Escuela Técnica Superior de Ingenieros
Industriales y de Telecomunicación

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Aplicación e integración de sensores para posicionamiento indoor

(Application and integration of sensors for indoor tracking)

Para acceder al Título de

Grado en Ingeniería de Tecnologías de Telecomunicación

Autor: **Diego Cruz Fernández**

Julio 2021

E.T.S. de Ingenieros Industriales y de Telecomunicación

Grado en Ingeniería de Tecnologías de Telecomunicación

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por:	**Diego Cruz Fernández**
Director del TFG:	**Adolfo Cobo García**
Codirector del TFG:	**Luis Rodríguez Cobo**
Título:	"Aplicación e integración de sensores para posicionamiento indoor"
Title:	"Application and integration of sensors for indoor tracking"

Presentado a examen el día: 29 de julio de 2021

para acceder al Título de

Grado Universitario en Ingeniería de Tecnologias de Telecomunicación

Composición del tribunal:

Presidente:	Cobo García, Adolfo
Secretario:	Mirapeix Serrano, Jesús María
Vocal:	Vía Rodriguez, Javier

Este Tribunal ha resuelto otorgar la califación de: .....

Fdo.: El Presidente	Fdo.: El Secretario
Fdo.: El Vocal	Fdo.: El Director del TFG (sólo si es distinto del secretario)
Vº Bº del Subdirector	Trabajo Fin de Grado Nº (a asignar por Secretaría)

Trabajo realizado gracias a los medios aportados por los proyectos PID2019-107270RB-C21 y VirtualMAR (PPNN2017).

Agradecimientos

El primer agradecimiento de este *TFG* es para Luis Rodríguez, Adolfo Cobo y para Jesús Mirapeix por haber depositado su confianza en mí para realizar este trabajo, por su disposición en todo momento y por haberme dado una visión mas amplia de como es el mundo de la *Fotónica*.

Quiero agradecer a mi grupo del *café*. A Vero, Alba y Mariela, por ayudarme durante todo este tiempo y hacerme más llevadero la realización de este trabajo.

También quiero agradecer a Pedro su ayuda con la programación, ya que sin su ayuda no podría haber empezado con buen pie a programar.

Extensible el agradecimiento al resto del Grupo de Ingeniería Fotónica, no solo por dejarme realizar el *TFG* allí, sino también por el ambiente humano tan bueno que se respira en el grupo.

Muy importante también durante todos estos años a mi grupo de amigos: Sergio, Adri, las Ángelas, Elena, Pablo y Luis. Sin su incondicional apoyo y su manera de conseguir evadirme de vez en cuando de las preocupaciones, no hubiera llegado hasta aquí.

Especial mención, a mi mejor amiga, Marina. Hemos estado desde pequeños compartiendo recuerdos, lagrimas y risas.

Por ultimo, pero no por ello menos importante, dar las gracias a mis padres y a mi hermano, Rubén, por haberme apoyado, querido y motivado a llegar hasta aquí.

Universidad de Cantabria

Aplicación e integración de sensores para posicionamiento indoor

Grado en Ingeniería de Tecnologías de Telecomunicación

ETS de Ingenieros Industriales y de Telecomunicación
Grupo de Ingeniería Fotónica (GIF)

por *Diego Cruz Fernández*

Resumen

Es un hecho que en la actualidad los sistemas de navegación juegan un papel muy importante en nuestra vida. Tanta importancia tienen estos sistemas que incluso hemos dejado de lado en la mayoría de las ocasiones un mapa físico para posicionarnos o incluso seguir indicaciones para llegar a un punto especifico. Sin embargo, estos sistemas siguen dependiendo de una vista directa entre el receptor y el sistema satélital. Por ello, surge la necesidad de obtener trayectorias que no dependan directamente de un sistema satélital pero que a la vez dichas trayectorias puedan ser fácilmente representables en un mapa.

Se quiere conseguir posicionar cualquier *móvil*, incluso en interiores cuando no está disponible la señal satélital, también en entornos donde su disponibilidad sea limitada. Por tanto se busca el estudio, simulación y posterior optimizado de una serie de algoritmos capaces de obtener la posición de un *móvil* usando un sensor comercial de la familia *Intel®Realsense™*, trazar su trayectoria y representarla encima de un mapa, tomando como referencia inicial la localización dada por un receptor *GNSS*.

Palabras clave – *GNSS*, *Intel®Realsense™*, *T265*.

University of Cantabria

Application and integration of sensors for indoor tracking

Degree, Telecommunications Engineering

Superior Technical School of Industrial and Telecommunication Engineers
Photonics Engineering Group

by *Diego Cruz Fernández*

Abstract

It is a fact that navigation systems play an important role today in our life. These systems are so important that we have even left aside in most cases a physical map to position ourselves or even follow directions to get to a specific point. However, these systems continue depending on a direct view between the receiver and the satellite system. Therefore, appear the need to obtain trajectories that do not depend directly on a satellite system but at the same time these trajectories can be easily represented on a map.

This academic paper wants to be able to position any *mobile*, even indoors when the satellite signal is not available, also in environments where its availability is limited. Therefore, the study, simulation and subsequent optimization of a series of algorithms capable of obtaining the position of a *mobile* using a commercial sensor of the *Intel®Realsense™* family is sought. In addition to tracing its trajectory and representing it on a map, taking as initial reference, the location given by a *GNSS* receiver.

Key words – *GNSS*, *Intel®Realsense™*, *T265*.

Índice general

Índice de figuras

2 Estado del arte

3 Fundamentos físicos y sensores

4 Desarrollo del trabajo y simulaciones

Acrónimos

GPS	Global Positioning System
SLAM	Simultaneous Localization And Mapping
GNSS	Global Navigation Satellite System
IMU	Inertial Measurement Unit
VSLAM	Visual Simultaneous localization and mapping
ToF	Time of Flight
LED	Light Emitting Diode
AR	Augmented Reality
VR	Virtual Reality
SDK	Software Development Kit
RGB	Red Green Blue
NAD27	North America Datum of 1927
WGS84	World Geodetic System of 1984
GIS	Geographic Information System
GDOP	Geometric Dilution Of Precision
PDOP	Position Dilution Of Precision
DOF	Degrees Of Freedom
VPU	Video Processor Unit
QZSS	Quasi-Zenith Satellite System
TCXO	Temperature Compensated Crystal Oscillator
UART	Universal Asynchronous Receiver-Transmitter
NMEA	National Marine Electronics Association
RTK	Real-Time Kinematic positioning
DGPS	Differential Global Positioning System
DD	Decimal Degrees
DMS	Degrees Minutes Seconds
UTM	Universal Transverse Mercator

1

Introducción

Es un hecho que en la actualidad los sistemas de posicionamiento satelital (*GNSS*) juegan un papel muy importante. No solo por saber donde está un establecimiento o lugar determinado, sino también porque es una herramienta para la localización de dispositivos y de personas en caso de emergencia.

Tanto es así que los sistemas de navegación *GPS* tuvieron un movimiento económico de 150.7 billones de € y de 6.4 billones de dispositivos en 2019 [1].

Uno de los principales inconvenientes de la tecnología *GPS* es la dependencia que se tiene con un satélite para obtener el posicionamiento del dispositivo. Esto es muy crítico en aplicaciones en las que se necesita medir con exactitud la posición y trayectoria del móvil. Además, la situación geográfica del emplazamiento, las perturbaciones atmosféricas, como la lluvia, o la propia refracción de la atmósfera provocan que en el mejor de los casos el posicionamiento *GPS* tenga un error de medida de $3m$ y $5m$ horizontal y verticalmente respectivamente [2].

Sin embargo la propuesta que se implementará en este trabajo es el *SLAM*, un algoritmo basado en la visión y la odometría, utilizado en la industria robótica, el cual nos permite recrear el espacio recorrido a la vez que se consigue obtener la trayectoria del propio móvil. Esta propuesta tiene como ventaja frente al *GPS* la robustez para calcular trayectorias o posiciones con mucha exactitud y en entornos que no tienen visión directa con la constelación de satélites *GPS* [3].

Esta implementación se realizará con la ayuda de un sensor de la familia *Intel®Realsense™*, el sensor *T265* [4]. Cuya implementación se explicará mas adelante.

Esto nos hace tener que definir unas motivaciones y objetivos por los que realizar este trabajo.

1.1. Motivación del trabajo

Como se ha comentado anteriormente, uno de los principales inconvenientes del uso de la tecnología *GPS* u otras tecnologías de posicionamiento satelital, como la propuesta europea *GALILEO*, la propuesta rusa *GLONASS* o la propuesta china *BeiDou*. Es la dependencia que tenemos de un sistema satélital para obtener el posicionamiento del móvil. Sin embargo, utilizando otras técnicas como la que se explicará en este trabajo, podemos obtener un posicionamiento relativamente correcto sin depender de los satélites, algo muy crítico teniendo en cuenta el error de posicionamiento que tienen en el interior de los edificios al no tener visión directa con ellos.

Por ello surge la necesidad de implementar un sistema capaz de posicionar de forma correcta en el interior de los edificios.

Así como incorporar a mi formación académica mayor dominio sobre la programación y utilización de sensores/equipos comerciales para tener un acercamiento más real al mundo de la ingeniería.

1.2. Objetivos y estructura del trabajo

Una vez presentado el contexto y la motivación relativa de este trabajo, se define el objetivo de este Trabajo Fin de Grado como el estudio, simulación y posterior optimización de unos algoritmos capaces de obtener la posición del sensor *T265* de la familia *Intel®Realsense™*, trazar su trayectoria y dejarla en función de las coordenadas terrestres.

Por tanto, el desarrollo de este trabajo se puede dividir en dos partes.

- **Calculo de la trayectoria:** Programación de un algoritmo que recoja los datos del sensor y después los procese.
- **Representación:** Una vez obtenida la trayectoria, esta deberá ser dibujada sobre un mapa con el fin de comprobar que esta propuesta de trabajo consigue recrear una trayectoria completa sin depender constantemente de un sistema *GNSS*.

Estos objetivos serán contemplados a lo largo del trabajo, siguiendo la siguiente estructura:

En el Capítulo 2 se realizará una descripción teórica sobre el *estado del arte* de la tecnología *SLAM* y sobre la familia *Intel®Realsense™*. En el Capítulo 3 se describirá con detalle el sensor utilizado y los fundamentos fisico-matematicos que se utilizan para desarrollar este trabajo. En el Capítulo 4 se desarrollará con detalle el resultado de las simulaciones, los errores de medida y la elección del mejor entorno para el que la propuesta desarrollada es favorable. Por ultimo, en el Capítulo 5 se resumirán las conclusiones de este trabajo, junto con algunas observaciones y sugerencias para posibles mejoras y futuras líneas de investigación.

2

Estado del arte

En este capítulo se describe el estado del arte de las tecnologías presentes en este trabajo. En primer lugar se detallará la tecnología *SLAM*, el principio sobre el que se fundamenta el trabajo. Posteriormente, se hablará de la implicación de la familia *Intel®Realsense™* con la tecnología.

2.1. Tecnología *SLAM*

El origen del problema *SLAM* data de la decada de los 80's y 90's, cuando la industria robótica empezó a crear robots. Los ingenieros querían desarrollar robots que fueran capaces de circular por el taller sin chocarse contra las paredes u otros elementos [5].

Hoy en día es imprescindible para muchas aplicaciones, en interiores y exteriores. Un claro ejemplo, entre otros, es la conducción autónoma y los *robots aspiradora* que cada vez son mas utilizados en los hogares.

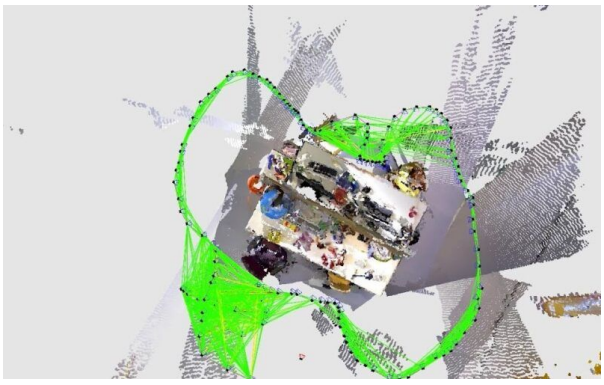


Figura 2.1. Modelado 3D del entorno con la tecnología SLAM.

Esta tecnología se basa en utilizar sensores y/o dispositivos que capten datos visibles, normalmente con una cámara, combinados con unos datos de posicionamiento básico. Recogidos por una unidad de medida inercial, *IMU* por sus siglas en inglés.

Juntos, estos sensores (cámara + *IMU*) crean una imagen del ambiente. El algoritmo *SLAM* ayuda a mejorar la estimación sobre la posición con ayuda del entorno. La posición de un elemento respecto al otro es relativa, pero el algoritmo *SLAM* utiliza un método iterativo para mejorar la estimación de la posición. Al ser un proceso iterativo, a mayor numero de iteraciones se mejora la precisión del posicionamiento. Sin embargo, esto incrementa el precio del *hardware* del equipo, ya que la potencia de procesamiento tiene que ser mayor.

La obtención de imágenes puede ser de dos tipos: *SLAM* visual, *VSLAM*, o con *LiDAR*. Nos centraremos en el *VSLAM* ya que ha sido el implementado en este trabajo.

El *SLAM* visual, *VSLAM*, utiliza una cámara para adquirir imágenes del entorno. Se pueden emplear para ello cámaras simples como gran angulares, "*ojos de pez*". Cámaras compuestas como multicámaras, o cámaras que integran un sensor de profundidad *ToF*.

La cámara *ToF* es una cámara de *rango* que utiliza la técnica de tiempo de vuelo para resolver la distancia a la que está la cámara y cada punto de la imagen capturada. Esta medida se realiza con el tiempo que tarda en volver una *luz artificial*, LED o láser, desde que sale del sensor hasta que vuelve rebotada de una superficie.

La implementación del *VSLAM* suele ser además de bajo coste, ya que se usan normalmente cámaras muy baratas. Sin embargo, estas cámaras proveen de una gran cantidad de información, la cual se utiliza para detectar puntos de referencia sobre los cuales aplicar una optimización para mejorar todavía más el desempeño del *SLAM*.

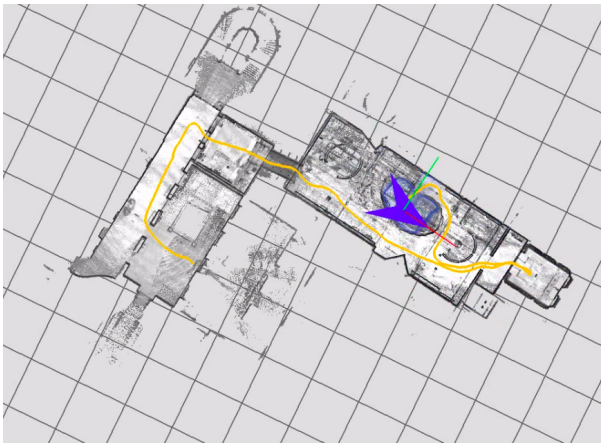


Figura 2.2. Demo del *cartógrafo* de *Google*.

Son muchas las implementaciones que se pueden realizar con la tecnología *SLAM*. Tanto es así que la propia *Google* en el año 2016 lanzó un algoritmo *open source* de cartografía basado en *SLAM* (Figura 2.2).

¿Por qué es tan importante *SLAM* y *VSLAM*?

Como se ha mencionado anteriormente el concepto de *SLAM* nació ante la necesidad de obtener el posicionamiento *indoor* para los *robots*. El *SLAM* es muy útil para las localizaciones en las que la disponibilidad de un sistema *GNSS* es limitada.

En concreto, en este trabajo, se intenta explotar la limitación de la disponibilidad de un sistema *GNSS* para poder obtener la trayectoria de un móvil y dejarla en función de las coordenadas terrestres. Para ello se utilizará un dispositivo de la familia *Intel®Realsense™*.

2.2. Familia *Intel®Realsense™*

Intel®Realsense™ es una gama de productos de *Intel®* que tiene como objetivo obtener el seguimiento y profundidad para brindar a las maquinas y dispositivos la capacidad de percepción de profundidad. Estas tecnologías de *Intel®* se emplean en varios tipos de dispositivos como son: drones autónomos, robots, realidad aumentada (AR), realidad virtual (VR) y dispositivos inteligentes del hogar.

Los productos de la familia *Intel®Realsense™* son extensos. *Intel®* dispone de procesadores de visión, módulos de seguimiento y profundidad y cámaras de profundidad [6]. Estas soluciones de *hardware* disponen de un kit de desarrollo de *software*, *SDK*, bastante completo. El cual es de código abierto, lo que abre un mundo de posibilidades. Este *SDK* es multiplataforma y además esta disponible en varios lenguajes de programación como, `C++`, Python, MATLAB o `C#`. Siendo el uso del *SDK* en `C#` más atractivo para la implementación buscada en este trabajo.

Como se ha comentado anteriormente, *Intel®* dispone de un amplio abanico de soluciones en el mercado. En este trabajo se ha implementado el sensor de posicionamiento T265, sobre el que se hablará con detalle en la Subsección 3.2.1.

Estos sensores tienen transcendencia incluso en la investigación ya que con ellos se han realizado propuestas muy buenas. Un ejemplo de ello está reflejado en un *paper* publicado en *IEEE* [7]. En el se detalla como han implementado el sensor "D435" de la familia *Intel®Realsense™* para hacer un mapa 3D del entorno outdoor.

Este sensor es una cámara *RGB-D*, es decir, dispone de un sensor *RGB* y un sensor de profundidad con el cual se calcula la distancia que hay desde la cámara a los objetos.

En la Figura 2.3 se puede ver un trozo del trayecto que ha seguido el sensor y la recreación en 3D de dicha trayectoria.

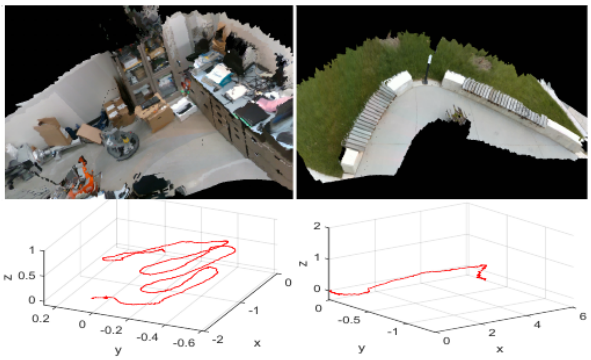


Figura 2.3. Modelado 3D del entorno outdoor.

Este *paper* ilustra uno de los grandes potenciales que tienen estos sensores para implementar las técnicas del *SLAM* en el mundo real. En concreto esta propuesta la podríamos enmarcar como un *cartógrafo 3D*.

Otro uso de esta familia es el que se explica en otro *paper*. Se usa un sensor diferente al anterior, el R200, para crear una cámara de profundidad [8].

(a) **(b)**

Figura 2.4. Ejemplo cámara volumétrica

Como se observa en la Figura 2.4, usando este sensor se ha conseguido recrear volumetricamente un objeto (Figura 2.4a) e incluso saber las dimensiones de otro objeto (Figura 2.4b).

En definitiva, las soluciones de *Intel®* tienen un potencial muy alto y por ello se ha optado por utilizar uno de sus sensores para el objetivo de este trabajo. El sensor utilizado, la T265, se abordará en el siguiente capítulo.

3

Fundamentos físicos y sensores

Este capítulo describe los fundamentos físico-matemáticos sobre los que se fundamenta este trabajo, así mismo, también se habla detalladamente sobre el sensor *T265* de la familia *Intel®Realsense™*, la cual se ha detallado en el apartado 2.2. Así como el receptor *GNSS u-blox NEO-M8N*.

3.1. Fundamentos físicos

Este trabajo se ha fundamentado con una base física con la que se ha tenido que lidiar en la realización del mismo.

Así mismo, hay ciertas cosas que se llegaron a programar pero no se implementaron ya que la *T265* las hacía directamente. Pero considero que son importantes mencionarlas y explicarlas, ya que parte de la problemática que tiene el *SLAM* nace de ahí.

3.1.1. Transformaciones y cuaterniones

Las transformaciones y los cuaterniones son un concepto fundamental a la hora de entender como el sensor *T265* ha ido obteniendo los distintos puntos que conforman la trayectoria.

Empecemos hablando sobre las transformaciones. Las transformaciones son cambios que se les hace a un vector x para convertirle en un vector y. Estas transformaciones, las cuales se realizan con una matriz de dimensión $\mathbf{M_{N+1}x_{N+1}}$, siendo en nuestro caso $N = 3$, ya que estamos en 3 dimensiones, son [9]:

- **Traslación:** Una matriz que mueve un objeto/vector a lo largo de uno o mas ejes.

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- **Rotación:** Un grupo de matrices que rotan un objeto/vector sobre uno de los 3 ejes de coordenadas. La matriz de rotación es más compleja que la de traslación. Se rota el objeto según un angulo de rotación (θ).

$$\mathbf{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta) & sin(\theta) & 0 \\ 0 & -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_y(\theta) = \begin{pmatrix} cos(\theta) & 0 & -sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ sin(\theta) & 0 & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_z(\theta) = \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 & 0 \\ sin(\theta) & cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- **Ángulos de *Euler* y cuaterniones:** Los ángulos de *Euler* son un conjunto de tres ángulos que representan la orientación en el espacio. Cada ángulo representa una rotación alrededor de uno de los 3 ejes ($\mathbf{X}, \mathbf{Y}, \mathbf{Z}$). El orden de las rotaciones es importante porque la multiplicación de matrices no es conmutativa. Sin embargo, los ángulos de *Euler* tienen un problema, el bloqueo de *cardán*. El bloqueo de *cardán* se produce cuando una rotación en un ángulo reorienta uno de los ejes para alinearlo con otro. Cualquier rotación adicional de cualquiera de los dos ejes dará como resultado la misma transformación del modelo, eliminando un grado de libertad del sistema. Por tanto, los ángulos de *Euler* no son ideales para realizar sucesivas rotaciones. Un mejor enfoque son los *cuaterniones*.

  Los cuaterniones ofrecen una forma alternativa de describir rotaciones. El cuaternión es un cantidad de 4 dimensiones, similar a un numero complejo. Mientras que un número complejo tiene una parte imaginaria, $\mathbf{i}$, el cuaternión tiene un vector de tres componentes complejas, $(\mathbf{i}, \mathbf{j}, \mathbf{k})$.

De la misma manera que una matriz de rotación toma un ángulo y un eje para rotar, un cuaternión usa el ángulo como parte real y el eje en la parte vectorial (imaginaria), lo que representa una rotación en cualquier eje. Lo trascendente de los cuaterniones es que es un conjunto de N rotaciones que se puede representar como un producto de N cuaterniones que da como resultado un único cuaternión con todas las rotaciones combinadas. Aunque esto también se podría hacer con una concatenación de matrices de rotación, nos podríamos encontrar con el problema del bloqueo del *cardán*. Por tanto, esta es una solución que no tiene ese problema de bloqueo. Se tiene que tener en cuenta que es posible convertir un cuaternión en una matriz de rotación y viceversa.

3.1.2. Sistemas *GNSS*

En el Capítulo 1 se presentaron las motivaciones y objetivos del trabajo. Se hizo hincapíe en la dependencia que tenemos de un sistema satélital para obtener la posición. Por ello en esta sección vamos a hablar de como funciona en concreto el sistema *GPS*, pero es extrapolable al resto de los sistemas.

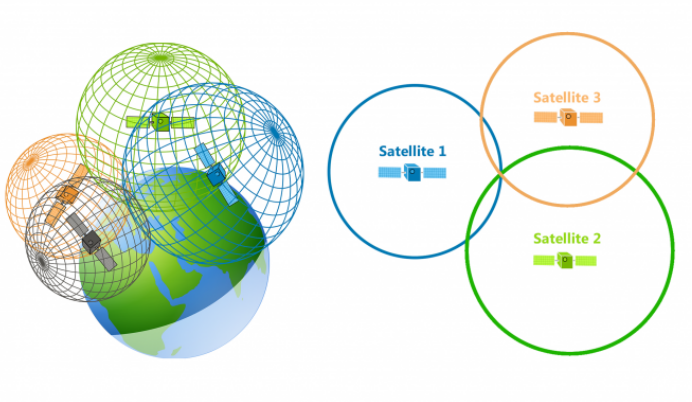


Figura 3.1. Sistemas *GNSS*.

Los receptores *GPS* utilizan un método conocido como *trilateración* [10]. La *trilateración* es una técnica geométrica que sirve para medir distancias, no ángulos.

Los satélites realmente lo que hacen es transmitir a los receptores *GPS* una señal para que el receptor capte una hora y una distancia especificas.

Vamos a explicar el concepto de *trilateración* para el caso 2D. Por ejemplo, el primer satélite emite una señal de forma periódica al receptor. No conocemos el ángulo, pero sabemos a que distancia está ese satélite. Por lo tanto podríamos estar en cualquier punto del circulo al que ilumina el satélite 1 (Figura 3.1).

Cuando recibimos la señal del segundo se produce el mismo proceso, podemos estar en cualquier intersección del circulo de la zona iluminada por el primer y segundo satélite. Esto se puede ver gráficamente en la Figura 3.2a.

(a) **(b)**

Figura 3.2. *Trilateración* con varios satélites.

Al recibir la señal del tercer satélite, ya podemos discernir claramente cual es el punto geográfico en el que nos encontramos, ya que la intersección de los 3 círculos nos da un único punto, tal como se observa en la Figura 3.2b.

Por otro lado, los topógrafos utilizan la *triangulación* para medir distancias desconocidas. Esto se hace estableciendo longitudes de referencia.

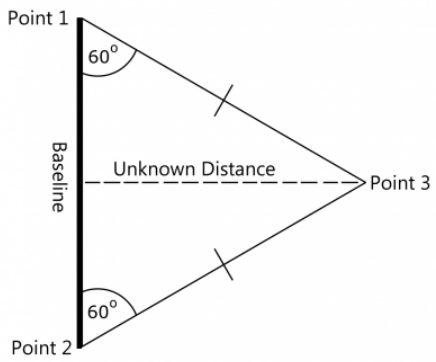


Figura 3.3. Esquema de la triangulación.

Desde cada punto, los topógrafos miden ángulos distantes. Cuando se conocen las longitudes y los ángulos, la triangulación determina la distancia formando un triangulo. Esto se observa en la Figura 3.3.

Los topógrafos tomaron alrededor de 26.000 estaciones como puntos de referencia. Esto lo hicieron para crear el *datum de América del norte de 1927*, *NAD27* por sus siglas en inglés. Sin embargo, el sistema *GPS* utiliza la *encuesta geodésica mundial de 1984*, *WGS84*.

Este ultimo, el *WGS84* [11], está compuesto por un elipsoide de referencia, un sistema de coordenadas estándar, datos de altitud y un geoide. Utiliza además, el centro de masa de la tierra como origen de coordenadas. Los *geodesistas* creen que el error es de menos de $2\,cm$.

Cuando necesitamos calcular tiempos, distancias entre distintos puntos, ciudades, etc. Recurrimos a sistemas *GIS* [12], sistema de información geográfica. Es un conjunto de herramientas que integra y relaciona diversos componentes que permiten la organización, almacenamiento, manipulación, análisis y modelado de grandes cantidades de datos geográficos procedentes del mundo real. Pero para poder introducir estas coordenadas con precisión, el primer paso es definir de forma única todas las coordenadas de la tierra.

Necesitamos un marco de referencia para la *latitud* y *longitud*. ¿Sin esta referencia, como sabríamos en que parte de la Tierra estamos?

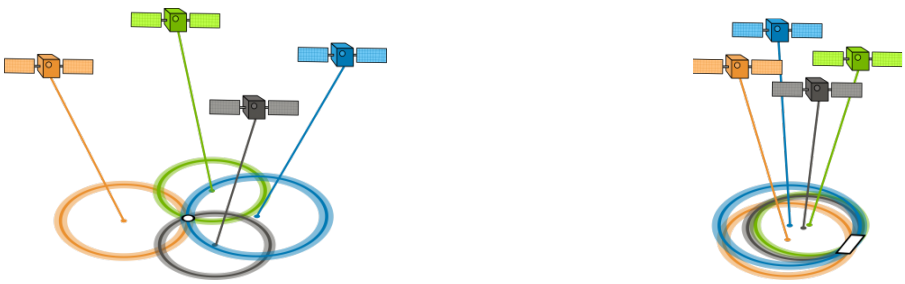
Debido a que la Tierra es curva hay que realizar proyecciones de la Tierra sobre mapas planos. Los topógrafos y los *geodesistas* han definido con precisión las ubicaciones en la Tierra. A diferencia de otros métodos que utilizaban un geoide, medidas del nivel del mar, el *WGS84* ,utiliza como se ha comentado anteriormente, un elipsoide el cual se ha estimado lo mejor que se ha podido a través de una colección masiva de mediciones de la superficie terrestre. Este sistema acompañado de la generalización del uso del *GPS* unificó en un único modelo de elipsoide toda la Tierra.

Sin embargo, en todo sistema real hay luces y sombras. Aunque en este trabajo no se traten los errores o problemáticas de los sistemas *GNSS*, me parece importante al menos mencionarlos brevemente.

Los problemas más relevantes son los siguientes [2]:

- ***GDOP/PDOP*:** El *GDOP*, dilución geométrica de la precisión, o el *PDOP*, dilución de precisión de la posición, describe el error causado por la posición relativa de los satélites *GPS*.

  Básicamente cuantas más señales pueda "ver" un receptor *GPS* mas precisa será la estimación de la localización.



(a) Buen *GDOP* **(b)** Mal *GDOP*

Figura 3.4. Ejemplos de *GDOP*.

Como se puede observar en la Figura 3.4a si desde el punto de vista del observador los satélites están dispersos en el cielo, entonces el receptor *GPS* tendrá un buen *GDOP* lo que se traduce en una buena precisión en la localización.

Sin embargo, si los satélites están fisicamente juntos, se tiene un *GDOP* pobre, Figura 3.4b, lo que se traduce en un mal posicionamiento, ya que el receptor *GPS* no sabe exactamente en que punto estará.

- **Refracción de la atmósfera:** La troposfera y la ionosfera pueden cambiar la velocidad de propagación de una señal *GPS*. Debido a las condiciones atmosféricas, la atmósfera refracta las señales de los satélites a medida que pasan en su camino hacia la superficie terrestre.

  Para solucionar esto, el sistema *GPS* puede usar dos frecuencias separadas para minimizar el error de la velocidad de propagación. Dependiendo de las condiciones, este tipo de error del *GPS* podría desviar la posición en hasta $5\ m$. La frecuencia *L1* a $1575.42\,MHz$ y la *L5* a $1176.45\,MHz$ [13] (sistemas modernos). Los sistemas modernos utilizan este *GPS dual* para mejorar el tiempo que se tarda en localizar un *móvil* y para que la localización sea mas precisa.

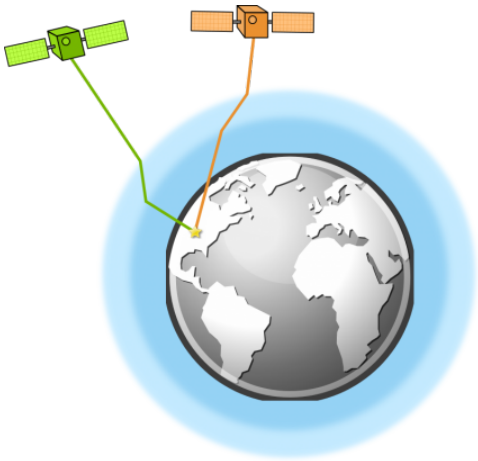


Figura 3.5. Refracción de la atmósfera.

En la Figura 3.5 se puede ver el efecto de las capas de la atmósfera en la propagación de los rayos de los satélites *GPS*.

- **Efectos multitrayecto:** Una fuente de error común en los cálculos del *GPS* es el efecto multitrayecto. El multitrayecto ocurre cuando la señal del satélite rebota en estructuras cercanas como edificios o montañas.

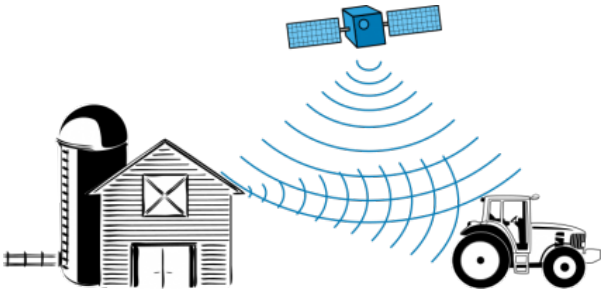


Figura 3.6. Efecto del multitrayecto.

El receptor detecta la misma señal dos veces en distintos instantes de tiempo, tal como se ve en la Figura 3.6. Aunque suele provocar un error leve de $1m$, unido a otros efectos distorsiona aún más la precisión de la localización.

- **Hora y ubicación satelital (efemérides):** La precisión del reloj atómico de un satélite *GPS* es de $1\ ns$ por cada "tic" de reloj. Utilizando la *trilateración* de señales horarias en órbita, los receptores *GPS* pueden obtener posiciones precisas. Pero debido a errores de sincronización del reloj atómico del satélite, el error de precisión en la localización puede alcanzar los $2\ m$.

  La información sobre las efemérides contiene detalles sobre la ubicación de un satélite específico. Si en algún momento no se conoce su ubicación exacta, esto originará errores.

3.2. Sensores

En esta sección se presentan los sensores utilizados en este trabajo, el sensor de *tracking Intel®Realsense™ T265* y el receptor *GNSS u-blox NEO-M8N*.

Vamos a detallar los componentes y funciones principales que tienen. Así como la relevancia de estas funciones para el desempeño del trabajo. La implementación de estos sensores será detallada en el Capítulo 4. Mientras tanto vamos a presentar la *T265* y al receptor *NEO-M8N*.

3.2.1. *Intel®Realsense™ T265*

Como se ha comentado en la Sección 2.2 la familia *Intel®Realsense™* tiene buenas propuestas de *hardware* además de un amplio abanico de publicaciones cientificas en las que se utilizan algunos de sus sensores.

El sensor que se ha implementado en este trabajo es el sensor de seguimiento *T265*.

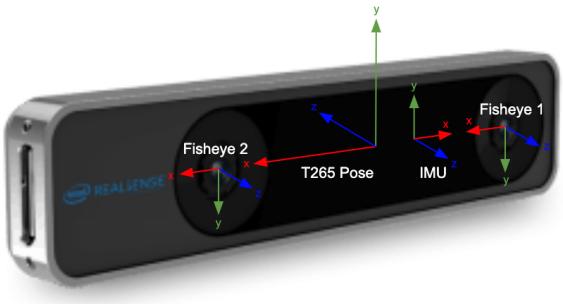


Figura 3.7. Sensor *Intel®Realsense™ T265*.

Apoyándonos en la Figura 3.7 vemos que el sensor dispone dos lentes con objetivo "ojo de pez" los cuales son monocromos, una *IMU* con 6 grados de libertad (*DOF*), una unidad de procesado, *VPU*, diseñada por *Intel®*, la *Intel® Movidius™ Myriad™ 2*. Esta *VPU* es la unidad que dispone de los algoritmos *VSLAM* diseñados por *Intel®*, los cuales son aplicados en ella. Adicionalmente esta unidad tiene una latencia muy baja y tiene un consumo energético muy eficiente $(1.5W)$ [14]. La *T265* no es una cámara de profundidad, por lo que no captura este tipo de datos, pero si que podría ser usada junto con una.

La *IMU* junto con los ojos de pez, capturan información sobre el movimiento y las imágenes del entorno que son procesadas por la *VPU*. Utilizando los algoritmos *VSLAM* se calcula la posición actual que tiene la *T265* en función de la posición de referencia, la posición inicial en la que se encendió la misma.

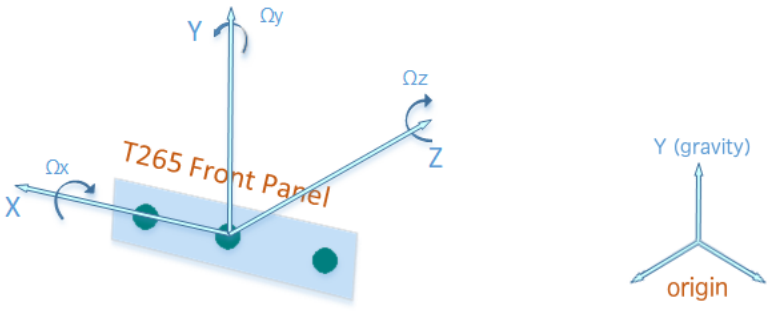


Figura 3.8. Ejes de la *Intel®Realsense™ T265*.

Como se puede ver en la Figura 3.8, los ejes que considera la *T265* para la posición no son los usuales. La *T265* utiliza los ejes típicos de *AR/VR*.

Estos ejes por tanto están orientados de la siguiente manera:

- Eje x es positivo hacia el "ojo de pez" derecho
- Eje y es positivo hacia arriba, parte superior del dispositivo
- Eje z es positivo hacia adentro, hacia la parte posterior del dispositivo

El centro de origen del seguimiento corresponde a la ubicación central entre los "ojos de pez" derecho e izquierdo.

Otro tema de interés es la *calibración*, la cual se comenta en la documentación. Sin embargo, la *T265* ha sido calibrada en el proceso de fabricación y no precisa calibración a diferencia del sensor de profundidad *D435i*.

Así mismo, la *T265* utiliza como tiempos de referencia el tiempo del computador host. Esta referencia está sincronizada con una precisión de pocos milisegundos.

La *T265* tiene un archivo de calibración de la odometría para tener en cuenta las dimensiones de las ruedas del *robot* u otro dispositivo en el que vaya embarcado.

Por último comentar la importancia que tienen estas funciones en el desempeño de este trabajo. Gracias a la *IMU* y a los algoritmos *VSLAM* presentes en la *VPU*, este sensor de seguimiento de *Intel*® ha ido obteniendo los puntos que conforman una trayectoria, lo cual es crucial para la aplicación de "posicionamiento indoor".

3.2.2. Receptor *GNSS u-blox NEO-M8N*

Se ha hecho hincapíe a lo largo del documento sobre la importancia de posicionar *móviles* sin depender de una constelación de sistemas *GNSS*. Este trabajo pretende conseguir esto, no obstante, utilizamos este receptor para poder localizar en el instante inicial a nuestro *movil*, para dejar todo su movimiento en función de las coordenadas terrestres, lo cual se comentará en el Capítulo 4, concretamente en la Subsección 4.2.1.

Figura 3.9. Receptor *GNSS u-blox NEO-M8N*.

El Receptor *GNSS u-blox NEO-M8N* [15] (figura 3.9) es un módulo versatil con varios servicios *GNSS* incorporados, los cuales son: *GPS/QZSS, GLONASS, GALILEO* y *BeiDou*. Siendo *QZSS* la propuesta complementaria a *GPS* en Japón.

Adicionalmente, el receptor *u-blox NEO-M8N* utiliza 72 canales *L1* y según el fabricante la precisión de la localización es de $2.5\ m$. Si la información que tiene el receptor sobre los satélites sigue siendo valida, es decir, si la información de las órbitas de los satélites, su posición y hora es correcta hablaríamos de que el arranque del receptor es en caliente, *hot start*. Esto implica que la ubicación actual de los satélites es cercana a la que tiene almacenada. En concreto el *hot start* del *u-blox NEO-M8N* es de $1\ s$. Si por el contrario las condiciones anteriores no se cumplen, no teniendo datos almacenados o relativamente actualizados o no sabiendo ni siquiera que hora es, el receptor tiene que hacer una búsqueda a ciegas en todo el espacio de búsqueda [16]. Esto es conocido como arranque en frío, *cold start*, el cual en el receptor *u-blox NEO-M8N* es de $26\ s$.

El receptor *NEO-M8N* utiliza como oscilador local un *TCXO*, es decir, un oscilador de cristal con un circuito de reactancia sensible a la temperatura en su bucle oscilación. Esto se hace para compensar las características de frecuencia-temperatura inherentes a la unidad de cristal [17]. Además el *TCXO* esta especialmente diseñado para poder trabajar a temperaturas altas, ya que estas vuelven inestables la oscilación [18].

Este modulo es de muy alta eficiencia y tiene una buena recepción de la señal de la constelación *GNSS* pertinente, según la compañía la sensibilidad es de hasta $-167\ dBm$.

A continuación, en el capítulo 4 vamos a ver como se han implementado estos sensores.

4

Desarrollo del trabajo y simulaciones

Este capítulo tiene por objetivo presentar el desarrollo de este Trabajo Fin de Grado. Podemos dividir este capítulo en 3 grandes partes, las cuales son:

- Sintaxis *NMEA* y captación de datos capturados por los sensores.
- Procesado de datos y representación de los mismos.
- Simulaciones y resultados.

A lo largo de este capítulo se irán detallando todas las decisiones tomadas en el trabajo para conseguir implementar la propuesta de "posicionamiento indoor", se comentará también, a lo largo del capítulo, cuan tan buenas son los resultados obtenidos. Se explicará brevemente cómo se ha implementado a nivel de código esta propuesta y por ultimo se enlazará con el Capítulo 5 para concluir este trabajo.

4.1. Sintaxis *NMEA* y captura de datos

Lo primero que se ha tenido que hacer en este trabajo fue capturar de forma coherente los datos de los sensores. Esto se programó utilizando *.NET* (C#) ya que el *SDK* de *Intel®Realsense™* ofrece diversas funciones para acceder de forma sencilla a los datos, y además está soportado por la comunidad de desarrolladores. Además, se disponía de algo de experiencia en programación en *NET*.

Inicialmente se tuvo que investigar sobre cómo conectar el receptor *u-blox NEO-M8N* al ordenador, ya que este utiliza una interfaz *UART*, la cual comunica la información utilizando los puertos serie. Después de instalar los pertinentes *drivers*, ya que este receptor no tiene una interfaz *plug and play*, se utilizó el software *u-center*, diseñado por *u-blox* para ver que tipo de interfaz e información arrojaba el receptor *GNSS*.

Después de ver la interfaz de *u-center* el siguiente paso fue conseguir sacar la posición obtenida por el receptor y usarla para el interés del trabajo.

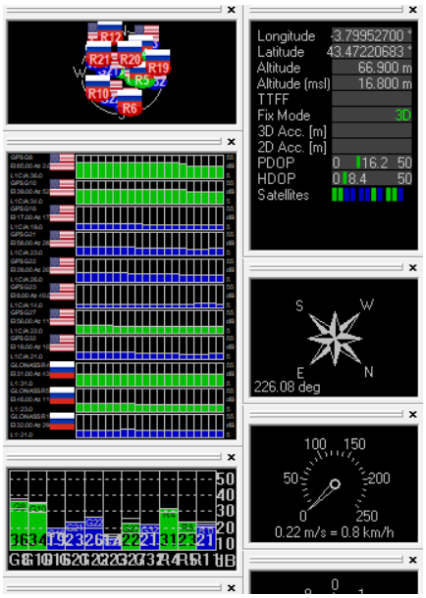


Figura 4.1. Ejemplo de la localización obtenida por el receptor *u-blox NEO-M8N* vista en *u-center*.

Como se puede ver en la Figura 4.1 *u-center* está organizado en pequeñas vistas en forma de ventana que arrojan distintos tipos de datos. Empezando por arriba a la izquierda se ven todos los satélites de distintas constelaciones (*GPS, GLONASS, GALILEO...*) a los que se ha conectado y en un diagrama polar se ven cómo están dispuestos actualmente los mismos. A la derecha se ven los datos geográficos en los que se está, en el momento de la prueba en la *ETS de Ingenieros Industriales y de Telecomunicación* de la UC. La ventana central muestra información sobre cada satélite al que se ha conectado para obtener el posicionamiento. Debajo tenemos una información similar pero en forma de gráfico de barras y justo a la derecha de este tenemos la velocidad a la que se esta desplazando el receptor *GNSS*, que en el momento de la prueba estaba parado pero figura como que se estaba desplazando a $0.8\,Km/h$

Estos parámetros que obtiene el receptor *GNSS* están codificados en un formato conocido como *NMEA*, el cual vamos a comentar a continuación.

4.1.1. Sintaxis *NMEA*

NMEA es el acrónimo de la *asociación nacional de electrónica marina de los estados unidos*. Esta asociación data de 1957, sus creadores eran distribuidores electrónicos que buscaban tener mejor comunicaciones con los fabricantes.

Hoy en día el estándar del formato de datos *GPS* es *NMEA*. Es extrapolable a otros tipos de *GNSS*, hay ciertos mensajes que cambian pero en términos generales el formato es el mismo.

La sintaxis *NMEA* tiene el siguiente formato [19]:

```
$ GPGGA, 181908.00,3404.7041778, N, 07044.3966270,
W, 4,13,1.00,495.144, M, 29.200, M, 0.10,0000 * 40
```

Figura 4.2. Ejemplo sintaxis *NMEA*

Todos los los mensajes *NMEA* comienzan por "$" y cada campo está separado por una coma.

A continuación vamos a explicar el ejemplo de la Figura 4.2:

- **GP** representa que la posición obtenida es *GPS*. Si fuese **GL** denotaría en su lugar que es *GLONASS*.
- **181908.00** es la marca de tiempo: en hora *UTC*, es decir, en horas minutos y segundos.
- **3404.7041778** es la latitud en el formato *DDMM.MMMMM*.
- **N** denota latitud *norte*.
- **07044.3966270** es la longitud en el formato *DDDMM.MMMMM*.
- **W** denota longitud oeste.
- **4** denota el indicador de calidad, que puede tomar estos valores:
  - **1** : Indica que la coordenada está sin corregir.
  - **2** : Indica que la coordenada se ha corregido diferencialmente.
  - **4** : Indica coordenada fija *RTK* (precisión centimétrica)
  - **5** : Indica *RTK float* (precisión decimétrica)

Los sistemas con *RTK*, navegación cinética satélital en tiempo real, se conectan a una estación base para corregir todavía más la posición, consiguiendo una exactitud submétrica [20]. Estas técnicas son las diferenciales, como *DPGS*, la cual consigue que la precisión llegue a ser de $1 - 3\,cm$ [21].

El mensaje $ GPGGA es el mensaje *GPS NMEA* básico. Hay otros tipos de mensajes que dan información alternativa o complementaria. Estos son por ejemplo $ GPGSA, que contiene información del seguimiento de los satélites, el $ GPGSV, información sobre la azimut y elevación de los satélites conectados, el $ GPVTG, velocidad sobre el suelo y desplazamiento, por último el $ GPGST, precisión horizontal y vertical estimada, en el caso de un sistema *GNSS* este mensaje sería $ GNGST.

4.1.2. Captura de datos

Una vez entendida la sintaxis *NMEA* se procedió a implementarla en la programación. Como se ha comentado anteriormente se ha programado en *.NET*.

```
// Vamos a obtener la posición actual gracias al receptor GNSS.

Console.WriteLine("Vamos a obtener la posición actual");

string portname = "COM8"; // Introducir el puerto serie
int baudrate = 115200; // Velocidad en baudios.
var port = new SerialPort(portname, baudrate);
var device = new NmeaParser.SerialPortDevice(port);
```

Figura 4.3. Conexión receptor *GNSS u-blox NEO-M8N*

Tal como se puede observar en la Figura 4.3 el receptor se ha conectado a través de un puerto *USB* pero comunicándose como si de un puerto serie se tratase. El puerto al que siempre se ha conectado el receptor es el puerto serie 8 (*COM8*), pero podría haber sido otro, en ese caso se hubiese cambiado el numero en el código.

Después se tuvo que fijar la velocidad de trasferencia de datos en baudios (símbolos por segundo). Originalmente la velocidad de transferencia era de $9600\ baudios$, pero este receptor fue modificado y ya tenia una velocidad mas alta de $11500\ baudios$.

Por último había que interactuar con el receptor, para poder sacar la información de la latitud y longitud que este obtuviese. Para ello se busco una librería que ya tuviese programada la sintaxis *NMEA*. Estas librerías reciben el nombre de *parser* ya que traducen una sintaxis en otra. La librería utilizada recibe el nombre de *NMEAParser* [22], la cual nos ha permitido obtener la latitud y longitud de la localización obtenida por el receptor *u-blox NEO-M8N*.

El receptor *GNSS* es capaz de estimar la altura a la que estamos del nivel del mar, pero aunque se intento implementar no se consiguió, así que la altura se pregunta por "consola" al usuario.

La información de la latitud y longitud se imprime en un fichero para poder ser luego procesada. Esto será explicado con detalle en la Subsección 4.2.1.

Después se procedió a programar la captura de datos del sensor de seguimiento *Intel®Realsense™ T265*. Esto se realizó con el *SDK* que *Intel®* dispone en *.NET* [23].

En el código se menciona y de hecho se imprime por "consola" el que hay que ejecutar el código mirando hacia el *norte*, ya que al encenderse la *T265* esta apuntará hacia allí. Esto es debido a que no tenemos una brújula y que todas las rotaciones que hace la *T265* de forma interna cojan como referencia el norte, el cual también es la referencia para la latitud y longitud terrestre.

De la información que nos da la *T265* sacamos los cuaterniones, ya que en un primer momento se pensaba que la posición obtenida había que rotarla según la cantidad que los cuaterniones indicaran, sin embargo, se comprobó que la información ofrecida era directamente la posición y ángulos en cada instante respecto a la posición en el encendido, con lo que no fue necesario realizar ninguna transformación adicional. También se sacó la posición $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ del sensor, así como la fidelidad de la posición, es decir, cuan de fiable es el dato que nos ha arrojado en un momento dado el sensor.

Inicialmente el código estaba programado para que parara de obtener datos de la *T265* al cabo de N muestras, sin embargo esto no era muy eficiente ya que no podías controlar cuanto tiempo iba a tardar. Al final se optó por pedirle al usuario por "consola" cuanto tiempo, en segundos, se querían obtener muestras. El código "interroga" al *sistema operativo* para saber la hora actual y cuando la hora actual sea más grande que el tiempo introducido, entonces en ese momento el programa termina.

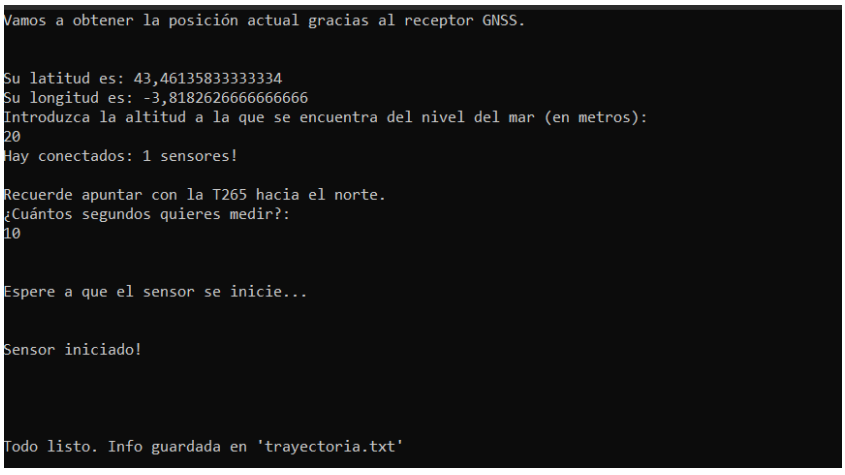


Figura 4.4. Ejemplo de la captación de datos.

En la Figura 4.4 se ve un ejemplo del algoritmo implementado para captar la información dada por el *GNSS* y la posición captada de la *T265* durante, en este caso, $20\ segundos$.

4.2. Procesado y representación de los datos

En esta sección se va a exponer como se han procesado los datos obtenidos en *.NET* de los sensores, así como la representación gráfica de los mismos y las aclaraciones pertinentes sobre ellos.

4.2.1. Procesado

Vamos a hablar primero del procesado de los datos. Los datos han sido procesados con un *script* de *MATLAB*.

Los datos han tenido que ser procesados ya que la *T265* nos da unos datos de posicionamiento locales y el objetivo de este trabajo es conseguir dejar unos datos de localización locales en función de unas coordenadas terrestres. De esta forma tendremos una localización absoluta.

El código de *MATLAB* está dividido en 3 partes. Las que conciernen al procesado son las siguientes:

- **Lectura de ficheros:** Lo primero que se hizo fue preparar los ficheros para que tuvieran un formato valido para la sintaxis de lectura de *MATLAB*. Esto es así ya que *.NET* escribe los números con la coma como separador decimal y *MATLAB* espera que el separador decimal sea un punto.

```
% Leemos el fichero como una serie de cadena de datos
fid = fopen('gps.txt', 'rb');
strings = textscan(fid, '%s', 'Delimiter', '');
fclose(fid);

% Reemplazamos todos las comas por puntos (para el manejo de datos en
% MATLAB)
decimal_strings = regexprep(strings{1}, ',', '.');

% Agrupamos todo en una matriz
data = cellfun(@str2num, decimal_strings, 'uni', 0);
deg = cat(1, data{:});

% Con esto hemos leido la latitud y longitud
```

Figura 4.5. Preparación de los datos para ser leídos correctamente.

Tal como se ve en la Figura 4.5 para los datos provenientes del fichero de la localización se realizó para el fichero de las posiciones obtenidas con la *T265*.

- **Conversión de datos:** Una vez leídos los datos de los ficheros se realizó una conversión de datos de las coordenadas terrestres.

Las coordenadas terrestres según se obtienen del receptor *GNSS u-blox NEO-M8N* están en grados decimales (*DD*), es decir, que estos valores no están en el clásico *grados, minutos, segundos* (*DMS*). Sin embargo, como la posición obtenida con la *T265* esta expresado en coordenadas $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ se realizó una conversión de *DD* a coordenadas *UTM*. Las coordenadas *UTM*, *sistema de coordenadas universal transversal de Mercator*, es un sistema de coordenadas basado en una proyección cartográfica transversa de *Mercator* con la que podemos obtener unas coordenadas $(\mathbf{x}, \mathbf{y})$,las cuales solo son validas si permanecemos en la misma zona *UTM* [24].

Una vez obtenidas las coordenadas *UTM* podemos usarlas como referencia para convertir la trayectoria "local" obtenida por el sensor *T265* en absoluta. para ello se han sumado o restado las componentes de esta trayectoria de forma coherente ya que como se ha comentado en la Subsección 3.2.1 y visto con mas detalle en la Figura 3.8 los ejes de la *T265* están dispuestos según las aplicaciones de *AR/VR*.

```
%% Creamos una versión de los datos en UTM

Lat=deg(1);Lon=deg(2);

[x,y,utmzone] = deg2utm(Lat,Lon);

utm=zeros(ndatos,3);

utm(:,1)=x-datos(:,7);

utm(:,2)=datos(:,5)+y;
utm(:,3)=datos(:,6);
```

Figura 4.6. Conversión de datos *DD* a *UTM* y procesado.

Como se puede ver en la Figura 4.6 se ha usado una función externa a *MATLAB* que transforma las coordenadas en grados decimales, *DD*, en coordenadas *UTM* [25]. Como el eje $\mathbf{z}$ es negativo avanzando hacia la parte delantera del sensor *T265* se ha cogido en todas las posiciones el valor *UTM* perteneciente a la *latitud*, $\mathbf{x}$, y se le ha restado el valor de $\mathbf{z}$ para cada punto de la trayectoria obtenida con el sensor. El eje $\mathbf{x}$ es positivo hacia el "ojo de pez" derecho así que a cada punto de la trayectoria le sumamos el valor *UTM* perteneciente a la longitud, $\mathbf{y}$. El eje $\mathbf{y}$ solo depende de la altitud captada por la *T265*, cuyo valor fue parametrizado como un valor de entrada tal como se ve en la Sección 4.1.

Después estos datos procesados fueron reconvertidos de *UTM* a *DD*.

```
%% Convertimos los datos de UTM procesados a grados decimales (DD)

deg_=zeros(ndatos,3);

for i=1:ndatos

    [Lat,Lon] = utm2deg(utm(i,1),utm(i,2),utmzone);
    deg_(i,1:3)=[Lat Lon utm(i,3)];

end
```

Figura 4.7. Conversión de datos *UTM* a *DD*.

Una vez más se hizo uso de una función externa de *MATLAB*, la cual ahora transforma las coordenadas *UTM* a coordenadas en grados decimales, *DD* [26].

Finalmente estos datos procesados en coordenadas *UTM* y en *DD* fueron escritos en ficheros de texto para poder ser usados posteriormente si fuera pertinente.

4.2.2. Representación de los datos

Una vez procesados los datos el siguiente paso es representarlos. En esta sección vamos a presentar cómo se ha realizado en *MATLAB*, adicionalmente se ha representado la trayectoria encima de un mapa.

```
%% Representamos la trayectoria

figure(1)
comet3(utm(:,1),utm(:,2),utm(:,3));

xlabel('X');ylabel('Z');zlabel('Y');
title('Trayectoria del T265');
```

Figura 4.8. Código para representar la trayectoria.

En la Figura 4.8 se puede ver el código con el cual se representa la trayectoria seguida por el *T265*.

Se ha empleado la instrucción *comet3* para que *MATLAB* vaya trazando en una gráfica la trayectoria en *3D*, de esta forma se ve gráficamente como se va conformando la misma.

Por último, vamos a observar como se ve una trayectoria de ejemplo.

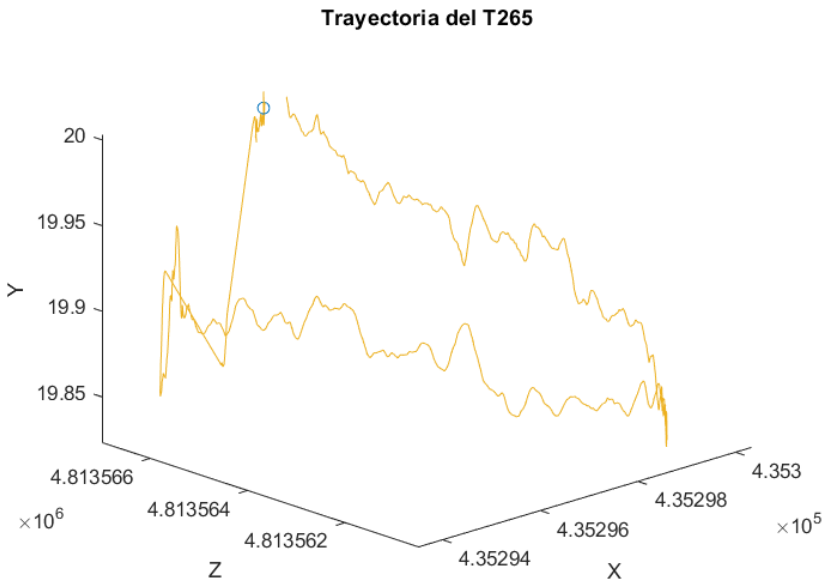


Figura 4.9. Trayectoria descrita por la *T265*.

En la Figura 4.9 podemos observar un ejemplo de trayectoria, la cual tiene sus coordenadas en función de la localización inicial dada por el receptor *GNSS*. Estas coordenadas cómo se han comentado anteriormente son absolutas. Si cogiésemos las coordenadas $(\mathbf{z}, \mathbf{x})$ estas corresponderían a las coordenadas $(\mathbf{x}, \mathbf{y})$ de *UTM*. Sin embargo, para una mejor entrada de datos se ha usado la conversión a coordenadas *DD*, que tenemos en un fichero, para introducirlas en un mapa. La representación de la trayectoria en el mapa se verá en multiples ejemplos a lo largo de la siguiente Sección 4.3.

4.3. Simulaciones y resultados

Una vez presentada la captura de datos en la Subsección 4.1.2 y el procesado de datos y su representación en la Sección 4.2 vamos a abordar a continuación las simulaciones realizadas y los resultados de las mismas. Todas las gráficas que se presentarán a continuación tienen las unidades en metros.

4.3.1. Primera aproximación

En una primera aproximación de las simulaciones, no se disponía del receptor *GNSS* programado. Por lo tanto solo se capturaban y procesaban los datos provenientes de la *T265* y sabiendo las coordenadas de origen con un *smartphone*, se dejó la trayectoria de la *T265* en función de las coordenadas terrestres.

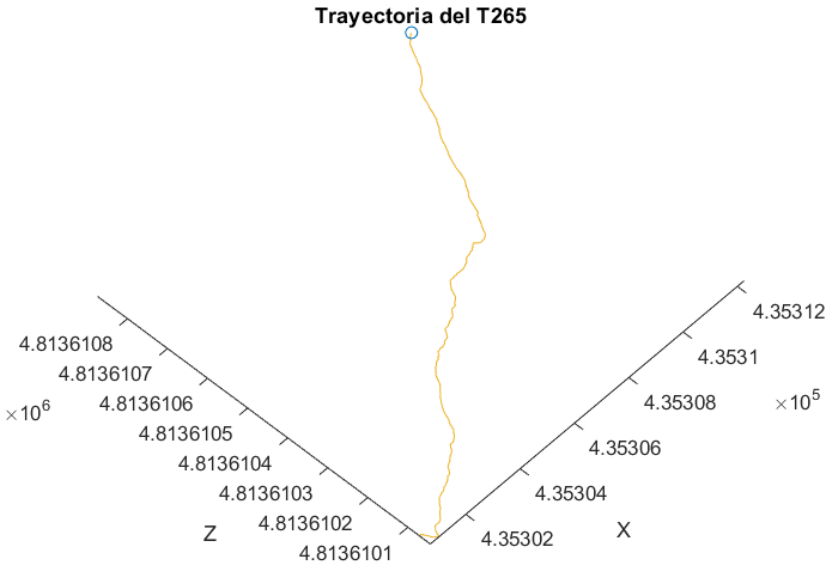


Figura 4.10. Ejemplo 1 de la trayectoria descrita por la *T265*.

En la Figura 4.10 se puede ver una trayectoria descrita al andar en diagonal por el exterior del edificio de I+D+i de telecomunicación de la *UC* junto al grupo de ingeniería fotónica, *GIF*.

Esta fue la primera trayectoria con relevancia ya que previamente lo único que se hacia era moverse pocos centímetros con la *T265* para comprobar que el algoritmo implementado en *.NET* capturaba bien los datos. Esta trayectoria aunque se intentó realizar andando en linea recta por fallo humano no se pudo realizar, la trayectoria en ningún momento se desvió a lo largo del eje $(\mathbf{x})$ de la *T265* mas de $0.6\,m$, puede parecer mucho pero esta dependía de una persona andando sujetando un portátil con la *T265* agarrada a la tapa del mismo, por lo

que era muy difícil mantener siempre el portátil en la misma posición y andar completamente recto.

En la siguiente figura, la Figura 4.11, se puede observar otro ejemplo de trayectoria.

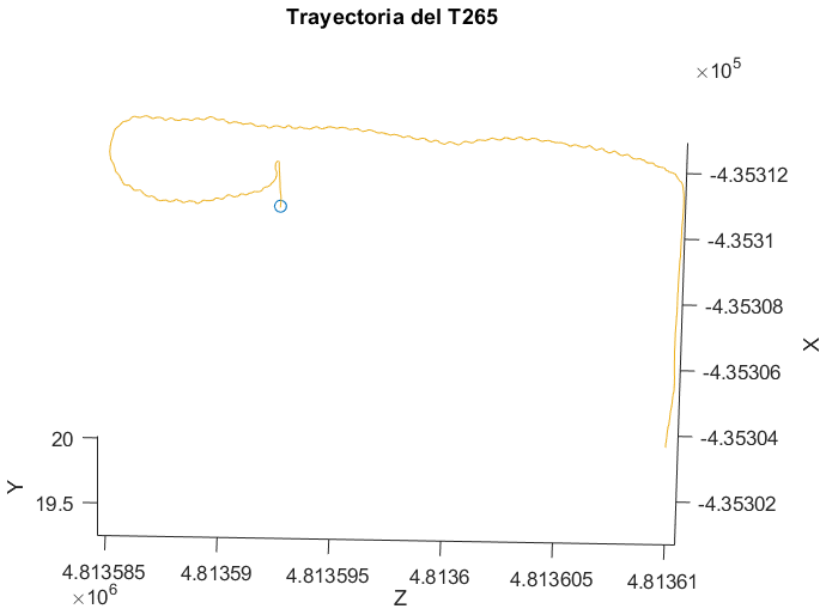


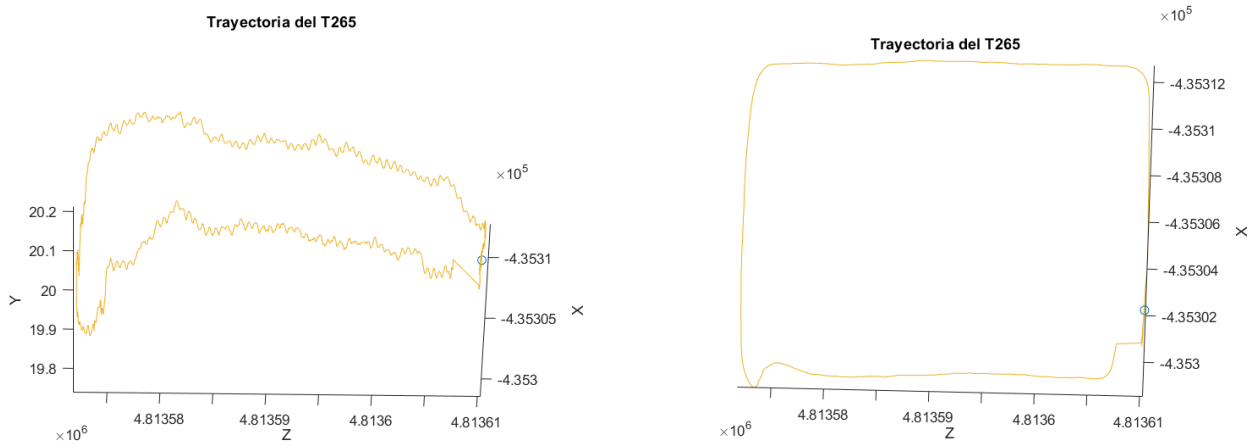
Figura 4.11. Ejemplo 2 de la trayectoria descrita por la *T265*.

En este caso la trayectoria descrita era por dentro del *GIF* entrando por la puerta del departamento, recorriendo el pasillo hacia la izquierda, entrando al despacho de los *doctorandos* y sentándose en un sitio dentro de el despacho. Esta trayectoria es tal cual a cómo es en la realidad, comentar que la bajada en la trayectoria en el último punto (circulo azul) es debido a la diferencia de altura que tuvo el portátil al posarlo en una mesa.

4.3.2. Trayectorias relevantes

Una vez realizadas las primeras simulaciones funcionales se procedió a realizar trayectorias mas largas y complejas para comprobar la viabilidad de la implementación.

En la siguiente ejemplo vamos a ver un recorrido realizado también en el *GIF*.



(a) Vista en perspectiva, *3D*

(b) Vista desde arriba, *2D*

Figura 4.12. Ejemplo 3 de la trayectoria descrita por la *T265*.

En la Figura 4.12 se puede ver una trayectoria bastante importante. Esta se realizó entrando por el *GIF*, recorriendo el pasillo de la izquierda y saliendo por una puerta que da la calle antes de llegar a los servicios. Lo que se pretendía con esta prueba era realizar una trayectoria completa, guiándose con la baldosas del suelo y volviendo a un punto inicial de referencia y ver si la trayectoria captada tenia la forma de "rectángulo" y volvía al mismo punto. Cómo se puede ver esto sucedió. en la Figura 4.12a se puede ver en perspectiva la altura y cómo esta varió solo $30\,cm$ debido al error humano de llevar el portátil en una mano y con otra intentar abrir una puerta por la que salir a la calle.

El siguiente ejemplo es el último que se hace sin el uso del receptor *GNSS* y por lo tanto sin la representación de la trayectoria encima de un mapa.

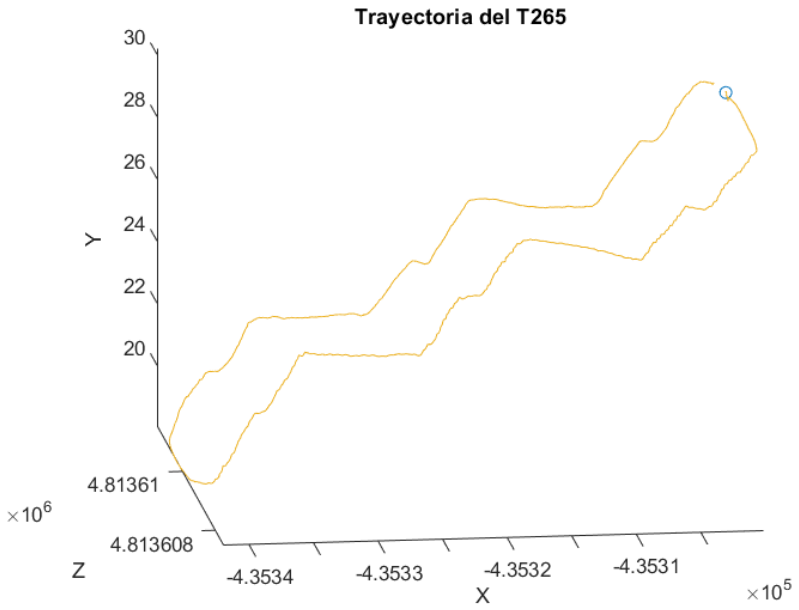


Figura 4.13. Ejemplo 4 de la trayectoria descrita por la *T265*.

En la Figura 4.13 podemos ver la trayectoria descrita al bajar las escaleras de la plaza de la ciencia desde la entrada superior hasta el inicio de la planta -3. Podemos ver en la trayectoria lo bien marcadas que están las plantas y los descansillos intermedios entre planta y planta.

En el siguiente ejemplo vamos a ver cómo una trayectoria descrita por la *T265* ha sido representada encima de un mapa.

Como se ha comentado anteriormente, se escribieron los datos procesados en coordenadas *UTM* y en coordenadas *DD* en ficheros de texto. El fichero de las coordenadas *DD*, llamado "deg.txt" de la trayectoria descrita en la Figura 4.9 se introdujo en *Google Earth* para poder ver la trayectoria dibujada en un mapa.

Figura 4.14. Representación de la trayectoria en *Google Earth*.

Tal como se puede ver en la Figura 4.14 si usamos la trayectoria obtenida con la *T265* y la dejamos en función de las coordenadas terrestres gracias a la localización dada por el receptor *GNSS u-blox NEO-M8N*, conseguimos representar en el mapa la trayectoria como si esta hubiese salido solo de un sistema de posicionamiento *GNSS*. Sin embargo, no es perfecta esta trayectoria, ya que está desviada aproximadamente $1.5\,m$ ya que la posición inicial dada por el receptor *GNSS* no era exactamente esa, difería aproximadamente ese metro y medio mencionado.

4.3.3. Simulaciones acuáticas

En este apartado vamos a mencionar la viabilidad de utilizar esta propuesta en un medio acuoso, en concreto el mar o una ría.

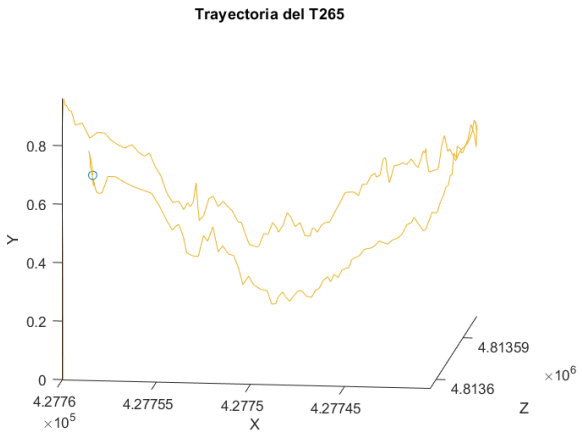
Paralelamente a la realización de las simulaciones se estaban realizando pruebas con otra *T265*, solo que esta es sumergible.

Figura 4.15. *T265* sumergible.

La *T265* que se ve en la Figura 4.15 ha sido introducida en un armazón estanco para que no le entre agua. Está controlada por una *Raspberry* y como método de interacción con ella tiene unos pulsadores magnéticos, ya que la conexión con cables desde fuera es inviable ya que va a estar sumergida en el agua. Por falta de espacio esta *T265* no iba complementada con el receptor *GNSS*, sin embargo, se utilizó el receptor *GNSS* de un *smartphone* y se anotó en que coordenadas se estaban en el momento de iniciar esta *T265*, por lo que el procesado y representación en el mapa a efectos prácticos es el mismo.

A continuación vamos visualizar las simulaciones mas relevantes con esta *T265* especial.

(a) Vista de la trayectoria en *Google Earth*, *3D*



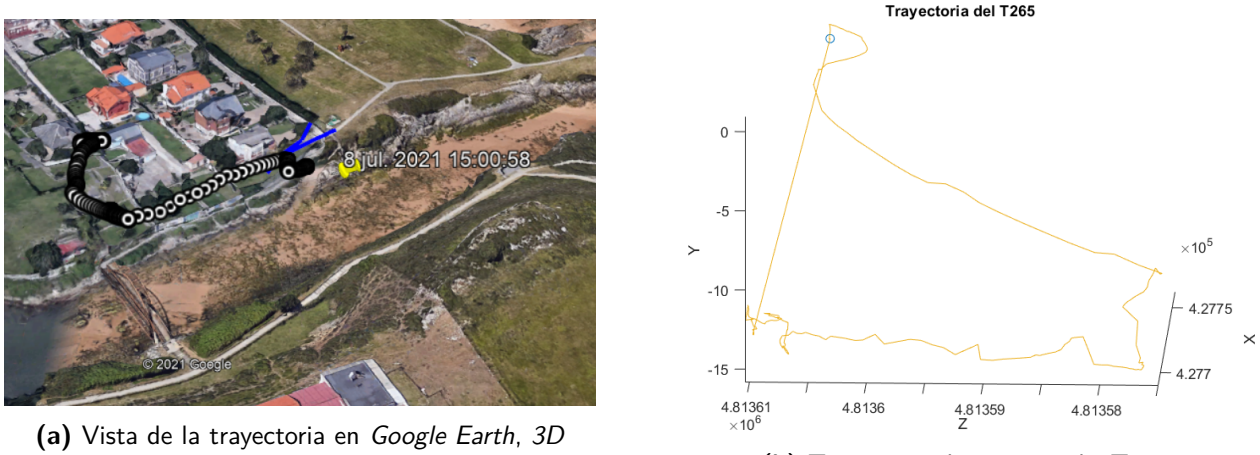
(b) Trayectoria descrita por la *T265*

Figura 4.16. Trayectoria descrita fuera del agua por la *T265* en la ría de San Juan de la Canal.

En la Figura 4.16 podemos ver que trayectoria ha descrito la *T265* sin haber entrado en al agua todavía. Es relevante esta simulación ya que vemos que funciona en otro ambiente totalmente distinto que donde se probó originalmente, el edificio de I+D+i.

En la Figura 4.16a podemos ver complementado con la Figura 4.16b cómo se movió por una zona rocosa y luego se volvió prácticamente al mismo punto.

Una prueba de la viabilidad de usar la *T265* debajo del agua es la siguiente.



(a) Vista de la trayectoria en *Google Earth*, *3D*

(b) Trayectoria descrita por la *T265*

Figura 4.17. Trayectoria descrita dentro del agua por la *T265* en la ría de San Juan de la Canal.

Como se puede observar en la Figura 4.17 al entrar dentro del agua la *T265* no es capaz de aplicar bien el algoritmo *SLAM* ya que por sus cámaras "ojos de pez" esta viendo pasar partículas a una velocidad moderada por efecto de la corriente del agua. Esto provoca que la *T265* considere que se está moviendo de forma anómala cuando en realidad no es así. La trayectoria descrita en la Figura 4.17a dista mucho de la que se realizó, la cual fue dentro de la ría (zona marrón). Así mismo, en la Figura 4.17b se puede ver cómo se ha llegado a bajar $-15\,m$, algo que es imposible y mas teniendo en cuenta el supuesto recorrido que ha hecho la *T265*.

Por tanto, se puede afirmar que para aplicaciones terrestres en los que el medio siga siendo el aire esta propuesta funciona. Pero tal y como se ha demostrado, en medios acuáticos no funciona. Pudiera ser que si la turbidez del agua sea mínima y que en situaciones muy concretas y controladas pudiera funcionar, sin embargo, no se ha llegado a profundizar tanto.

En el siguiente capítulo se va a proceder a concluir el trabajo y hablar de posibles lineas futuras del mismo.

5

Conclusiones y líneas futuras

Este último capítulo resume las principales conclusiones de este Trabajo de Fin de Grado, en base a los resultados obtenidos en el Capítulo 4. Además, se introducirán posibles líneas de investigación futura.

5.1. Conclusiones

En el Capítulo 1 se presentaron unos objetivos de este trabajo. Los cuales abarcaban la captura de datos y el posterior calculo de la trayectoria y la representación de la trayectoria en un mapa.

Estos objetivos se han logrado alcanzar. Pasando por distintas fases y pruebas se ha conseguido calcular una trayectoria y dejándola en función de las coordenadas terrestres representarla tanto en *MATLAB* como en un mapa, en concreto en *Google Earth*.

Así mismo, se han comprendido los conceptos físicos y de los sistemas *GNSS*, que propiedades tienen y cuales son sus principales problemáticas. También se ha entendido el *V/SLAM* y la potencialidad del mismo.

Por otra parte se ha obtenido el razonamiento crítico necesario para discernir cuando una trayectoria era correcta o no. En base a esto se puede afirmar que en ambientes terrestres la propuesta de este trabajo, *T265* + *NEO-M8N*, consigue trazar una trayectoria correcta con un error aproximado de $1.5\,m$ en las pruebas realizadas.

Aunque el nombre de este trabajo haga mención solo a *posicionamiento indoor* lo cierto es que como se ha comprobado en la Sección 4.3 que funciona también en entornos *outdoor*. Cabe destacar que la posición inicial dada por el receptor *GNSS u-blox NEO-M8N* solo estará disponible en exteriores con el cielo descubierto. Si esta propuesta se aplicase en *indoor* este receptor no obtendría la localización. Sin embargo, se podría usar otro o conocidas unas coordenadas de referencia inicial usarlas directamente en el procesado en *MATLAB*.

Por último mencionar que no se ha conseguido que funcione dentro del agua ni una vez, por lo que asumimos que no funciona en el agua o que la condición del agua en el que se comprobó no es valida.

5.2. Líneas futuras

Aunque los resultados obtenidos son completos y validos en esta sección se van a presentar potenciales líneas de investigación futura.

Bien es cierto como se ha comentado en la sección anterior que los objetivos marcados para este trabajo se han cumplido, pero aún así siempre hay un margen de mejora. El método de procesado de los datos es valido, pero para según que tipo de aplicaciones puede que no sea suficiente. Una posible línea futura sería obtener convertir esta propuesta en un sistema telemétrico, es decir, obtener datos de seguimiento, procesarlos en tiempo real y observarlos a distancia. De esta forma si se implementase, por ejemplo, en un *robot autónomo* podríamos ver en tiempo real donde esta y a la vez tomar decisiones sobre que tarea podría realizar a continuación.

Otra posible línea de investigación es la de utilizar un sensor adicional, un sensor de profundidad, mismamente de la familia *Intel®Realsense™* o cualquier otro del mercado. Con esto conseguiríamos no solo obtener la trayectoria de un *móvil* si no que además podríamos reconstruir paralelamente en *3D* el entorno por el que se esta moviendo dicho *móvil*.

Adicionalmente, sería interesante comprobar si realmente en medios acuáticos esta propuesta con la *T265* no funciona, para ello habría que hacer pruebas en medios menos turbios, es decir, menos partículas en suspensión. Si se confirmase que podría llegar a funcionar se abriría una linea de investigación muy potente ya que esto podría impulsar la navegación acuática autónoma.

Por último, hubiese estado bien acabar creando una aplicación sencilla con la cual controlar gráficamente los sensores, procesar los datos y visualizar la información fácilmente.

Bibliografía

[1] T. Alsop, "Navigation devices & services." https://www.statista.com/topics/2221/navigation-devices-and-usage/, Nov 2020. [Online; Ultimo acceso 25-Junio-2021].

[2] GISGeography, "Gps accuracy: Hdop, pdop, gdop, multipath & the atmosphere." https://gisgeography.com/gps-accuracy-hdop-pdop-gdop-multipath/, Jun 2021. [Online; Ultimo acceso 25-Junio-2021].

[3] N. Karlsson, E. di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. Munich, "The vslam algorithm for robust localization and mapping," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 24–29, 2005.

[4] Intel, "Tracking camera t265." https://www.intelrealsense.com/tracking-camera-t265/, May 2021. [Online; Ultimo acceso 25-Junio-2021].

[5] GisResources, "What is slam algorithm and why slam matters?." https://www.gisresources.com/what-is-slam-algorithm-and-why-slam-matters/, Oct 2020. [Online; Ultimo acceso 2-Julio-2021].

[6] Intel, "Lidar cameras, stereo depth cameras, coded light and tracking cameras from intel realsense." https://www.intelrealsense.com/, Jun 2021. [Online; Ultimo acceso 12-Julio-2021].

[7] G. Brahmanage and H. Leung, "Outdoor rgb-d mapping using intel-realsense," in *2019 IEEE SENSORS*, pp. 1–4, 2019.

[8] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, "Intel realsense stereoscopic depth cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[9] BrainVoyager, "Spatial transformation matrices." https://www.brainvoyager.com/bv/doc/UsersGuide/CoordsAndTransforms/SpatialTransformationMatrices.html. [Online; Ultimo acceso 17-Julio-2021].

[10] GISGeography, "How gps receivers work - trilateration vs triangulation." https://gisgeography.com/trilateration-triangulation-gps/, Jun 2021. [Online; Ultimo acceso 18-Julio-2021].

[11] GISGeography, "World geodetic system (wgs84)." https://gisgeography.com/wgs84-world-geodetic-system/, Jun 2021. [Online; Ultimo acceso 18-Julio-2021].

[12] Wikipedia, "Geographic information system." https://en.wikipedia.org/wiki/Geographic_information_system, Jun 2021. [Online; Ultimo acceso 18-Julio-2021].

[13] Wikipedia, "Gps signals." https://en.wikipedia.org/wiki/GPS_signals, Jun 2021. [Online; Ultimo acceso 18-Julio-2021].

[14] IntelRealSense, "Intelrealsense/librealsense." https://github.com/IntelRealSense/librealsense/blob/master/doc/t265.md, Mar 2020. [Online; Ultimo acceso 19-Julio-2021].

[15] u blox, "Neo-m8 series." https://www.u-blox.com/en/product/neo-m8-series#tab-documentation-resources, Apr 2021. [Online; Ultimo acceso 20-Julio-2021].

[16] E. StackExchange, "Warm/hot starting a gsm/gps module." https://electronics.stackexchange.com/questions/230305/warm-hot-starting-a-gsm-gps-module. [Online; Ultimo acceso 21-Julio-2021].

[17] Epson, "Tcxo (temperature compensated crystal oscillator)." https://www5.epsondevice.com/en/products/tcxo/. [Online; Ultimo acceso 20-Julio-2021].

[18] everything RF, "What is a tcxo - everything rf." https://www.everythingrf.com/community/what-is-a-tcxo. [Online; Ultimo acceso 20-Julio-2021].

[19] GPSWORLD, "What exactly is gps nmea data?." https://www.gpsworld.com/what-exactly-is-gps-nmea-data/, May 2016. [Online; Ultimo acceso 21-Julio-2021].

[20] Wikipedia, "Real-time kinematic positioning." https://en.wikipedia.org/wiki/Real-time_kinematic_positioning, Jun 2021. [Online; Ultimo acceso 21-Julio-2021].

[21] Wikipedia, "Differential gps." https://en.wikipedia.org/wiki/Differential_GPS, Jun 2021. [Online; Ultimo acceso 21-Julio-2021].

[22] dotMorten, "dotmorten/nmeaparser: Library for handling nmea message.." https://github.com/dotMorten/NmeaParser. [Online; Ultimo acceso 22-Julio-2021].

[23] IntelRealSense, "librealsense/wrappers/csharp at master intelrealsense/librealsense." https://github.com/IntelRealSense/librealsense/tree/master/wrappers/csharp. [Online; Ultimo acceso 22-Julio-2021].

[24] Wikipedia, "Universal transverse mercator coordinate system." https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system, Jun 2021. [Online; Ultimo acceso 22-Julio-2021].

[25] R. Palacios, "deg2utm." https://es.mathworks.com/matlabcentral/fileexchange/10915-deg2utm. [Online; Ultimo acceso 22-Julio-2021].

[26] R. Palacios, "utm2deg." https://es.mathworks.com/matlabcentral/fileexchange/10914-utm2deg. [Online; Ultimo acceso 22-Julio-2021].