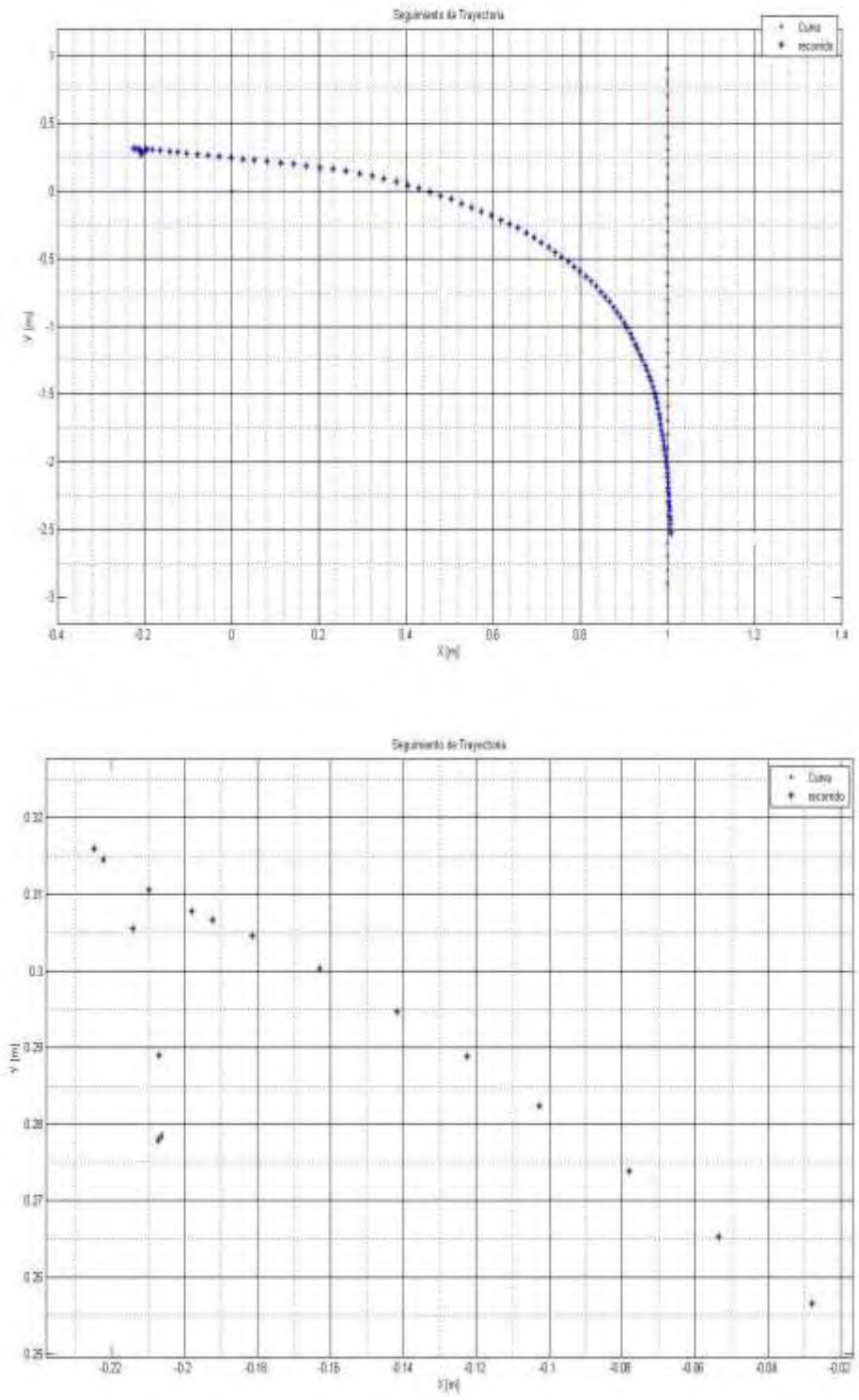

Figura 3. 14. Seguimiento de Trayectoria paralela al eje Y, posición inicial desplazada del origen, módulo de la velocidad escalado. Superior: recorrido total. Inferior: el arranque, entre -0.22m y -0.02m en horizontal y 0.25m y 0.32 en vertical.

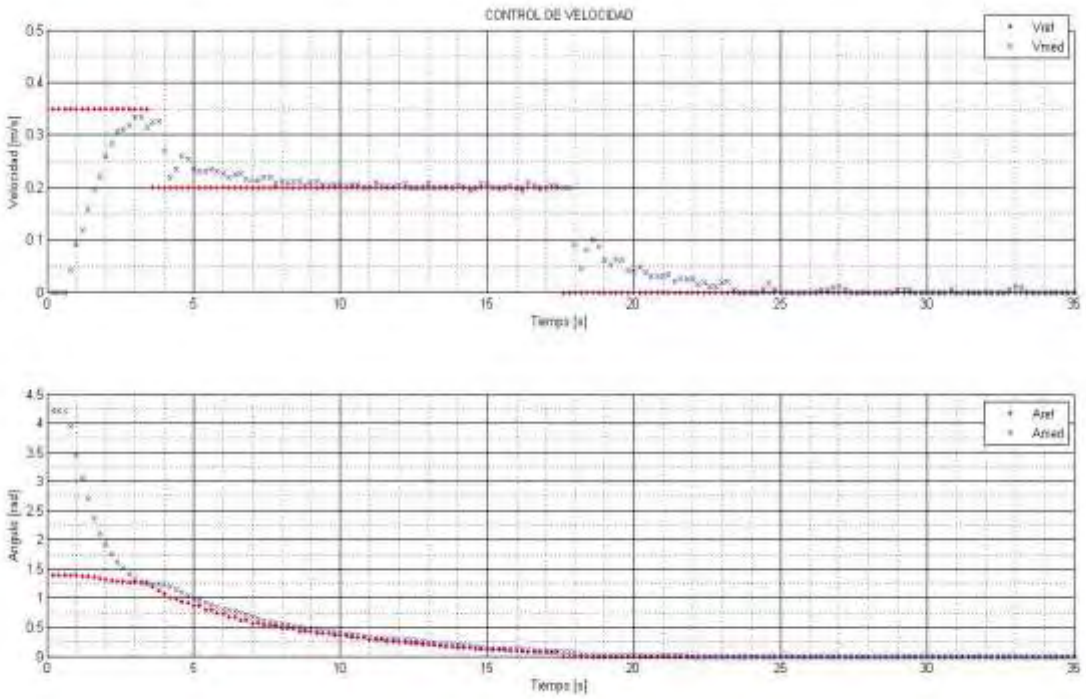

Figura 3. 15. Control de vector de velocidad sin haber escalado el módulo de la velocidad.

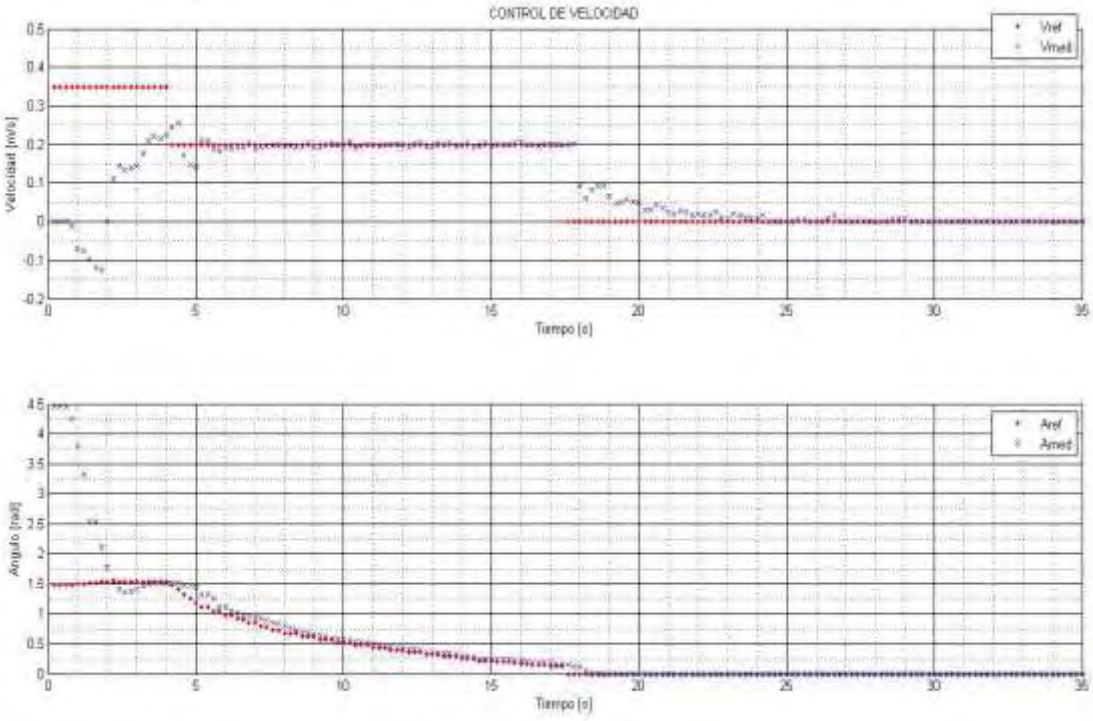

Figura 3. 16. Control de vector de velocidad el módulo de la velocidad escalado.

3.2. INTEGRACIÓN DEL KINECT

De acuerdo a lo establecido en la sección 2.1, se posee un servicio de datos relativo a la adquisición del Kinect y a un procesamiento de la información enfocado a sentar las bases para trabajos futuros. Por otra parte, se tiene un servicio de datos dirigidos a realizar navegación en un pasillo de características conocidas. En este sentido se presentan los siguientes resultados.

3.2.1 SERVICIO DE DATOS PARA PROPOSITOS GENERALES

Se ejecutaron las pruebas en la segunda área de trabajo, figura 2.4, y en una pared con una distancia al Kinect de 1,9 metros, ver figura 3.17.

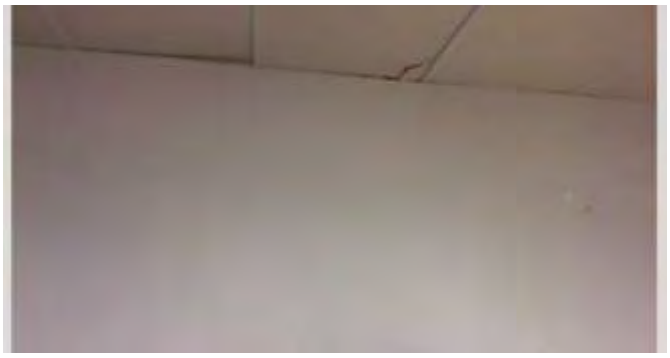

Figura 3. 17. Pared vista con la cámara RGB del Kinect.

Se realizó una captura de datos en los dos entornos de estudio, pasillo y pared, para luego visualizarlos por medio de diferentes aplicaciones y realizar análisis comparativo con diferentes tipos de procesamiento.

De esta manera, se tiene que los datos se pueden visualizar por medio de: las herramientas de visualización de PCL, del entorno grafico "Rviz" de ROS o por otros programas que tengan compatibilidad con el formato en que se almacene la información, como por ejemplo Matlab®, con un archivo en formato texto (.txt).

Para empezar, se empleó Rviz con la nube de puntos en niveles de profundidad, con tomas en diferentes perspectivas (frontales, en plano y laterales), ver figura 3.18, de las que se puede apreciar: la vista del pasillo desde la base del Kinect como una imagen RGB (a) y de profundidad (f), la diferencia de alturas entre las paredes que conforman el pasillo (c), la presencia del techo (representado en color morado, b), que el pasillo izquierdo, desde el punto de vista de la cámara, es más uniforme que el derecho (d), en este se pueden apreciar discontinuidades, producto de las columnas y la puerta que se encuentra presente (la puerta es el espacio vacío en la mitad de la pared derecha, e y g).

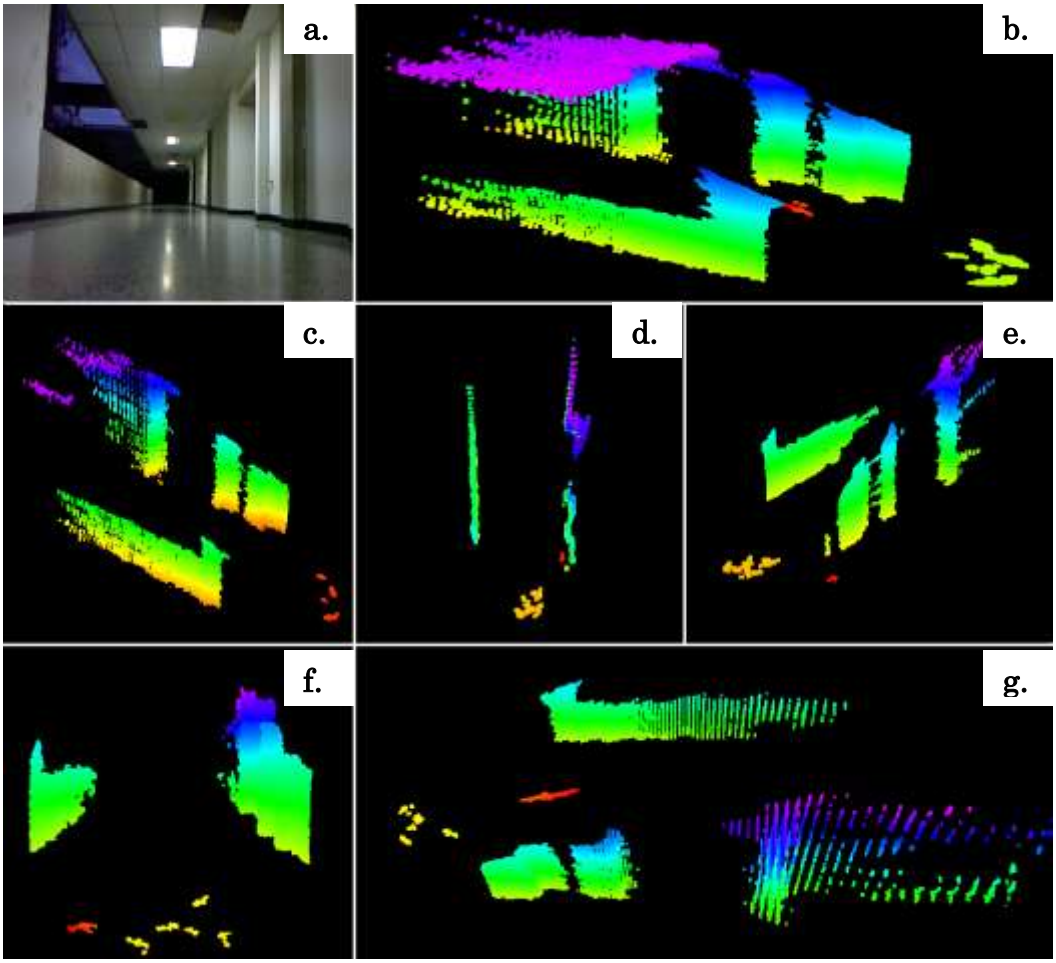

Figura 3. 18. Visualización del pasillo por medio de Rviz.

Con las librerías de visualización de PCL también se pueden tener representaciones gráficas del entorno percibido, en el caso de la figura 3.19 en RGBXYZ, esto depende del formato con que se manipulen los datos, bien podrían ser las coordenadas sin color, un plano o una representación de una imagen de profundidad.

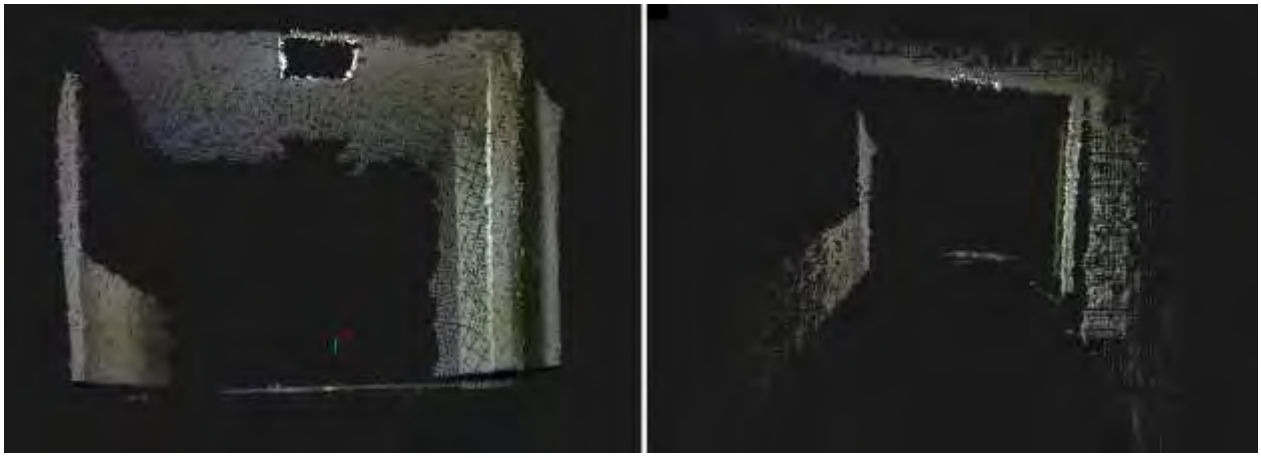

Figura 3. 19. Visualización del pasillo por medio de PCL.

La ventaja o desventaja que pueda tener una herramienta con la otra radica en el formato en que se emplee la información, ya que Rviz, por ser propio de ROS, tiene una estructura de mensaje de tipo PoinCloud2 que es estándar para su entorno, en cambio para PCL se debe realizar una transformación de este mensaje a uno estándar en PCL, para poder emplear sus herramientas. Cabe destacar, que las estructuras con las que se manipula la nube de puntos con PCL pueden ser implementadas como mensajes en ROS.

Ahora, se procesará la nube de puntos para obtener una representación del plano XY, referenciado al IC.

En este sentido y haciendo uso de PCL y del filtro Passthrough Filter, nodo VerXY (sección 2.1.2.1.1.), se puede obtener la siguiente representación, figura 3.20, que resulta ser un plano de la figura 3.19, donde se pueden apreciar los dos pasillos,

con una pared lateral mas lineal que la otro, debido a la presencia de columnas y puertas.

Figura 3. 20. Visualización del pasillo por medio del nodo VerXY.

Otro medio empleado es que, a partir de una estructura de PCL en XYZ se toma una fila de la matriz de 640x480 para obtener una visión del plano que representa. De esta forma al variar la fila se pueden obtener diferentes planos de la imagen, sistema cliente-servidor nodos Indice-Kinav3 (sección 2.1.2.1.1.), ver figura 3.21.

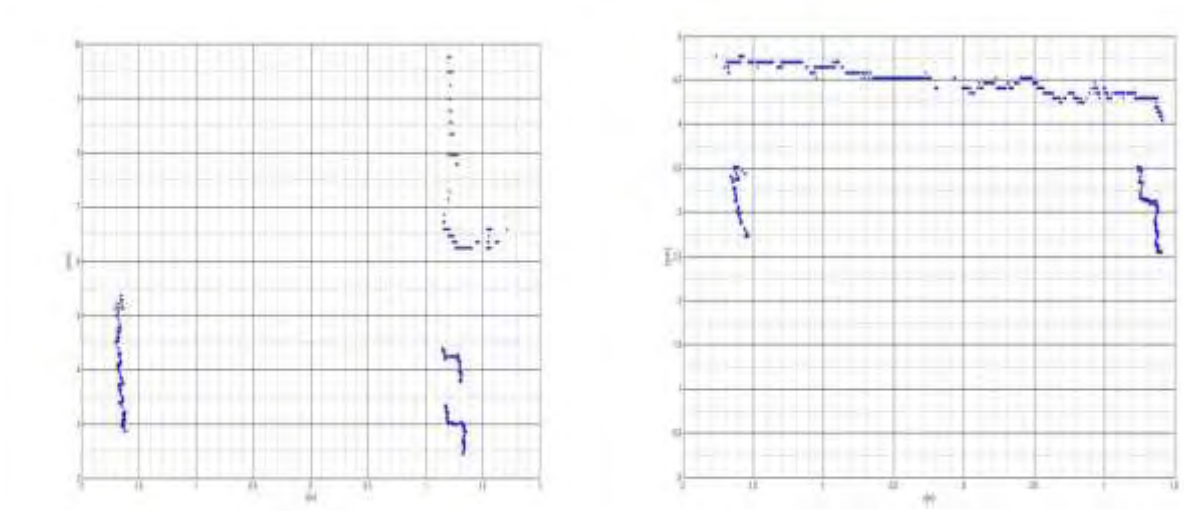

Figura 3. 21. Visualización del pasillo por medio de los nodos Índice y Kinav3.

En la figura anterior se puede apreciar que en la imagen de la izquierda se ve el sistema pasillo y en la de la derecha parte del pasillo y el techo, debido al cambio de índice, que implica un cambio de altura relativa a la distancia de los objetos al Kinect y a la fila de la matriz de puntos que se escoja.

3.2.2. SERVICIO DE DATOS PARA NAVEGACIÓN

Otro modo empleado para la visualización de un plano es la trasformación de la nube de puntos a una estructura tipo láser, nodo To_laser, con esto se logra que los datos del sensor Kinect sean manipulados como si se tratara de un *scanner laser*, ver figura 3.22.

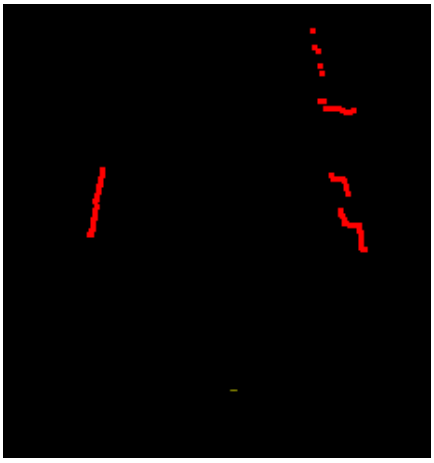

Figura 3. 22. Visualización del pasillo por medio del nodo To_Laser.

Esta estructura se puede aplicar el nodo to_splitter, para seccionar el lado izquierdo y derecho de la cámara, ver figura 3.23.

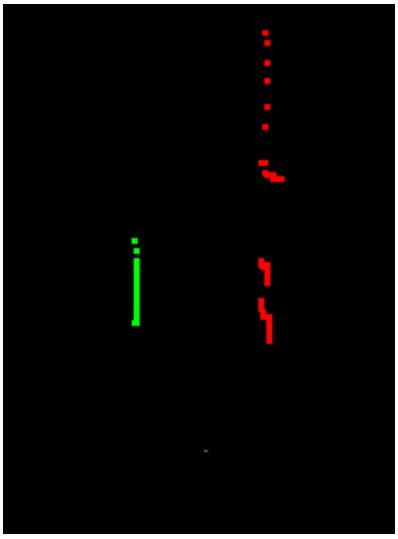

Figura 3. 23. Visualización del pasillo por medio del nodo To_Splitter.

Los mensajes que genera to_splitter se emplearon como entrada al nodo NovelScan, para generar una recta de aproximación, de esta manera al visualizar una pared (figura 3.17), se obtuvo el siguiente resultado, figura 3.24.

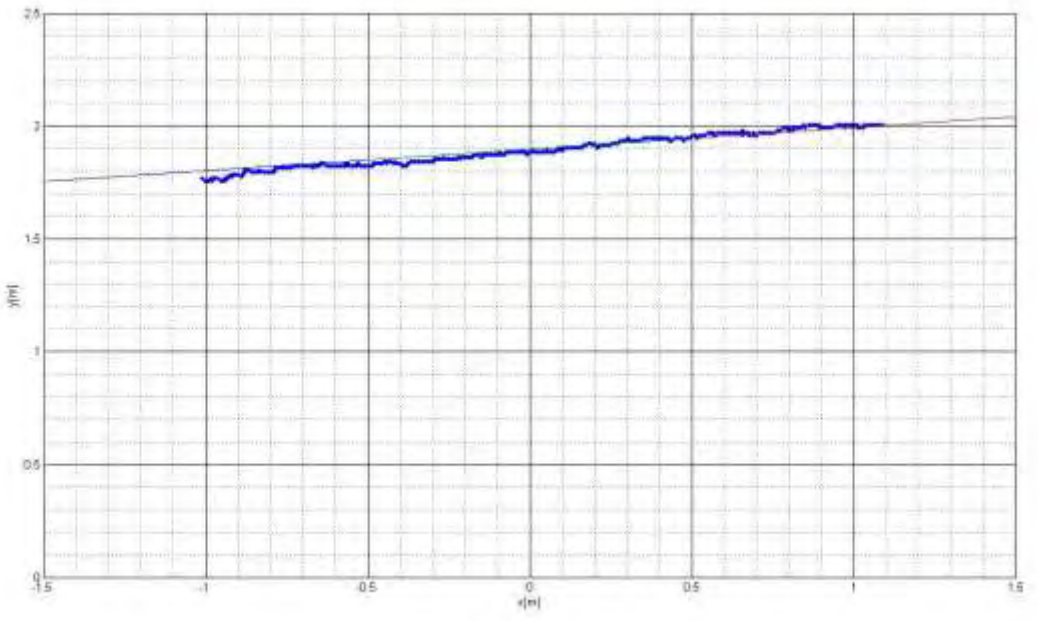

Figura 3. 24. Creación de una recta a partir de una secuencia de puntos.

Se obtuvo una recta de la forma "Y = 0,095546X + 1,897504", con lo que al evaluar en el punto X = 0, se obtiene que la separación del Kinect respecto a la pared es de 1,89 metros, con un error de 0,52% de la distancia real. Para la aproximación se emplearon 118 muestras y el método arrojó un factor de regresión igual a 1. Cabe destacar que la gráfica se encuentra representada por una nube de puntos proporcionada por Kinav3 y que la recta fue calculada por el mensaje de to_splitter, con lo que se establece la factibilidad de usar tanto el formato de PointCloud2 - PCL como el de LaserScan.

Dado que la determinación de rectas es la base de la navegación del ICR, se puede prever que si se ejecuta seguimiento de trayectoria a una recta determinada con el procedimiento anterior se provocará el choque del móvil con una pared, por esto se determina una recta paralela a la real, para evitar colisiones. En este sentido, aplicando la misma metodología se procedió a calcular una recta paralela a la secuencia de puntos, obteniendo como resultado una recta de la forma "Y=0,104309X + 1,566499". Esta se hace de modo que quede en el espacio interno entre el Kinect y la pared, ver figura 3.25.

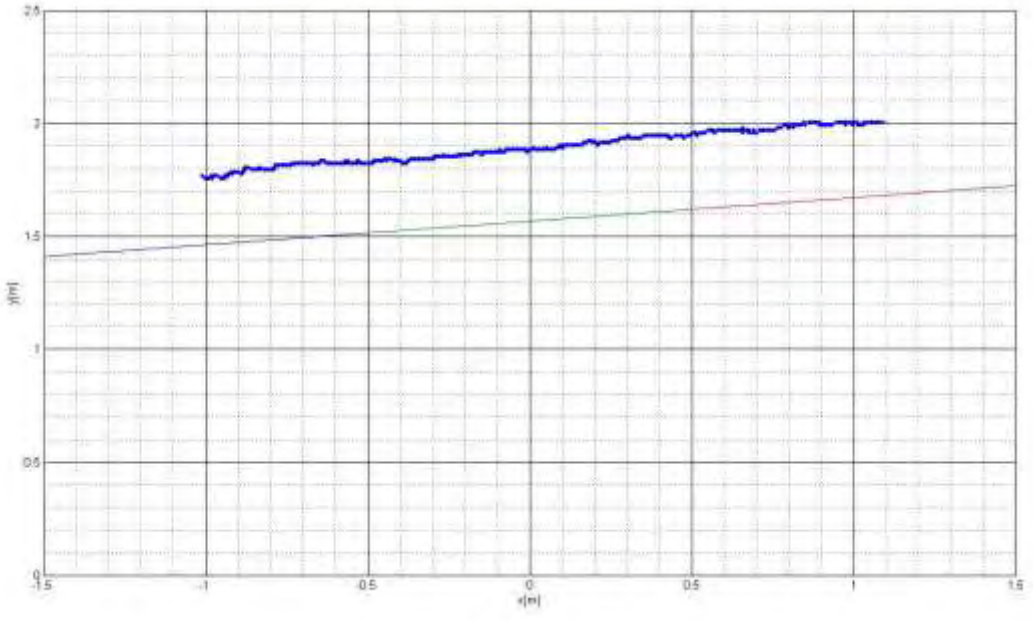

Figura 3. 25. Creación de una recta paralela a una secuencia de puntos.

Respecto al pasillo, se calculó una recta que se aproxime al sistema, y el resultado fue una expresión que pasa prácticamente por la mitad de las paredes, ver figura 3.26, con una ecuación: Y = 0,024056, con 52 muestras. A partir de la imagen se puede constatar que la separación de las paredes es de aproximadamente 2,8m, esto hace constar que las mediciones hechas por el sensor Kinect y por ende por los nodos Kinav3 y NovelScan son acertadas con la realidad.

Por último, se halló una recta paralela en el sistema pasillo, y se obtuvo como resultado una expresión de la forma "Y = 0,372376", ver figura 3.27.

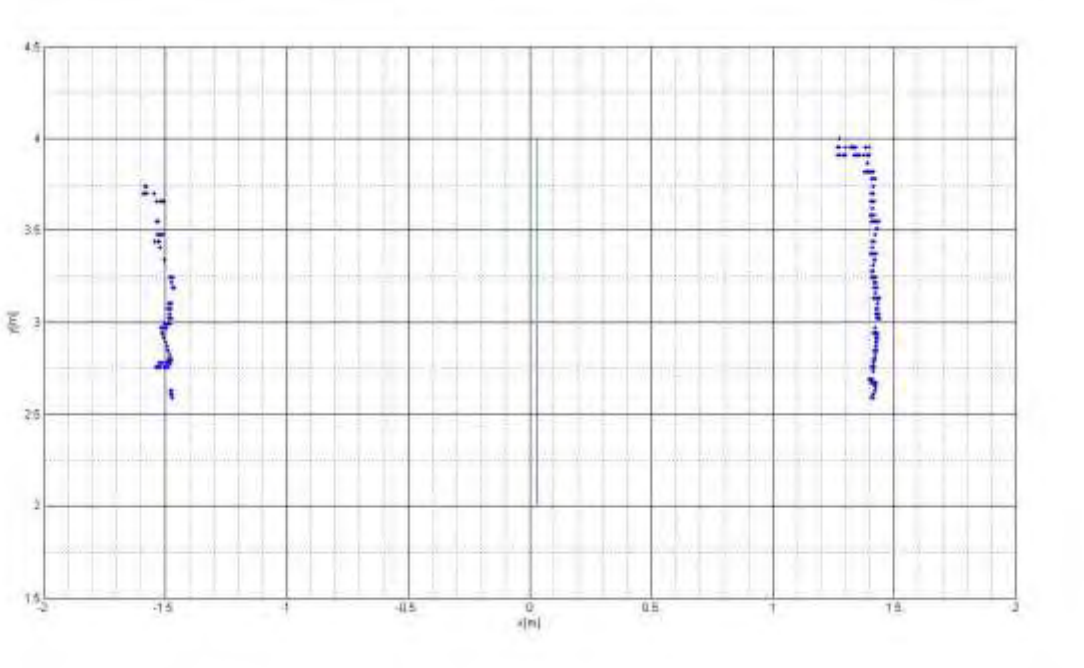

Figura 3. 26. Creación de una recta en el sistema pasillo.

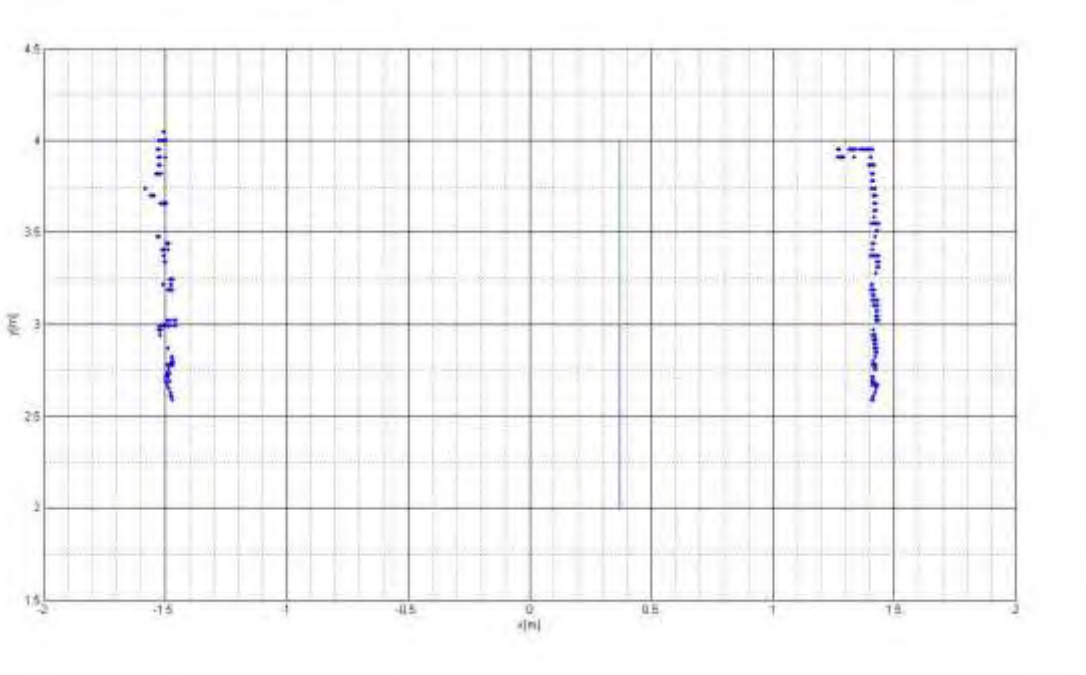

Figura 3. 27. Creación de una recta paralela en el sistema pasillo.

Ahora**,** se establece el uso de los nodos Vecref y Vecref2, para establecer la relación del Kinect al ICR, ya que estos están encargados de tomar una recta generada por la data del sensor Kinect y transformarla en un comando de velocidad para el móvil.

La primera prueba se basa en que a partir de una recta generada se construye una trayectoria de cinco metros y una vez esta sea utilizada se hace una nueva solicitud de recta, para construir otra sección del recorrido, Esto se puede apreciar en la siguiente imagen (figura 3.28).

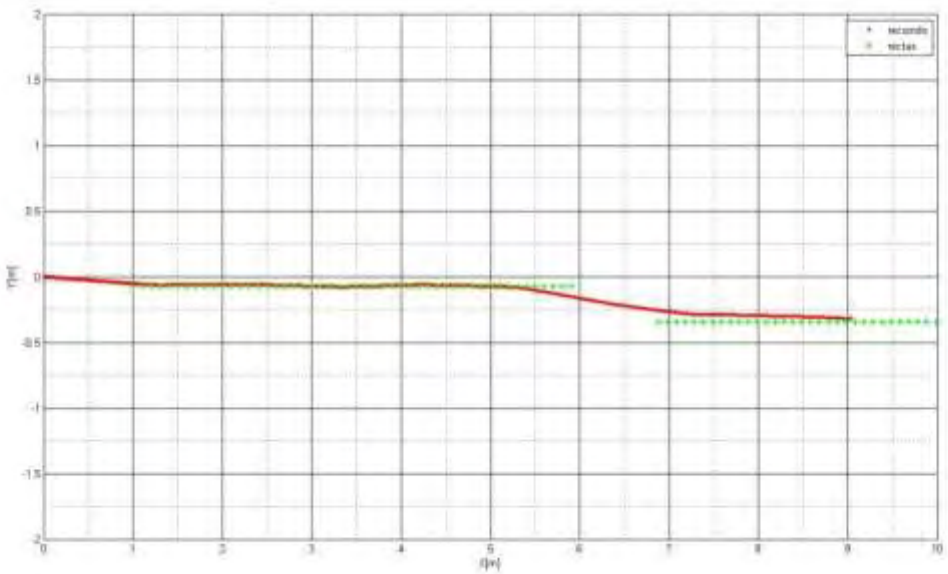

Figura 3. 28. Recorrido de pasillo con el uso del nodo Vecref.

Respecto al control que se genera, se puede apreciar en la figura 3.29 que el ángulo que se envía como referencia se encuentra entre los cero radianes y su equivalente en vuelta completa 2π radianes, lo que señala la tentativa de quedarse en el medio del pasillo.

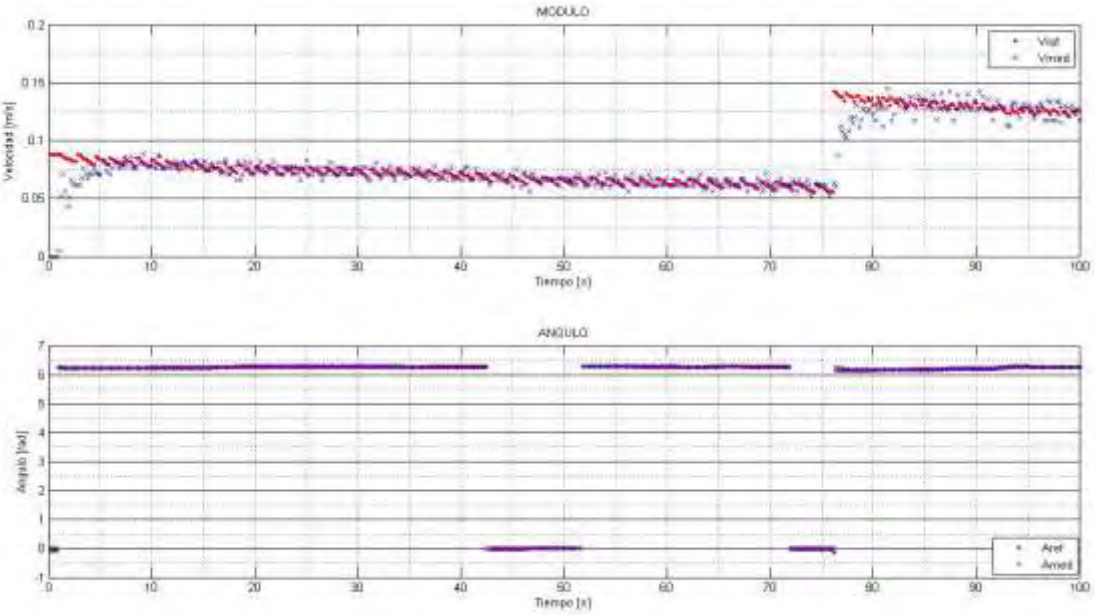

Figura 3. 29. Control de velocidad en el sistema pasillo.

Sin embargo, por tratarse de un entorno real pueden presentarse problemas en el desarrollo de la cinemática, el control del móvil y la odometría ya que las condiciones de operación no se cumplen totalmente (sección 1.1), como por ejemplo que la superficie no presente irregularidades. Se puede apreciar en la siguiente imagen (figura 3.30) que hay un fallo en la orientación de la segunda recta, no queda paralela a su anterior, esto se debe a que una vez generada la primera recta, la generación de vectores de referencia queda a cargo de la odometría, por lo que se produce acumulación de errores durante cinco metros de recorrido. Sin embargo, con la nueva recta el recorrido queda establecido por la posición en la que se cree estar, y la misma recta queda orientada de esta manera por lo que el recorrido se hace de manera acertada a pesar de poseer de los errores en el cálculo de la posición real.

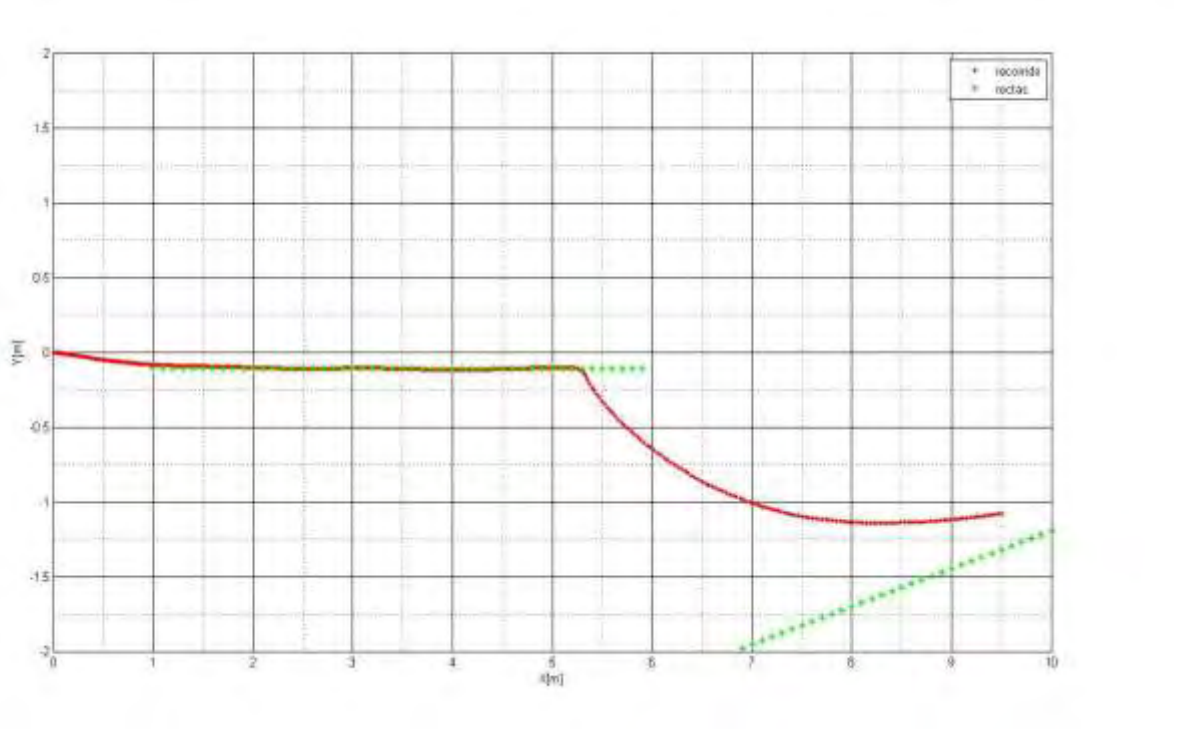

Figura 3. 30. Error en el recorrido de pasillo con el uso del nodo Vecref.

En el control (figura 3.31) se puede apreciar un cambio de ángulo alrededor del segundo 75, que es precisamente la corrección de la orientación dada por la nueva recta. Este efecto es mucho menor en la experiencia anterior.

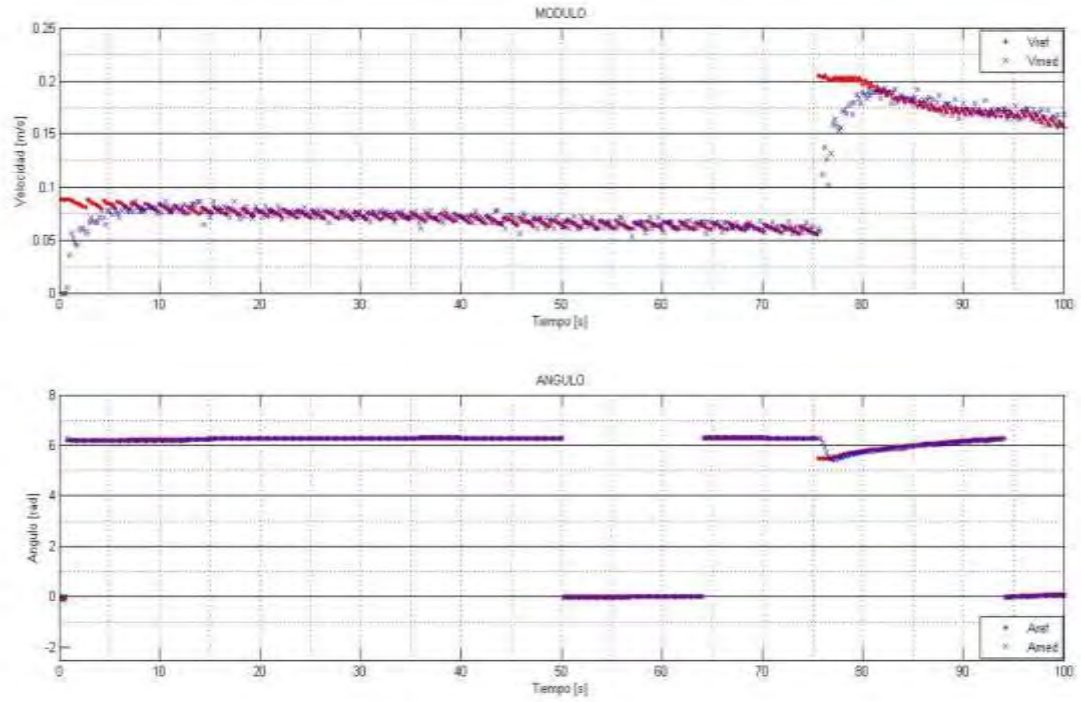

Figura 3. 31. Control de velocidad en el sistema pasillo, con corrección de orientación.

La segunda prueba se basa en el cálculo de un vector de velocidad a partir de la ecuación de una sola recta, es decir, por cada recta determinada se genera un vector. Con esto se le quita el control que ejerce la odometría en el cálculo del vector de referencia y se le pasa a la generación de rectas, en un control a lazo abierto.

En este sentido, se presentan dos casos, uno donde la odometría no presenta errores por lo que el cálculo de vectores de referencia es constante y otro donde la odometría posee fallos por lo que la secuencia de vectores de referencia varía para corregir estos errores.

El primer caso muestra un seguimiento de trayectoria por el medio del pasillo (figura 3.32) y con una secuencia de vectores en el tiempo de mismo módulo y ángulo, ver figura 3.33.

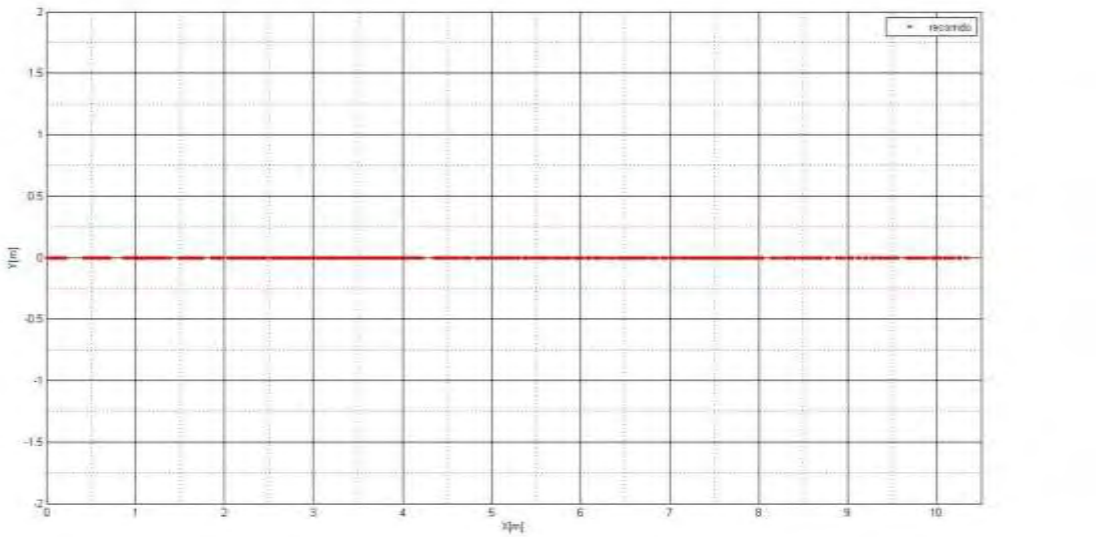

Figura 3. 32. Desplazamiento producido en el recorrido de un pasillo, caso 1.

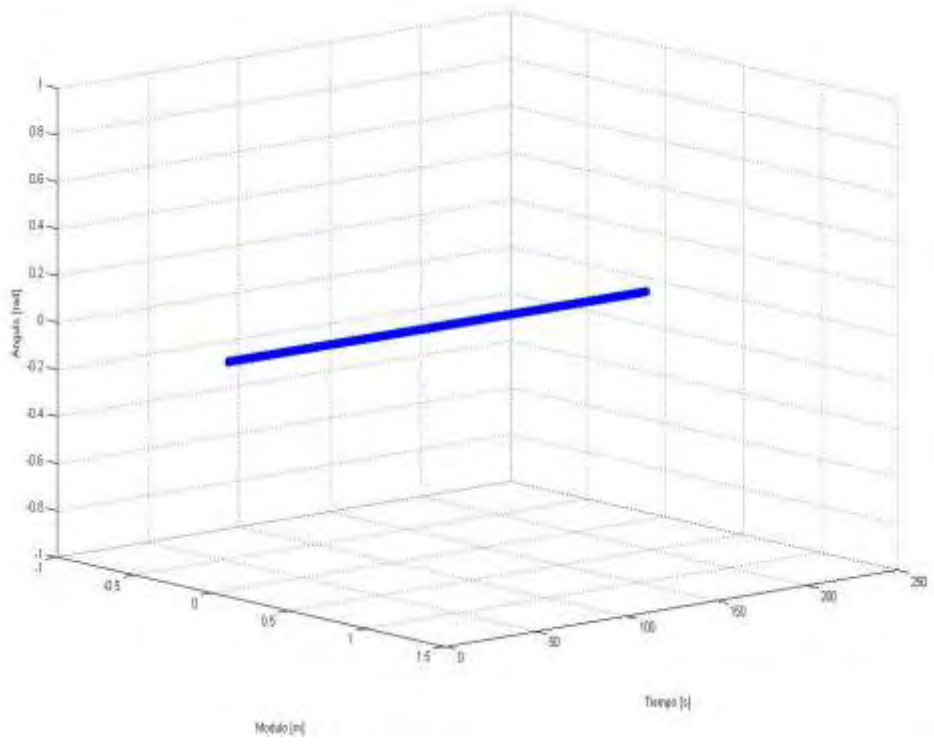

Figura 3. 33. Vectores de referencia producidos en el recorrido de un pasillo, caso 1.

El segundo caso, muestra igual que el anterior un seguimiento de trayectoria por el medio del pasillo (figura 3.34), pero para poder realizarlo tuvo que realizar ajustes por medio de los vectores de referencia como se puede ver en la figura 3.35.

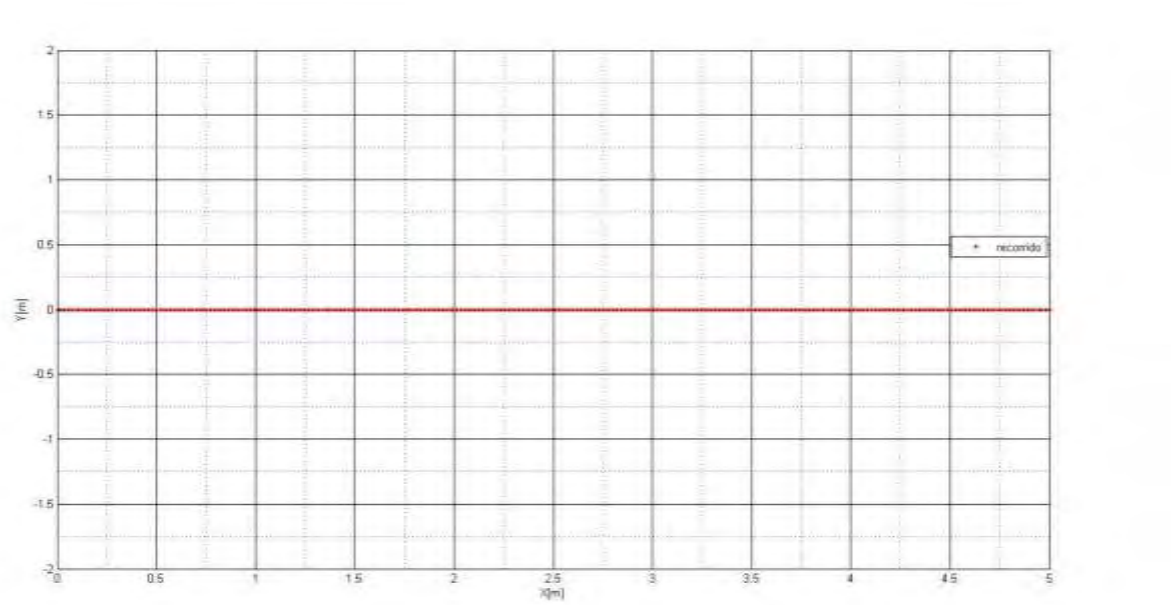

Figura 3. 34. Desplazamiento producido en el recorrido de un pasillo, caso 2.

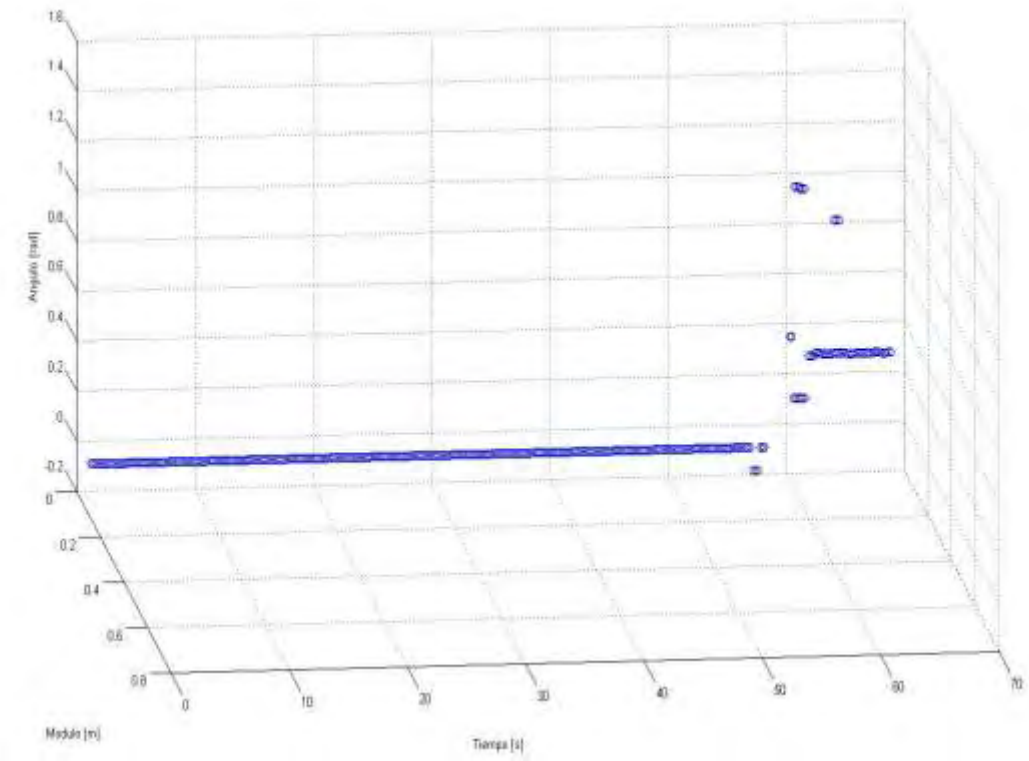

Figura 3. 35. Vectores de referencia producidos en el recorrido de un pasillo, caso 2.

3.3. EXPORTACIÓN DE LA INFORMACIÓN

El uso de múltiples maquinas para realizar tareas de monitoreo, captura de datos y operación es de provecho en el entorno de ROS ya que permite aumentar el poder de computo al contar con más de una instancia de procesamiento. De esta manera, se tiene que dos computadoras enlazadas pueden compartir y manipular información, como se ve en la siguiente imagen (figura 3.36). En donde el nodo de Kinect es proporcionado por una de las instancias de cómputo y el nodo de Rviz es proporcionado por la otra.

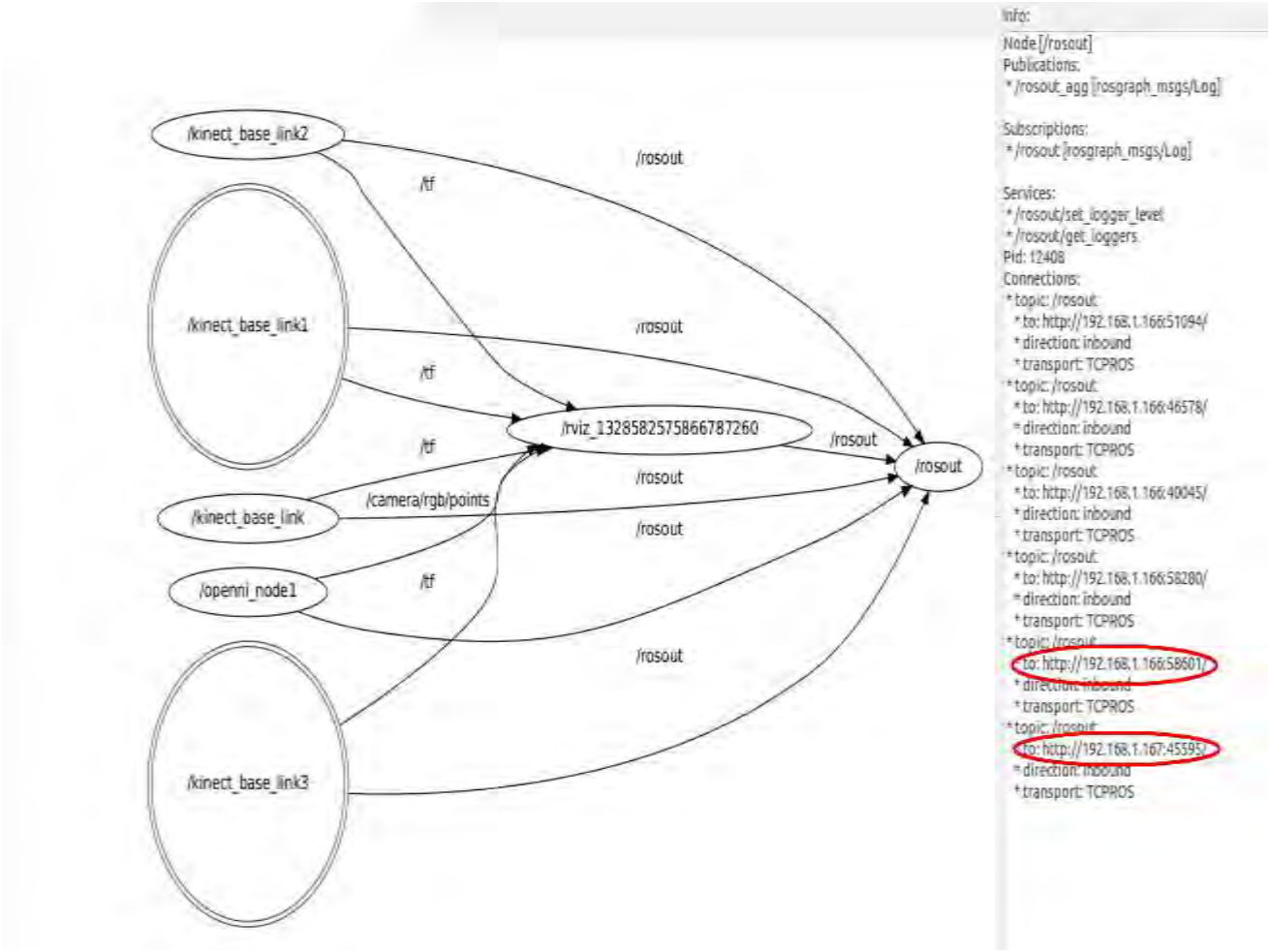

Figura 3. 36. RXGRAPH de ROS en configuración de múltiples maquinas.

Para finalizar, como validación extra de la aplicabilidad del sistema desarrollado, los datos adquiridos usando el sensor Kinect en el segundo espacio de trabajo, fueron exportados a Matlab® donde se les aplicó un algoritmo implementado en la tesis de maestría del Prof. Novel Certad (trabajo en curso, pendiente de publicación, Universidad Simón Bolívar) para hacer detección de segmentos, esquinas y flancos. En la figura 3.37, se muestran los puntos obtenidos de la adquisición. Los puntos marcados con 'x' verde corresponden a esquinas detectadas, las 'x' rojas corresponden a flancos. Los puntos marcados en azul son detectados como parte de un segmento, y los puntos marcados en rojo son puntos que corresponden a rupturas o discontinuidades en la data (dado que la distancia que los separa es muy grande) o corresponden a datos que se encuentran fuera del rango de medición del Kinect por lo que corresponden a puntos inválidos. La línea sólida negra representa un aproximación de la forma real del pasillo, hecha con mediciones tomadas a mano en el sitio, en la figura se aprecia un fuerte error de calibración en el sensor, por lo cual se recomienda diseñar un protocolo de calibración en futuros trabajos a desarrollar.

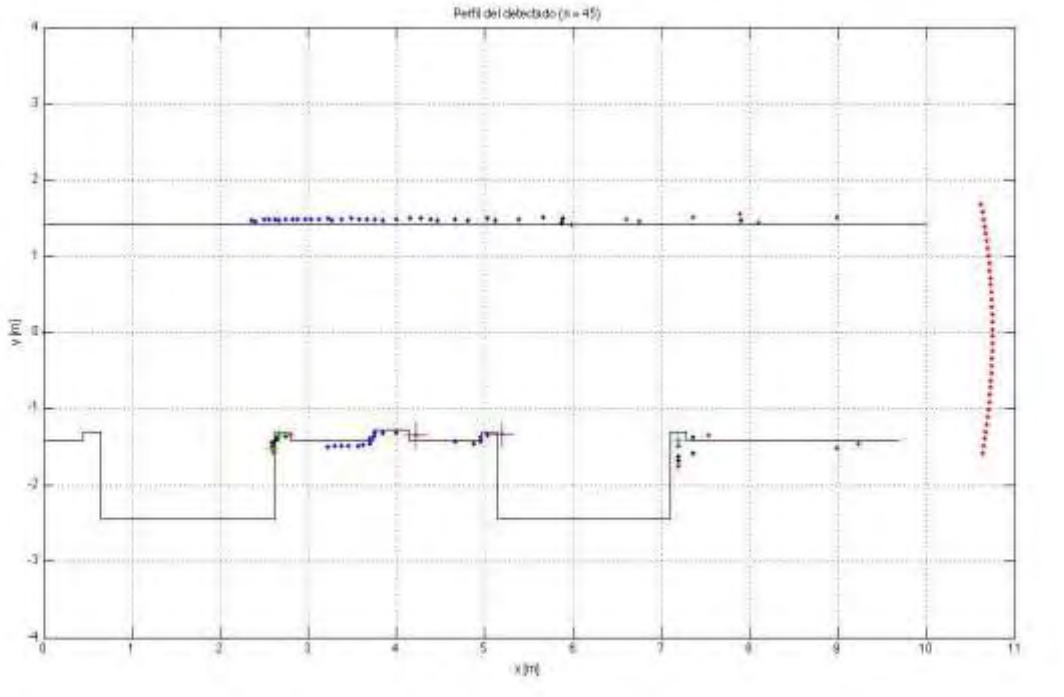

Figura 3. 37. Detección de bordes y perfil de curva estimada.

CONCLUSIONES

El objetivo principal del presente trabajo fue desarrollar un sistema de control utilizando ROS, el cual debía permitir el control del robot Roomba por medio de referencias vectoriales de velocidad, e integrar al sensor Kinect. Para ello se dividió el trabajo en varias etapas y a continuación se presentan las conclusiones en consecuencia al desarrollo que se presentó.

a) El uso de ROS como plataforma para el desarrollo de los diferentes programas resultó una decisión acertada, dada la versatilidad de sus cualidades y características. Además, ROS cuenta con una extensa gama de paquetes en diferentes áreas y con diversos desarrolladores lo que hace que se tengan propuestas y soluciones diferentes para problemas similares, otorgándole diversidad a su entorno.

b) Con el uso de la capa de comunicación de ROS es posible tener nodos ejecutándose en paralelo e interactuando entre sí. Por lo que, se puede dividir el problema en módulos y realizar tareas particulares de una forma práctica y ordenada. De la misma manera, la implementación de medios de suscripción y publicación de mensajes hace viable el uso y la estandarización de los programas. Por lo que, la manipulación y comprensión de paquetes hechos por terceros es posible siempre y cuando se tenga conocimiento general de las intenciones del autor.

c) La herramienta asociada a almacenar información, "rosbag", permite no solo poseer los datos de determinado tópico, sino reproducir las condiciones en las que este se desarrolló, por lo que al realizar la grabación de una experiencia, esta se podrá reproducir virtualmente de forma parcial o total, teniendo sin ningún problema mensajes de una grabación y mensajes nuevos coexistiendo, sin ningún inconveniente.

d) La configuración en múltiples máquinas permite exportar la información a una instancia remota al igual que interactuar con el servidor local, pudiendo hacer monitoreo, control o simplemente tareas de almacenaje. La principal ventaja es poder sumar capacidad de cómputo a la experiencia que se esté llevando a cabo.

e) El uso de control por seguimiento de vectores de velocidad hace que el robot se comporte como un sistema cuya entrada es una referencia, que puede ser dada por cualquier programa, siempre y cuando respete el formato del mensaje. En este sentido se logra compatibilidad, demostrado con el uso del nodo "Control" que podía unirse tanto con el nodo de teleoperación, el de seguimiento de una trayectoria planificada y libre de obstáculos o con el nodo "Vecref" y "Vecref2" que proporcionaban vectores de referencia a partir de los datos del Kinect.

f) Respeto al sistema Kinect, computadora e Irobot Create® se puede decir, en un primer lugar, que se logró construir una plataforma para el alojamiento de los dos primeros elementos en el carro, además de un cable para suministrar voltaje de alimentación al Kinect por parte del ICR y la conexión USB entre el Kinect y la computadora. Con esto, no se altero el cable original del sensor, permitiendo que este sea usado en trabajos futuros y aplicaciones diferentes al móvil Roomba. Así mismo la estructura de acople no altero el diseño del ICR.

g) El servicio de datos del sistema integrado, puede servir para análisis on line u off line. Esto, gracias a los mecanismos de rosbag, múltiples maquinas y las demás configuraciones de suscripción y publicación de mensajes.

h) Respecto a los datos del Kinect, se manipularon en dos formatos: PCL y LaserScan. Para el uso de PCL como el de LaserScan se tuvo que pasar del mensaje tipo PointCloud (estándar de ROS) a un tipo estándar de PCL (PointType), o realizar una conversión al mensaje estándar LaserScan. Esto, permitió trabajar con la nube de puntos completa o con características precisas, para: realizar visualización de la data adquirida, realizar procesos de filtrado y segmentación. La decisión de usar un formato u otro depende de la aplicación que se vaya a realizar, por lo menos el segundo formato permite apreciar la data del Kinect como si se tratara de un láser *Range Finder* (telémetro) y obtener un plano del entorno sin necesidad de más procesamiento. Debido a que en trabajos presentes del Grupo de Mecatrónica de la Universidad Simón Bolívar se encuentran simulaciones con el uso del sensor *Range Finder*, el manipular la data del Kinect con este formato permitió un enlace entre ambos trabajos, por compatibilidad en los tipos de datos. En este sentido, se exportó un mensaje en formato LaserScan para realizar detección de segmentos, esquinas y flancos para el sistema pasillo, Esto demuestra la flexibilidad que se posee al momento de manipular la información, con el uso de ROS.

i) Se alcanzó controlar el móvil a partir de referencias vectoriales, suministradas por el procesamiento de información del entorno. Es así, que tanto con el uso del nodo "Vecref" y "Vecref2", en cooperación con los demás programas esenciales para el funcionamiento del sistema (driver del ICR, del Kinect, control, NovelScan) se consiguió que el ICR pudiera navegar en un pasillo. Los resultados arrojaron que es preferible usar como control la generación continúa de vectores de velocidad con la data del Kinect y no generar una recta a partir de un dato y cederle el control a la odometría, que por razones de acumulación de errores en la medición y el no poseer las condiciones ideales del medio no lo hace confiable.

RECOMENDACIONES

a) Generar el *stack* de la Universidad Simón Bolívar y el paquete relacionado al presente trabajo. Con esto, se podrá compartir con otros desarrolladores las modificaciones hechas a los paquetes aquí empleados y los programas nuevos empleados. De esta manera, se podría tener una crítica externa y además la posibilidad de establecer contactos de cooperación.

b) Monitorear las actualizaciones y las adiciones que se presentan en ROS, ya que se encuentra en continuo crecimiento.

c) El uso de ROS como plataforma de desarrollo para trabajos futuros, ya que permite estandarizar los procedimientos, los programas y así obtener una mejor reutilización de lo ya existente.

d) Al configurar un nodo se recomienda el uso de mensajes estándares de ROS, ya que desvincula la necesidad de poseer el paquete que genera el nodo y el hecho de incluir nuevas dependencias en programas de otros paquetes.

e) Respecto al método de sintonización de los controladores se recomienda revisar y chequear los diferentes valores de ganancias, si se replican las experiencias en otras plataformas hermanas, ya que se pueden presentar variaciones en la escogencia de los parámetros, por las mediciones obtenidas o por las características propias de cada móvil.

REFERENCIAS

[1] Hernandez G, Silva O. “Una panorámica de los robots móviles”, Telematique volumen 6- edición 3, 2007.

[2] Muir P, Neuman C. “Kinematic modeling of wheeled mobile robots”, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-86-12, 1986.

[3] Bambino I. “Una introducción a los robots móviles”, 2008.

[4] Berenguel M, Dormido S, Guzman J. “Herramienta interactiva para robótica móvil”, Universidad de Almeria, España, 2003.

[5] Alvarez L, Figueroa J. “Implementación de algoritmos de navegación utilizando la plataforma del irobot create y módulos de comunicación inalámbrica xbee”, Escuela Politécnica del Ejercito, Ecuador, 2011.

[6] Guy S, Manocha D, Snape J, Van den Berg J. “Smooth Coordination and Navigation for Multiple Differential-Drive Robots”, University of North Carolina at Chapel Hill, USA, 2010.

[7] Serrano L. “Virtual Blacboar: colour and human gestures motion tracking”, Escuela Politécnica de Cáceres, España. 2011.

[8] Aritz L. " Sistema de localización y seguimiento de personas en interiores mediante camara PTZ basado en las tecnologías Kinect y Ubisense", Universidad del País Vasco, 2011.

[9] Tanenbaum A. "Sistemas operativos modernos". Segunda edición. Prentice-Hall Pearson, Mexico, 2003.

[10] Berger E, Conley K, Faust J, Foote T, Gerkey B, Leibs J, Ng A, Quigley M, Wheeler R. "ROS: an open-source Robot Operating System", Stanford University, USA, 2009.

[11] Página web oficial de PCL, http://www.pointclouds.org, consultado el 19 de Septiembre de 2010.

[12] Franco A. " Regresión Lineal", disponible en internet: http://www.sc.ehu.es/sbweb/fisica/cursoJava/numerico/regresion/regresion.htm. Consultado el 16 de Enero de 2012.

[13] Deyle T. "The Bilibot Project: A Low-Cost Robot Platform with iRobot Create, Kinect, and ROSified Computer". Disponible en internet: http://www.hizook.com/blog/2011/02/10/bilibot-project-low-cost-robot-platform-irobot-create-kinect-and-rosified-computer. Consultado el 17 de Octubre de 2011.

[14] Página web oficial de ROS, http://www.ros.org, consultado el 25 de Abril de 2010.

[15] Crick C, Jay G, Jenkins O, Osentoski S. "Brown ROS Package: Reproducibility for Shared Experimentation and Learning from Demonstration", Brown University Providence, USA, 2010.

[16] Medina W, Fermín L, Cappelletto J, Murrugarra C, Fernandez G, Grieco J. "Vision-Based Dynamic Velocity Field Generation for Mobile Robots". LNCIS Robot Motion and Control, vol. 360, 2007.

[17] D´Arpino C. "Control de Velocidad de un Robot Móvil asistido por Visión Artificial", Universidad Simón Bolívar, Venezuela, 2008.

[18] Estevez C. “Generación dinámica de campos de velocidades para navegación coordinada”, Universidad Simon Bolivar, Venezuela, 2007.

[19] Bueno M, Ríos L. "MODELO MATEMÁTICO PARA UN ROBOT MÓVIL", Scientia Et Technica, Vol. XIV, Núm. 38, junio-sin mes, 2008, pp. 13-18 Universidad Tecnológica de Pereira, Colombia.

[20] Ogata K. "Ingeniería de Control Moderno". Tercera edición. Prentice-Hall Pearson, Mexico 1998.

[21] Medina W. "Generación Dinámica de Campos de Velocidades Usando Visión Artificial". Universidad Simón Bolívar, Venezuela, 2007.

[22] Página web oficial de Ubuntu, http://www.ubuntu.com, consultado el 25 de Abril de 2010.

[23] Certad N. “Diseño de algoritmo para generación de mapas con localización simultanea en ambientes estructurados para un vehículo autónomo”, Universidad Simón Bolívar, pendiente de publicación. Venezuela, 2012.