Implementación de sensores 3D para la navegación de robots móviles y vehículos autónomos

3D sensors implementation for mobile robots and autonomous vehicles navigation

Sergio Valverde-Moreno[1]

Valverde-Montero, S. Implementación de sensores 3D para la navegación de robots móviles y vehículos autónomos. *Tecnología en Marcha*. Vol. 33, especial Movilidad estudiantil. Pág 176-186.

https://doi.org/10.18845/tm.v33i7.5492

1 Ingeniero Mecatrónico. Área Académica Mecatrónica, Instituto Tecnológico de Costa Rica. Costa Rica. Correo electrónico: sergiovalverdem@hotmail.com

Palabras clave

Navegación autónoma; Robótica móvil; Pioneer P3-DX; Kinect, Campos potenciales; Point Clouds; Waypoints.

Resumen

La navegación autónoma de vehículos, así como el mapeo de zonas tiene un sinfín de aplicaciones comerciales como industriales. Este articulo explica el desarrollo de una aplicación para que un robot móvil pueda navegar de forma autónoma, evitando obstáculos, hasta llegar a una coordenada específica definida por el usuario, utilizando sensores 3D de bajo costo. Para diseñar e implementar esta aplicación se debe tomar en cuenta que la navegación y detección de obstáculos se realizó con base en sensores 3D, para lo cual se seleccionó un Kinect. La aplicación se simuló y probó en un ambiente de simulación especial para aplicaciones robóticas, llamado V-REP. El algoritmo de navegación para evitar obstáculos se implementó por medio de la teoría de campos potenciales. Otros sensores, como un GPS y un IMU, fueron utilizados para determinar la posición y orientación del robot en el espacio, con el objetivo de definir una trayectoria hasta la coordenada meta y así hacer la navegación más eficiente. Por último, se trabajó con los datos reales de los sensores en físico para corroborar el funcionamiento de la aplicación y así compararlo con los resultados obtenidos anteriormente de la parte simulada.

Keywords

Autonomous navigation; Mobile robotics; Pioneer P3-DX; Kinect; Potential fields; Point Clouds; Waypoints.

Abstract

Autonomous vehicles navigation, as well as mapping areas have endless commercial and industrial applications. This article explains the development of an application that allows a mobile robot to navigate autonomously, avoiding obstacles, until it reaches a specific coordinate point defined by the user, using low cost 3D sensors. For the design and implementation of this application, it is necessary to keep in mind that navigation and obstacle detection are based on the use of 3D sensors. For this purpose, the Kinect was selected. The simulation and testing of the project was done in a special simulation environment for robotic applications, called V-REP. The navigation algorithm for obstacle avoidance was implemented through the potential fields theory. Other sensors, as GPS and IMU, were used to determine the robot's spatial position and orientation, with the objective of defining a trajectory to the goal coordinate point, thus doing a more efficient navigation. Finally, real data from the physical sensors was used to corroborate the functioning of the application, with the purpose of comparing it with the previously obtained results from the simulated part.

Introducción

La navegación autónoma de vehículos, así como el mapeo de zonas tiene un sinfín de aplicaciones. En zonas de riesgo, como por ejemplo donde han ocurrido desastres naturales, estas aplicaciones toman aún más relevancia. En estas situaciones es muy peligroso enviar a una persona a realizar tareas de exploración ya que puede resultar herida o en el peor de los casos perder la vida. Es aquí, donde un vehículo o robot autónomo, que pueda llevar a cabo estas tareas es una excelente solución.

Por otro lado, este tipo de vehículos también tienen aplicaciones muy importantes a nivel comercial e industrial. Por ejemplo, en una industria, un vehículo capaz de llegar a una coordenada específica, con la capacidad de evitar obstáculos y evitar colisiones con objetos o personas, sería de gran provecho ya que se pueden mejorar los procesos productivos y la eficiencia de las fábricas.

En el sector comercial, se pueden realizar aplicaciones para facilitar la vida cotidiana. Un robot autónomo con las características descritas anteriormente se puede utilizar para seguir a una persona con el fin de cargarle sus pertenencias. Por ejemplo, en un aeropuerto se podrían usar como un servicio de transporte de maletas para los viajeros. También se podría usar como un método para llevar las pertenencias de personas adultas mayores o de personas con alguna discapacidad, lo cual les beneficiaría considerablemente en sus tareas diarias.

Si bien es cierto existen vehículos especializados para realizar este tipo de tareas, estos poseen un alto costo y no es tan fácil tener acceso a este tipo de tecnología. Por esta razón en el siguiente artículo se presenta una aplicación basada en sensores 3D de bajo costo para habilitar la navegación autónoma de un robot o vehículo.

La aplicación está basada en el uso de un sensor 3D Kinect junto con un giroscopio y un GPS. Los datos de estos sensores se integraron en el modelo de control de un robot Pioneer P3-DX para que navegue de forma autónoma, utilizando los métodos de campos potenciales y waypoints. Los resultados iniciales se verificaron por medio del simulador V-Rep y luego se verificó la aplicación por medio del análisis de los datos reales de los sensores.

Consideraciones para navegación autónoma

El desarrollo de esta aplicación se basa en el uso de tres sensores: un Kinect, el cual permite mapear el entorno por medio de points clouds y a través de ellos detectar obstáculos en la trayectoria de navegación. Una unidad de medición inercial (IMU), la cual permite obtener información de la velocidad angular y la aceleración lineal. Y finalmente un GPS, el cual permite obtener coordenadas precisas de la ubicación del robot móvil.

La información de estos sensores se combina con técnicas de navegación como lo son campos potenciales y waypoints. La primera técnica se basa en la idea de concebir al robot como una partícula que se ve influenciada por la fuerza de campos artificiales de atracción y repulsión. El espacio por el que se desplaza la partícula es el resultado de la suma de un campo potencial de atracción, el cual posee una fuerza que jala la partícula hacia el destino o meta, y también el de un campo de repulsión que contiene fuerzas que la alejan de los obstáculos [1]. Mientras que los waypoints son una serie de puntos o coordenadas que definen una trayectoria de navegación.

Análisis de point clouds

El point cloud generado por el Kinect posee una resolución de 640 x 480 pixeles. Sin embargo, para fines de simulación en V-Rep y desarrollo del algoritmo de detección de obstáculos se trabajó con una resolución de 64 x 48 pixeles, ya que se obtienen mejores velocidades de simulación, con variaciones insignificantes en los resultados. [2]

El algoritmo de análisis del point cloud busca detectar el punto más cercano al robot. Este comienza recorriendo los 64 pixeles de resolución en el eje X del sensor y posteriormente calcula el ángulo entre este eje y cada uno de los puntos. De igual forma este proceso se repite para los 48 pixeles del eje Y. El cálculo de los ángulos con respecto a los ejes X y Y viene dado por las siguientes fórmulas:

$$xAngle=\left(\left(32\text{-}i\text{-}0.5\right)/32\right)*camXHalf\,Angle$$
$$yAngle=\left(\left(\mathrm{j}-24+0.5\right)/24\right)*camYHalf\,Angle$$

Donde i es posición de un pixel en el eje X y j la posición de un pixel en el eje Y. Las variables *camXHalfAngle* y *camYHalfAngle* corresponde a la mitad del ángulo de visión en cada uno de los ejes. Dicha información se puede ver con detalle en la figura 1.

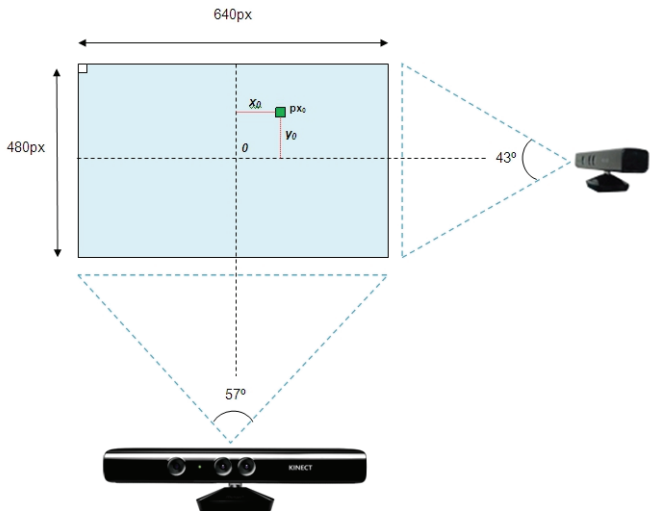

Figura 1. Características del sensor Kinect. Recuperado de: [3]

De igual forma para cada uno de los elementos del point cloud se calcula su profundidad. Estos valores se almacenan en un vector de 3072 elementos (64x48) llamado *depthBuffer*. Estos valores de profundidad se encuentran entre 0 y 1 y vienen dados por la siguiente fórmula.

$$depthValue=depthBuffert\left[i+\left(j\text{-}1\right)*64\right]$$

Luego la variable *depthValue* se multiplica por la constante de la máxima distancia de profundidad que se puede medir y se le suma el "NearClippingPlane", que es la menor distancia a partir de la cual puede medir el Kinect, esto con el fin de obtener el valor de la coordenada en el eje Z del punto i,j. [2]

Una vez con la coordenada en Z (valor de profundidad), se puede obtener por medio de trigonometría las coordenadas en X y en Y, ya que anteriormente se habían calculado las variables *yAngle* y *xAngle*. Dichos cálculos se pueden ver a continuación:

$$xCoord=math.tan\left(xAngle\right)*zCoord$$
$$yCoord=math.tan\left(yAngle\right)*zCoord$$

Finalmente se obtiene la distancia euclídea del punto i,j del point cloud con las coordenadas tridemensionales *xCoord, yCoord* y *zCoord* y se almacena en una variable llamada *dist*.

El siguiente paso es determinar si esa variable *dist* es la distancia más cercana al sensor del Kinect de todos los puntos del point cloud. Para ello se realiza una simple comparación, en caso

de el valor actual almacenado en *dist* sea menor que el valor anterior, se actualiza la distancia más cercana y se procede a almacenar las coordenadas tridimensionales de dicho punto.

Una vez que se cuenta con la información del elemento del point cloud más cercano al sensor, se utilizan sus coordenadas para la implementación de los campos potenciales de repulsión que serán generados por los obstáculos.

Implementación de campos potenciales

De acuerdo al acercamiento de campos potenciales expuesto en [6], donde se define el gradiente del campo potencial en función del vector de posición de la partícula., se obtienen las funciones del gradiente del campo de atracción en función de distintos radios de influencia:

$$\Delta x = \begin{cases} 0, & d<r \\ \alpha(d-r)\cos\theta, & r\le d\le(s+r) \\ \alpha*s*\cos\theta, & (s+r)<d \end{cases}$$

$$\Delta y = \begin{cases} 0, & d<r \\ \alpha(d-r)\,\mathrm{sen}\,\theta, & r\le d\le(s+r) \\ \alpha*s*\mathrm{sen}\,\theta, & (s+r)<d \end{cases}$$

Las fórmulas anteriores dan como resultado los componentes X, Y, del gradiente del campo de atracción. Donde *d* es la distancia euclídea entre el robot y la meta, la cual se calcula a partir delos datos generados por el GPS. La variable *r* es la distancia con respecto a la meta, a la cual se desea que la influencia del campo potencial sea nula o deje de actuar. Se puede ver como la distancia de la meta a la que se desea que el robot se detenga.

La variable ***s*** es la distancia sobre la cual se extiende el campo de atracción. La variable es una constante con la cual se puede ajustar la magnitud del campo. La variable θ es el ángulo existente entre la posición del robot y la posición de la meta, con respecto al marco universal, la cual viene dado por la unidad de medición inercial.

Basado en la misma idea para obtener los campos potenciales de atracción, se obtienen los de repulsión que están descritos por las siguientes fórmulas [4]:

$$\Delta x = \begin{cases} -\mathrm{sign}(\cos\theta)\gamma, & d<r \\ -\beta(s+r-d)\cos\theta, & r\le d\le(s+r) \\ 0, & (s+r)<d \end{cases}$$

$$\Delta y = \begin{cases} -\mathrm{sign}(\mathrm{sen}\,\theta)\gamma, & d<r \\ -\beta(s+r-d)\,\mathrm{sen}\,\theta, & r\le d\le(s+r) \\ 0, & (s+r)<d \end{cases}$$

Se puede observar que para este caso se mantienen las mismas condiciones de los intervalos. La diferencia es que ahora la variable ***d***, denota la distancia euclídea entre el robot y el obstáculo

la cual viene dada por el análisis del point cloud generado por el Kinect. La variable ***r*** denota la distancia dentro de la cual se presenta la mayor influencia del campo de repulsión y la variable ***s*** ahora representa la distancia sobre la cual se extiende el campo de repulsión.

Se puede observar que se tienen dos nuevas variables β y γ. La primera viene a cumplir la función que anteriormente cumplía , con la diferencia de que este ángulo viene dado por los cálculos previamente explicados del Kinect, y la segunda es una constante para definir el máximo valor que se desea que posea el gradiente.

Cuando se acaban de calcular los gradientes de ambos campos, los componentes se suman para obtener el vector final que da la dirección de la trayectoria que debe seguir el robot para evitar obstáculos y llegar a la meta. Este vector también da una velocidad proporcional a la que debe navegar el robot de acuerdo a las distancias a las que se encuentre de la meta y los objetos. Esta velocidad se calcula por medio de la norma de dicha vector, la cual corresponde a la raíz cuadrada de la suma de cada uno de sus componentes al cuadrado.

Desarrollo del modelo de control de velocidad y ángulo resultante de la trayectoria

Para obtener el ángulo final con respecto al cual debe girar el robot, se debe tener en cuenta la diferencia entre el ángulo que está girado el robot, dado por la unidad de medición inercial y el ángulo resultante de la trayectoria, dado por los campos potenciales, todo esto referenciado con respecto al marco universal. Esto se hace con el fin de que el robot tenga que girar el menor ángulo posible y determinar si debe girar en sentido horario o anti horario y que así la navegación sea lo más eficiente posible.

Una vez que se conoce el ángulo resultante de la trayectoria y el sentido de giro se procede a desarrollar el control de velocidad para el robot Pioneer P3-DX. Para ello se utiliza como base el modelo cinemático de un robot de tracción diferencial de dos ruedas. El esquema que lo describe se muestra en la figura de abajo.

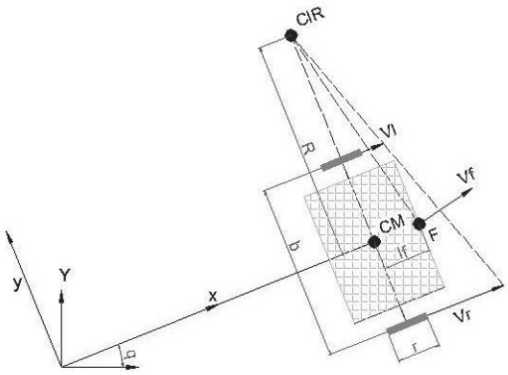

Figura 2. Modelo cinemático de un robot móvil de tracción diferencial de dos ruedas. [4]

Las ecuaciones de velocidad lineal y angular que describen este modelo se presentan a continuación [5]:

$$\nu = \frac{\nu r + \nu l}{2}$$
$$\omega = \frac{\nu r - \nu l}{b}$$

La variable corresponde a la velocidad lineal del robot, ω a la velocidad angular, νr es la velocidad lineal de la llanta derecha, νl es la velocidad lineal de la llanta izquierda y b es la distancia entre el centro de una llanta y el centro de la otra [4]. Estas ecuaciones se reescriben en función de la velocidad angular y la velocidad lineal, ya que lo se requiere saber es la velocidad que se le debe asignar a cada llanta del robot.

$$\nu r = \nu + \frac{\omega * b}{2}$$
$$\nu l = \nu - \frac{\omega * b}{2}$$

Como se puede observar las velocidades de las ruedas están compuestas por un componente de velocidad lineal y angular. Esta ecuación final se ajusta para que dependiendo de la distancia a la que se encuentre el robot de un obstáculo, los componentes angulares y lineares respondan de una forma específica. Se definieron dos intervalos para la respuesta de velocidad de cada una de las llantas.

Si la distancia con respecto al obstáculo es menor a 0,5 metros, el robot únicamente tendrá velocidad angular (gira sobre su propio eje) con el fin de que pueda evitar el obstáculo y una posible colisión contra él. En este punto el robot también girará con la máxima velocidad posible, por lo que no se usa el valor proporcional dado por los campos potenciales.

El valor proporcional de los campos potenciales se utiliza cuando la distancia del robot a los obstáculos es mayor a 0,5 metros. Para este caso, se definió también una velocidad angular proporcional al ángulo resultante que tiene que girar el robot (denominado *angle*). Este ángulo de giro nunca supera 180°, por lo tanto si se quiere girar este ángulo máximo se debe establecer la velocidad máxima del robot Pioneer (1,2 m/s), y para ángulos menores a 180° se obtiene la siguiente fórmula de velocidad que vendría a reemplazar el término $(\omega * b)/2$ de las ecuaciones descritas anteriormente:

$$\nu 2 = \frac{1{,}2 * angle}{\pi}$$

Finalmente, el término de la velocidad lineal de las ecuaciones anteriores se sustituye por el valor de velocidad proporcional dado por los campos potenciales (denominado como ν) multiplicado por la velocidad máxima del robot. Esto hace que las ecuaciones de velocidad para una navegación autónoma, basadas en el modelo cinemático del robot y el algoritmo para evitar obstáculos basado en campos potenciales, sean las siguientes:

$$\nu r = 1{,}2\left(\nu + \frac{\text{angle}}{\pi}\right)$$

$$\nu l = 1{,}2\left(\nu - \frac{\text{angle}}{\pi}\right)$$

Implementación de waypoints

Los waypoints se utilizaron con el fin de que el robot pueda navegar mayores distancias y así no tenga problemas con los mínimos locales que pueden presentarse durante la navegación del robot. Una consideración importante al utilizar esta técnica es que, para fines de la aplicación el usuario debe tener un conocimiento básico de dónde podrían originarse estos puntos conflictivos.

Los waypoints son puntos que definen la trayectoria del robot. El algoritmo de control de trayectoria y velocidad se aplica para el primer waypoint guardado, una vez que este punto es alcanzado, se actualiza el algoritmo con el siguiente waypoint y así consecutivamente hasta que el robot alcanza la coordenada meta o waypoint final [5].

Resultados

Por medio de la simulación realizada en V-Rep se pudo corroborar que el algoritmo de navegación autónoma basado en los datos del sensor 3D Kinect para la detección de obstáculos y la definición de trayectoria por medio de campos potenciales y waypoints tuvo resultados positivos.

El robot fue capaz de desplazarse sin problemas, evitando obstáculos y llegando a su objetivo en un espacio de 400 m^2 el cual cuenta con pasillos de 1,70 metros, anchos de puertas de 1 metro y un pasillo principal de 2 metros de ancho.

Durante la navegación el robot esquivó con éxito objetos que se encontraban en los pasillos, pasó sin problemas a través de las puertas de los recintos y de igual forma fue capaz de rotar completamente su posición para retornar a un punto, todo esto sin colisionar con objetos en ningún momento. En la figura 3 se puede observar el espacio simulado para probar los algoritmos de detección de obstáculos y campos potenciales, así como parte de la navegación realizada por el robot Pioneer utilizando como base un sensor Kinect junto con un GPS y una unidad de medición inercial. Los puntos en rojo son los waypoints establecidos para definir los tramos de navegación.

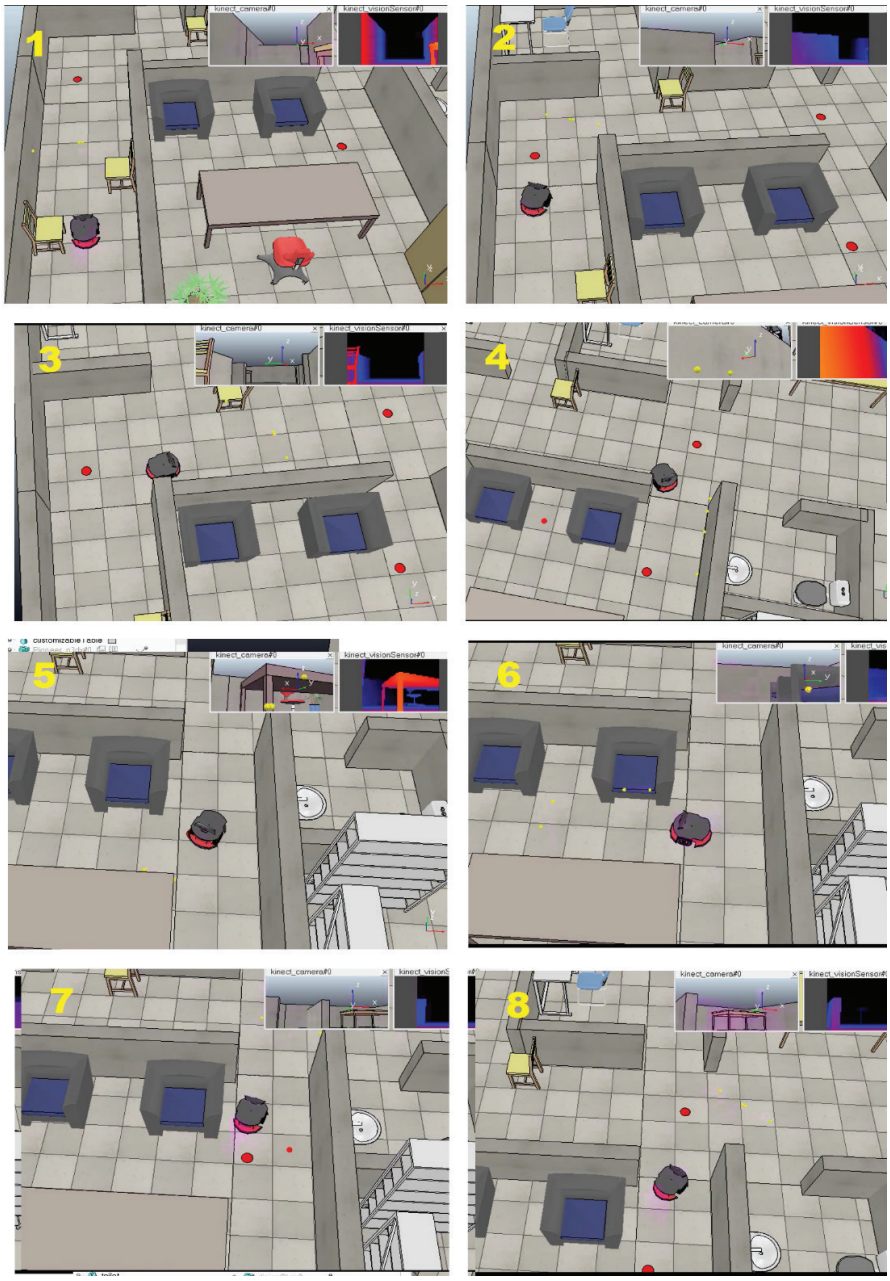

Figura 3. Parte de la navegación final realizada por el Pioneer P3-DX.

De igual forma se pudo verificar que al utilizar los sensores en físico, la aplicación desarrollada fue capaz de detectar el punto más cercano al robot dentro el point cloud generado por el Kinect. Esta actividad se llevó a cabo en un ambiente con obstáculos similar al que se simuló, donde se pudo verificar el correcto funcionamiento del algoritmo.

En la figura 4 se analiza un point cloud donde hay un obstáculo muy cercano al robot, que para este caso es una persona de pie. En esta situación, una persona se posicionó a medio metro del robot. La distancia entre la posición del robot y el waypoint es de 5 metros y el ángulo de orientación del robot es de 1,61 radianes (92,24°). Se recuerda que todos los ángulos son referenciados respecto al marco universal. En la captura de la izquierda se puede observar el point cloud capturado por el Kinect mientras que en la captura de la derecha se muestra el resultado del punto más cercano detectado (en rojo).

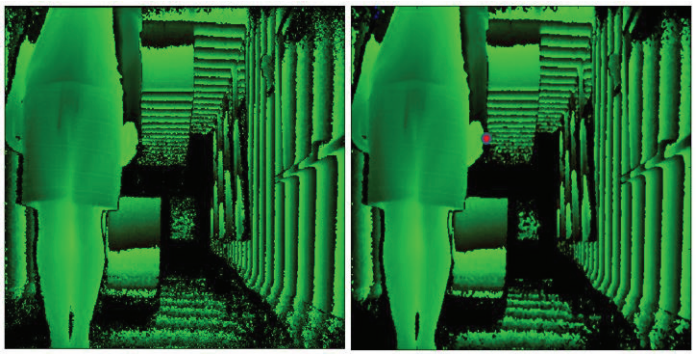

Figura 4. Caso de análisis. Pasillo con obstáculo principal a 0,5 m.

A continuación, se presentan los datos reales brindados del análisis del point cloud y la respuesta del robot en función del obstáculo más cercano al robot.

Datos Kinect (Sistema de coordenadas del sensor):

- Coordenada X: 0,043658 m.
- Coordenada Y: 0,043915 m.
- Coordenada Z: 0,50941 m.
- Distancia al punto más cercano: 0,51128 m.

Señales de actuación en el robot (Sistema de coordenadas universal):

- vLeft: 0,68 m/s.
- vRight: 0,56 m/s.
- Ángulo resultante de la trayectoria: 1,45 radianes (83,08°).
- Ángulo entre el objeto y el robot: 1,70 radianes (97,41°)
- Ángulo entre la meta y posición del robot: 1,57 radianes (90°).

Estos datos corroboran que el robot efectivamente evitaría la colisión con el objeto que se encuentra en su trayectoria y puede continuar su navegación de tal modo que alcance el punto meta. Como se puede observar la velocidad de la llanta izquierda es mayor que la de la derecha lo cual hace que el robot gire en sentido contrario al que se encuentra el obstáculo. También se puede corroborar como el ángulo o trayectoria resultante se ajusta en función del ángulo existente entre el robot y el obstáculo y el robot y la meta.

Conclusiones

Se demostró que a partir del análisis de point clouds generados por sensores 3D de bajo costo y la utilización de técnicas de navegación como campos potenciales se puede desarrollar una aplicación para que un robot móvil sea capaz de navegar de forma autónoma. A nivel de simulación de la aplicación se obtuvieron excelentes resultados donde el robot logró alcanzar todos los puntos del trayecto evitando obstáculos y colisiones. Por otro lado, también se pudo verificar con los sensores en físico el correcto funcionamiento de la detección de obstáculos por medio del point cloud y de igual forma se verificó una correcta respuesta del control de velocidad del robot para definir una trayectoria. Los resultados con los sensores en físico confirman los positivos resultados obtenidos por medio de la simulación en V-Rep.

Si bien es cierto los resultados obtenidos fueron los deseados, todavía existe espacio para mejora. Uno de estos puntos es el de evitar con métodos más eficientes que los waypoints, posibles mínimos locales del algoritmo de campos potenciales. Esto con el fin de incrementar el nivel de autonomía de la aplicación desarrollada.

Referencias

[1] H. Espitia y J. Sofrony, «Sistema de Información Científica Redalyc,» 2012. [En línea]. Consultado en: http://www.redalyc.org/articulo.oa?id=91126903005. [Último acceso: 03 Octubre 2015].

[2] Sociedade Brasilera de Computação, Anais da 34a Jornada de Actualização em Informática JAI 2015, Recife: Sociedade Brasilera de Computação, 2015, pp. 275-290.

[3] Grupo de Investigación en Robótica Autónoma, «GIRA,» 15 julio 2015. [En línea]. Consultado en: http://tecnodacta.com.ar/gira/2015/07/conociendo-coordenadas-reales-con-kinect/. [Último acceso: 06 octubre 2015].

[4] A. Bañó, «Análisis y diseño del control de posición de un robot móvil con tracción diferencial,» junio 2003. [En línea]. Consultado en: http://deeea.urv.cat/public/PROPOSTES/pub/pdf/333pub.pdf. [Último acceso: 16 noviembre 2015].

[5] T. Puls, M. Kemper y R. Küke, «GPS-based position control and waypoint navigation system for quadrocopters,» 2009. [En línea]. Consultado en: https://ieeexplore.ieee.org/document/5354646/authors#authors.

[6] M. Goodrich, «Potential Fields Tutorial,» 12 Mayo 2008. [En línea]. Consultado en: http://phoenix.goucher.edu/~jillz/cs325_robotics/goodrich_potential_fields.pdf.