Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA EN SISTEMAS DE TELECOMUNICACIÓN

Trabajo Fin de Grado

Optimización de la etapa de pre-procesado de la señal GPS en sistemas radar pasivos

Autor: Oscar Vinicio Peña Quezada

Tutor/es: María Pilar Jarabo Amores

Cotutor: Pedro José Gómez del Hoyo

2019

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

GRADO EN INGENIERÍA EN SISTEMAS DE TELECOMUNICACIÓN

Trabajo Fin de Grado

Optimización de la etapa de pre-procesado de la señal GPS en sistemas radar pasivos

Autor: Oscar Vinicio Peña Quezada

Tutora: María Pilar Jarabo Amores

Cotutor: Pedro José Gómez del Hoyo

TRIBUNAL:

Presidente: José María Muñoz Ferreras

Vocal 1º: Judith María Redoli Granados

Vocal 2º: María Pilar Jarabo Amores

FECHA: 14/11/2019

“La vida es una continua superación”

Oscar Peña

Agradecimientos

En primer lugar, quiero agradecer a mi familia: a mi padre Luis Vinicio Peña Peña, a mi madre Aida Narcisa Quezada Villavicencio y a mi hermana Allison Vanessa Peña Quezada, que siempre me han brindado apoyo y unas palabras de ánimo durante todo este tiempo. Sin ellos, colocar la última teja del tejado no habría sido posible por lo que se merecen aparecer en esta cita de agradecimiento.

También me gustaría recordar en estas líneas a mucha gente que mencionaré como si fuese una lista de la compra pero no por ello con menos gratitud: a mis primos/as María Alicia Aguilar Peña, Josué Leonardo Cabrera Quezada, Viviana Durvy Cabrera Quezada, Bryan Antonio Quezada Reinoso, Gustavo Mauricio Quezada Reinoso; tíos/as Gustavo Quezada Villavicencio, Elena Quezada Villavicencio, Laura Virginia Peña Peña; y amigos, Eduardo Castillo Jiménez, Carlos Castillo Jiménez, Jose Antonio Delgado Urbina, Raúl Izquierdo Iniesta y Yeray Pérez Moreno; gracias por estar ahí.

Y en especial me gustaría dedicarles unas palabras de agradecimiento a ellos, mis compañeros de camino, mis compañeros de estudio, mis compañeros de batallas, mis compañeros de desayunos, mis compañeros de almuerzos, mis compañeros de cenas, mis compañeros de estrés, mis profesores, mis alumnos, mis compañeros de suspensos pero sobre todo mis compañeros de aprobados, mis compañeros que se han convertido en amigos de vida: Alejandro Santo Tomás Rodríguez, Francisco Javier Posada González, Francisco Javier la Roda (al que bendije como el Sabio), Juan José Rojo Sánchez, Eduardo Mir Carnicero, Carlos Sánchez Monedero, Ernesto Carrasco Rivillos y Mohamed Malki.

Para terminar, me gustaría reconocer la oportunidad que me ha ofrecido mi tutora para realizar este TFG: María Pilar Jarabo Amores; y la paciencia que ha tenido mi Cotutor en la confección de este documento.

Índice General

Índice de figuras

Índice de tablas

Lista de acrónimos

ADC. *Analog-to-Digital Converter*

AFS. *Atomic Frequency Standard*

AMCS. *Alternate Master Control System*

BER. *Bit Error Rate*

BPSK. *Binary Phase Shift Keying*

CAF. *Cross-Ambiguity Function*

CDMA. *Code Division Multiple Access*

CUDA. *Compute Unified Device Architecture*

DAB. *Digital Audio Broadcasting*

DFT. *Discrete Fourier Transform*

DLL. *Delay Lock Loop*

DME. *Distance Measuring Equipment*

DPI. *Direct Path Interference*

DRM. *Digital Radio Mondiale*

DSSS. *Direct Sequence Spread Spectrum*

DVB-S. *Digital Video Broadcast – Satellite*

DVB-S2. *Digital Video Broadcast – Satellite, second generation*

DVB-T. *Digital Video Broadcast - Terrestrial*

GNSS. *Global Navigation Satellite System*

GPS. *Global Positioning System*

GSM. *Global System for Mobile communications*

HOW. *Hand-Over Word*

IDEPAR. Improved Detection techniques for Passive Radars

IDFT. *Inverse Discrete Fourier Transform*

IO. *Illuminators of Opportunity*

LNA. *Low Noise Amplifier*

LPF. *Low-Pass Filter*

LTE. *Long Term Evolution*

MCS. *Master Control Station*

MEO. *Medium Earth Orbit*

NRT. *Near Real Time*

PLL. *Phase Lock Loop*

PPS. *Precision Positioning Service*

PR. *Passive Radar*

PRN. *Pseudo-Random Noise*

PSD. *Power Spectral Density*

Radar. *Radio Detection and Ranging*

RDM. *Range-Doppler Map*

SNR. *Signal-Noise Ratio*

SoL. *Safety of Life*

SPS. *Standard Positioning Service*

TACAN. *Tactical Air Navigation*

TLM. *Telemetry*

TOW. *Time of Week*

UAV. *Unmanned Aerial Vehicle*

UMTS. *Universal Mobile Telecomunications System*

US. *United States*

UTC. *Universal Time Coordinated*

WiFi. *Wireless Fidelity*

Glosario

Banda L. *Banda de frecuencia sobre la que actúa la tecnología GPS*

Clutter. *Ecos no deseados*

Disturbance Cancellation. *Filtrado de la señal de vigilancia en el cual se busca eliminar las contribuciones interferentes de la señal, clutter pero sobre todo DPI*

Local oscillator. *Generador local de portadoras*

Órbita geoestacionaria. *órbita en la que el satélite aparece como un punto fijo en el firmamento. Para ello, el satélite se debe encontrar sobre una órbita circular sobre el Ecuador y que su periodo de traslación sea exactamente igual al de rotación de la Tierra*

PRN code generator. *Generador de secuencias de código PRN*

Reference Channel. *Canal de Referencia del receptor de radar pasivo*

RF. *Radiofrecuencia*

Surveillance Channel. *Canal de Vigilancia del receptor de radar pasivo*

Target. *Objeto de búsqueda del Radar*

Threshold. *Umbral de correlación. Es una medida de calidad de la etapa de adquisición*

Resumen

El inconveniente de los sistemas de radar pasivo basados en tecnología radar/radio definida por software es la adquisición y el procesamiento de las señales con latencias compatibles con los requisitos de tiempo real impuestos por las aplicaciones de interés. En este Trabajo Fin de Grado se ha realizado un estudio sobre la optimización de los algoritmos implementados en la etapa de pre-procesado del demostrador IDEPAR para la utilización de las señales transmitidas por satélite GPS (Global Positioning System), con el fin de mejorar los tiempos de procesado y reducir el gap que hoy en día se aleja de cumplir los requisitos de tiempo real. Para la optimización, se han aplicado distintas técnicas: pre-asignación de memoria, vectorización y paralelización de código. Como resultado, se ha mejorado el sistema con una reducción significativa del tiempo de procesamiento, cumpliendo los objetivos definidos.

Palabras clave: IDEPAR, radar pasivo, constelación GPS, señal GPS, pre-asignación de memoria, vectorización y paralelización de código.

Abstract

The issue of passive radar systems implemented by means of software defined Radar/Radio technologies is the signal acquisition and processing with latency, compatible with de real-time requirements imposed by the applications of interest. In this final project a study on the optimization of the algorithms implemented in the pre-processing stage of the demonstrator IDEPAR for the use of signals transmitted by satellite GPS (Global Positioning System) has been performed, for the purpose of improving the processing time and reducing the gap that nowadays moves away from the real-time requirements. Different techniques for optimization have been implemented as pre-allocating memory, vectorization and code parallelization. As a result, the system has been improved with a significant processing time reduction that meets the marked objectives.

Keywords: IDEPAR, passive radar, GPS constellation, GPS signals, pre-allocating memory, vectorization and code parallelization.

Resumen extendido

La creciente demanda en prevención de amenazas, organización de tráfico, tanto aéreo como terrestre o marítimo, control meteorológico o vigilancia de infraestructuras, entre otras aplicaciones, ha generado el incremento de desarrollo de sensores y radares de diferentes tipos. Donde destaca el desarrollo de radares que emplean transmisores no dedicados, denominados radares pasivos.

Estos radares suprimen el desarrollo de hardware para fines de transmisión, evitando problemas asociados como la radiación de energía electromagnética y la necesidad de asignación de frecuencias. Debido a la ausencia de un transmisor dedicado, los gastos de desarrollo, mantenimiento y despliegue son menores que los de los radares activos, haciendo que el radar pasivo se encuentre en gran auge, puesto que existe multitud de tecnología que se puede aprovechar como IO: la radio analógica, la radio digital, la televisión digital terrestre o su homólogo satelital, la telefonía móvil, los sistemas inalámbricos WIFI o los sistemas de localización por satélite.

Luego para los sistemas de radar pasivo es necesario desarrollar tecnología radar/radio definida por software compatible con la multitud de tecnologías empleadas como transmisor no dedicado. El principal inconveniente de los sistemas de radar pasivos es el tiempo de adquisición y procesamiento de las señales con latencias.

Por esta razón, este Trabajo de Fin de Grado nace con la necesidad de mejorar este inconveniente. Se realiza un estudio sobre la optimización e integración de los algoritmos implementados en la etapa de pre-procesado del demostrador IDEPAR para la utilización de las señales transmitidas por satélite GPS, con el fin de mejorar los tiempos de procesado y reducir el gap que hoy en día se aleja de cumplir los requisitos de aplicaciones de tiempo real.

Las señales GPS presentan algunas ventajas como el empleo de altas frecuencias de trabajo, la compartición de la misma banda L de frecuencias por todos los satélites, el uso de señales de espectro ensanchado que presentan mejor inmunidad frente a interferencias rápidas o la constelación de satélites con cobertura global.

Para la adaptación de la señal GPS como transmisor no dedicado es necesario modificar el demostrador IDEPAR, incorporando una nueva etapa de pre-procesado de la señal debido al uso compartido de frecuencia, la baja potencia de la señal recibida y la geometría variable del sistema. Este último inconveniente genera variaciones en las componentes de la señal como retardos de fase del código PRN (Pseudo-Random Noise) y desplazamiento Doppler en la frecuencia de portadora que son necesarias estimar.

Esta etapa de pre-procesado incorpora los bloques de adquisición y tracking, cuyo elevado tiempo de procesamiento aleja el demostrador IDEPAR de los requisitos de sistemas Near Real Time (NRT).

Los objetivos de este TFG se centran, por tanto, en el estudio, desarrollo e integración de distintas técnicas para la aceleración del código implementado en MATLAB del demostrador IDEPAR, con el fin de reducir el gap que le separa de una solución compatible con los requisitos de tiempo real exigidos en aplicaciones de vigilancia.

Para ello se han estudiado distintas técnicas, destacando la pre-asignación de memoria, la vectorización y la paralelización de código; que se han implementado sobre los algoritmos de adquisición y tracking en lenguaje de MATLAB, operativos en IDEPAR en el momento de comenzar este TFG. Además, se ha realizado un estudio mediante señales filtradas que comprueba el comportamiento de los algoritmos de adquisición y de tracking y revelar si realmente genera influencia sobre estos algoritmos.

Como resultado del estudio de implementación de algoritmos con señales filtradas se observa que no provoca ninguna variación en las prestaciones de los algoritmos. Por otra parte, en el estudio de rendimiento de los códigos que incorporan las técnicas anteriormente nombradas se observa un gran incremento de las prestaciones temporales, destacándose el empleo de funciones optimizadas sobre la GPU, es decir, paralelización de código que permite el software MATLAB. Tanto en las etapas de adquisición como de tracking de la señal GPS se consiguen grandes reducciones de tiempo. Como resultado, se ha mejorado el sistema con una reducción significativa del tiempo de procesamiento, cumpliendo el objetivo de aceleración de código para acercar el sistema del demostrador IDEPAR a sistemas NRT.

Dado que las funciones optimizadas sobre la GPU son funciones pertenecientes a la programación paralela, aparece una nueva incógnita de estudio que es la programación paralela sobre la plataforma de computación paralela CUDA; que es un modelo de programación desarrollado por NVIDIA para la computación sobre la GPU que permite una aceleración drástica de las aplicaciones informáticas. Luego se propone el estudio de un nuevo desarrollo sobre esta plataforma aplicando las técnicas vistas, de los algoritmos aplicados en este TFG.

Otra línea futura de gran interés es el estudio de la posibilidad de aplicar las técnicas estudiadas al procesado de otras señales. IDEPAR puede adquirir señales multicanal de la televisión digital terrestre y satelital. En las primeras, se han propuesto algoritmos de reconstrucción de la señal para la implementación de la etapa de rechazo de interferencias previa a la correlación cruzada. Estos algoritmos tienen un elevado coste computacional asociado que podría reducirse utilizando las técnicas estudiadas.

El procesado de señales multicanal y con elevados tiempos de integración, también requieren muchos recursos de cómputo. Se han propuesto metodologías de paralelización en tiempo y frecuencia, que podrían mejorarse mediante la aplicación de las técnicas estudiadas.

1 Introducción

Como bien es cierto las sociedades modernas presentan una creciente demanda en prevención de amenazas, organización de tráfico, tanto aéreo como terrestre o marítimo, control meteorológico o vigilancia de infraestructuras, entre otras aplicaciones. Para ello se ha generado un gran desarrollo de sensores y radares de diferentes tipos. Cada solución aporta ciertas ventajas como, por ejemplo, el uso de sensores ópticos consigue un buen rendimiento en cobertura de corta y media distancia, pero que si le añadimos condiciones climatológicas adversas o de baja iluminación su rendimiento decrece; o el uso de radares convencionales con mejor rendimiento que los sensores ópticos, o el uso de radares pasivos, que emplean transmisores no dedicados o Iluminadores de oportunidad (IO), suprimiendo el desarrollo de hardware para tales fines, evitando problemas asociados la radiación de energía electromagnética y la necesidad de asignación de frecuencias.

Debido a la ausencia de un transmisor dedicado, los gastos de desarrollo, mantenimiento y despliegue son menores que los de los radares activos, haciendo que el radar pasivo se encuentre en gran auge, puesto que existe multitud de tecnología que se puede aprovechar como IO: la radio analógica, la radio digital, la televisión digital terrestre o su homólogo satelital, la telefonía móvil, los sistemas inalámbricos WIFI o los sistemas de localización por satélite.

En concreto, en los sistemas de localización por satélite, destacan las ventajas del Sistema de Posicionamiento Global (GPS, Global Positioning System) como IO, como el empleo de altas frecuencias de trabajo, la compartición de la misma banda L de frecuencias por todos los satélites, el uso de señales de espectro ensanchado que presentan mejor inmunidad frente a interferencias o la cobertura global. La constelación de satélites GPS cubre el globo completo con un total de 32 satélites sobre órbitas intermedias (MEO, Medium-Earth-Orbit).

Debido a las ventajas que presenta GPS como IO se ha modificado el demostrador IDEPAR para aprovechar este tipo de señal. IDEPAR es un radar pasivo diseñado y desarrollado por el grupo de investigación dirigido por la Dra. Mª del Pilar Jarabo Amores en la Universidad de Alcalá y financiado por diversos proyectos nacionales.

Los radares pasivos están constituidos de forma general por una etapa de adquisición y una etapa de procesado. La etapa de adquisición se corresponde con la sección hardware del demostrador encargada de adquirir y preparar la señal para un posterior procesado, con las antenas receptoras, el front-end de RF y las tarjetas de adquisición. La etapa de procesado se corresponde con la sección software donde se realiza el procesado coherente de las señales digitalizadas cuyo objetivo es detectar la presencia de los blancos y extraer propiedades como su posición y velocidad.

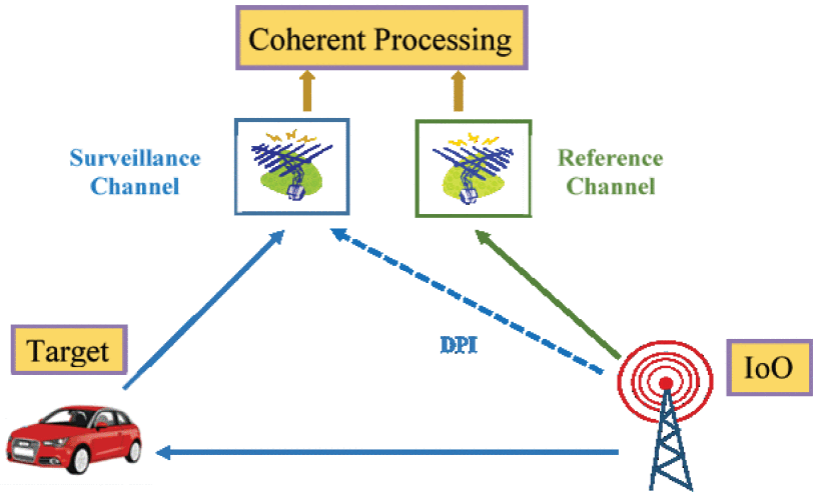

Ilustración 1-1. Geometría general de Radar Pasivo Biestático [1]

En la Ilustración 1-1, se observa la geometría general del radar pasivo biestático. Se le denomina biestático debido a que el transmisor (IoO) y el receptor se encuentran en localizaciones diferentes. Al no tener control sobre la señal transmitida, es necesario incorporar dos canales en el receptor: el canal de referencia (Reference Channel) y el canal de vigilancia (Surveillance Channel). El canal de referencia es el encargado de adquirir la señal del IO y el canal de vigilancia es el encargado de adquirir la señal de los ecos de los blancos (Target). La sección hardware viene representada por las antenas receptoras, el front-enf de RF y las tarjetas de adquisición de los canales de referencia y de vigilancia; la sección software viene representada por el procesado coherente.

Un gran inconveniente en este tipo de sistemas es la recepción de interferencia DPI (Direct Path Interference) a través del canal de vigilancia.

En la Ilustración 1-2, se observa el esquema de procesado general de un radar pasivo:

- Filtrado de DPI sobre la señal de vigilancia (Disturbance Cancellation).
- Correlación cruzada de las señales de referencia y vigilancia (range-Doppler Cross-Correlation).
- Detección de blancos (Targets Detection) a partir del mapa rango-Doppler (RDM, Range-Doppler Map).
- Seguimiento de los blancos (Target Tracking).

Como las componentes de DPI tienen mayor potencia que los ecos de los blancos, es necesario eliminarlas antes de la correlación cruzada entre la señal de referencia y la señal de vigilancia.

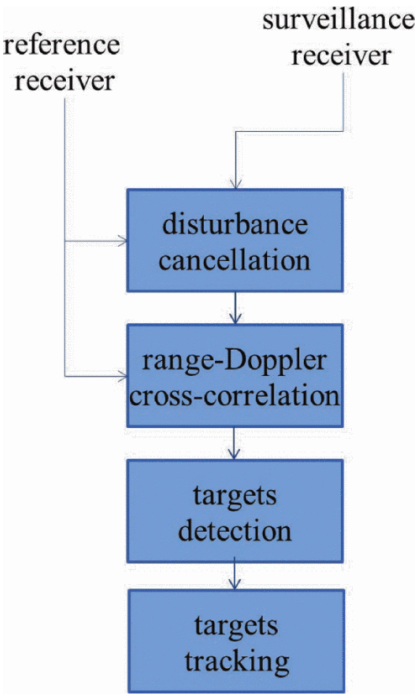

Ilustración 1-2. Esquema básico de la etapa de procesado de un radar pasivo [2]

Los satélites de la constelación GPS comparten la banda L de frecuencias y transmiten una señal de espectro ensanchado que incluye un mensaje de datos de navegación con información acerca de la órbita, el estado del satélite y datos de corrección. El espectro ensanchado permite transmitir señales con muy poca potencia en un amplio ancho de banda. En el caso de la señal GPS, se consigue gracias a una modulada con códigos PRN (Pseudo-Random Noise).

En la Ilustración 1-3, se observan las componentes de la señal GPS: una portadora (a), un mensaje de navegación (b) y un código PRN (c). El resultado de la modulación (d) de la portadora con los datos de navegación y el código PRN. Esta modulación es posible gracias a que emplea CDMA (Code Division Multiple Access) que es una técnica de acceso múltiple que permite identificar a cada usuario mediante códigos únicos en un mismo canal de comunicación, en este caso una misma portadora.

Por tanto, al utilizar el sistema de GPS como IO en el demostrador IDEPAR, se genera la necesidad de pre-procesar la señal debido al uso compartido de frecuencia, la baja potencia de señal recibida y la geometría variable del sistema. Este último inconveniente genera variaciones en las componentes de la señal, debido a retardos de fase del código PRN asociados a la distancia entre el satélite y el receptor de radar pasivo, y a desplazamientos Doppler en la frecuencia de portadora producidos por el movimiento del satélite, que no se encuentra en la órbita geoestacionaria.

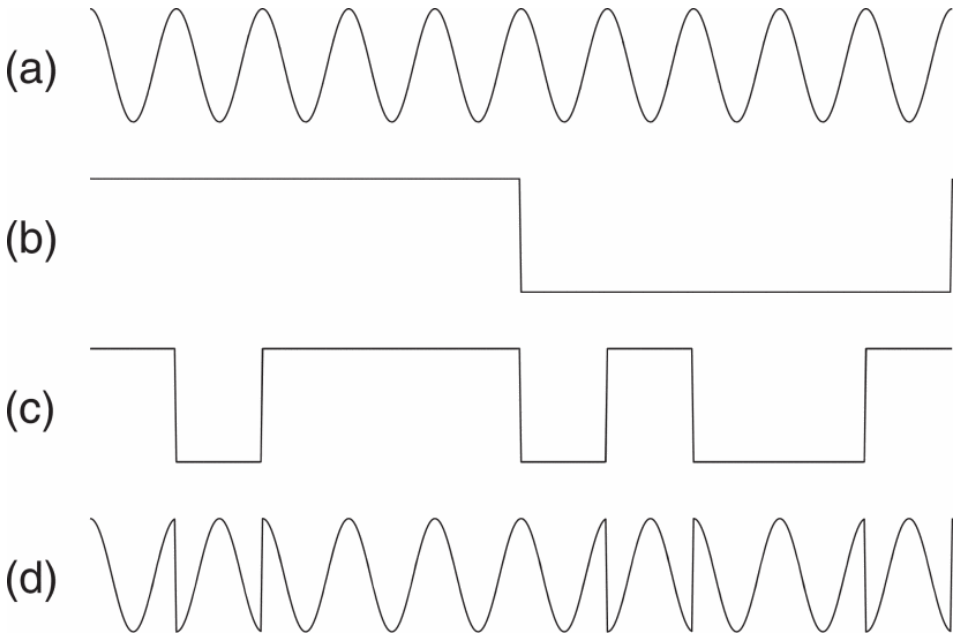

Ilustración 1-3. Componentes de la señal GPS: (a) Portadora sin modular; (b) Bits de datos del mensaje de navegación, D; (c) Chips de códigos PRN (C/A o P(Y)); (d) Portadora modulada por los datos de navegación y el Código PRN [3]

En la Ilustración 1-4, se observa la nueva arquitectura de procesado de IDEPAR al incluir la señal GPS como IO:

- Un bloque de pre-procesado (1) de la señal de referencia que resuelve los inconvenientes de identificación de satélites, baja potencia de señal y geometría variable.

  El proceso de reconstrucción de la señal se debe realizar en diferentes fases: identificación de los satélites disponibles (búsqueda PRN), estimación de parámetros variables como la frecuencia Doppler y el retardo de fase (Seguimiento de Señal), extracción del mensaje de navegación y la propia reconstrucción de la señal a partir de los parámetros. Para la reconstrucción de la señal de referencia se ha empleado algoritmos de adquisición y tracking.

- Bloque de procesado (2), similar al bloque de radar pasivo general mostrado en la Ilustración 1-2, donde la etapa de filtrado adaptativo es el equivalente al Disturbance Cancellation (Ilustración 1-2) que en este caso recibe la señal de referencia perfectamente reconstruida. La señal de vigilancia, una vez filtrada, se aplica la etapa que realiza la correlación cruzada (Cross-Ambiguity Function), que genera el mapa RDM, el cual constituye el espacio de observaciones del detector.

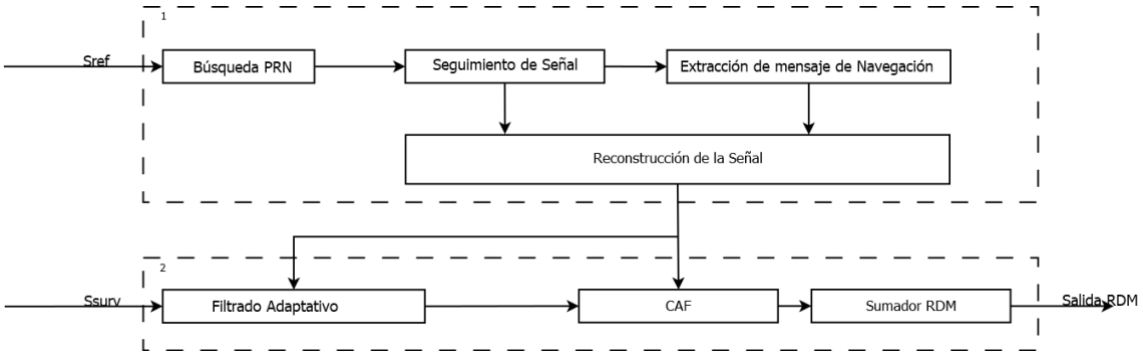

Ilustración 1-4. Nuevo esquema de la sección software del demostrador IDEPAR. (1) Bloque de pre-procesado; (2) Bloque de procesado

1.1 Objetivos

La problemática que motivó la propuesta del presente Trabajo Fin de Grado (TFG) fue el elevado tiempo de procesamiento de los algoritmos de adquisición y tracking del demostrador IDEPAR (implementados en el software MATLAB), que alejaba el sistema de cumplir los requisitos de Near Real Time, NRT.

Los objetivos de este TFG se centran, por tanto, en el estudio, desarrollo e integración de distintas técnicas para la aceleración del código implementado en MATLAB del demostrador IDEPAR, con el fin de reducir el gap que le separa de una solución compatible con los requisitos de tiempo real exigidos en aplicaciones de vigilancia. Para ello se han estudiado técnicas como la pre-asignación de memoria, la vectorización y la paralelización de código, sobre los algoritmos de adquisición y tracking implementados en MATLAB, operativos en IDEPAR en el momento de comenzar este TFG.

1.2 Estructura

El presente documento se inicia con la caracterización de los sistemas radar biestáticos pasivos, sobre los que está diseñado IDEPAR, para posteriormente analizar la estructura que debe tomar el bloque de procesado de señal, con la incorporación de una nueva etapa de pre-procesado que permite la utilización de las señales GPS con propósitos de detección.

Como primer paso, se presenta un análisis teórico de la señal GPS como fundamento de los algoritmos aplicados en la etapa de pre-procesado.

Vistas las bases fundamentales de la señal GPS y los algoritmos implementados para la etapa de pre-procesado, se detallan los algoritmos mejorados mediante la utilización de técnicas de optimización como la pre-asignación de memoria, la vectorización y la paralelización de código.

La estructura que sigue este documento será la siguiente:

- **Capítulo 1. Introducción.**

- **Capítulo 2. Radar pasivo.** En este capítulo se detallan los conceptos teóricos de radar, haciéndose hincapié en los sistemas pasivos. También se describen las diferentes tecnologías que se pueden emplear como IO, así como las ventajas e inconvenientes de los radares pasivos y sus aplicaciones.
- **Capítulo 3. GPS como IO.** En concreto se estudian las variaciones que sufre la etapa de procesado de señal que implementa el demostrador IDEPAR. Se analiza la nueva geometría del sistema cuando incorpora este tipo de señales, haciendo necesario la inclusión de una etapa de pre-procesado.
- **Capítulo 4. Sistema GPS.** En este capítulo se estudia la señal GPS para uso civil, como base de los algoritmos implementados en el pre-procesado. Incluye una introducción al sistema GPS y características de la señal como la frecuencia de transmisión, los códigos PRN y el mensaje de navegación; además, se describe la modulación, las propiedades de señal de espectro ensanchado y CDMA, y las características propias de la señal GPS.
- **Capítulo 5. Implementación de la etapa de pre-procesado.** En este capítulo se describen los algoritmos empleados en la etapa de pre-procesado: adquisición y tracking.
- **Capítulo 6. Optimización de la etapa de pre-procesado.** Estudio comparativo de métodos de optimización de código como la pre-asignación de memoria, la vectorización y la paralelización del código. Desarrollo e integración de mejoras en el código implementado en MATLAB y operativo en IDEPAR en el momento de comenzar este TFG.
- **Capítulo 7. Resultados.** En este capítulo se realizan dos estudios, uno sobre la implementación de códigos con señales filtradas y otro de rendimiento de los códigos empleados. En el primero se busca la influencia de señales filtradas sobre los bloques de adquisición y tracking. En el segundo, se comprueba el incremento de rendimiento de los códigos implementados a partir de la ley de Amdahl.
- **Capítulo 8. Conclusiones y líneas futuras**. En este capítulo se desarrollan las conclusiones de los estudios y desarrollos realizados. Por otra parte, se destaca las posibilidades de la optimización de la etapa de pre-procesado sobre otro lenguaje de programación como CUDA (Compute Unified Device Architecture), potenciándose la programación paralela.

2 Radar pasivo

2.1 Sistemas Radar

Def. Radar. "Sistema electromagnético para la detección y localización de objetos que operan mediante la transmisión de señales electromagnéticas, recibiendo los ecos de los objetos (blancos) dentro de su volumen de cobertura y extrayendo la localización y otra información proveniente del eco de la señal" [4].

A estos objetos se les denominará blancos y pueden ser aviones, barcos, naves espaciales, vehículos terrestres, personas, animales o elementos del medio como la meteorología, entre otros. La energía electromagnética que retorna se denomina eco, obteniéndose datos como la localización del blanco, así como su velocidad, forma o aceleración.

Def. Blanco (Target). "Objeto a detectar/seguir, pero también puede ser cualquier objeto que dispersa energía hacia el receptor radar al ser iluminado por su transmisor" [4].

Def. Clutter. "Ecos no deseados, típicamente suelo, mar, lluvia u otras precipitaciones, pájaros, insectos, meteoritos y auroras" [4].

Por tanto, en los sistemas radar se distingue los blancos como los ecos de los objetos que se desean localizar/seguir, y el clutter como los ecos de los objetos no deseados y que limitan las capacidades detectoras del radar.

En la Ilustración 2-1, se observa el entorno radar con diversidad de clutter; con lo cual, es importante caracterizar el nivel de clutter en los sistemas radar ya que no es lo mismo rastrear en el suelo, en el mar o en el cielo; además, hay que añadir la situación geográfica o las condiciones meteorológicas. Por lo tanto, es necesario definir correctamente el entorno para minimizar el impacto en la capacidad detectora del radar, ya que puede provocar detecciones erróneas o falsas alarmas.

Def. Falsa alarma. "Decisión errónea de detección del radar producida por ruido u otra fuente de interferencia que excede el umbral de detección" [4].

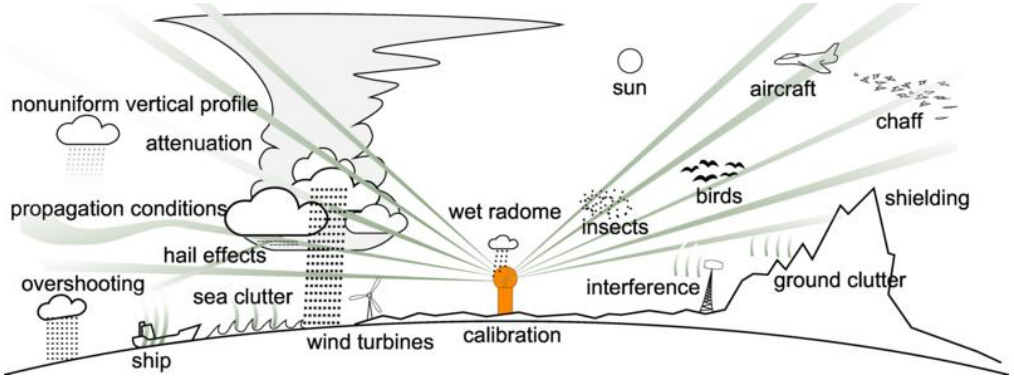

Ilustración 2-1. Entorno Radar con diversidad de clutter [5]

2.1.1 Clasificación de los Sistemas Radar

Los sistemas radar se puede clasificar según el tipo de blanco, su ubicación, su finalidad, su resolución, el tipo de señal y su fuente de señal.

- Según tipo de blanco, se puede distinguir dos tipos: los radares primarios, que actúan recibiendo los ecos procedentes de blanco que actúa de forma pasiva, mientras que los radares secundarios, envían una señal, normalmente codificada, al blanco que responde con otra señal revelando información sobre su posición o características.
- Según su ubicación, posición relativa del transmisor y del receptor, se puede distinguir entre radares monoestáticos o multiestáticos. Los radares monoestáticos utilizan la misma antena tanto para transmisión como para recepción, mientras que en los radares multiestáticos, los sistemas de transmisión y recepción están en emplazamientos diferentes y pueden hacer uso de antenas ubicadas en diferentes localizaciones para las labores de transmisión y/o de recepción.
- Según la finalidad. Se diferencian entre radares de vigilancia o de seguimiento. Los primeros, también llamados radares de exploración, simplemente rastrean un área determinada en busca de blancos disponibles, mientras que los segundos, fijan su objetivo en un blanco en concreto para realizar su seguimiento dentro del área de cobertura. Estos radares de seguimiento son usados mayormente en operaciones militares.
- Según la resolución: radares convencionales y de alta resolución. En el caso de los radares convencionales, la celda de resolución es mayor que las dimensiones del blanco, mientras que, en los radares de alta resolución, al usar señales de gran ancho de banda, su celda de resolución es menor que el blanco, por lo que reciben múltiples ecos de un mismo blanco (firma radar).
- Según el tipo de señal: radares pulsados y los radares de onda continua. Los radares pulsados emiten pulsos de radiofrecuencia con duración determinada y "escuchan" entre pulsos; los radares de onda continua emiten una onda electromagnética de forma ininterrumpida, basándose su detección en los desplazamientos Doppler que sufre la señal recibida.
- Según la fuente de señal: radares activos, incorporan un transmisor dedicado; radares pasivos, utilizan transmisores de otros servicios de radiocomunicación o radiodeterminación como iluminadores de oportunidad.

2.2 Definición de Radar Pasivo

Def. Radar biestático pasivo. "Es un conjunto de técnicas que usan señales de sistemas de comunicaciones, radar o radionavegación como fuentes de iluminación en lugar de utilizar un transmisor radar dedicado. Otros términos que se usan incluyen Passive Coherent Location (PCL), Passive and Convert Radar (PCR), Convert Radar, Non-cooperative Radar, Broadcast Radar, Parasitic Radar y Opportunistic Radar". [4]

Entonces, según la definición un ejemplo de radar biestático es el que usa antenas en distintas localizaciones, una para transmisión y otra para recepción. Los sistemas multiestáticos pueden tener uno o más transmisores y/o uno o más receptores en emplazamientos diferentes.

Por otra parte, estos radares biestáticos/multiestáticos pueden funcionar con sus propios transmisores dedicados, que son estrictamente diseñados para esta labor, o hacer uso de transmisores de oportunidad, diseñados para otros propósitos. A estos tipos de radares que hacen uso de transmisores no dedicados se les denomina radares pasivos, PR (Passive Radar). [6]

2.3 Radares Pasivos Biestáticos

En la Ilustración 2-2, se observan los canales de Referencia (Reference Channel) y de Vigilancia (Surveillance Channel), ya que al no tener control sobre la señal transmitida por el IO es necesario el uso de dos canales en el receptor del PR.

Al emplear un IO se cumple la característica de "pasivo", y al tener transmisor (IO) y el receptor en localizaciones diferentes cumple la condición de "biestático". Por lo que, en su conjunto, esta es una configuración de Radar Pasivo Biestático.

La ecuación del alcance de un radar biestático permite determinar la distancia máxima a la que el radar es capaz de detectar un blanco [7] (2.1):

$$\langle R_T R_R \rangle_{max} = \left[\frac{P_T G_T G_R \lambda^2 \sigma_B F_T^2 F_R^2}{(4\pi)^3 K T_s B_n (S/N)_{min} L_T L_R} \right]^{1/2} \tag{2.1}$$

Donde,

- P_T es la potencia transmitida por el IO.
- G_T y G_R son las ganancias de las antenas transmisoras y receptoras.
- R_T y R_R son las distancias transmisor-blanco y blanco-receptor.
- σ_B es la sección recta radar biestática.

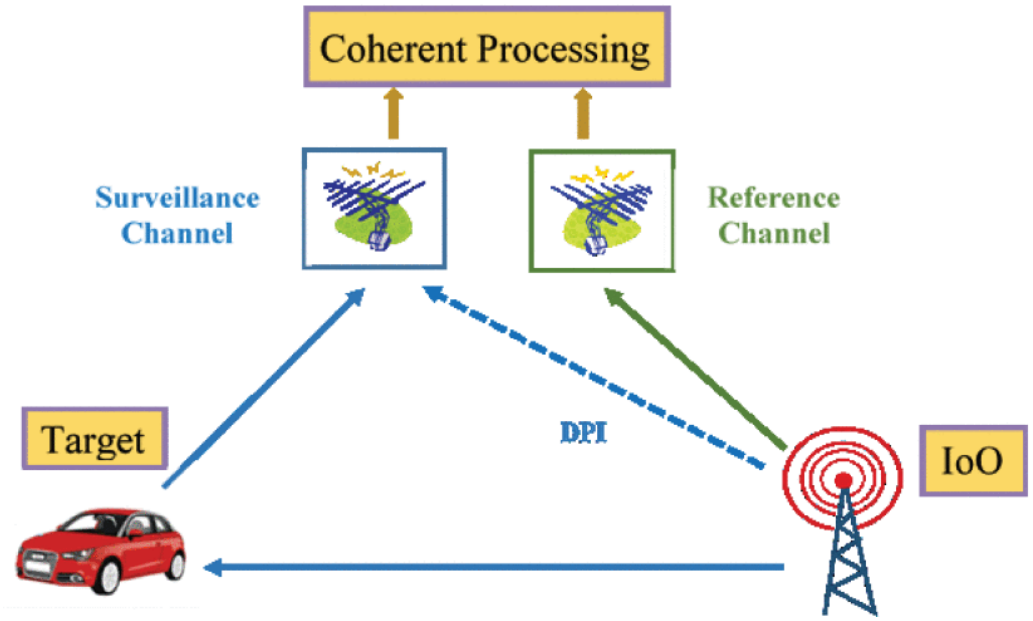

Ilustración 2-2. Geometría general de Radar Pasivo Biestático [1]

- λ es la longitud de onda.
- F_T y F_R son los factores de propagación transmisor-blanco y blanco-receptor (multitrayecto, difracción, refracción, gases e hidrometeoros).
- K es la constante Boltzmann.
- T_S es la temperatura de ruido del sistema receptor.
- B_n es el ancho de banda de ruido.
- L_T y L_R son las pérdidas de los sistemas transmisor y receptor.
- S/N_{min} es la relación de potencia señal-ruido mínima requerida para cumplir los requisitos de detección.

La definición de resolución biestática, es el grado en el cual dos o más blancos se pueden separar en una o más dimensiones, tanto en rango, como en velocidad (Doppler), así como en ángulo [7]. Estas resoluciones dependerán de las características de la señal del IO, el tiempo de integración y el diagrama de radiación de la antena receptora.

La **resolución en rango** se formula en (2.2). Depende del ancho de banda de la señal, B, y de la geometría que definen el IO (*Tx*), receptor radar (*Rx*) y los blancos (target#1, target#2 y target#3).

$$\Delta R \approx \frac{c}{2B\cos(\beta/2)\cos\psi} \tag{2.2}$$

En la Ilustración 2-3, β es el ángulo biestático, formado por IO-blanco-receptor, mientras que ψ o ángulo de aspecto, se corresponde con el ángulo de aspecto que se forma respecto al eje bisector biestático ($\beta/2$).del escenario a través de los ángulos β y ψ, que se definen en y de la velocidad a la que se transmite c, que corresponde con la velocidad de la luz.

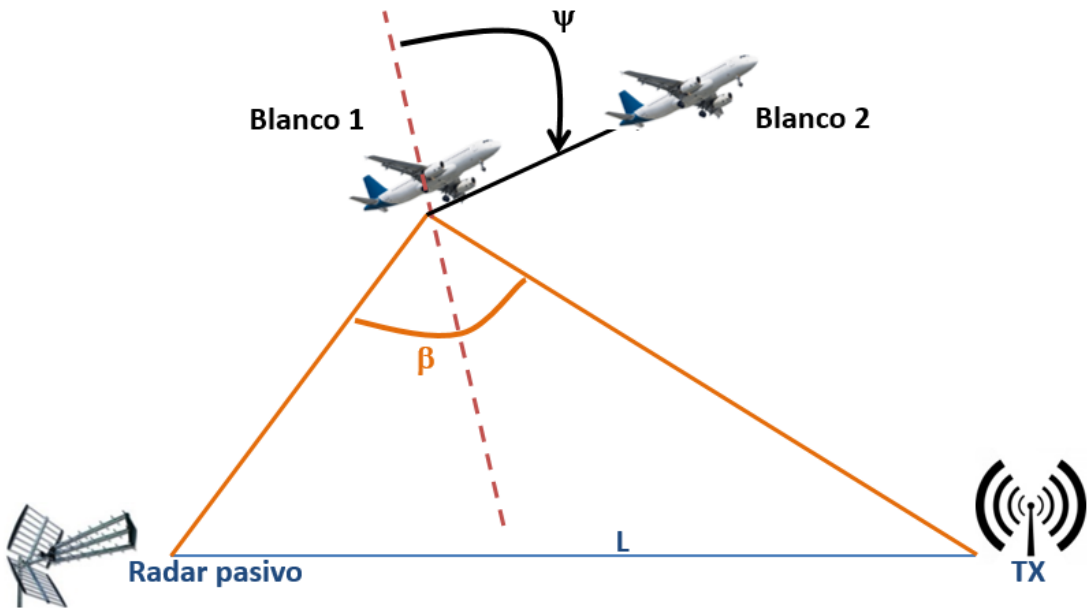

Ilustración 2-3. Geometría para la resolución de rango biestático

En cuanto a la **resolución Doppler**, es el grado de separación Doppler mínimo entre dos ecos de los blancos en el receptor. Viene definido por el inverso del tiempo de integración coherente T_D (2.3). El desplazamiento Doppler del eco recibido por un blanco es el que se indica en (2.4), que depende de la geometría del sistema (Ilustración 2-4).

$$\left|f_{Tgt1} - f_{Tgt2}\right| = \frac{1}{T_D} \tag{2.3}$$

$$f_{Tgt} = \frac{2V}{\lambda}\cos\delta \quad \cos\beta/2 \tag{2.4}$$

Combinando las ecuaciones (2.3) y (2.4) se obtiene la diferencia entre dos vectores de velocidad, de dos blancos que comparten el mismo bisector biestático, que asegura la resolución Doppler (2.5)

$$\Delta V = (V_1\cos\delta_1 - V_2\cos\delta_2) = \frac{\lambda}{2T_D\cos(\beta/2)} \tag{2.5}$$

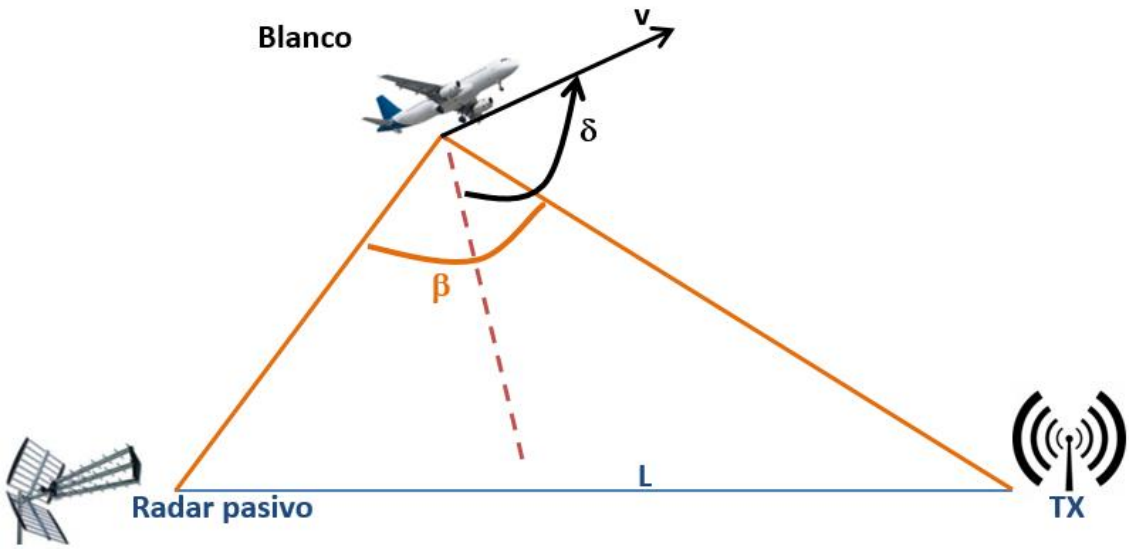

Ilustración 2-4. Geometría para la resolución Doppler en configuración biestática

La **resolución angular** que se define como la diferencia angular mínima que tiene que haber entre dos blancos que se encuentran en el mismo rango biestático. Si $\Delta\theta_T R_T > 2\Delta\theta_R R_R$, siendo $\Delta\theta_T$ y $\Delta\theta_R$ los anchos de los haces a -3 dB de la antena transmisora y receptora respectivamente, la resolución en ángulo se calcula como (2.6):

$$(\Delta R_\theta)_u \approx \frac{2\Delta\theta_R R_R}{\cos(\beta/2)} \tag{2.6}$$

2.4 Iluminadores de oportunidad

Los transmisores de los siguientes sistemas de comunicación y radiodeterminación se han utilizado como IOs para radares pasivos:

- Radio analógica [8]: que usa una banda de frecuencias FM entre los 88 y 108 MHz.
- Radio digital [9]: como *Digital Audio Broadcasting* (DAB) y *Digital Radio Mondiale* (DRM).
- Televisión digital: tanto terrestre, *Digital Video Broadcast – Terrestrial* (DVB-T) [10], como satélite, *Digital Video Broadcast – Satellite* (DVB-S) [11] o, su sucesor, *Digital Video Broadcast – Satellite, second generation* (DVB-S2) [12]. Las bandas de frecuencias varían según la región y la legislación actual.
- Telefonía móvil: se ha desarrollado implementaciones sobre distintos estándares como *Global System for Mobile communications* (GSM, también conocido como 2G) [13], *Universal Mobile Telecomunications Systems* (UMTS, también conocido como 3G) [14] y *Long Term Evolution* (LTE, también conocido como 4G) [15].
- Sistemas de localización: como *Global Positioning System* (GPS).
- Otras redes inalámbricas: como el estándar 802.11 también conocido como WiFi, *Wireless Fidelity* [16].

2.5 Ventajas e inconvenientes de un radar pasivo

Estos sistemas PR tienen unas ventajas potenciales:

- El receptor es pasivo y, por lo tanto, potencialmente indetectable excepto el impacto visual que genera el sistema de antenas receptoras.
- Los receptores PR a menudo suelen ser simples y de bajo coste, y no necesitan de asignación de frecuencias al no disponer de elemento transmisor.
- Hay muchos IO que pueden ser usados, muchos de ellos con mucha potencia y correctamente situados. Hoy en día hay mucho interés en el uso de iluminadores satelitales, por las mejoras esperables en cobertura y su invulnerabilidad frente a catástrofes naturales o provocadas por el hombre.
- No generan problemas de compatibilidad electromagnética.

Pero también se encuentran desventajas a considerar:

- Como las ondas emitidas por el IO no están diseñadas para fines radar, es necesario entender el efecto de las ondas sobre el rendimiento del PR, para tener la capacidad de procesarlas y sacar el máximo rendimiento.
- Incremento de la complejidad del sistema. Es necesario incrementar la potencia de cálculo para procesar las señales y aproximarse a un sistema de procesado en tiempo real.
- Es necesaria la sincronización de las señales captadas por los sistemas receptores.
- Dependencia de los IO, es decir, si se encuentran operativos o no.
- Además, nos encontramos en un entorno hostil de muchas señales e interferencias, especialmente en ciudades donde la densidad de transmisores y el nivel de señales es elevado. Con lo cual, se requiere de técnicas de procesamiento precisas que ayuden a suprimir cualquier interferencia y así poder detectar el blanco deseado.
- En el caso de los iluminadores satelitales, en concreto el sistema GPS, ofrece desventajas en términos de potencia recibida y geometría del sistema.

2.6 Aplicaciones de los radares pasivos

La multitud de aplicaciones de un radar pasivo hace que sea una tecnología en continuo crecimiento. Las aplicaciones que se destacan son las siguientes:

- Defensa militar. La vigilancia y control de tráfico aéreo. También son sistemas usados en el guiado y/o en la detección de balística militar [17].
- Control de tráfico aéreo civil. Los sistemas tradicionales de control de tráfico aéreo pueden tener ciertas limitaciones en la detección de blancos a baja altura o con escasa sección transversal, por lo que en [18], se hace una propuesta de mejora de las capacidades de un radar primario con la incorporación de un radar pasivo.
- Control de tráfico marítimo. Los radares marinos son fundamentales en condiciones de baja visibilidad. Los radares pasivos son soluciones alternativas a los problemas debidos al clutter marino, que hace difícil la detección de pequeñas embarcaciones [19].

- Control de tráfico urbano. En este caso se generan distintas aplicaciones que van desde el control de congestión de vehículos, monitoreo de tráfico en entorno urbano, así como el control de velocidad [20].
- Meteorología. Al tratarse de receptores de radar pasivo livianos también es posible incluirlos en pequeños UAV para uso meteorológico [21]. El empleo de este tipo de vehículos incrementa el número de posibles aplicaciones como puede ser el de vigilancia de infraestructuras, cartografía, hidrología, transporte, seguridad y control fronterizo.
- Vigilancia en infraestructuras. Se extiende su uso desde plantas petrolíferas a zonas muy pobladas donde la proximidad a IOs lo hacen un sistema de bajo coste y fiable. En [22], se realiza una propuesta de vigilancia de secciones ferroviarias para evitar el uso malintencionado de drones.

3 GPS como IO

A lo largo de los años se ha investigado en distintos IOs basados en sistemas radio analógica (*FM*) o digital (*DAB* o *DRM*), en sistemas de televisión digital tanto terrestre (*DVB-T*) como satélite (*DVB-S*), en telefonía móvil (*GSM*, *UMTS* o *LTE*), en sistemas inalámbricos como Wifi, así como en sistemas de posicionamiento (*GPS*), siendo este último uno de los sistemas más prometedores.

Este TFG se centra en el empleo de la señal GPS como IO de un radar pasivo para la detección de blancos terrestres. La adquisición de la señal GPS y su posterior procesado se realiza con demostrador IDEPAR. Por lo que primero es necesario conocer ¿qué es?, y ¿en qué se basa?, para posteriormente conocer las ventajas e inconvenientes del empleo de la señal GPS como IO y comprobar en qué afecta al demostrador IDEPAR.

3.1 Demostrador IDEPAR

Las siglas *IDEPAR* vienen de *Improved DEtection techniques for PAssive Radars,* y es un demostrador de radar pasivo desarrollado por el grupo de investigación de la Universidad de Alcalá dirigido por la Dra. Mª de Pilar Jarabo Amores. Este demostrador ha sido financiado por diferentes proyectos de investigación nacionales.

El demostrador nace con la necesidad de impulsar mejoras en las capacidades de detección de los radares pasivos, actualizando su hardware y software dependiendo del IO seleccionado.

3.1.1 Principio de funcionamiento

En la Ilustración 3-1, se observa la estructura del demostrador IDEPAR, donde se distinguen dos etapas [23]:

- **Etapa de Adquisición**. Se corresponde con la sección hardware del demostrador, encargada de adquirir y preparar la señal para un posterior procesado, con las antenas receptoras, el front-end de RF y las tarjetas de adquisición.
- **Etapa de Procesado**. Se corresponde con la sección software donde se realiza el procesado coherente de las señales digitalizadas cuyo objetivo es detectar la presencia de los blancos y extraer propiedades como la posición y la velocidad.

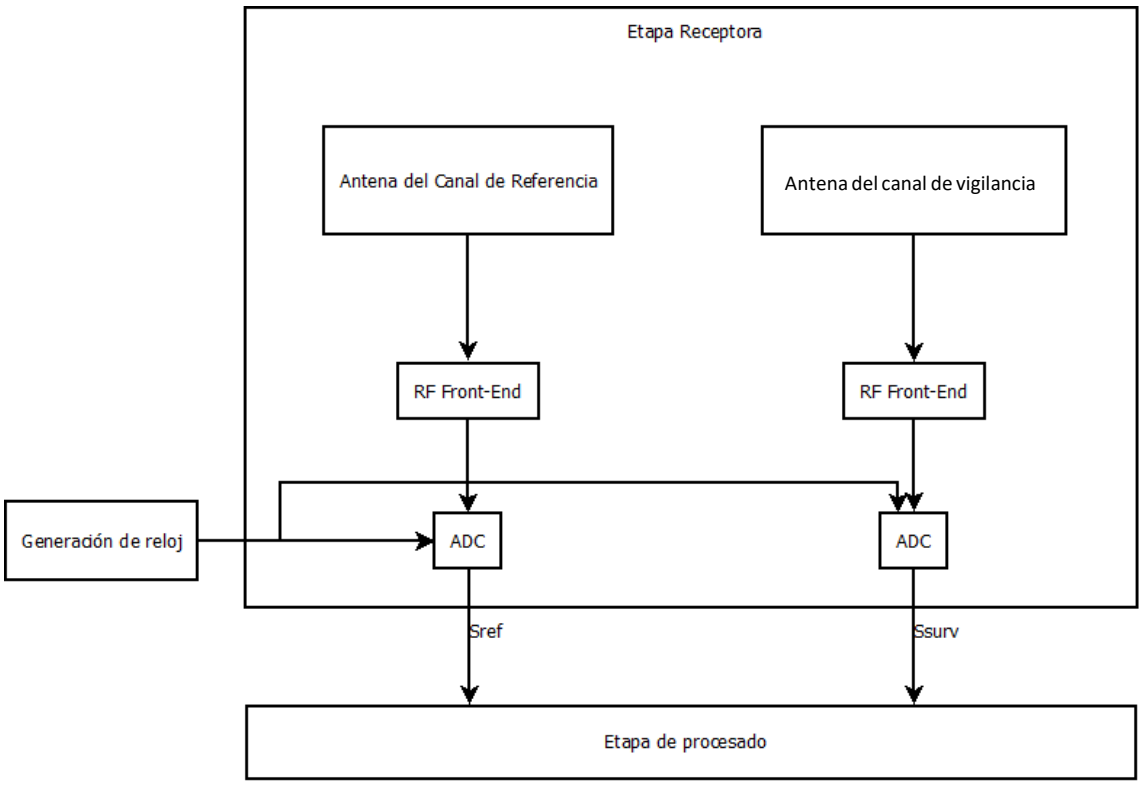

Ilustración 3-1. Estructura del demostrador IDEPAR

3.1.1.1 Etapa de adquisición

En la Ilustración 3-1, se puede observar la estructura del receptor de demostrador IDEPAR, con el canal de Referencia y el canal de Vigilancia. Por el canal de Referencia se debe recibir la señal que proviene del IO y por el canal de Vigilancia se captan los ecos generados por los blancos al ser iluminados por el IO. Las señales son captadas a través de las antenas receptoras, posteriormente se les aplica un tratamiento de radiofrecuencia (RF Front-End) para desplazar las señales adquiridas a banda base y, por último, un conversor *ADC* que transforma las señales analógicas en señales digitales, listas para la etapa de procesado. Por otra parte, un sistema de sincronización asegura la sincronización en frecuencia y fase de los ADCs., imprescindible para el procesado coherente posterior.

Los componentes de la etapa de adquisición son:

- **Antena** [24]. Para la recepción de señales GPS de los IOs y de los ecos de los blancos IDEPAR está provisto de dos antenas receptoras activas *GPS-L1L2-28MAG*. Algunas de las características de las antenas son:
  - Bandas de frecuencia: *L1* (1575,42 +/- 10 MHz) y *L2* (1227,60 +/- 10 MHz)
  - Impedancia nominal: 50 Ω
  - Polarización: Circular a derechas
  - Amplificador: *LNA* (*Low Noise Amplifier*) de 33 +/- 3 dB
  - Ganancia: 10º > 3dBi

El amplificador de bajo ruido se suele incorporar en el canal de Vigilancia para aumentar el nivel de los ecos.

- **RF Front-End y ADC** [25]. Las señales captadas por las antenas son inyectadas a la cadena de RF Front-End y ADC, tal como se observa en la Ilustración 3-1. Este proceso se realiza gracias a las 2 tarjetas de adquisición USRP X310 que se equipan con 2 daugtherboards TwinRX y 2 FPGAs de alto rendimiento, por cada tarjeta de adquisición. Las daugtherboards tienen la capacidad de analizar 2 canales analógicos, con lo que en total se obtiene un sistema capaz de captar hasta 8 canales analógicos con un ancho de 80 MHz por canal con el fin de incrementar las capacidades del demostrador IDEPAR. La etapa de RF es llevada a cabo por las daughterboards TwinRX. La señal llega a la tarjeta USRP X310 con el fin de realizar un muestreo digital de la señal. Por último, mediante FPGA se realiza un Down-converter para adaptar las señales a 25 MS/s.

3.1.1.2 Etapa de procesado

En la Ilustración 3-2, se observa un esquema general de procesado de señal de un radar pasivo.

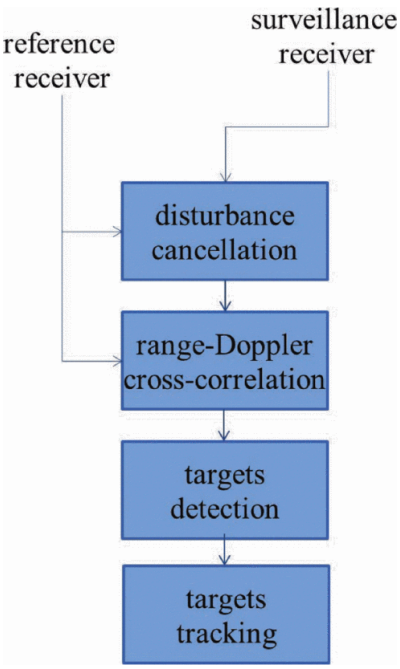

Ilustración 3-2. Esquema básico de la etapa de procesado de un radar pasivo [2]

El punto clave en la etapa de procesado para la detección de blancos se encuentra en el cálculo de la CAF (Cross-Ambiguity Function), que es la función de correlación cruzada (3.1) entre las copias retrasadas en tiempo y desplazadas en frecuencia de la señal de referencia y los ecos adquiridos por el canal de vigilancia, con la que se consigue una estimación de rango biestático y el desplazamiento Doppler de los blancos presentes.

$$S_{CAF}[m,p] = \sum_{n=0}^{N-1} s^{*}_{ref}[n-m] \cdot s_{surv}[n] \cdot exp^{-j2\pi\frac{p}{N}n}$$

(3.1)

- s_{ref} es la señal de Referencia
- s_{surv} es la señal de Vigilancia
- N es el número de muestras. $N=CPI \cdot f_s$
- CPI es el tiempo de procesamiento [s]
- f_s es la frecuencia de muestreo [muestras/s o Hz]
- m es el bin de tiempo asociado al retardo temporal. $\tau_m = m/f_s$
- *p es el* bin desplazamiento Doppler correspondiente a $f_D = f_s(p/N)$

En la Ilustración 3-3, se representa el proceso de generación de la CAF.

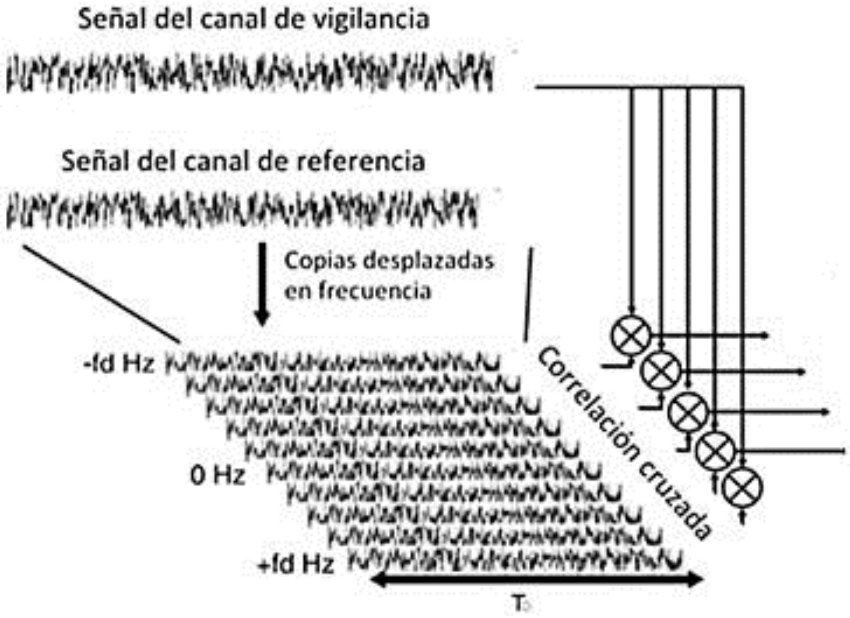

Ilustración 3-3. Proceso de generación de la CAF [26]

Gracias al procesado coherente de las señales adquiridas se genera el mapa Rango-Doppler RDM (Range-Doppler Map), espacio de trabajo del detector. Los máximos generados en el RDM resultante, representan los blancos en el área de interés, siempre que se tenga en cuenta las siguientes condiciones:

- En la etapa de adquisición hay que rechazar las señales interferentes en el mayor grado posible.
- En el canal de referencia hay que evitar los multitrayectos, ya que esto puede generar copias CAF que pueden dar lugar a blancos falsos (blancos fantasma) [27].

- En el canal de vigilancia hay que tener en cuenta principalmente la influencia del DPI, a parte del clutter y otras fuentes de interferencia, ya que este genera máximos CAF en zonas de zero-Doppler y zero-Delay. Esto puede reducir mucho las capacidades de detección del radar ya que se trata de una señal entre 80-100 dB superior a la de un eco de un blanco.

Por ello es necesario incorporar una etapa de filtrado (Disturbance Cancellation, en la Ilustración 3-2) sobre la señal de vigilancia, previa al cálculo de la CAF, que ayuden a eliminar en gran medida las contribuciones de interferencia, clutter y DPI.

En los siguientes apartados se comprueba las ventajas e inconvenientes de la señal GPS como IO (apartado 3.2), así como su geometría (apartado 3.3), punto importante en las variaciones necesarias en la etapa de procesado cuando se adapta este tipo de señales (apartado 3.4).

3.2 Fundamentos de la señal GPS como IO

Hay muchas ventajas derivadas del uso del sistema GPS como IO: las altas frecuencias de trabajo, la compartición de la Banda L por todos los satélites, el uso de señales de espectro ensanchado que presentan mejor inmunidad frente a interferencias rápidas o la más importante, su cobertura global las 24 horas del día. El sistema GPS cuenta con una constelación de hasta 32 satélites que viajan sobre orbitas medias MEO, a una altitud aproximada 20200 km sobre la superficie terrestre [28]. Para cubrir todos los puntos de la Tierra se distribuyen en 6 planos orbitales igualmente espaciados. Esta disposición garantiza que haya presentes al menos 4 satélites visibles en cualquier punto del planeta.

Pero también se observan desventajas muy importantes: su ancho de banda de 2 MHz que permite una resolución en rango relativamente baja de 75 metros en equivalente monoestáticos (en comparación a los 6 metros que permite el sistema DVB-T), la geometría del sistema, ya que, al no estar en una *órbita geoestacionaria* el satélite no aparece en un punto fijo en el firmamento, y la baja potencia de señal recibida.

Para enfrentarse a estos inconvenientes se usan distintas técnicas que combina la reconstrucción de la señal de referencia, el uso de filtros adaptativos y grandes tiempos de procesamiento.

3.3 Geometría del radar pasivo con GPS como IO

En la Ilustración 3-4, se describe la geometría espacial del sistema y se observa la etapa emisora conformada por el IO (satélite GPS, en la ilustración IoO), la etapa receptora del radar pasivo conformada por las antenas receptoras del canal de referencia (Reference Channel) y del canal de vigilancia (Surveillance Channel), el blanco terreno (automóvil rojo), así como de los trayectos de la señal del IO y sus ecos. Las distancias de los trayectos vienen representadas por L (IO – PR), R_T (IO – blanco terreno) y R_R (blanco terreno – PR). El ángulo biestático viene dado por β. Y, por último, la etapa de procesado que viene representada por la CAF.

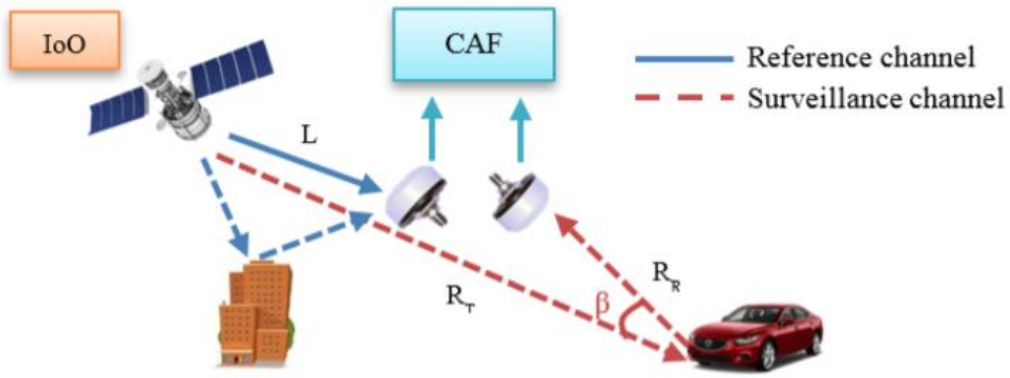

Ilustración 3-4. Geometría del Sistema de PR con GPS como IO

3.4 Necesidad de pre-procesado y procesado de la señal GPS

En el apartado 3.1.1.2, se analiza la estructura general de la etapa de procesado de un PR, donde el principal inconveniente es conseguir una señal de vigilancia libre de interferencias, clutter o DPI y una señal de referencia libre de ecos de los blancos y de fuentes de interferencias.

Al incluir la señal GPS como IO del radar pasivo aparecen inconvenientes que son necesarios remediar como la geometría variable del sistema, la escasa potencia de señal recibida y la transmisión de todos los satélites a la misma frecuencia (al haber mezcla de señales no se puede realizar la correlación directa porque se tiene la influencia de todos los satélites disponibles).

En la Ilustración 3-5, se observa el nuevo esquema de procesado que busca resolver los inconvenientes de GPS como IO. Para resolver estos inconvenientes es necesario la identificación del satélite, la estimación de parámetros variables (frecuencia Doppler y retardo de fase), extracción del mensaje de navegación y la propia reconstrucción de la señal de Referencia. Este proceso de reconstrucción de la señal de referencia se realiza en el bloque 1. Por otra parte, este esquema también incorpora una etapa de procesado coherente, donde se realiza el filtrado adaptativo de la señal de vigilancia para eliminar las componentes de DPI y, posteriormente, el cálculo de la CAF que permite obtener el mapa RDM. Esta etapa de procesado de la señal se realiza en el bloque 2.

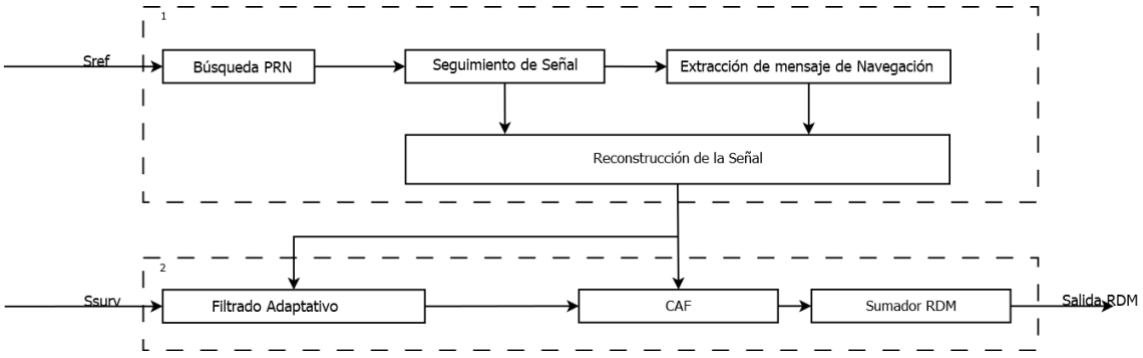

Ilustración 3-5. Esquema de la nueva sección de procesado del PR con GPS como IO. (1) Reconstrucción de la señal; (2) Procesado coherente

3.4.1 Etapa de pre-procesado

Como la señal de referencia captada por el receptor de radar pasivo es muy débil, es necesario reconstruir la señal para realizar el procesado coherente correctamente. Para ello el primer bloque de la etapa de pre-procesado es el de identificación del satélite, ya que estos hacen uso compartido del espectro de frecuencias aplicando técnicas CDMA (Code Division Multiple Access). El proceso de identificación del satélite se hace gracias a los códigos PRN que identifican la señal. La componente civil de la señal transmitida se describe en (3.2):

$$SAT_i(t) = A_{p_i} \cdot N_{M_i}(t) \cdot CA_i(t) \cdot \cos(2\pi f_c t) \tag{3.2}$$

Donde:

- A_P es el factor de amplitud de la señal
- N_M es el mensaje de navegación
- CA es el código PRN
- f_c es la frecuencia de portadora
- i es el subíndice del satélite

La distancia entre el satélite y el receptor de radar pasivo, y el movimiento del satélite, generan retardos temporales y desplazamientos Doppler en la señal de referencia recibida que, después de la demodulación en cuadratura se puede modelar de la siguiente manera (3.3) [27]:

$$s_{ref_i}(t) = N_{M_i}\big(t - \tau_i(t)\big) \cdot CA_i\big(t - \tau_i(t)\big) \cdot exp\big(2\pi f_{D_i}(t)t\big) \tag{3.3}$$

Donde:

- τ es el retardo temporal
- f_D es la frecuencia Doppler
- i es el subíndice del satélite

La identificación del satélite y la estimación en primera estancia del retardo temporal y la frecuencia Doppler se hacen en el bloque de Adquisición (apartado 5.1.1).

El movimiento del satélite hace que el retardo temporal y la frecuencia Doppler sean dependientes del tiempo por lo que es necesario introducir un proceso de Tracking (apartado 5.1.3) en intervalos de 1 ms para estimarlos. Una vez obtenidos es posible extraer el mensaje de navegación y posteriormente reconstruir la señal de referencia. En el capítulo 5, se ampliará la información acerca de los bloques de adquisición y Tracking.

3.4.2 Etapa de procesado coherente

La señal recibida por el canal de Vigilancia se puede modelar de la siguiente manera (3.4):

$$s_{surv}[n] = s_{target}[n] + A_{tt_{DPI}} s_{DPI}[n] + s_{clutter}[n] + s_{noise}[n] \tag{3.4}$$

Donde:

- s_{target} es la componente de los ecos producidos por los blancos
- A_{tt} es un factor de atenuación
- s_{DPI} es la componente de DPI adherida a la señal de Vigilancia
- $s_{clutter}$ es la componente de clutter
- s_{noise} es la componente de ruido

Luego la señal de Vigilancia debe pasar por un bloque de filtrado adaptativo para eliminar el DPI y las posibles contribuciones de clutter. Este filtrado se realiza mediante el algoritmo FAST ECA [29] y emplea la señal de referencia reconstruida previamente. Una vez filtrada la señal de vigilancia pasa al bloque de procesado coherente con la CAF.

En cuanto al tiempo de procesamiento CPI, se debe lograr un equilibrio entre la ganancia de procesamiento, el desplazamiento Doppler del blanco y la resolución en rango y Doppler.

La resolución en rango está relacionada con el ancho de banda de la señal GPS (2 MHz) y el tiempo de integración coherente viene relacionado con el desplazamiento del blanco terrestre, que se espera una velocidad de hasta 34 m/s, lo cual permite grandes tiempos de integración. También hay que tener en cuenta la migración en Doppler lo que produce que utilicemos 4 RDMs con CPI de 250ms para incrementar el intervalo de integración hasta 1 s controlando la resolución Doppler (4Hz).

4 Sistema GPS

Los sistemas GNSS (Global Navigation Satellite System), en concreto GPS, son servicios que proveen a los usuarios de posicionamiento, navegación y sincronización [30].

GPS es un sistema DUAL, es decir, proporciona diferentes servicios como el servicio de posicionamiento standard (SPS, Standard Positioning Service) que está destinado para la comunidad civil, y el servicio de posicionamiento de precisión (PPS, Precision Positioning Service) que está destinado para usuarios militares autorizados y determinadas agencias de gobierno de los Estados Unidos (US, United States). El acceso GPS PPS se controla mediante criptografía. En cambio, el GPS SPS está disponible para todos los usuarios terrestres, sin restricciones de uso.

4.1 Introducción al sistema GPS

El sistema GPS consta de tres segmentos: el segmento espacial, el segmento de control y el segmento de usuario.

- **Segmento espacial.** Consiste en una constelación de satélites, actualmente 32, transmitiendo señales radio a los usuarios. Los satélites están colocados sobre 6 planos orbitales, tal como se observa en la Ilustración 4-1, y en cada uno de ellos 4 satélites uniformemente espaciados, es decir, un total de 24 satélites disponibles que permiten una cobertura global durante las 24 horas del día. Los satélites extra permiten incrementar las capacidades del sistema.

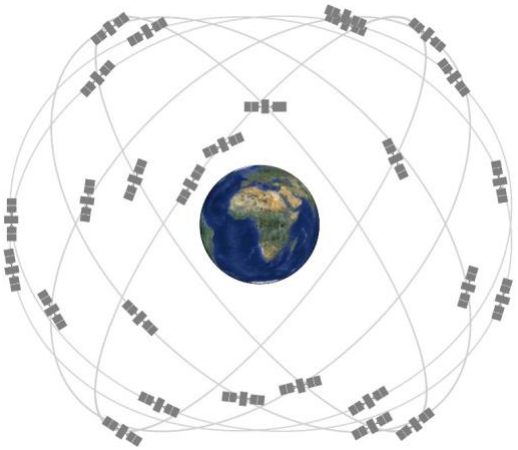

Ilustración 4-1. Constelación de satélites GPS [30]

- **Segmento de control.** Consiste en una red global terrestre que rastrean los satélites, monitorean sus transmisiones, realizan análisis y envían comandos y

datos a la constelación. En la Ilustración 4-2, se observa la distribución de las estaciones del segmento de control. En la actualidad cuenta con 16 estaciones de monitoreo (Air Force Monitor Station y NGA Monitor Station), una estación de control maestra (MCS, Master Control Station), una estación maestra de reserva (AMCS, Alternate Master Control System) y 11 antenas de comando y control (Ground Antenna y AFSCN Remote tracking Station).

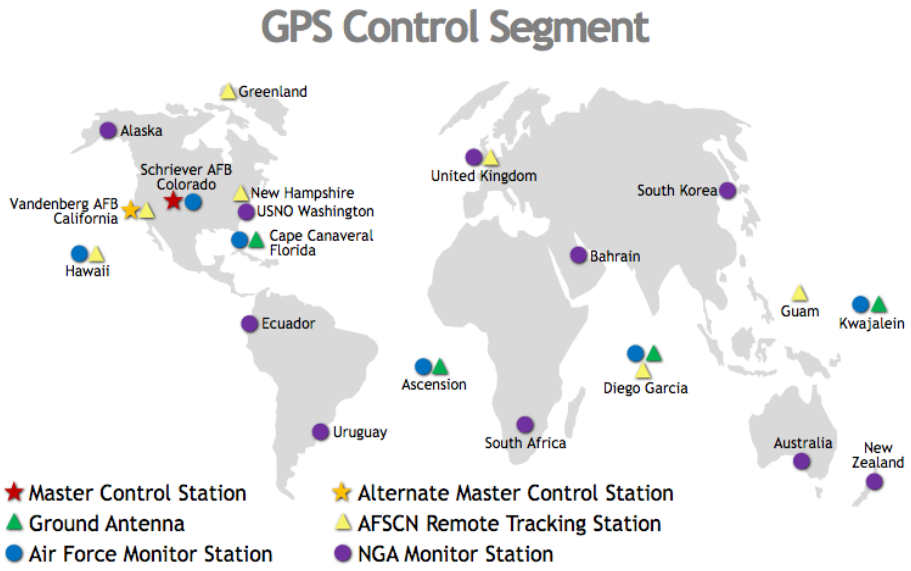

Ilustración 4-2. Segmento de control GPS [30]

- **Segmento de usuario**. Consiste en todo el hardware y software capaz de recibir y procesar la señal de los satélites; y usar la información transmitida para diferentes fines como la agricultura, la aviación, el medio ambiente, la marina, la seguridad, los primeros auxilios, las carreteras y autopistas, los trenes, el espacio, la topografía o en la sincronización de relojes. Este documento se sitúa en este segmento y usa la señal proveniente de los satélites GPS como elemento transmisor de un radar pasivo para la detección de blancos terrestres.

4.2 Características de la señal GPS

Las características de la señal GPS vienen dadas por la frecuencia de transmisión, los códigos PRN, el mensaje de navegación y su modulación.

4.2.1 Frecuencia de transmisión

La señal GPS se transmite en la banda L de frecuencias que va desde 1 a 2 GHz. En concreto se transmite en las bandas L1, L2 y L5 que derivan de la frecuencia estándar fundamental AFS (Atomic Frequency Standard), f_o=10,23 MHz.

En la Tabla 4-1, se resumen algunas características de las bandas de frecuencias de transmisión, como su factor de multiplicación con AFS, su frecuencia o su longitud de onda:

Tabla 4-1. Características de las frecuencias de transmisión de la señal GPS

Bandas	**Factor de multiplicación ($\cdot f_0$)**	**Frecuencia [MHz]**	**Long. de onda [cm]**
L1	154	1575,42	19,0*
L2	120	1227,60	24,4*
L5	115	1176,45	25,5*

*Su gran longitud de onda permite penetrar a través de estructuras terrestres.

Los códigos que se transmiten sobre estas bandas de frecuencias son:

- L1 se usa para transmitir los códigos C/A, P(Y), M (código Militar) y L1C (código civil sobre L1).
- L2 se usa para transmitir los códigos P(Y), M y L2C (código civil sobre L2).
- L5 se usa para transmitir la señal SoL (Safety of Life). Los servicios militares la usan para la transmisión de información, equipos de medición de distancia DME (Distance Measuring Equipment) y navegación aérea táctica TACAN (Tactical Air Navigation).

En este documento se hace hincapié sobre la banda L1 de frecuencia y el código C/A, necesario para la identificación del satélite y condición necesaria para la reconstrucción de la señal de referencia, apartado 3.4.1.

4.2.2 Códigos PRN

Los códigos PRN son códigos de espectro ensanchado que sirven para la identificación de los satélites a partir de técnicas CDMA.

Sobre la banda de frecuencia L1, tal como se ha visto en el apartado 4.2.1, se usa para transmitir los códigos C/A, P(Y), M y L1C. Siendo de uso civil los códigos: C/A y L1C; y de uso militar: P(Y) y M.

En el apartado 4.3, se incluye la ecuación (4.4) de la modulación de la frecuencia portadora L1.

4.2.2.1 Código C/A

El código C/A se define en el SPS. “Este servicio provee de posicionamiento y temporización a través de la señal GPS L1. Esta banda, que transmiten todos los satélites, contiene un código C/A, un mensaje de datos de navegación, que está disponible para uso civil, comercial y científico” [31].

El código C/A tiene una longitud de 1023 chips (al bit de código se le denomina "chip") con una frecuencia de código de 1,023 Mcps (f_o/10). Por tanto, la duración del código es de 1 ms y la longitud del chip corresponde aproximadamente de 293 m.

El código C/A es un código GOLD generado por dos registros de desplazamiento de realimentación lineal de 10 bits, tal como se observa en la Ilustración 4-3:

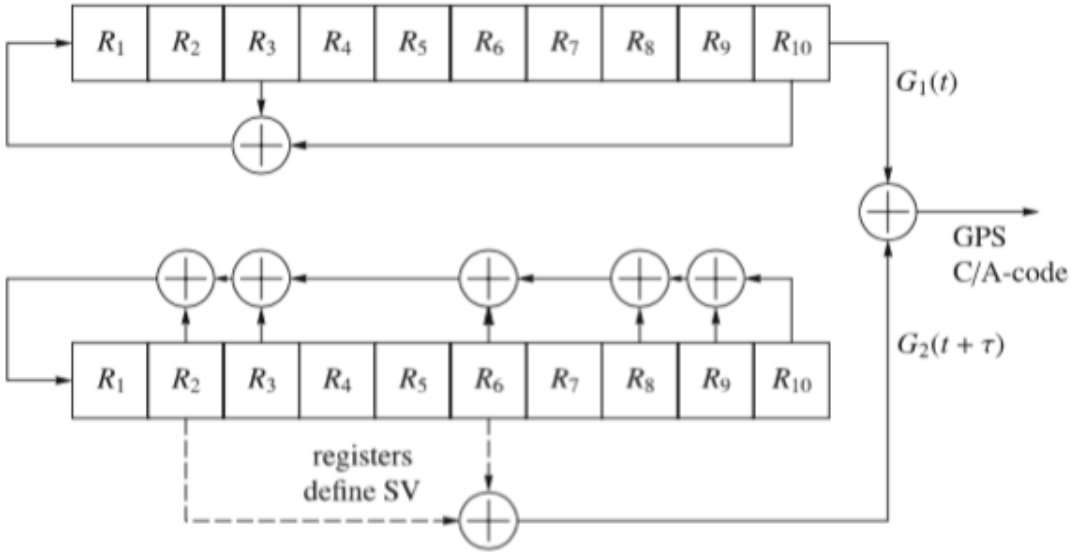

Ilustración 4-3. Esquema de generación de código C/A [31]

Los polinomios característicos son G1 (4.1) y G2 (4.2):

$$G_1 = 1 + x^3 + x^{10} \tag{4.1}$$

$$G_2 = 1 + x^2 + x^3 + x^6 + x^8 + x^9 + x^{10} \tag{4.2}$$

El estado inicial de cada uno de los registros es 1. El código C/A se genera a partir de la suma lógica XOR de los polinomios característicos G1 y G2. Por otra parte, la secuencia G2 es retrasada un número entero de chips τ (Selector de Fase) para producir secuencias de código únicas. Con lo cual se obtiene una secuencia que identifica a cada satélite.

En la Tabla 4-2, se define el código PRN asociado a cada satélite, además del selector de fase de código G2 para conseguir el retardo en chips correspondiente y sus primeros 10 chips en octal.

Tabla 4-2. Asignación de fase de código C/A

ID Satélite	Número PRN	Selector de fase de código G2	Retraso de código [chips]	Primeros 10 chips [octal]
1	1	2⊕6	5	1440
2	2	3⊕7	6	1620
3	3	4⊕8	7	1710
4	4	5⊕9	8	1744
5	5	1⊕9	17	1133
6	6	2⊕10	18	1455
7	7	1⊕8	139	1131
8	8	2⊕9	140	1454
9	9	3⊕10	141	1626
10	10	2⊕3	251	1504
11	11	3⊕4	252	1642
12	12	5⊕6	254	1750
13	13	6⊕7	255	1764
14	14	7⊕8	256	1772
15	15	8⊕9	257	1775
16	16	9⊕10	258	1776
17	17	1⊕4	469	1156
18	18	2⊕5	470	1467
19	19	3⊕6	471	1633
20	20	4⊕7	472	1715
21	21	5⊕8	473	1746
22	22	6⊕9	474	1763
23	23	1⊕3	509	1063
24	24	4⊕6	512	1706
25	25	5⊕7	513	1743
26	26	6⊕8	514	1761
27	27	7⊕9	515	1770
28	28	8⊕10	516	1774
29	29	1⊕6	859	1127
30	30	2⊕7	860	1453
31	31	3⊕8	861	1625
32	32	4⊕9	862	1712

4.2.3 Mensaje de navegación

Los mensajes de navegación contienen esencialmente información acerca de la órbita del satélite (efemérides), el estado del satélite, varios datos de corrección, mensajes de estado (horario y estado del reloj del satélite), y otros mensajes de datos (almanaque, información acerca de la constelación de satélites). La tasa de bit de datos (4.3) del mensaje de navegación es baja en comparación a la tasa de chip de los códigos PRN.

$$T_{bit,mensaje\ de\ navegación} = \frac{1}{50\ [Hz]} = \ 20\ ms$$

(4.3)

Debido a la baja SNR (Signal-Noise Ratio, relación señal-ruido), es necesaria una baja tasa de datos para garantizar una baja BER (Bit Error Rate, Tasa de error de bit). Estos mensajes de navegación son actualizados desde las estaciones de control terrestres.

La estructura del mensaje de navegación completo se representa en la Ilustración 4-4:

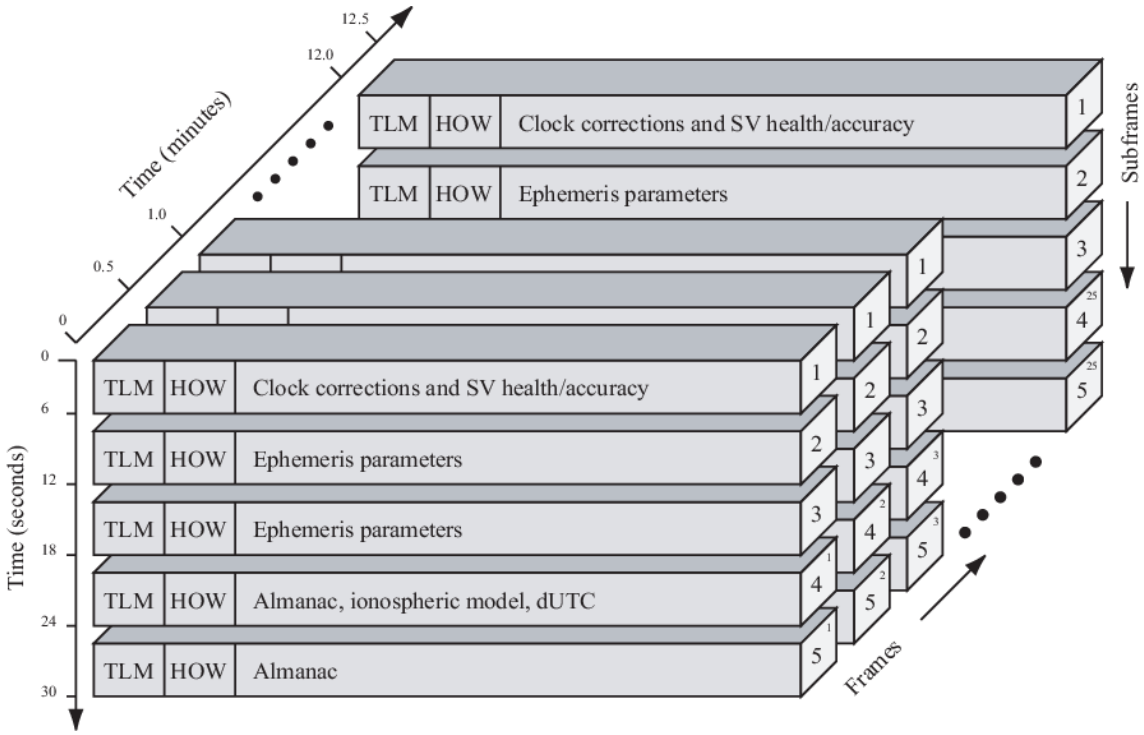

Ilustración 4-4. Estructura general del mensaje de navegación [32]

Para transmitir un mensaje de navegación completo es necesario 12,5 minutos y se transmite mediante tramas de la siguiente manera:

- 1 mensaje de navegación (12,5 min) = 25 tramas (30 s) = 37500 bits de datos
- 1 trama = 5 subtramas = 1500 bits de datos
- 1 subtrama (6 s) = 10 palabras = 300 bits de datos
- 1 palabra (600 ms) = 30 bits de datos

La estructura y el contenido de los mensajes de navegación son comunes en todos los satélites. La primera subtrama contiene información acerca de la semana GPS, coeficientes para algoritmos de corrección de reloj de los satélites además de indicadores de estado del satélite. La segunda y tercera subtrama transmiten efemérides del satélite. El contenido de las primeras tres subtramas se repite en todas las tramas. La cuarta subtrama contiene información sobre la ionosfera, los datos UTC (Universal Time Coordinated) y los datos de almanaque. La quinta subtrama está dedicada a datos de

almanaque. El contenido de las subtramas cuarta y quinta se cambian cada trama. Por otra parte, cada subtrama inicia con la palabra TLM (Telemetry) y HOW (Hand-Over Word). La palabra TLM contiene un patrón de sincronización de ocho bits (10001011) llamado preámbulo que permite reconocer el inicio de cada subtrama. La segunda palabra HOW contiene un indicador TOW (Time of Week), entre otros.

4.3 Modulación de la señal GPS L1

La señal GPS con los códigos PRN y el mensaje de navegación se transmite mediante la modulación BPSK (Binary Phase Shift Keying). BPSK es una modulación digital en la cual la portadora se transmite con un cambio de fase de 0º a 180º dependiendo si se transmite un 0 o 1 [33].

En la Ilustración 4-5, se muestra un ejemplo de modulación de la señal GPS para las bandas de frecuencia L1 y L2:

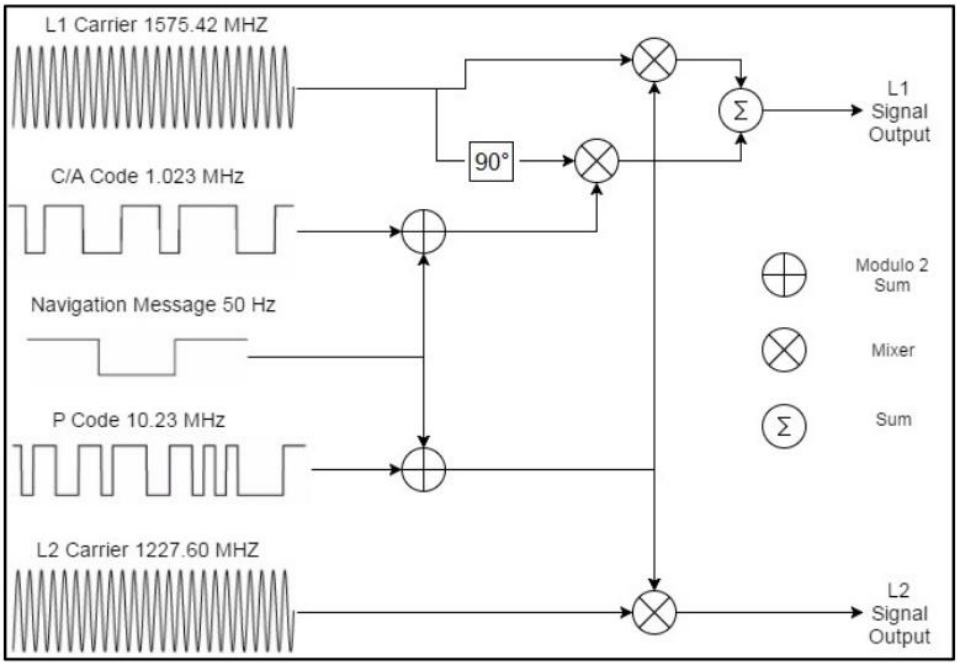

Ilustración 4-5. Modulación señal GPS [33]

El código C/A se modula solo en la frecuencia de portadora L1, mientras que el código P se modula tanto en la frecuencia L1 como en la frecuencia L2. De este modo, el código C/A se coloca en fase-cuadratura al código P.

La señal GPS correspondiente a la banda L1, de forma general es como la ecuación (4.4):

$$s_{L1}(t) = a_P[c_P(t) \oplus d(t)] \cos(2\pi f_{L1} t) + a_C\left[c_{C/A}(t) \oplus d(t)\right] \sin(2\pi f_{L1} t) \tag{4.4}$$

Donde:

- $c_P(t)$, código de precisión P (si se activa el modo Anti-Spoofing se usa el código Y, encriptado).
- $c_{C/A}(t)$, código C/A.
- $d(t)$, mensaje de navegación.
- $a_P - a_C$, atenuación de las componentes de señal.
- f_{L1}, frecuencia L1

En la Ilustración 4-6, se observan las componentes de la señal GPS, para una mejor comprensión de la modulación BPSK:

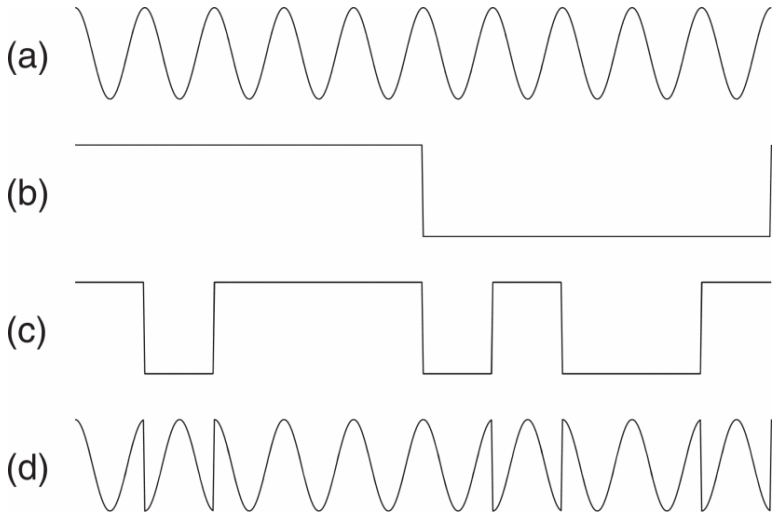

Ilustración 4-6. Componentes de la señal GPS: (a) Portadora sin modular; (b) Bits de datos del mensaje de navegación, D; (c) Chips de códigos PRN (C/A o P(Y)); (d) Portadora modulada por los datos de navegación y el Código PRN [3]

Los chips de códigos PRN (c) se suman según una adición de módulo 2 con los bits del mensaje de navegación (b). Esta adición corresponde a la operación lógica XOR ($C \oplus D$). El resultado de esta suma modula a la portadora (L1 o L2) mediante una modulación BPSK. La portadora cambia de fase 180º siempre que la salida de la operación XOR cambie.

4.4 DSSS y CDMA

4.4.1 Espectro Ensanchado

El espectro ensanchado es una técnica que permite transmitir señales con muy poca potencia en un amplio ancho de banda. Los códigos PRN pertenecen a secuencias de espectro ensanchado DSSS (Direct Sequence Spread Spectrum).

Es posible conseguir señales DSSS a partir de ondas de espectro ensanchado, tal como aparece en la Ilustración 4-7, como por ejemplo al modular una portadora RF con una onda de datos y una onda de espectro ensanchado. Tal como se detalló en el apartado 4.3, es el equivalente a la modulación BPSK de la señal GPS y dado que los códigos PRN son códigos de espectro ensanchando, la señal resultante que se transmite es una señal DSSS.

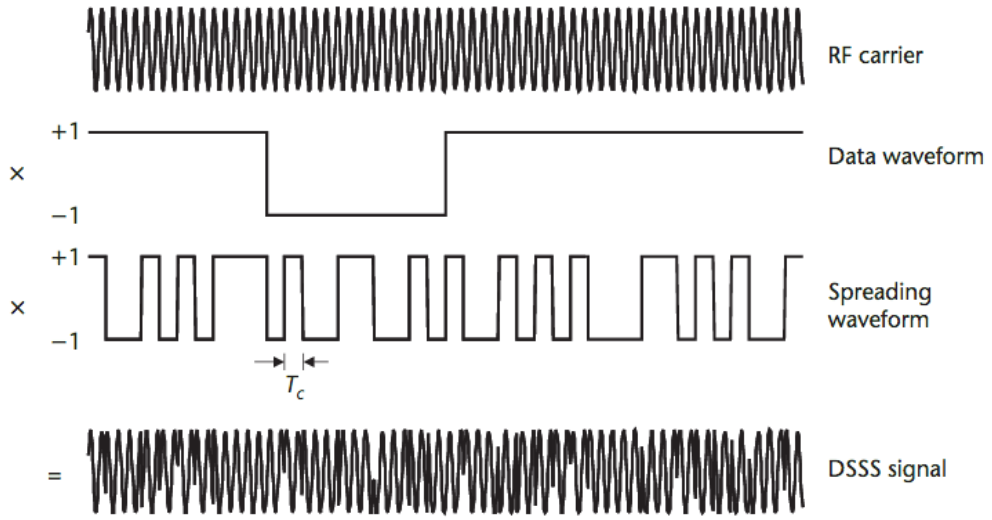

Ilustración 4-7. Modulación DSSS: RF carrier (Portadora RF); Data waveform (onda de Datos); Spreading waveform (onda de Espectro Ensanchado); DSSS signal (señal DSSS) [34]

Hay tres ventajas primordiales del uso de modulaciones DSSS sobre la señal GPS:

- Los continuos cambios de fase de los códigos PRN permiten gran precisión por parte del receptor.
- El uso de diferentes secuencias de códigos PRN permite a los receptores distinguir múltiples satélites transmitiendo simultáneamente y a la misma frecuencia. Este factor está muy relacionado con CDMA.
- Estas secuencias proveen al sistema mayor robustez frente a desvanecimiento o interferencias estrechas.
- También tiene otras ventajas como inmunidad frente al multitrayecto, capacidad de compartir el ancho de banda con otras señales, y la privacidad debido a la codificación de la información.

4.4.2 Autocorrelación cruzada y CDMA

El concepto de modulación/demodulación de la señal GPS se basa en el uso de un código PRN que identifique a cada satélite, pero que en común usen la misma tasa de chip y la misma frecuencia de portadora. Esta modulación/demodulación es posible gracias a CDMA que es una técnica de acceso múltiple que permite identificar a cada usuario mediante códigos únicos en un mismo canal de comunicación. Esta técnica hace uso de códigos PRN de espectro ensanchado. Por tanto, CDMA tendrá las mismas ventajas de autocorrelación y correlación cruzada. La función de autocorrelación de los códigos PRN queda definida en (4.5):

$$\begin{aligned} R(\tau) &= A^2\left(1-\frac{|\tau|}{T_c}\right) && para\ |\tau| \le T_c \\ &\approx 0 && para\ |\tau| > T_c \end{aligned} \tag{4.5}$$

Esta ecuación denota que la autocorrelación de un código PRN es máxima cuando $\tau \approx 0$, es decir, el retardo de fase sea nulo. Y que decrece hasta que hasta anularse cuando $\tau > T_c$, es decir, cuando el retardo de fase sea superior al periodo de chip T_c del código PRN.

En la Ilustración 4-8, se representa un ejemplo de función de autocorrelación de la secuencia *r(t)*. Apreciando gráficamente lo visto en la ecuación (4.5).

La función de autocorrelación cruzada de los códigos PRN queda definida en (4.6):

$$R_{ij}(\tau) = \int_{-\infty}^{\infty} PN_i(t)PN_j(t+\tau)d\tau \approx 0 \tag{4.6}$$

Donde:

- $PN_i(t)$ = código PRN del satélite i
- $PN_j(t)$ = código PRN del satélite j, siendo i≠j

Esta ecuación denota que ningún código PRN_i se correlaciona con otro código PRN_j para cualquier desplazamiento de fase τ.

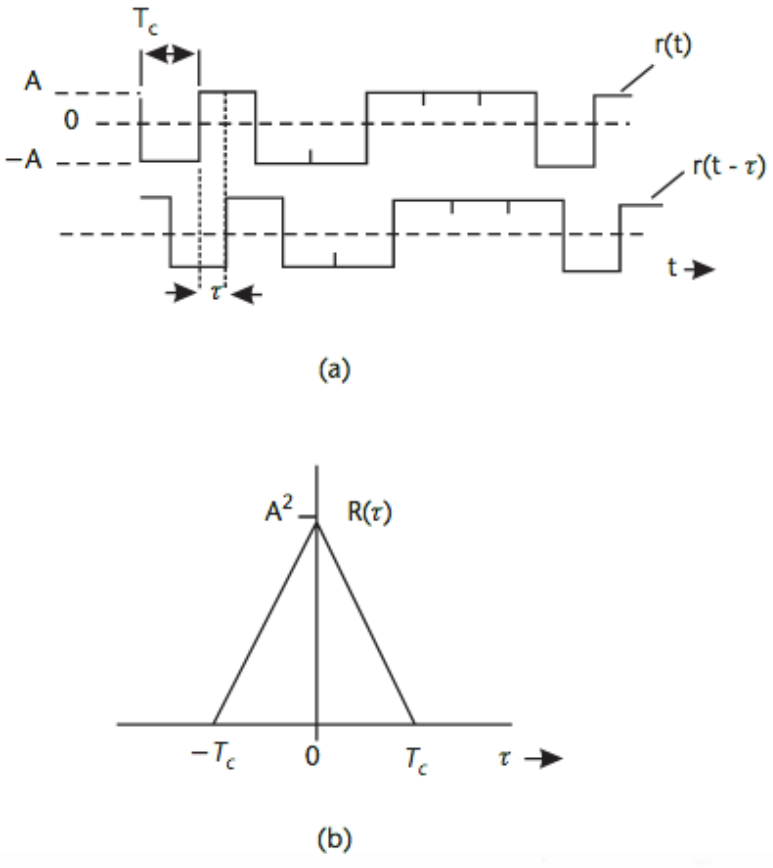

Ilustración 4-8. Ejemplo de función de autocorrelación de un código PRN: (a) ejemplo de código PRN; (b) función de autocorrelación [34]

5 Implementación de la etapa de pre-procesado

En el capítulo 3, se detallan las características generales del radar pasivo con GPS como IO, observando la necesidad de reconstrucción de la señal de Referencia previo al procesado coherente con la CAF. En el capítulo 4, se estudian las características de la señal GPS centrándose en la componente civil de la banda de frecuencias L1.

Durante este capítulo se estudian los distintos métodos que incluye el pre-procesado, haciendo hincapié en el bloque de Adquisición y Tracking de la señal de referencia.

5.1 Etapa de pre-procesado

En la Ilustración 5-1, se observa el diagrama de bloques de la etapa de pre-procesado de la señal de Referencia:

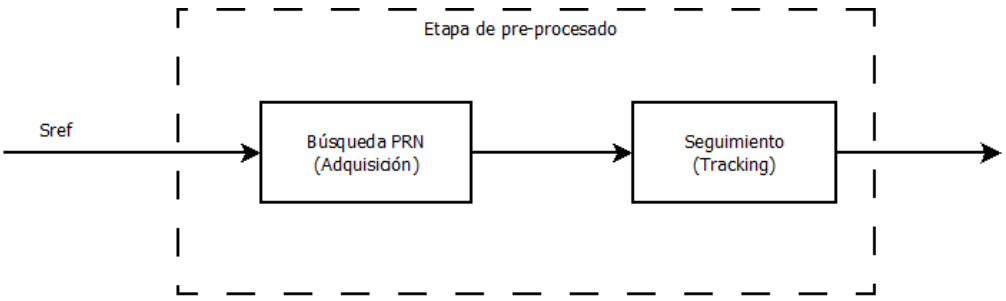

Ilustración 5-1. Diagrama de bloques de la etapa de pre-procesado de la señal de Referencia

Tal como se analizó en el apartado 3.4.1, debido a la identificación del satélite, la distancia satélite-receptor y su geometría variable, es necesario incluir una etapa de pre-procesado al esquema general de procesado del radar pasivo. Debido a estos inconvenientes es necesario identificar el satélite, así como las distorsiones que sufre la señal de Referencia (3.3). Estas distorsiones puede ser el retardo temporal del código o el desplazamiento Doppler de la portadora. Para ello, la etapa de pre-procesado incorpora dos bloques: bloque de Adquisición y bloque de Tracking [35].

5.1.1 Bloque de Adquisición: algoritmos y conclusiones

En el bloque de adquisición de la señal de referencia se busca identificar (mediante códigos PRN) el satélite y posteriormente extraer dos datos importantes para el bloque de Tracking posterior, estos son: el retardo de código C/A (asociado al inicio del código PRN) y la frecuencia Doppler (asociada a la frecuencia de la portadora de la señal de Referencia).

El retardo de código es necesario para comprimir[1] la señal de Referencia, es decir, demodular la señal obteniéndose una señal de onda continua cuya frecuencia Doppler puede ser encontrada.

Para el bloque de Adquisición hay que tener en cuenta lo siguiente:

- Hay que buscar un método que compense entre velocidad computacional y sensibilidad. Ya que, si la señal de entrada es fuerte, es decir, su SNR es alta, no hay ningún inconveniente en usar un algoritmo con baja sensibilidad, pero si la señal es débil, un algoritmo de baja sensibilidad no sería capaza de encontrar ninguna señal, con lo cual convendría incrementar la longitud de datos de adquisición, es decir, incrementar el tiempo de Adquisición.
- De forma convencional se puede realizar la Adquisición mediante hardware, donde la entrada de datos se realiza de manera continua, y una vez se adquiere el retardo y la frecuencia, se transmiten inmediatamente al hardware de Tracking. Este proceso de adquisición por hardware también se puede realizar en paralelo.
- La adquisición mediante software es muy diferente ya que se realiza mediante bloques de datos almacenados. De igual manera cuando los parámetros se captan pasan al Tracking mediante software. Por tanto, hay un intervalo de tiempo entre los datos recibidos y los datos que se rastrean en adquisición. Por ello el uso de bloques de adquisición por software es factible ya que puede procesar datos almacenados, aunque la búsqueda de adquisición en tiempo real sería el camino ideal. Luego es muy importante implementar un sistema de adquisición cuyo tiempo de desempeño sea el menor posible. Una de las características fundamentales es la elección de la longitud de los bloques de datos. Una mayor longitud de datos genera una relación señal-ruido mayor. Con lo cual, lo ideal sería usar un bloque de datos extenso, pero requiere un mayor tiempo de cálculo. Por otra parte, hay otros factores que limitan el tamaño de los bloques, como lo es la transición de datos de navegación y el efecto Doppler en el código C/A. Ambos generan inconvenientes en la adquisición. El primero de ellos, teóricamente puede producir un ensanchado del espectro, con lo cual, obtener una señal sinusoidal para la recuperación de frecuencia se complica, y el segundo, puede producir una mala correlación y por tanto una baja relación señal-ruido. Para evitar estos problemas se ha dispuesto que la adquisición trabaje con dos bloques de datos de entrada de 1 ms cada uno.
- Otro factor para tener en cuenta es el intervalo de búsqueda de la frecuencia de la portadora. Ya que, en la correlación de dos señales, si la separación de frecuencia es superior a medio ciclo, el resultado tiende a cero, por tanto, el límite de intervalo de frecuencias será de 0,5 ciclos como máximo. En otras palabras, si el bloque de datos es de 1 ms, una señal de 1 KHz, tome 1 ciclo en 1 ms, y como el límite máximo se ha establecido en 0,5 ciclos, el intervalo de búsqueda de frecuencia del bloque de adquisición es de 500 Hz. Se ha dispuesto también que el rango total de desplazamiento Doppler es de ±10 kHz. Para realizar una

[1] La señal de Referencia es una señal de espectro ensanchado, por ello, al localizar el retardo del código es posible demodular la señal, comprimiéndose el espectro de la señal resultante.

búsqueda rápida el intervalo de búsqueda no puede ser muy estrecho, ya que esto implica incrementar el número de intervalos de frecuencias para cubrir el rango deseado. Un intervalo de búsqueda estrecho implica un incremento de la sensibilidad. Con lo cual, la sensibilidad del bloque de adquisición es una conjunción entre tiempo y ancho del intervalo de búsqueda.

Una vez vistos los elementos necesarios para el bloque de adquisición, se comprueban los diferentes algoritmos que existen para este desempeño.

5.1.1.1 Algoritmo de búsqueda en serie [36]

El algoritmo de búsqueda en serie es un método de adquisición básico en sistemas CDMA.

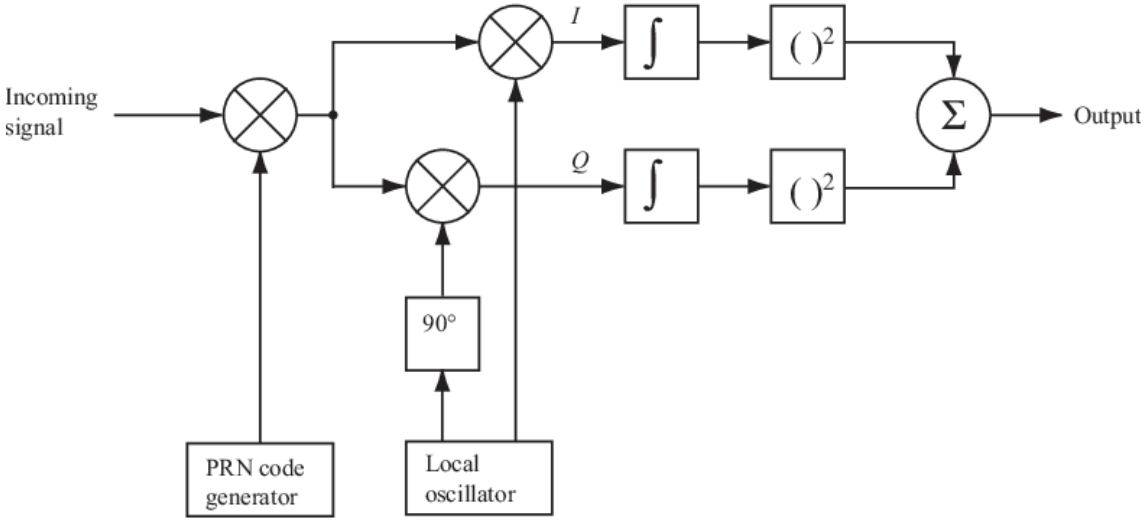

Ilustración 5-2. Diagrama de bloques del algoritmo de búsqueda en serie [32]

En la Ilustración 5-2, se representa el diagrama de bloques del algoritmo de búsqueda en serie [32]. Este algoritmo de búsqueda se basa en el producto de secuencias de código PRN (PRN code generator) y de portadoras (Local oscillator), ambas generadas localmente. Inicialmente se realiza el producto entre la señal de entrada (Incoming Signal) y las copias de secuencias de código PRN generadas con retardo de fase.

En la Ilustración 5-3, se observa la longitud de cada secuencia generada (Replica code) es de L=1023 chips, por tanto, la secuencia de señal de entrada (Received code) debe ser del doble de longitud que la secuencia generada.

A continuación, se consiguen las componentes en fase I y cuadratura Q de la señal, mediante un oscilador local, que genera tantas portadoras como frecuencias de búsqueda se desee analizar. El rango de búsqueda es de IF±10 kHz con pasos de 500 Hz, es decir, un total de 41 frecuencias de búsqueda. La componente en fase I se obtiene a partir del producto con la señal portadora y la componente en cuadratura Q se obtiene a partir del producto con la señal portadora desfasada 90º.

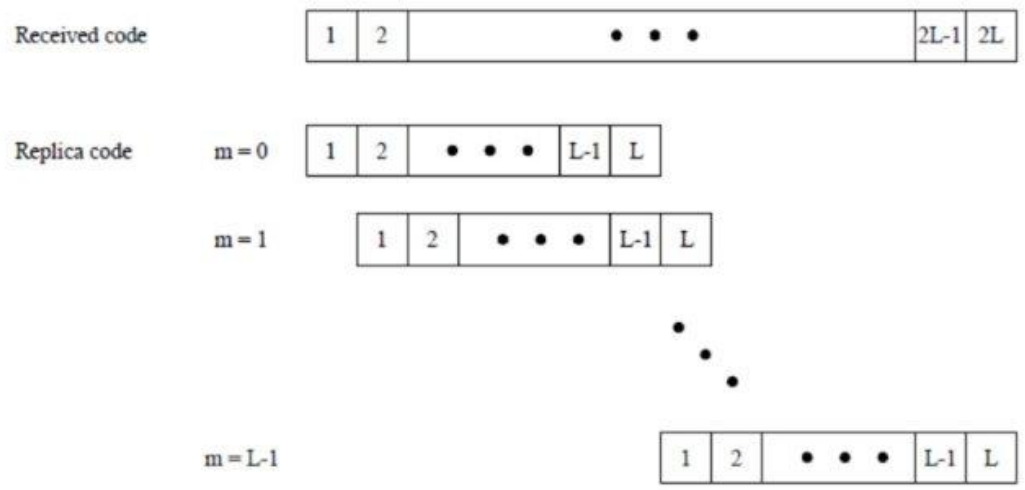

Ilustración 5-3. Generador de secuencias de código PRN [36]

El resultado es la correlación entre la señal de entrada y una señal generada localmente. A priori, la componente en fase I debe tener mayor correlación[2] pero como la fase de la señal recibida es desconocida se debe examinar ambas componentes. Por ello, ambas componentes deben pasar por un bloque de integración, elevarlas al cuadrado y sumarlas. Al elevar al cuadrado lo que se consigue es la potencia de la señal.

El resultado de la adquisición por búsqueda en serie se resume en la expresión (5.1):

$$R^2[m] = \sum_{j=0}^{K-1} \left(\left[\sum_{n=jNL}^{(j+1)N(L-1)} x[n] \cdot CA[n] \cdot \cos[\Omega n] \right]^2 + \left[\sum_{n=jNL}^{(j+1)N(L-1)} x[n] \cdot CA[n] \cdot \sin[\Omega n] \right]^2 \right) \quad (5.1)$$

- L = longitud de la secuencia de código PRN
- m = retardo temporal de código PRN, m = 0 … L-1
- N = número de periodos de código. Hace referencia a los periodos de integración de 1 ms
- K = número de correlaciones

En la ecuación (5.1), se observan los valores N y K que si se incrementan mejorarán los niveles de SNR y también reducirán la probabilidad de falsa alarma. Pero esto perjudicará el rendimiento computacional del método, ya que provocará grandes periodos de pre-

[2] El código C/A va modulado en Fase en la señal GPS

procesado. Un elevado nivel de *N* puede acarrear problemas con las transiciones del mensaje de navegación. Por otra parte, incrementar el nivel de *K* requiere también incrementar el nivel de *Threshold*[3] para evitar problemas de falsa adquisición.

Si el código está alineado y la frecuencia Doppler coincide con la frecuencia local generada de entrada se produce un máximo de correlación, obteniéndose de esta manera el retardo de fase y la frecuencia de portadora aproximada.

5.1.1.2 Algoritmo de búsqueda paralela en la fase del código [36]

Una mejora del algoritmo de adquisición por búsqueda en serie es aplicar la correlación mediante la convolución circular de señales, aplicado en el algoritmo de búsqueda paralela en la fase del código. En el presente documento, este método de adquisición se denominará ***algoritmo de Adquisición gruesa***.

La correlación entre la señal de entrada y una secuencia de código C/A se puede aplicar mediante desplazamientos circulares del código C/A (5.2):

$$R[m] = \sum_{n=0}^{L-1} x[n] \cdot CA[(n+m)_L] \tag{5.2}$$

Siendo:

- L = número de chips
- m = retardo temporal que toma el código PRN

En la Ilustración 5-4, se observa el desplazamiento circular aplicado sobre la secuencia de código PRN generada, visto en la ecuación (5.2):

La ecuación (5.2) equivale a una convolución circular de la siguiente manera (5.3).

$$R[m] = x[n] \otimes CA[-n] \tag{5.3}$$

Si a la convolución circular se le aplica las propiedades de la transformada de Fourier, se obtiene la siguiente expresión (5.4). De esta forma no es necesario generar tantas secuencias de código PRN como retardos temporales de búsqueda sean necesarios.

$$R[m] = \mathfrak{F}^{-1}(\mathfrak{F}(x[n]) \cdot \mathfrak{F}(CA[n])^*) \tag{5.4}$$

[3] Threshold. Umbral de correlación. Es una medida de calidad de la etapa de Adquisición

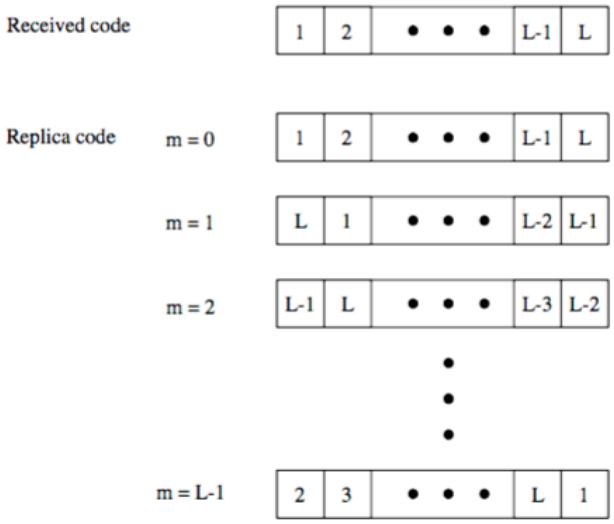

Ilustración 5-4. Desplazamiento circular sobre secuencias de código PRN [36]

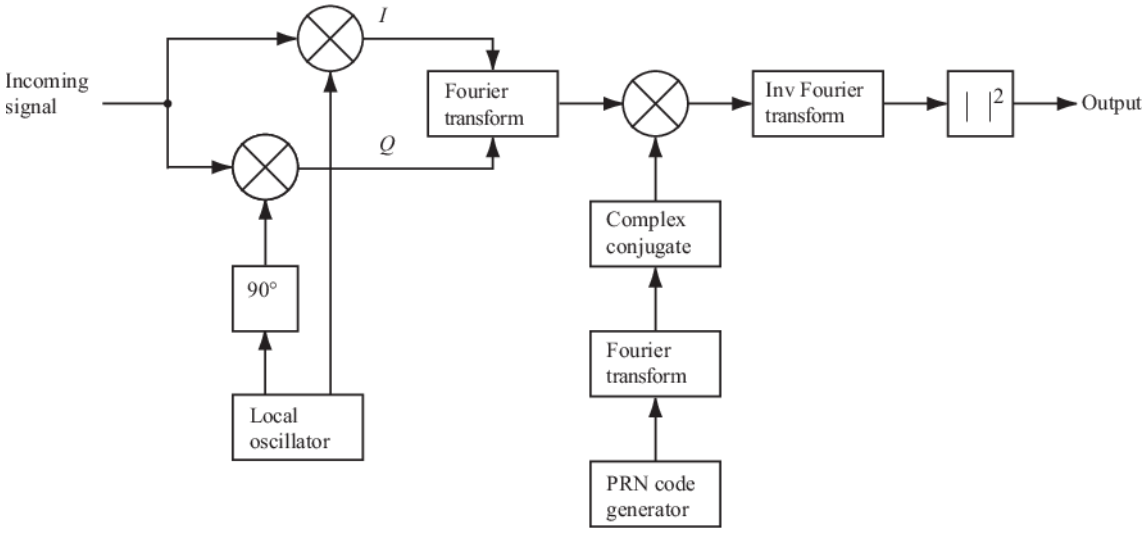

Ilustración 5-5. Diagrama de bloques del algoritmo de Adquisición gruesa [32]

En la Ilustración 5-5, se observa el diagrama de bloques del *método de adquisición gruesa* para calcular la correlación (5.4). Inicialmente se consiguen las componentes en fase I y cuadratura Q de la señal (Incoming signal), mediante un oscilador local (Local oscillator), que genera tantas portadoras como frecuencias de búsqueda se desee analizar. El rango de búsqueda es de IF±10 kHz con pasos de 500 Hz, es decir, un total de 41 frecuencias de búsqueda. Al igual que el método en serie, la componente en fase I se obtiene a partir del producto con la señal portadora y la componente en cuadratura Q se obtiene a partir del producto con la señal portadora desfasada 90º. Que en este caso se combinan para formar una señal compleja x(n)=I(n)+jQ(n). Se realiza el producto entre la DFT (Discrete Fourier Transform, Transformada Discreta de Fourier) de la señal compleja y DFT conjugada del código C/A generado localmente. El resultado se encuentra en el dominio frecuencia, luego se aplica la IDFT (Inverse Discrete Fourier Transform, Transformada inversa discreta de Fourier) para devolver la señal resultante al dominio temporal, obteniendo la correlación entre la señal de entrada y el código C/A [32].

Con este algoritmo se reduce la carga computacional al eliminar las combinaciones generadas por el retardo temporal. El número de frecuencias de búsqueda, al igual que el algoritmo anterior, se mantiene en 41.

5.1.1.3 Algoritmo de búsqueda paralela en el espacio de frecuencia [35]

Para la comprensión de este algoritmo es preciso conocer previamente la demodulación del código PRN.

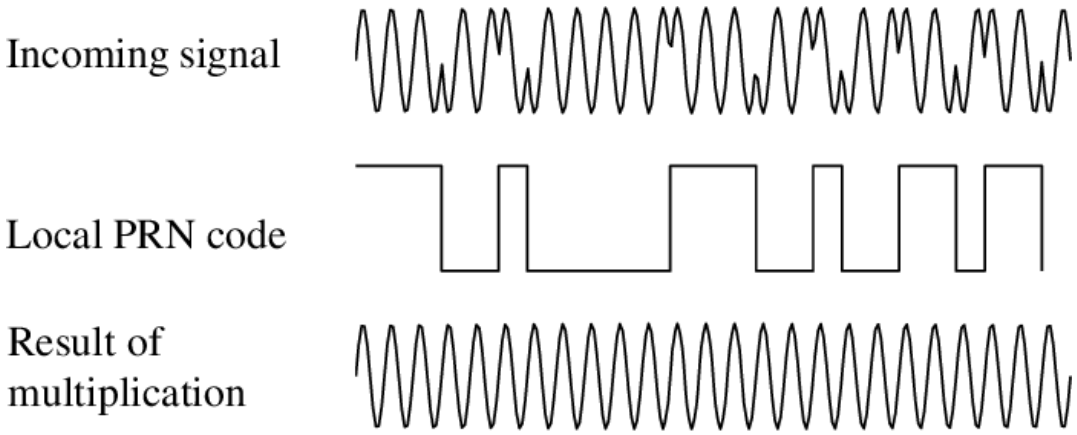

Ilustración 5-6. Demodulación del código PRN [32]

En la Ilustración 5-6, se describe el proceso de demodulación del código PRN. Para comprimir el espectro de la señal de entrada (Incoming signal) es necesario que el código PRN con el que está modulado esté alineado con la secuencia de código PRN generada localmente (Local PRN code). El resultado de su producto (Result of multiplication) es una onda continua si están perfectamente alineadas.

En este proceso de demodulación de código PRN se basa el algoritmo de búsqueda paralela en el espacio de frecuencia, que en el presente documento se le denominará ***algoritmo de adquisición fina***.

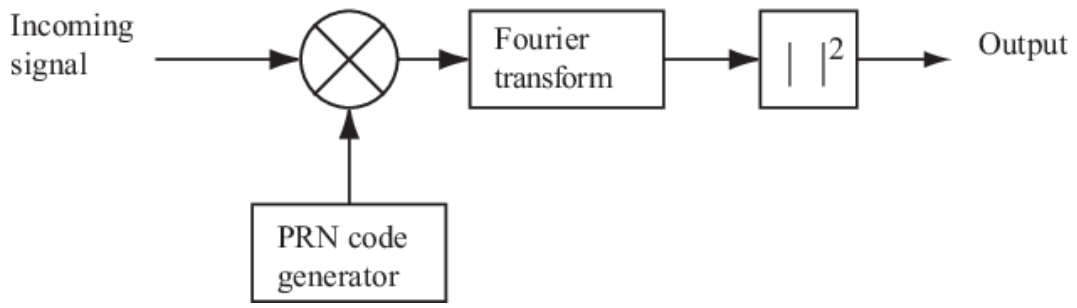

Ilustración 5-7. Diagrama de bloques del algoritmo de adquisición fina [32]

En la Ilustración 5-7, se observa el diagrama de bloques del algoritmo de Adquisición fina que inicialmente incluye un demodulador de código PRN, y una vez se elimina el código PRN de la señal de entrada se analiza el espectro de la señal resultante a partir del módulo al cuadrado de la DFT [32], obteniéndose la densidad espectral de potencia (PSD, Power Spectral Density).

5.1.1.4 Ejemplo de Adquisición

Dados los algoritmos de adquisición 5.1.1.1, 5.1.1.2 y 5.1.1.3, en el presente documento se ha seleccionado una mezcla de dos algoritmos para el bloque de adquisición de la señal de referencia. Estos son el algoritmo de adquisición gruesa y el algoritmo de adquisición fina.

Ambos algoritmos comparten que son más rápidos computacionalmente que el algoritmo de búsqueda en serie:

- En el caso del algoritmo de búsqueda en serie el número de combinaciones depende del retardo en fase que se aplica a las secuencias de código C/A locales (1023) y de la búsqueda en frecuencia que se aplica a las portadoras locales (41), que provocan un total de 41943 combinaciones.
- En el caso del algoritmo de adquisición gruesa el número de combinaciones se reduce al número de frecuencias de búsqueda, es decir, un total de 41 combinaciones.
- En el caso del algoritmo de adquisición fina el número de combinaciones depende del retardo temporal que se aplica a las secuencias de código C/A locales, es decir, 1023 combinaciones. En este ejemplo de adquisición el código C/A local se encuentra alineado con el código C/A que modula la señal, con lo cual el número de combinaciones se reduce a una.

Y si además de realizar el cálculo para identificar un solo satélite, se realiza para toda la constelación de satélites GPS, el número de combinaciones se dispara. A priori, por el número de combinaciones el algoritmo seleccionado debe ser el algoritmo de búsqueda gruesa, pero la estimación en frecuencia es inferior al algoritmo de adquisición fina. Es por esta razón que se decide realizar una mezcla de ambos algoritmos.

Durante la primera etapa del bloque de Adquisición se aplica el algoritmo de Adquisición gruesa, dado que tiene mejores prestaciones[4] en la estimación de fase que el algoritmo de Adquisición fina.

En la Ilustración 5-8, se observa la matriz de correlación obtenida por el algoritmo de Adquisición gruesa para el PRN 13.

Esta matriz de correlación sólo se ha podido obtener si el pico de correlación es superior al Threshold establecido. El código PRN identificado ha sido el 13 asociado al satélite número 13. Otros datos de interés que se pueden sacar son el retardo de fase localizado en la muestra 11735 que corresponde al chip 480 de la secuencia generada de código C/A.

[4] La estimación de fase del algoritmo de Adquisición gruesa va asociada a la longitud de la secuencia de código C/A generada localmente. $L_{secuencia\ de\ código\ C/A} = \frac{F_S \cdot L_{C/A}}{F_{C/A}} = 25000\ muestras$, siendo F_S=25 MHz, $L_{C/A}$=1023 chips y $F_{C/A}$=1,023 MHz

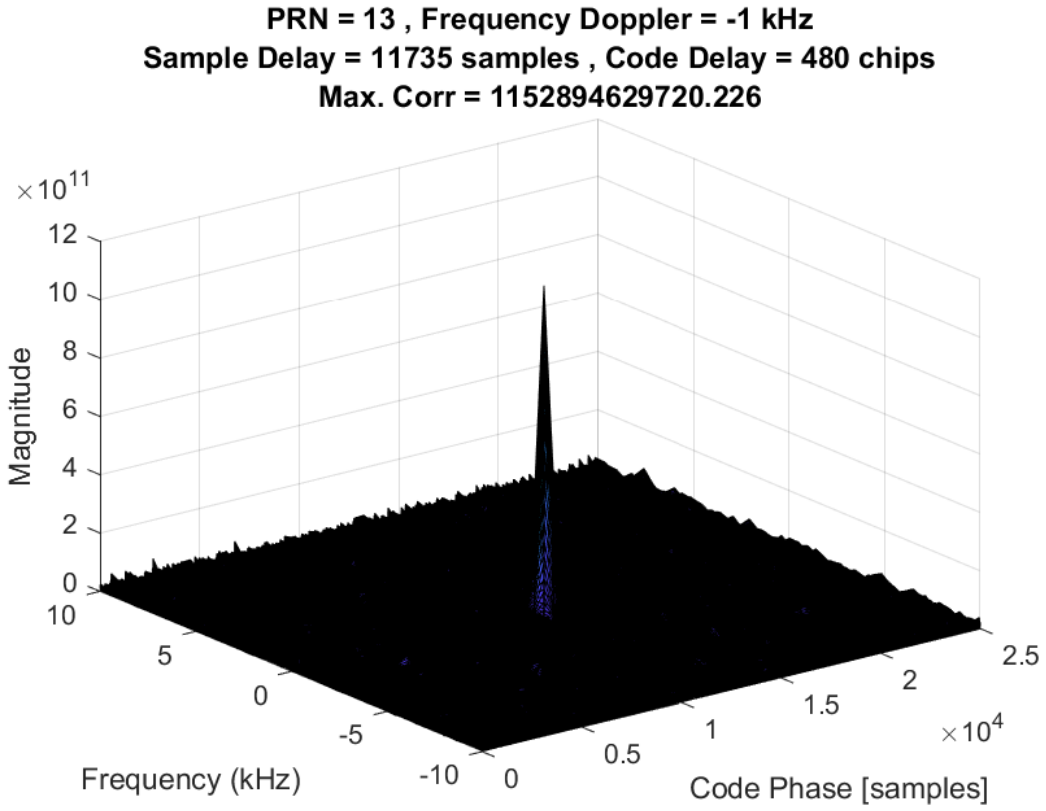

Ilustración 5-8. Matriz de correlación obtenida aplicando el algoritmo de adquisición gruesa

Por último, la frecuencia Doppler es de -1 kHz. Hay que recordar que el Algoritmo de Adquisición gruesa tiene una resolución en frecuencia de 500 Hz. Por esta razón, se mezclan ambos algoritmos, ya que el Algoritmo de Adquisición fina tiene una resolución de frecuencia superior. Para incrementar la resolución en frecuencia se ha optado por incrementar la resolución de la FFT, incrementando el número de bloques de señal de entrada (cada bloque de 1ms de señal está compuesto por 25000 muestras) de 1 ms a 10 ms y multiplicando el número de muestras resultante por un factor de 8, para conseguir una FFT de $2 \cdot 10_6$, e incrementar la resolución hasta los 12,5 Hz[5].

Ahora con el retardo de fase ya calculado se aplica el algoritmo de Adquisición fina. Con el código alineado ya es posible eliminar el código C/A de la señal de entrada y analizar en frecuencia con mayor precisión.

En la Ilustración 5-9, se observa la PSD resultante, donde la frecuencia Doppler se sitúa en -1084.80 kHz, mejorando las prestaciones en frecuencia del bloque de Adquisición de la señal de referencia.

[5] La resolución en frecuencia del algoritmo de Adquisición fina está asociado a la longitud de la FFT que se aplica. $R_{FFT} = \frac{F_s}{L_{FFT}} = 12{,}5\ Hz$, siendo F_s=25 MHz y $L_{FFT}=8 \cdot 10 \cdot L_{secCA/}=2 \cdot 10_6$ samples

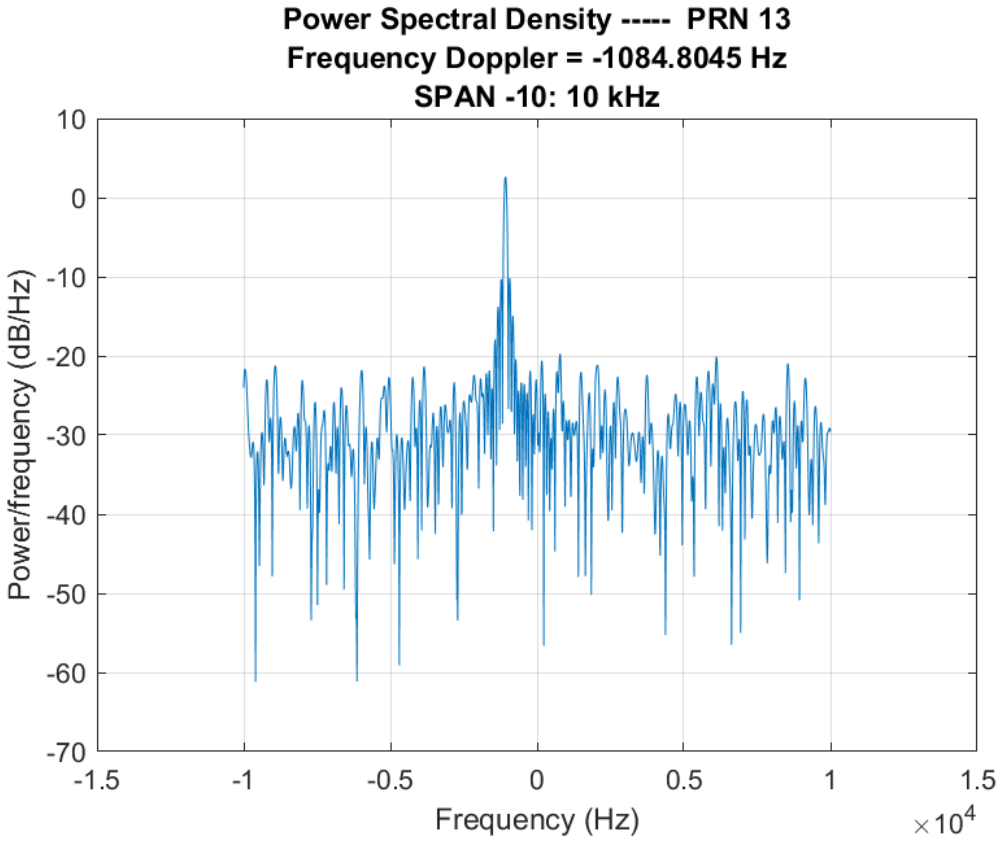

Ilustración 5-9. PSD resultante al aplicar el algoritmo de Adquisición fina

Por estas razones la solución que se ha llevado a cabo en el bloque de Adquisición es mezclar la Adquisición gruesa con la Adquisición fina. La primera de ellas da un buen resultado en tiempo, mientras que la segunda da un excelente resultado en frecuencia, elevando la sensibilidad y reduciendo la carga computacional.

5.1.2 Demodulador de la señal GPS [32]

Como la señal que se recibe por el canal de referencia del radar pasivo se encuentra en constante cambio debido al movimiento del satélite se genera una dependencia temporal sobre los parámetros de retardo de fase y desplazamiento Doppler (calculados en primera instancia en el bloque de Adquisición) que es necesario contrarrestar. La Ilustración 5-10, muestra el diagrama de bloques del demodulador de la señal GPS.

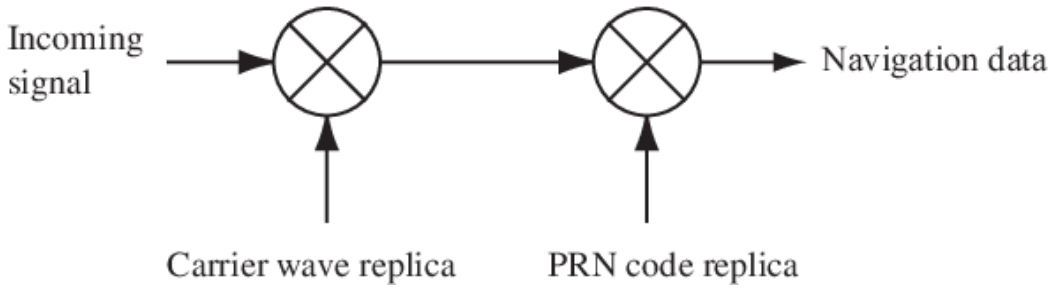

Ilustración 5-10. Diagrama de bloques del demodulador básico de la señal GPS [32]

En este esquema básico la señal de entrada (Incoming Signal) es multiplicada por una réplica de la portadora (Carrier wave replica). Esto elimina la portadora de la señal. El siguiente paso es multiplicar la señal resultante por una réplica de código PRN (PRN code replica), obteniéndose el mensaje de datos de navegación (Navigation data). Luego la finalidad del bloque de Tracking es conseguir copias exactas de la portadora y del código PRN.

Aplicando este demodulador básico sobre un modelo de señal equivalente a la parte civil de la señal GPS (3.2), se obtiene la señal (5.5):

$$S_i[n] = CA_i[n] \cdot N_{M_i}[n] \cdot \cos[2\pi f_I n] + e[n] \tag{5.5}$$

Siendo:

- *e[n]* es la componente de ruido aditivo
- f_I es la frecuencia intermedia del bloque Front-End

Luego, el primer paso es multiplicar la secuencia de entrada (5.5) por una réplica de portadora (5.6):

$$S_i[n]\cos[2\pi f_I n] = \frac{1}{2} CA_i[n] \cdot N_{M_i}[n] + \frac{1}{2} CA_i[n] \cdot N_{M_i}[n] \cos[2 \cdot 2\pi f_I n] \tag{5.6}$$

A continuación, se aplica un filtro paso bajo sobre (5.6) para eliminar esta componente $2w_i$. A la salida del filtro paso bajo se obtiene el mensaje de navegación junto con el código *C/A* (5.7):

$$S_{out-filter_i}[n] = \frac{1}{2} CA_i[n] \cdot N_{M_i}[n] \tag{5.7}$$

Finalmente, para extraer el mensaje de navegación hay que eliminar el código C/A. Para ello se realiza la correlación entre la señal y una réplica del código (5.8), siendo N el número de muestras de la secuencia de entrada

$$N_{M_i}[n] = \frac{1}{N} \sum_{n=0}^{N-1} \frac{1}{2} CA_i[n] \cdot CA_i[n] \cdot N_{M_i}[n] \tag{5.8}$$

5.1.3 Bloque de Seguimiento (Tracking) [32]

El bloque de tracking es un bloque de realimentación que consta de dos bucles, el bucle de seguimiento de portadora o PLL (Phase Lock Loop) y el bucle de seguimiento de código o DLL (Delay Lock Loop).

5.1.3.1 PLL

Para el PLL se selecciona el bucle de Costas por su insensibilidad a los cambios de fase en las transiciones del mensaje de datos de navegación.

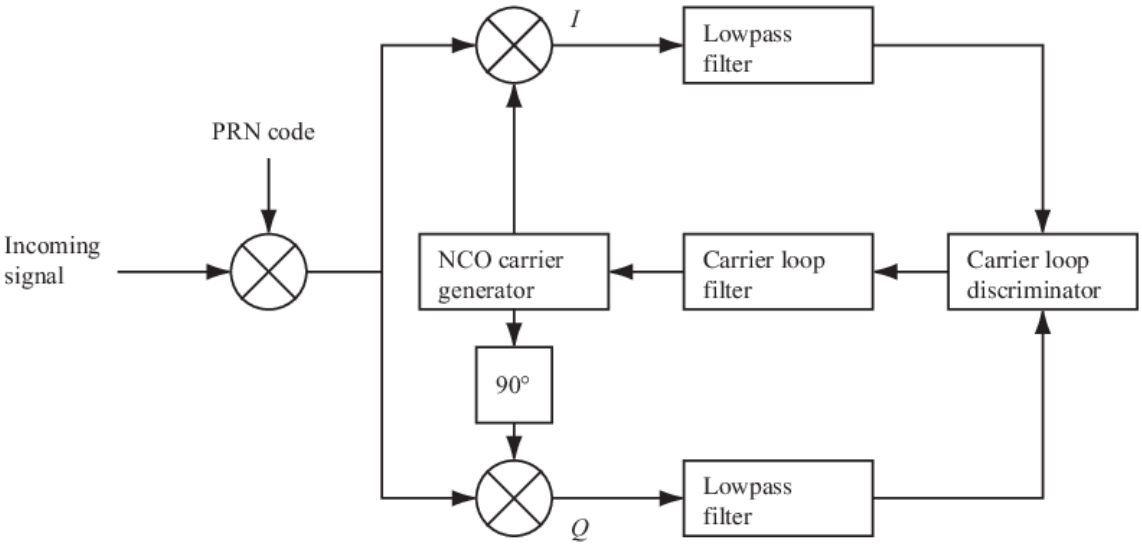

Ilustración 5-11. Diagrama de bloques del PLL [32]

En la Ilustración 5-11, se observa el diagrama de bloques del PLL. Para la comprensión de este diagrama se presupone que el código C/A (PRN code) está alineado con la señal de entrada. En la rama en fase I, se tiene la siguiente secuencia (5.9), siendo ψ la diferencia de fase que aplica el NCO al generar la portadora.

$$\begin{aligned} I_i[n] &= N_{M_i}[n]\cos[2\pi f_I n]\cos[2\pi f_I n + \psi] \\ &= \frac{1}{2}N_{M_i}[n]\cos[\psi] + \frac{1}{2}N_{M_i}[n]\cos[2\cdot 2\pi f_i n + \psi] \end{aligned} \tag{5.9}$$

En la rama en cuadratura Q, se tiene la siguiente secuencia (5.10):

$$\begin{aligned} Q_i[n] &= N_{M_i}[n]\cos[2\pi f_I n]\cos[2\pi f_i n + \psi] \\ &= \frac{1}{2}N_{M_i}[n]\sin[\psi] + N_{M_i}[n]\sin[2\cdot 2\pi f_i n + \psi] \end{aligned} \tag{5.10}$$

Al pasar ambas ramas por un filtrado paso bajo se obtiene: (5.11) para la componente en fase I y (5.12) para la componente en cuadratura Q:

$$I_i = \frac{1}{2}N_{M_i}[n]\cos[\psi] \tag{5.11}$$

$$Q_i = \frac{1}{2}N_{M_i}[n]\sin[\psi] \tag{5.12}$$

Aplicando el discriminador (5.13) sobre (5.11) y (5.12) se obtiene el error de fase ψ:

$$Discriminador_{salida} = \tan^{-1}\left(\frac{Q_i}{I_i}\right) = \psi$$

(5.13)

A partir de esta ecuación (5.13) es sencillo comprobar que si la fase de la señal rota 180º debido al mensaje de navegación la salida del discriminador no varía.

En la Ilustración 5-12, se observa un diagrama fasorial del comportamiento del bucle de Costas cuando la señal de entrada cambia de fase por la transición del mensaje de navegación. Rotando 180º, la salida del discriminador hace que rote ψ hasta conseguir que la señal de entrada se encuentre en Fase.

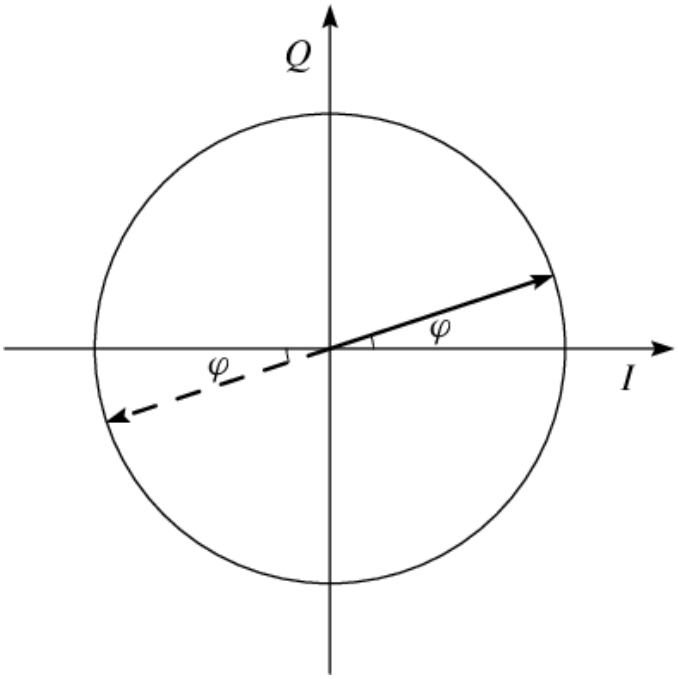

Ilustración 5-12. Diagrama Fasorial del bucle de Costas llevando la energía de la señal a la componente en Fase [32]

El error de fase usando este discriminador es el error de fase real, pero debido al ruido aditivo que se introduce en la señal de entrada al PLL es necesario introducir un filtro de segundo orden (Carrier Loop Filter) para que a la salida del NCO se genere una portadora con el error de fase correcto.

La salida del NCO genera, por tanto, las siguientes componentes I, Q (5.14) (5.15).

$$Portadora^I = \cos(2\pi f_I n + \psi)$$

(5.14)

$$Portadora^{Q} = \sin(2\pi f_I n + \psi)$$

(5.15)

Si la portadora generada por el NCO (NCO Carrier generator) es correcta, el resultado es que la señal se mantiene alineada con la componente en fase I. Como el código C/A se encuentra en la componente en fase I de la señal, el bucle de Costas intenta mantener la energía en esta rama. En el caso de que se produzca una transición del mensaje de datos de navegación el bucle de Costas actúa, evitando cualquier inconveniente.

5.1.3.2 DLL

En la Ilustración 5-13, se aprecia un diagrama de bloques sencillo del DLL, centrándose en la componente en fase de la señal. Esta distribución supone que se genera una réplica local de la portadora (Local oscillator), es decir, que la fase y la frecuencia de la portadora están alineadas con la señal entrante.

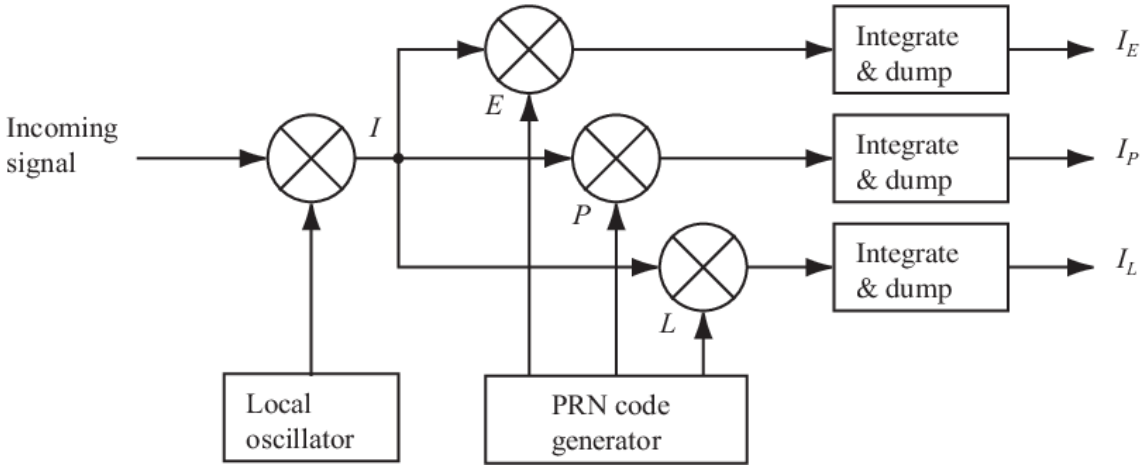

Ilustración 5-13. Diagrama de bloque del DLL con tres correladores [32]

Para el DLL de forma general se hace uso de las propiedades de autocorrelación de los códigos C/A. La idea es llevar la señal de entrada (Incoming signal) a banda base al multiplicarla por la réplica de portadora. A continuación, se realiza el producto de la señal resultante con tres réplicas de código C/A, una adelantada[6](Early), una retrasada (Late)y una alineada (Prompt). Este margen de adelanto o retraso provoca la pérdida de la mitad de energía en el cálculo de la correlación, tal como se aprecia en la Ilustración 5-14.

En (a) se observa que la réplica Late genera mayor correlación con la señal de entrada por lo que la fase del código debe retrasarse, y en (b) se observa como la réplica Prompt tiene mayor correlación por lo que el código está perfectamente alineado.

6 El margen de adelantar o retrasar una réplica de código C/A es de 0,5 chip

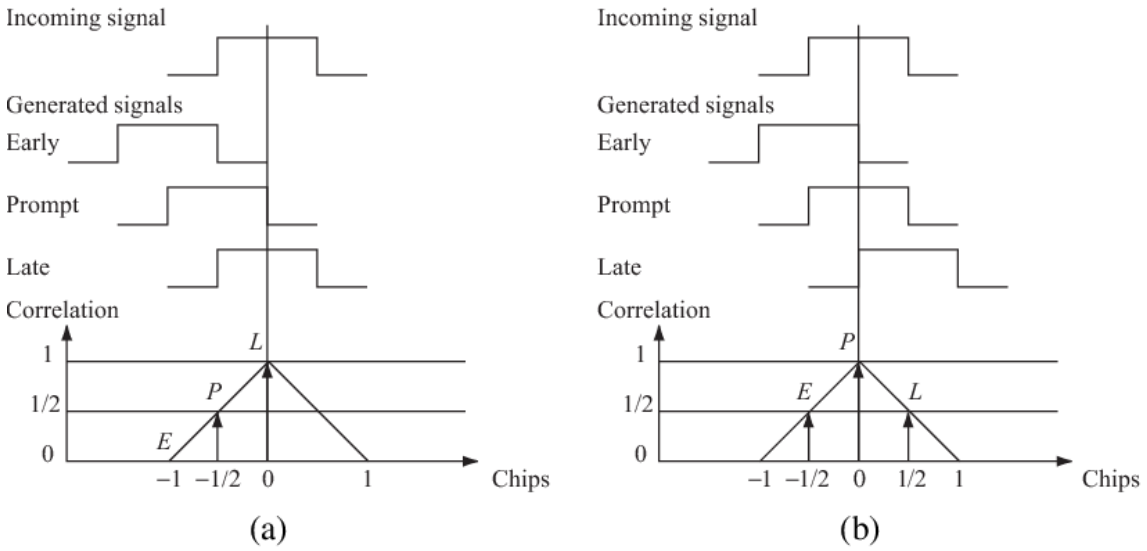

Ilustración 5-14. Correlación entre la señal entrante y las réplicas de los códigos C/A, Early, Prompt y Late [32]

En el caso de que la portadora generada por el oscilador local no esté alineada en fase y en frecuencia con la portadora de la señal entrante es necesario comprobar la rama en cuadratura Q. En la Ilustración 5-15, se observa la nueva configuración del bloque DLL con un total de 6 correladores, tres en la rama en fase I, y tres en la rama de cuadratura Q. Esta nueva configuración asume que no siempre la energía se encontrará en la rama I.

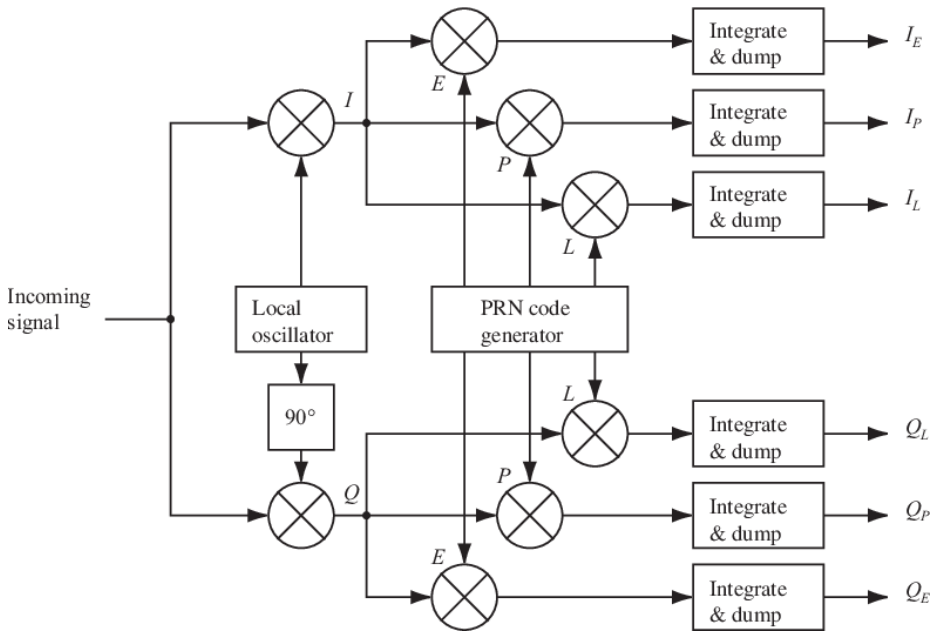

Ilustración 5-15. Diagrama de bloques del DLL [32]

En el caso de que la portadora sí esté en fase con la señal de entrada, toda la energía se concentrará en fase. Pero en el caso de que la portadora no esté que la fase con la señal entrante, la energía se distribuirá por ambas ramas. Al tener correladores en ambas ramas, la falta de sincronía del PLL no es un inconveniente a la hora de alinear el código.

Al igual que el PLL, en el DLL también es necesario un bucle de realimentación que sea sensible a las variaciones temporales de la señal recibida. En la Ilustración 5-16, se puede observar que para a la salida de los correladores del DLL se incluye un discriminador (Code Loop Discriminator), un filtro de segundo orden (Code Loop Filter) para filtrar el ruido, y posteriormente el generador de código (PRN code generator). El discriminador del DLL es del tipo (5.16). Se ha seleccionado este discriminador no coherente debido a sus buenas propiedades cuando el error de fase es superior a 0.5 chip.

$$Discriminador_{DLL} = \frac{(I_E^2 + Q_E^2) - (I_L^2 + Q_L^2)}{(I_E^2 + Q_E^2) + (I_L^2 + Q_L^2)} \tag{5.16}$$

La salida del discriminador del DLL también es ruidosa, por tanto, hay que aplicar un filtro para reducir el ancho de banda de ruido del DLL. Una vez filtrada la salida del DLL se generan nuevas réplicas de los códigos C/A.

En la Ilustración 5-16, se observa el diagrama de bloques completo del Tracking, con el PLL y DLL realimentándose. A la salida del filtro del PLL se obtiene la fase de la portadora necesaria para generar una nueva portadora, sincronizada con la señal de entrada y, por otra parte, a la salida del filtro del DLL se obtiene la fase del código para que el generador de código aplique el retardo o adelanto a las réplicas del código C/A pertinentes. Al aplicar el bloque de Tracking con la señal de referencia es posible extraer el mensaje de datos de Navegación. Este se recupera a la salida del correlador Prompt de la rama en fase I.

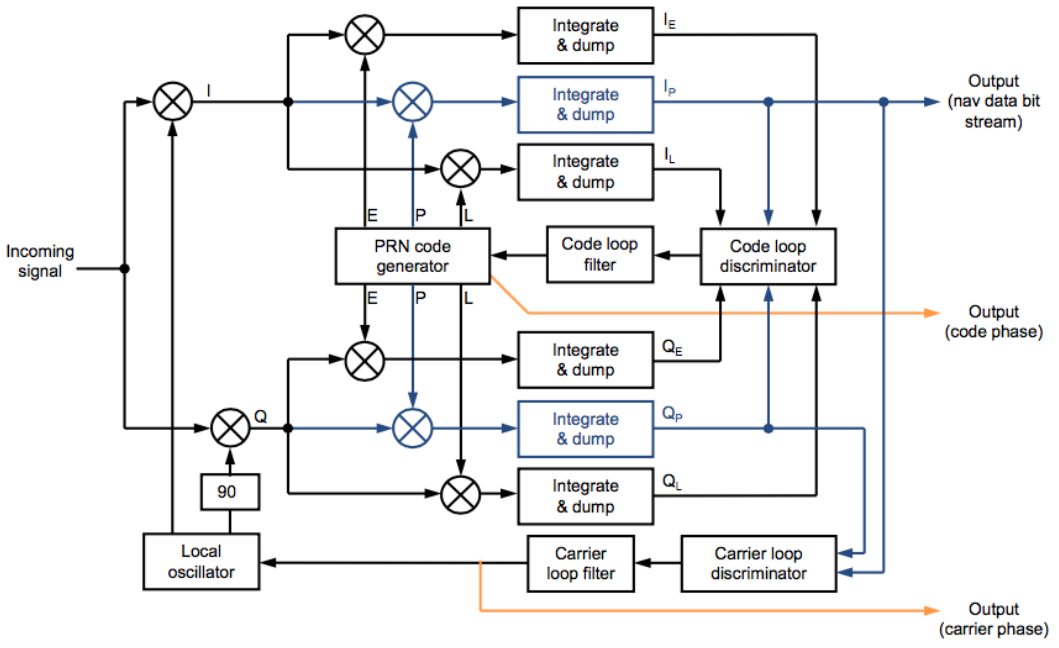

Ilustración 5-16. Diagrama de bloques completo del Tracking

5.1.3.3 Ejemplo de Tracking

En la Ilustración 5-17, se observan los resultados obtenidos tras la Adquisición y Tracking de la señal de referencia GPS durante 20s de pre-procesado.

En ella se observa:

- Diagrama fasorial (a), con un control adecuado de la fase de portadora, corregida correctamente por el bucle de costas ya que no se observa dispersión excesiva de los símbolos.
- El correcto funcionamiento del PLL también se puede observar a partir de las salidas del discriminador (b) y posteriormente el filtro (c), donde inicialmente sufre un transitorio que corrige rápidamente hasta estabilizarse.
- También se aporta la salida del discriminador (d) y su paso por el filtro (e), donde su rápida estabilización es un indicador de su correcto funcionamiento.
- Por otro lado, se encuentran los resultados de correlación (f), debidos a la aplicación del DLL, en él se observa como toda la energía de la señal se mantiene en la componente PROMPT, un indicador de que el retardo que aplica el discriminador del DLL al generador de código es el correcto.
- Por último, dado que el funcionamiento del DLL y PLL es el correcto, es posible obtener el mensaje de datos de navegación sin errores, tal como se observa en (g), perfectamente recuperados durante el pre-procesado de 20 s de la señal de referencia.

La reconstrucción de la señal de referencia [27] se puede modelar como en (3.3) que por conveniencia se replicará nuevamente en (5.17):

$$s_{ref_i}(t) = N_{M_i}\big(t - \tau_i(t)\big) \cdot CA_i\big(t - \tau_i(t)\big) \cdot exp\big(2\pi f_{D_i}(t)t\big) \tag{5.17}$$

Donde:

- τ es la estimación del retardo temporal
- f_D es la estimación de la frecuencia Doppler
- i es el subíndice del satélite

Una vez reconstruida la señal de referencia se pasa a la etapa de procesado coherente con el cálculo de la CAF que permite localizar los blancos adquiridos en el mapa rango-Doppler.

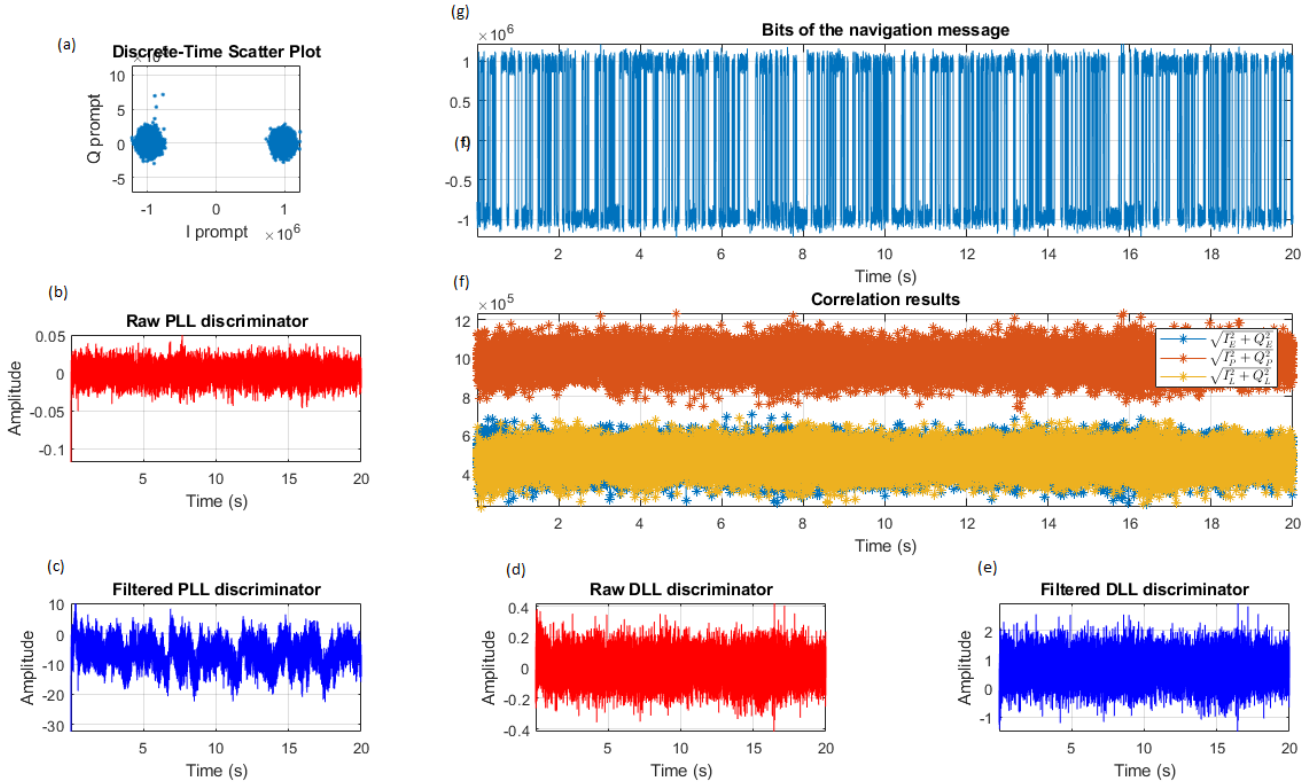

Ilustración 5-17. Resultados del bloque de tracking de la señal GPS durante 20 s para el código PRN 13: (a) Discrete-Time Scatter plot; (b) Raw PLL Discriminator; (c) Filtered PLL Discriminator; (d) Raw DLL Discriminator; (e) Filtered DLL Discriminator; (f) Correlation Results; (g) Bits of the navigation message

6 Optimización de la etapa de pre-procesado

En el capítulo 5, se detallan los algoritmos implementados en la etapa de pre-procesado, en concreto en los bloques de adquisición y tracking. Los tiempos de procesamiento de la solución mixta en el bloque de adquisición y el bloque de tracking son correctos, pero para intentar aproximarse a un sistema de tiempo real es necesario adaptar métodos de aceleración de código.

Estos métodos son: la vectorización de código, la pre-asignación de memoria y la programación paralela.

Por último, es necesario tener en cuenta que implica incluir estos métodos en los algoritmos de adquisición y tracking.

6.1 Métodos de optimización de código en Matlab

6.1.1 Vectorización de código

La vectorización de código en Matlab implica la transformación de matrices en vectores, con el fin de reducir la carga computacional de los algoritmos implementados.

Las razones por la que es preferible vectorizar el código son [37]:

- La primera razón tiene que ver con el aspecto del código. Ya que un código vectorial es mucho más inteligible para el desarrollador. Se transforman las matrices en simples vectores que hacen las expresiones más sencillas.
- La segunda razón es la reducción de errores. Ya que el uso de bucles, o en el peor de los escenarios bucles anidados puede dar lugar a errores de codificación.
- La tercera razón fundamental es la velocidad de funcionamiento de Matlab con vectores, en vez de con bucles.

En el código 'pruebasRendimiento.m' se realiza el ejemplo 'Vectorización de código' [38] para comprobar las mejoras en rendimiento frente al uso de bucles FOR.

Tabla 6-1. Tiempos medios de cálculo obtenidos en 'Vectorización de código'

	Media [s]
bucle FOR	8,6303e-05
Vectorización	4,9871e-05

La Tabla 6-1 se obtiene a partir de la media de tiempo que tarda en ejecutarse cada ejemplo por completo (cada ejemplo se repite durante 100 iteraciones y se calcula su media por iteración). En el primer caso (bucle FOR) se hace uso de un bucle FOR para el cálculo de un seno que requiere de ampliaciones de memoria por cada iteración del bucle, con un total de 61 ampliaciones de memoria. En el segundo caso (Vectorización) se vectoriza el cálculo realizando solo una operación de asignación de memoria.

Se obtiene hasta un 42.21% de aceleración de código en el caso de emplear vectorización. Como consecuencia se tiene que el uso de metodologías en bucle es ineficiente para MATLAB, con lo cual la prioridad de optimización de código en este documento se centrará en la reducción de bucles en todos los casos posibles a partir de la vectorización de código.

6.1.2 Pre-asignación de memoria

La pre-asignación de memoria de variables consiste en asignar memoria, valga la redundancia, de variables previamente a su empleo, en especial cuando son variables de gran tamaño que guardan resultados en bucles. Por el contrario, es muy perjudicial la ampliación de memoria (reserva de memoria) de variables dentro de un bucle.

En el código 'pruebasRendimiento.m' se realiza el ejemplo 'Preasignación de memoria' [39] para comprobar la importancia de la pre-asignación de memoria.

Tabla 6-2. Tiempos de medios de cálculo obtenidos en 'Preasignación de memoria'

	Media [s]
Sin pre-asignación	0,010931
Con pre-asignación y rellenado de ceros	0,00407
Con pre-asignación y rellenado de cero la última posición	0,004268

La Tabla 6-2 se obtiene a partir de la media de tiempo que tarda en ejecutarse cada ejemplo por completo (cada ejemplo se repite durante 100 iteraciones y se calcula su media por iteración). En el primer caso (sin pre-asignación) no se realiza reserva de memoria alguna, obteniéndose los peores resultados. El segundo (con pre-asignación y rellenado de ceros) y tercer caso (con pre-asignación y rellenado de cero la última posición) sí se realizan reservas de memoria previas, con relleno de ceros en todas las posiciones de la variable y con relleno de cero la última posición de la variable respectivamente.

La diferencia entre el segundo y el tercer método es insignificante, y ambos métodos incluyen pre-asignación de memoria. Por lo que la comparación se realiza entre algoritmos sin pre-asignación y con pre-asignación, obteniéndose hasta un 62.77% de aceleración de código con la reserva de memoria de variables previa a su empleo. Si se incrementa el tamaño de la variable durante su empleo se comprueba que el tiempo de cálculo es superior, por tanto, es otra característica para tener en cuenta en los algoritmos.

6.1.3 Programación paralela

La programación paralela consiste en realizar cálculos de forma simultánea. El inconveniente se encuentra en traducir el algoritmo en serie en formato paralelo, siendo en algunos casos imposible, como cuando se requiere de un cálculo para realizar otro posterior. Una de las alternativas que permite Matlab es la programación paralela a través de los núcleos de la CPU y el uso de la GPU para acelerar el código; que junto con la vectorización de código y la pre-asignación de memoria se obtienen muy buenos resultados en cuanto a tiempo de procesamiento.

Las vertientes de programación paralela tienen las siguientes características [37]:

- **Uso de Bucle PARFOR**. Este método permite la aceleración de los algoritmos internos de los bucles FOR que se dispongan dentro del código, pero con ciertas restricciones. La primera de ellas es que cada iteración del bucle tiene que ser independiente de ninguna otra, es decir, se tiene que conseguir paralelizar el código antes de hacer uso de esta función. Otra de las restricciones, es que se debe evitar realizar operaciones complejas o anidar bucles, ya que de ser así se provocaría el efecto contrario al que se desea, generándose grandes tiempos de procesado. Esto es debido a que MATLAB envía cada iteración a un worker, que en programación paralela es lo más parecido a un hilo de trabajo, y el coste de transferencia de datos es elevado. Habrá tantos workers como núcleos disponga el equipo.

  En el código 'pruebasRendimiento.m' se realiza un ejemplo 'Comparación entre bucles FOR y PARFOR' [37] que compara el tiempo de cálculo que toma un bucle FOR frente a un bucle PARFOR.

Tabla 6-3. Tiempos medios obtenidos en 'Comparación entre bucles FOR y PARFOR'

	Media [s]
Con bucle FOR	13,966
Con bucle PARFOR	4,5011

En la Tabla 6-3, se obtiene a partir de la media de tiempo que tarda en ejecutarse cada ejemplo por completo. Cada ejemplo realiza un conjunto de instrucciones sobre una matriz de orden 50. En el primer caso (con bucle FOR) la estructura de programación es en serie, mientras que el segundo caso (con bucle PARFOR) la estructura de programación es en paralelo.

Como se puede comprobar el bucle PARFOR se toma un 65,88% menos de tiempo de cálculo, es decir, la programación paralela consigue mejor resultado que la programación en serie. Para incrementar el rendimiento del bucle PARFOR, los cálculos que se aplican a la matriz son vectorizados, es decir, en una misma instrucción se realizan varias funciones, evitándose tener que asignar memoria a cada resultado de cada operación por cada iteración del bucle. De lo contrario, la eficiencia del bucle PARFOR decaería, llegando a tener peor rendimiento que el bucle FOR.

- **Uso de la GPU**. La segunda alternativa de programación paralela que se ofrece es el uso de funciones optimizadas para la GPU, como la FFT, IFFT entre las más destacadas. Estas funciones se ejecutan directamente en la GPU, incrementándose la velocidad de cálculo. Se obtiene buenos resultados siempre que se tenga en cuenta las siguientes condiciones: que el uso de la GPU se maximice al uso de todos los núcleos de esta (para mejorar el rendimiento de la GPU al igual que el uso del PARFOR, los elementos que se emplean deben ser vectorizados) y, por otra parte, que el tiempo requerido en el cálculo sea más grande que el de transferencia de archivos entre CPU y GPU.

  En el código 'PruebasRendimiento.m' se realiza un ejemplo, 'Ejemplo de cálculo con GPU' [37], de filtrado sobre los dispositivos disponibles en Matlab, como lo son: la CPU y la GPU.

Tabla 6-4. Tiempos de cálculo obtenidos en 'Ejemplo de cálculo con GPU'

	Tiempo [s]
Cálculo sobre la CPU	0,51012
Transferencia y cálculo sobre la GPU	0,24515
Cálculo sobre la GPU	0,067087

En la Tabla 6-4, se obtiene a partir del tiempo que tarda en ejecutarse cada ejemplo por completo. Cada ejemplo realiza un filtrado sobre una matriz de orden 5000. En el primer caso (cálculo sobre la CPU) se genera una matriz cuadrada y se filtra, pero el tiempo que se valora sólo es el de cálculo sobre la CPU. En el segundo caso (transferencia y cálculo sobre la GPU) se mide el tiempo de cálculo incluyendo la transferencia de la matriz a la GPU, con la función gpuArray. En el tercer caso (cálculo sobre la GPU) como en el primero solo se valora el tiempo de cálculo, pero en este caso sobre la GPU.

En los resultados se observa como la transferencia y el cálculo sobre la GPU es un método muy superior que el empleo de la CPU. Se consigue reducir un 51,94% de tiempo de procesamiento. Y si a tiempo de cálculo se refiere la GPU emplea un 86,85% menos de tiempo que la CPU. Por tanto, el rendimiento de la GPU sobre funciones optimizadas es muy superior al rendimiento de la CPU sobre las mismas.

Con lo cual, maximizar el rendimiento en el código implementado en Matlab está asegurado. Pero hay que respetar las restricciones de cada método.

En este documento se ha buscado la mayor eficiencia posible a partir de reservas de memoria previas a los bucles, vectorización de la mayor parte de código y uso de funciones optimizadas en la GPU. Con ello se ha logrado elevar el rendimiento tanto del algoritmo de adquisición como el de tracking.

6.2 Algoritmos optimizados en la etapa de pre-procesado

En este apartado se detalla los distintos métodos de optimización aplicados sobre los algoritmos propuestos, tanto para el bloque de adquisición como para el bloque de tracking.

6.2.1 Optimización en el bloque de Adquisición

En el bloque de adquisición se tiene que buscar tres variables: el código PRN, el retardo de fase de código y la frecuencia Doppler de la portadora. Por otra parte, hay que tener en cuenta que la fase de código y la frecuencia Doppler se realiza por cada código PRN establecido, con lo cual es fácil pensar que realizar un tipo de algoritmo con paralelización de código es la forma más factible de conseguir el objetivo.

En los siguientes apartados, se observará como la paralelización de código no es el algoritmo adecuado para el bloque de adquisición. Con lo cual, es necesario encontrar un algoritmo adecuado para el bloque de adquisición.

En este documento se realizan tres propuestas de algoritmos optimizados para el bloque de adquisición.

6.2.1.1 Algoritmo de adquisición con implementación de GPU Arrays y vectorización de código

Este algoritmo se implementa sobre el código 'acquisition_def.m' (Pseudocódigo adq_1) que se le denominará como adq_1, y está basado en el código inicial de referencia 'acquisition.m'(Pseudocódigo adq_0), que se le denominará *adq_0*.

En la Ilustración 6-1, se observa una captura del algoritmo de adquisición gruesa que se implementa en el código *adq_0*. El primer bucle 'bucle de control PRN' controla el código PRN que se inserta al algoritmo de adquisición gruesa, es decir, permite la identificación del satélite en el caso de que la correlación resultante supere en nivel de Threshold establecido. En el segundo bucle 'bucle de control de frecuencias' controla la frecuencia de portadora empleada, es decir, en él se genera el número de portadoras necesarias para realizar el barrido de frecuencia de IF±10 KHz en pasos de 500 Hz.

Hay que destacar que *adq_0* realiza la adquisición gruesa (apartado 5.1.1.2) a partir de dos bucles anidados, tal como se observa en la Ilustración 6-1. El rendimiento de Matlab decae fuertemente debido al empleo de bucles anidados.

En la Ilustración 6-2, se observa una captura del algoritmo de adquisición gruesa que se implementa en el código adq_1. El punto clave de este algoritmo es la creación de matrices de 2 dimensiones que permiten el control de tiempo y de frecuencia, y la vectorización de las funciones del algoritmo, lo que permite reducir el número de bucles de control y una mejor gestión de cálculo. Cabe destacar que se ha declarado las variables sobre la GPU como gpuArray, ya que las funciones que se emplean se encuentran optimizadas en el software Matlab, obteniéndose mayor velocidad de cálculo.

```
for PRN = settings.acqSatelliteList   bucle de control de PRN
%%%%%%%%%%

%% Correla las señales ====================================================
    %--- Lleva a cabo la DFT del código C/A -------------------------------
    caCodeFreqDom = conj(fft(caCodesTable(PRN, :)));

    %--- Realiza la correlación de toda la banda de frecuencias
    %    (para todos los bins de frecuencia)
    for frqBinIndex = 1:numberOfFrqBins      bucle de control de frecuencias

        %--- Genera el grid de frecuencias de portadora (pasos de 0.5kHz) -
        frqBins(frqBinIndex) = settings.IF - ...
                               (settings.acqSearchBand/2) * 1000 + ...
                               0.5e3 * (frqBinIndex - 1);

        %--- Genera el seno y el coseno locales -------------------------
        sinCarr = sin(frqBins(frqBinIndex) * phasePoints);
        cosCarr = cos(frqBins(frqBinIndex) * phasePoints);
```

Ilustración 6-1. Captura de bucles de control de la adquisición gruesa del código *adq0*

```
%% Función de búsqueda de máximos
for PRN = settings.acqSatelliteList    bucle de control de PRN
    % Réplica de matriz de códigos para cada PRN y traslado a dominio
    % frecuencial
    caCodesTableAux = caCodesTable(PRN,:); % Genera código durante 1 ms

    % Paso el código a la correlación directamente conjugado
    caCodeFreqDomAux = conj(fft(repmat(caCodesTableAux,...
        [length(freqBins),1]),[],2));

    % Correlación en el dominio de la frecuencia
    convCodeIQ1 = IQfreqDom1 .* caCodeFreqDomAux;
    convCodeIQ2 = IQfreqDom2 .* caCodeFreqDomAux;

    % Retorno a dominio temporal
    acqRes1 = abs(ifft(convCodeIQ1,[],2)) .^ 2;
    acqRes2 = abs(ifft(convCodeIQ2,[],2)) .^ 2;

    %--- Comprueba qué msg tiene la máxima potencia y lo guarda
    % (combina los msgs primero y segundo pero corrije posibles problemas
    % en los bits de datos)
    if (max(max(acqRes1)) > max(max(acqRes2)))
        results = acqRes1;
    else
        results = acqRes2;
    end
```

Ilustración 6-2. Captura de bucle de control de la adquisición gruesa del código adq1

Por último, la adquisición que se emplea en este sistema de radar pasivo (apartado 5.1.1.4) es una mezcla entre adquisición gruesa y fina, pero la adquisición fina sólo se activa en el caso de identificar un código PRN por lo que el número de combinaciones se reduce a una. Es por esto por lo que la optimización que se realiza sobre la adquisición fina de adq1 (vectorización y empleo de funciones optimizadas sobre la GPU) no destaca grandes mejoras en cuanto a velocidad computacional se refiere.

6.2.1.2 Algoritmo de adquisición con implementación de matrices tridimensionales

Este algoritmo se implementa sobre el código 'acquisitionTriple.m' (Pseudocódigo adq$_2$) que se le denominará adq$_2$. En este caso lo que se buscar es la supresión de todos los bucles de control mediante el empleo de matrices tridimensionales. Cada eje de la matriz controla un parámetro de búsqueda del algoritmo de adquisición, como el eje de códigos PRN, de frecuencias de búsqueda y temporal, que identifican al satélite, la frecuencia Doppler de la portadora y al retardo de fase del código respectivamente.

```
%% Adquisición Gruesa - Función de búsqueda de máximos ====================
                          supresión de bucles
% Se buscan resultados para todos los bins de frecuencia y desplazamientos
% de código (para un satélite)

% Réplica de matriz de códigos para cada PRN y traslado a dominio
% frecuencial
% Generar 1ms de código para cada PRN
% Paso el código a la correlación directamente conjugado
% *Tridimensional
caCodeFreqDomAux = conj(fft(repmat(caCodesTable2,...
    [length(freqBins),1]),[],2));
                                                  matrices tridimensionales
% Retorno a dominio temporal
% *Tridimensional
acqRes1 = abs(ifft(IQfreqDom1 .* caCodeFreqDomAux,[],2)) .^ 2;
acqRes2 = abs(ifft(IQfreqDom2 .* caCodeFreqDomAux,[],2)) .^ 2;

%--- Comprueba qué msg tiene la máxima potencia y lo guarda
```

Ilustración 6-3. Captura de la supresión de bucles de control del código adq$_2$

En la Ilustración 6-3, se observa la supresión de bucles de control del código adq$_2$ gracias al empleo de matrices tridimensionales. Para optimizar el cálculo, además, se ha empleado vectorización sobre las funciones.

6.2.1.3 Algoritmos de adquisición con implementación de paralelización de código

Por último, el algoritmo de paralelización de código que está implementado en el código 'acquisition_parfor.m' (Pseudocódigo adq$_3$) que se denominará adq$_3$. Como en los algoritmos adq$_1$ y adq$_2$, se busca suprimir los bucles de control, en este caso de tiempo y de frecuencia mediante el empleo de paralelización de código sobre CPU aplicando un bucle PARFOR que gestionará la identificación del satélite.

```
parfor PRN = settings.acqSatelliteList
[peakMetric_aux(PRN), carrFreq_aux(PRN), codePhase_aux(PRN), ...
    presencia_aux(PRN)] = acquisition_parfor(signal0DC, IQfreqDom1,...
    IQfreqDom2, freqBins, settings, ...
    samplesPerCode, caCodesTable(PRN,:), PRN, samplesPerCodeChip, ...
    fftFreqBins, fftNumPts, msAcqP);
end
timeAcquisitionParfor(i) = toc;
```

Ilustración 6-4. Captura del bucle de paralelización PARFOR que gestiona la adquisición del código adq$_3$

En la Ilustración 6-4, se observa el nuevo bucle de PARFOR del algoritmo adq3, donde se lanzan un hilo de trabajo en paralelo por cada código PRN de búsqueda. Estos hilos se ejecutan sobre la CPU con lo cual se lanzan tantos hilos en paralelo como procesadores tenga la CPU. Este método requiere la independencia de las operaciones por hilo de trabajo, por lo que es necesario pasar a la función todas las variables que sean necesarias.

6.2.2 Optimización en el bloque de Tracking

En el caso del bloque de tracking, al tratarse de seguimiento de la señal GPS, es necesario hacer lecturas continuas de bloques de datos para detectar las variaciones temporales de los parámetros de frecuencia Doppler y retardo de fase. Luego como existe una dependencia de variables dentro del algoritmo, es decir, es necesario conocer el Doppler y el retardo de fase de la iteración previa para corregirlo en la siguiente iteración, no es viable emplear algoritmos de paralelización de código. Tampoco es posible el empleo de matrices tridimensionales ya que además de no eliminar el bucle de seguimiento, incrementaría la carga computacional al emplear variables de gran tamaño, reduciendo el rendimiento del código inicial de referencia 'tracking.m' (Pseudocódigo trck0) que se denominará *trck0*.

Por otra parte, cada lectura de datos que se realiza en el bloque de tracking corresponde a 1ms de señal de referencia, por lo que las iteraciones del bucle corresponden con 1ms de procesado de señal con un total de 20 s de señal disponible, con lo cual se requieren 20000 iteraciones por canal adquirido. Por canal adquirido se entiende al canal activo cuando el bloque de adquisición ha identificado un código PRN y adquirido los parámetros de Doppler y retardo de fase necesarios para el tracking.

Debido a estas causas es necesario encontrar un algoritmo que sea compatible con el bucle de tracking y que además reduzca el tiempo computacional que requiere el bucle para realizar 20000 iteraciones por canal activo.

En este documento se presenta una solución optimizada.

6.2.2.1 Algoritmo de Tracking con implementación de vectorización de código y cálculo sobre GPU

Este algoritmo se implementa sobre el código 'tracking6.m' (Pseudocódigo trck1) que se denominará *trck1*. Este algoritmo presenta distintas técnicas de optimización como la precarga de datos a función, que reduce cálculos innecesarios ya que se han realizado en bloques previos, la pre-asignación de memoria, técnica fundamental para almacenar parámetros fundamentales como el Doppler o el retardo de fase (necesarios para la posterior reconstrucción de la señal de referencia), y el empleo de funciones optimizadas sobre la GPU, que permite acelerar el código. Otro punto importante sobre este algoritmo es la generación de códigos PRN (Early, Prompt y Late) para los correladores mediante desplazamientos circulares, además de la vectorización del código en el mayor número de operaciones posibles.

7 Resultados

En el capítulo 5, se realiza un análisis de los algoritmos implementados en la etapa de pre-procesado de la señal de referencia.

En el capítulo 6, se describe los métodos que se emplean en el desarrollo de este documento para la optimización de la etapa de pre-procesado. Estos métodos son la vectorización de código, la pre-asignación de memoria y la programación paralela.

Tras la implementación de los métodos de optimización en los bloques de Adquisición y Tracking se han obtenido los siguientes resultados.

7.1 Estudio de implementación de algoritmos con señales filtradas

La adquisición de las señales de GPS de referencia y de vigilancia se realiza a 25 MHz para mantener las características de demostrador y mantener el tamaño de la celda del radar pasivo. Uno de los inconvenientes de la adquisición es la presencia de ruido en las señales, con lo cual, se filtrarán digitalmente hasta los 2 MHz (el ancho de banda de la señal GPS C/A es de 2 MHz) para reducir el ruido y comprobar el impacto que genera sobre los bloques de adquisición y de tracking.

Para el filtrado digital de la señal se ha diseñado un filtro digital paso bajo (LPF)de orden 600. El inconveniente de este tipo de filtros es que introduce un retardo de grupo que hay que tener en cuenta, ya que esto generaría errores en la adquisición de la señal de referencia y por consiguiente errores en el tracking y la extracción del mensaje de datos de Navegación. La ventaja es que el retardo de grupo es constante a todas las frecuencias por lo que es fácilmente compensable [40]. Otra ventaja es que este tipo de filtro no introduce ningún retardo de fase.

Las frecuencias de corte de los filtros digitales son: 2 MHz y 4 MHz.

Respuesta en frecuencia y en fase del LPF con frecuencia de corte a 2 MHz

La Ilustración 7-1, muestra la respuesta en frecuencia medida en dB de un LPF digital de 2 MHz de frecuencia de corte.

La Ilustración 7-2, muestra la respuesta en fase medida en radianes de un LPF digital de 2 MHz de frecuencia de corte.

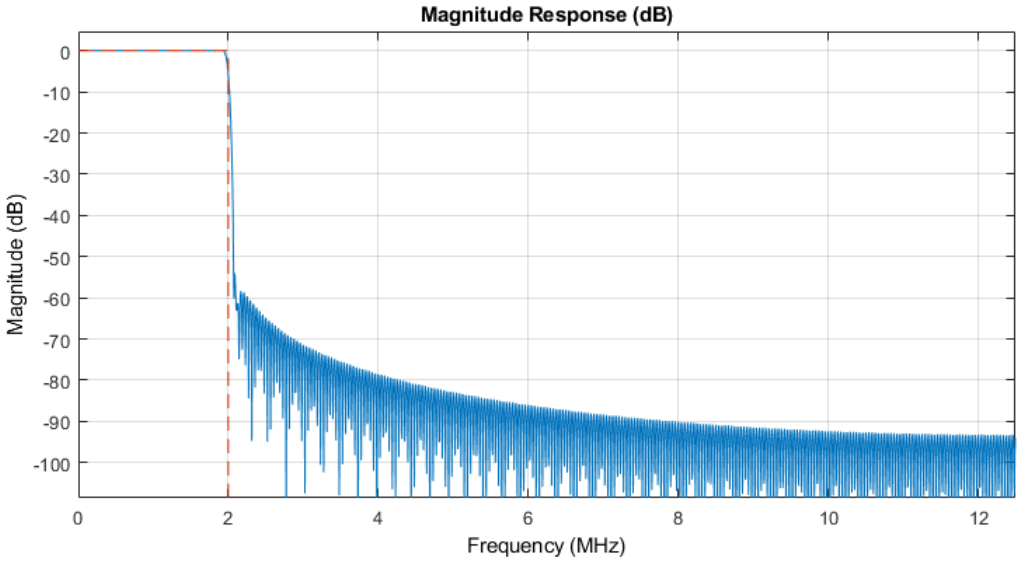

Ilustración 7-1. Respuesta en frecuencia [dB] para LPF 2 MHz

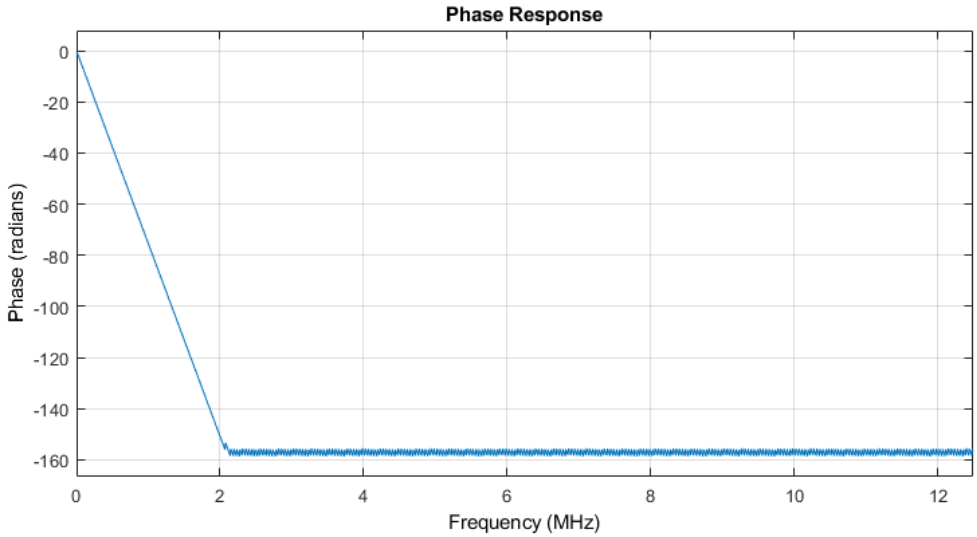

Ilustración 7-2. Respuesta de fase [rad] para LPF 2 MHz

Respuesta en frecuencia y en fase del LPF con frecuencia de corte a 4 MHz

La Ilustración 7-3, muestra la respuesta en frecuencia medida en dB de un LPF digital de 4 MHz de frecuencia de corte.

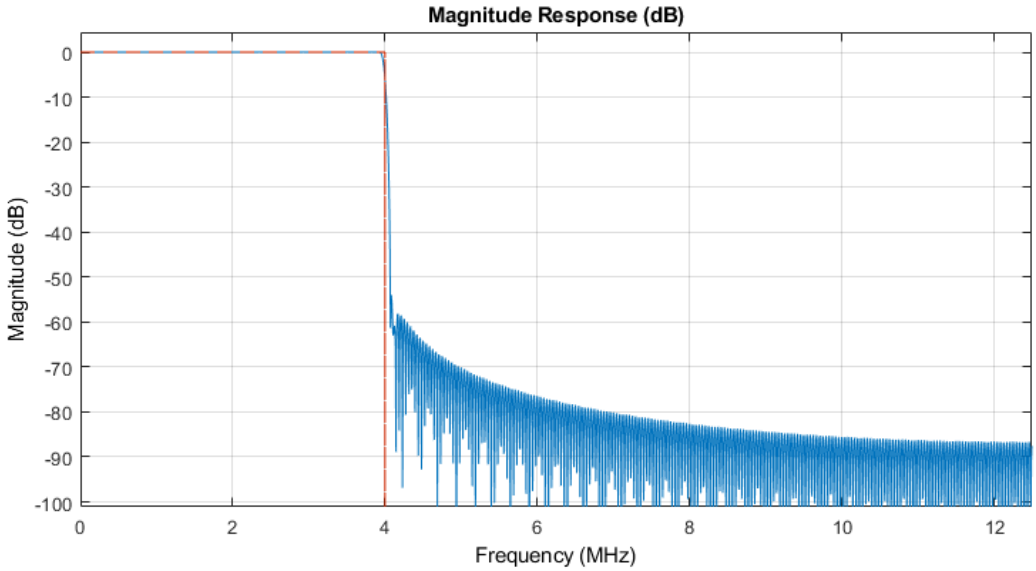

Ilustración 7-3. Respuesta en frecuencia [dB] para LPF 4 MHz

La Ilustración 7-4, muestra la respuesta en fase medida en radianes de un LPF digital de 4 MHz de frecuencia de corte.

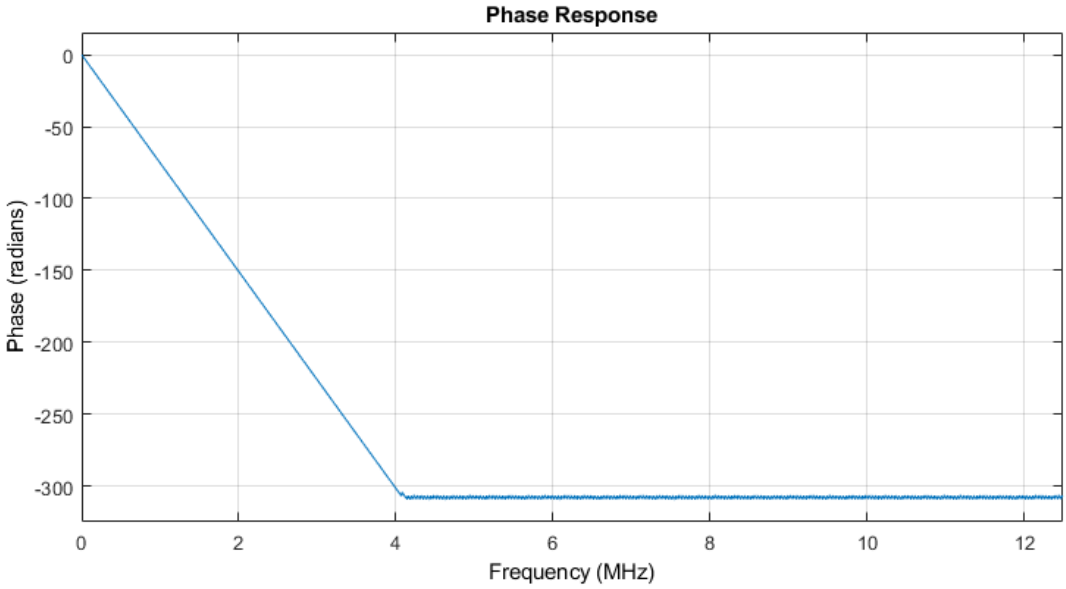

Ilustración 7-4. Respuesta de fase [rad] para LPF 4 MHz

Retardo de grupo constante y retardo de fase nulo de LPF

La Ilustración 7-5, muestra el retardo de grupo constante de un LPF digital medido en radianes.

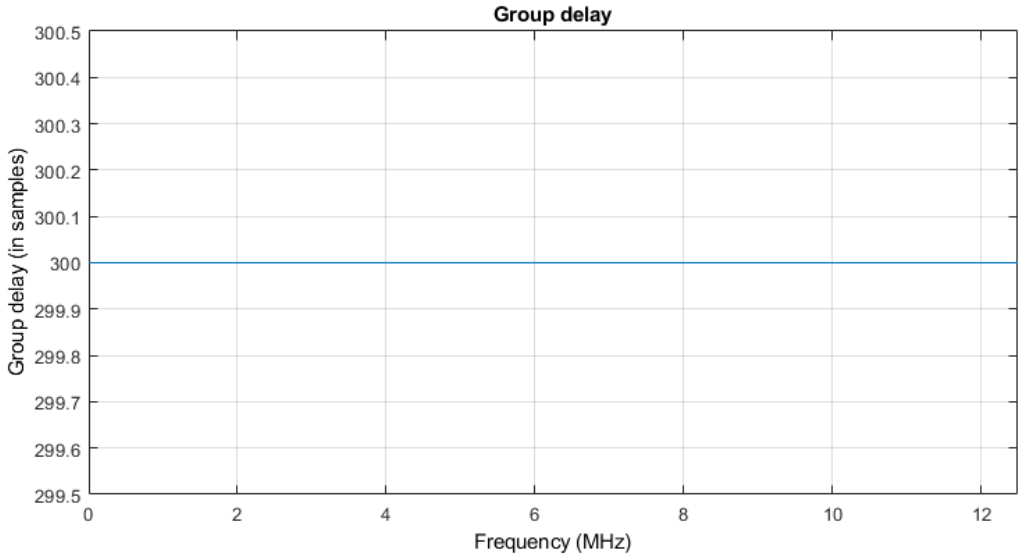

Ilustración 7-5. Retardo de grupo constante = 300 samples

La Ilustración 7-6, muestra el retardo de grupo nulo de un LPF digital medido en radianes/Hz.

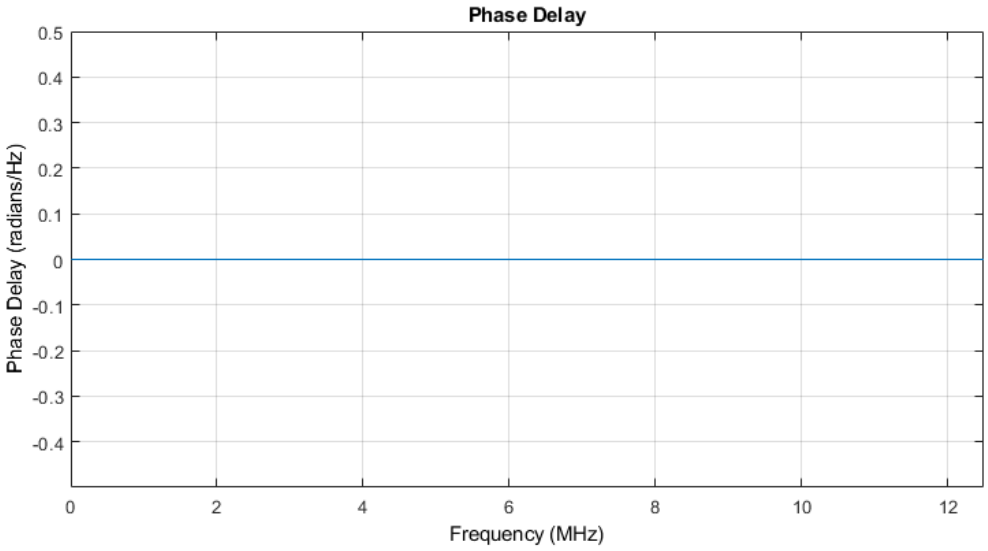

Ilustración 7-6. Retardo de fase nulo

7.1.1 Bloque de Adquisición

Dado que el bloque de adquisición se repite para todos los códigos PRN, en este apartado se mostrarán los resultados de uno de los códigos PRN que se ha identificado en la señal de referencia (las capturas de los resultados de todos los códigos PRN identificados se encontrarán en el Anexo II).

Los códigos identificados de la señal de referencia en la adquisición gruesa son el PRN 5, PRN 13, PRN 15 y PRN 23. Se muestran las correlaciones del código PRN 5 correspondientes a la adquisición gruesa para la señal sin filtrar (Ilustración 7-7), la señal filtrada a 4 MHz (Ilustración 7-8) y la señal filtrada a 2 MHz (Ilustración 7-9).

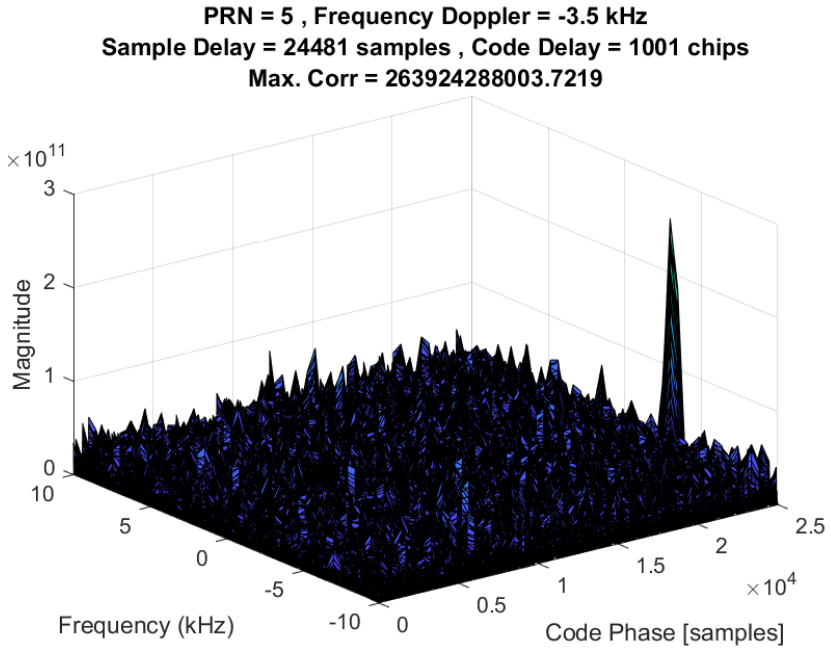

Ilustración 7-7. Adquisición gruesa. Señal sin filtrar: PRN 5; f_d=-3.5 kHz; τ=1001 chips

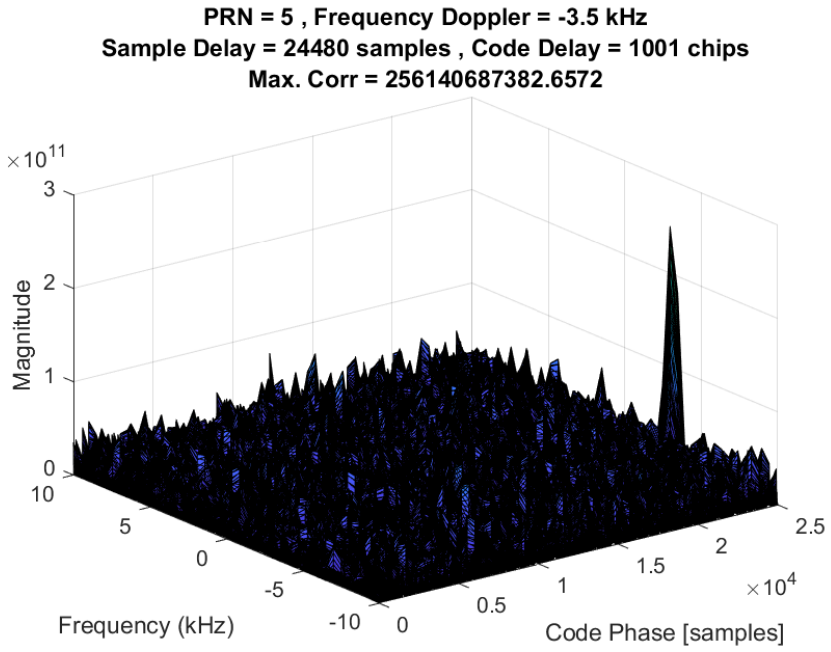

Ilustración 7-8. Adquisición gruesa. Señal filtrada 4 MHz: PRN 5; f_d=-3.5 kHz; τ=1001 chips

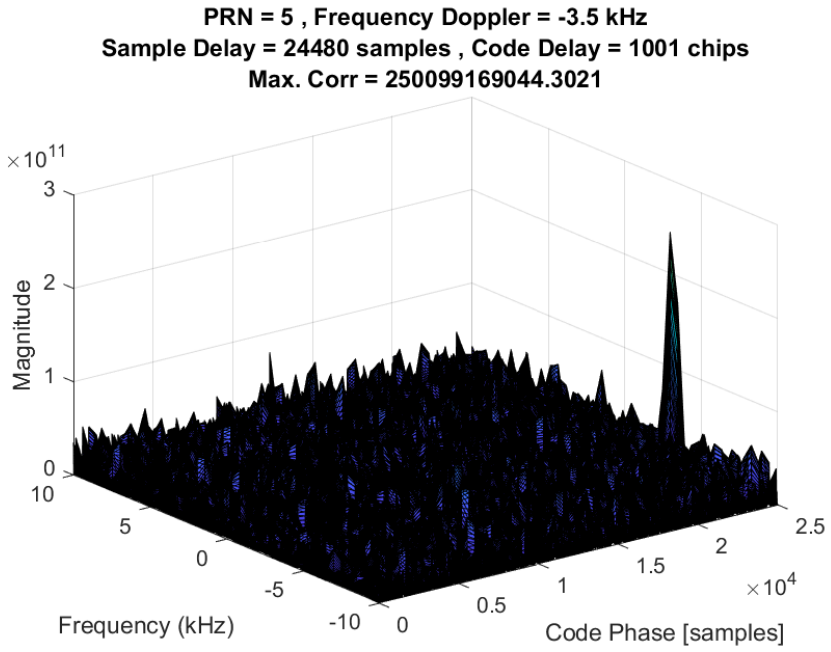

Ilustración 7-9. Adquisición gruesa. Señal filtrada 2 MHz: PRN 5; f_d=-3.5 kHz; τ=1001 chips

De la matriz de correlación obtenida se puede extraer una estimación de los parámetros Doppler (eje Frequency) y retardo de fase (eje Code Phase).

La siguiente Tabla 7-1, resume los resultados de la adquisición gruesa para las tres secuencias disponibles y los códigos PRN identificados.

Tabla 7-1. Resumen de resultados obtenidos en la adquisición gruesa

PRN	**5**	**13**	**15**	**24**
f_d [kHz]	-3,5	-1	1	2
τ [chips]	1001	480	376	74
Max. Corr [·10_{11}]	2,6392	11,6412	10,6060	7,0054
Max. $Corr_4$ [·10_{11}]	2,5614	11,5289	10,3808	6,9030
Max. $Corr_2$ [·10_{11}]	2,5010	11,1669	9,9719	6,5161

Donde PRN son los códigos que se han identificado tras el método de adquisición gruesa, f_d es el Doppler, τ es el retardo de fase y *Max. $Corr_i$* es la máxima correlación conseguida por cada secuencia.

Como se puede observar en la tabla, el algoritmo de adquisición gruesa ha identificado 4 códigos PRN correctamente, detectándose la misma frecuencia Doppler y el mismo retardo de código empleando distintas secuencias de señal de referencia. Donde se observan diferencias es en la correlación máxima del algoritmo, obteniéndose menor correlación en los casos de señal de referencia filtrada. En cualquier caso, las variaciones que se producen no son significativas como para dictaminar que afecta al comportamiento del algoritmo. En los tres casos el nivel de correlación máxima es el adecuado para superar el Threshold establecido.

Una vez identificados los códigos PRN, se toma el retardo de fase τ calculado en la adquisición gruesa y se pasa al algoritmo de adquisición fina, donde se busca obtener el parámetro Doppler de la señal con mayor precisión. Se presentan las densidades espectrales de potencia del código PRN 5, conseguidas en la adquisición fina para la señal sin filtrar (Ilustración 7-10), la señal filtrada a 4 MHz (Ilustración 7-11) y la señal filtrada a 2 MHz (Ilustración 7-12).

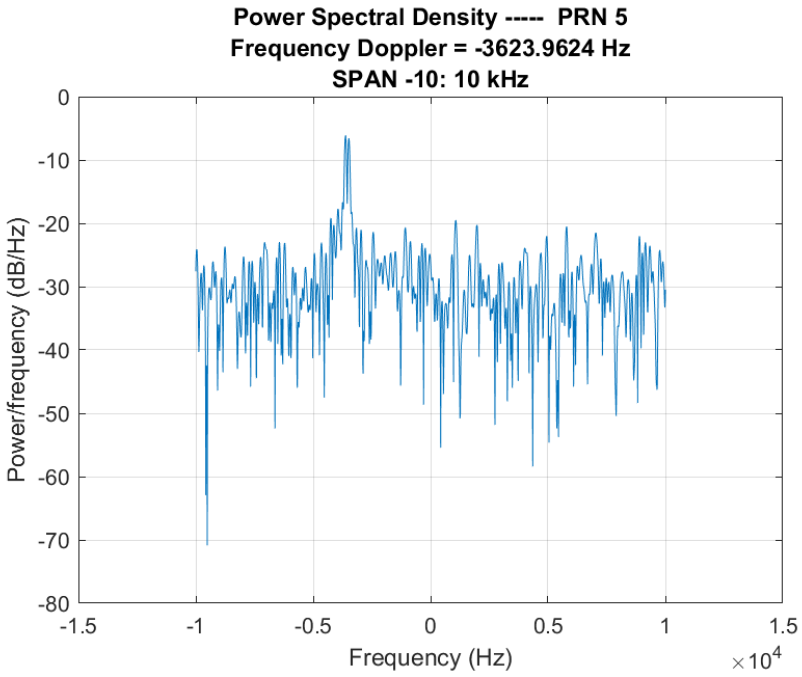

Ilustración 7-10. Adquisición fina. Señal sin filtrar: PRN 5; fd=-3623,9624 Hz

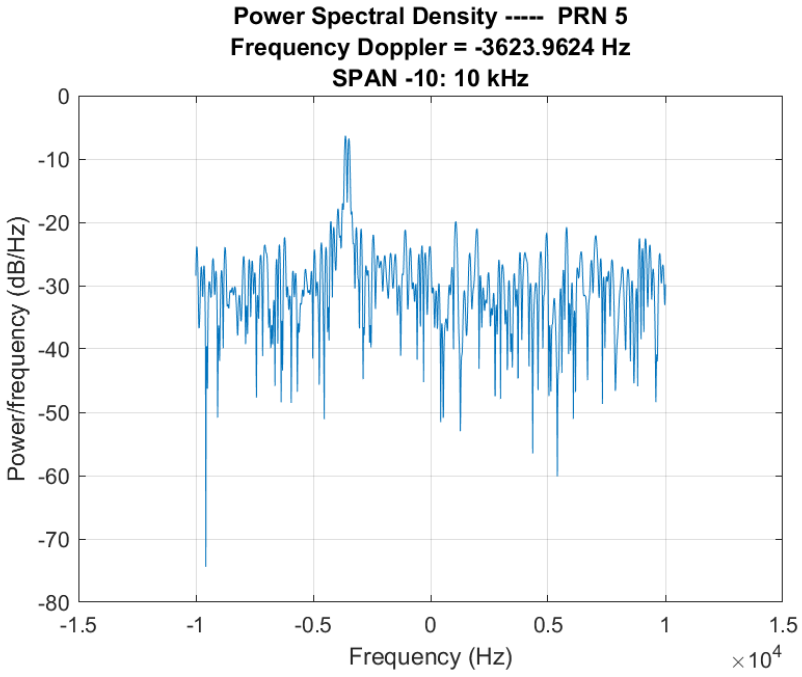

Ilustración 7-11. Adquisición fina. Señal filtrada 4MHz: PRN 5; f_d=-3623,9624 Hz

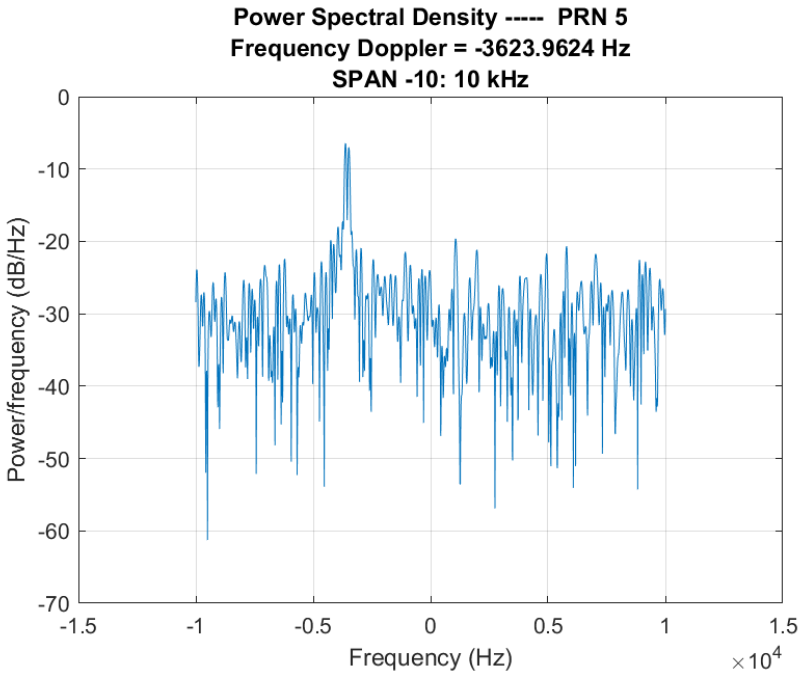

Ilustración 7-12. Adquisición fina. Señal filtrada 2 MHz: PRN; f_d=-3623,9624 Hz

Al emplear este método se obtiene el parámetro Doppler en el máximo de potencia. Para mejor detalle del máximo se ha decidido hacer un SPAN de IF±10 KHz sobre el espectro, identificándose que la portadora asociada al código PRN 5 tiene su máximo en los -3623,9624 Hz.

En la Tabla 7-2, se resumen los resultados de la adquisición fina, obteniéndose la misma frecuencia Doppler por código PRN asociado tras emplear las tres secuencias.

Tabla 7-2. Resumen de resultados obtenidos en la adquisición fina

PRN	**5**	**13**	**15**	**24**
f_d [Hz]	-3623,9624	-1084,8045	786,7813	2026,5579

Si se comparan los resultados de la adquisición gruesa (Tabla 7-1) frente a la adquisición fina (Tabla 7-2), se observa el incremento de sensibilidad de un método frente a otro. El algoritmo de adquisición gruesa presenta una definición entorno a los 500 Hz frente a los 12,5 Hz del algoritmo de adquisición fina. Pero en ninguno de los casos el incremento de sensibilidad en frecuencia es debido al filtrado de señal, ya que en los tres casos se ha obtenido la misma frecuencia Doppler.

La correcta adquisición de los parámetros de retardo de fase (adquisición gruesa) y Doppler (adquisición fina) permite inyectar al bloque de tracking datos fiables que permiten el correcto funcionamiento del bucle de seguimiento, tanto del DLL y PLL manteniéndose la sincronía con la señal de referencia.

7.1.2 Bloque de Tracking

El bloque de tracking se implementa durante 20 segundos de la señal de referencia en bucles de 1 milisegundo. La finalidad del bloque de tracking es mantener el retardo de fase del código y la frecuencia Doppler de la portadora en sincronía con la señal de referencia, y de esta forma, recuperar el mensaje de datos de navegación para posteriormente reconstruir la señal de referencia.

En el estudio del bloque de tracking se desea observar si existen diferencias tras emplear secuencias filtradas, recordando siempre la finalidad de este bloque que es recuperar la señal de referencia. Los parámetros con los que el bloque de tracking va a iniciar su ciclo son los que se obtienen en el bloque de adquisición y se resumen en la Tabla 7-3:

Tabla 7-3. Resumen de resultados obtenidos por el bloque de adquisición

PRN	**5**	**13**	**15**	**24**
f_d [Hz]	-3623,9624	-1084,8045	786,7813	2026,5579
τ [chips]	1001	480	376	74

Las capturas tras la implementación del algoritmo de tracking se pueden encontrar en el Anexo II. A modo comparativo en este apartado sólo se incluirá las capturas asociadas al código PRN 5 de la señal sin filtrar (Ilustración 7-13), la señal filtrada a 4 MHz (Ilustración 7-14) y la señal filtrada a 2 MHz (Ilustración 7-15).

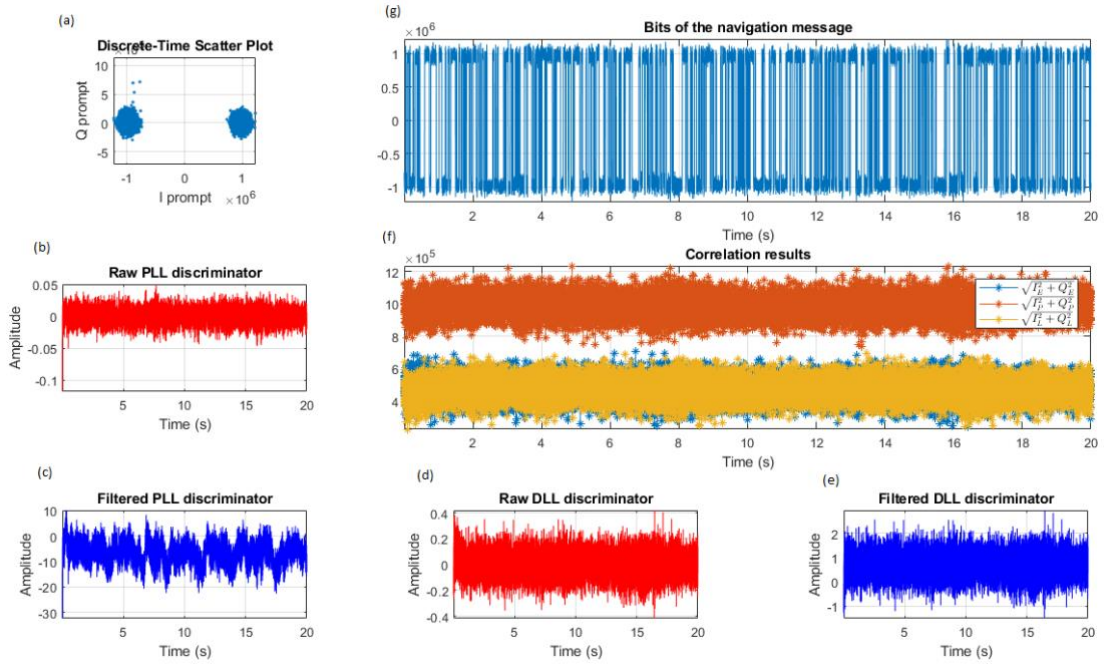

Ilustración 7-13. Tracking 20 s. Señal sin filtrar. PRN 5. (a) Discrete-Time Scatter plot; (b) Raw PLL Discriminator; (c) Filtered PLL Discriminator; (d) Raw DLL Discriminator; (e) Filtered DLL Discriminator; (f) Correlation Results; (g) Bits of the navigation message

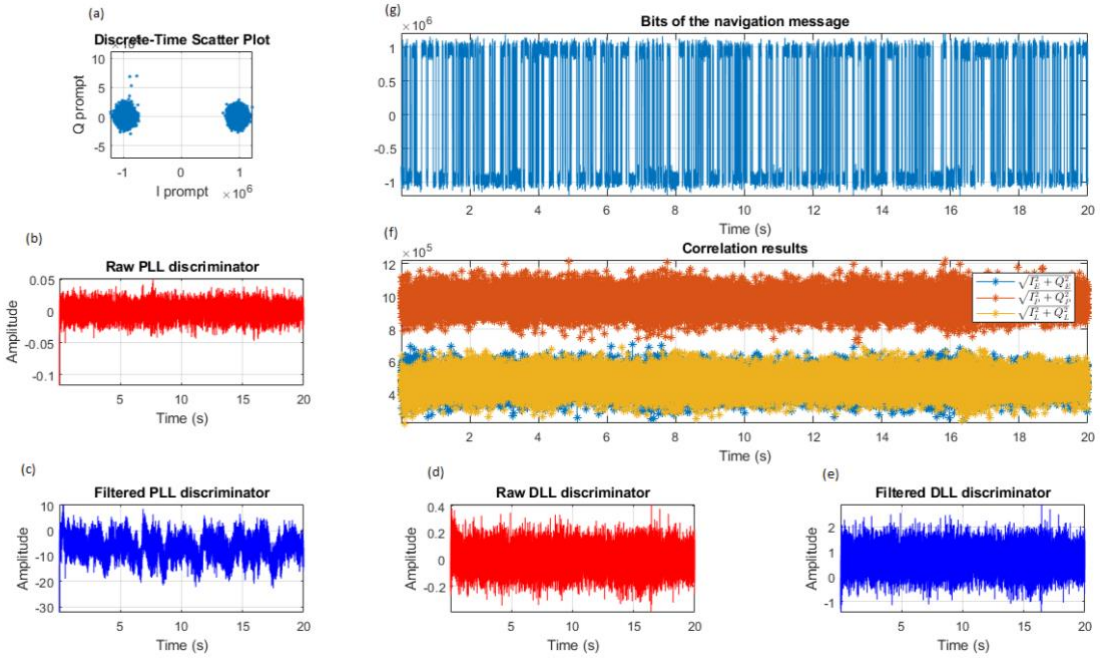

Ilustración 7-14. Tracking 20 s. Señal filtrada 4 MHz. PRN 5. (a) Discrete-Time Scatter plot; (b) Raw PLL Discriminator; (c) Filtered PLL Discriminator; (d) Raw DLL Discriminator; (e) Filtered DLL Discriminator; (f) Correlation Results; (g) Bits of the navigation message

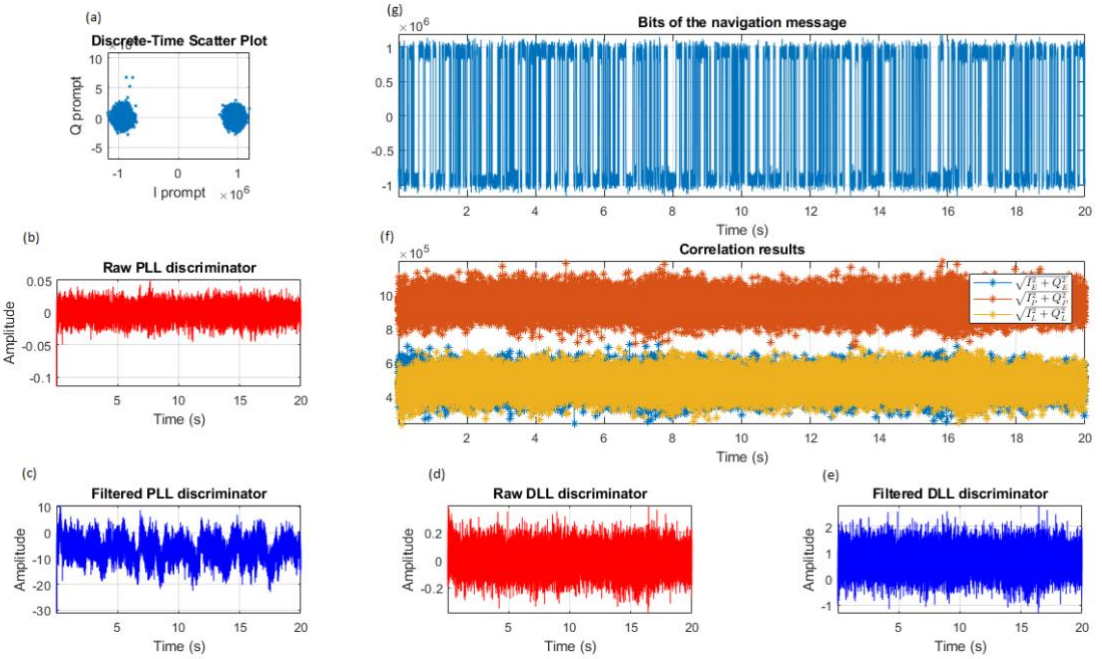

Ilustración 7-15. Tracking 20 s. Señal filtrada 2 MHz. PRN 5. (a) Discrete-Time Scatter plot; (b) Raw PLL Discriminator; (c) Filtered PLL Discriminator; (d) Raw DLL Discriminator; (e) Filtered DLL Discriminator; (f) Correlation Results; (g) Bits of the navigation message

Como se puede observar en estas ilustraciones, el mensaje de navegación (g) ha sido recuperado correctamente durante los 20 segundos, con las tres secuencias de señal de referencia. Por lo que es posible reconstruir la señal de referencia sin errores. Los diagramas: (a), (b), (c), (d), (e) y (f), informan del correcto comportamiento de los bucles PLL y DLL, sin apreciar en las gráficas de un mejor comportamiento de los bucles al emplear señales filtradas. Por lo que filtrar señales no implica un mejor o peor desempeño del bloque de tracking, ya que su finalidad es recuperar la señal de referencia y se ha conseguido.

7.2 Estudio de rendimiento de los algoritmos empleados

En este apartado se analizarán las estimaciones de tiempo computacional que emplea el software Matlab en resolver los algoritmos vistos en el apartado 6.2.

A diferencia del apartado anterior, esta prueba se centra solo en rendimiento temporal de los algoritmos de adquisición y tracking, por lo que se utiliza una secuencia de señal de referencia sin filtrar, ya que la prueba dependerá del número de muestras de la señal de referencia y no del filtrado de estas muestras.

7.2.1 Resultados temporales del bloque de Adquisición

Para esta prueba se realiza un total de 10 iteraciones por código de adquisición (apartado 6.2.1), calculándose el máximo, mínimo y mediana del conjunto de tiempos. Se obtiene los siguientes resultados:

7.2.1.1 Tiempos de *adq0*

En la Ilustración 7-16, se observa el conjunto de tiempos obtenidos por el código *adq0*. Este código no presenta ningún tipo de optimización y se comprueba como punto de referencia del análisis.

En la Tabla 7-4, se resume el máximo, mínimo y mediana del conjunto de tiempos obtenidos.

Tabla 7-4. Resumen de tiempos obtenidos por el código *adq0*

	Tiempo [s]
Máximo	3,9010
Mediana	3,8731
Mínimo	3,8421

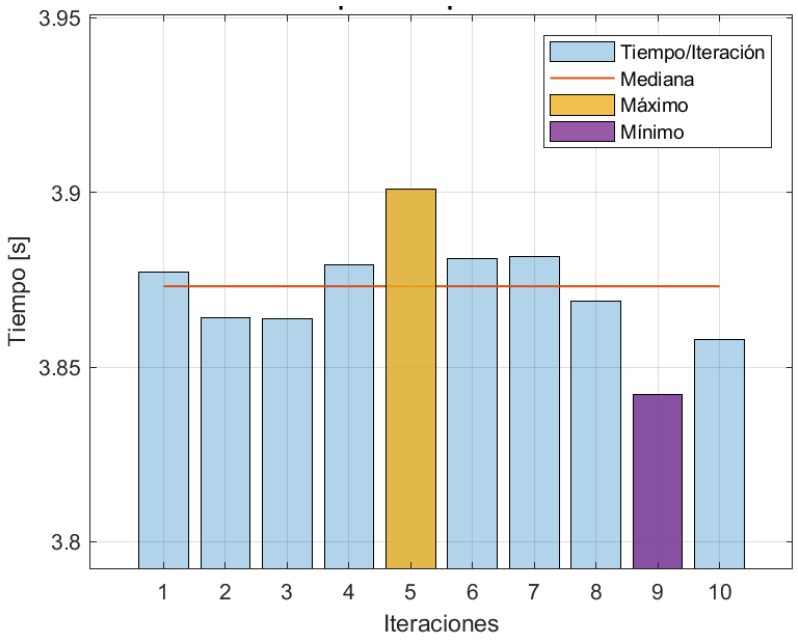

Ilustración 7-16. Tiempos por iteración obtenidos por el código *adq*$_0$

7.2.1.2 Tiempos de *adq*$_1$

En la Ilustración 7-17, se observa el conjunto de tiempos obtenidos por el código *adq*$_1$. Los detalles de este código se listan en 6.2.1.1.

En la Tabla 7-5, se resume el máximo, mínimo y mediana del conjunto de tiempos obtenido:

Tabla 7-5. Resumen de tiempos obtenidos por el código adq$_1$

	Tiempo [s]
Máximo	1,2652
Mediana	1,1706
Mínimo	1,1666

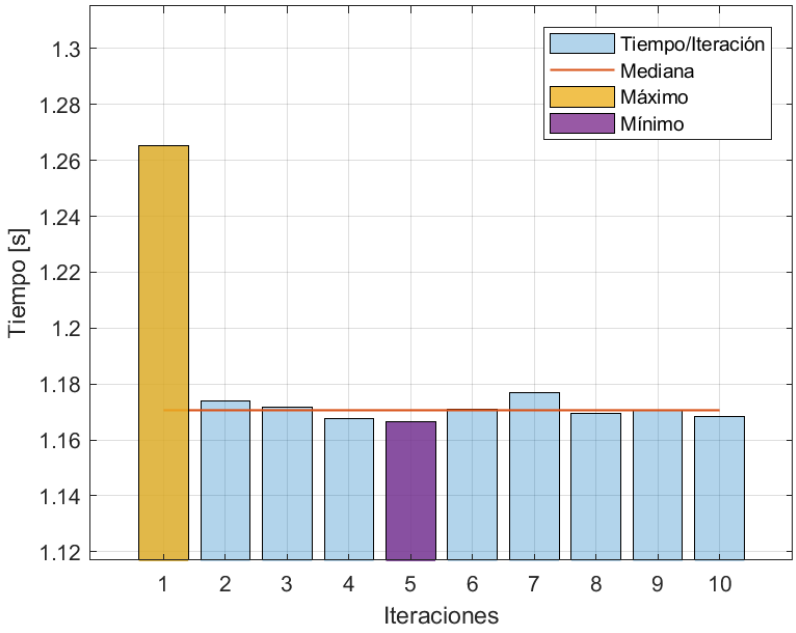

Ilustración 7-17. Tiempos por iteración obtenidos por *adq1*

7.2.1.3 Tiempos de adq2

En la Ilustración 7-18, se observa el conjunto de tiempos obtenidos por el código adq2. Los detalles de este código se listan en 6.2.1.2.

En la Tabla 7-6, se resume el máximo, mínimo y mediana del conjunto de tiempos obtenido:

Tabla 7-6. Resumen de tiempos obtenidos por el código adq2

	Tiempo [s]
Máximo	1,4425
Mediana	1,3503
Mínimo	1,3476

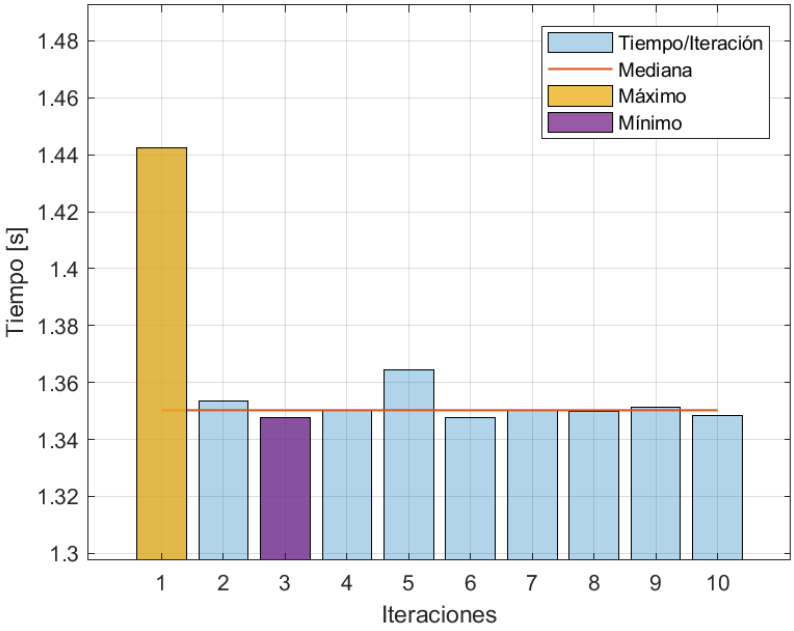

Ilustración 7-18. Tiempos por iteración obtenidos por adq2

7.2.1.4 Tiempos de adq3

En la Ilustración 7-19, se observa el conjunto de tiempos obtenidos por el código adq3. Los detalles de este código se listan en 6.2.1.3.

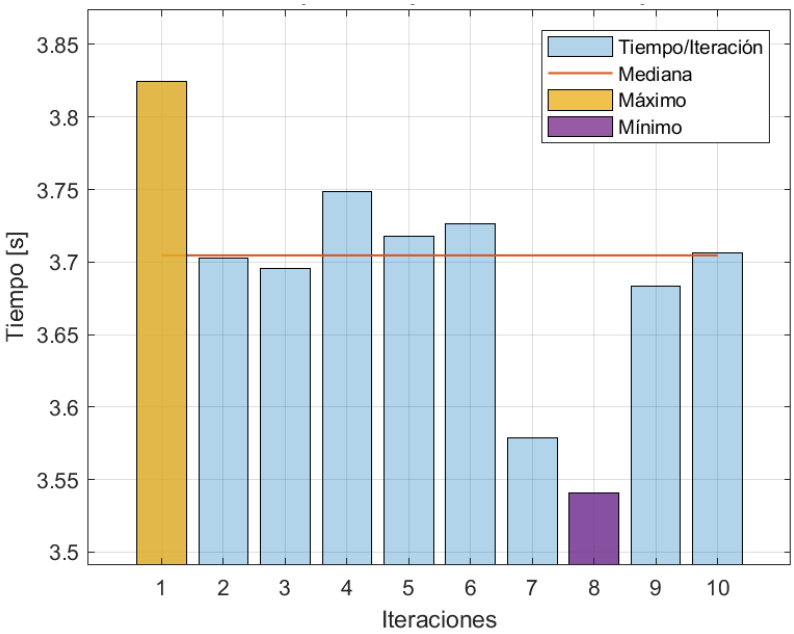

Ilustración 7-19. Tiempos por iteración obtenidos por adq3

En la Tabla 7-7, se resume el máximo, mínimo y mediana del conjunto de tiempos obtenido:

Tabla 7-7. Resumen de tiempos obtenidos por el código adq_3

	Tiempo [s]
Máximo	3,8243
Mediana	3,7046
Mínimo	3,5410

7.2.1.5 Diferencias en las gráficas de adquisición

En las gráficas que se presentan para los códigos adq_1 (Ilustración 7-17), adq_2 (Ilustración 7-18) y adq_3 (Ilustración 7-19) se observa un inconveniente común: la primera iteración consigue un tiempo que se desvía mucho respecto a la mediana de cada caso. Esto es debido a la transferencia de datos que se realiza a la GPU, en los dos primeros casos dado que emplea funciones optimizadas en ella; y la transferencia de datos a los hilos o workers que se forman por cada núcleo de la CPU, en el tercer caso dado que emplea bucles parfor. Se tiene en cuenta que la mediana se aproxima al tiempo de cálculo.

7.2.2 Resultados temporales del bloque de Tracking

Para esta prueba se realiza un total de 10 iteraciones por algoritmo de tracking (apartado 6.2.2), calculándose el máximo, mínimo y mediana del conjunto de tiempos obtenido. Se comprueban los resultados tanto por canal activo (1 canal activo por código PRN identificado) como por iteraciones completas (iteración se entiende al cálculo del bloque de tracking para todos los canales activos) del bloque.

7.2.2.1 Tiempos de *trck*$_0$

En la Ilustración 7-20, se observa el conjunto de tiempos resultante tras el desempeño del código *trck*$_0$. Se toman los resultados **por canales activos**.

En la Tabla 7-8, se resume el máximo, mínimo y mediana del conjunto de tiempos obtenido por canal activo:

Tabla 7-8. Resultados obtenidos de *trck*$_0$ por canal

	Máximo [s]	**Mediana [s]**	**Mínimo [s]**
Canal 1	35,8893	34,8287	34,5309
Canal 2	34,5915	34,2385	34,0008
Canal 3	35,0188	34,5643	34,4917
Canal 4	35,0134	34,7439	34,5377

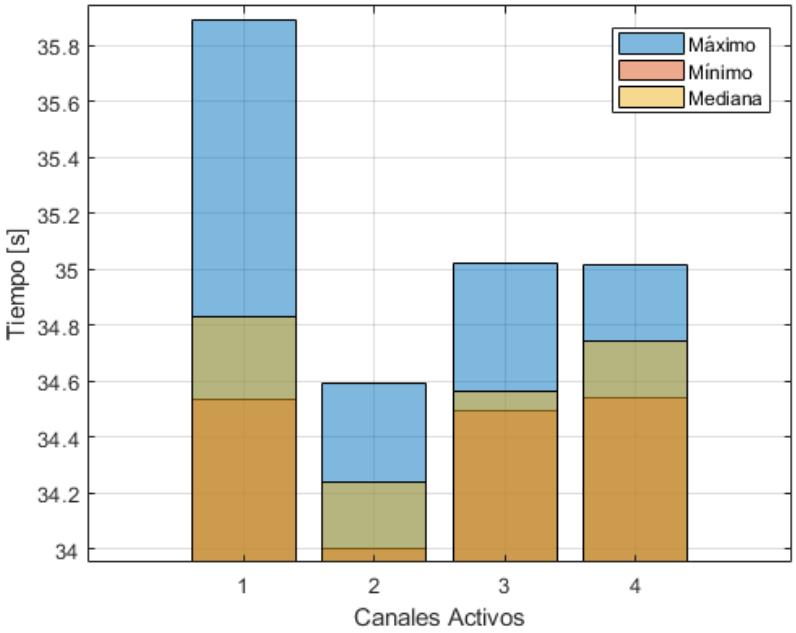

Ilustración 7-20.Conjunto de tiempos de *trck₀* por canal, duración de la señal 20 s

Por otro lado, se calcula un conjunto de tiempos en base a iteraciones completas del algoritmo de *trck₀* y de forma general se calcula un conjunto de tiempos por iteraciones completas. Como por cada iteración se evalúan todos los canales activos, el tiempo por iteración equivale a la suma de tiempos de todos los canales activos, tal como se resume en la siguiente ecuación (7.1):

$$t_{iteración} = \sum_{c=0}^{N_c} t_{canal_c} \tag{7.1}$$

Siendo *c* el indicador de canal y N_c el número de canales activos.

En la Ilustración 7-21, se observa el conjunto de tiempos resultante tras el desempeño del código *trck₀*. Se toman los resultados **por iteraciones completas**.

En la Tabla 7-9, se resume el máximo, mínimo y mediana del conjunto de tiempos obtenido por iteraciones.

Tabla 7-9. Resultados obtenidos de *trck₀* por iteración

	Tiempo/iteración [s]
Máximo	139,7546
Mediana	138,4269
Mínimo	138,0341

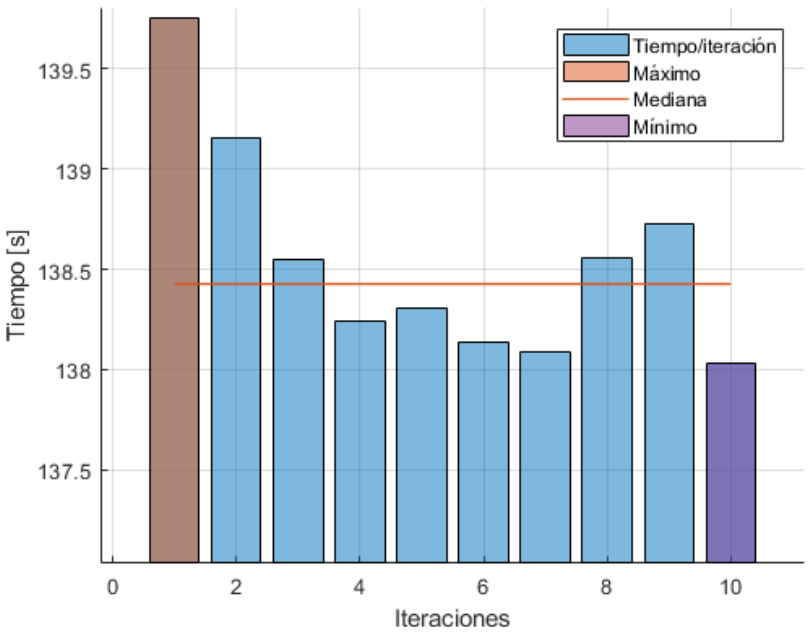

Ilustración 7-21. Conjunto de tiempos de *trck₀* por iteración, duración de la señal 20 s

Aplicando la ecuación (7.1) sobre los datos obtenidos en la Tabla 7-8, se obtienen valores aproximados a los de la Tabla 7-9. Se dice valores aproximados y no exactamente lo iguales ya que en la Tabla 7-8 se calculan los tiempos en sólo una iteración en cambio en la Tabla 7-9 se calcula sobre 10 iteraciones.

7.2.2.2 Tiempos de *trck₁*

En la Ilustración 7-22, se observa el conjunto de tiempos resultante tras el desempeño del código *trck₁*. Se toman los resultados **por canales activos**.

En la Tabla 7-10, se resume el máximo, mínimo y mediana del conjunto de tiempos obtenido por canal.

Tabla 7-10. Resultados obtenidos de *trck₁* por canal

	Máximo [s]	**Mediana [s]**	**Mínimo [s]**
Canal 1	30,5235	25,5262	25,2195
Canal 2	29,5743	29,3959	29,2947
Canal 3	29,9098	29,5013	29,3057
Canal 4	25,9853	25,6716	25,4486

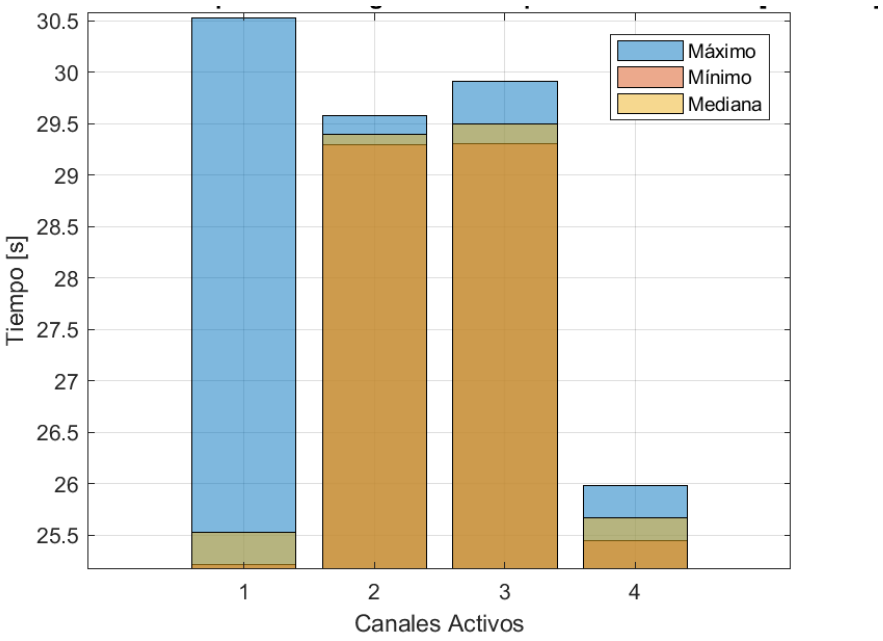

Ilustración 7-22. Conjunto de tiempos de *trck1* por canal, duración de la señal 20 s

En la Ilustración 7-23, se observa el conjunto de tiempos resultante tras el desempeño del código *trck1*. Se toman los resultados **por iteraciones completas**.

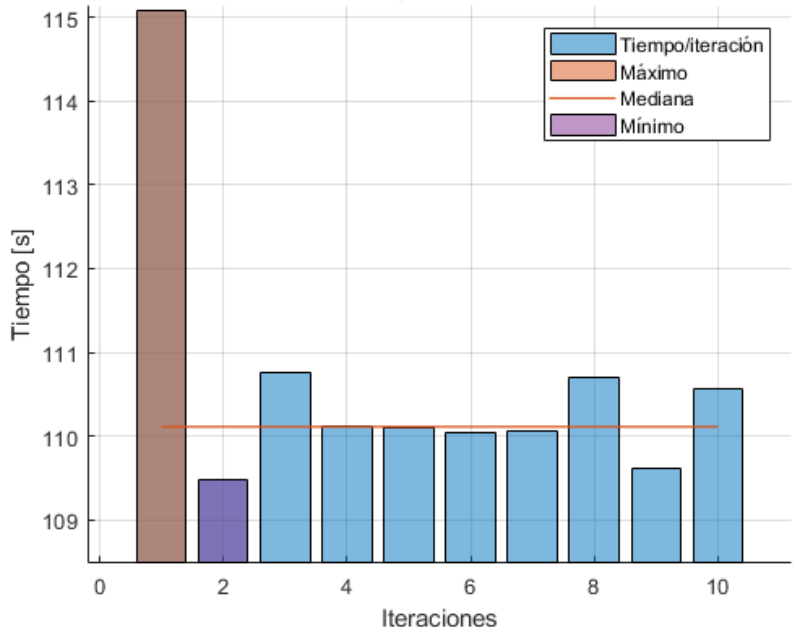

Ilustración 7-23. Conjunto de tiempos de *trck1* por iteraciones, duración de la señal 20 s

La Tabla 7-11, se resume el máximo, mínimo y mediana del conjunto de tiempos obtenido por iteraciones:

Tabla 7-11. Resultados obtenidos de *trck1* por iteración

	Tiempo/iteración [s]
Máximo	115,0807
Mediana	110,1118
Mínimo	109,4814

Al igual que sucede en *trck0*, si se aplica (7.1) sobre la Tabla 7-10, se obtienen valores aproximados de la Tabla 7-11.

7.2.2.3 Diferencias en las gráficas de tracking

En la gráfica que se presenta para el código trck1 (Ilustración 7-23) se observa nuevamente el mismo inconveniente que en adquisición: la primera iteración consigue un tiempo que se desvía mucho respecto a la mediana. Esto es debido a la transferencia de datos que se realiza a la GPU dado que este código emplea funciones optimizadas en ella. En este caso la transferencia de datos toma un total de 5 segundos aproximadamente, mucho mayor que en los casos de adquisición debido a que el procesado de señal en tracking es mucho mayor.

7.2.3 Rendimiento: Ganancia de velocidad

De cara a comprender las mejoras que se obtiene cuando se aplica diferentes técnicas de optimización, se analizarán los resultados aplicando la Ley de Amdahl a partir de la ganancia de velocidad: "la mejora obtenida en el rendimiento global de un computador (en este caso, de un algoritmo) al utilizar algún modo de ejecución más rápido está limitada por la fracción de tiempo que se puede utilizar ese modo más rápido" [41].

Luego la ganancia global de un algoritmo se formula de la siguiente manera (7.2):

$$gv_{global} = \frac{T_{sin}}{T_{con}} \tag{7.2}$$

Siendo:

- T_{sin} es el tiempo que emplea el algoritmo original.
- T_{con} es el tiempo que emplea el algoritmo mejorado.

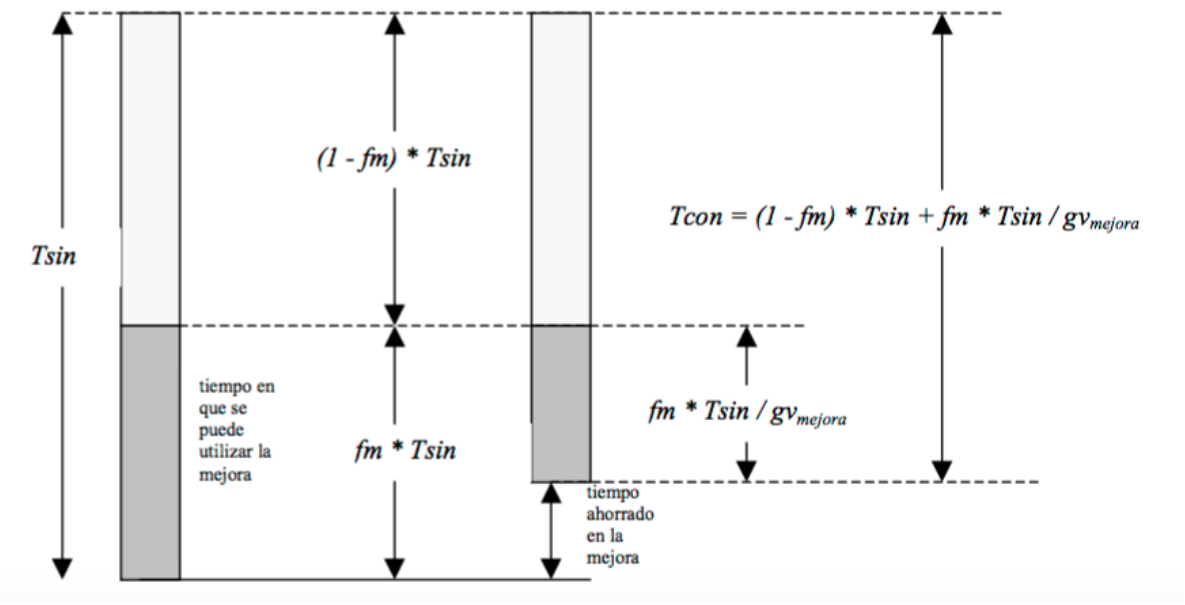

Ilustración 7-24. Diagrama de Ley de Amdahl

En la Ilustración 7-24, se muestra el diagrama explicativo de la ley de Amdahl, donde se observan parámetros con los que también se puede obtener la ganancia global:

- f_m es la fracción de tiempo en que se puede utilizar la mejora.
- gv_{mejora} es la ganancia de velocidad propia del elemento mejorado.
- t es el tiempo ahorrado en la mejora.

Aplicando estos parámetros, la ganancia global del sistema se puede reformular de la siguiente forma (7.3):

$$gv_{global} = \frac{T_{sin}}{T_{con}} = \frac{T_{sin}}{(1 - f_m) \cdot T_{sin} + f_m \cdot \frac{T_{sin}}{gv_{mejora}}} = \frac{1}{(1 - f_m) + \frac{f_m}{gv_{mejora}}}$$

(7.3)

De esta manera se observa que la ganancia global depende de los parámetros vistos f_m y gv_{mejora}.

Suponiendo que la $gv_{mejora} \to \infty$, el tiempo en que se puede utilizar la mejora ($fm \cdot T_{sin}$) coincide con el tiempo ahorrado en la mejora (t) y la ganancia de velocidad global se reduce a (7.4).

$$gv_{global} = \frac{1}{(1 - f_m)}$$

(7.4)

Por otra parte, dado que la ganancia de velocidad global de sistema se puede obtener a partir de T_{sin} y T_{con}, como en (7.2), se puede despejar de (7.4), el parámetro de fracción de mejora f_m (7.5).

$$f_m = 1 - \frac{1}{gv_{global}} \tag{7.5}$$

Luego para comprobar las mejoras realizadas por la optimización de código, se calcularán los parámetros gv_{global} y f_m, que describen correctamente la optimización de código.

7.2.4 Comparativas de los algoritmos implementados

Para comparar la mejora de los algoritmos implementados en adquisición y tracking se aplicará la ley de Amdahl (apartado 7.2.3) sobre los resultados temporales obtenidos (apartados 7.2.1 y 7.2.2, datos de adquisición y tracking, respectivamente), en concreto sobre la mediana de tiempos; ya que es un dato estadístico que se sitúa en un punto central de probabilidad, y que además, evita efectos no deseados sobre los resultados como picos de estrés computacional.

7.2.4.1 Bloque de Adquisición

En el bloque de adquisición se aplica la ley de Amdahl sobre los datos recogidos en las tablas: Tabla 7-4, Tabla 7-5, Tabla 7-6 y Tabla 7-7, correspondientes a los algoritmos adq_0, adq_1, adq_2 y adq_3. Las medianas de tiempo conseguidas por los algoritmos son:

- Med_adq_0 = 3,8731 s
- Med_adq_1 = 1,1706 s
- Med_adq_2 = 1,3503 s
- Med_adq_3 = 3,7046 s

Al aplicar las medianas de tiempo sobre la ecuación de ganancia de velocidad (7.2), siendo T_{sin} igual a Med_adq_0 y T_{con} las demás medianas, se obtiene:

- $gv_{global\_adq\_17}$ = 3,3086. adq_1 es un 230 % más veloz que adq_0, aproximadamente.
- $gv_{global\_adq\_28}$ = 2,8683. adq_2 es un 186 % más veloz que adq_0, aproximadamente.
- $gv_{global\_adq\_39}$ = 1,0455. adq_3 es un 4 % más veloz que adq_0, aproximadamente.

7 $gv_{global\_adq\_1}$ es la ganancia de velocidad de adq_1 frente a adq_0.

8 $gv_{global\_adq\_2}$ es la ganancia de velocidad de adq_2 frente a adq_0.

9 $gv_{global\_adq\_1}$ es la ganancia de velocidad del adq_3 frente a adq_0.

Por último, se comprueba el tiempo ahorrado al aplicar la ganancia de velocidad global en la ecuación (7.5), obteniéndose:

- $f_{m\_adq\_1}$[10] = 0,7418. adq_1 ahorra un 74 % de tiempo frente adq_0, aproximadamente.
- $f_{m\_adq\_2}$[11] = 0,6513. adq_2 ahorra un 65 % de tiempo frente adq_0, aproximadamente.
- $f_{m\_adq\_3}$[12] = 0,0435. adq_3 ahorra un 4 % de tiempo frente adq_0, aproximadamente.

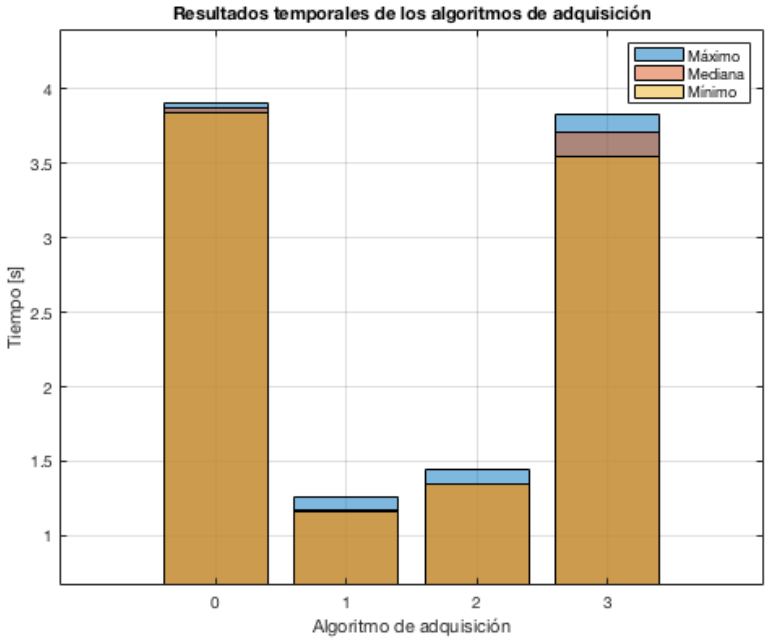

Ilustración 7-25. Resultados temporales de los algoritmos de adquisición

La Ilustración 7-25, resume los resultados temporales obtenidos en adquisición. En ella se puede apreciar la diferencia de tiempo de los algoritmos adq_1, adq_2 y adq_3 frente a adq_0 (algoritmo original). El algoritmo de adq_1 consigue reducir un 74 %, el algoritmo adq_2 reduce un 65% y el algoritmo adq_3 un 4% de tiempo, respecto al algoritmo adq_0. Esto es debido a que la optimización ha conseguido una ganancia de velocidad global del 230% para adq_1, del 186% para adq_2 y del 4 % para adq_3.

Los algoritmos adq_1 y adq_2 consiguen los mejores resultados debido a la reducción de bucles de control, a partir de vectorización de código, pre-asignación de memoria y empleo de funciones optimizadas sobre la GPU. La diferencia entre ambos radica en el empleo de matrices dobles (adq_1) y matrices triples (adq_2), es decir, el empleo de algoritmos con variables más 'pequeñas' consigue ganancias de velocidad superiores, con lo cual mayor reducción de tiempo, luego el tamaño de las variables es un factor muy

10 $f_{m\_adq\_1}$ es el factor de mejora de adq_1 frente a adq_0, lo que equivale al tiempo ahorrado.

11 $f_{m\_adq\_2}$ es el factor de mejora de adq_2 frente a adq_0, lo que equivale al tiempo ahorrado.

12 $f_{m\_adq\_3}$ es el factor de mejora de adq_3 frente a adq_0, lo que equivale al tiempo ahorrado.

importante en la optimización de código en Matlab. En el caso del algoritmo adq_3, la reducción de tiempo conseguida no es tan significativa debido al empleo del bucle PARFOR para la paralelización de código. El empleo de bucle PARFOR a priori debe ser la opción con mayor potencial, pero no se consigue grandes reducciones de tiempo debido a la carga computacional que se genera con la gran la cantidad de operaciones que se producen en los hilos[13] del bucle. Por tanto, el bucle PARFOR es una buena solución siempre el número de iteraciones sea muy superior al número de operaciones dentro de cada hilo. Hay que destacar que el algoritmo emplea un hilo por cada código PRN, es decir, un total de 32 hilos, con lo cual no supera en gran cantidad al número de operaciones dentro del algoritmo.

7.2.4.2 Bloque de tracking

En el bloque de tracking se aplica la ley de Amdahl sobre los datos recogidos en las tablas: Tabla 7-9 y Tabla 7-11, correspondientes a los algoritmos $trck_0$ y $trck_1$.

Las medianas de tiempo conseguidas por los algoritmos son:

- Med_trck_0 = 138,4269 s
- Med_trck_1 = 110,1118 s

Al igual que en el bloque de adquisición se aplica las medianas de tiempo sobre la ecuación de ganancia de velocidad (7.2), siendo T_{sin} igual a Med_trck_0 y T_{con} igual a Med_trck_1, se obtiene:

- $gv_{global\_trck\_1}$[14] = 1,2571. $trck_1$ es un 25 % más veloz que $trck_0$, aproximadamente.

Por último, se comprueba el tiempo ahorrado al aplicar la ganancia de velocidad global en la ecuación (7.5), obteniéndose:

- $f_{m\_trck\_1}$[15] = 0,2045. $trck_1$ ahorra un 20 % de tiempo frente $trck_0$, aproximadamente.

La Ilustración 7-26, se observan los resultados temporales de los algoritmos de tracking. En ella se observa la diferencia de tiempo de los algoritmos $trck_0$ y $trck_1$. El algoritmo $trck_1$ consigue reducir un 20 % del tiempo empleado por $trck_0$ (algoritmo original), debido a un incremento de velocidad del 25 %.

[13] Se entiende hilo como "worker" de trabajo que emplea MATLAB para la programación paralela.

[14] $gv_{global\_trck\_1}$ es la ganancia de velocidad de $trck_1$ frente a $trck_0$.

[15] $f_{m\_trcl\_1}$ es el factor de mejora de adq_1 frente a adq_0, lo que equivale al tiempo ahorrado.

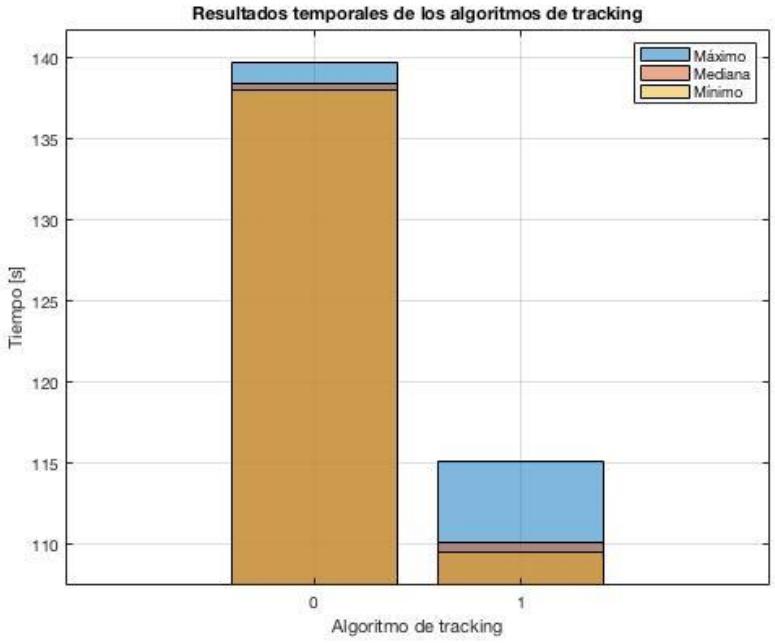

Ilustración 7-26. Resultados temporales de los algoritmos de tracking

En el tracking se procesa 20 s de señal de referencia frente a 1 ms de señal procesada por el bloque de adquisición. Por cada milisegundo de señal de referencia se completa un ciclo del bucle de tracking, luego para procesar 20 segundos es necesario un total de 20000 ciclos de bucle de tracking, sumado al número de canales activos que en este caso son 4 canales (1 canal activo por código PRN identificado) se necesita un total de 80000 ciclos para completar el tracking. Luego el bloque de tracking requiere un mayor tiempo de procesado respecto al bloque de adquisición. Por esta razón, la reducción de un 20 % (a priori una reducción de tiempo menor) corresponde con reducir cerca de 28 segundos respecto al algoritmo original, es decir, hay una ganancia de velocidad notable debido al empleo de variables de tipo gpuArray que realizan cálculos de funciones optimizadas sobre la GPU, vectorización de código y pre-asignación de memoria. Sin embargo, no se ha utilizado técnicas de paralelización de código con PARFOR debido a que el algoritmo de tracking es un algoritmo de seguimiento, dependiente de resultados, gran inconveniente de la programación paralela. Por otra parte, en el caso de emplear matrices tridimensionales no es viable debido a que no reduce el número de bucles internos en el algoritmo y que, además, el tamaño de datos sería mucho mayor.

8 Conclusiones y líneas futuras

8.1 Conclusiones del estudio de implementación de algoritmos con señales filtradas

El estudio de implementación de algoritmos con señales filtradas busca comprobar si de alguna manera el hecho de filtrar señales genera variaciones en la sensibilidad de los bloques de adquisición y tracking.

En el bloque de adquisición se busca comprobar si realmente mejora las capacidades de identificación de códigos PRN en la señal de referencia y sus parámetros Doppler y retardo de fase durante el primer milisegundo de señal.

En cuanto al bloque de tracking busca comprobar si se mejora el comportamiento de los bucles PLL y DLL durante 20 segundos de señal.

Dadas las gráficas y resultados obtenidos en los apartados 7.1.1 y 7.1.2, se puede concluir que filtrar la señal de referencia, elimina ruido, pero no genera mejoras en los algoritmos. Debido a las siguientes causas:

- En el bloque de adquisición se buscaba comprobar si filtrar la señal incrementa las capacidades detectoras de código PRN en la señal de referencia, donde se consiguió correlaciones diferentes aplicando diferentes secuencias, pero diferencias no significativas hasta el punto de que ningún otro código PRN consiguió superar el factor de calidad Threshold. Por otra parte, los parámetros obtenidos tanto en la adquisición gruesa como en la fina han sido exactamente los mismos empleando secuencias sin filtrar y filtradas, por lo que en cuestión de sensibilidad de tiempo (retardo de fase) y de frecuencia (frecuencia Doppler) tampoco se ha observado mejoras.

- En el bloque de tracking se busca comprobar si los bucles DLL y PLL tienen mejor comportamiento al aplicar señales filtradas, pero no se observa ninguna diferencia en el comportamiento de los bucles, además cabe destacar que la finalidad del bloque de tracking es la recuperación del mensaje de datos de navegación y puesto que en los tres casos se consigue recuperar este, el filtrado no aporta mejoras.

8.2 Conclusiones del estudio de rendimiento de los algoritmos empleados

El estudio del rendimiento de algoritmos empleados busca comprobar si la aplicación de técnicas de optimización de código genera mejoras de rendimiento de los algoritmos de los bloques de adquisición y de tracking.

Para la optimización de código se ha decidido realizar un desarrollo de algoritmos a partir de la vectorización de código, la pre-asignación de memoria y la programación paralela.

En el bloque de adquisición se optan por emplear estas técnicas de formas alternativas. El algoritmo *adq1* emplea técnicas como la vectorización de código, pre-asignación de memoria y matrices del tipo gpuArray de dos dimensiones optimizadas sobre la GPU que permite acelerar el código eficazmente. En el algoritmo *adq2* se emplean los mismos métodos que el algoritmo anterior, pero con la diferencia de que se generan matrices de tres dimensiones, potenciando la reducción de bucles dentro del algoritmo a costa de un mayor coste computacional. En el algoritmo *adq3* se decide aplicar técnicas de paralelización de código mediante el empleo de un bucle PARFOR.

En el bloque de tracking se estudia la posibilidad de emplear técnicas de paralelización de código mediante bucles PARFOR, sin embargo, este método no es viable dado que se trata de un bloque de seguimiento de señal y por tanto dependiente de resultados previos. Luego en el algoritmo *trck1* se aplican técnicas de pre-asignación de memoria y vectorización de código para optimizar código de una manera más eficaz.

Al aplicar distintas técnicas en la optimización de código se ha obtenido las siguientes conclusiones:

- La gran transferencia de datos ya sea entre funciones o entre la CPU y la GPU, genera grandes inconvenientes. Hay que evitar en mayor medida de lo posible variables del tipo global puesto que puede generar pérdidas de rendimiento en el código de MATLAB.

- Se debe evitar en gran medida la ampliación de variables aplicando pre-asignación de memoria. Esta es una técnica que mejora el rendimiento del código sobre todo cuando incluye bucles o bucles anidados.

- Se debe evitar el uso de bucles mediante la vectorización de códigos, aplicando operaciones matriciales y vectoriales. MATLAB tiene un gran rendimiento en operaciones vectoriales o matriciales por lo que el uso de bucles está desaconsejado.

- Se debe evitar repetir el uso de operaciones independientes dentro de bucles ya que generaría operaciones redundantes.

- Se debe transformar el código serie en código paralelo. Puesto que si el cálculo de un bucle o una función es concurrente se puede emplear la programación en paralelo como alternativa.

Dadas las gráficas y resultados obtenidos en los apartados 7.2.4.1 y 7.2.4.2 a partir de la aplicación de la ley de Amdahl que comprueba el incremento de rendimiento de los algoritmos a partir de la ganancia de velocidad, se obtienen las siguientes conclusiones:

- Se observa como los algoritmos que aplican gpuArray y vectorización de código para la aceleración de código son los algoritmos ganancia de velocidad, reducción de tiempo y por consecuencia mejor rendimiento. Esto se cumple tanto en el bloque de adquisición como en el bloque de tracking. Esto es debido también a que MATLAB ha optimizado funciones sobre la GPU como: *fft, ifft, exp, fftshift, circshift, abs, sum, horzcat, nex2pow;* y que la gestión de datos vectoriales es mejor que la gestión de datos matriciales. De esta forma se genera un código mucho más fácil de leer, de modificar y que, además, está optimizado.
- El empleo de matrices de menor tamaño influye en los resultados, este es el caso de los algoritmos *adq1* y *adq2* que emplean las mismas técnicas de optimización, pero con tamaños de variables diferentes.
- El empleo de paralelización de código mediante bucle PARFOR genera es viable en el caso del bloque de adquisición sin embargo no lo es en el bloque de tracking, debido a que emplea ciclos dependientes. Los resultados de la paralelización de *adq3* no generan una gran aceleración de código, debido a que el número de iteraciones no es muy superior al número de operaciones empleadas en el bucle.
- Con el algoritmo de *trck3* se observa una gran reducción temporal debido a que es un bloque que procesa 20 segundos de señal de referencia frente a 1 milisegundo del bloque de adquisición, por lo que es necesario un total de 20000 ciclos de tracking por cada canal activo. Esto genera una gran reducción de tiempo, sin embargo, en cuanto a mejora de rendimiento no supera la reducción de casi un 75% conseguida en adquisición frente a un 20% en el caso del tracking.

8.3 Líneas futuras

En este TFG se han estudiado y aplicado diferentes formas de optimizar código para reducir el tiempo de procesado del demostrador IDEPAR cuando utiliza satélites del sistema GPS como iluminadores de oportunidad.

Se han considerado técnicas destacando la vectorización de código, la pre-asignación de memoria y la programación paralela.

En relación con la programación paralela, se han detallado dos posibilidades: el empleo de funciones optimizadas sobre la GPU con variables del tipo gpuArray que aceleran el código o el empleo de bucles PARFOR. Los bucles PARFOR requieren la identificación de las partes del código que son paralelizables y las que son secuenciales. Se ha demostrado que parte del código del bloque de adquisición es paralelizable mediante un bucle PARFOR, en concreto el algoritmo adq3. Sería muy interesante abordar el estudio de la paralelización de código sobre otro tipo de lenguajes, como por ejemplo implementaciones sobre CUDA.

CUDA es una plataforma de computación paralela y un modelo de programación desarrollado por NVIDIA para la computación en GPU. Con CUDA, los desarrolladores pueden acelerar drásticamente las aplicaciones informáticas al aprovechar el poder de las GPU. En las aplicaciones aceleradas por GPU, la parte secuencial de la carga de trabajo

se ejecuta en la CPU, que está optimizada para un rendimiento de un subproceso único, mientras que la parte de computación intensiva de la aplicación se ejecuta en miles de núcleos de la GPU en paralelo [42].

Otra línea futura de gran interés es el estudio de la posibilidad de aplicar las técnicas estudiadas al procesado de otras señales. IDEPAR puede adquirir señales multicanal de la televisión digital terrestre y satelital. En las primeras, se han propuesto algoritmos de reconstrucción de la señal para la implementación de la etapa de rechazo de interferencias previa a la correlación cruzada. Estos algoritmos tienen un elevado coste computacional asociado que podría reducirse utilizando las técnicas estudiadas.

El procesado de señales multicanal y con elevados tiempos de integración, también requieren muchos recursos de cómputo. Se han propuesto metodologías de paralelización en tiempo y frecuencia, que podrían mejorarse mediante la aplicación de las técnicas estudiadas.

Pliego de condiciones

Hardware

El equipo seleccionado para realizar las pruebas necesarias para la optimización de código tiene las siguientes características:

- Fabricante: Hewlett-Packard
- Modelo: HP Z230 Tower Workstation
- Memoria RAM: 16 GB
- Número de procesadores: 4
- Procesador: Intel(R) Core(TM) i7-4790 CPU @3.60 GHz

La tarjeta gráfica instalada en el equipo tiene las siguientes características:

- NVIDIA Quadro K4000
- Memory 3GB GDDR5
- Ancho de banda de memoria: 134.0 GB/s
- Número de procesadores: 768

Software

El software donde se ha implementado el código es MATLAB. Se ha elegido este software por su versatilidad frente al tratamiento digital de la señal que se desea procesar.

El lenguaje de MATLAB se basa en el cálculo matricial por lo que se convierte en el lenguaje idóneo para implementar los algoritmos necesarios en el procesado de la señal de GPS para la localización de blancos [43].

Bibliografía

[1] J. L. Bárcena Humanes, P. J. Gómez del Hoyo, M. P. Jarabo Amores, J. Rosado Sanz y M. d. C. Benito Ortiz, «First Approach to Motion Compensation Considerations for Passive Radar System Based on GPS Signals,» de *2018 19th International Radar Symposium (IRS)*, Bonn, Germany, 2018.

[2] H. Kuschel, D. Cristallini y K. E. Olsen, «Tutorial: Passive radar tutorial,» *IEEE Aerospace and Electronic Systems Magazine* , vol. 34, nº 2, pp. 2-19, 2019.

[3] M. Braasch y A. Dempster, «Tutorial: GPS receiver architectures, front-end and baseband signal processing,» *IEEE Aerospace and Electronic Systems Magazine,* vol. 34, nº 2, pp. 20 - 37, 2019.

[4] IEEE, *Standard for Radar Definitions,* 2017.

[5] «baltrad.fmi.fi,» [En línea]. Available: http://baltrad.fmi.fi/software/rack/doc/rack/html/andrepage.html. [Último acceso: 18 09 2019].

[6] H. D. Griffiths y N. J. Willis, Advances in Bistatic Radar, SciTech Publishing, Inc, 2007.

[7] N. J. Willis, Bistatic Radar, SciTech Publishing, 2005.

[8] A. Lauri, F. Colone, R. Cardinali, C. Bongioanni y P. Lombardo, «Analysis and Emulation of FM Radio Signals for Passive Radar,» de *2007 IEEE Aerospace Conference*, Big Sky, MT, USA, 2007.

[9] C. Coleman, R. Watson y H. Yardley, «A practical bistatic passive radar system for use with DAB and DRM illuminators,» de *2008 IEEE Radar Conference*, Rome, Italy , 2008.

[10] D. Langellotti, F. Colone, P. Lombardo, M. Sedehi y E. Tilli, «DVB-T based Passive Bistatic Radar for maritime surveillance,» de *2014 IEEE Radar Conference,* Cincinnati, OH, USA, 2014.

[11] Z. Sun, T. Wang, T. Jiang, C. Chen y W. Chen, «Analysis of the properties of DVB-S signal for passive radar application,» de *2013 International Conference on Wireless Communications and Signal Processing*, Hangzhou, China, 2013.

[12] S. Brisken, M. Moscadelli, V. Seidel y C. Schwark, «Passive radar imaging using DVB-S2,» de *2017 IEEE Radar Conference (RadarConf)*, Seattle, WA, USA , 2017.

[13] H. Sun, D. K. P. Tan y Y. Lu, «Aircraft target measurements using A GSM-based passive radar,» de *2008 IEEE Radar Conference,* Rome, Italy, 2008.

[14] D. Petri, A. Capria, M. Martorella y F. Berizzi, «Ambiguity function study for UMTS Passive Radar,» de *2009 European Radar Conference (EuRAD)*, Rome, Italy, 2009.

[15] A. Salah, R. Raja Abdullah, A. Ismail, F. Hashim y N. Abdul Aziz, «Experimental study of LTE signals as illuminators of opportunity for passive bistatic radar applications,» *Electronics Letters,* vol. 50, nº 7, pp. 545 - 547, 2014.

[16] H. Guo, K. Woodbridge y C. J. Baker, «Evaluation of WiFi beacon transmissions for wireless based passive radar,» de *2008 IEEE Radar Conference*, Rome, Italy, 2008.

[17] D. Oikonomou, P. Nomikos, G. Limnaios y K. C. Zikidis, «Passive Radars and their use in the Modern Battlefield,» *Journal of Computations & Modelling,* vol. 9, nº 2, pp. 37-61, 2019.

[18] A. Macera, M. Caruso, C. Bongioanni, F. Colone, P. Lombardo, E. Anniballi y R. Cardinali, «Civil Air Traffic surveillance with passive radar for anti-terrorism,» de *2012 Tyrrhenian Workshop on Advances in Radar and Remote Sensing (TyWRRS)*, Naples, Italy, 2012.

[19] R. Zemmari, M. Daun, M. Feldmann y U. Nickel, «Maritime Surveillance with GSM Passive Radar: Detection and Tracking of Small Agile Targets,» de *2013 14th International Radar Symposium (IRS)*, Dresden, Germany, 2013.

[20] M. K. Bączyk, P. Samczynski, P. Krysik y K. Kulpa, «Traffic density monitoring using passive radars,» *IEEE Aerospace and Electronic Systems Magazine,* vol. 32, nº 2, pp. 14 - 21, 2017.

[21] A. Byrd, R. Palmer y C. Fulton, «Concept for a passive multistatic UAV-borne weather radar,» de *2018 IEEE Radar Conference (RadarConf18)*, Oklahoma City, OK, USA, 2018.

[22] R. Blázquez García, J. Casamayón Antón y M. Burgos García, «Nuevo concepto de radares pasivos multiestáticos basados en LTE-R para vigilancia de líneas ferroviarias de alta velocidad,» de *VI Congreso Nacional de I+D en Defensa y Seguridad*, Valladolid, 2018.

[23] P. J. Gómez del Hoyo, N. Del Rey Maestre, D. Mata Moya, M. P. Jarabo Amores, J. L. Bárcena Humanes y J. Rosado Sanz, «Passive Radar Array Processing with Non-Uniform Linear Arrays for Ground Target's Detection and Localization,» 2017.

[24] PCTEL, Inc, «Datasheet GNSS Antennas. GPS L1/L2 Antennas. GPS L1/L2 Active, High Gain, High Performance Magnetic Mount Antenna,» PCTEL, Inc, [En línea]. Available: https://www.pctel.com/antenna-product/gps-l1-l2-active-high-gain-high-performance-magnetic-mount-antenna-gps-l1l2-28mag/. [Último acceso: 15 07 2019].

[25] A. Almodóvar Hernancez, P. J. Gómez del Hoyo, N. Del Rey Maestre, D. Mata Moya, M. P. Jarabo Amores y F. J. Gaitán Cabañas, «Plataformas USRP para el desarrollo de radares pasivos de altas prestaciones reconfigurables,» de *Congreso Nacional de I+D en Defensa y Seguridad (DESEi+d 2018)*, Valladolid, 2018.

[26] J. Rosado Sanz y M. P. Jarabo Amores, Diseño de un array de antenas de parche de banda ancha en UHF para sistemas radar pasivos, Alcalá de Henares: Escuela Politécnica Superior - Universidad de Alcalá de Henares, 2017.

[27] P. Gomez del Hoyo, J. L. Bárcena Humanes, N. del Rey Maestre, J. Rosado Sanz y M. P. Jarabo Amores, «Study of the ghost target phenomenon on a real DVB-T passive radar scenario,» de *2017 Signal Processing Symposium (SPSympo)*, Jachranka, 2017.

[28] Space-Based Positioning Navigation & Timing. National Executive Commitee, «www.gps.gov,» [En línea]. Available: https://www.gps.gov/systems/gps/space. [Último acceso: 17 06 2019].

[29] R. Cardinali, F. Colone, C. Ferretti y P. Lombardo, «Comparison of Clutter and Multipath Cancellation Techniques for Passive Radar,» de *2007 IEEE Radar Conference*, Boston, MA, USA, 2007.

[30] Space-Based Positioning Navigation & Timing National Executive Commitee, «www.gps.gov,» [En línea]. Available: https://www.gps.gov/systems/gps/. [Último acceso: 20 07 2019].

[31] Department of Defense. United States of America, Global Positioning System Standard Positioning Service Performance Standard, 2008.

[32] K. Borre, D. M. Akos, N. Bertelsen, P. Rinder y S. Holdt Jensen, A Software-Defined GPS and Galileo Receiver: A single-Frequency Approach, Birkhauser, 2007.

[33] A. E. Süzer y H. Oktal, «PRN Code Correlation in GPS Receiver,» de *2017 8th International Conference on Recent Advances in Space Technologies (RAST)*, Istanbul, Turkey, 2017.

[34] E. D. Kaplan y C. J. Hegarty, Understanding GPS. Principles and Applications, Artech House, 2006.

[35] J. Bao-Yen Tsui, Fundamentals of Global Positioning System Receivers a Software Approach, New York: John Wiley & Sons, Inc, 2000.

[36] F. Johansson, R. Mollaei, J. Thor y J. Uusitalo, GPS Satellite Signal Acquisition and Tracking, 1998.

[37] MathWorks Parallel Computing Toolbox Team, «Tutorials: Parallel and GPU Computing with MATLAB: All in one (9 parts),» MathWorks, 02 06 2014. [En línea]. Available: https://es.mathworks.com/matlabcentral/fileexchange/46711-tutorials-parallel-and-gpu-computing-with-matlab-all-in-one-9-parts?focused=6798491&tab=function. [Último acceso: 01 07 2019].

[38] 5minutosdematlab.blogspot.com, «Uso de operaciones vectoriales en lugar de ciclos for para aumentar el desempeño de Matlab,» [En línea]. Available: http://5minutosdematlab.blogspot.com/2012/08/uso-de-operaciones-vectoriales-en-lugar.html. [Último acceso: 01 07 2019].

[39] 5minutosdematlab.blogspot.com, «Uso de pre-asignación de memoria en Matlab para aumentar la velocidad de cálculos,» [En línea]. Available: http://5minutosdematlab.blogspot.com/2012/08/uso-de-pre-asignacion-de-memoria-en.html. [Último acceso: 01 07 2019].

[40] Mathworks, «Introducción práctica al filtrado digital,» Mathworks, [En línea]. Available: https://es.mathworks.com/help/signal/examples/practical-introduction-to-digital-filtering.html. [Último acceso: 1 06 2019].

[41] Estructura de computadores, «Tema 4: Rendimiento del procesador,» Facultad de informática, UCM, Madrid, 2011.

[42] NVIDIA Corporation, «NVIDIA. High Performance Computing,» NVIDIA Corporation, [En línea]. Available: https://developer.nvidia.com/cuda-zone. [Último acceso: 30 09 2019].

[43] MathWorks, «Descripción del producto MATLAB,» MathWorks, [En línea]. Available: https://es.mathworks.com/help/matlab/learn_matlab/product-description.html. [Último acceso: 01 07 2019].

[44] J. L. Bárcena Humanes, N. del Rey Maestre, M. P. Jarabo Amores, D. Mata Moya y P. Hoyo Gomez del, «Passive radar imaging capabilities using space-borne

commercial illuminators in surveillance applications,» de *2015 Signal Processing Symposium (SPSympo)*, Debe, Poland, 2015.

[45] B. Hofmann Wellenhof, H. Lichtenegger y E. Wasle, GNSS – Global Navigation Satellite Systems. GPS, GLONASS, Galileo, and more, New York: Springer Wien New York, 2007.

[46] S. Theodoridis y R. Chellappa, Academic Press Library in Signal Processing: Volume 2, Academic Press, 2013.

Anexo I

Pseudocódigo adq0

Proceso acquisition.m

- Iniciando variables: samplesPerCode, signal1, signal2, signal0DC, ts, phasePoints, numberOfFrqBins, caCodesTable, results, frqBins.
- Iniciando estructura: acqResults.carrFreq, acqResults.codePhase, acqResults.peakMetric, acqResults.presencia.
- **Bucle de control de PRN**
    - Cálculo del conjugado de la DFT del código C/A
    - **Bucle de control de frecuencias** adquisición gruesa
        - Genera el grid de frecuencias de portadora
        - Genera las portadoras locales: seno y coseno
        - Calcula las componentes en fase y cuadratura para signal1 y signal2
        - Reconstruye las señales a partir de las componentes en fase y cuadratura y calcula su DFT
        - Producto de la señal de referencia y el código PRN (correlación en el dominio del tiempo) para signal1 y signal2.
        - Paso del resultado al dominio temporal y cálculo de la potencia para signal1 y signal2
        - Comprueba en qué bloque (signal1 o signal2) se alcanza mayor correlación y almacena los resultados
    - **Fin Bucle de control de frecuencias** de adquisición gruesa
    - Encuentra el pico máximo de correlación (peakSize) y la frecuencia portadora (frequencyBinIndex).
    - Encuentra la fase de código (codePhase)
    - Búsqueda de un segundo máximo de correlación con una distancia de al menos chip
    - Encuentra el segundo máximo (secondPeakSize)
    - Almacena en acqResults.peakMetric, el factor de calidad resultante de peakSize/secondPeakSize
    - **Inicio de adquisición fina** si el factor de calidad supera Threshold. Si no, fin de la adquisición fina
        - Muestra código PRN detectado
        - Genera una secuencia de código C/A asociado de 10 ms (caCode)
        - Elimina la modulación de código C/A de la señal (xCarrier)
        - Calcula la DFT (fftxc)
        - Comprueba su máximo y almacena la frecuencia de portadora asociada
        - Almacena los resultados en estructuras: acqResults.carrFreq, acqResults.codePhase y acqResults.presencia
    - **Fin de adquisición fina**
- **Fin del bucle de control PRN**

Fin del acquisition.m

Pseudocódigo adq1

Proceso acquisition_def.m

- Iniciando variables de tipo gpuArray: signal1, signal2, signal0DC, ts, phasePoints, freqBins, carrFreq_aux, codePhase_aux, peakMetric_aux, presencia_aux
- Genera la matriz portadora residuo (expMatrix)
- Reordena tabla de códigos caCodesTable (los códigos PRN se generan fuera de la función 'acquisition_def.m' en chips) en samples
- Cálculo de la DFT del producto entre la señal de entrada (signal1 y signal2) y la matriz portadora residuo. Almacenar resultados en IQfreqDom1 e IQfreqDom2. (Este proceso equivale a: genera el grid de frecuencias de portadora, genera las portadoras locales: seno y coseno, producto de la señal de entrada con las portadoras locales, calcula las componentes en fase y cuadratura, y calcula su DFT; operaciones del bucle de control de frecuencias que se han eliminado)
- **Bucle de control de PRN**
    - Cálculo del conjugado de la DFT del código C/A
    - **Inicio de Adquisición gruesa**
    - Producto de la señal de referencia (IQfreqDom1 e IQfreqDom2) y el código PRN. Este proceso es correlación en el dominio del tiempo
    - Paso del resultado al dominio temporal y cálculo de la potencia para signal1 y signal2
    - Comprueba en qué bloque (signal1 o signal2) se alcanza mayor correlación y almacena los resultados
    - Encuentra el pico máximo de correlación (peakSize) y la frecuencia portadora (frequencyBinIndex)
    - **Fin de la adquisición gruesa**
    - Encuentra la fase de código (codePhase)
    - Búsqueda de un segundo máximo de correlación con una distancia de al menos chip
    - Encuentra el segundo máximo (secondPeakSize)
    - Almacena en acqResults.peakMetric, el factor de calidad resultante de peakSize/secondPeakSize
    - **Inicio de adquisición fina** si el factor de calidad supera Threshold. Si no, fin de la adquisición fina
        - Muestra código PRN detectado
        - Genera una secuencia de código C/A asociado de 10 ms (caCode)
        - Elimina la modulación de código C/A de la señal (xCarrier)
        - Calcula la DFT (fftxc)
        - Comprueba su máximo y almacena la frecuencia de portadora asociada
        - Almacena los resultados en variables: acqResults.carrFreq, acqResults.codePhase y acqResults.presencia (almacenar los resultados en variables es más rápido que almacenarlos en estructuras)
    - **Fin de adquisición fina**
- **Fin del bucle de control PRN**
- Almacena los resultados en estructuras: acqResults.carrFreq, acqResults.codePhase y acqResults.presencia

Fin acquisition_def.m

Pseudocódigo adq2

Proceso acquisitionTriple.m

- Iniciando variables de tipo gpuArray: signal1, signal2, signal0DC, ts, phasePoints, freqBins
- Genera la matriz portadora residuo (expMatrix)
- Reordena tabla de códigos caCodesTable2 (los códigos PRN se generan fuera de la función 'acquisitionTriple.m' en chips) en samples formando una matriz triple (con el empleo de matrices triples se consigue eliminar el bucle de control de PRN)
- Cálculo de la DFT del producto entre la señal de entrada (signal1 y signal2) y la matriz portadora residuo. Almacenar resultados en IQfreqDom1 e IQfreqDom2 (Este proceso equivale a: genera el grid de frecuencias de portadora, genera las portadoras locales: seno y coseno, producto de la señal de entrada con las portadoras locales, calcula las componentes en fase y cuadratura, y calcula su DFT; operaciones del bucle de control de frecuencias que se han eliminado)
- Cálculo del conjugado de la DFT del código C/A (caCodeFreqDomAux)
- Producto de la señal de referencia (IQfreqDom1 e IQfreqDom2) y el código PRN (caCodeFreqDomAux). Este proceso es correlación en el dominio del tiempo. Paso del resultado al dominio temporal y cálculo de la potencia para signal1 y signal2
- Comprueba en qué bloque (signal1 o signal2) se alcanza mayor correlación y almacena los resultados
- Encuentra el pico máximo de correlación (peakSize) y la frecuencia portadora (frequencyBinIndex).
- Encuentra la fase de código (codePhase)
- **Bucle de búsqueda de segundo máximo** de correlación
    - Búsqueda de un segundo máximo de correlación con una distancia de al menos chip (secondPeakSize)
- **Fin de Bucle de búsqueda de segundo máximo** de correlación
- Almacena en acqResults.peakMetric, el factor de calidad resultante de peakSize/secondPeakSize
- **Inicio de adquisición fina** si el factor de calidad supera Threshold. Si no, fin de la adquisición fina
    - Genera una secuencia de código C/A asociado de 10 ms (caCode)
    - Elimina la modulación de código C/A de la señal (xCarrier)
    - Calcula la DFT (fftxc)
    - Comprueba su máximo y almacena la frecuencia de portadora asociada
    - Almacena los resultados en variables: acqResults.carrFreq, acqResults.codePhase y acqResults.presencia (almacenar los resultados en variables es más rápido que almacenarlos en estructuras)
- **Fin de adquisición fina**
- Almacena los resultados en estructuras: acqResults.carrFreq, acqResults.codePhase y acqResults.presencia

Fin acquisitionTriple.m

Pseudocódigo adq3

Proceso acquisition_parfor.m

- **Inicio Bucle PARFOR de control de PRN**
    - Iniciando variables de tipo gpuArray: signal1, signal2, signal0DC, ts, phasePoints, freqBins
    - Genera la matriz portadora residuo (expMatrix)
    - Reorganiza el código PRN (los códigos PRN se generan fuera de la función en chips) en samples. Almacena en caCodesTables
    - Cálculo de la DFT del producto entre la señal de entrada (signal1 y signal2) y la matriz portadora residuo (IQfreqDom1 e IQfreqDom2). (Este proceso equivale a: genera el grid de frecuencias de portadora, genera las portadoras locales: seno y coseno, producto de la señal de entrada con las portadoras locales, calcula las componentes en fase y cuadratura, y calcula su DFT)
    - Cálculo del conjugado de la DFT del código C/A
    - Producto de la señal de referencia (IQfreqDom1 e IQfreqDom2) y el código PRN (correlación en el dominio del tiempo) para signal1 y signal2.
    - Paso del resultado al dominio temporal y cálculo de la potencia para signal1 (acqRes1) y signal2 (acqRes2)
    - Comprueba en qué bloque (signal1 o signal2) se alcanza mayor correlación y almacena los resultados (results) por bloque PRN
    - Encuentra el pico máximo de correlación (peakSize) y la frecuencia portadora (frequencyBinIndex).
    - Encuentra la fase de código (codePhase)
    - Búsqueda de un segundo máximo de correlación con una distancia de al menos chip (secondPeakSize)
    - Almacena en acqResults.peakMetric, el factor de calidad resultante de peakSize/secondPeakSize
    - **Inicio de adquisición fina** si el factor de calidad supera Threshold. Si no, fin de la adquisición fina
        - Muestra código PRN detectado
        - Genera una secuencia de código C/A asociado de 10 ms (longCaCode)
        - Elimina la modulación de código C/A de la señal (xCarrier)
        - Calcula la DFT (fftxc)
        - Comprueba su máximo y almacena la frecuencia de portadora asociada
        - Almacena los resultados en variables: acqResults.carrFreq, acqResults.codePhase y acqResults.presencia (almacenar los resultados en variables es más rápido que almacenarlos en estructuras)
        - Devuelve los resultados de frecuencia de portadora, fase de código y código PRN identificado
    - **Fin de adquisición fina**
- **Fin del bucle de control PRN**

Fin acquisition_parfor.m

Pseudocódigo trck0

Proceso tracking.m

- Inicializa la estructura de resultados (trackResults)
- Inicializa las variables de seguimiento
- **Bucle de control de canales de procesado**
    - **Condicional de control de canal activo if.** Sólo entra si el canal está activo, en caso contrario finalizar el condicional
        - **Bucle de tracking**
            - Lectura de la señal de referencia por milisegundo (rawSignal)
            - Generación de los códigos: Early, Prompt y late (con los datos recogidos de las iteraciones anteriores)
            - Generación de la portadora (con los datos recogidos de las iteraciones anteriores), la componente seno y coseno
            - Producto de la señal (rawSignal) con la portadora. Como resultado se obtiene las componentes en fase (iBaseBandSignal) y cuadratura (qBaseBandSignal).
            - Paso por los 6 correladores del DLL, obteniéndose los valores Early, Late y Prompt tanto de la componente en fase como de cuadratura
            - Encuentra el error de frecuencia (carrFreq) a partir del discriminador del PLL y actualiza el NCO (carrNCO)
            - Encuentra el error de fase (codeFreq) a partir del discriminador del DLL y actualiza el NCO (codeNCO)
            - Almacenar los resultados del bucle de tracking en registros por cada ms
        - **Fin del bucle de tracking**
    - **Fin de condicional de control de canal activo if**
- **Fin de Bucle de control de canales de procesado**

Fin tracking.m

Pseudocódigo trck1

Proceso tracking6.m

- **Condicional de control de canal activo if**. Sólo entra si el canal está activo, en caso contrario finalizar el condicional
    - Inicializa la estructura de resultados (trackResults)
    - Inicializa las variables de seguimiento
    - **Bucle de tracking**
        - Lectura de la señal de referencia por milisegundo (rawSignal)
        - Generación de los códigos a partir de desplazamientos circulares: Early, Prompt y late (con los datos recogidos de las iteraciones anteriores)
        - Generación de la portadora (con los datos recogidos de las iteraciones anteriores), la componente seno y coseno
        - Producto de la señal (rawSignal) con la portadora. Como resultado se obtiene las componentes en fase (iBaseBandSignal) y cuadratura (qBaseBandSignal).
        - Paso por los 6 correladores del DLL, obteniéndose los valores Early, Late y Prompt tanto de la componente en fase como de cuadratura
        - Encuentra el error de frecuencia (carrFreq) a partir del discriminador del PLL y actualiza el NCO (carrNCO)
        - Encuentra el error de fase (codeFreq) a partir del discriminador del DLL y actualiza el NCO (codeNCO)
        - Almacenar los resultados del bucle de tracking en variables por cada ms (almacenar los resultados en variables es más rápido que almacenarlos en estructuras)
    - **Fin del bucle de tracking**
    - Almacena los resultados en los registros trackResults
- **Fin de condicional de control de canal activo if**

Fin tracking6.m

Anexo II

Capturas del bloque de adquisición

Señal sin filtrar

Adquisición gruesa

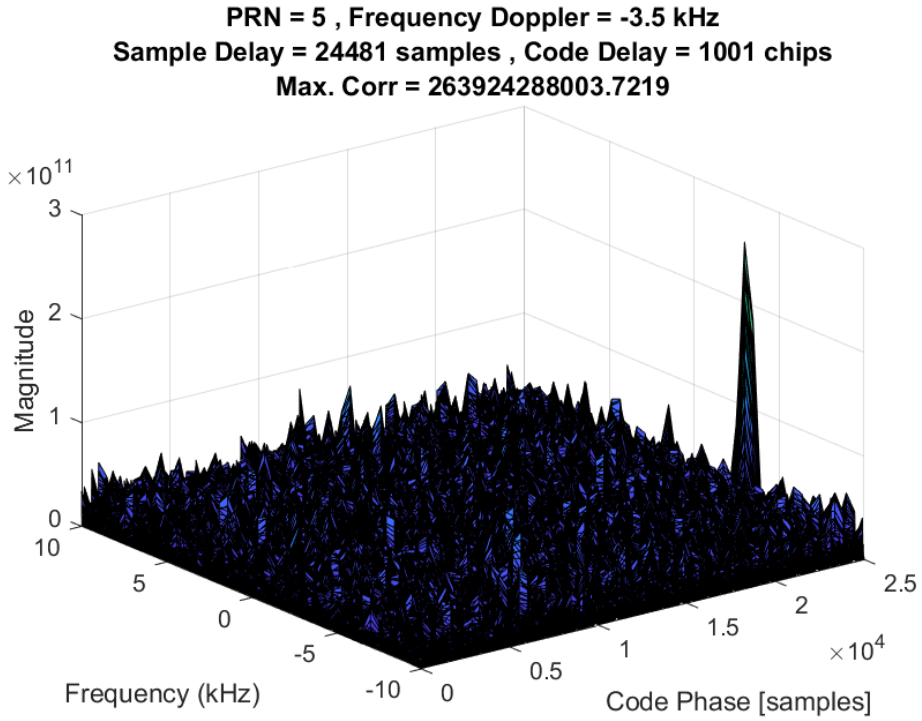

Ilustración 0-1. Adquisición gruesa. Señal sin filtrar: PRN 5; f_d=-3.5 kHz; τ=1001 chips

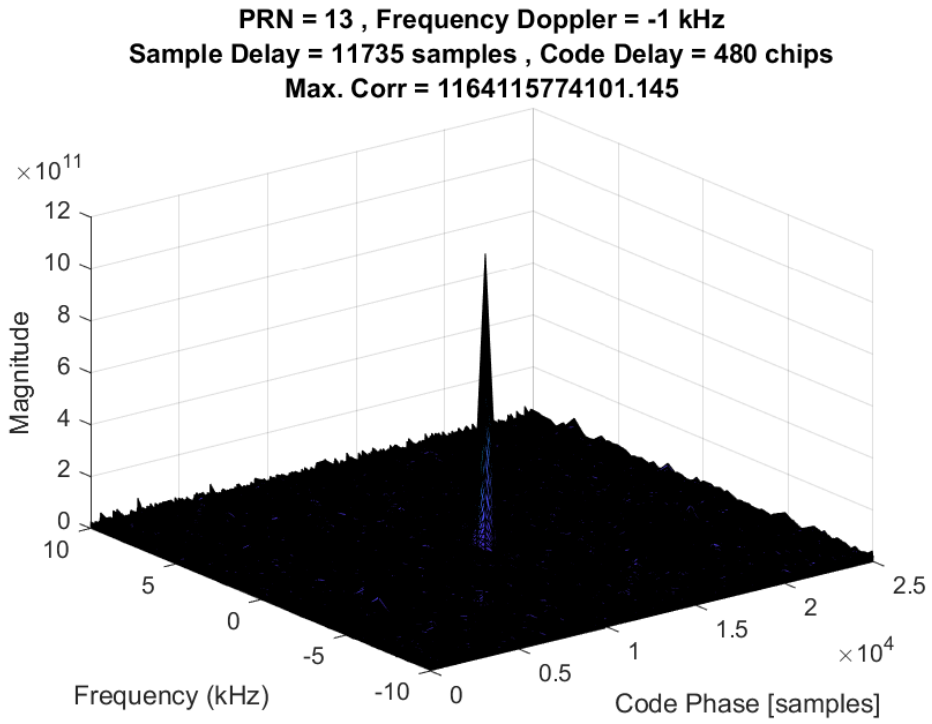

Ilustración 0-2 Adquisición gruesa. Señal sin filtrar: PRN 13; f_d=-1 kHz; τ=480 chips

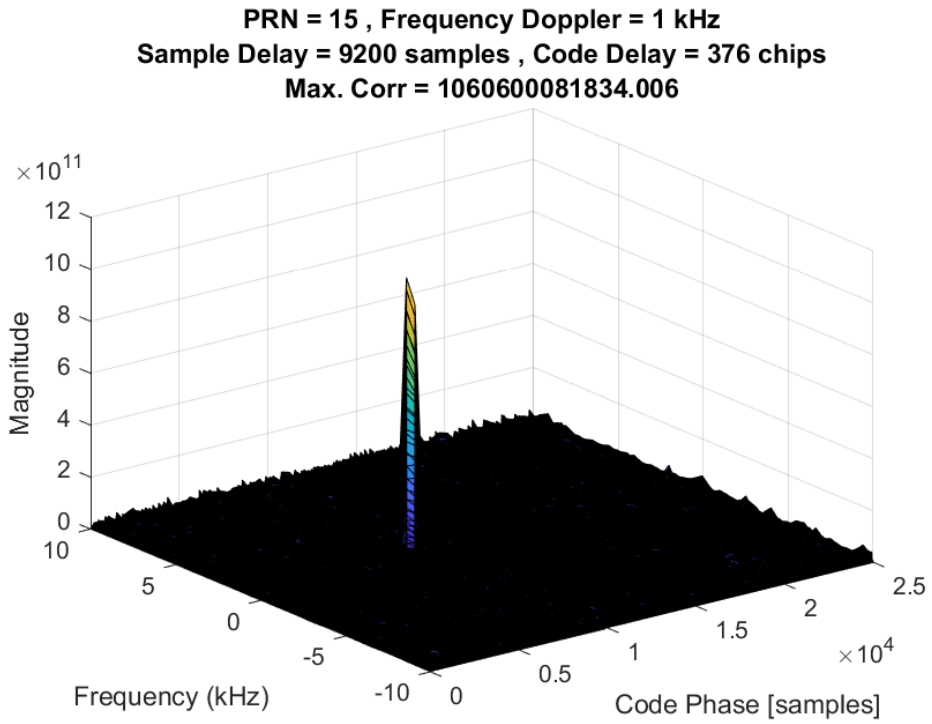

Ilustración 0-3. Adquisición gruesa. Señal sin filtrar: PRN 15; f_d=1 kHz; τ=376 chips

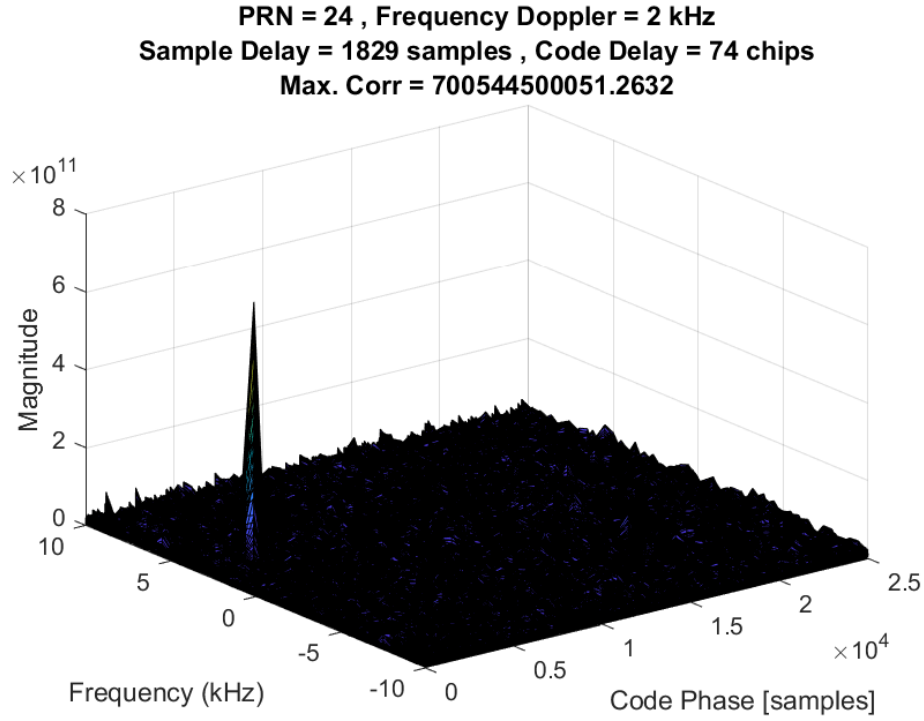

Ilustración 0-4. Adquisición gruesa. Señal sin filtrar: PRN 24; f_d=2 kHz; τ=74 chips

Adquisición fina

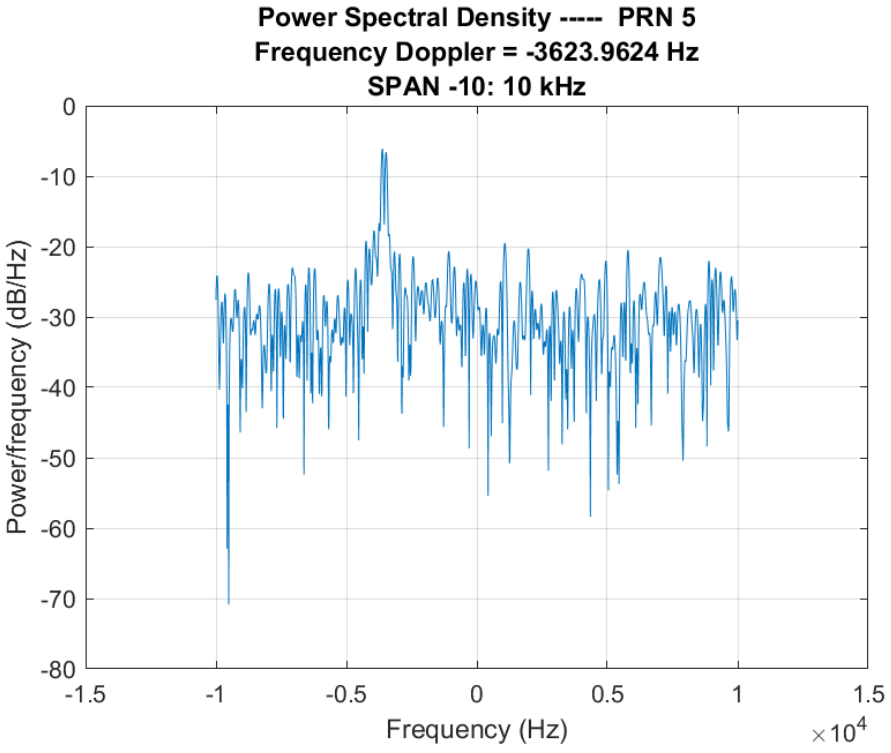

Ilustración 0-5. Adquisición fina. Señal sin filtrar: PRN 5; f_d=-3623,9624 Hz

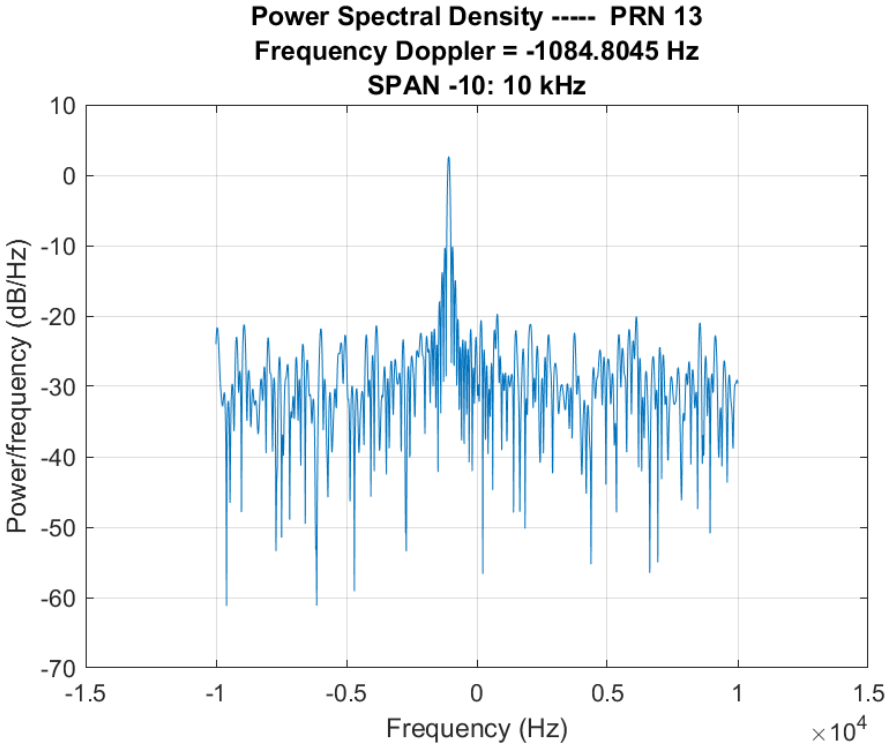

Ilustración 0-6. Adquisición fina. Señal sin filtrar: PRN 13, fd = -1084,8045 Hz

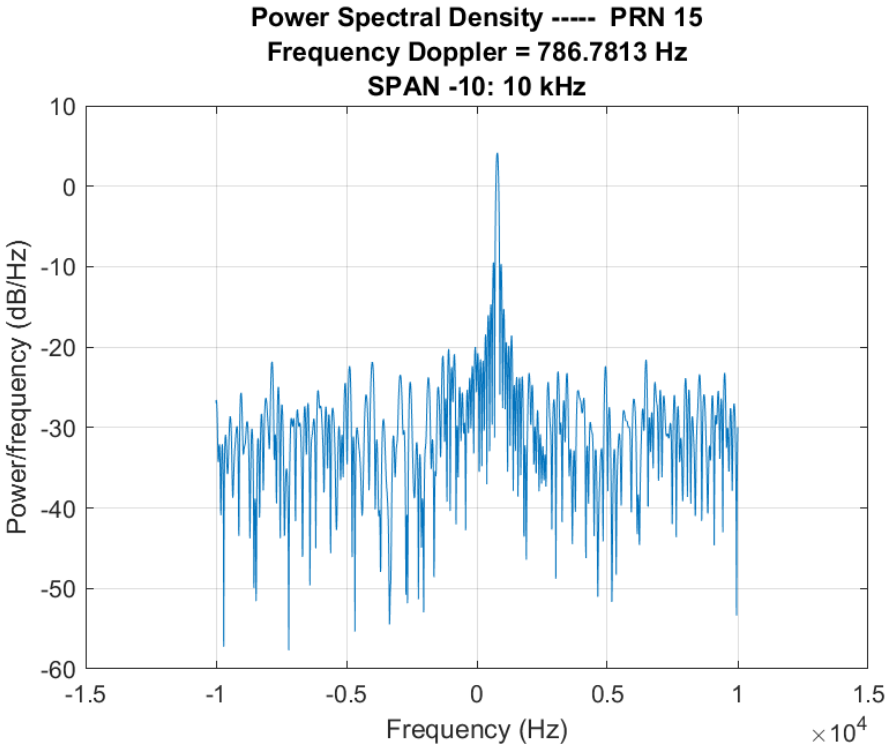

Ilustración 0-7. Adquisición fina. Señal sin filtrar: PRN 15; fd=786,7813 Hz

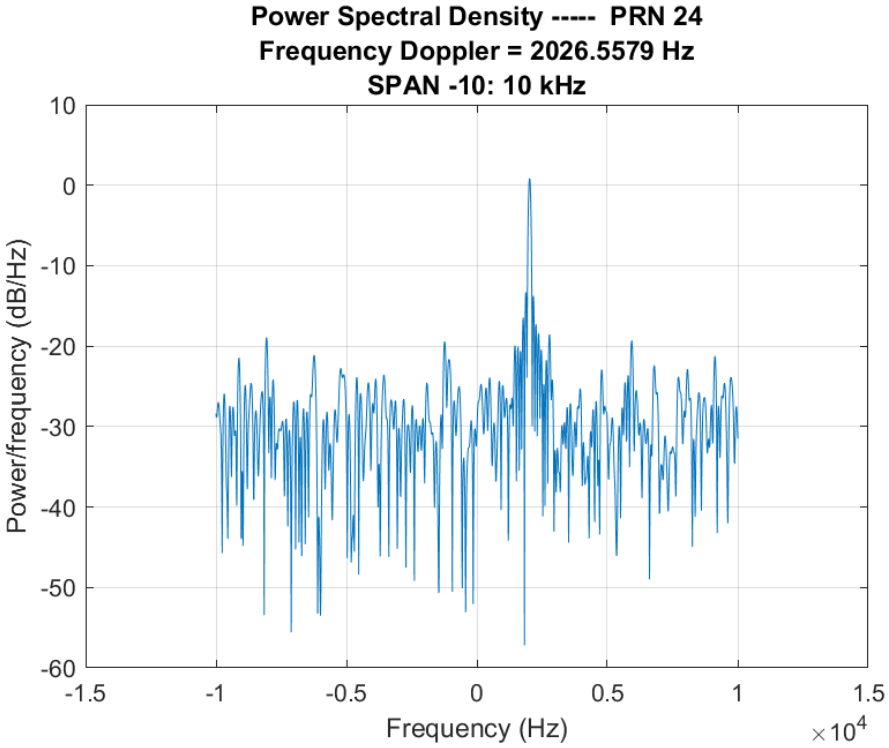

Ilustración 0-8. Adquisición fina. Señal sin filtrar: PRN 24; fd=2026,5579 Hz

Resultados de adquisición. Códigos PRN detectados

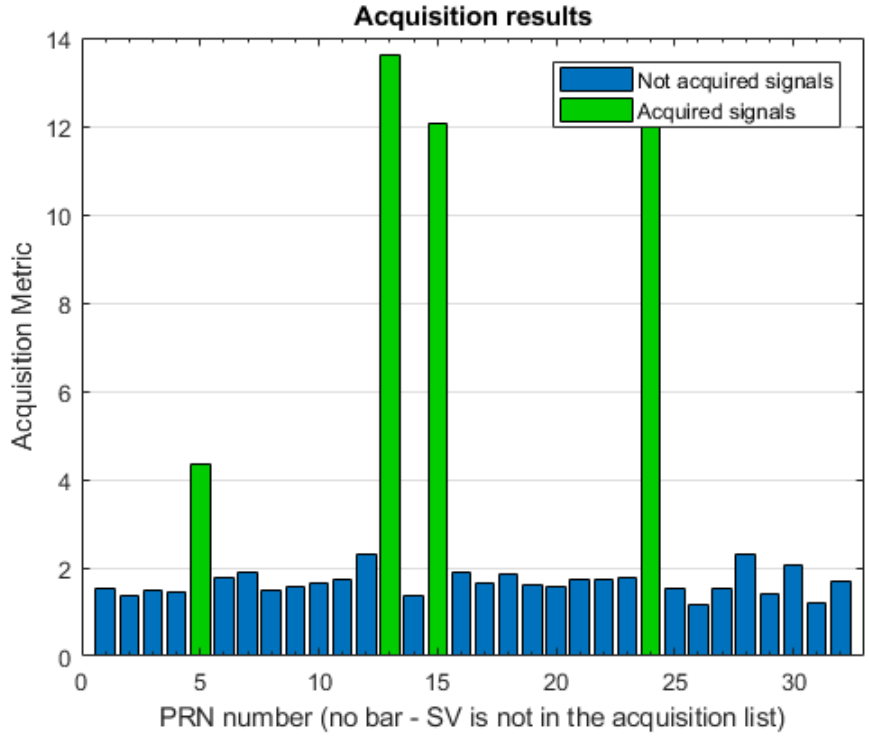

Ilustración 0-9. Resultados del bloque de Adquisición completo. Señal sin filtrar

Señal filtrada a 4 MHz

Adquisición gruesa

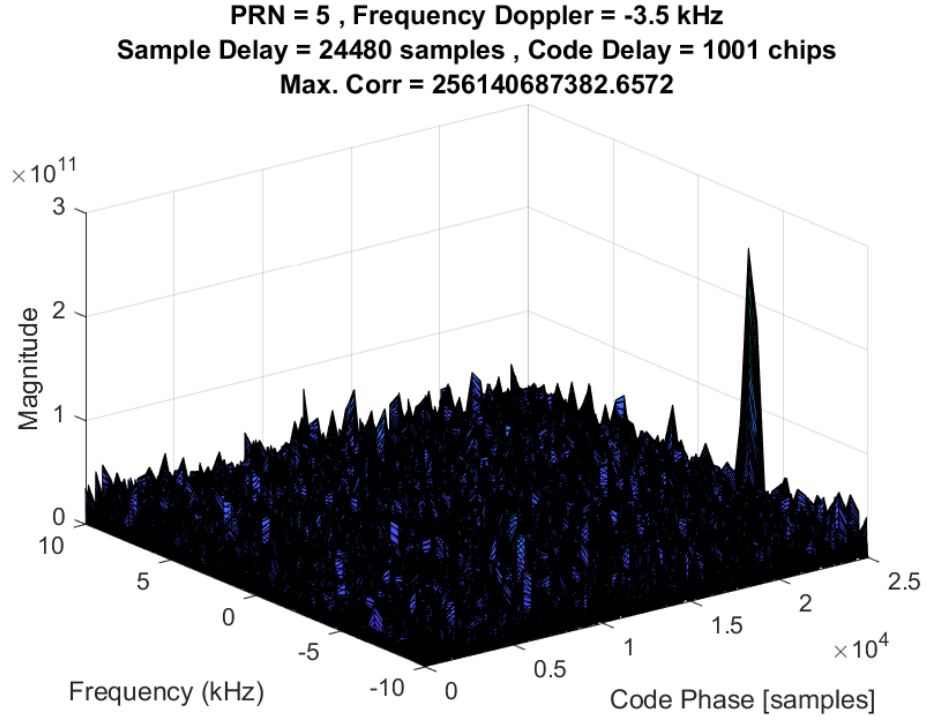

Ilustración 0-10. Adquisición gruesa. Señal filtrada 4 MHz: PRN 5; f_d=-3.5 kHz; τ=1001 chips

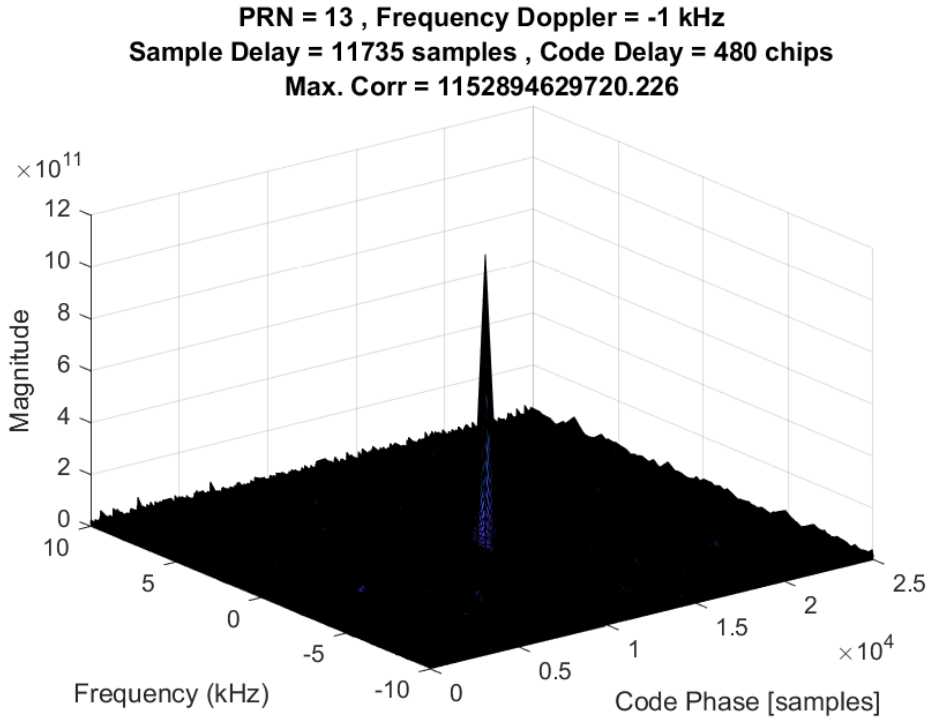

Ilustración 0-11. Adquisición gruesa. Señal filtrada 4 MHz: PRN 13; f_d=-1 kHz; τ=480 chips

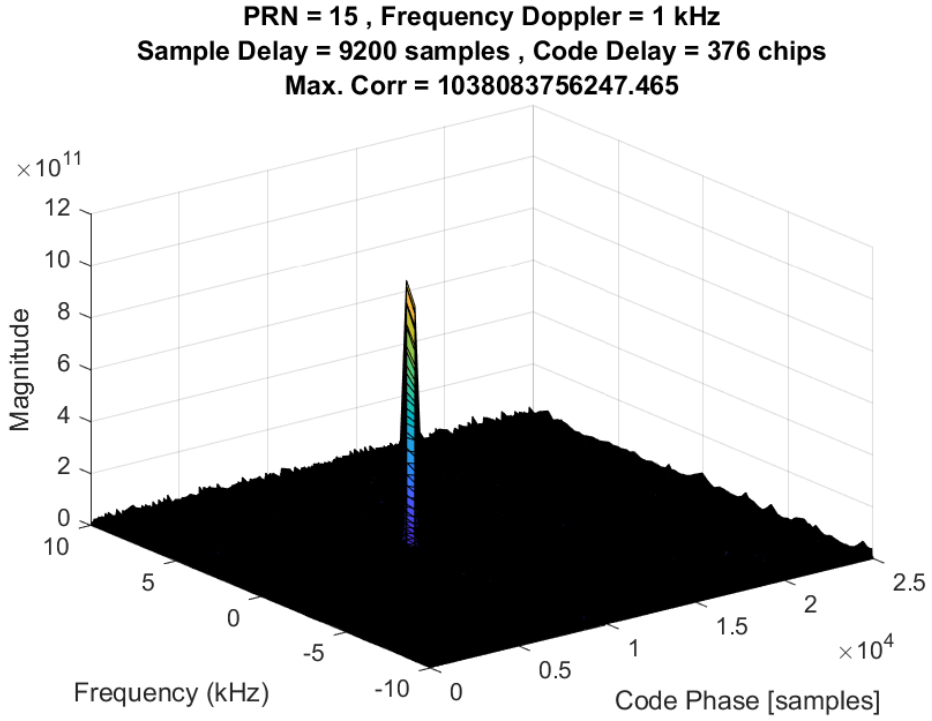

Ilustración 0-12. Adquisición gruesa. Señal filtrada 4 MHz: PRN 15; f_d=1 kHz; τ=376 chips

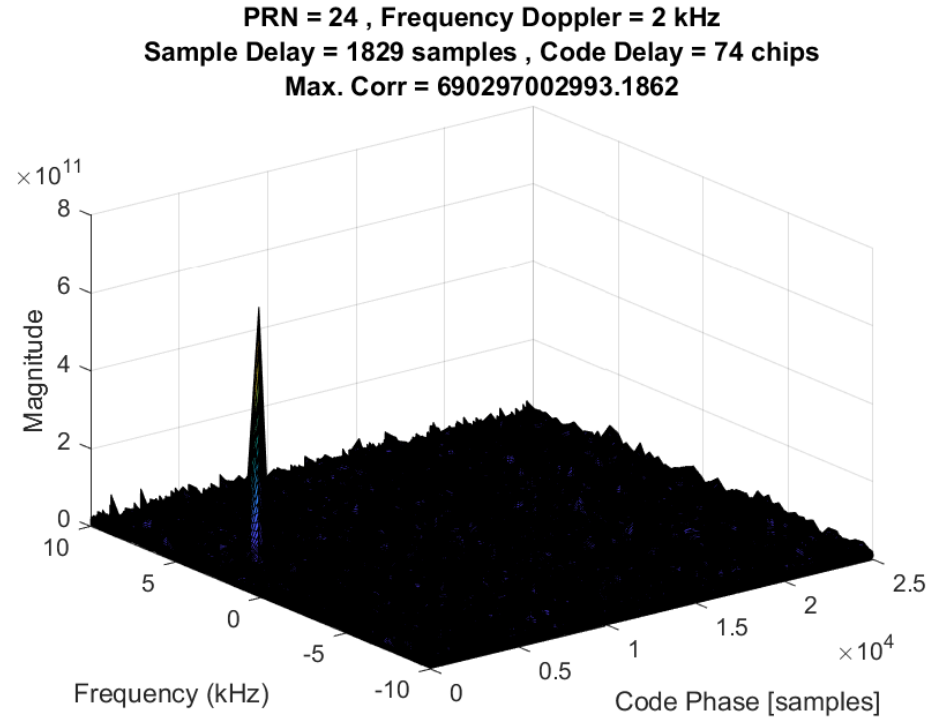

Ilustración 0-13. Adquisición gruesa. Señal filtrada 4 MHz: PRN 24; f_d=2 kHz; τ=74 chips

Adquisición fina

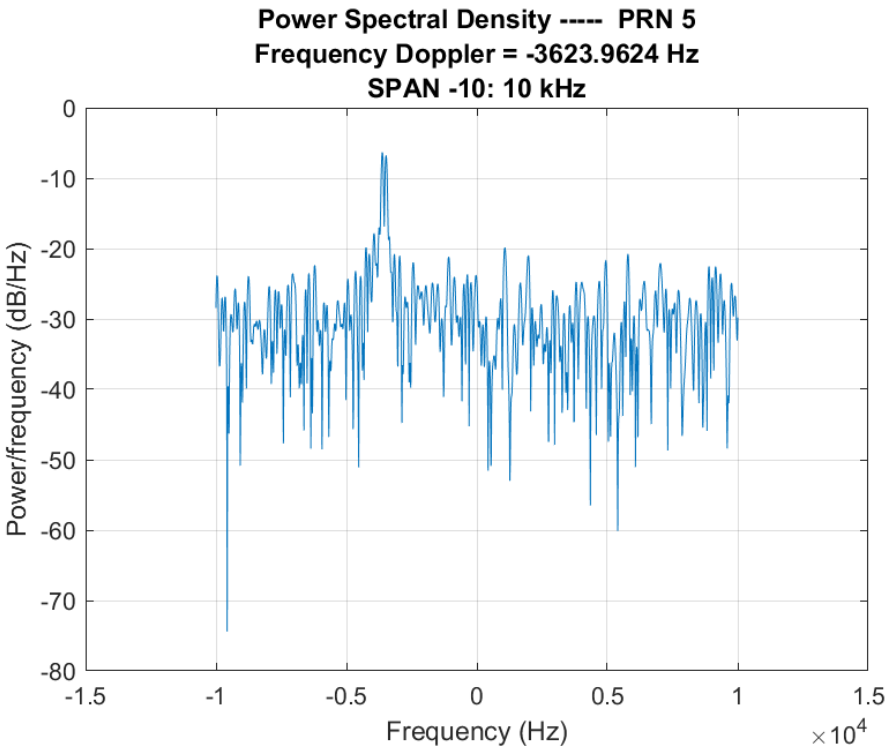

Ilustración 0-14. Adquisición fina. Señal filtrada 4MHz: PRN 5; f_d=-3623,9624 Hz

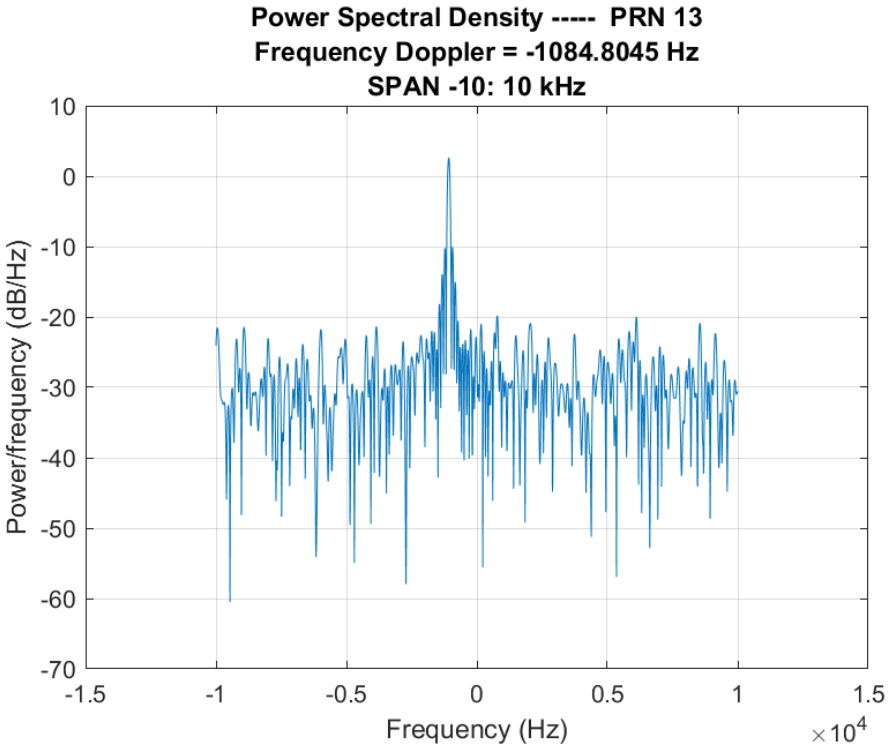

Ilustración 0-15. Adquisición fina. Señal filtrada 4MHz: PRN 13; f_d=-1084,8045 Hz

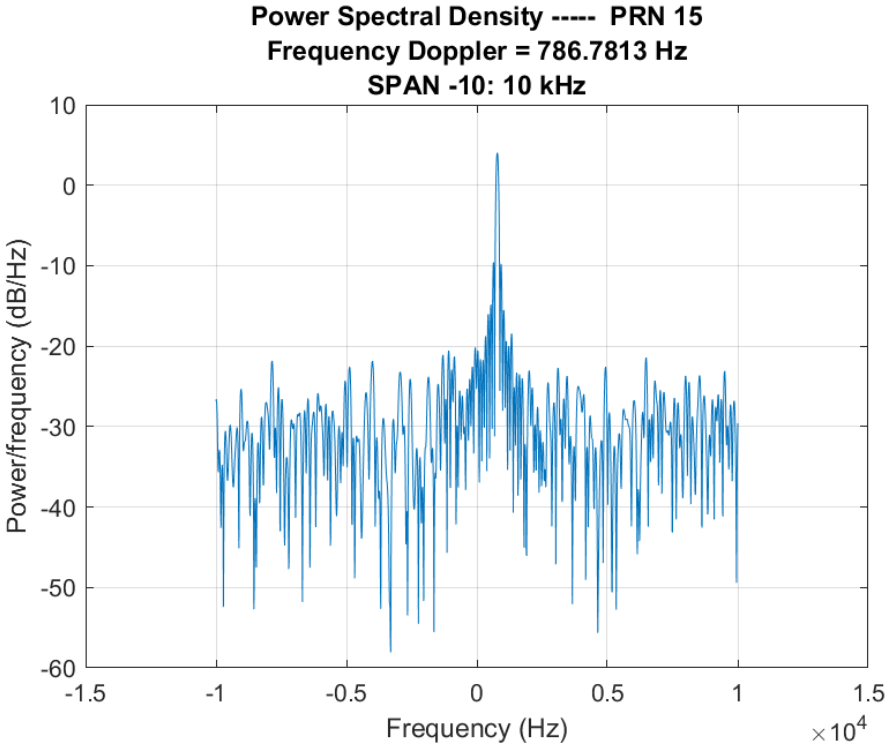

Ilustración 0-16. Adquisición fina. Señal filtrada 4 MHz: PRN 15; f_d=786,7813 Hz

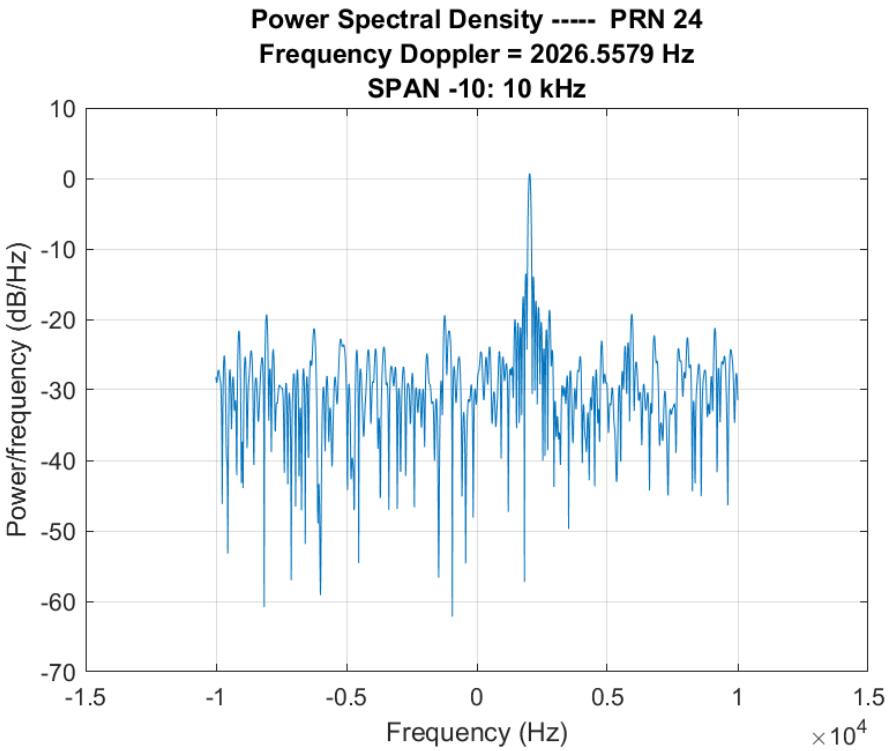

Ilustración 0-17. Adquisición fina. Señal filtrada 4 MHz: PRN 24; f_d=2026,5579 Hz

Resultados de adquisición. Códigos PRN detectados

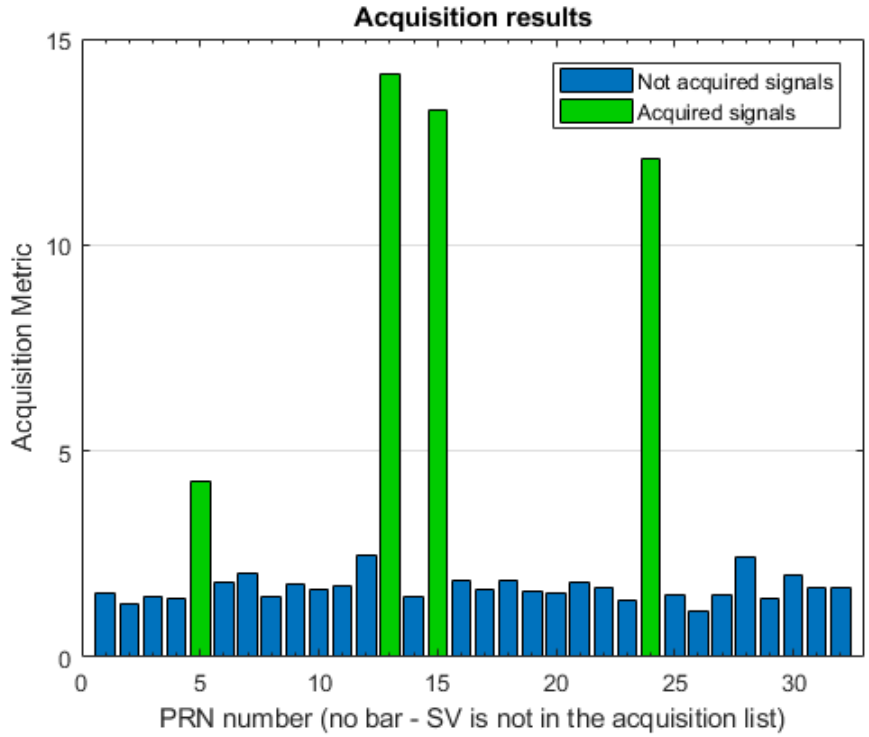

Ilustración 0-18. Resultados del bloque de Adquisición completo. Señal filtrada 4 MHz

Señal filtrada a 2 MHz

Adquisición gruesa

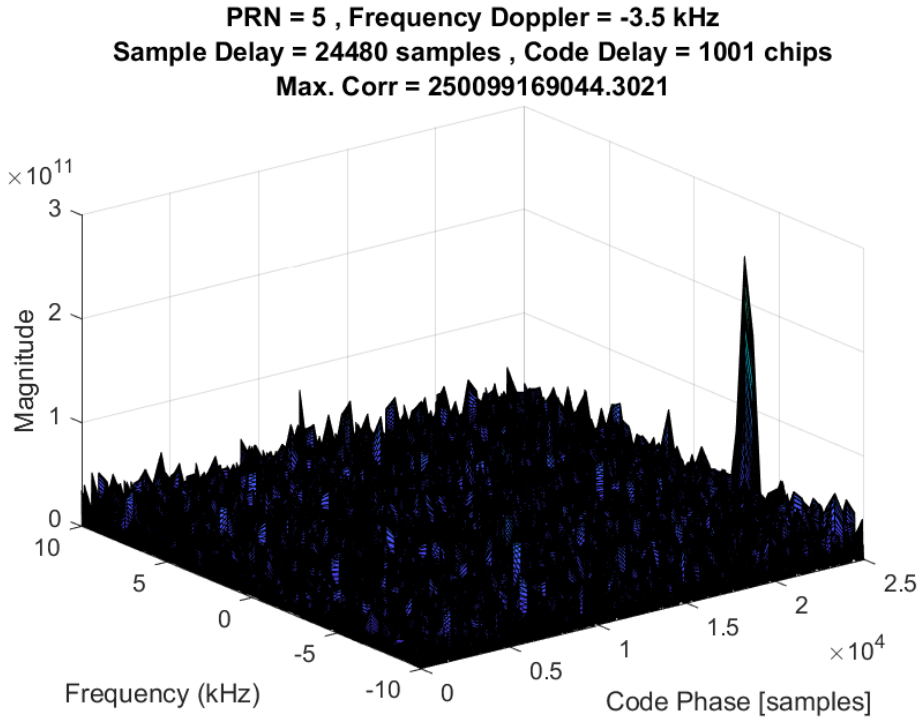

Ilustración 0-19. Adquisición gruesa. Señal filtrada 2 MHz: PRN 5; f_d=-3.5 kHz; τ=1001 chips

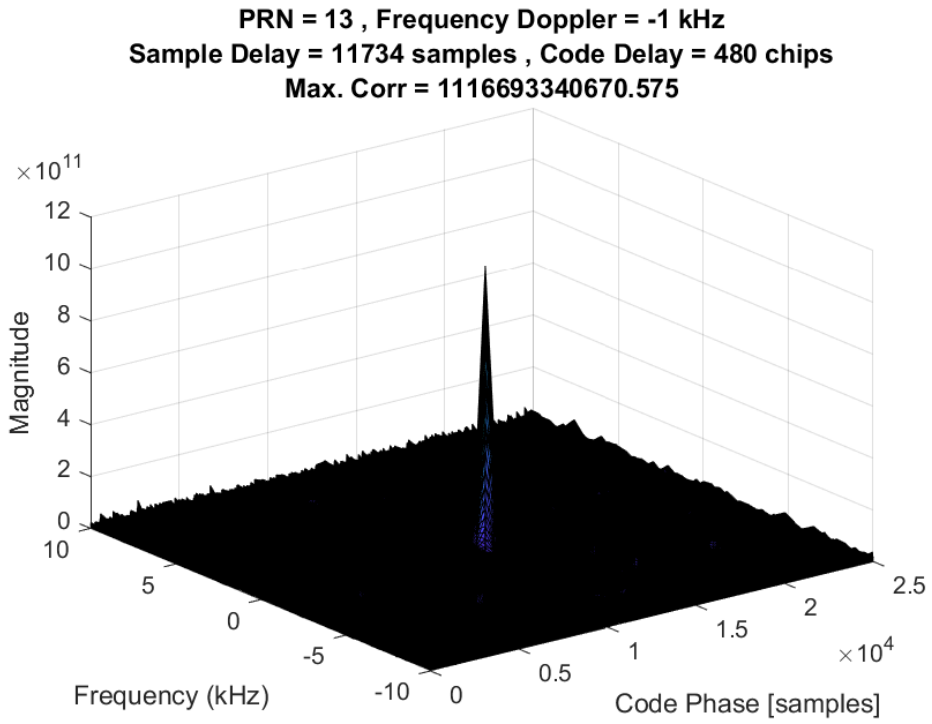

Ilustración 0-20. Adquisición gruesa. Señal filtrada 2 MHz: PRN 13; f_d=-1 kHz; τ=480 chips

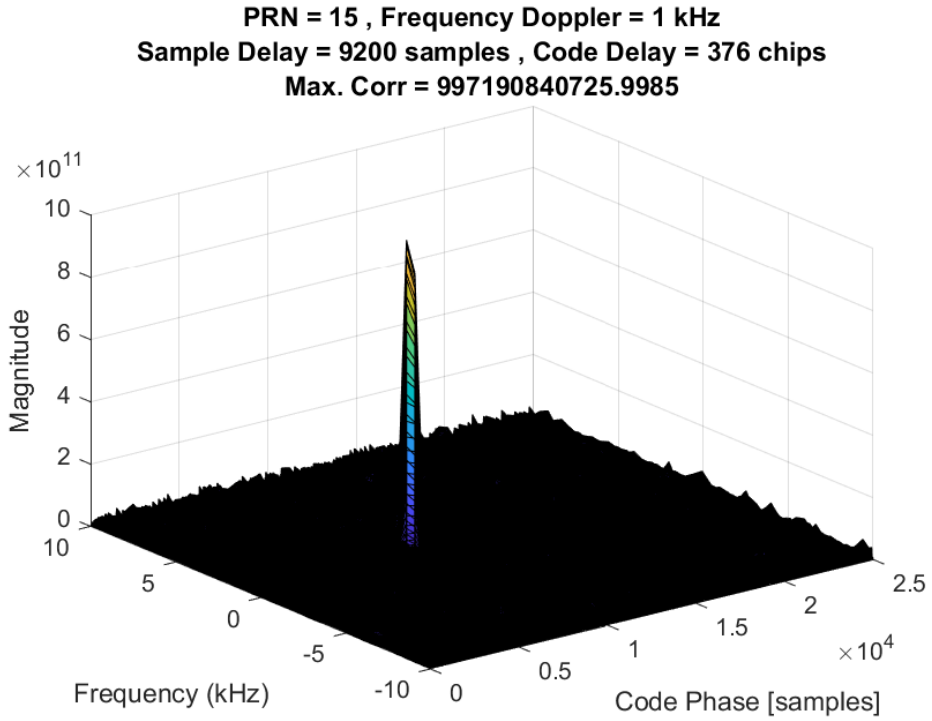

Ilustración 0-21. Adquisición gruesa. Señal filtrada 2 MHz: PRN 15; f_d=1 kHz; τ=376 chips

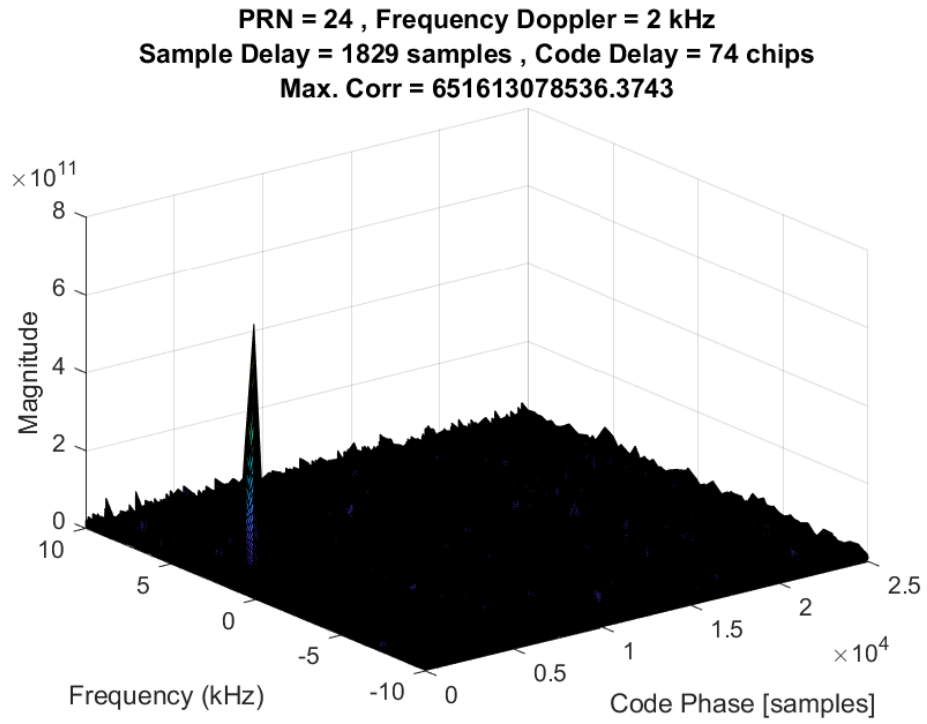

Ilustración 0-22. Adquisición gruesa. Señal filtrada 2 MHz: PRN 24; f_d=2 kHz; τ=74 chips

Adquisición fina

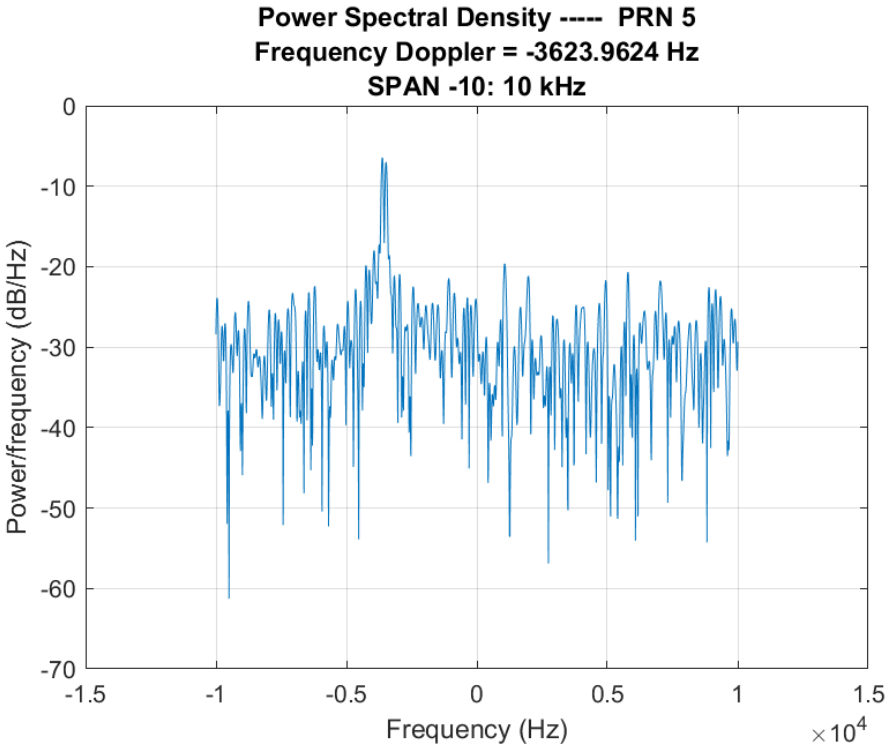

Ilustración 0-23. Adquisición fina. Señal filtrada 2 MHz: PRN; f_d=-3623,9624 Hz

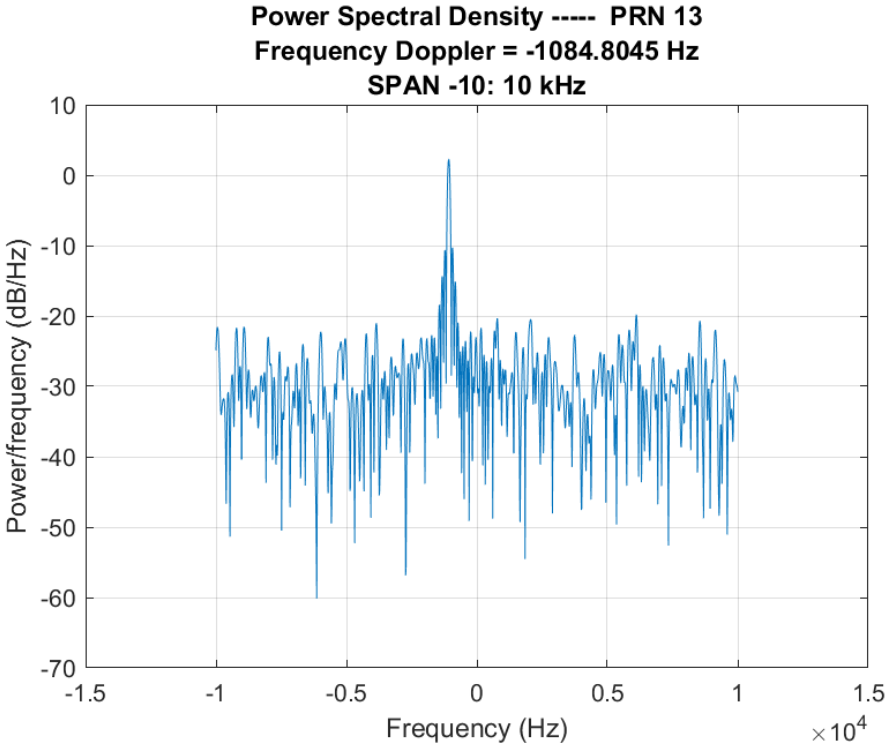

Ilustración 0-24. Adquisición fina. Señal filtrada 2 MHz: PRN 13; f_d=-1084,8045 Hz

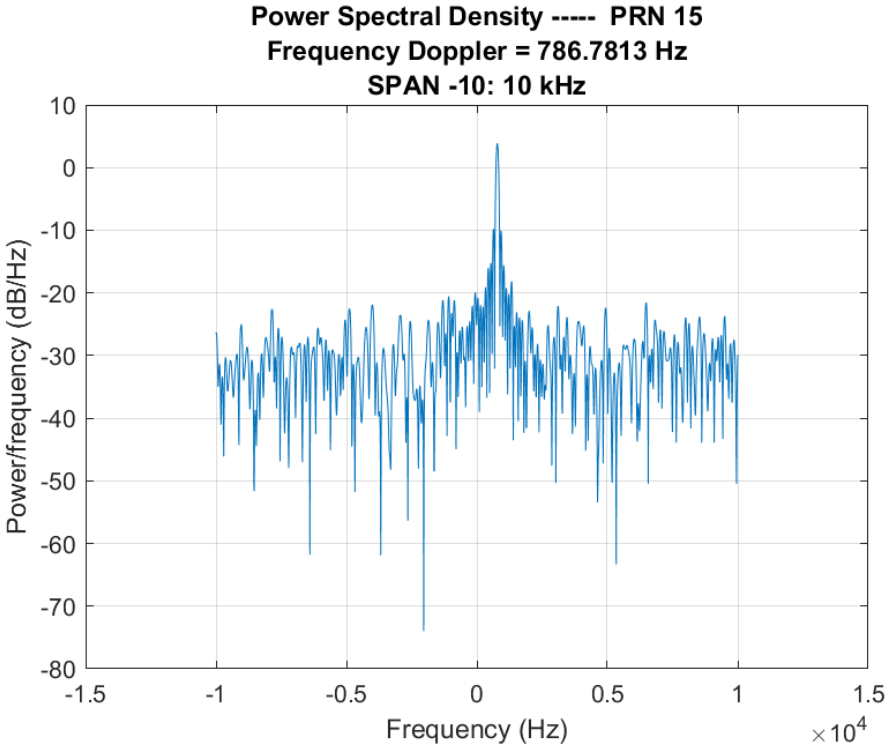

Ilustración 0-25. Adquisición fina. Señal filtrada 2 MHz: PRN 15; f_d=786,7813 Hz

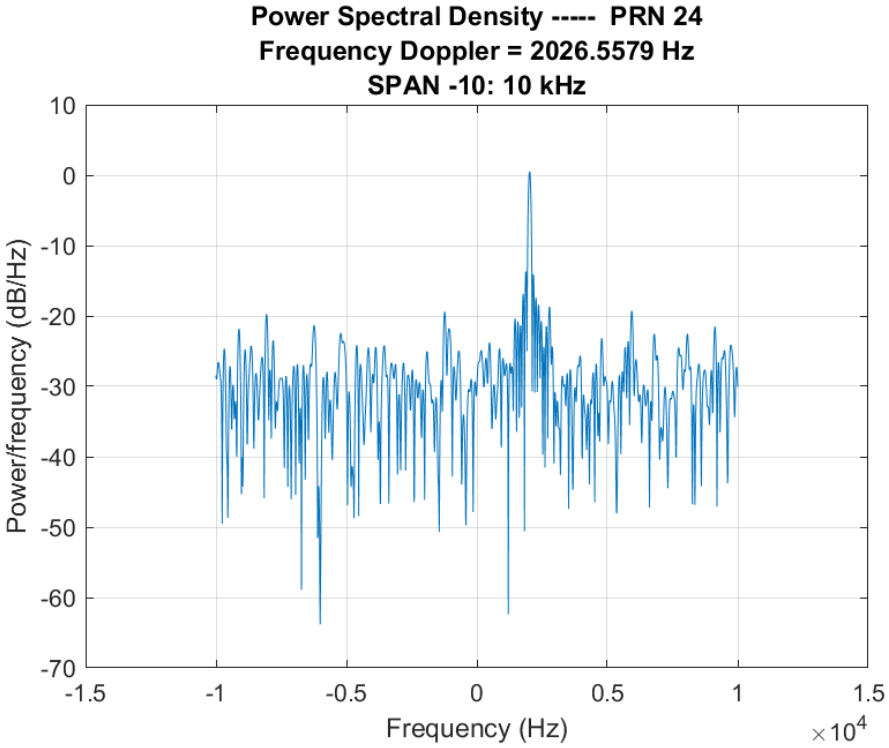

Ilustración 0-26. Adquisición fina. Señal filtrada 2 MHz: PRN 24; f_d=2026,5579 Hz

Resultados de adquisición. Códigos PRN detectados

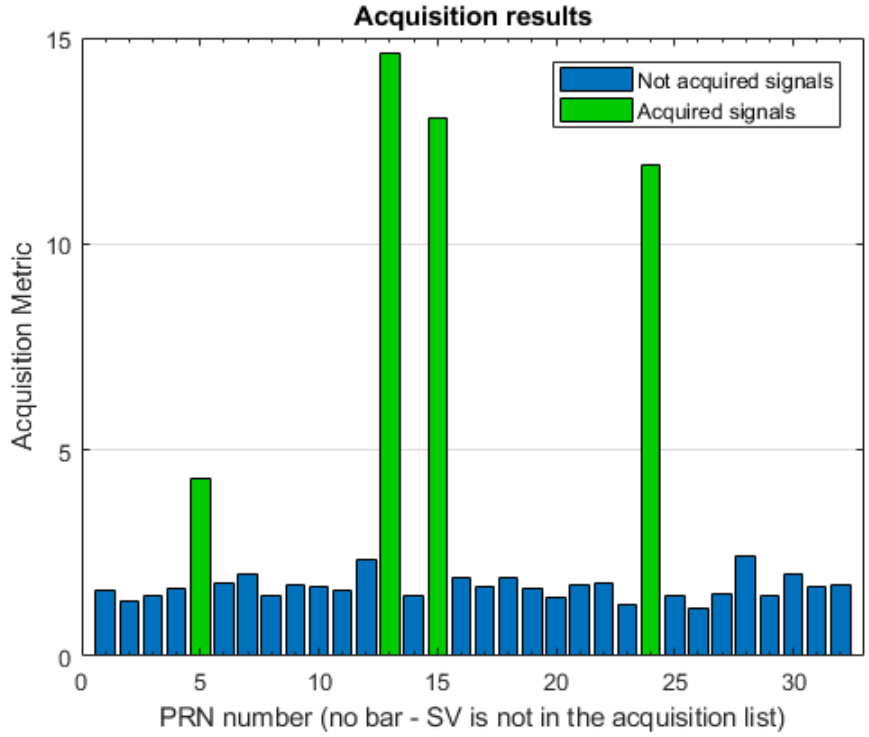

Ilustración 0-27. Resultados del bloque de Adquisición completo. Señal filtrada 2 MHz

Capturas del bloque de Tracking

Señal sin filtrar

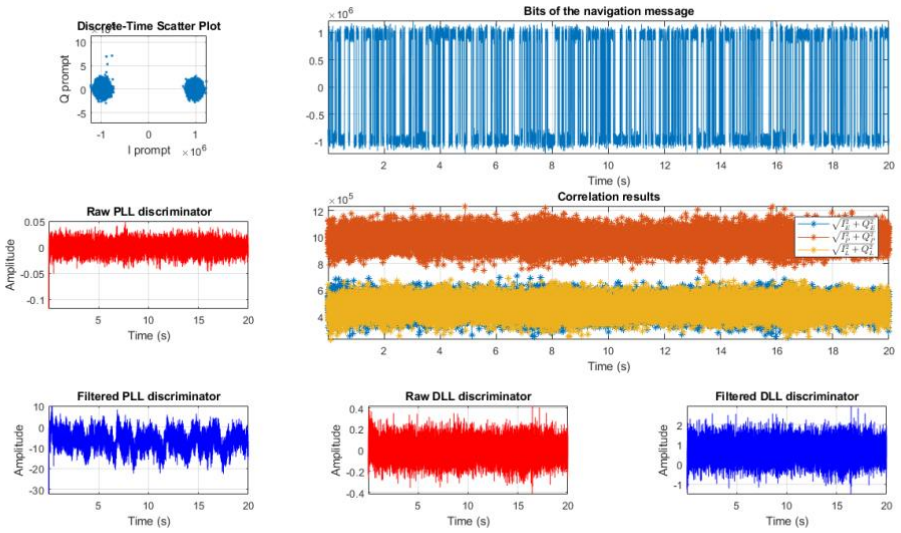

Ilustración 0-28. Tracking 20 s. Señal sin filtrar. PRN 5. Diagramas: (a) Discrete-Time Scatter plot; (b) Raw PLL Discriminator; (c) Filtered PLL Discriminator; (d) Raw DLL Discriminator; (e) Filtered DLL Discriminator; (f) Correlation Results; (g) Bits of the navigation message

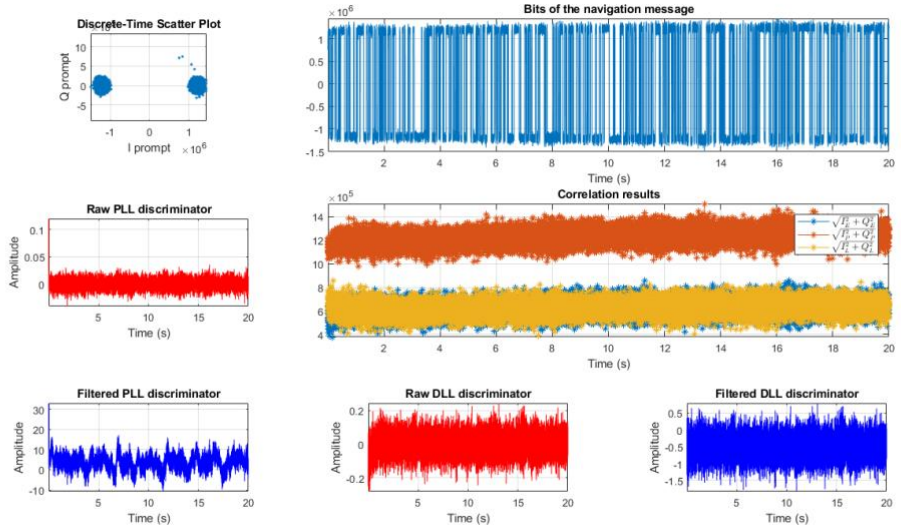

Ilustración 0-29. Tracking 20 s. Señal sin filtrar. PRN 13. Diagramas: (a) Discrete-Time Scatter plot; (b) Raw PLL Discriminator; (c) Filtered PLL Discriminator; (d) Raw DLL Discriminator; (e) Filtered DLL Discriminator; (f) Correlation Results; (g) Bits of the navigation message

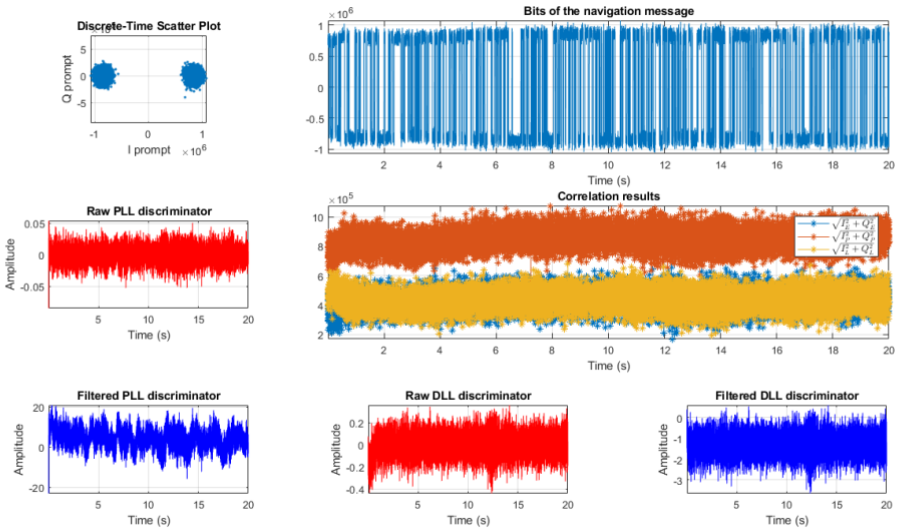

Ilustración 0-30. Tracking 20 s. Señal sin filtrar. PRN 15. Diagramas: Discrete-Time Scatter plot; Bits of the navigation message; Raw PLL Discriminator; Correlation Results; Filtered PLL Discriminator; Raw DLL Discriminator; Filtered DLL Discriminator

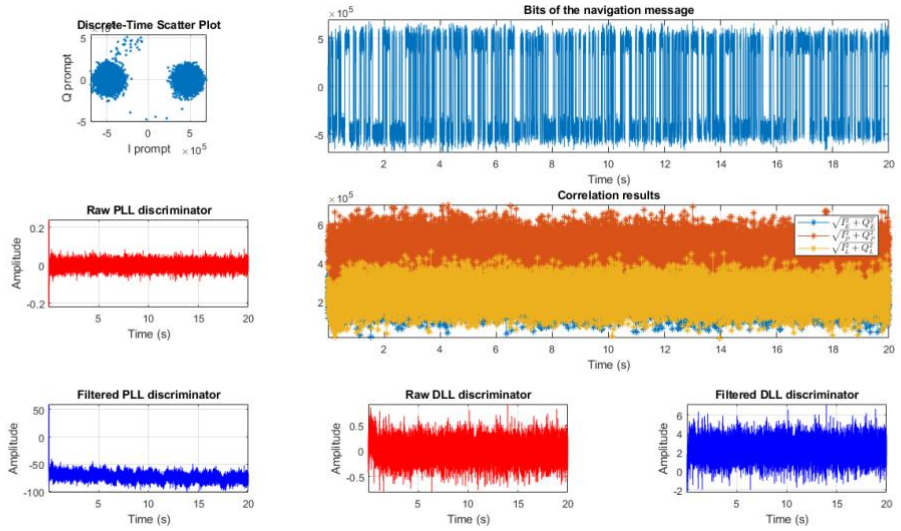

Ilustración 0-31. Tracking 20 s. Señal sin filtrar. PRN 24. Diagramas: (a) Discrete-Time Scatter plot; (b) Raw PLL Discriminator; (c) Filtered PLL Discriminator; (d) Raw DLL Discriminator; (e) Filtered DLL Discriminator; (f) Correlation Results; (g) Bits of the navigation message

Señal filtrada a 4 MHz

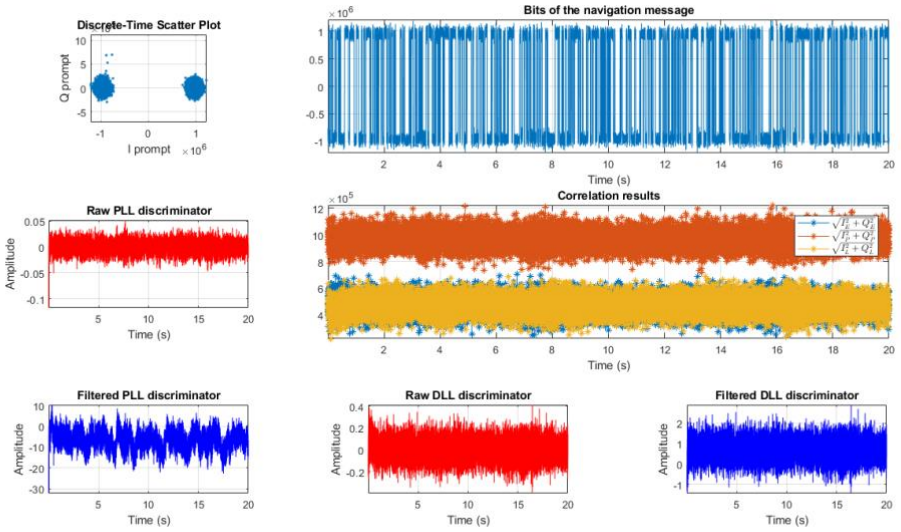

Ilustración 0-32. Tracking 20 s. Señal filtrada 4 MHz. PRN 5. Diagramas: (a) Discrete-Time Scatter plot; (b) Raw PLL Discriminator; (c) Filtered PLL Discriminator; (d) Raw DLL Discriminator; (e) Filtered DLL Discriminator; (f) Correlation Results; (g) Bits of the navigation message

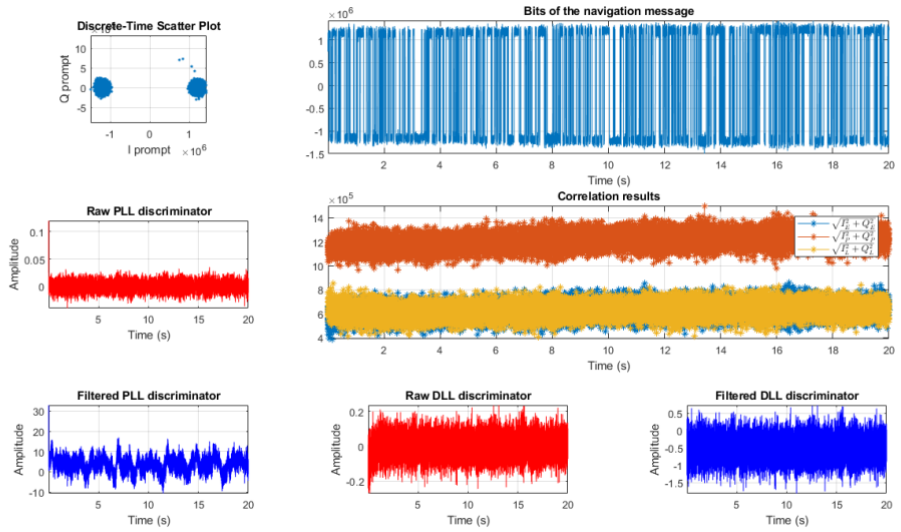

Ilustración 0-33. Tracking 20 s. Señal filtrada 4 MHz. PRN 13. Diagramas: (a) Discrete-Time Scatter plot; (b) Raw PLL Discriminator; (c) Filtered PLL Discriminator; (d) Raw DLL Discriminator; (e) Filtered DLL Discriminator; (f) Correlation Results; (g) Bits of the navigation message

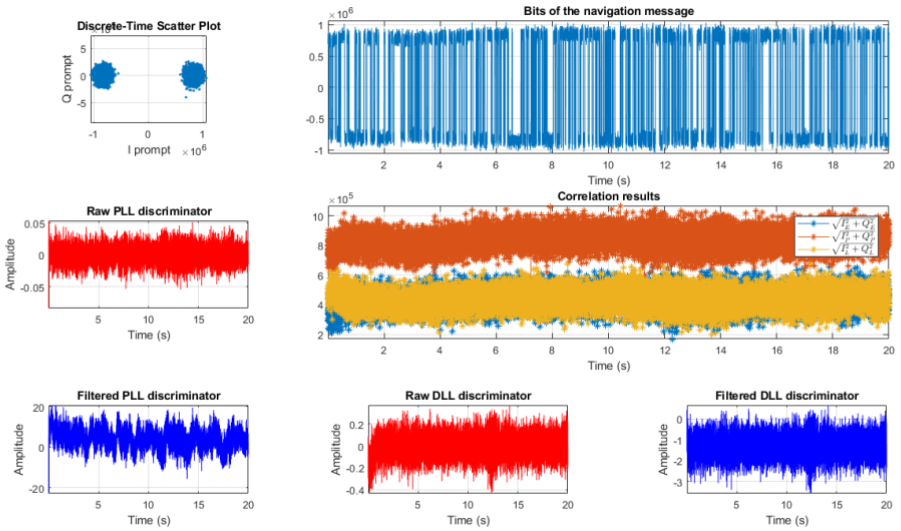

Ilustración 0-34. Tracking 20 s. Señal filtrada 4 MHz. PRN 15. Diagramas: (a) Discrete-Time Scatter plot; (b) Raw PLL Discriminator; (c) Filtered PLL Discriminator; (d) Raw DLL Discriminator; (e) Filtered DLL Discriminator; (f) Correlation Results; (g) Bits of the navigation message

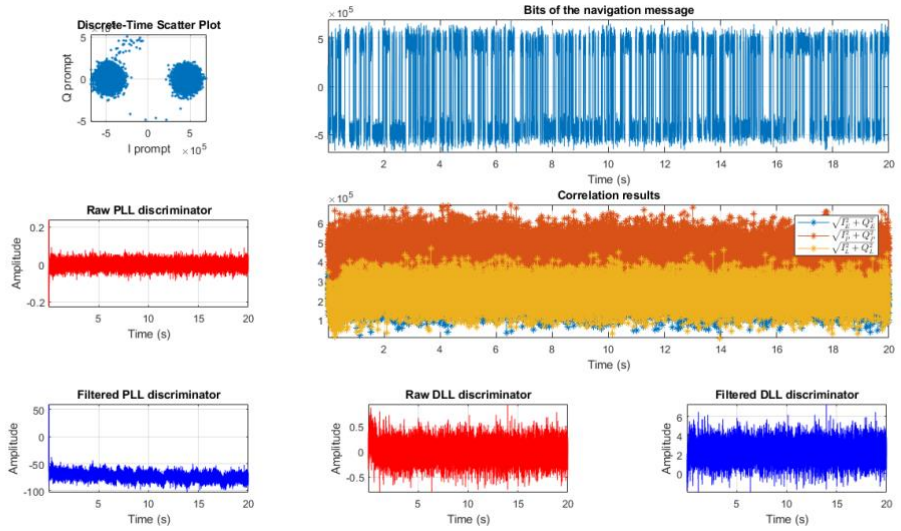

Ilustración 0-35. Tracking. Tracking 20 s. Señal filtrada 4 MHz. PRN 24. Diagramas: (a) Discrete-Time Scatter plot; (b) Raw PLL Discriminator; (c) Filtered PLL Discriminator; (d) Raw DLL Discriminator; (e) Filtered DLL Discriminator; (f) Correlation Results; (g) Bits of the navigation message

Señal filtrada a 2 MHz

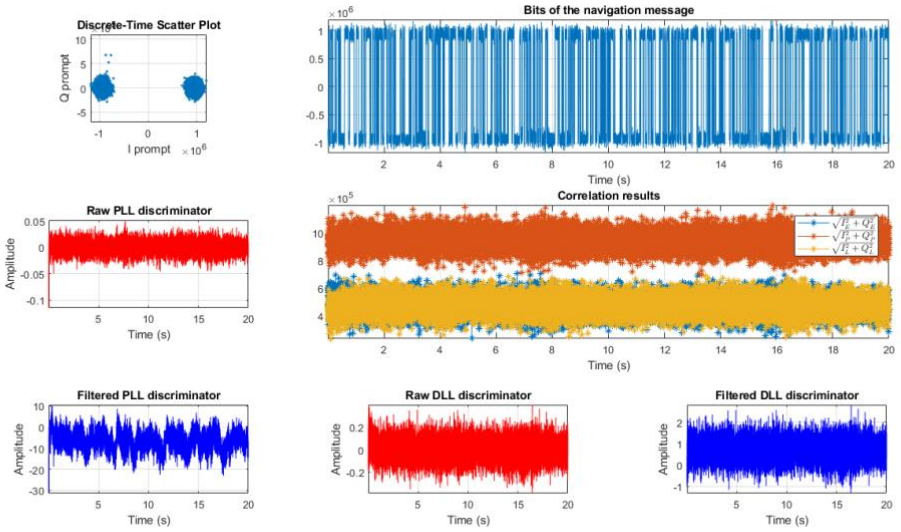

Ilustración 0-36. Tracking 20 s. Señal filtrada 2 MHz. PRN 5. Diagramas: (a) Discrete-Time Scatter plot; (b) Raw PLL Discriminator; (c) Filtered PLL Discriminator; (d) Raw DLL Discriminator; (e) Filtered DLL Discriminator; (f) Correlation Results; (g) Bits of the navigation message

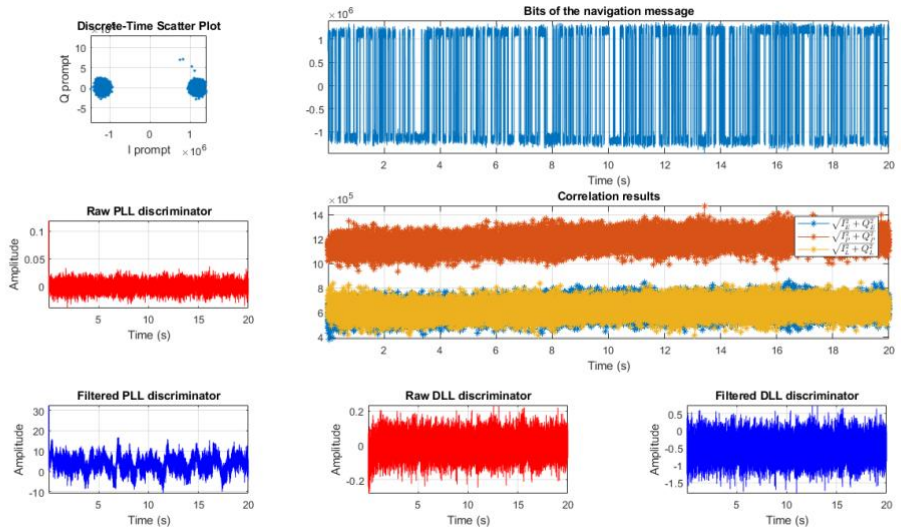

Ilustración 0-37. Tracking 20 s. Señal filtrada 2 MHz. PRN 13. Diagramas: (a) Discrete-Time Scatter plot; (b) Raw PLL Discriminator; (c) Filtered PLL Discriminator; (d) Raw DLL Discriminator; (e) Filtered DLL Discriminator; (f) Correlation Results; (g) Bits of the navigation message

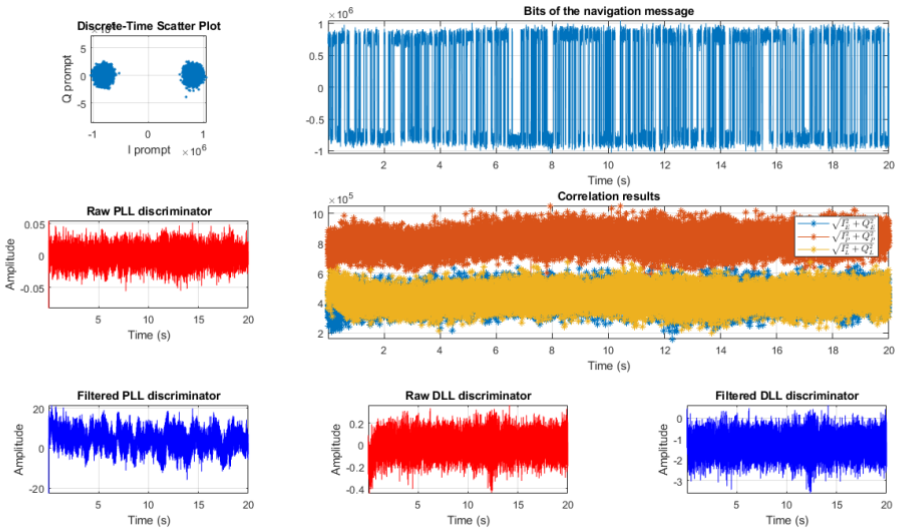

Ilustración 0-38 Tracking 20 s. Señal filtrada 2 MHz. PRN 15. Diagramas: (a) Discrete-Time Scatter plot; (b) Raw PLL Discriminator; (c) Filtered PLL Discriminator; (d) Raw DLL Discriminator; (e) Filtered DLL Discriminator; (f) Correlation Results; (g) Bits of the navigation message

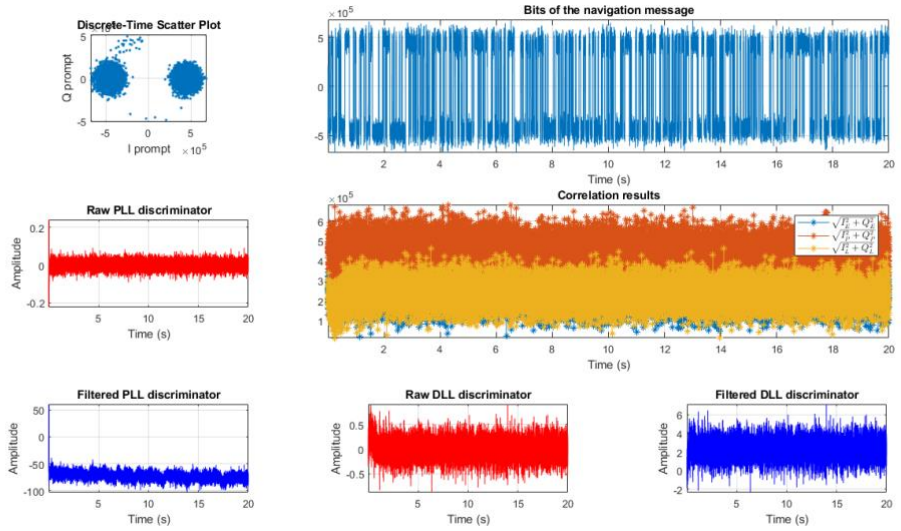

Ilustración 0-39. Tracking 20 s. Señal filtrada 2 MHz. PRN 24. Diagramas: (a) Discrete-Time Scatter plot; (b) Raw PLL Discriminator; (c) Filtered PLL Discriminator; (d) Raw DLL Discriminator; (e) Filtered DLL Discriminator; (f) Correlation Results; (g) Bits of the navigation message

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá