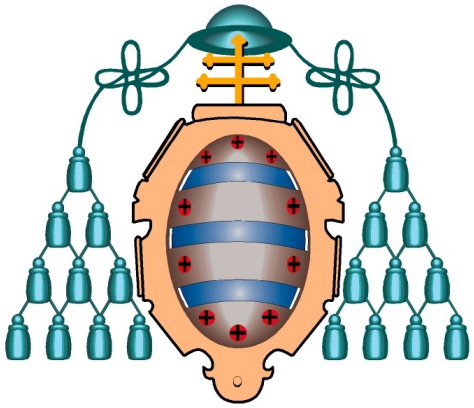


UNIVERSIDAD DE OVIEDO

Máster en Modelización Matemática, Estadística y Computación

Trabajo Fin de Máster

Filtro IMM para Sistema de Vigilancia Aeroportuaria A-SMGCS

Autora:
Vanesa Ibáñez Llanos

Directores:
César Luis Alonso González
Marta Campo Varela

26 de Julio de 2012

Dedicado a mi abuelo Hernán ...
a quién le hubiese encantado formar parte de esto

Agradecimientos

Primeramente quiero agradecer de todo corazón a la gente de Indra Software Labs Gijón que, sin conocerme, confiasen en mí y me diesen la oportunidad de introducirme un tema tan interesante y realizar un trabajo que me ha entusiasmado: Ramón, Marcelino, Marcos y Venancio, pero sobre todo a Marta, siempre amable, que ha estado disponible en todo momento y me ha dedicado un tiempo que ni tenía.

Me gustaría también mostrar mi agradecimiento a César L. Alonso por ofrecerme su tutela, por su guía y sus muy buenos consejos.

A mis padres, hermanos, tíos, primos, amigos y demás familia, que siempre han creído en mí, incluso más que yo misma, su cariño y apoyo. En especial a Jose, Moni y Alba por acogerme durante el periodo Zaragoza y hacerme sentir en casa, a Sara y César por esas risas en momentos de estrés, a Fermín, fuente inagotable de sabiduría y siempre dispuesto a compartirla y a mi querido Rubén por estar siempre ahí.

And above all, I am most grateful to my hubby James for his patience and for helping, supporting, encouraging and pushing me to pursue my dreams, and without whom I wouldn't have been able to undertake this crazy adventure let alone finish it.

Índice general

Índice de figuras

Índice de cuadros

Introducción

El proyecto surge ante la necesidad de mejorar los sistemas de vigilancia en la superficie de los aeropuertos europeos dentro del ámbito del Espacio Único Europeo. Los nuevos sistemas pretenden integrar la información aportada por todos los sensores instalados en el aeródromo para hacer seguimiento de los vehículos de forma muy precisa. Esto requiere cambios entre los que están la utilización de filtros de estimación de trayectorias a partir de datos que pueden ser de diversa naturaleza (posición o posición y velocidad en coordenadas cartesianas, posición en coordenadas polares,...), procedentes de distintos tipos de sensores con características específicas.

En los estudios técnicos y teóricos llevados a cabo en el marco del Working Package 12 del *Single European Sky ATM Research* (SESAR) [5] sobre los cambios que hay que hacer en los sistemas actuales para consolidar el nivel 2 de A-SMGCS se proponen los tipos de filtros de tracking que se consideran adecuados para estimar la trayectoria de vehículos circulando en los aeropuertos: *Filtro de Kalman* (KF), *Múltiples Modelos Interactuantes* (IMM) de estructura fija o variable y *filtros de partículas* (con y sin incluir información de mapa del aeropuerto).

En este trabajo de fin de máster se diseñan, implementan y evalúan filtros, basados en las técnicas de Kalman, para estimar/suavizar la trayectoria de objetos en movimiento en el área visible desde la torre de control de los aeropuertos, a partir de las observaciones aportadas por radares de superficie y multilateración. Teniendo en cuenta los distintos tipos de vehículos que pueden circular por el aeropuerto (camiones, autobuses, aviones,...) se han simulado varios escenarios con sus trayectorias, movimientos y maniobras típicos, y los sensores que los detectan, e implementado cuatro configuraciones de filtro IMM, compuestos por filtros Kalman ajustados a varios modelos de movimiento. El comportamiento de los filtros diseñados se estudia comparándolo con la técnica utilizada en los sistemas desplegados actualmente en las torres de control para ver si suponen una mejora a la hora de localizar los vehículos circulando por el aeropuerto.

El capítulo 1 resume el fondo histórico y técnico relevante para este proyecto. En particular, incluye una visión general del proyecto internacional de múltiples fases que establece las bases para acometer una mejora de los sistemas de control de tráfico aéreo europeos. Esto nos permite caracterizar de forma clara el ámbito y los objetivos de este trabajo de fin de máster, el cual se engloba dentro del área relacionada con la investigación y mejora del seguimiento de vehículos en superficie.

El capítulo 2 introduce y explica las herramientas matemáticas de interés para el proyecto. Éstas son los *Modelos Dinámicos*, utilizados en el capítulo 3 para la simulación de trayectorias de vehículos en los aeropuertos, y los *Filtros de Estimación* (Kalman e IMM), de los que se valorará más tarde, cuál es su utilidad en la mejora de la precisión del seguimiento de vehículos en superficie cuando se procesan varias señales de radar simultáneamente.

En el capitulo 3 desarrollamos los modelos dinámicos utilizados para describir los distintos movimientos de vehículos que pueden observarse en aeropuertos reales. Para ello tenemos en cuenta datos de trayectorias reales de los aeropuertos de Palma, Barcelona y Barajas. Para completar el entorno de simulación también se elabora un modelo para los radares.

Esto nos lleva directamente al capítulo 4, donde, de nuevo con la ayuda de datos reales y documentación técnica para los tipos de radar específicos que se utilizan, detallamos el funcionamiento de nuestro sistema de simulación.

En el capítulo 5 se proponen, describen y configuran varias estructuras de filtros IMM para utilizarlos en la simulación.

En el capítulo 6 presentamos, por medio de gráficos intuitivos e informativos, los resultados obtenidos al aplicar los distintos filtros IMM diseñados a una variedad de escenarios que representan diverso combinaciones de configuraciones de sensores y trayectorias de vehículos, y ofrecemos una interpretación general de estos resultados.

Tras el análisis de estos resultados extraemos conclusiones acerca de los beneficios que ofrece utilizar filtros IMM sobre el método de seguimiento actual basado en prioridades de radar, y las presentamos en el capitulo 7.

En los Apéndices se incluye:

- El código fuente en Matlab implementado y utilizado para todos los elementos de la simulación y filtrado
- Tablas que proporcionan en detalle los parámetros numéricos que describen las trayectorias simuladas.
- Tablas adicionales de resultados, resumen comparativo de los filtros probados.
- Ejemplo de tablas de datos de detecciones de radares reales, extraídos del sistema de radar instalado en las dependencias aeroportuarias.

Capítulo 1

Antecedentes, Motivación y Características del Trabajo

La seguridad en las operaciones que se realizan en los aeropuertos es uno de los principales focos de atención de la *Organización de Aviación Civil Internacional* (OACI), habiéndose convertido en un elemento fundamental en la gestión del tráfico en los aeropuertos la regulación de los procesos de detección, gestión de la información y mejora continua de los estándares de seguridad. El procesamiento de los datos de los sensores es un elemento clave en los centros de *Control de Tránsito Aéreo* (ATC) y ha de actualizarse para adaptarse a la evolución de los sensores y a las características particulares de control.

1.1. Control del Espacio Aéreo en Europa

1.1.1. Gestión de Tráfico Aéreo

La *Gestión del Tránsito Aéreo* (ATM), concepto inherente a la existencia del movimiento de aeronaves, incluye todas las tareas necesarias para garantizar la seguridad y el flujo regular del tráfico aéreo, lo que comprende tres servicios principales [3]:

***Control de Tránsito Aéreo* (ATC)** cuya labor es distribuir el Espacio Aéreo (que en el caso de un país abarca su territorio, desde la superficie hasta una altitud ilimitada) y separar Aeronaves (entre sí y con los obstáculos en tierra) que pretenden utilizarlo, a fin de evitar colisiones tanto en aire como en tierra. Este servicio se realiza, principalmente, por *Controladores* ubicados en los *Centros de Operaciones* y *Torres de Aeródromos* comunicados entre sí y con los *Pilotos* de las aeronaves.

***Gestión del Flujo del Tránsito Aéreo* (ATFM)** su objetivo de contribuir a una circulación segura y ordenada del tránsito aéreo asegurando que se usa al máximo posible la capacidad ATC, y que el volumen de tránsito es compatible con las capacidades declaradas por la autoridad *Servicios de Tráfico Aéreo* (ATS) competente.

***Gestión del Espacio Aéreo* (ASM)** cuyo fin es administrar lo mejor posible el espacio aéreo, un recurso que escasea, de forma que se puedan satisfacer las necesidades de todos sus usuarios, tanto civiles como militares.

Las prioridades de un sistema de ATM son garantizar la correcta organización y la seguridad del tráfico en el espacio aéreo y aeropuertos.

1.1.2. Hacia el Cielo Único Europeo

La *Gestión del Tránsito Aéreo* (ATM) en la Eurozona consiste en un sistema muy complejo (gestiona más de 35000 vuelos comerciales diarios). A diferencia de EEUU, dispone de un cielo fragmentado (dividido en 27 espacios aéreos) en el que la gestión del tráfico aéreo se realiza a nivel nacional.

El espacio aéreo europeo se caracteriza por ser uno de los más densos del mundo, cada vez más saturado y contaminado, y con aeropuertos, de infraestructuras limitadas, muy concurridos. Pese a haberse ralentizado en los últimos tiempos, la previsión de la demanda del transporte aéreo sigue siendo de crecimiento.

El sistema de ATM existente no puede hacer frente a las demandas crecientes de este sector de una manera eficiente por lo que en 1999 la Comisión Europea lanza la iniciativa de un *Cielo Único Europeo (Single European Sky)* (SES), cuyo objetivo es reorganizar el cielo europeo en función del flujo de tráfico y no de las fronteras nacionales, estableciendo reglas técnicas y de procedimiento comunes y promoviendo el desarrollo de un sistema europeo unificado de Gestión del Tránsito Aéreo.

Se espera que la iniciativa SES resulte en la mejora en la seguridad de los aeropuertos y del rendimiento medioambiental de la aviación, en un aumento de la capacidad del espacio aéreo europeo y el incremento de la eficiencia y flexibilidad del sistema de ATM Europeo.

Programa SESAR

Single European Sky ATM Research (SESAR) es un proyecto conjunto de la comunidad de transporte aéreo europea (militares y civiles, legisladores, industria, operadores, usuarios..) [12] cuyo objetivo es modernizar y optimizar la *Gestión del Tránsito Aéreo* (ATM) mediante el desarrollo e implantación, para 2020, de una red ATM europea de altas prestaciones, homogeneizando el tráfico aéreo en la unión europea, que sea capaz de garantizar la seguridad y fluidez del transporte aéreo en Europa en los próximos 30 años.

Se compone de tres fases:

- *Fase de Definición (2006-2008)*: consistió en el desarrollo de las bases del Cielo Único y en la elaboración de un Plan Maestro Europeo de ATM
- *Fase de Desarrollo (2009-2016)*: fase en la que se encuentra actualmente y en la que se están ejecutando las tareas de investigación, desarrollo y validación recogidas en el Plan Maestro ATM.
- *Fase de Despliegue (2014-2020)*: durante la cual se implantarán, de forma progresiva, las soluciones operativas y sus componentes técnicos, identificados en el Plan Maestro Europeo.

El programa comprende 16 paquetes de trabajo, uno de los cuales, el WP12, engloba todas las actividades de I+D para definir, especificar y validar los Sistemas Aeroportuarios necesarios respaldar el concepto objetivo del ATM del SESAR. En particular, el proyecto 12.3.1 se centra en **mejorar la gestión de la superficie de los aeropuertos** (incluyendo técnicas de vigilancia avanzadas A-SMGCS, sistemas de guía y en-rutamiento en tierra, redes de seguridad...).

1.1.3. Mejora del Control de Tráfico en los Aeródromos: A-SMGCS

El *Sistema Avanzado de Guía y Control del Movimiento en la Superficie* (A-SMGCS) es un paradigma para la **integración de la información de vigilancia de aeropuerto**, con finalidad de mejorar sus condiciones operativas, en especial en condiciones de visibilidad reducida,

operaciones nocturnas y en general en zonas que no se puedan visualizar bien desde las torres de control. Este tipo de sistemas puede resolver, en potencia, los cuellos de botella en la capacidad del aeropuerto manteniendo o mejorando los niveles de seguridad actuales. Ayudan al controlador de torre, ya que detectan situaciones de peligro, y ofrecen un abanico de funcionalidades como alarmas y etiquetado, fruto de fusionar diferentes informaciones.

El objetivo principal de un A-SMGCS es incrementar la seguridad de las aeronaves monitorizando todo tipo de tráfico y facilitando a la Torre de Control de Aeródromo (TWR) el conocimiento de la situación del tráfico en toda la zona de movimientos y ofreciendo directivas para controlar y guiar la aeronave en tierra.

Consiste en un sistema modular que integra 4 funciones principales [2]:

- Vigilancia: Seguimiento de los móviles de interés proporcionando posición, velocidad y aceleración. Está constituido principalmente por sensores junto con un sistema que fusiona datos multi-sensor. La salida de este sistema es lo que se presenta en las posiciones de control.
- Supervisión y control: Mejora de la vigilancia de pistas y rodaduras detectando incursiones y posibles conflictos, dando la resolución más idónea si fuese preciso.
- Sistema de ruta: Generación de las rutas más eficientes para hacer uso de la total capacidad aeroportuaria.
- Guiado: Proporciona una indicación clara a pilotos conductores y controladores de la ruta asignada a las aeronaves y vehículos en el área de maniobras.

Tiene cuatro niveles de implantación que ofrecen distintos niveles de funcionalidad:

- *Nivel I*: presenta al controlador la posición (ubicación) de una aeronave o vehículo y su identidad.
- *Nivel II*: Incorpora la mejora de la vigilancia de incursiones y ofrece la función de predicción de conflictos para alertar al controlador sobre posibles colisiones (entre aeronave/vehículo o aeronave/aeronave) en la superficie de la pista o en áreas protegidas, o accesos de aeronaves o vehículos a áreas restringidas.
- *Nivel III*: añadirá características complementarias de la detección de conflictos en el área de movimiento y la mejora de guiado y planificación a utilizar por los controladores.
- *Nivel IV*: complementará el nivel II con función de resolución de conflictos y ofrecerá planificación automaática y el guiado para uso por pilotos y controladores

El objetivo actual en el paradigma SESAR es desarrollar al máximo los niveles I y II de A-SMGCS hasta que se complete su implantación operativa, y a continuación pasar a los niveles III y IV.

1.2. Vigilancia como parte de A-SMGCS

1.2.1. Vigilancia en Área de Aeropuerto

Los controladores de superficie, antes de tomar responsabilidad de una aeronave tienen que ser capaz de verla desde su posiciones de controlador. Para realizar su labor con eficiencia y seguridad, disponen en sus emplazamientos de equipamientos técnicos de presentación de radar que permiten visualizar las aeronaves y vehículos en movimiento por el aeródromo (fig.1.1).

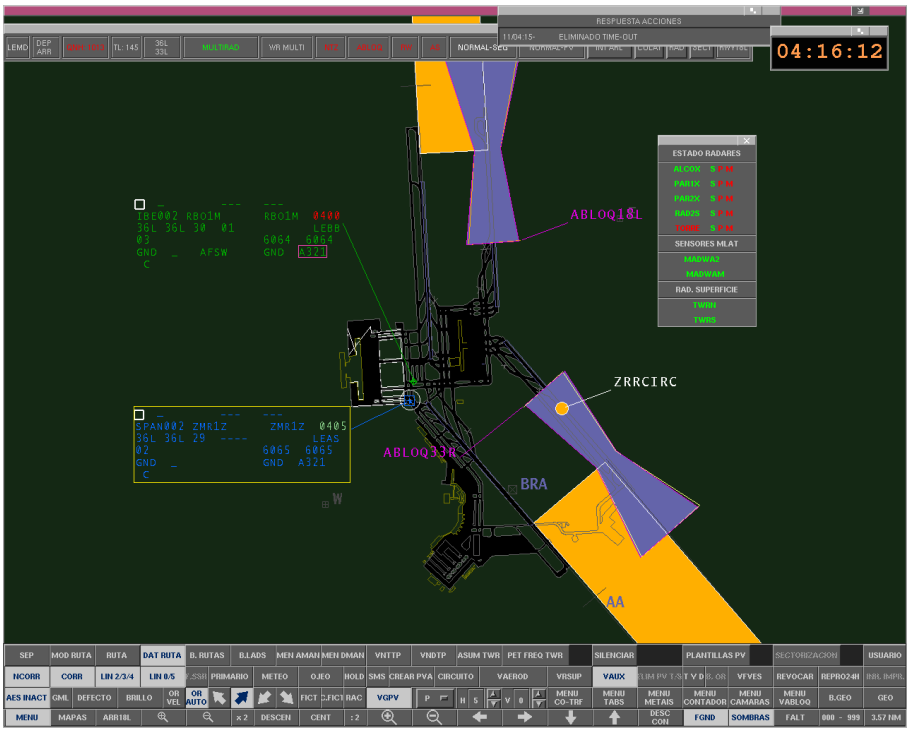


Figura 1.1: Seguimiento Radar. Sistema Automatizado de Control de Tránsito Aéreo

El seguimiento en el aeródromo no se limita a los vehículos en tierra sino que se extiende a las etapas finales de la operación de aproximación y aterrizaje.

Las áreas geográficas de interés para la vigilancia incluyen:

- pistas en la superficie del aeropuerto
- pistas en aproximación final, y especialmente en la trayectoria de planeo de aterrizaje.
- pistas en proceso de salida

Es fundamental hacer seguimiento de los aviones, pero también es importante vigilar otros blancos puesto que pueden comprometer la seguridad de las aeronaves. Los tipos de objetos que se pueden observar/detectar por los sensores desplegados en un aeropuerto son los siguientes:

- Aeronaves
- Helicópteros
- Vehículos, incluyendo camiones, buses, coches y vehículos de servicio
- Obstáculos.

El controlador y, por tanto, el sistema de proceso y presentación del Centro de Control donde se encuentra, necesita información externa en tiempo real de la posición de los vehículos para conocer donde se encuentran de manera precisa en un determinado momento. Esa información externa llega, principalmente, a través de los mensajes radar provenientes de las detecciones de los radares de vigilancia desplegados por la geografía y recientemente, de los mensajes de informe ADS aire-tierra enviados por la propia aeronave. Los sensores de vigilancia de superficie más comunes en los aeropuertos son:

- Radares de Aproximación: Son estaciones con radar secundario y, pocas veces, radar primario de medio alcance y periodos de antena comprendidos entre 4 y 8 segundos. Se utilizan para

la detección de aeronaves en fases de aproximación y despegue de un aeródromo. Se ubican en picos cercanos al aeródromo a cubrir.

  - *Radar de Vigilancia Primario* (PSR),
  - *Radar de Vigilancia Secundario* (SSR), o
  - Radar Modo S.

- Radares de Movimiento en Superficie (*Radar de Movimiento en Superficie* (SMR)) o Equipo de detección en la superficie del aeropuerto Airport Surface Detection Equipment *(ASDE)*: Son estaciones de radar primario de corto alcance, de alta resolución y periodos de antena comprendidos normalmente entre 1 y 4 segundos. Se utilizan para la detección de aeronaves en tierra circulando por el aeródromo en muchos grandes aeropuertos y se ubican en la Torre de Control. Proporcionan rango y azimut del blanco con mucha más exactitud que los radares PSR y SSR.
- Sistemas Multilateración (MLAT) Se usan para garantizar cobertura y buenos niveles de precisión en todo el área de interés. Es un sistema capaz de detectar, identificar y seguir blancos, independientemente de su tamaño, gracias a un proceso de triangulación de las señales recibidas en varios puntos del aeropuerto o próximos a él que permiten determinar su posición. Los periodos de actualización suelen ser de un segundo.
- GPS diferencial transmitido a través de enlace de datos digital (DGPS),
- *Vigilancia Dependiente Automática-Broadcast* (ADS-B),
- TV y/o cameras.

1.2.2. Función de Vigilancia de A-SMGCS

La función de vigilancia es una de las funciones principales del A-SMGCS, proporcionando al mismo de los datos cinemáticos y la identificación de los blancos. Recibe entradas de otras funciones del sistema, de sensores externos y de otras funciones de vigilancia que corren en los centros de ATC.

En [13] se describe la arquitectura funcional de una *Función de Vigilancia en Superficie de Aeropuerto* para cumplir con los requisitos del proyecto *P12.3.1 "Improved Surveillance for surface Management"* para la mejora de los sistemas actuales de vigilancia en superficie. En el diagrama 1.2 se muestran las funciones principales de la función de vigilancia A-SMGCS entre las que están la adquisición, preprocesado y procesado de datos de pistas y de sensores, identificación y la fusión de datos multi-sensor.

La parte de fusión de datos multisensor es la que genera y mantiene las trazas de Vigilancia de Superficie que describen la situación del tráfico en la superficie y cercanías del aeropuerto. La realización de este tratamiento implica procesos de asociación de los datos recibidos a trazas del sistema y actualización de las trazas incorporando información de las mediciones a las pistas asociadas.

En los estudios técnicos y teóricos llevados a cabo en el marco del Working Package 12 del SESAR [5] se identifican las posibles tecnologías que pueden aplicarse en la consolidación del *nivel II de A-SMGCS* para que sean analizadas.

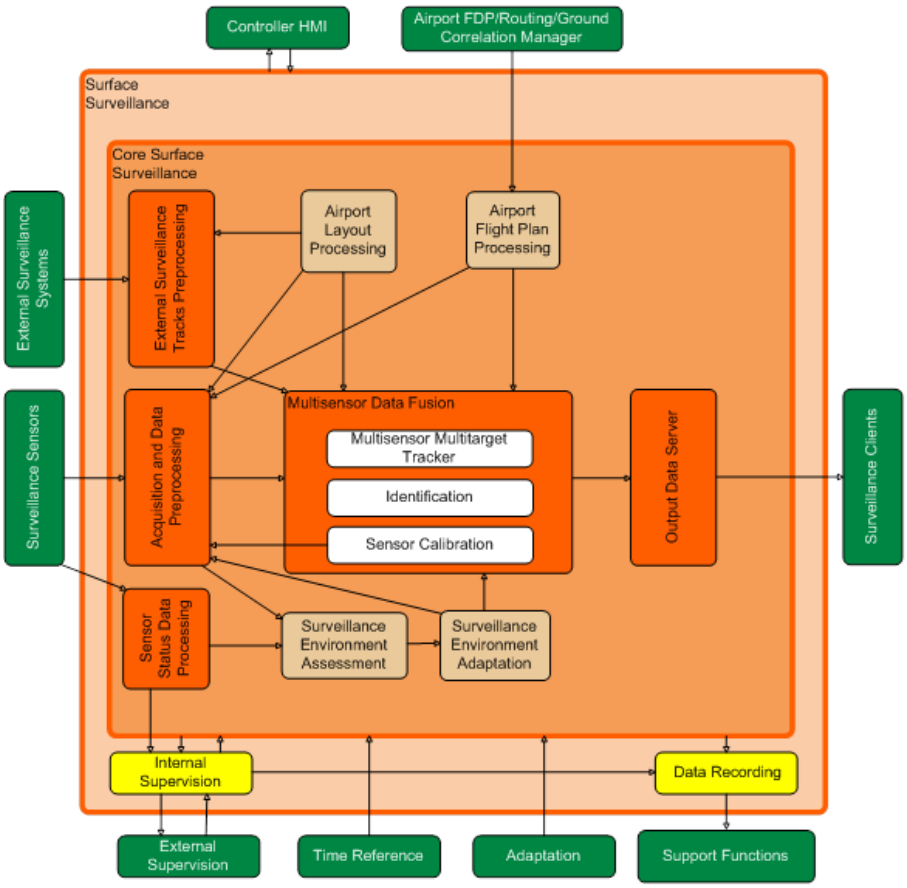


Figura 1.2: Arquitectura lógica de la función de vigilancia

1.3. Contexto de Actuación

1.3.1. Mejora del Sistema Automático de Vigilancia en Aeropuertos

Como se ha explicado en la sección anterior, la función de vigilancia se encarga de la detección automática, identificación/clasificación y seguimiento (tracking) de todos las aeronaves, vehículos relevantes en tierra, y obstáculos en el área de movimiento del aeropuerto.

El sistema de vigilancia instalado actualmente en las "Torres de Control de Aeródromo", para hacer el seguimiento de los objetivos móviles, no utiliza la información aportada por todos los sensores, sino que de las detecciones recibidas cada intervalo, se actualizan los datos cinemáticos del blanco a partir de la detección del sensor de mayor prioridad (de acuerdo con determinados criterios) sin realizar ningún tipo de procesado que pueda servir, por ejemplo para minimizar los errores de la detección de radar.

Para lograr los objetivos de *nivel II* de A-SMGCS y mejorar la seguridad de las aeronaves se necesita *información mucho más precisa sobre la cinemática de todos los blancos moviéndose en el área del aeropuerto.*

En los estudios del SESAR sobre los cambios que hay que hacer para mejorar los sistemas de vigilancia en superficie [5] se propone la *aplicación de algoritmos de estimación* en la función de fusión de datos multisensor con el fin de integrar los datos procedentes de los distintos radares minimizando los errores de detección.

1.3.2. Conceptos Básicos

Resulta útil en este punto aclarar algunos conceptos antes de comenzar con los aspectos más técnicos de este capítulo.

Estimación es el proceso de inferencia del valor de una variable de interés a partir de observaciones indirectas, imprecisas e inciertas.

Tracking (seguimiento) es la estimación del estado de un objeto en movimiento basado en medidas remotas. Se hace utilizando uno o más sensores en localizaciones fijas o en plataformas móviles 2.1. El *tracking* puede parecer un caso especial de estimación, pero en realidad su ámbito es mayor ya que no sólo utiliza todas las herramientas de estimación, sino que además requiere el uso extensivo de la teoría de la decisión estadística al tratar algunos de los problemas prácticos (por ejemplo la asociación de datos).

Filtrado es la estimación del estado(actual) de un sistema dinámico. Se utiliza la palabra 'filtro' porque el proceso para obtener el mejor estimado a partir de datos ruidosos equivale a filtrar/eliminar el ruido.

Estimador óptimo es un algoritmo computacional que procesa observaciones (medidas) para producir un estimado de una variable de interés, que optimiza un cierto criterio.

1.3.3. Dificultades Generales del Problema

La vigilancia de blancos en superficie del aeropuerto se trata de un problema de seguimiento de múltiples blancos en tierra utilizando varios sensores.

Aunque el seguimiento de múltiples blancos es una tecnología bastante madura, con aplicaciones en control de tráfico aéreo, defensa, aviónica, vigilancia oceánica, hasta hace poco no empezó a hacerse énfasis en el caso especial de seguimiento de blancos en tierra (*Ground Target Tracking* [6]) .

La localización de blancos en tierra presenta características que lo distinguen del seguimiento de otro tipo de blancos, como pueden los del aire: la alta densidad y maniobrabilidad de los blancos (maniobras que están limitadas por las condiciones del terreno), menor visibilidad , presencia de obstáculos... por lo que es necesario buscar otro tipo de estrategias.

En particular los requisitos de vigilancia de un sistema A-SMGCS suponen un gran desafío debido a la difíciles condiciones que se dan en el entorno de un aeropuerto (sección 1.2): en los aeropuertos hay muchos obstáculos que pueden bloquear la línea de visión de los sensores causando puntos negros y sombras, los objetos de interés efectúan movimientos condicionados por la forma y física de las carreteras, hay que mantener una distancia mínima entre blancos, etc.

El enfoque que prevalece en la actualidad para hacer seguimiento de blancos con maniobras es el uso de múltiples modelos (y filtros simultáneamente)

Por otro lado, la dinámica de los blancos varía considerablemente con respecto a objetivos desplazándose en el aire, y a otros problemas de seguimiento en tierra, al haber distintos tipos de blancos (aeronaves y vehículos terrestres) y diferentes tipos de movimiento (despegue, aterrizaje, aparcamiento, rodadura . . .). Los vehículos pueden acelerar, disminuir la velocidad, frenar y girar frecuentemente, dependiendo de las condiciones locales variables, de las condiciones del terreno y de la situación del tráfico.

1.3.4. Herramientas Matemáticas Propuestas

Los algoritmos que se consideran adecuados y que hay que analizar en profundidad para estimar la trayectoria de vehículos circulando en los aeropuertos según [5] incluyen:

- *Filtro de Kalman* (KF): algoritmo de estimación recursivo óptimo para sistemas que pueden ser descritos a través de un modelo estocástico lineal ya que los valores que calcula son probabilísticamente óptimos. Existen dos variaciones sub-óptimas del filtro para los casos no lineales: *Filtro Extendido de Kalman* (EKF) y *Filtro de Kalman Unscented* (UKF).
- *Múltiples Modelos Interactuantes* (IMM): filtro híbrido (asume que el sistema obedece a un número finito de modelos) que ha demostrado ser uno de los esquemas de estimación de estado más rentables; el algoritmo combina las estimaciones producidas por un número fijo de filtros, cada uno adaptado a un tipo de modo de movimiento
- *Filtro IMM de Estructura Variable* (VS-IMM): filtro IMM con un conjunto variable de modelos que se actualizan en el tiempo.
- *Filtro de Partículas* (PF)

1.3.5. Funcionamiento de la Función de Seguimiento Radar Mejorada

La función de seguimiento radar consiste en crear y seguir actualizando continuamente un elemento, llamado pista del vuelo o traza (track), que proporciona, en cada momento, la posición (X, Y) y la velocidad del vehículo de la manera más fiable y exacta posible, minimizando los errores de la detección radar.

Cada sensor del aeropuerto envía periódicamente al sistema central un mensaje radar con los datos de detección del blanco. Dependiendo del tipo de radar, el mensaje contendrá unos datos u otros y vendrá en cierto formato. Todos los radares envían la posición detectada, bien en coordenadas cartesianas o en rango y azimut, algunos envían información de identificación, velocidad, etc... Además la distribución del error de detección de los radares es conocida.

El centro de fusión del sistema recibe entonces periódicamente (cada T segundos) los datos que han sido detectados por los sensores, los procesa y transforma a un formato común teniendo en cuenta los parámetros del modelo del sensor que lo genera. A continuación asigna estas medidas detectadas por los radares a blancos y luego se actualiza la trayectoria y cinemática del blanco, mediante un *proceso de filtrado.*

1.4. Objetivos del Trabajo de Fin de Máster

Basándonos en los estudios teóricos [5] llevados a cabo por las empresas que colaboran en el proyecto WP12 del SESAR donde presentan los tipos de arquitecturas y algoritmos que se pueden aplicar para desarrollar un sistema de vigilancia que cumpla con los requisitos de implantación del nivel II de A-SMGCS, este trabajo se centra en el análisis de un *filtro de estimación de trayectorias (actualización del vector de estado)* para la función de seguimiento radar 1.3.5.

Se intenta diseñar y seleccionar el filtro adaptativo IMM que mejor se comporta a la hora de calcular la trayectoria real de un vehículo cuando se mueve por la superficie de un aeropuerto, teniendo en cuenta las características de los distintos clases de vehículos y tipos de sensores presentes.

Se pretende obtener una estimación lo más precisa posible de la actitud y posición de un vehículo mediante la integración en el filtro IMM de todas las medidas de los distintos sensores asociadas a dicho vehículo.

Características Particulares

- Desarrollo de un algoritmo de seguimiento para vehículos circulando por el área visible desde la torre de control de un aeropuerto, basado en técnicas IMM y filtros Kalman (No tendremos

en cuenta los mapas de aeropuertos)

- Simulación de escenarios, que serán definidos teniendo en cuenta:

  Tipos de vehículos Aeronaves realizando maniobras de desplazamiento , despegue y aterrizaje, y vehículos de tierra circulando por el aeropuerto.

  Trayectorias Las trayectorias se definirán en un modelo de tierra plana con un conjunto de segmentos contiguos en el tiempo, pudiendo ser el movimiento: velocidad constante, giros a velocidad constante(aceleración transversal constante), aceleración longitudinal constante, aceleración transversal y longitudinal constantes.

  Cantidad y tipos de sensores Dos radares de movimiento en superficie SMR, que facilitan medidas del rango y azimut observados, un *Sistema de Multilateración* (MLAT) que proporciona los datos de detecciones en coordenadas cartesianas, y un radar de aproximación. Los parámetros de los radares serán la precisión de la medida de rango y azimut (radares de superficie), o de posición en coordenadas cartesianas. Asumimos que los errores en las medidas de los radares siguen una distribución gaussiana con media cero.

- Análisis de los errores (Raíz del Error Cuadrático Medio) de los resultados obtenidos al aplicar los filtros de estimación, comparándolos con los que se obtendrían utilizando únicamente las pistas maestras.

Capítulo 2

Herramientas Matemáticas

El objetivo principal del seguimiento de un objeto en movimiento es estimar su trayectoria. El desempeño de la mayoría de las técnicas para el seguimiento (incluidos el KF, y el IMM) se basa en modelos matemáticos de los blancos definidos a priori, los cuales se supone que son lo suficientemente precisos.

En este capítulo comenzaremos introduciendo algunas ideas matemáticas básicas para a continuación, describir las herramientas matemáticas más relevantes para el problema de seguimiento de objetos (modelos dinámicos y filtros), las cuales combinaremos en los siguientes capítulos para diseñar el filtro IMM pertinente y generar el entorno de simulación.

2.1. Principios Básicos

A modo de introducción muy básica se presentan a continuación nociones elementales que consideramos útiles para comprender más adelante el funcionamiento de los filtros y el proceso de simulación.

2.1.1. Modelos Matemáticos

Un **Sistema**, en su acepción general, se define como una parte del universo aislada, conceptualmente, para su estudio. Para nosotros es un objeto o colección de objetos con entradas y salidas. Un sistema dinámico es aquel cuyo comportamiento cambia con el tiempo.

Un **Modelo** es una representación simplificada de la realidad, constituye una herramienta que permite comprender, predecir y controlar el comportamiento del sistema que representa. Está formado por elementos, que caracterizan la realidad modelada y las relaciones existentes entre ellos.

Un **Modelo Matemático** es un modelo abstracto que utiliza el lenguaje matemático para describir el comportamiento de un sistema. Sus elementos son esencialmente variables y funciones, y las relaciones entre ellos vienen expresadas a través de relaciones matemáticas (ecuaciones, operadores lógicos...) que se corresponden con las correspondientes relaciones del mundo real que modelan (relaciones tecnológicas, leyes físicas...).

Estado de un sistema es el conjunto mínimo de variables necesarias para caracterizar o describir todos aquellos aspectos de interés del sistema en un cierto instante de tiempo.

Los modelos que los representan, se pueden clasificar según distintos criterios:

- Los ***modelos estáticos*** representan objetos: se interpreta la realidad en un instante con-

creto, como resultado de procesos que no intervienen en la modelización. Los ***modelos dinámicos*** representan procesos, simulan mecanismos de cambio. Alguno(s) de los elementos que intervienen en el modelado no permanecen invariables sino que se consideran como funciones del tiempo y describen trayectorias temporales.

- Los ***modelos deterministas*** generan los mismos resultados si se parte del mismo escenario (mismos datos y mismos algoritmos). En los ***estocásticos*** se introduce ruido en el proceso y se generan diferentes resultados a partir de un mismo escenario de partida; los valores que toma a lo largo del tiempo no son determinados con certeza absoluta sino que siguen una distribución de probabilidad.
- Los ***modelos continuos*** se caracterizan por representar la evolución de las variables de interés de forma continua. Suelen utilizarse ecuaciones diferenciales ordinarias si se considera simplemente la evolución de una propiedad respecto al tiempo, o bien ecuaciones en derivadas parciales si se considera también la evolución respecto a otras variables adicionales. Los ***modelos discretos*** se caracterizan por representar la evolución de las variables de interés de forma discreta.

Es importante notar, a partir de la clasificación de modelos realizada, que es posible describir un sistema continuo mediante un modelo discreto y un sistema discreto mediante un modelo continuo. La decisión de utilizar un modelo continuo o discreto depende de los objetivos particulares de cada estudio y no tanto de las características del sistema.

2.1.2. Teoría Estadística

En este apartado definimos de modo conciso algunos términos estadísticos que son utilizados por los filtros de estimación para aclarar la notación que se utilizará a lo largo de este trabajo. Para profundizar más en estas cuestiones consultar por ejemplo [15], [7] y [16].

Una variable aleatoria X es una función que mapea todos los puntos de un espacio de muestreo a números reales (por ejemplo, para cualquier punto en el tiempo, $X(t)$ nos diría la posición esperada. La probabilidad de un evento discreto cualquiera tiene probabilidad cero $p(A)x = 0$. Una variable aleatoria continua es una variable aleatoria asociada a un espacio muestral continuo de tal manera que los posibles valores que toma esa variable pertenecen a los números reales.

Valor Esperado de una variable aleatoria es su valor medio y se denota por $\mu = E[X]$

$$E[X] = \int_{-\infty}^{\infty} x f_X(x) dx$$

El valor esperado de una función $g(x)$ de una variable aleatoria es el valor medio de g(x)

$$E[g(X)] = \int_{-\infty}^{\infty} g(x)^k f_X(x) dx$$

El Momento k-ésimo de una variable aleatoria es el valor esperado de la función $g(x) = x^k$

$$m_k = E[X^k] = \int_{-\infty}^{\infty} x^k f_X(x) dx$$

El primer momento de una variable aleatoria es su valor esperado.

El Momento Central k-ésimo con respecto al valor esperado de una variable aleatoria es el valor esperado de la función $g(x) = (X - E[X]))^n$.

La Varianza de una variable mide la dispersión de los valores de la variable respecto de su media μ. Es el valor esperado de las distancias de la variable a su valor esperado, medidas en forma cuadrática (es el segundo momento respecto a la media):

$$V(X) = E[(X - E[X])^2] = E[X^2] - E[X]^2$$

Varianza Conjunta o Covarianza de dos variables aleatorias X e Y mide la relación entre ellas. Se mide en función del valor esperado conjunto y los valores esperados marginales:

$$Cov(X, Y) = E[(X - E[X])(Y - E[Y])]$$

Si no están correlacionadas, su covarianza es 0. Dos variables aleatorias independientes tienen covarianza cero.

Vector Aleatorio conjunto de variables aleatorias $x = (x_1, x_2, \dots x_n)$

- Matriz de Correlación (Autocorrelación): $R_x = E[\boldsymbol{x}\boldsymbol{x}^T]$
- Matriz de Covarianza (Autocovarianza): $C_x = E[(\boldsymbol{x} - E[\boldsymbol{x}])(\boldsymbol{x} - E[\boldsymbol{x}])^T]$

Distribución de Probabilidad Normal o Gaussiana Los datos se agrupan en torno a la media μ, y su amplitud viene dada por la varianza σ^2. Se representa por $X \cong N(\mu, \sigma^2)$

2.2. Modelos Dinámicos

Un modelo dinámico, también llamado modelo de movimiento, describe la evolución del estado de un objetivo x con respecto al tiempo. Estos modelos asumen que el movimiento del objetivo junto con sus observaciones puede ser representado de forma precisa por medio de algún modelo matemático conocido previamente.

2.2.1. Estado, Vector de Estado

Se denomina ***Vector de Estado*** al mínimo conjunto de n variables

$$\boldsymbol{x}(t) = [x_1(t), x_2(t), ...x_n(t)]$$

cuyo conocimiento en el instante, $t = t_0$, junto con el conocimiento (opcional) de una entrada $u(t)$ para $t \geq t_0$ nos permiten predecir la salida, $y(t)$, para $t \geq t_0$.

El vector de estado es, por tanto, un conjunto de funciones del tiempo, $x_i(t)$ llamadas ***variables de estado***, que son las variables de interés (por ejemplo: la posición, velocidad, aceleración, ángulos de orientación, ...).

Al conjunto de valores numéricos que en un instante dado t_1 , adoptan las variables de estado, le llamamos ***estado*** del sistema en el instante t_1.

> El *estado de un sistema* en un instante dado representa la mínima memoria del pasado necesaria para predecir su respuesta ante una entrada futura.

2.2.2. Modelo de Espacio de Estados

El tipo de modelo más utilizado es el *Modelo de Espacio de Estados*, mediante el cual se describe la dinámica del sistema en función del valor del vector de estado y una señal de entrada. En general se describe con dos ecuaciones:

Ecuaciones del Sistema que modelan la dinámica del estado de las variables

$$\boldsymbol{x}(k+1) = f_k(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{\nu}(k)) \tag{2.2.2.1}$$

siendo

- $\boldsymbol{x}(k)$ el vector de estado (n variables) del sistema en el instante t_k,
- $\boldsymbol{u}(k)$ la entrada de control opcional
- $\boldsymbol{\nu}(k)$ el ruido de proceso
- f_k es una función que modela la forma en la que el sistema evoluciona del tiempo $k-1$ al k

Ecuaciones de las Observaciones modelan el estado observado de las variables

$$\boldsymbol{z}(k) = h_k(\boldsymbol{x}(k)) + \boldsymbol{w}(k) \tag{2.2.2.2}$$

con

- $\boldsymbol{z}(k)$ el vector de observación
- $\boldsymbol{w}_k$ es el ruido de medida
- h_k es una función que modela la relación existente entre las observaciones realizadas y el estado del sistema

Las ecuaciones 2.2.2.1 y 2.2.2.2, conocidas como modelo del proceso y modelo de mediciones, son la base de prácticamente cualquier método de estimación, como el filtro Kalman.

Para utilizar estas ecuaciones en los filtros conviene formular de forma más precisa las funciones que describen la evolución del sistema y la relación entre las observaciones y el estado del mismo mediante los modelos de tiempo continuo y discreto.

2.2.3. Modelos de tiempo continuo

Un modelo de tiempo continuo es aquel en que el estado del sistema cambia de forma continua respecto al tiempo. Estos cambios se describen por medio de las derivadas de las variables que resumen el estado del sistema, con respecto al tiempo. En general suelen utilizarse ecuaciones diferenciales ordinarias si se considera simplemente la evolución de una propiedad respecto al tiempo, o bien ecuaciones en derivadas parciales si se considera también la evolución respecto a otras variables adicionales.

Los sistemas lineales estocásticos de tiempo continuo se pueden representar en espacio de estados según 2.2.3.1:

$$\dot{\boldsymbol{x}}(t) = A(t)\boldsymbol{x}(t) + B(t)\mathbf{u}(t) + D(t)\boldsymbol{\nu}(t) \tag{2.2.3.1}$$

$$\boldsymbol{z}(t) = C(t)\boldsymbol{x}(t) + \boldsymbol{w}(t) \tag{2.2.3.2}$$

- La matriz A de dimensión $n \times n$ es la matriz de sistema y define la dinámica lineal; relaciona el estado en el periodo previo $(t-1)$ con el estado en el momento actual t en ausencia de ruido.
- La matriz B de dimensión $n \times 1$ refleja la influencia de la entrada de control o función externa opcional $u \in R^1$.

- La matriz C de dimensión $m \times n$ relaciona el estado con la medición realizada en el tiempo actual $\boldsymbol{z}(t)$.
- $\dot{x}(t) = d\boldsymbol{x}(t)/dt$

Si el sistema no varía en el tiempo, las matrices son constantes y el modelos se puede simplificar. Además, para nuestros sistemas no habrá entrada de control, por lo que las ecuaciones quedarían:

$$\dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t) + D\boldsymbol{\nu}(t) \tag{2.2.3.3}$$
$$\boldsymbol{z}(t) = C\boldsymbol{x}(t) + \boldsymbol{w}(t) \tag{2.2.3.4}$$

2.2.4. Modelos de tiempo discreto

En los sistemas en los que las mediciones de las características que describen el comportamiento del objeto que se analiza se realizan en instantes de tiempo concretos (por ejemplo la posición de un vehículo en un instante dado) se describen por medio de ecuaciones de diferencias. Se utilizan ecuaciones que son las equivalentes discretas de las de los modelos de tiempo continuo.

Para los ***sistemas lineales estocásticos de tiempo discreto***, el modelo se puede describir por medio de

$$\boldsymbol{x}(k+1) = F(k)\boldsymbol{x}(k) + G(k)\mathbf{u}(k) + \boldsymbol{\nu}(k) \tag{2.2.4.1}$$

- F la matriz de transición del estado del sistema
- G la matriz de ganancia en tiempo discreto de la entrada de control
- $\boldsymbol{\nu}(t)$ representa el ruido del proceso en tiempo discreto.

Normalmente F se obtiene discretizando una la ecuación diferencial ordinaria apropiada y G será el homólogo en tiempo discreto del ruido aplicado a la *Ecuación Diferencial Ordinaria* (EDO).

Y la ecuación de mediciones en tiempo discreto esta dada por

$$\boldsymbol{z}(k) = H(k)\boldsymbol{x}(k) + \boldsymbol{w}(k) \tag{2.2.4.2}$$

con H la matriz de mediciones y $w(k)$ el ruido de mediciones.

En [16] aclaran que a veces es conveniente definir un ***modelo directo de tiempo discreto***, en lugar de una versión discretizada de un modelo de tiempo continuo. En estos casos se modifica el modelo para incluir una matriz de ganancia de ruido del proceso, $\Gamma(k)$, a través de la cual el ruido $\boldsymbol{\nu}(k)$ entra al sistema, por lo que la ecuación que caracteriza la evolución del sistema del tiempo k al tiempo $k+1$ es 2.2.4.3:

$$\boldsymbol{x}(k+1) = F(k)\boldsymbol{x}(k) + G(k)u(k)t + \Gamma(k)\boldsymbol{\nu}(k) \tag{2.2.4.3}$$

Utilizar $\Gamma(k)$ hace posible definir la matriz de covarianza de ruido del proceso $Q(k)$ de forma directa.

Para el caso en que el sistema sea invariante en el tiempo y no haya entrada de control, la ecuación quedaría como 2.2.4.4:

$$\boldsymbol{x}(k+1) = F\boldsymbol{x}(k) + \Gamma\boldsymbol{\nu}(k) \tag{2.2.4.4}$$

Nota sobre Modelos de Movimiento

Como normalmente las observaciones solo están disponibles en instante de tiempo discretos se hace necesario utilizar modelos de tiempo discreto. Sin embargo, normalmente los movimientos de blancos se describen de manera más precisa mediante modelos de tiempo continuo.

Cuando no se dispone de las ecuaciones de estado en tiempo discreto hay que realizar un proceso de discretización de las ecuaciones en tiempo continuo ([16]), que no se explicará aquí por quedar fuera del alcance de este proyecto.

2.3. Filtros de Estimación

Para hacer la estimación del estado de un sistema dinámico, un filtro de estimación utiliza el conocimiento que tiene acerca de

- la evolución de la variable (la dinámica del sistema)
- el modelo de la medida
- la caracterización probabilística de varios factores aleatorios (perturbaciones/ruidos)
- la información *a priori*

.

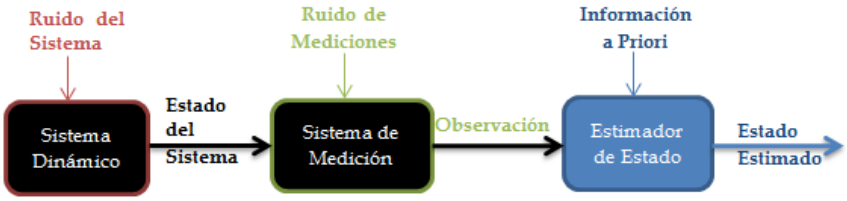


Figura 2.1: Estimación de Estado: extracción de información y mejora

Para hacer el seguimiento de un blanco existen dos enfoques principales:

- Aplicación de un filtro basado en un único modelo, que es muy simplista y solo funciona si el modo de movimiento del blanco es invariante y conocido. La única función de este filtrado elemental es en realidad reducir el efecto de las perturbaciones (producidas por el ruido de proceso y de medidas) utilizando los datos conocidos (del modo del blanco y las mediciones)
- Algoritmos basados en modelos múltiples, que utilizan un conjunto de modelos como posibles candidatos para describir el movimiento del blanco en cada instante de tiempo.

Cuando el comportamiento del sistema puede dividirse en dos o más modelos (como es el caso de los vehículos circulando en superficie, cuyo movimiento puede moldearse mediante trayectorias lineales y curvilíneas) el enfoque de un filtro elemental como el Filtro de Kalman (KF) para procesar el ruido para representar las aceleraciones, es inadecuado, ya que una maniobra puede implicar un amplio rango de comportamientos. En su lugar se puede aplicar el filtro IMM, que es un filtro híbrido sub-óptimo que ha de mostrado ser uno de los esquemas de estimación de estado más rentables.

El filtro IMM calcula la probabilidad que tiene cada modo de ser el correcto de acuerdo a lo observado y le asigna un peso. El algoritmo combina las estimaciones producidas por filtros, cada

uno adaptado a un tipo de modo de movimiento (tales como movimiento uniforme a velocidad constante, movimiento de maniobra con aceleración...).

En esta sección se describen los dos algoritmos que constituyen la base de nuestro trabajo: el Filtro de Kalman (KF) como elemento clave en los filtros de múltiples modelos y el Múltiples Modelos Interactuantes (IMM) que es el objetivo de este proyecto.

2.3.1. Filtro Kalman

El objetivo del filtro es la obtención de un estimador óptimo de las variables de estado de un sistema dinámico, basado en observaciones ruidosas y en un modelo de incertidumbre de la dinámica del sistema.

Un filtro Kalman es un algoritmo recursivo (no necesita almacenar todos los datos anteriores para procesarlos cada vez que una medida nueva llega) óptimo de procesamiento de datos.

Bajo ciertos supuestos, el filtro Kalman es óptimo con respecto a casi cualquier criterio con sentido. Un aspecto de la optimalidad es que incorpora toda la información que se le puede aportar [7]. Procesa todas las medidas disponibles con independencia de lo precisas que sean, para estimar el valor actual de las variables de interior, utilizando:

1. conocimiento del sistema y dinámica del dispositivo de medición,
2. la descripción estadística de los ruidos de sistema, errores de medida, e incertidumbre en la dinámica de los modelos, y
3. cualquier información disponible acerca de las condiciones iniciales de las variables de interés

Minimiza el Error Cuadrático Medio (MSE) de los parámetros estimados , pero lo hace basándose en más parámetros. Además predice cuál debería ser la salida para el estado siguiente y da una estimación a partir de las medidas en el mismo instante de tiempo. Todas estas medidas tienen su propia estadística y dependiendo de cual sea más fiable, se otorga más peso a la predicción o a la estimación. En el fondo se integran estados redundantes para obtener información más precisa.

El KF es el mejor estimador bayesiano disponible cuando se cumplen los requisitos para aplicarlo (que el modelo del sistema sea lineal, la relación entre las medidas del sistema y del estado sean lineales, que los modelos de ruido estén distribuidos uniformemente en todo el espectro y que presenten una distribución gaussiana con media cero), ya que los valores que calcula son probabilísticamente óptimos.

2.3.1.1. Ecuaciones del Filtro de Kalman

El filtro de Kalman [15] tiene como objetivo resolver el problema general de estimar el estado $x \in \Re^n$ (conjunto de variables de interés: posición y velocidad y posiblemente la aceleración y otras variables utilizadas para modelar la aceleración variable en el tiempo) de un proceso controlado en tiempo discreto, el cual es dominado por una *ecuación lineal en diferencia estocástica* (2.2.4.1) del tipo:

$$\boldsymbol{x}(k) = A\boldsymbol{x}(k-1) + Bu(k) + \boldsymbol{\nu}(k) \tag{2.3.1.1}$$

Con un vector de medida $\boldsymbol{z} \in \Re^m$ que corresponde a:

$$\boldsymbol{z}(k) = H\boldsymbol{x}(k) + \boldsymbol{w}(k) \tag{2.3.1.2}$$

- La matriz A de (2.3.1.1) de dimensión $n \times n$ define la dinámica lineal; relaciona el estado en el periodo previo $k-1$ con el estado en el momento actual t en ausencia de ruido.

- La matriz B de dimensión $n \times 1$ refleja la influencia de la entrada de control o función externa opcional $u \in R^1$.
- La matriz H de dimensión $m \times n$ de (2.3.1.2) relaciona el estado con la medición realizada en el tiempo actual $\boldsymbol{z}(k)$.

Las variables aleatorias $\boldsymbol{\nu}_t$ y $\boldsymbol{w}_t$ representan el ruido/error del proceso y de la observación respectivamente. Se asume que son independientes entre ellas, que son ruido blanco y con distribución de probabilidad normal:

$$p(\boldsymbol{\nu}) \cong N(0, Q) \qquad p(\boldsymbol{w}) \cong N(0, R)$$

Siendo Q y R la matriz de covarianza del ruido de proceso $\boldsymbol{\nu}$, y la matriz de covarianza del ruido de la observación $\boldsymbol{w}$, respectivamente, definidas como

$$Q = E[\boldsymbol{\nu}\boldsymbol{\nu}'] \qquad R = E[\boldsymbol{w}\boldsymbol{w}']$$

El filtro de Kalman es un algoritmo predictor-corrector que es dado en términos de actualización de tiempo y de medida.

Teniendo,

- $\hat{\boldsymbol{x}}_k^-$ estimación del estado en el tiempo k dadas las medidas durante el tiempo $k-1$; es decir, antes de conocer información sobre el instante k. Es la estimación a priori.
- $\hat{\boldsymbol{x}}_k$ estimación del estado en el tiempo k después de conocer la observación en el instante k.
- $e_k^- \equiv x_k - \hat{x}_k^-$ error de estimación *a priori*
- $e_k \equiv x_k - \hat{x}_k$ error de estimación *a posteriori*
- $P_k^- = E[e_k^- e_k^{-T}]$ Matriz de covarianza del error *a priori*
- $P_k = E[e_k e_k^T]$ Matriz de covarianza del error a posteriori de clasificación

El objetivo es dar una ecuación que calcule el estimador *a posteriori* a partir del estimador *a priori* (es estado predicho) y el error en la predicción de la observación.

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + K_k(\boldsymbol{z}_k - H\hat{\boldsymbol{x}}_{k-1}^-) \tag{2.3.1.3}$$

con

- $(\boldsymbol{z}_k - H\hat{\boldsymbol{x}}_{k-1}^-)$ el residuo o innovación en la observación.
- K es la ganancia de Kalman (2.3.1.6), o factor de mezcla. Establece la cantidad de influencia del error entre la estimación y la medida Es importante notar:
  - Si la covarianza del error en la medición se aproxima a cero

    $$\lim_{R \to 0} K_k = H^{-1}$$

    la ganancia K otorga mayor peso a la innovación de la medición, i.e. se confía más en la medición
  - Si el estimador de la covarianza del error a priori se aproxima a cero

    $$\lim_{P_k^- \to 0} K_k = 0$$

    la ganancia K otorga menor peso a la innovación de la medición.

El filtro mantiene los dos primeros momentos de la distribución de los estados

$$E[x_k] = \hat{x}_k \tag{2.3.1.4}$$
$$E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] = P_k \tag{2.3.1.5}$$

Si el ruido es gaussiano, entonces se cumple que la distribución de los estados es también gaussiana, con media la estimación *a posteriori* del estado, y covarianza la del error de la estimación

$$p(x_k|z_k) \sim N(\hat{x}_k, P_k)$$

Por tanto, las ecuaciones del filtro se pueden clasificar en dos tipos:

Ecuaciones de actualización en el tiempo o de predicción son responsables de proyectar hacia el futuro los estimadores del estado actual y de la covarianza del error, para obtener los estimadores *a priori* del siguiente estado. Predicen el estado a partir del estado anterior y las ecuaciones dinámicas

$$\hat{x}_k^- = A\hat{x}_{k-1}^- + Bu_k$$
$$P_k^- = AP_{k-1}AT + Q$$

Ecuaciones de actualización de medidas o de corrección son responsables de la realimentación, incorporando una nueva medida en el estimado *a priori* para obtener un estimado *a posteriori* mejorado.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{2.3.1.6}$$
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_{k-1}^-) \tag{2.3.1.7}$$
$$P_k = (I - K_k H)P_k^- \tag{2.3.1.8}$$

Primero se calcula la ganancia K, se obtiene la medida z_k y se genera un estimado del estado *a priori* al incorporar la medición. Para finalizar se obtiene una estimación de la covarianza del error.

Observaciones

Las matrices Q y R son críticas para el funcionamiento del filtro. Si son constantes y están bien estimadas mediante un proceso de identificación, las matrices de covarianza del error a posteriori y a priori convergen rápidamente y son constantes que pueden ser evaluadas off-line de una vez. Si no son constantes, estimar on-line para tener resultados aceptables, es un problema no trivial.

2.3.2. Filtro Múltiples Modelos Interactuantes (IMM)

En el enfoque de múltiples modelos o sistema híbrido, se asume que el sistema obedece a uno de entre un número finito de modelos y se utiliza un framework bayesiano: partiendo de las probabilidades a priori de que cada modelo sea el correcto (que el sistema esté en un modo particular), se obtienen las probabilidades a posteriori correspondientes [6].

El modelo que se asume que está en efecto es uno de entre r modelos posibles:

$$M \in M_j \qquad j = 1, \dots r$$

y la probabilidad a priori de que el modelo M_j sea el correcto es

$$P\{M_j|Z^0\} = \mu_j(0) \qquad j = 1, \dots r$$

siendo Z^0 la información a priori y

$$\sum_{j=1}^{r} \mu_j(0) = 1$$

Cada modelo/modo tiene asociada una probabilidad de modo μ_j (la probabilidad de que el blanco esté siguiendo ese tipo de movimiento en el tiempo actual) y su estado estimado y covarianza, pero para la salida del filtro IMM, estos se combinarán resultando en el estado estimado final $\hat{\boldsymbol{x}}$.

La metodología de seguimiento IMM mantiene un conjunto de modelos dinámicos diferentes, cada uno ajustado a un tipo específico de patrón de movimiento (por ejemplo velocidad constante, aceleración constante...), y representa la trayectoria de un blanco como una serie de estados, con una secuencia de transiciones modelada como una cadena de markov.

En cada instante de tiempo se asume que existe una probabilidad p_{ij}, que debe conocerse a priori, de que el blanco haga una transición del modelo i al j.

$$p_{ij} = P(M_k = M_j|M_{k-1} = M_i)$$

Se representa en la *Matriz de Transición* (estocástica) 2.3.2.

$$P_T = \begin{pmatrix} p_{11} & .. & p_{1m} \\ .. & .. & p_{1m} \\ p_{m1} & .. & p_{mm} \end{pmatrix}$$

dicha matriz de transición no tiene por qué ser estacionaria, sino puede variar según el intervalo de actualización, adaptarse a ciertos eventos, etc.

El algoritmo utiliza r filtros al mismo tiempo, uno para cada modo de movimiento, que calculan independientemente estimados del sistema: en el tiempo k se computa el estado estimado bajo cada uno de los posibles modelos utilizando los r filtros, cada uno de los cuales utiliza una combinación diferente de los estimados condicionados previos.

Denotando j el modelo que esté en efecto, las ecuaciones de los sistemas lineales las podemos representar ahora por :

$$\boldsymbol{x}_k = F^j_{k-1}\boldsymbol{x}_{k-1} + \boldsymbol{\nu}^j_{k-1}$$
$$\boldsymbol{z}_k = H^j_k\boldsymbol{x}_k + \boldsymbol{w}^j_k$$

Con cada nueva medida el IMM calcula la probabilidad que tiene cada modo de ser el correcto de acuerdo a lo observado (*actualiza las probabilidades de modo*) y le asigna un peso. Por medio de estos pesos, calcula la contribución de cada filtro individual al resultado total y construye un estimado del estado del sistema.

El método consiste en 4 partes,

- *Interacción:* los estimados de cada filtro son mezclados según la probabilidad predicha de cada modelo.
- *Filtrado según cada modelo:* cada filtro predice y actualiza su estado estimado utilizando el modelo dinámico que asume.
- *Actualización de la probabilidad de modo:* la probabilidad de modo de cada modelo es actualizada de acuerdo con la innovación del error.
- *Combinación:* para generar la salida del filtro se calcula un estado estimado combinado a partir de los estimados ponderados.

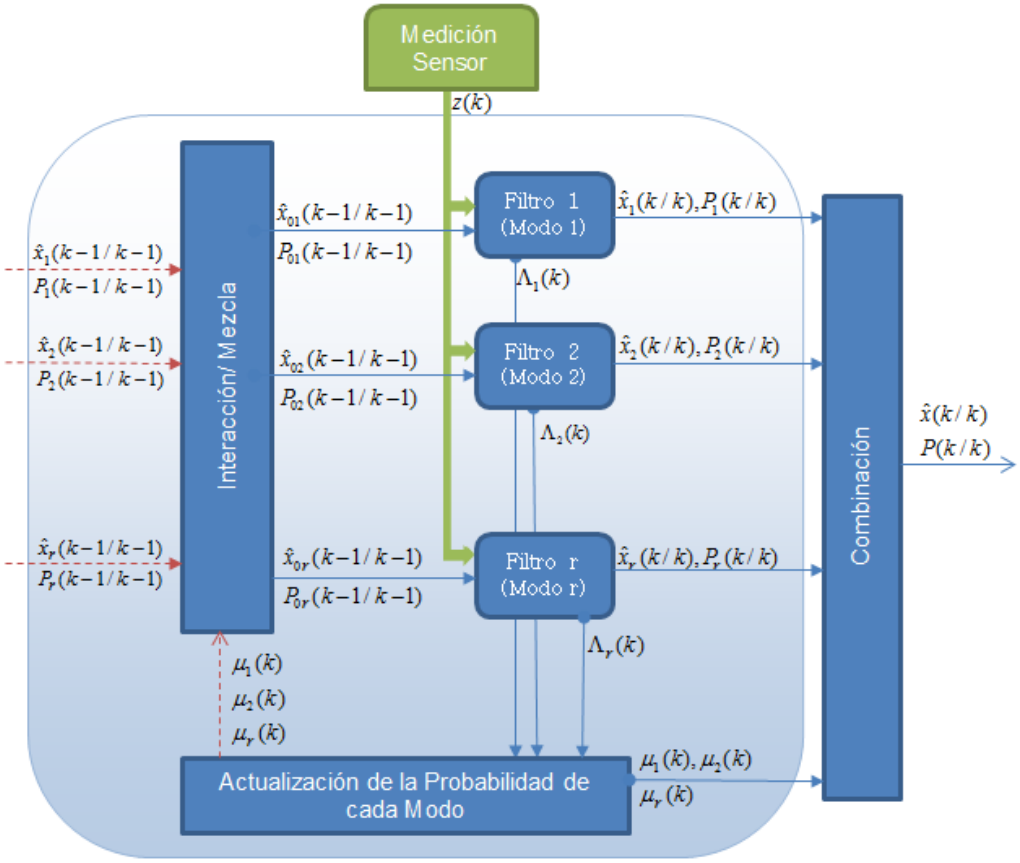


Figura 2.2: Estructura general del algoritmo de estimación IMM

2.3.2.1. Ciclo de IMM

Bar-Shalom y otros autores proporcionan una explicación detallada del algoritmo ([16],[8]). A continuación se presenta una descripción resumida de un ciclo del mismo:

1. *Probabilidades de mezcla*: se calcula la probabilidad de que el fenómeno se estuviese comportando de acuerdo al modo M_i en el tiempo $k-1$ dada la hipótesis de que el modo M_j está en efecto en el tiempo actual k, condicionado por las mediciones del tiempo anterior $\boldsymbol{z}_{k-1}$

$$\mu_{i|j}(k-1|k-1) = \frac{1}{\bar{c}_j} p_{ij}(k-1)\mu_i(k-1) \tag{2.3.2.1}$$

donde $\mu_i(k-1)$ es la probabilidad del modelo M_i en el paso $k-1$ y $\bar{c}_j$ el factor de normalización:

$$\bar{c}_j = \sum_{i=1}^{r} p_{ij}\mu_i(k-1) \tag{2.3.2.2}$$

2. *Cálculo de las condiciones iniciales mezcladas*: se obtienen los estimadores [2.3.2.3], y covarianzas [2.3.2.3] mezclados, asociados a cada modo ($j = 1, \dots r$), los cuales servirán de entrada para el filtro correspondiente a cada uno:

$$\hat{x}_{0j}(k-1) = \sum_{i=1}^{r} \hat{x}_i(k-1)\mu_{i|j}(k-1) \tag{2.3.2.3}$$

$$P_{0j}(k-1) = \sum_{i=1}^{r} \mu_{i|j}(k-1) P_i(k-1) + X_j(k-1) \tag{2.3.2.4}$$

siendo $\hat{x}_i(k-1)$ y $P_i(k-1)$ los valores de la media y covarianza para el modelo i en el tiempo $k-1$. La segunda parte de la ecuación 2.3.2.4 es un término de corrección de la desviación de cada modo.

$$X_j(k-1) = [\hat{x}_i(k-1) - \hat{x}_{0j}(k-1)][\hat{x}^i(k-1) - \hat{x}_{0j}(k-1)]^T$$

3. *Filtrado según cada uno de los r modos*: una vez realizado el mezclado de modos, se hace la predicción del vector de estado correspondiente a cada modelo. Lo que distingue a cada modelo son sus ecuaciones de predicción, que pueden ser lineales o no, en cuyo caso se precisa un filtro de tipo Kalman extendido (EKF), y la varianza del ruido de planta del modelo, es decir: El estimado [2.3.2.4] y la covarianza [2.3.2.4] se utilizan como entrada para los filtros correspondientes a cada modo. Utilizando la medida en el instante actual, cada filtro efectúa la predicción de los vectores de estado de un modelo M_j y su covarianza, obteniéndose $\hat{x}_j(k)$ y $P_j(k)$

   Además se computan las funciones de verosimilitud Λ_j correspondientes a cada modo.

$$\Lambda_j = |S_j|^{1/2} \exp\{-\frac{1}{2} v_j^T(k) S_j^{-1} v_j(k)\} \qquad j = 1, \dots r \tag{2.3.2.5}$$

   donde v_j y S_j son la innovación(residuo) en la medida y su covarianza del filtro j.

4. *Actualización de las probabilidades de modo* (probabilidades de modo *a posteriori*). Para cada filtro se actualiza la probabilidad $\mu_j(k)$ de que el j-ésimo modo sea el correcto de acuerdo a

$$\mu_j(k) = \frac{1}{c} \Lambda_j(k) \overline{c}_j \qquad j = 1, \dots r \tag{2.3.2.6}$$

   siendo c un factor de normalización

$$\overline{c} = \sum_{j=1}^{r} \Lambda_i \overline{c}_j \qquad j = 1, \dots r$$

5. Combinación: Para obtener un valor final combinado a efectos de salida del filtro IMM se combinan los estimadores arrojados por cada uno de los filtros y su covarianza asociada de acuerdo a sus probabilidad de ser correctos.

$$\hat{x}(k) = \sum_{j=1}^{r} \hat{x}_j(k) \mu_j(k) \tag{2.3.2.7}$$

$$P(k|k) = \sum_{j=1}^{r} \mu_j(k) \{P_j(k) + [\hat{x}_j(k) - \hat{x}(k)][\hat{x}_j(k) - \hat{x}(k)]^T\} \qquad j = 1, \dots r \tag{2.3.2.8}$$

El algoritmo IMM tiene tres propiedades que son deseables: es recursivo, modular, y los requisitos computacionales son fijos para cada ciclo.

Observaciones

La elección de los filtros correspondientes a los modelos de movimiento del objetivo y la adecuación de sus parámetros es un problema difícil de resolver.

La robustez del filtro IMM se consigue a expensas de la precisión en la estimación. Por ejemplo, si uno de los filtros se adapta de forma exacta al modo de movimiento del blanco, la estimación se ve afectada por la influencia de los demás filtros, lo que produce estimaciones mas pobres.

La elección del modelo de extrapolación en cada estado, junto con la matriz de transición de probabilidades, son los elementos claves en el diseño de un filtro IMM que determinan sus prestaciones finales.

Capítulo 3

Modelos para Seguimiento en Aeropuerto

A la hora de diseñar el filtro de estimación IMM es importante escoger los modelos dinámicos que describan de la manera más precisa los movimientos de los blancos que son sujetos del seguimiento, y conocer los modelos de las mediciones así como las perturbaciones (ruidos) que se introducen en el sistema.

En este capítulo hacemos un análisis de escenarios (tipos de movimientos que realizan los vehículos, sensores que los detectan, ...) que se dan en tres aeropuertos españoles y utilizaremos esta información para describir los modelos que nos serán de utilidad para representar los movimientos observados. Para terminar se explica cómo se modelan las detecciones de los radares SMR y SSR que se emplearán en nuestra simulación y los errores en las mediciones.

3.1. Datos Reales de Aeropuertos

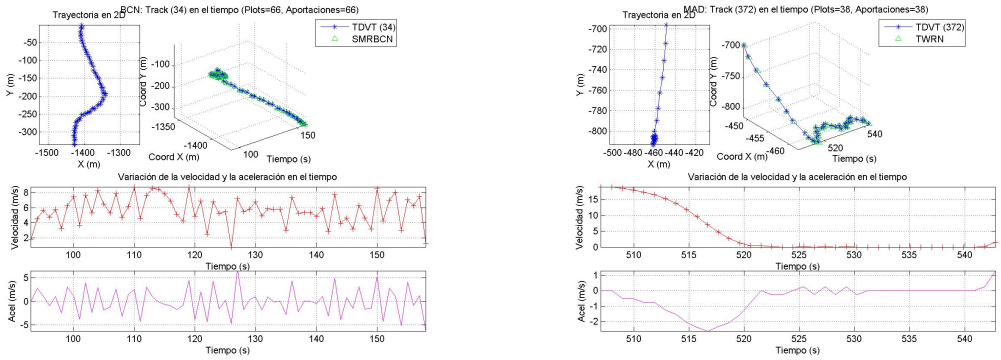
Para conocer los escenarios que se pueden dar en aeropuertos reales (tipo de radares, de vehículos, trayectorias, valores típicos de la velocidad y aceleración, etc.) y adquirir un conocimiento más profundo del problema a resolver, se han estudiado datos facilitados por Indra, procedentes del sistema de vigilancia en tierra instalado en los aeropuertos de Barcelona, Palma y Barajas.

Se ha realizado un análisis de tres ficheros con datos procesados por el sistema de seguimiento radar de los tres aeropuertos, durante 16, 3 y 21 minutos respectivamente, que contienen información de blancos detectados por radares, asignados a pistas, y los datos centrales enviados a torre de control en esos periodos de tiempo.

Cada registro del fichero incluye el tiempo de recepción, la pista radar, posición en coordenadas cartesianas, velocidad calculada por el sistema, el nombre del radar (puede haber registros vacíos) y si se trata de un registro maestro (el que se envía al sistema de presentación), además de otra información que a nosotros no nos concierne.

Para cada pista/registro de salida hacia el sistema de presentación *pista TDVT* que aparece en el fichero, los registros que le siguen son los datos de los radares que aportaron información ese instante de tiempo. De todas las informaciones que llegan para una actualización, sólo se utilizan los datos de uno de esos registros para actualizar la cinemática del blanco, que es el correspondiente a la denominada *pista maestra*, que aparece marcado en uno de los campos como tal.

El procesado de estos archivos de datos nos permite obtener rutas reales y datos cinemáticos en el tiempo y valorar cómo afectan los datos de los distintos radares a la información de salida hacia



(a) Desplazamiento en Barcelona

(b) Maniobra de parada en Madrid

Figura 3.1: Trayectorias, velocidades y aceleraciones de vehículos en aeropuertos

el centro de control (ej, figura 3.1a, en la que se presentan los datos de un vehículo circulando por el aeropuerto a una velocidad media de unos $6m/s$ durante unos minutos y reduciendo la velocidad al final, y 3.1b, en la que se observa la maniobra de parada de un vehículo que circulaba por el aeropuerto a $20m/s$, siendo la aceleración media durante la maniobra de $-1m/s^2$).

En la figura 3.2 se pueden ver los datos de varios vehículos circulando por distintas zonas del aeropuerto de Barajas. Se puede ver que las velocidades alcanzan valores de hasta $20m/s$ y existen periodos de aceleración y desaceleración con valores absolutos de aceleración comprendidos en el intervalo $[1, 5]m/s^2$

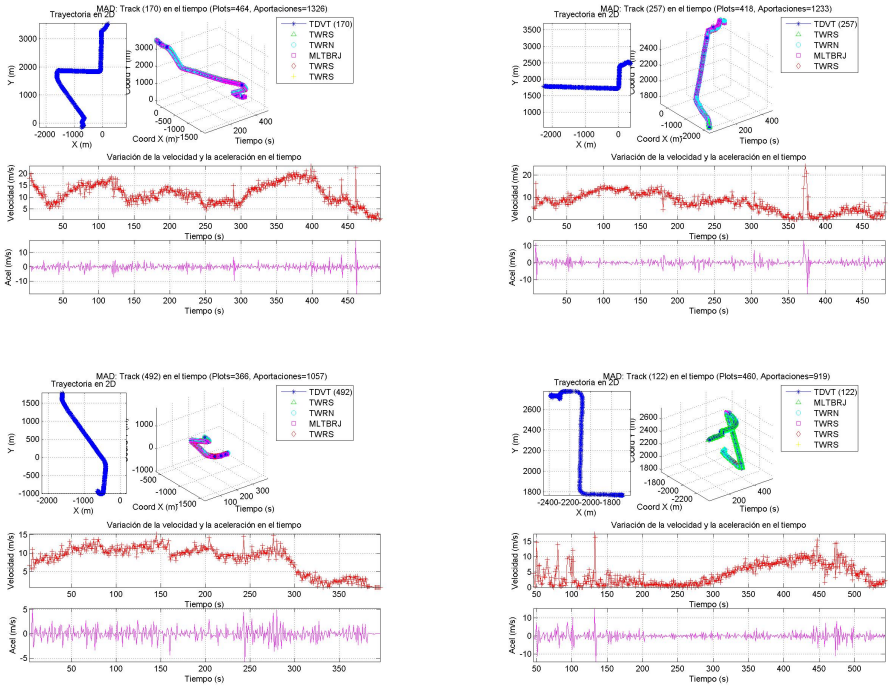
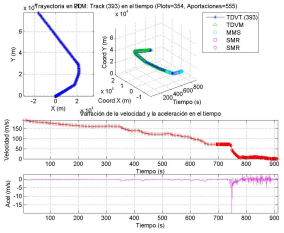
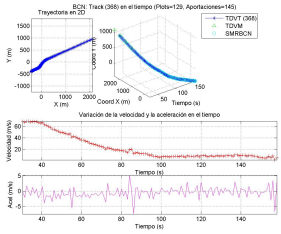


Figura 3.2: Desplazamientos de varios vehículos por distintas zonas de Barajas

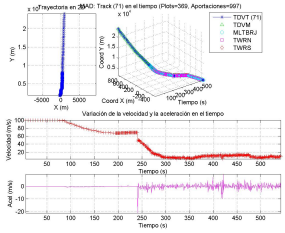
Por otro lado las figuras 3.3a, 3.3b y 3.3c representan datos reales de las maniobras finales de aterrizajes llevados a cabo en los aeropuertos de Palma, Barcelona y Madrid respectivamente. Las aeronaves reducen su velocidad hasta los 80 nudos, se alinean con la pista de aterrizaje centrada y desaceleran (con valores de aceleración de entre *-4* y *-2*m/s^2) hasta tomar tierra, para a continuación desplazarse por la pista.



(a) Datos reales de descenso y aterrizaje en Palma



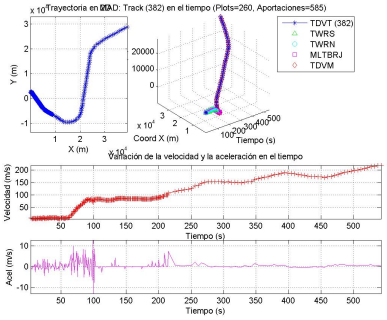
(b) Datos de Aeronave aterrizando en Barcelona



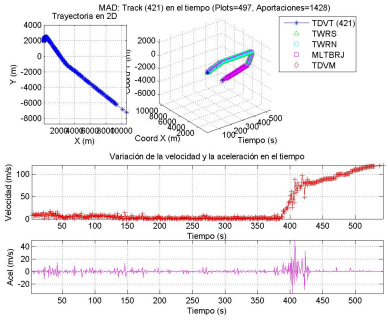
(c) Aterrizaje en Barajas, con aportaciones de radares SMR, MLAT y pistas TDMV

Figura 3.3: Ejemplos de maniobras de aterrizajes en Barcelona, Palma y Madrid

En la figura 3.4 se observan los datos detectados por varios radares del aeropuerto de Madrid durante varios minutos de las maniobras de despegue de dos aeronaves. En el primer caso, la aceleración media durante la maniobra ronda los $6m/s^2$, mientras que en el segundo, el valor promedio es de unos $12m/s^2$ durante el despegue.



(a) Aeronave 1 despegando

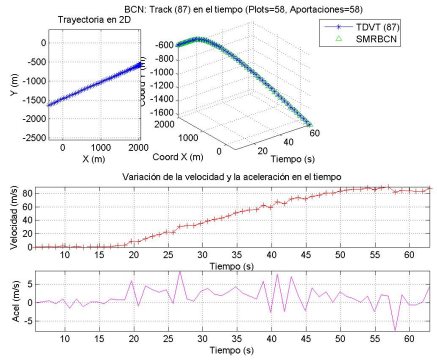


(b) Aeronave 2 despegando

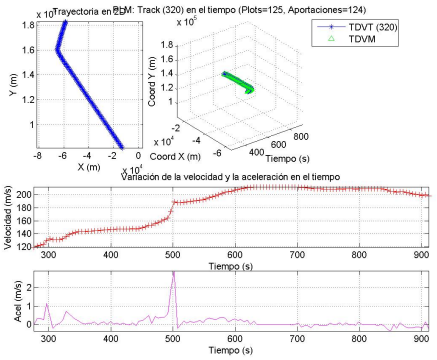
Figura 3.4: Datos de despegue de dos aeronaves en el aeropuerto de Madrid

Dependiendo de los sensores de cada aeropuerto y de la zona del mismo en la que se encuentre el blanco, hay instantes en los que, para una pista central, sólo se recibe información de un sensor y directamente con esa se actualiza la pista (ejemplo fig. 3.5), e instantes en los que dos o más datos de posición llegan a la vez para actualizar la misma pista. Estos son los casos en los que se aplican los criterios de prioridad específicos.

En la figura 3.5 podemos ver representados la trayectoria y los datos cinemáticos de una aeronave durante el despegue en el aeropuerto de Barcelona con pistas actualizadas a partir de un solo radar de superficie y de una aeronave realizando una maniobra de aceleración en el espacio aéreo de aeropuerto de Palma, detectada por su sistema de vigilancia de aire *(SURVA)*. Las figuras 3.6 muestran la información, detectada por un radar SMR, de vehículos desplazándose a velocidad cuasi-constante y en realizando maniobras de giro, aceleración y desaceleración en la superficie aeropuerto de Barcelona.

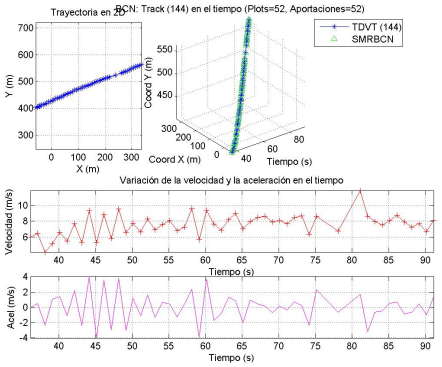


(a) Despegue en BCN detectado por SMR

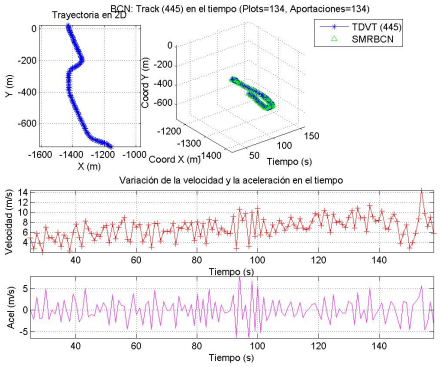


(b) Aceleración en Aire en Palma detectado por SSR

Figura 3.5: Maniobras detectadas por un único radar



(a) Aceleración/Desaceleración



(b) Varias maniobras

Figura 3.6: Vehículos maniobrando en superficie

3.1.1. Valores Cinemáticos

Analizando estos datos nos hemos podido hacer una idea de cuáles son los valores *típicos* de las velocidades y aceleraciones de los distintos tipos de maniobras que se pueden dar en los aeropuertos y estableceremos que los valores cinemáticos de las operaciones en aeropuertos que trataremos serán semejantes a los de la tabla 3.1.

Cuadro 3.1: Valores cinemáticos límite de operaciones en aeropuerto

Patrón de movimiento	Variable	Márgenes
Aproximación Final	Velocidad	80 m/s
y Pistas	Acel. Longitudinal	[-2,2] m/s^2
Giros	Velocidad	[5,15] m/s.
	Acel.Transversal	[0,4] m/s^2
Despegues	Aceleración	[6,12] m/s^2.
Stop and Go	Acel. Longitudinal	[-1,1] m/s^2
Estacionamiento	Groundspeed	[0,10]m/s

3.1.2. Resumen de los tipos de radar

De lo observado se desprende que los tipos de sensores que aportan información al sistema de vigilancia de los aeropuertos estudiados son:

Palma Un radar de movimiento en superficie (denominado SMR), y pistas de aire (TDVM).

Barcelona Un radar de movimiento en superficie (SMRBCN) y pistas de aire (TDVM)

Madrid Dos radares SMR, uno emplazado en Torre Norte (TWRN) y otro en la Torre Sur (TWRS), un sistema MLAT (MLRBRJ) y pistas de aire (TDVM)

3.2. Modelos de Movimiento en Superficie de Aeropuerto

Los desplazamientos en superficie se supone que se realizan sobre un modelo de tierra plana. A continuación se describen los modelos de movimiento horizontal en dos dimensiones que consideramos adecuados para aplicar a nuestro diseño.

3.2.1. Modelos de Movimiento en Superficie

Para definir un modelo planar, el vector de estado es de al menos cuatro dimensiones. La elección de las componentes de estado (y por ende su modelo cinemático) no es un problema trivial. Como exponen [9, 16], a partir las ecuaciones cinemáticas de modelo estándar de movimiento curvilíneo se pueden proponer varios modelos cinemáticos para hacer tracking de un blanco moviéndose en el plano horizontal:

$$\begin{aligned} \dot{x}(t) &= V(t)cos\phi(t) \\ \dot{y}(t) &= V(t)sin\phi(t) \\ \dot{V}(t) &= a_t(t) \\ \dot{\phi}(t) &= \frac{a_n(t)}{V(t)} \end{aligned}$$

Siendo $x = (x, y)$ la *posición* del blanco en coordenadas cartesianas, V la *velocidad*, ϕ el *ángulo de dirección* (heading), a_t la *aceleración tangencial* (a lo largo de la trayectoria) y a_n la *aceleración normal* (perpendicular).

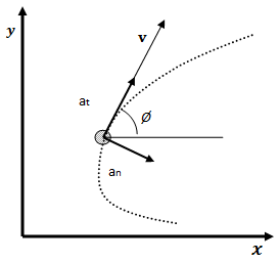


Figura 3.7: Geometría de Movimiento en 2D

Estas son las ecuaciones genéricas y se reducen a los siguientes casos especiales:

1. $a_n = 0$, $a_t = 0$: movimiento rectilíneo a velocidad constante
2. $a_n = 0$, $a_t \neq 0$: *movimiento rectilíneo acelerado* (CA, cuando a_t es constante)

3. $a_n \neq 0$, $a_t = 0$: *movimiento circular a velocidad constante.* (Cuando a_n es constante se conoce como giro constante, CT).

A continuación detallamos las ecuaciones discretas del movimiento a velocidad constante, aceleración constante y giro a velocidad angular constante.

3.2.1.1. Modelo para Velocidad Constante o Casi Constante

Un objeto se mueve con velocidad constante cuando su aceleración nula (es decir la segunda derivada de su posición, es cero). En la práctica la velocidad del blanco nunca es constante [5] sino que siempre sufre pequeñas variaciones, por lo que estas pequeñas aceleraciones se asocian a una variable de ruido

$$\boldsymbol{\nu}(t) = \begin{bmatrix} \nu_x(t) \\ \nu_y(t) \end{bmatrix}$$

cuyas componentes se refieren a la variación en la velocidad durante el intervalo de tiempo $\triangle t$ de $\dot{x}$ e $\dot{y}$.

El vector de estado incluye la posición y la velocidad en dos dimensiones:

$$\boldsymbol{x}(t) = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}$$

.

La ecuación de estado en tiempo continuo de este modelo está dada por 3.2.1.1:

$$\dot{\boldsymbol{x}}(k) = A\boldsymbol{x}(k) + D\boldsymbol{\nu}(k) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \boldsymbol{\nu} \tag{3.2.1.1}$$

El vector $\boldsymbol{\nu}$ es un ruido blanco gaussiano con media cero ($E[\boldsymbol{\nu}] = 0$) , cuya matriz de covarianza es:

$$Q_{\boldsymbol{\nu}}(k) = E[\boldsymbol{\nu}(k)\boldsymbol{\nu}(k)^T] = \begin{pmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{pmatrix} \tag{3.2.1.2}$$

donde σ_x^2 y σ_y^2 son las desviaciones estándar de la aceleración de las componentes en el eje X e Y.

Discretizando la ecuación de estado, estableciendo $\triangle_k = t_k - t_{k-1}$, el sistema estocástico en tiempo discreto relacionado a la dinámica del blanco viene dado por 3.2.1.3:

$$\boldsymbol{x}(k) = F\boldsymbol{x}(k-1) + \Gamma_k\boldsymbol{\nu}(k-1) \tag{3.2.1.3}$$

Denotando por T el intervalo de muestreo, la matriz de transición de estado F es

$$F_{CV} = \begin{pmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.2.1.4}$$

Y la matriz de ganancia que multiplica al ruido de proceso se denota por:

$$\Gamma_{CV} = \begin{pmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \end{pmatrix}$$

La matriz de covarianza de la cantidad $\Gamma\boldsymbol{\nu}$ es:

$$Q_{CV} = E[\Gamma\boldsymbol{\nu}(k)\boldsymbol{\nu}(k)^T\Gamma^T] = \Gamma Q_\nu \Gamma^T \tag{3.2.1.5}$$

- Este modelo es adecuado para seguimiento de blancos maniobrando lentamente.
- Se utiliza para cubrir pequeñas aceleraciones, rozamientos, etc. con una covarianza apropiada Q , que es un **parámetro de diseño**.
- Una trayectoria rectilínea casi uniforme, **Modelo CV**, se obtiene eligiendo una intensidad de ruido $\sigma_{\nu_x} \equiv \sigma_{\nu_y}$ pequeña en el siguiente sentido: los cambios en la velocidad, que son del orden de σ_ν han de ser pequeños comparados con la velocidad real.
- El modelo de segundo orden es más conveniente cuando se trata con intervalos de muestreo variables.

3.2.1.2. Modelo de Aceleración Constante (CA)

El movimiento de un objeto con aceleración constante se da al establecer la tercera derivada de la posición en cero. Al igual que ocurre con el caso de la velocidad constante, en la práctica la aceleración nunca es perfectamente constante, y se pueden modelar sus cambios para obtener resultados fiables por medio de un ruido blanco continuo $\boldsymbol{\nu}$ con media cero .

El vector de estado de este modelo, también conocido como de tercer orden, consta la posición, velocidad y aceleración en dos dimensiones.

$$\boldsymbol{x}(t) = \begin{bmatrix} x & \dot{x} & \ddot{x} & y & \dot{y} & \ddot{y} \end{bmatrix}^T \tag{3.2.1.6}$$

La ecuación de estado en tiempo continuo de este modelo está dada por 3.2.1.7:

$$\dot{x}(k) = A\boldsymbol{x}(k) + D\boldsymbol{\nu}(k) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \boldsymbol{\nu} \tag{3.2.1.7}$$

Cada componente del vector $\boldsymbol{\nu}(t)$ es un ruido blanco gaussiano con media cero, representando incrementos en la aceleración, que sirve para ajustar el modelo al movimiento del objeto de interés, y cuya matriz de covarianza es:

$$Q_\nu = \begin{pmatrix} \sigma^2_{v_x} & 0 \\ 0 & \sigma^2_{v_y} \end{pmatrix} \tag{3.2.1.8}$$

Discretizando la ecuación de estado (ver [16]), estableciendo $\triangle_k = t_k - t_{k-1}$, el sistema estocástico en tiempo discreto relacionado a la dinámica del blanco viene dado por 3.2.1.3

Para un movimiento en 2D las matrices quedan como sigue:

La matriz de transición de estados F :

$$F_{CA} = \begin{pmatrix} 1 & T & T^2/2 & 0 & 0 & 0 \\ 0 & 1 & T & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & T & T^2/2 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.2.1.9}$$

La matriz de ganancia del ruido Γ:

$$\Gamma_{CA} = \begin{pmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 1 & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \\ 0 & 1 \end{pmatrix} \tag{3.2.1.10}$$

Como en el modelo de CV, la matriz de covarianza del error de proceso viene dada por la expresión:

$$Q = E[\Gamma \boldsymbol{\nu}(k)\boldsymbol{\nu}(k)^T \Gamma^T] = \Gamma q_{\boldsymbol{\nu}} \Gamma^T$$

La covarianza del ruido de proceso se simplifica, se asume que la varianza del ruido de proceso es la misma para cada coordenada, y constante $\sigma^2_{v_x} \equiv \sigma^2_{v_y} = q$; q puede escogerse utilizando la siguiente desigualdad:

$$0.5 \triangle a_{max} \leq \sqrt{q} \leq \triangle a_{max}$$

donde $\triangle a_{max}$ es el incremento máximo de la aceleración durante el intervalo de muestreo.

- Un movimiento de *aceleración casi constante* se obtiene escogiendo valores de la varianza q del ruido pequeños. Los cambios en la aceleración deberían ser pequeños en relación a los niveles de aceleración reales.
- Este modelo es adecuado para hacer seguimiento de*blancos realizando maniobras* y es modelo que se usa con más frecuencia.

3.2.1.3. Modelos de Giros

Es el movimiento de un blanco haciendo un giro en el plano (O, x, y) y se caracteriza por las siguientes ecuaciones de movimiento.

$$\begin{aligned} \ddot{x}(t) &= -\omega \dot{y} \\ \ddot{y} &= -\omega \dot{x} \end{aligned}$$

Cuando $\omega > 0$ el giro es en el sentido de las agujas del reloj. Cuando $\omega < 0$ el giro es en el sentido contrario al de las agujas.

Caracterizado por una tasa de giro ω (casi)constante y el módulo de la velocidad (casi)constante $V = \sqrt{\dot{x}^2 + \dot{y}^2}$ (si la tasa angular de giro no es constante, se necesita un modelo no lineal) [16].

Para los modelos de giros, la elección de los vectores de estado no es trivial, y existen dos propuestas que se diferencian en la representación del vector de velocidad: en coordenadas cartesianas y en polares.

En este caso hemos elegido la representación en coordenadas cartesianas

Modelo Giro Constante (CT)

Si se conoce la velocidad angular ω, el vector de estado tendrá como componentes la posición y la velocidad en 2D [[9]],

$$\boldsymbol{x}(t) = \begin{bmatrix} x & \dot{x} & y & \dot{y} \end{bmatrix}^T$$

donde x e y representan la posición del objetivo, mientras que $\dot{x}$ y $\dot{y}$, representan sus velocidades respectivas.

El modelo de espacio de estados en tiempo continuo que caracteriza el movimiento CT es

$$\dot{\boldsymbol{x}}(k) = \begin{bmatrix} \dot{x} \\ -\omega\dot{y} \\ \dot{y} \\ \omega\dot{x}) \end{bmatrix} (k) + \boldsymbol{\nu}(k) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \omega \\ 0 & 0 & 0 & 1 \\ 0 & \omega & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} + \boldsymbol{\nu}(k) \tag{3.2.1.11}$$

Este modelo es lineal porque la tasa del ángulo de giro ω es conocida. Discretizando 3.2.1.11, la ecuación en tiempo discreto que define la dinámica del movimiento, teniendo en cuenta el ruido de estado en las componentes de la velocidad, que se supone blanco Gaussiano es:

$$\boldsymbol{x}(k) = F_{CT}(k-1)\boldsymbol{x}(k-1) + \Gamma_k\boldsymbol{\nu}(k-1) \tag{3.2.1.12}$$

con la siguiente matriz de transición de estados:

$$F_{CT} = \begin{pmatrix} 1 & \frac{(sin(\omega*T))}{\omega} & 0 & \frac{-(1-cos(\omega*T))}{\omega} \\ 0 & cos(\omega*T) & 0 & -sin(\omega*T) \\ 0 & \frac{(1-cos(\omega*T))}{\omega} & 1 & \frac{(sin(\omega*T))}{\omega} \\ 0 & sin(\omega*T) & 0 & cos(\omega*T) \end{pmatrix} \tag{3.2.1.13}$$

La matriz de ganancia de ruido Γ

$$\Gamma_{CT} = \begin{pmatrix} \frac{1}{2}T^2 & 0 \\ T & 0 \\ 0 & \frac{1}{2}T^2 \\ 0 & T \end{pmatrix} \tag{3.2.1.14}$$

El ruido de proceso $\boldsymbol{\nu}(k) = [\nu_x(k)\nu_y(k)]'$ incluye términos de ruido para las aceleraciones en x y en y, es utilizado para modelar las perturbaciones en la trayectoria que la alejan de un movimiento CT ideal. Como se dijo, se asume ruido blanco gausiano de media cero y su covarianza es:

$$Q_\nu = E[\boldsymbol{\nu}(k)\boldsymbol{\nu}(k)^T] \tag{3.2.1.15}$$

En las escasas ocasiones en las que la velocidad de giro es (aproximadamente) conocida a priori, este modelo ofrece buenos resultados. Pero la necesidad de conocer dicho valor, hacen que sea poco realista su utilización en la mayoría de los casos prácticos.

Una forma de remediarlo es reemplazar el valor de la velocidad angular con una estimación, basándose en los últimos resultados de la velocidad, aunque esto convierte al modelo en un modelo no lineal (por lo que no se podría utilizar en algunos filtros tales como Kalman)

Otra solución consiste en utilizar varios modelos con diferentes velocidades angulares. Este enfoque mejora los resultados y a la vez preserva la linealidad del modelo, pudiendo aplicarlo en la mayoría de los filtros.

Modelo CT con Tasa de Giro Desconocida

Este modelo se diferencia del anterior en que la velocidad angular no se conoce; el vector de estado tiene cinco componentes, ya que ésta se incluye como una componente a ser estimada más.

$$\boldsymbol{x}(t) = \begin{bmatrix} x & \dot{x} & y & \dot{y} & \omega \end{bmatrix}^T \tag{3.2.1.16}$$

Y hay que incrementar la dimensión de las matrices de transición de estados y la ganancia:

$$F(k) = \begin{pmatrix} 1 & \frac{(sin(\omega * T))}{\omega} & 0 & \frac{-(1-cos(\omega * T))}{\omega} & 0 \\ 0 & cos(\omega * T) & 0 & -sin(\omega * T) & 0 \\ 0 & \frac{(1-cos(\omega * T))}{\omega} & 1 & \frac{(sin(\omega * T))}{\omega} & 0 \\ 0 & sin(\omega * T) & 0 & cos(\omega * T) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.2.1.17}$$

$$\Gamma(k) = \begin{pmatrix} \frac{T*T}{2} & 0 & 0 \\ T & 0 & 0 \\ 0 & \frac{T*T}{2} & 0 \\ 0 & T & 0 \\ 0 & 0 & T \end{pmatrix} \tag{3.2.1.18}$$

En este caso la matriz F se evaluará en cada paso, usando la velocidad angular estimada $\hat{\omega}$ en el paso anterior del filtro.

3.3. Modelo Matemático de los Radares

Como se ha explicado en la introducción, para poder hacer seguimiento de un blanco, hay que extraer toda la información posible acerca de su estado a partir de las observaciones, por lo que además de modelar la dinámica del blanco, es necesario modelar el sensor.

Entre los *sensores* que aportan observaciones al sistema de seguimiento del aeropuerto están los *radares* SMR y SSR, que miden la posición de los blancos y proporcionan el rango r y el *azimut* (ver fig. 3.8), en coordenadas polares.

Las observaciones del radar en el sistema de coordenados del sensor se modelan con el consiguiente ruido aditivo según la expresión 3.3.0.19.

$$\boldsymbol{z}_k^p = \begin{bmatrix} r \\ \theta \end{bmatrix} = \begin{bmatrix} \hat{r} \\ \hat{\theta} \end{bmatrix} + \begin{bmatrix} \nu_r \\ \nu_\theta \end{bmatrix} \tag{3.3.0.19}$$

Siendo $\boldsymbol{x}_k^p = \begin{bmatrix} \hat{r} \\ \hat{\theta} \end{bmatrix}$ la posición de blanco en coordenadas polares respecto al radar y $\boldsymbol{\nu}_k^p = \begin{bmatrix} \nu_r \\ \nu_\theta \end{bmatrix}$ el vector de error compuesto por los respectivos errores de medida, que se asumen gaussianos, de media cero y no-correlados:

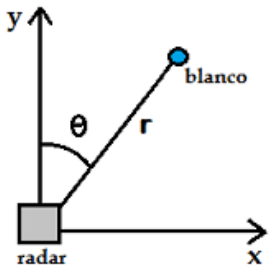


Figura 3.8: Sistema de Coordenadas del Radar

$$\nu_k^p \sim N(0, R_k^p)$$

con matriz de covarianza:

$$R_k^p = cov(\nu_k^p) = diag(\nu_r^2, \nu_\theta^2) = R^p, \forall k$$

3.3.1. Conversión a Coordenadas Cartesianas

En simulación, la trayectoria predeterminada del blanco se convierte a coordenadas polares, y luego de vuelta a coordenadas cartesianas después de mezclar con el ruido de medida. Varios autores ([4] y [11]) explican el proceso de transformación que resumimos a continuación:

Se define $\boldsymbol{x}_k^c = \begin{bmatrix} x \\ y \end{bmatrix}$ como la posición del blanco en coordenadas cartesianas correspondiente al par $\boldsymbol{x}_k^p = \begin{bmatrix} r \\ \theta \end{bmatrix}$.

$$\boldsymbol{x}_k^c = \begin{bmatrix} x \\ y \end{bmatrix} = \phi(\boldsymbol{x}_k^p) = \phi(\hat{r}, \hat{\theta}) = \begin{bmatrix} \hat{r}\cos(\hat{\theta}) \\ \hat{r}\sin(\hat{\theta}) \end{bmatrix} \tag{3.3.1.1}$$

donde ϕ es la transformación de polares a cartesianas.

Realizando esta conversión a coordenadas cartesianas, las observaciones del radar se pueden modelar como:

$$\boldsymbol{z}_k^c = H\boldsymbol{x}_k + \boldsymbol{\nu}_k^c \tag{3.3.1.2}$$

Donde $\boldsymbol{x}_k^c = H\boldsymbol{x}_k = \begin{bmatrix} x \\ y \end{bmatrix}$ es la posición del blanco en coordenadas cartesianas (un subvector del vector de estado $\boldsymbol{x}$) , y $\boldsymbol{\nu}_k^c$ es el ruido de medida convertido a coordenadas cartesianas. La matriz H es ahora invariante en el tiempo .

El mayor reto de la ecuación 3.3.1.2 es obtener una conversión linealizada del ruido de medida, ya que $\boldsymbol{\nu}_k^c$ normalmente está acoplado, es no gaussiano y dependiente del estado y esto provoca que, al utilizarlo en el filtro Kalman, los resultados obtenidos no sean óptimos. Si bien, la presencia de $\boldsymbol{\nu}$ no lineal es mucho menos significativo que utilizar una función de medida no lineal.

Como se manifiesta en [11], uno de los enfoques utilizados es realizar el desarrollo de taylor de $\phi(x_k^p)$ alrededor de la observación ruidosa $\boldsymbol{z}_k^p$

$$x^c = \phi(x^p) = \phi(z^p - v^p) = \phi(z^p) - J(z^p)v^p + \xi(v^p) \tag{3.3.1.3}$$

donde ξ representa los términos de orden ≥ 2 y $J(z^p)$ es el Jacobiano dado por:

$$J(z^p) = \frac{\partial \Phi}{\partial x^p}|_{x^p=z^p} = \begin{pmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{pmatrix} \tag{3.3.1.4}$$

Entonces, el término ν_k^c de la ecuación 3.3.1.2 se puede escribir como:

$$\boldsymbol{\nu}_k^c = J(z_k^p)\boldsymbol{\nu}_k^p + \xi(v_k^p) \tag{3.3.1.5}$$

el planteamiento mas común, aunque consiste en tratar ν_k^c como una secuencia de media cero y covarianza:

$$R_k = J(z_k^p)R_k^p J(z_k^p)' \tag{3.3.1.6}$$

Lo que se consigue ignorando los términos de mayor orden en el desarrollo de taylor $\xi(v_k^p)$

$$\boldsymbol{z}_k = \Phi(z_k^p) \approx Hx_k + J(z_k^p)\nu_k^p \tag{3.3.1.7}$$

donde $\boldsymbol{\nu}_k^c$ se aproxima mediante la versión linealizada

$$\boldsymbol{\nu}_k = J(z_k^p)\nu_k^p$$

La linealización que se ha descrito es considerada válida cuando se cumple que (ver [10]):

$$r\sigma_\theta^2/\sigma_r \leq 0.4 \tag{3.3.1.8}$$

Capítulo 4

Simulación de Escenarios Aeroportuarios

Utilizando Matlab como entorno de desarrollo y aplicando el paradigma de la *Programación Orientada a Objetos* (POO) se ha desarrollado una aplicación para simular trayectorias basadas en las operaciones fundamentales que realizan los vehículos en la superficie de los aeródromos.

En este capítulo ofrecemos una breve descripción de cómo se realiza la simulación: los elementos emulados (radares), las operaciones y escenarios (stop&go, rodaje, despegue . . .) que se reproducen y la forma en la que se inyectan los datos en la simulación. El código fuente de las clases que encarnan las distintas piezas que conforman nuestro simulador se puede consultar en el apéndice A.

4.1. Simulación de Sensores

Los tipos de sensores que suministrarán detecciones al sistema de vigilancia y cuyos mecanismos de medición reproduciremos para alimentar los filtros de estimación serán:

- *Radar de Movimiento en Superficie* (SMR): que proporciona la posición detectada del blanco en coordenadas polares
- *Pistas del sistema de vigilancia en aire* (TDVM) que corresponden a datos detectados por radares secundario de aproximación SSR (aportan la posición detectada del blanco en coordenadas polares)
- *Sistema de Multilateración* (MLAT), *MMS*, que proporciona la posición detectada del blanco en coordenadas cartesianas

Las medidas de los radares que proporcionan rango y azimut, se transforman a coordenadas cartesianas en el plano de la superficie del aeropuerto, utilizando el método de transformación de coordenadas no lineal explicado en la sección 3.3.

Inspirándonos en los sensores desplegados en Barajas que analizamos en la sección 3.1, para realizar la simulación se plantea un escenario semejante utilizando cuatro sensores con las particularidades y la disposición que se indica en la tabla 4.1.

Cuadro 4.1: Sensores Simulados

Sensor	Posición	Periodo	Error	Cobertura
SMR1	(-900, 400) m	1s	$\sigma_r = 5m$, $\sigma_\theta = 0.15^o$	5 Km
SMR2	(-600, 3700) m	1s	$\sigma_r = 5m$, $\sigma_\theta = 0.15^o$	5 Km
MMS	(0,0)	1s	$\sigma_x = 5m$, $\sigma_y = 5m$	6 Km(Todo el Aeropuerto)
ASR	(10000, 0) m	5 s	$\sigma_r = 10m$, $\sigma_\theta = 0.09^o$	100 Km

4.2. Simulación de Trayectorias

4.2.1. Simulación Categorías de movimiento

Las trayectorias de los blancos de definen mediante composición de segmentos que describen modos básicos de movimiento, y que modelan el movimiento. El estado cinemático inicial de cada trayectoria incluye:

- Posición en X e Y
- Módulo de la velocidad
- Aceleración
- Ángulo o heading.

Lo segmentos son definidos mediante un conjunto de parámetros, y los movimientos deben ser descompuestos en una colección de segmentos a aceleración constante, segmentos a lo largo de una trayectoria, movimiento transversal, ... Los tipos de segmentos se enumeran en la tabla 4.2 de forma más detallada.

Tipo	Descripción	Parámetros
Velocidad Constante	Sin aceleración en ninguna coordenada.	velocidad, ángulo inicial, distancia
Giro	Aceleración transversal	Ángulo de giro, velocidad angular
Parada	Desaceleración longitudinal hasta alcanzar velocidad cero.	Velocidad final (0), aceleración
Detenido	Sin aceleración ni velocidad	Tiempo que estará parado
Arranque	Aceleración longitudinal desde velocidad cero	Velocidad final, aceleración hasta alcanzar esa velocidad.
Cambio de velocidad	Aceleración/desaceleración hasta un valor diferente de la velocidad.	Velocidad final, aceleración hasta alcanzar esa velocidad.

Cuadro 4.2: Tipos de Segmentos de Movimiento

La posición,velocidad y aceleración del blanco serán calculados integrando estos modos de forma consecutiva. Al final de cada segmento seremos capaces de calcular los valores iniciales del segmento siguiente.

4.2.2. Proceso de Simulación de Operaciones Aeroportuarias

Comenzamos describiendo cómo se simularán los escenarios simples que comprenden las operaciones básicas que se realizan en la superficie aeroportuaria y que incluyen: velocidad constante, stop&go, rodaje, despegue y aterrizaje.

4.2.2.1. Velocidad Constante

La trayectoria completa comenzará en una posición dada, con velocidad y ángulo que forma la trayectoria con respecto eje horizontal del sistema de coordenadas cartesianas. Se compondrá de los siguientes segmentos 4.3:

Velocidad Constante		
N	Descripción	Parámetros
1	Velocidad constante	metros

Cuadro 4.3: Descripción Trayectoria a Velocidad Constante por Segmentos

4.2.2.2. Stop&Go

La trayectoria completa comenzará en una posición dada, con velocidad y ángulo determinados. Se compondrá de los segmentos listados en la tabla 4.4:

Stop&Go		
N	Descripción	Parámetros
1	Velocidad constante	metros
2	Parada	$-m/s^2$ hasta velocidad cero
3	Detenido	segundos
4	Arranque	m/s^2, hasta $v = m/s$
5	Velocidad constante	metros

Cuadro 4.4: Descripción Stop-and-Go por Segmentos

4.2.2.3. Despegue

La trayectoria completa comenzará en una posición dada, con velocidad y ángulo determinados. Se compondrá de los segmentos listados en la tabla 4.5:

Despegue		
	Descripción	Parámetros
1	Detenido	segundos
2	Arranque	m/s^2, hasta $v = m/s$
3	Velocidad constante	metros
4	Cambio de Velocidad (aceleración)	m/s^2, hasta v= m/s

Cuadro 4.5: Maniobra Despegue por Segmentos

4.2.2.4. Aterrizaje

La trayectoria completa comenzará en una posición dada, con velocidad y ángulo determinados. Se compondrá de los segmentos listados en la tabla 4.6:

Stop&Go		
N	Descripción	Parámetros
1	Velocidad Constante	
2	Cambio de Velocidad (desaceleración)	$-m/s^2$, hasta $v = m/s$
3	Velocidad Constante	metros
4	Giro (pequeño, derecha)	grados, aceleración angular
5	Velocidad Constante	metros
6	Parada	$-m/s^2$
7	Detenido	segundos

Cuadro 4.6: Descripción Maniobra Aterrizaje por Segmentos

4.2.2.5. Rodaje

La trayectoria completa comenzará en una posición dada, con velocidad cero y un ángulo determinado respecto al eje de abscisas. Se compondrá de los segmentos listados en la tabla 4.7:

Rodaje		
	Descripción	Parámetros
1	Detenido	segundos
2	Arranque	m/s^2, hasta $v = m/s$
3	Velocidad Constante	metros
4	Giro (pequeño, derecha)	grados, aceleración angular
5	Velocidad Constante	metros.
6	Giro (pequeño, izquierda)	grados, aceleración angular
7	Velocidad Constante	metros
8	Giro (grande, derecha)	grados, aceleración angular
9	Velocidad Constante	metros.
10	Giro (grande, izquierda)	grados, aceleración angular
11	Velocidad Constante	metros
12	Parada	$-m/s^2$
13	Detenido	segundos
14	Arranque	m/s^2, hasta $v = m/s$
15	Velocidad Constante	metros
16	Cambio de Velocidad (desaceleración)	$-m/s^2$, hasta v= m/s
17	Velocidad Constante	metros
18	Cambio de Velocidad (aceleración)	m/s^2, hasta $v = m/s$
19	Velocidad Constante	metros

Cuadro 4.7: Descripción de Rodaje por Segmentos

4.2.3. Escenarios Complejos

Para analizar el comportamiento de los filtros de estimación se simularán (basándonos en los datos reales observados en los aeropuertos de Barajas y Barcelona) varios recorridos completos que podrían efectuar aeronaves para realizar operaciones de despegue y aterrizaje, desde y hasta el aparcamiento respectivamente, en un aeropuerto. Se aplicarán valores típicos de velocidad y aceleración para los aviones en aproximación y superficie, y trayectos de vehículos circulando por el aeropuerto.

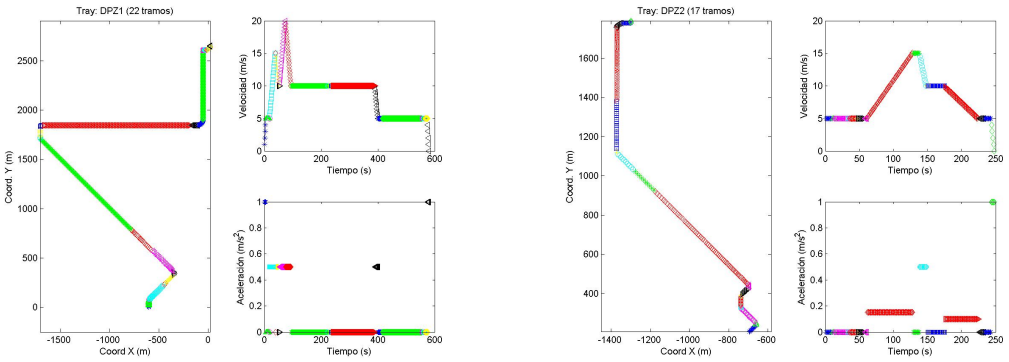
A continuación describimos de forma general cuales son los escenarios representados. La especificación completa de las trayectorias simuladas, en forma de sucesión de segmentos, se pueden encontrar en el apéndice B.

4.2.3.1. Desplazamientos en el Aeropuerto

En los desplazamientos en el aeropuerto vamos a simular:

Vehículos que parten de estar parados y aceleran durante unos segundos hasta alcanzar velocidades tipo comprendidas entre 5 y 20 m/s, ejecutan maniobras con aceleraciones (desaceleraciones) entre $[0.1, 1]m/s^2$ y realizan giros de distinta amplitud, con tasas de giro de una media de $10^o/s$

DPZ1 simula una trayectoria similar a la realizada por la pista identificada como 170 (fig. 3.2) por el sistema de radar emplazado en el aeropuerto de Barajas, en la que el vehículo realiza maniobras de aceleración y deceleración suaves (del orden de $\pm 0.5m/^2$) con velocidades variando entre 0 y 20 m/s; con giros de 45 y 90 grados a velocidad angular de unos $0.4rad/seg$. Termina ejecutando una maniobra de parada.



(a) Desplazamiento 1

(b) Desplazamiento 2

Figura 4.1: Ejemplos de datos reales de vehículos en aeropuertos

En la figura 4.1a podemos ver los 22 tramos que constituyen la trayectoria.

DPZ2 Trayectoria similar a la realizada por la pista 356 3.2 en aeropuerto de Barajas.

El vehículo se mueve a una velocidad constante de 5, 10 y 15 m/s, realizando una maniobra de aceleración constante a $0.2m/s^2$ y dos de desaceleración constante con valores de aceleración de -0.5, -0.1 realizando varios giros de 45 y 90 grados (tanto a la izquierda como a la derecha) y realizando una parada final (aceleración constante a $-1m/s^2$)

En la figura 4.1b se pueden contemplar ver los 17 tramos que constituyen la trayectoria así como los valores de la velocidad y la aceleración en cada tramo.

DPZ3 Trayectoria parecida a la realizada por la pista 122 en Barajas. Incluye arranque con aceleración longitudinal a $0.5m/s^2$ hasta los $5m/s$, tramos a velocidad constante de 5m/s realizando 4 maniobras de giros de 45, 135, 90 y 90 grados respectivamente, para después entrar en un tramo recto donde primero acelera hasta los 10m/s, continúa a velocidad constante, y realiza un giro final de 90 grados para terminar con un tramo recto y frenada

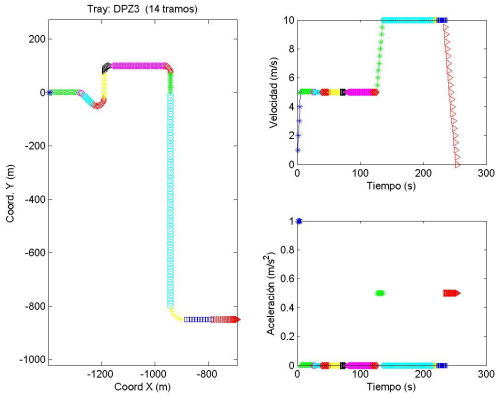


Figura 4.2: Desplazamiento por aeropuerto 3

En la figura 4.2 se pueden ver los 14 tramos que constituyen la trayectoria así como los valores de la velocidad y la aceleración en cada tramo.

4.2.3.2. Despegues

Se simulan cuatro escenarios de despegue completos, uno para cada pista de despegue del aeropuerto de Barajas (fig. 4.3), con trayectorias y valores cinemáticos conformes a los datos observados en la sección 3.1. Las operaciones desde las terminales hacia la pista de despegue que se simulan transcurren a velocidades medias de 10 m/s realizando giros (con aceleraciones transversales de $4m/s^2$ y $10m/s^2$) para seguir el trazado de las pistas y una vez en la cabecera de pista de despegue, hacen una parada (a $-1m/s^2$), arranque y aceleración fuerte para el despegue (valores de $11m/s^2$, $6m/s^2$ durante unos segundos, hasta alcanzar velocidades de vuelo. La representación de estas operaciones se pueden ver en las figuras **??** y **??**;

4.2.3.3. Aterrizajes

Simulamos dos aterrizajes en pistas del aeropuerto de Barajas utilizando los valores propios de la cinemática de los aterrizajes, que tienen la particularidad de ser maniobras con desaceleraciones fuertes. En los datos reales se observaron aeronaves aproximándose al aeropuerto a velocidades de $100m/s$ descendiendo con aceleraciones constantes de $-0.25m/s^2$, seguidas por tramos a velocidad constante de entre 60 y 80 m/s sucedidos por descensos finales de tramos a $-1m/s^2$ y finalizando en frenadas fuertes en superficie de unos -4 o $-5m/s^2$

Se van a reproducir entonces trayectorias similares a las observadas manejando valores cinemáticos equivalentes. Los recorridos de las aeronaves en tierra serán los inversos de los simulados en los despegues 4.3.

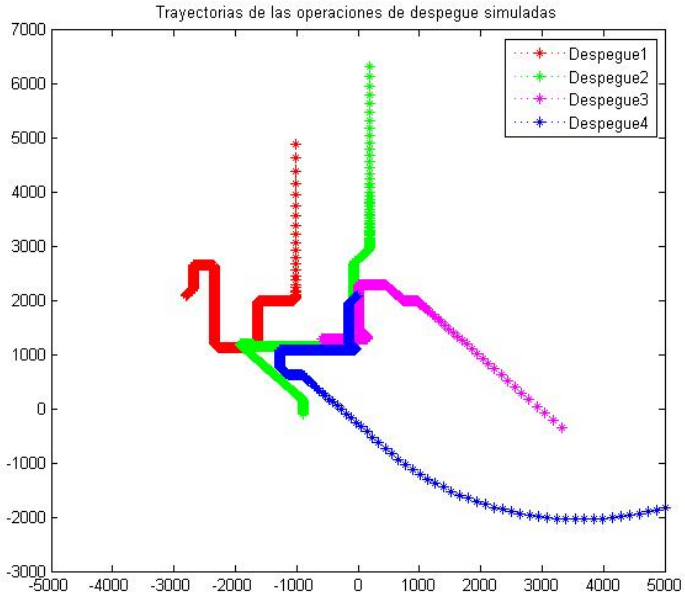


Figura 4.3: Representación en 2D de los recorridos de 4 aeronaves hasta su despegue

Capítulo 5

Filtros IMM Propuestos para Seguimiento de Radar

Tratando el problema de vigilancia en superficie como el problema en el aire, donde el movimiento del blanco no esta significativamente restringido/limitado por las carreteras, se puede utilizar un estimador *Múltiples Modelos Interactuantes* (IMM) con un número pequeño de modelos, que pueda manejar tanto blancos moviéndose a velocidad constante como realizando maniobras.

Como se expuso en la sección 2.3, la metodología de seguimiento IMM mantiene un conjunto de modelos dinámicos diferentes, cada uno ajustado a un tipo específico de patrón de movimiento (parado, acelerando, girando, frenando...), y representa la trayectoria de un blanco como una serie de estados, con una secuencia de transiciones modelada como una cadena de markov.

En este capítulo presentamos los filtros IMM que proponemos para nuestra función de vigilancia multiradar y cuyas prestaciones serán analizadas y comparadas entre sí y con el sistema de seguimiento radar actual. Explicamos brevemente el proceso seguido para diseñarlos y describimos las cuatro estructuras IMM concretas que planteamos, con sus modelos de movimiento específicos y parámetros.

5.1. Diseño de Filtros IMM para el Sistema de Seguimiento de Radar

EL diseño de un estimador IMM consiste en los siguientes pasos:

* *Selección del conjunto de modelos que describen la dinámica de los vehículos y su estructura:* La clave del algoritmo IMM es cómo seleccionar modelos que representen los modos de movimiento reales; es decir, la combinación de modelos en el IMM juega un papel muy importante en la precisión final del seguimiento.
* *Selección de la intensidad del ruido de proceso para cada modelo.*
* *Selección de las probabilidades de transición:* La elección de la matriz de transición depende del problema y de la información inicial, si hay alguna.

5.1.1. Modelos e Intensidad de Ruido de Proceso

Teniendo en cuenta que los tipos de móviles de interés son las aeronaves y vehículos de superficie, y que las operaciones en el aeropuerto se realizan en las zonas de maniobras (compuestas de

pistas y calles de rodaje adyacentes), en las plataformas donde se llevan a cabo las operaciones de carga y descarga y en las "runways glidepaths" (áreas de planeo en pista); existen diferentes tipos de movimiento que pueden modelarse con modelos dinámicos con distintos parámetros de velocidad y aceleración según el área:

- Movimiento a lo largo de un segmento: modelos de velocidad constante y de aceleración o desaceleración constante.
- Movimientos en intersecciones: modelos de giro.
- Para movimientos *stop&go*: modelos con cambio de velocidad (desaceleración), detenciones, y cambio de velocidad (aceleración).
- Aterrizaje: Modelos de cambio de velocidad con valores altos de desaceleración longitudinal.
- Despegue: Modelos de aceleración con valores de aceleración longitudinal altos.

De todo esto inferimos que los modelos dinámicos interesantes a la hora de estimar la posición y cinemática de los blancos moviéndose en la superficie del aeropuerto son:

1. Modo Velocidad Constante
2. Modo Aceleración Longitudinal Constante, (para aterrizaje , despegue, stop&go)
3. Modo de Giro, para curvas en intersecciones

Con este razonamiento y teniendo en cuenta las reflexiones de distintos autores ([14]) y el hecho de que la estimación de la velocidad angular es bastante lenta para nuestro tipo de maniobras, vamos a considerar los siguientes modelos para ajustar los filtros Kalman que integren el filtro IMM:

Modelo 1 (M1) : Modelo CV 3.2.1.1- vehículos moviéndose en linea recta a velocidad constante, con $\sigma_{v_x} = \sigma_{v_y} = q$, y $\sigma_{v_x v_y} = 0$, con valores de q *pequeños*, para modelar pequeños valores ruido (velocidad casi constante), como se describe en [14].

La ecuación que describe el sistema es, como vimos 3.2.1.3:

$$\boldsymbol{x}(k) = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}(k-1) + \begin{bmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \end{bmatrix} \boldsymbol{\nu}(k-1) \tag{5.1.1.1}$$

quedando la matriz de covarianza del error de proceso:

$$Q_{\boldsymbol{\nu}}(k) = E[\boldsymbol{\nu}(k)\boldsymbol{\nu}(k)^T] = \begin{pmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{pmatrix} \tag{5.1.1.2}$$

Modelo 2 (M2) : Modelo CV 3.2.1.1 - vehículos moviéndose en linea recta a velocidad constante. con $\sigma_{v_x} = \sigma_{v_y} = q$ y $\sigma_{v_x v_y} = 0$; con valores de q *mayores* que en modelo anterior, para introducir valores de ruido más amplios (puede modelar maniobras "bruscas"). Las ecuaciones son, para este modelo, las mismas que las de M1 (5.1.1.1 y 5.1.1.2).

Modelo 3 (M3) : Modelo CA 3.2.1.2 (modelos para cuando está acelerando o frenando) con valores de ruido q pequeños, vehículos moviéndose con aceleración casi constante. Este tipo de modelo proporciona estimación más precisa que el 2 durante una maniobra.

$$\boldsymbol{x}(k) = \begin{bmatrix} 1 & T & T^2/2 & 0 & 0 & 0 \\ 0 & 1 & T & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & T & T^2/2 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}(k-1) + \begin{bmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 1 & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \\ 0 & 1 \end{bmatrix} \boldsymbol{\nu}(k-1) \qquad (5.1.1.3)$$

Modelo 4 (M4) : Modelo CA 3.2.1.2 con valores de ruido *q altos*. Este tipo de modelo puede estimar de manera más precisa el comienzo y la finalización de maniobras. El rango de ruido debe ser del orden de la magnitud de la máxima variación de la aceleración durante un periodo de muestreo

$$0.5 \triangle a_{max} \leq q_4 \leq \triangle a_{max}$$

Modelos para cuando se realizan giros : Ya que los vehículos en los aeropuertos han de circular con unas normas y valores cinemáticos dentro de unos límites,vamos a considerar un modelo de giro que asume que la velocidad angular es un parámetro conocido en tiempo de diseño (utilizando los valores tipo de los giros en los aeropuertos). Se utilizarán modelos simétricos (con $\omega > 0$ y $\omega < 0$ para giros a la izquierda y a la derecha, respectivamente).

Para determinar el valor de la velocidad angular [9], se puede utilizar una solución basada en la utilización de múltiples modelos de giro cada uno con un valor distinto de ω, o bien obtenerla a partir de la aceleración tangencial, siendo ésta el parámetro de diseño (aunque esta solución puede incurrir no-linealidades en el sistema). Nosotros optamos por los dos modelos de giro siguientes: $\omega = \frac{a_t}{v}$

- **Modelo 5 (M5)**: Modelo CT 3.2.1.3 - vehículos moviéndose con velocidad constante y girando en el sentido de las agujas del reloj con velocidad de giro constante:

$$\omega_5 = -0.2$$

- **Modelo 6 (M6)**: Modelo CT 3.2.1.3 - vehículos moviéndose con velocidad constante y girando en el sentido contrario al de las agujas de un reloj, con velocidad de giro constante.

$$\omega_6 = 0.2$$

Las ecuaciones de sistema aplicadas en estos dos modelos corresponden a las de 3.2.1.12

$$\boldsymbol{x}(k) = \begin{bmatrix} 1 & \frac{(sin(\omega * T))}{\omega} & 0 & \frac{-(1-cos(\omega * T))}{\omega} \\ 0 & cos(\omega * T) & 0 & -sin(\omega * T) \\ 0 & \frac{(1-cos(\omega * T))}{\omega} & 1 & \frac{(sin(\omega * T))}{\omega} \\ 0 & sin(\omega * T) & 0 & cos(\omega * T) \end{bmatrix} \boldsymbol{x}(k-1) + \begin{bmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \end{bmatrix} \nu(k-1) \qquad (5.1.1.4)$$

5.1.2. Matrices de Probabilidades de Transición

Las probabilidades de transición de un modo a otro están relacionadas con el tiempo esperado de permanencia en cada modo. Estas probabilidades se eligen de acuerdo a las creencias en tiempo de diseño acerca de cuan frecuentemente se cambia de un modo a los demás. Los coeficientes de la diagonal se pueden determinar mediante la igualdad $p_{ii} = 1 - \frac{1}{E[\tau_i]}$ siendo τ_i el tiempo medio de permanencia en el modo de movimiento i.

Las matrices de transición son estocásticas, los elementos de las filas deben sumar uno, ya que la probabilidad total de pasar de un modo de movimiento a los demás ha de ser igual a la unidad. En este trabajo asumimos que los elementos de las matrices son constantes y los valores elegidos para cada IMM diseñado son los presentados en la tabla 5.1.

5.1.3. Modelos de Mediciones

En nuestro sistema disponemos solo de medidas de posición porque es lo que proporcionan los radares que simulamos; la ecuación de observación en nuestros filtros queda:

$$\boldsymbol{z}(k) = H\boldsymbol{x}(k) + \boldsymbol{w} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \boldsymbol{x}(k) + \boldsymbol{w} \tag{5.1.3.1}$$

5.2. Descripción de los Filtros Propuestos

Los algoritmos de los filtros IMM diseñados (y los filtros Kalman que los constituyen), implementados en Matlab se pueden consultar en el apéndice A.4 y A.6.

Se han evaluado y comparado cuatro filtros IMM con entre dos y cuatro subfiltros Kalman ajustados a varios tipos de movimiento:

1. IMM con M1, M3
2. IMM con M1, M2 y M4
3. IMM con M1, M3 y M4
4. IMM con M1, M4, M5 y M6 (figura 5.1)

Utilizando, en cada combinación los valores de los parámetros que se indican en la tabla 5.1.

IMM	**Modos**								**Matriz de Transición**
	M1	M2	M3	M4	M5		M6		
	q_1	q_2	q_3	q_4	q_5	w	q_6	w	
M1M3	0.01		0.01						$\begin{pmatrix} 0.95 & 0.05 \\ 0.1 & 0.90 \end{pmatrix}$
M1M2M4	0.01	2		2					$\begin{pmatrix} 0.96 & 0.02 & 0.02 \\ 0.15 & 0.8 & 0.05 \\ 0.15 & 0.05 & 0.8 \end{pmatrix}$
M1M3M4	0.001		0.01	2					$\begin{pmatrix} 0.97 & .015 & .015 \\ 0.15 & 0.70 & 0.15 \\ 0.15 & 0.15 & 0.70 \end{pmatrix}$
M1M4M5M6	0.01			2	0.01	0.2	0.01	-0.2	$\begin{pmatrix} 0.98 & 0.01 & 0.005 & 0.005 \\ 0.08 & 0.9 & 0.01 & 0.01 \\ 0.24 & 0.075 & 0.66 & 0.025 \\ 0.24 & 0.07 & 0.03 & 0.66 \end{pmatrix}$

Cuadro 5.1: Filtros IMM Simulados

El vector de estado que se actualiza en cada ciclo consta de seis variables, que corresponden a posición, velocidad y aceleración en cada una de las dos dimensiones 5.2.0.2.

$$\boldsymbol{x}(t) = \begin{bmatrix} x & \dot{x} & \ddot{x} & y & \dot{y} & \ddot{y} \end{bmatrix}^T \tag{5.2.0.2}$$

Cada subfiltro Kalman actualiza las variables que le afectan (según estén ajustados a modelos de segundo o tercer orden).

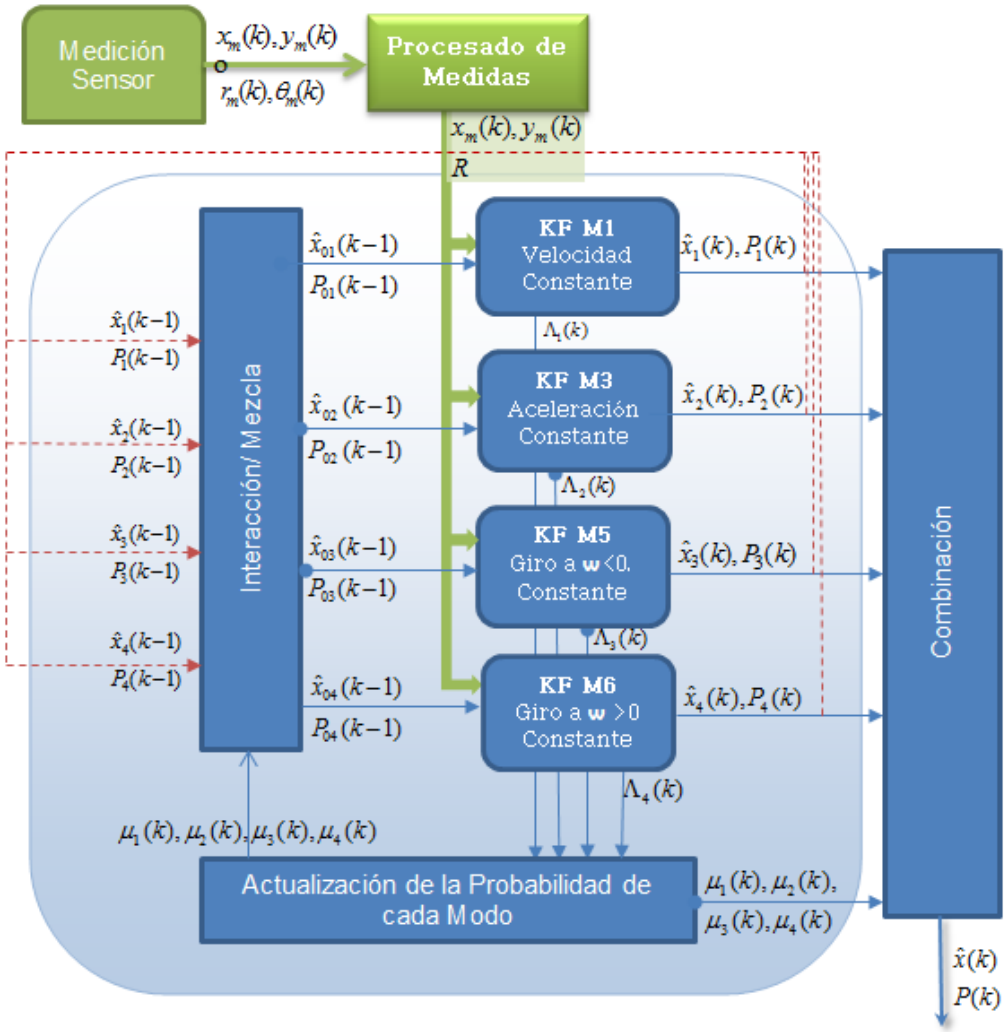


Figura 5.1: Estructura del filtro IMM con 4 Modos e Movimiento

5.2.1. Entradas

A continuación se enumeran los valores iniciales de los parámetros del filtro:

1. ***Medición de la cinemática del blanco en el instante k***: Posición del blanco en coordenadas cartesianas. Si el radar proporciona la posición en cartesianas, se trasladan al sistema de coordenadas del aeropuerto. Para el caso de los radares que aportan rango y azimut, se aplica el correspondiente proceso de transformación de polares a cartesianas (ver 3.3).

2. ***Matriz de transiciones entre modos:*** Utilizamos matrices estáticas (que no cambian en el tiempo). Los valores las elegimos según las frecuencias de permanencia en cada modo en los escenarios a simular.

3. ***Parámetros de los filtros Kalman internos***

   Los estimadores de estado $\hat{x}_{0i}$ y sus covarianzas P_{0i} de cada modelo Antes de realizar el premezclado, tienen que ser inicializados. En nuestro sistema calculamos los valores iniciales para la primera iteración del ciclo IMM a partir de las dos primera medidas asignadas a una pista.

Varianza en el ruido de proceso, Q Representa la variación entre el valor real de las características y la estimación obtenida a través de los sensores. Es una matriz de igual dimensión que P en la que los elementos de la diagonal indica los valores de Q para cada uno de los parámetros que se usan en el modelo. Se calculan como se explicó en la sección 3.2. Utilizamos los valores de ruido especificados en la tabla 5.1

Varianza del ruido de medida R En el caso de los radares que proporcionan la localización de los objetivos en coordenadas cartesianas la matriz nos va a quedar:

$$R = \begin{bmatrix} \sigma^x & 0 \\ 0 & \sigma^y \end{bmatrix} \tag{5.2.1.1}$$

y para los radares de tipo SMR y/o SSR, que detectan el rango y azimut, se construye según lo expuesto en la sección 3.3.1.6.

5.2.2. Salidas

Cada iteración del ciclo de un filtro IMM produce el estimado de estado del blanco (posición, velocidad y aceleración) en el instante k, $\hat{\boldsymbol{x}}(k)$ y la matriz de covarianza asociada al estimado de estado en el instante k, $P(k)$.

Capítulo 6

Experimentos y Resultados

En este capítulo presentamos resultados que cubren varios escenarios de simulación, con tres configuraciones de sensores para analizar el funcionamiento de los filtros en situaciones similares a los tres aeropuertos de los que disponemos de datos reales,

Configuración 1 (Barcelona) *Radar de Movimiento en Superficie* (SMR) (+datos de sistema de vigilancia de Aire (TDVM))

Configuración 2 (Palma) *Radar de Movimiento en Superficie* (SMR) y *Sistema de Multilateración* (MLAT) (+ datos sistema de vigilancia de Aire)

Configuración 3 (Madrid) *Radar de Movimiento en Superficie* (SMR) y *Sistema de Multilateración* (MLAT) y un segundo SMR (+ datos sistema de vigilancia de Aire)

y diversas trayectorias de vehículos, que dividimos en:

- Trayectorias Simples: secuencias de movimientos muy concretas y en su conjunto, simples, que se pueden contemplar de forma generalizada en un *aeropuerto real*; tales como despegue, aterrizaje, velocidad constante y giros.
- Trayectorias Complejas: combinaciones de trayectorias simples, donde las transiciones entre ellas constituyen un elemento clave.

Con *escenario* estamos haciendo referencia a una configuración de sensores en conjunción con una trayectoria específica (por ejemplo: configuración 1, despegue; configuración 2, velocidad constante; etc).

Los resultados son generados y presentados en conjuntos, con un conjunto por cada trayectoria, que engloba los resultados obtenidos en los distintos escenarios.

En total hay 3 configuraciones de sensores, 5 trayectorias simples y 9 trayectorias complejas, haciendo un total de 52 escenarios. Cada conjunto incluye los datos obtenidos al aplicar los 4 filtros IMM bajo análisis, permitiendo realizar una comparación de la efectividad de los mismos.

Para cada *escenario* se realizan 100 ejecuciones, simulando las trayectorias reales perfectas y generando observaciones de radar aleatorias con errores adaptados según los parámetros de los radares activos en la configuración, y se aplica cada filtro IMM obteniéndose trayectorias filtradas (suavizadas) según cada algoritmo IMM aplicado.

En cada conjunto de resultados se presentan los siguientes gráficos, que son auto-explicativos:

- Posición, velocidad y aceleración "real" de la correspondiente trayectoria.

- Posición media observada por cada uno de los radares (de la configuración 3)
- Datos promedio de posición y velocidad estimados por cada filtro IMM.
- Probabilidades de modo calculadas por los filtros IMM en una ejecución. Las probabilidad de que cada modelo dinámico sea el correcto en cada instante, según determine cada algoritmo IMM respectivo.
- Comparación del error cuadrático medio de la posición detectada por el radar y de la posición estimada por el cada filtro IMM; comparación del error cuadrático medio de la velocidad estimada por cada filtro IMM.
- Diagramas de cajas con la distribución de los Raíz del Error Cuadrático Medio de los datos de posición y velocidad obtenidos por cada filtro, para cada configuración, junto con los resultantes del proceso sin-filtrado.

Además en el apéndice C se pueden encontrar tablas resumen con los valores numéricos de los errores y las mejoras que suponen los datos filtrados respecto a las medidas sin someter a proceso de filtrado.

La combinación de todas las figuras nos ayuda a visualizar de forma directa y comprender de manera intuitiva:

- la trayectoria del vehículo bajo consideración,
- la calidad tanto de los datos fuente como de los filtrados.
- el comportamiento dinámico interno de los filtros IMM durante la trayectoria, en particular durante las transiciones en un modo de movimiento a otro.
- la efectividad relativa de los diferentes filtros, comparados con el método actual sin-filtrado, y entre sí.

6.1. Resultados

6.1.1. Velocidad Constante

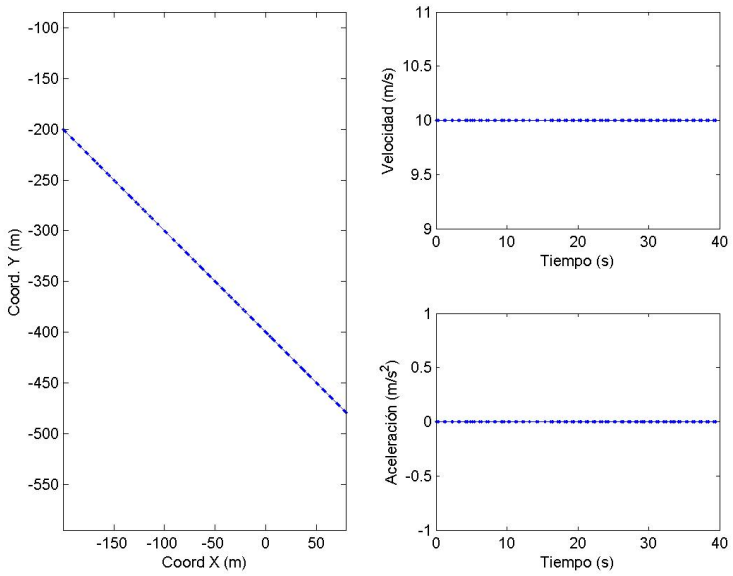
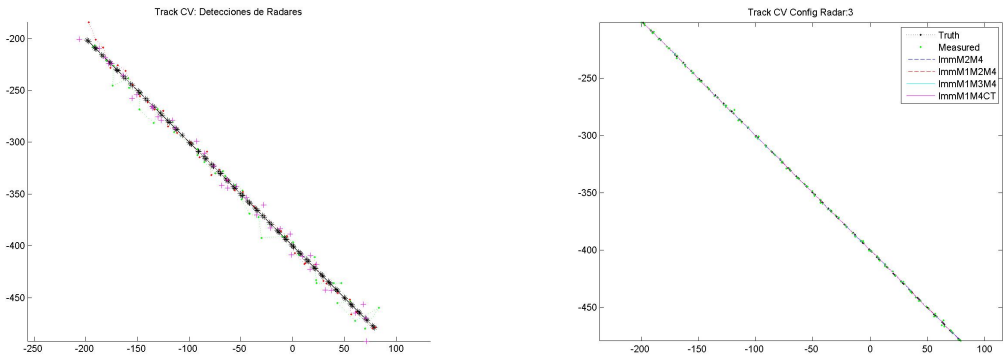


Figura 6.1: Descripción Escenario Velocidad Constante



(a) Mediciones Radar

(b) Trayectorias Filtradas

Figura 6.2: Tray Vel Constante: Detecciones Radar y Trayectorias Suavizadas por los Filtros

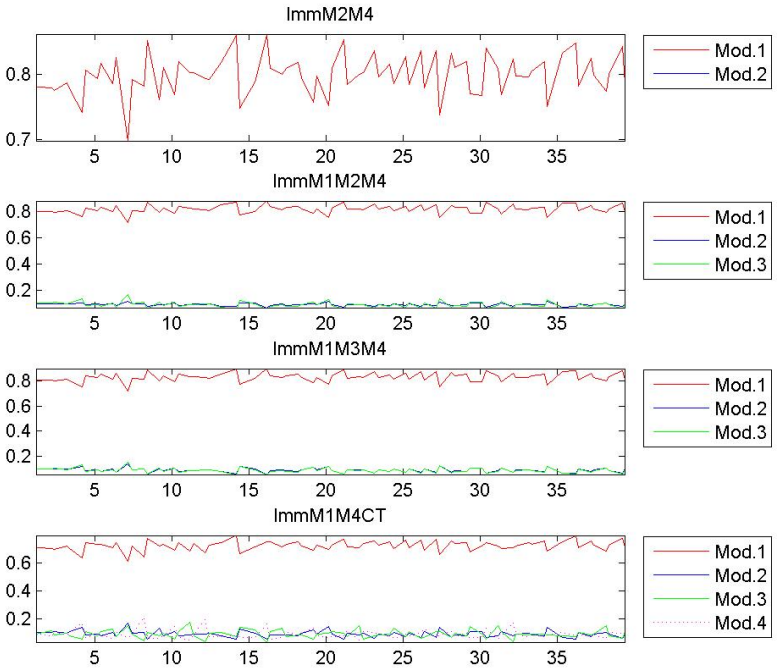


Figura 6.3: Tray Vel ConstanteGo: Probabilidades de Modo según cada Filtro IMM

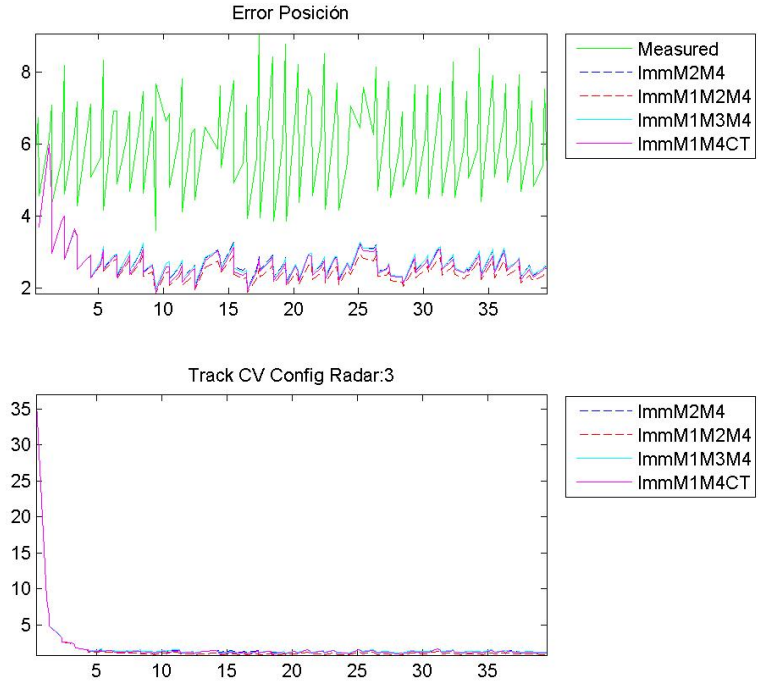
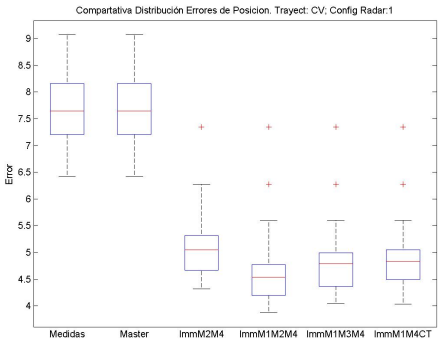
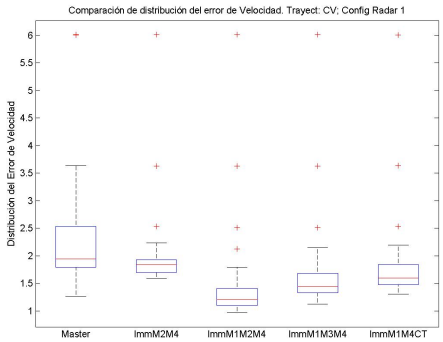


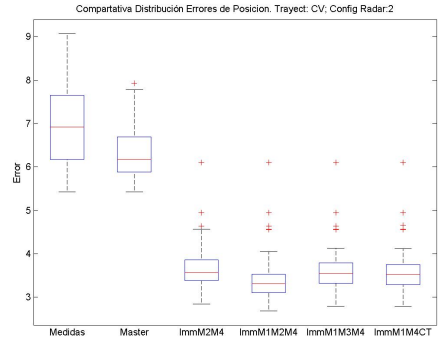
Figura 6.4: Tray Vel Constante: RSME de Posición y Velocidad



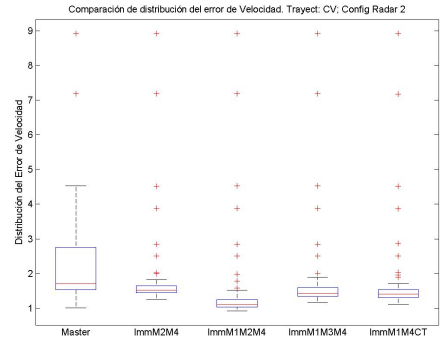
(a) Dist Errores Pos (Config.1)



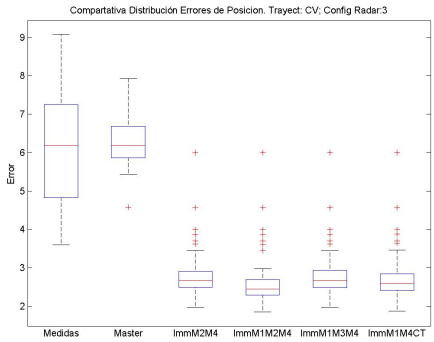
(b) Dist Errores Vel (Config.1)



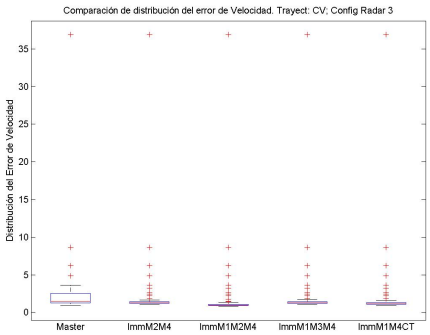
(c) Dist Errores Pos (Config.2)



(d) Dist Errores Vel (Config.2)



(e) Dist Errores Pos (Config.3)



(f) Dist. Errores Vel (Config.3)

Figura 6.5: Tray Vel Constante: Distribución del RSME en las Posiciones y Velocidad

6.1.2. Giros

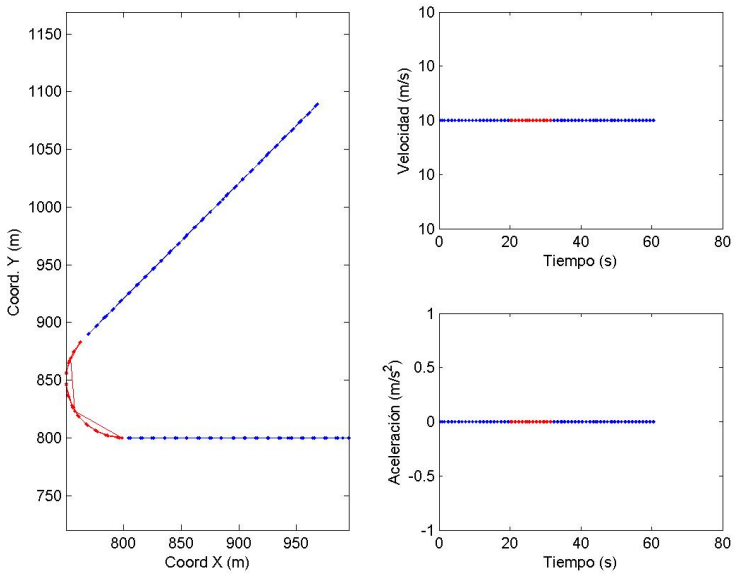


Figura 6.6: Descripción Escenario con Giro de 135 grados

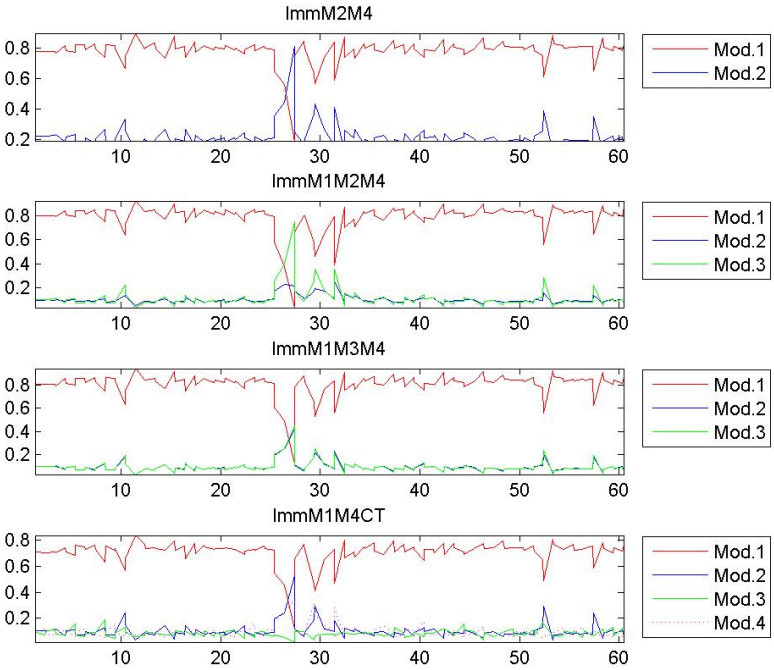


Figura 6.7: Tray Giro de 135: Probabilidades de Modo según cada Filtro IMM

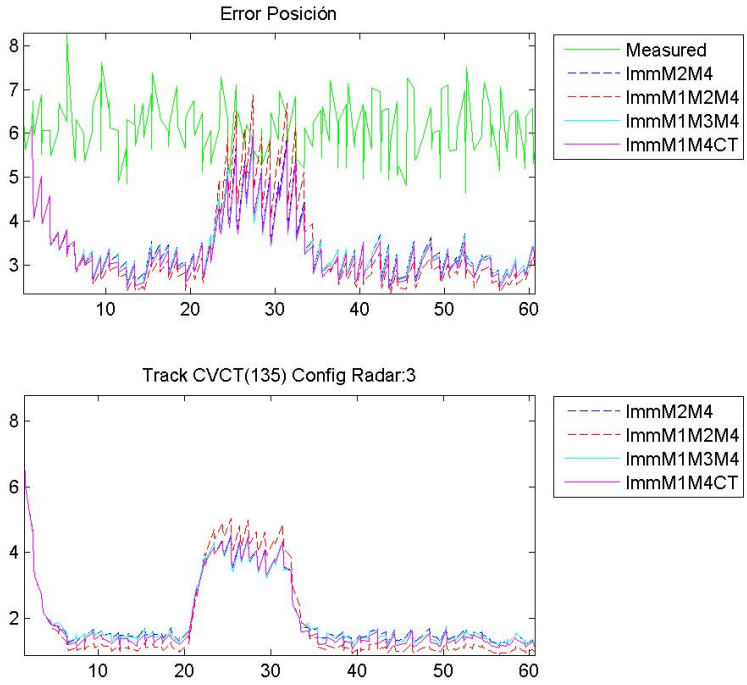
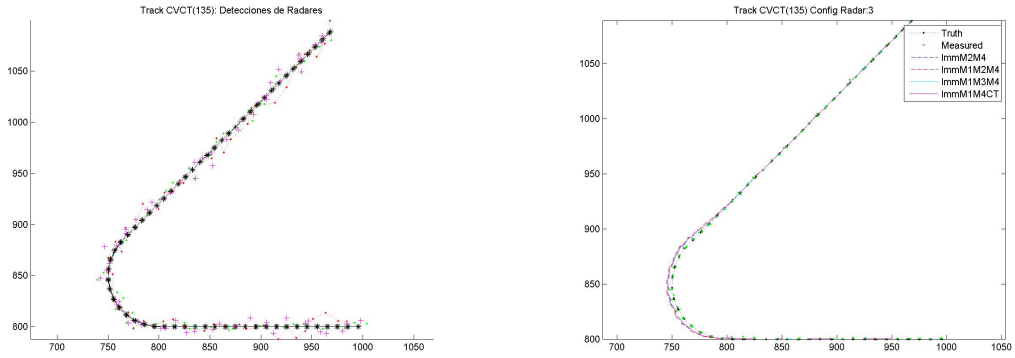


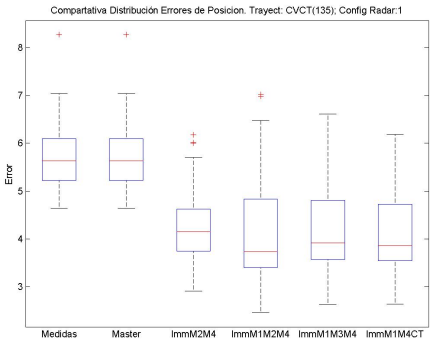
Figura 6.8: Tray Giro de 135: RSME de Posición y Velocidad



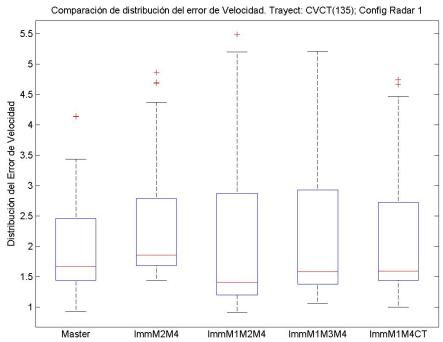
(a) Mediciones Radar

(b) Trayectorias Filtradas

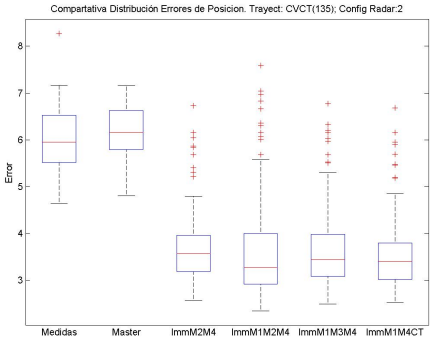
Figura 6.9: Tray Giro de 135: Detecciones Radar y Trayectorias Suavizadas por los Filtros



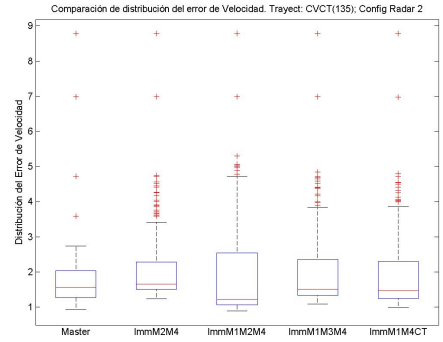
(a) Dist Errores Pos (Config.1)



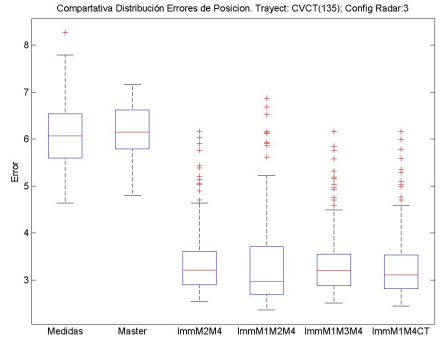
(b) Dist Errores Vel (Config.1)



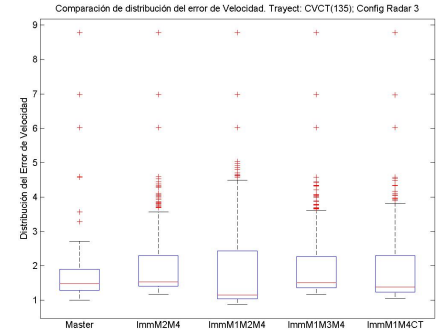
(c) Dist Errores Pos (Config.2)



(d) Dist Errores Vel (Config.2)



(e) Dist Errores Pos (Config.3)



(f) Dist. Errores Vel (Config.3)

Figura 6.10: Tray Giro de 135: Distribución del RSME en las Posiciones y Velocidad

6.1.3. Stop&Go

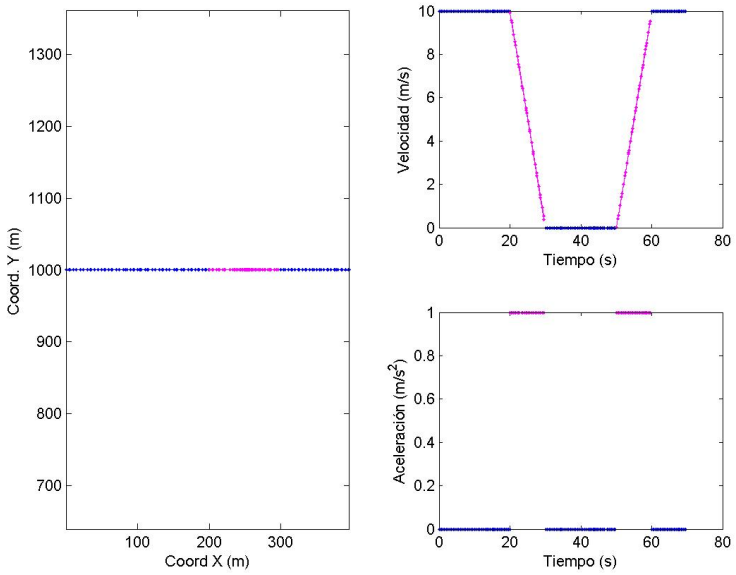


Figura 6.11: Descripción Escenario Stop-and-Go

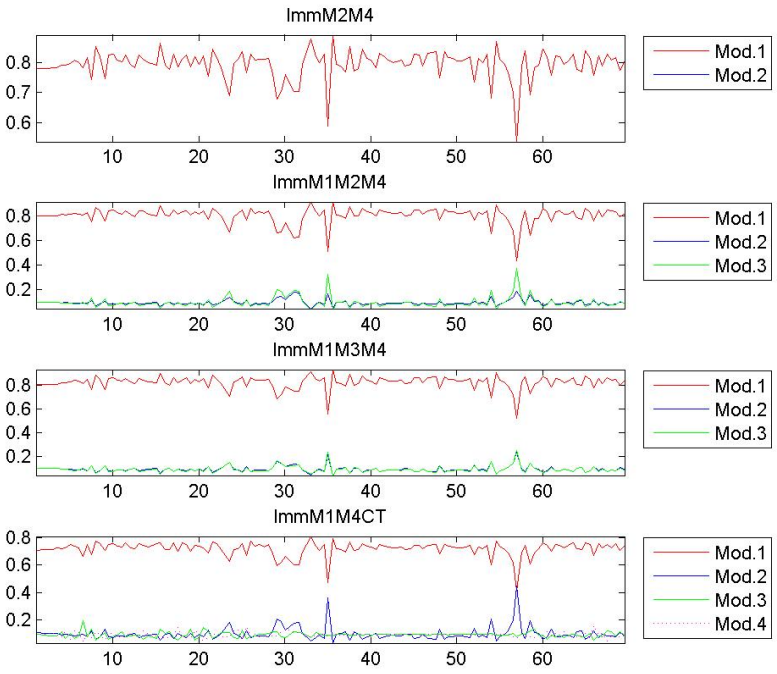


Figura 6.12: Tray.Stop-and-Go: Probabilidades de Modo según cada Filtro IMM

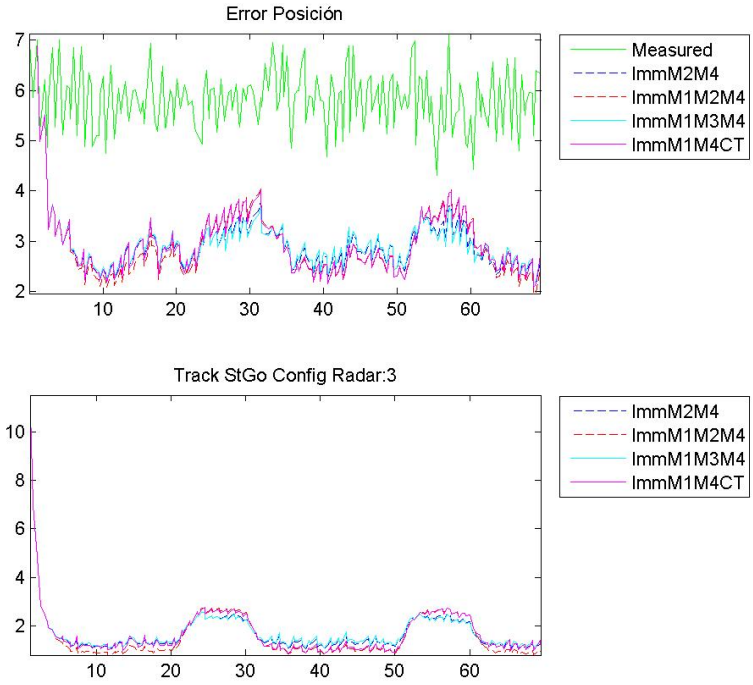
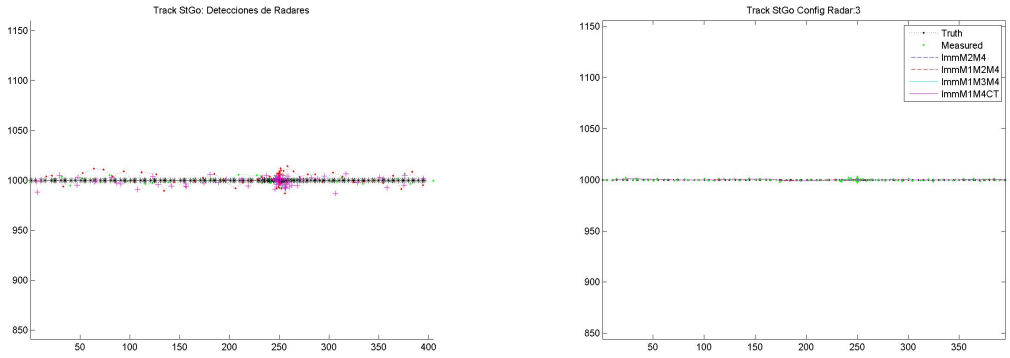


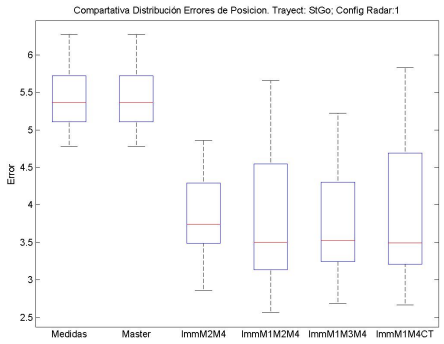
Figura 6.13: Tray. Stop-and-Go: RSME de Posición y Velocidad



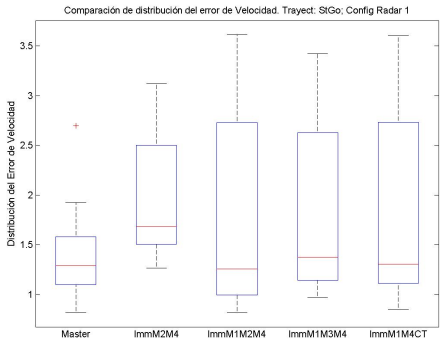
(a) Mediciones Radar

(b) Trayectorias Filtradas

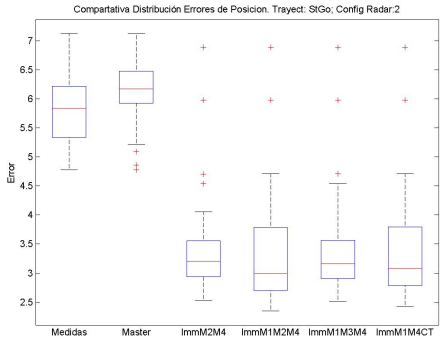
Figura 6.14: Tray Stop-and-Go: Detecciones Radar y Trayectorias Suavizadas por los Filtros



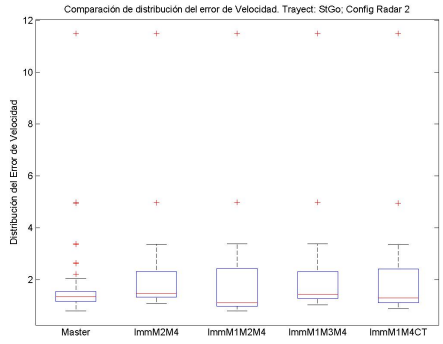
(a) Dist Errores Pos (Config.1)



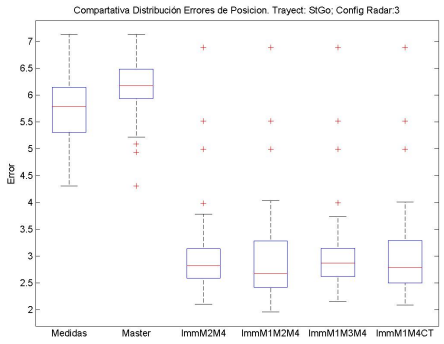
(b) Dist Errores Vel (Config.1)



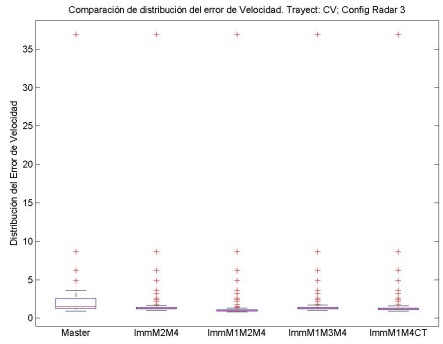
(c) Dist Errores Pos (Config.2)



(d) Dist Errores Vel (Config.2)



(e) Dist Errores Pos (Config.3)



(f) Dist. Errores Vel (Config.3)

Figura 6.15: Tray Stop-and-Go: Distribución del RSME en las Posiciones y Velocidad

6.1.4. Aterrizaje

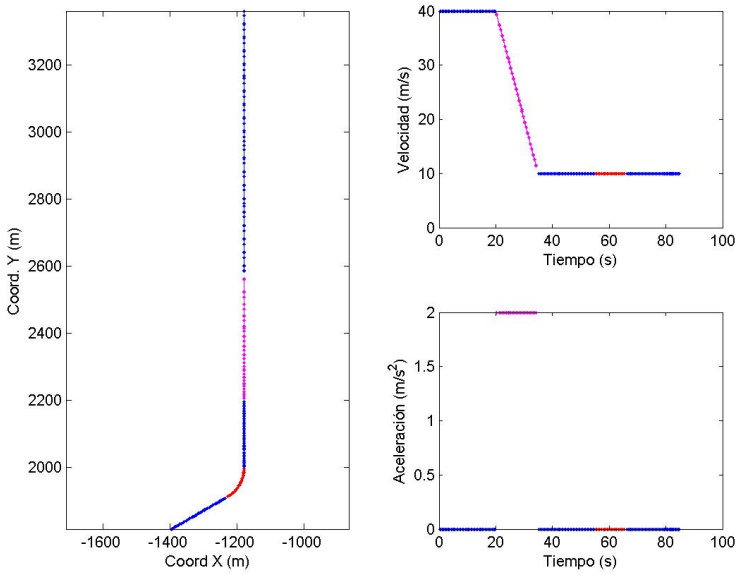


Figura 6.16: Descripción Maniobra Aterrizaje

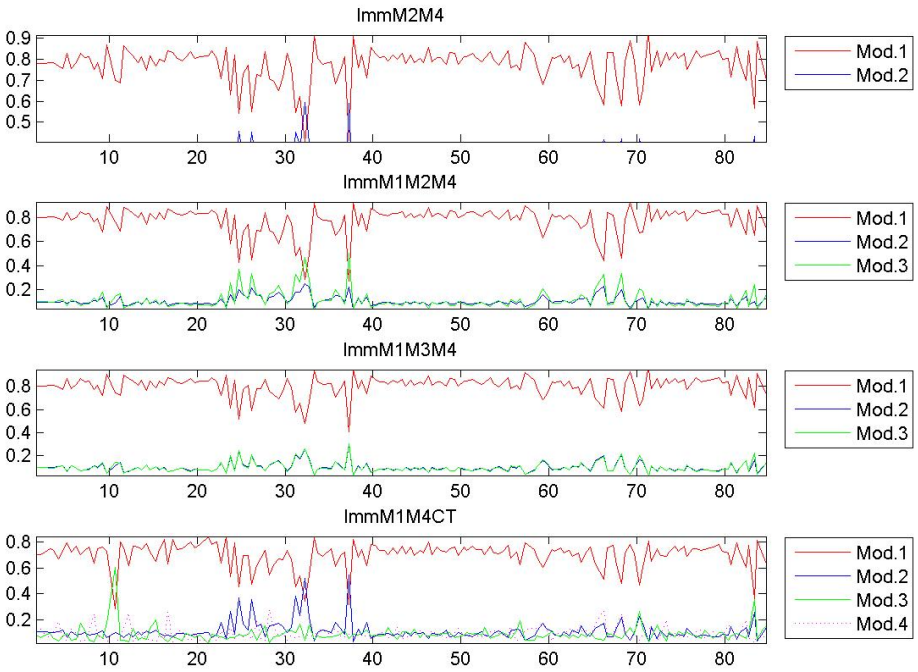


Figura 6.17: Maniobra Aterrizaje: Probabilidades de Modo según cada Filtro IMM

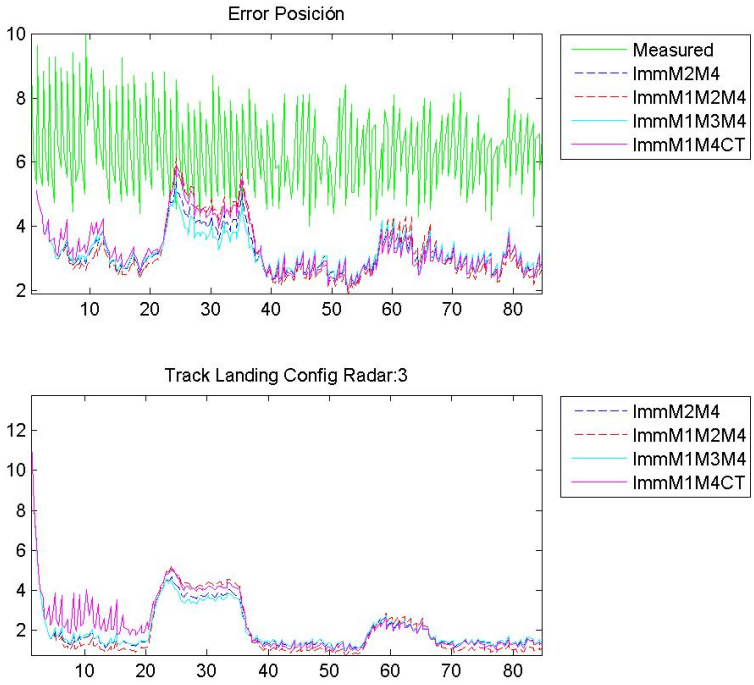
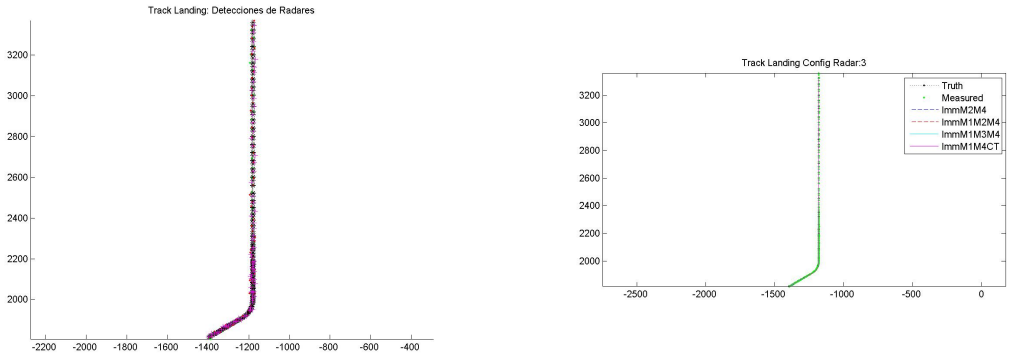


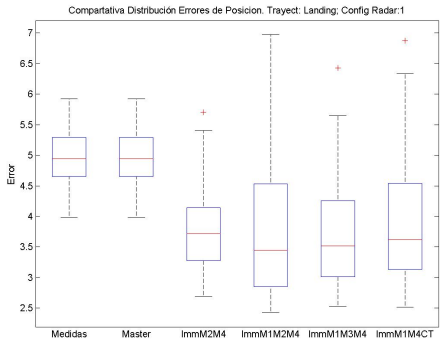
Figura 6.18: Maniobra Aterrizaje: RSME de Posición y Velocidad



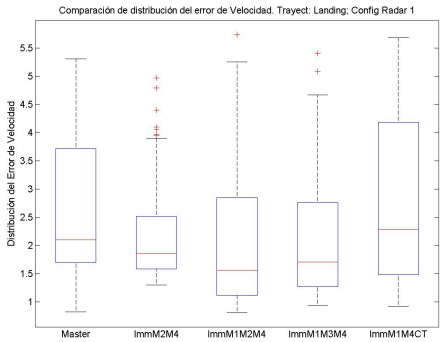
(a) Mediciones Radar

(b) Trayectorias Filtradas

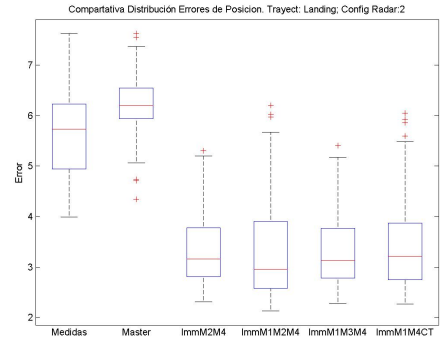
Figura 6.19: Maniobra Aterrizaje: Detecciones Radar y Trayectorias Suavizadas por los Filtros



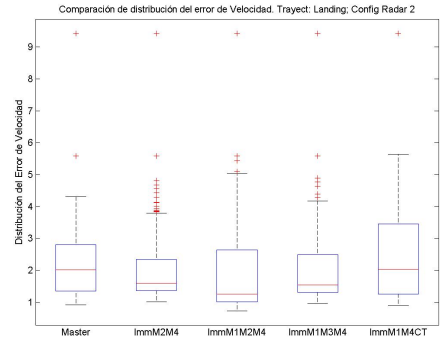
(a) Dist Errores Pos (Config.1)



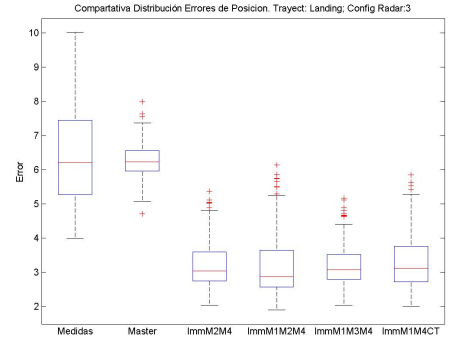
(b) Dist Errores Vel (Config.1)



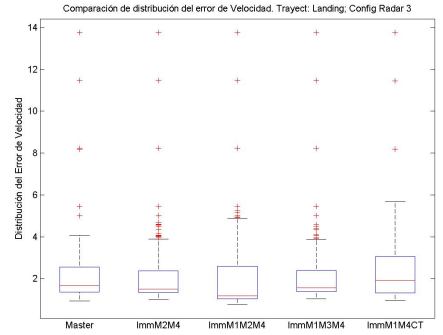
(c) Dist Errores Pos (Config.2)



(d) Dist Errores Vel (Config.2)



(e) Dist Errores Pos (Config.3)



(f) Dist. Errores Vel (Config.3)

Figura 6.20: Maniobra Aterrizaje: Distribución del RSME en las Posiciones y Velocidad

6.1.5. Despegue

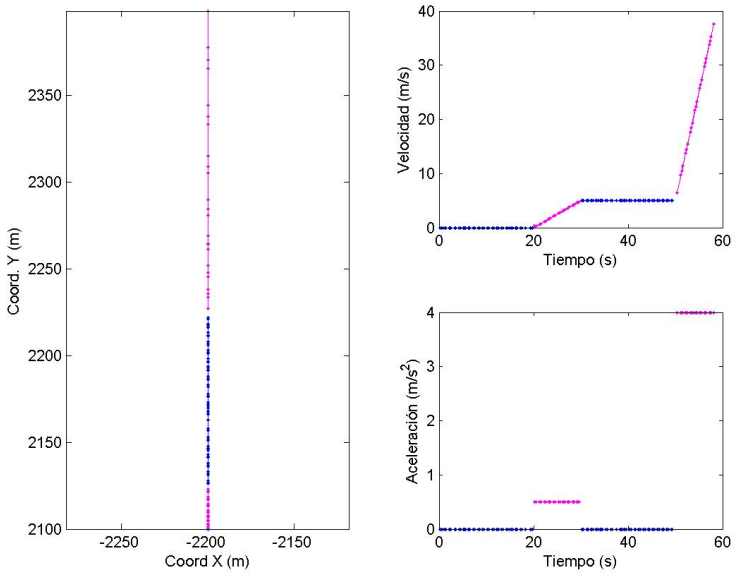
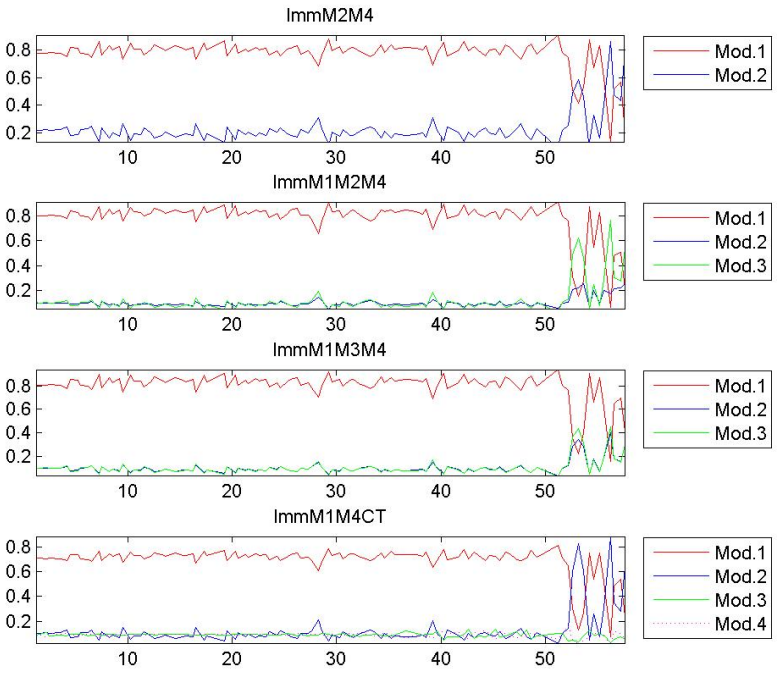


Figura 6.21: Descripción Maniobra Despegue



(a) Probabilidades de Modo

Figura 6.22: Maniobra Despegue: Probabilidades de Modo según cada Filtro IMM

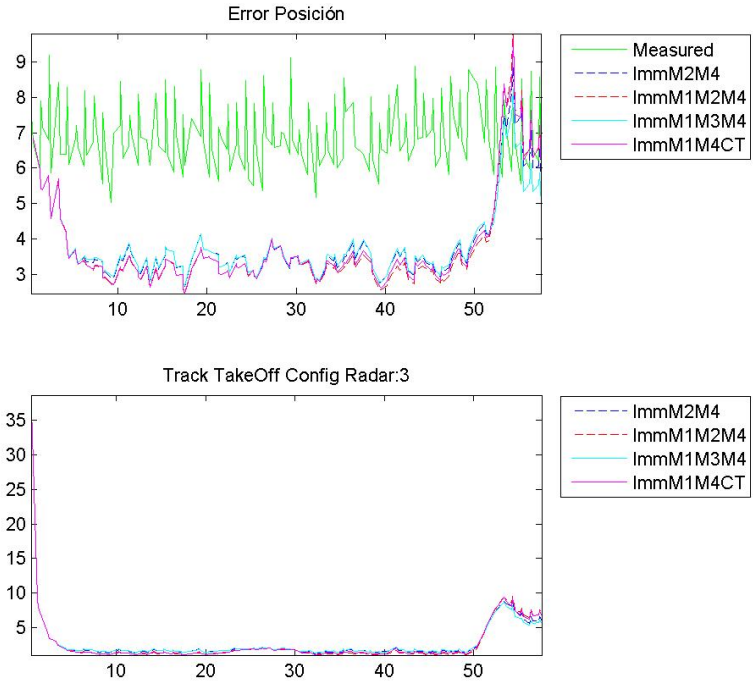
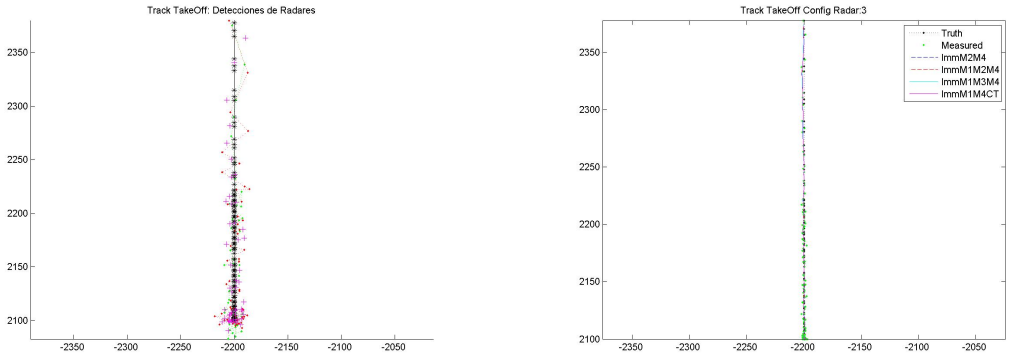


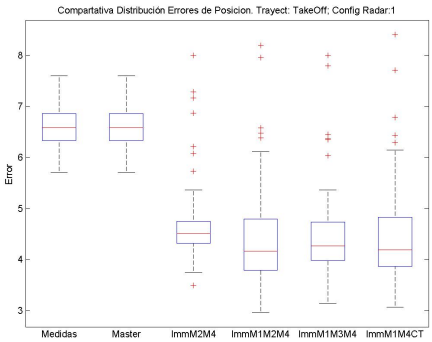
Figura 6.23: Maniobra Despegue: RSME de Posición y Velocidad



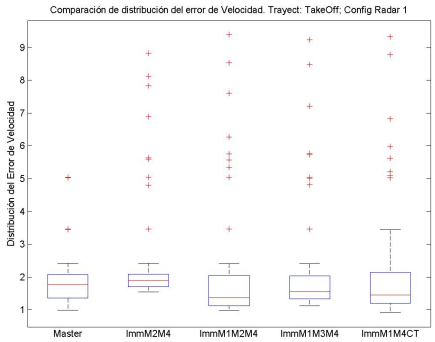
(a) Mediciones Radar

(b) Trayectorias Filtradas

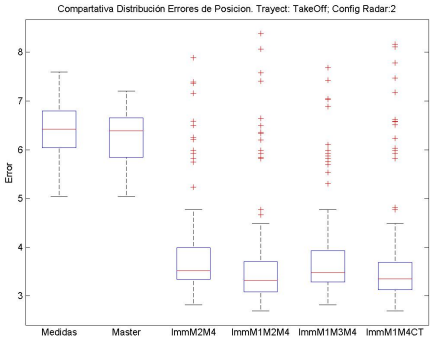
Figura 6.24: Maniobra Despegue: Detecciones Radar y Trayectorias Suavizadas por los Filtros



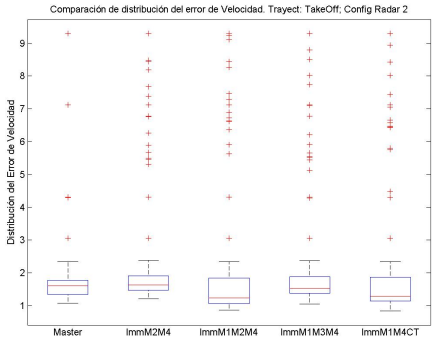
(a) Dist Errores Pos (Config.1)



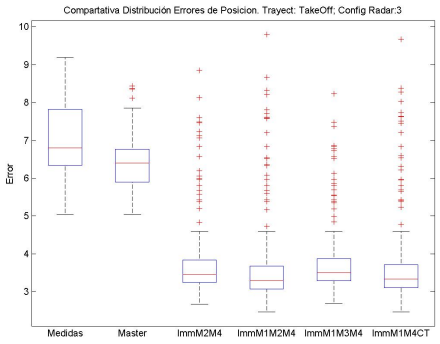
(b) Dist Errores Vel (Config.1)



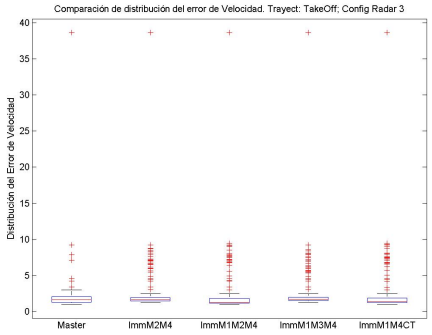
(c) Dist Errores Pos (Config.2)



(d) Dist Errores Vel (Config.2)



(e) Dist Errores Pos (Config.3)



(f) Dist. Errores Vel (Config.3)

Figura 6.25: Maniobra Despegue: Distribución del RSME en las Posiciones y Velocidad

6.1.6. Desplazamiento 1

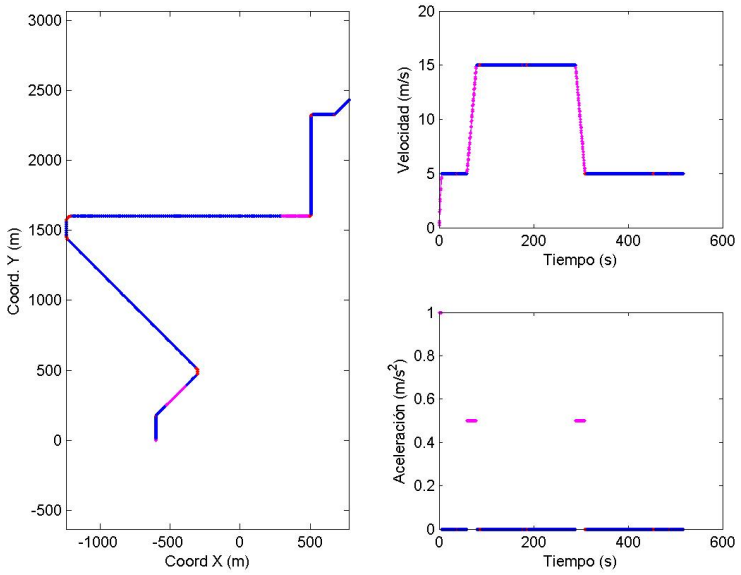
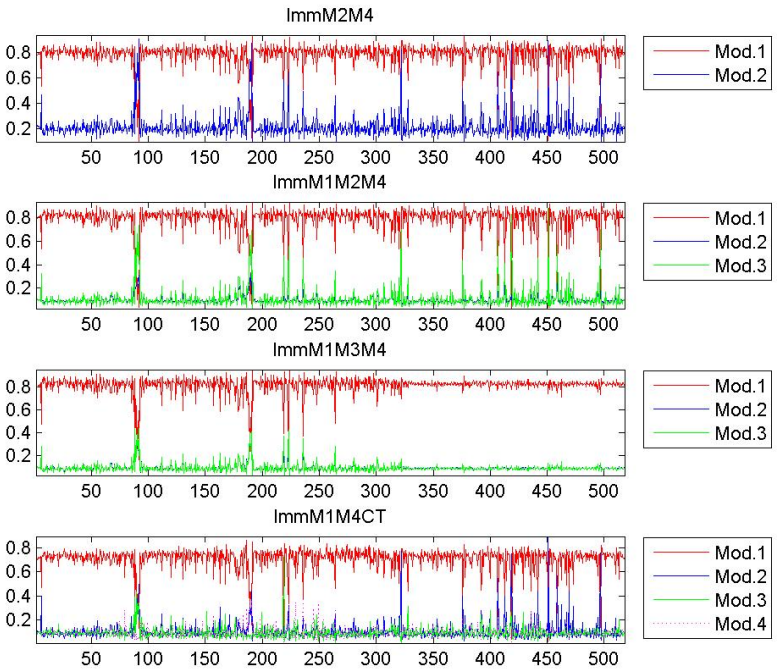


Figura 6.26: Descripción Escenario Desplazamiento 1



(a) Probabilidades de Modo

Figura 6.27: Tray DPZ1: Probabilidades de Modo según cada Filtro IMM

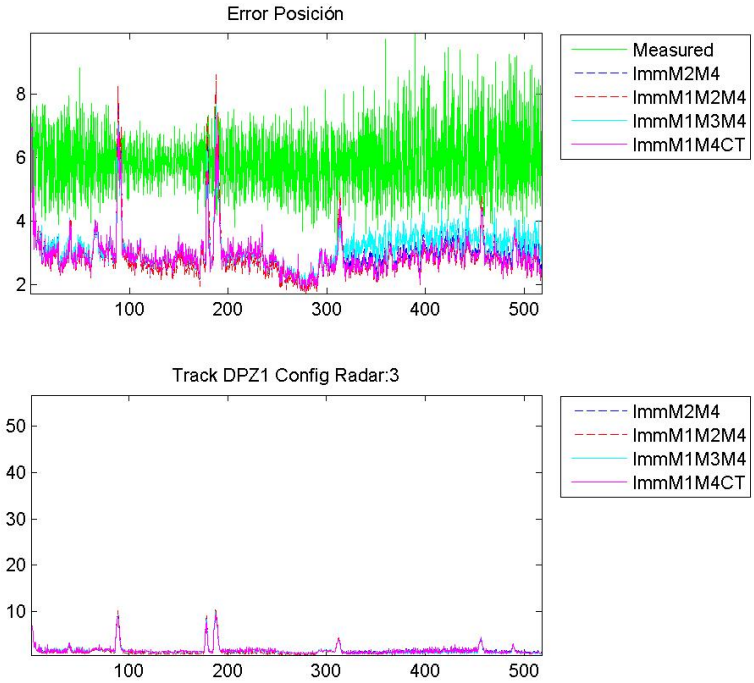
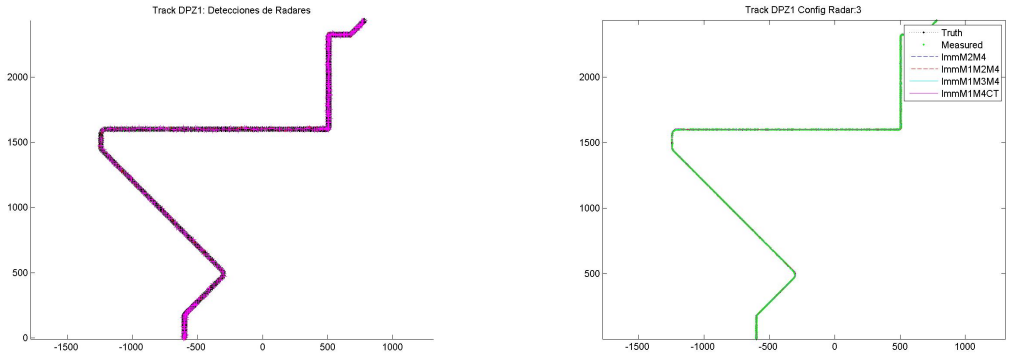


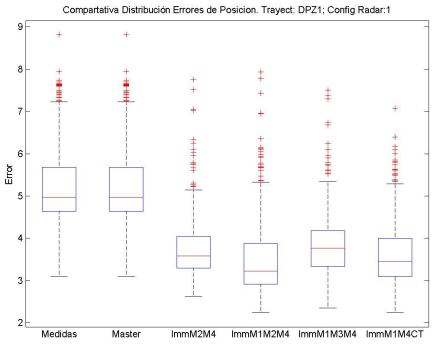
Figura 6.28: Tray DPZ1: RSME de Posición y Velocidad



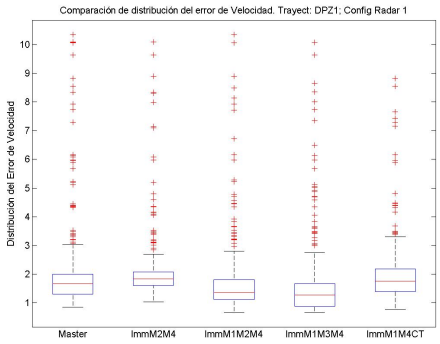
(a) Mediciones Radar

(b) Trayectorias Filtradas

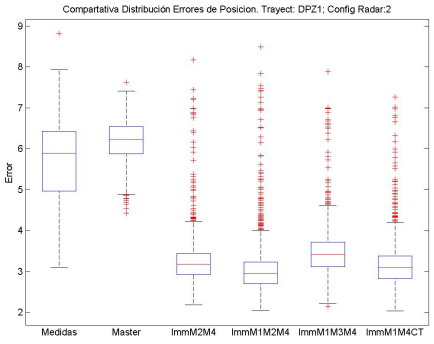
Figura 6.29: Tray DPZ1: Detecciones Radar y Trayectorias Suavizadas por los Filtros



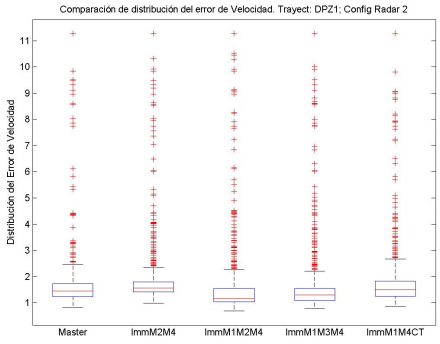
(a) Dist Errores Pos (Config.1)



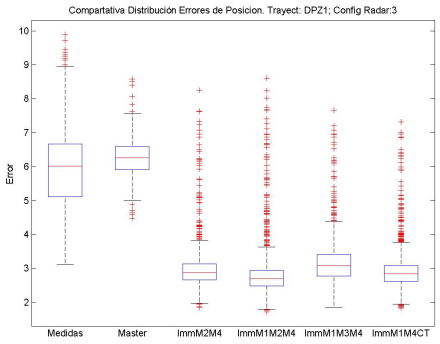
(b) Dist Errores Vel (Config.1)



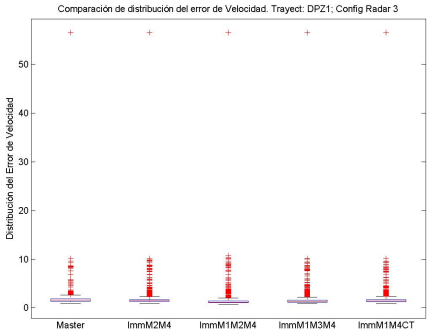
(c) Dist Errores Pos (Config.2)



(d) Dist Errores Vel (Config.2)



(e) Dist Errores Pos (Config.3)



(f) Dist. Errores Vel (Config.3)

Figura 6.30: Tray DPZ1: Distribución del RSME en las Posiciones y Velocidad

6.1.7. Desplazamiento 2

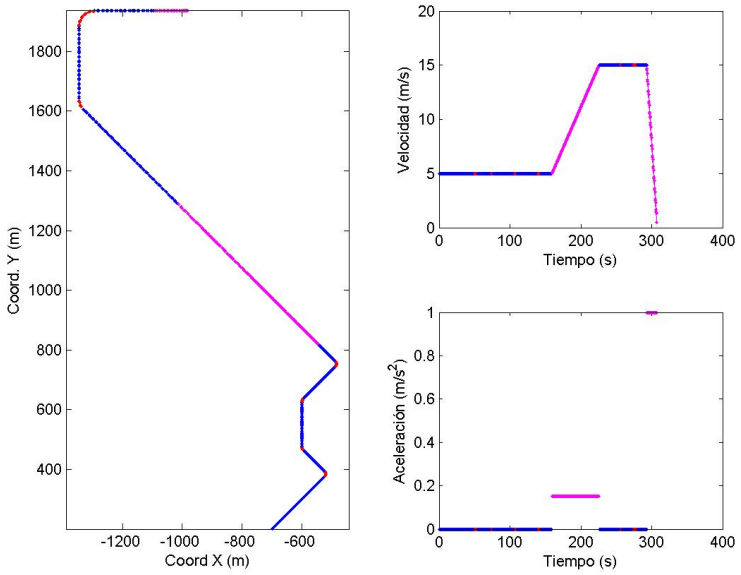
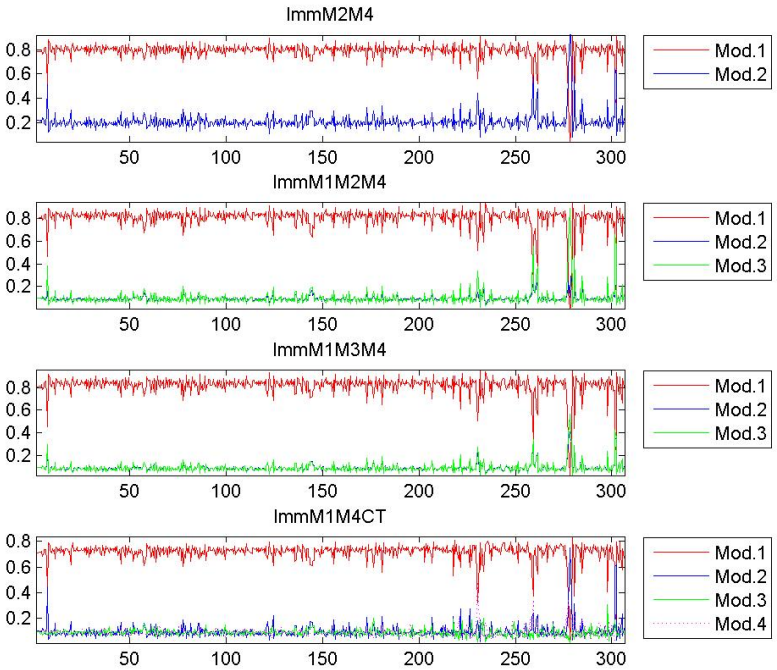


Figura 6.31: Descripción Escenario Desplazamiento 2



(a) Probabilidades de Modo

Figura 6.32: Tray DPZ2: Probabilidades de Modo según cada Filtro IMM

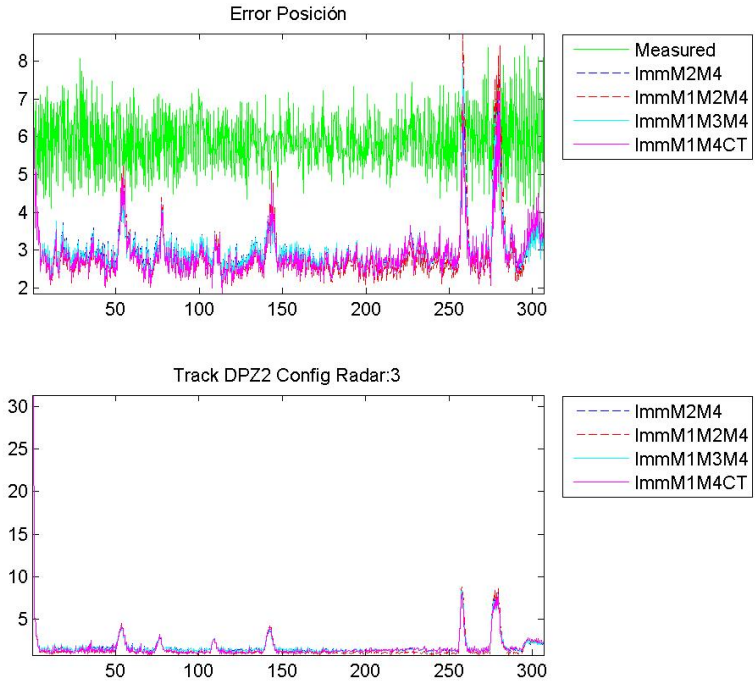
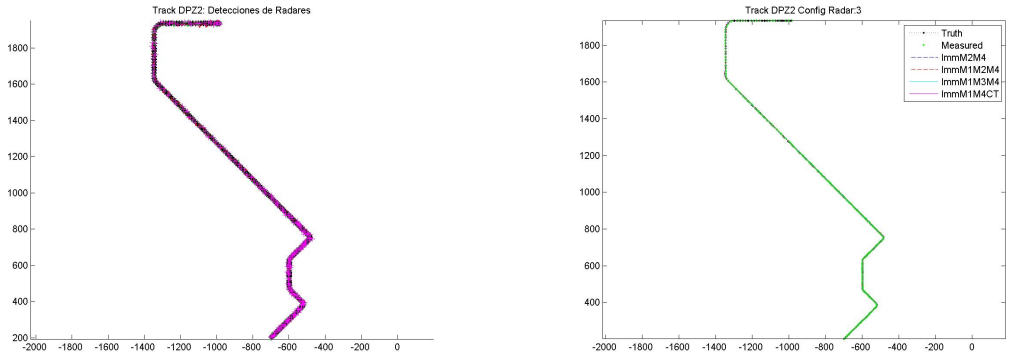


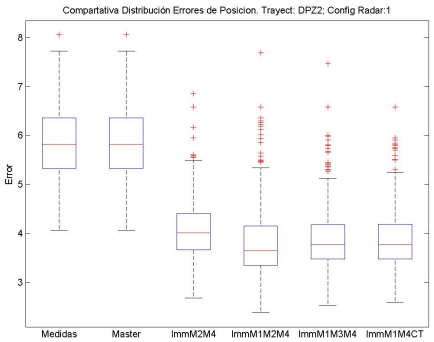
Figura 6.33: Tray DPZ2: RSME de Posición y Velocidad



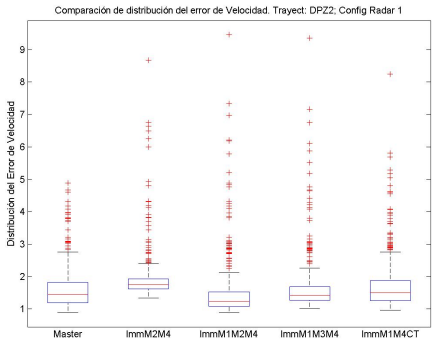
(a) Mediciones Radar

(b) Trayectorias Filtradas

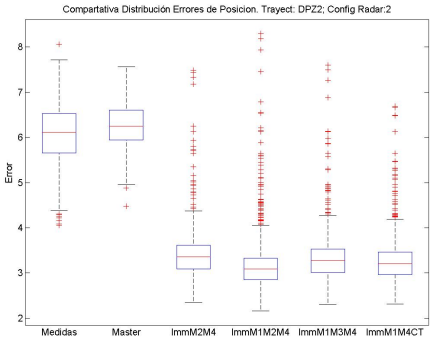
Figura 6.34: Tray DPZ2: Detecciones Radar y Trayectorias Suavizadas por los Filtros



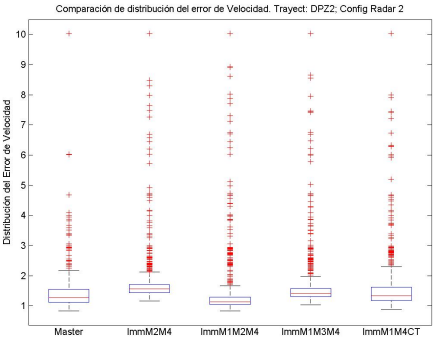
(a) Dist Errores Pos (Config.1)



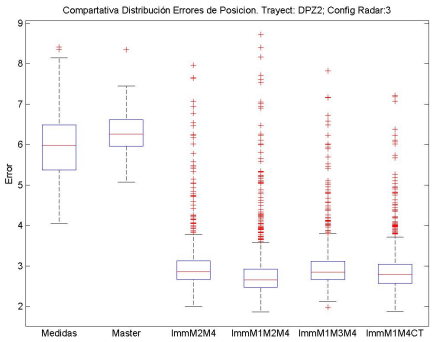
(b) Dist Errores Vel (Config.1)



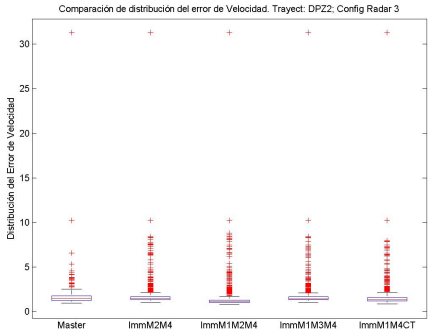
(c) Dist Errores Pos (Config.2)



(d) Dist Errores Vel (Config.2)



(e) Dist Errores Pos (Config.3)



(f) Dist. Errores Vel (Config.3)

Figura 6.35: Tray DPZ2: Distribución del RSME en las Posiciones y Velocidad

6.1.8. Desplazamiento 3

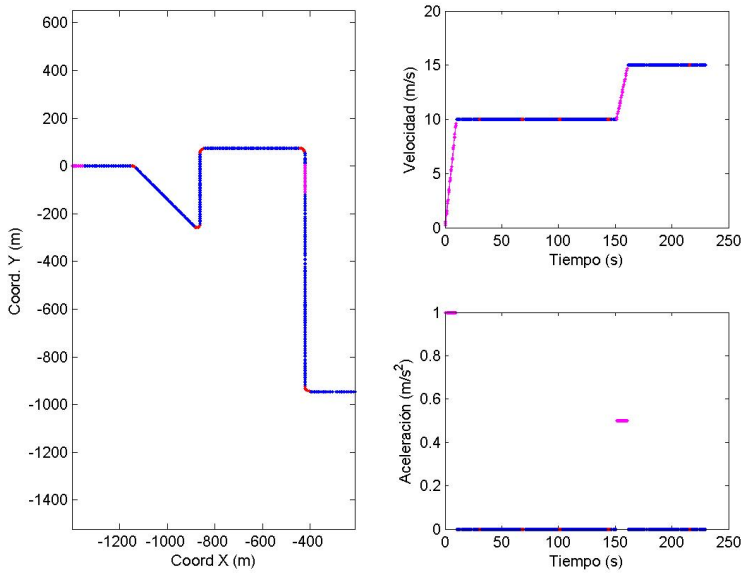
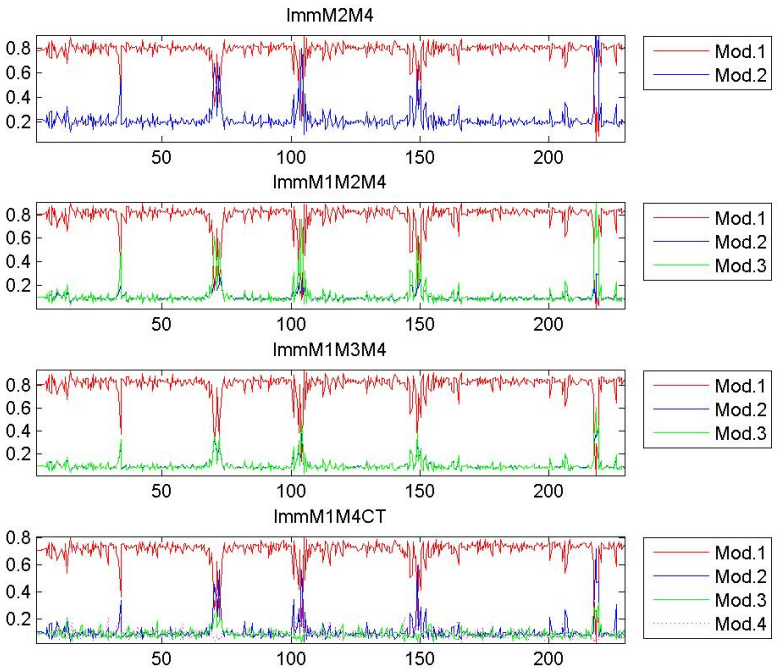


Figura 6.36: Descripción Escenario Desplazamiento 3



(a) Probabilidades de Modo

Figura 6.37: Tray DPZ3: Probabilidades de Modo según cada Filtro IMM

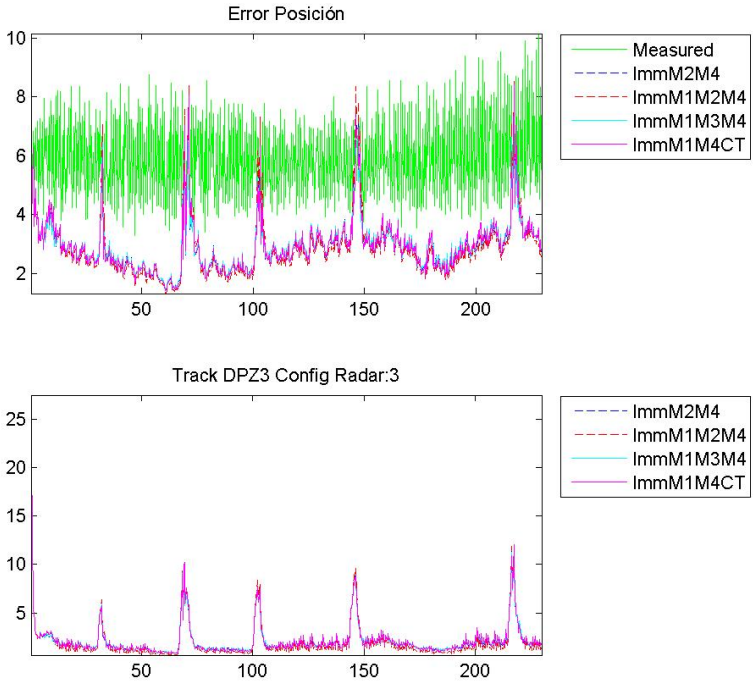
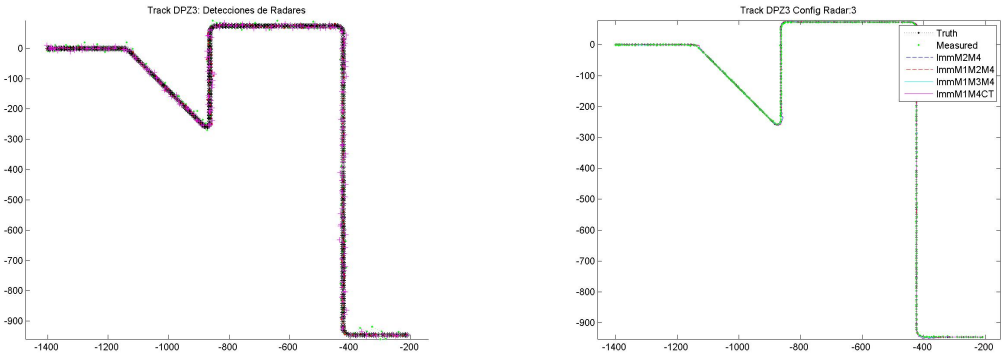


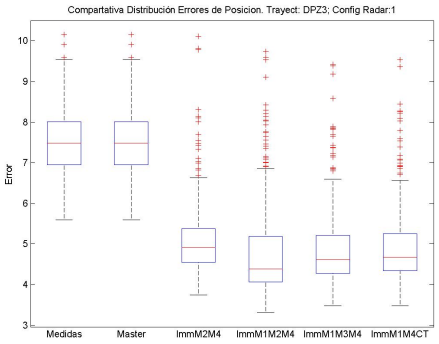
Figura 6.38: Tray DPZ2: RSME de Posición y Velocidad



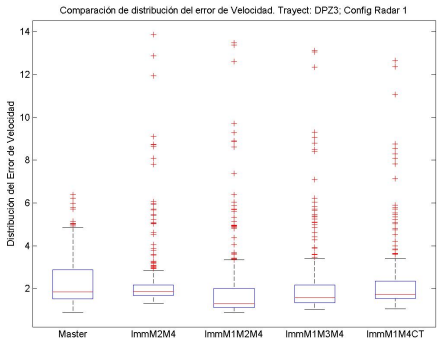
(a) Mediciones Radar

(b) Trayectorias Filtradas

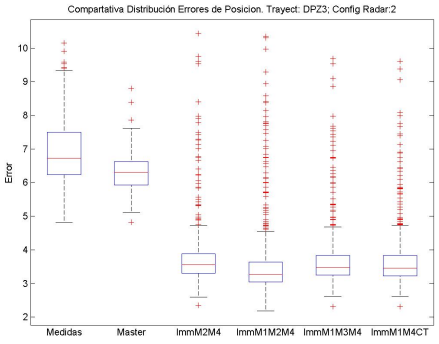
Figura 6.39: Tray DPZ3: Detecciones Radar y Trayectorias Suavizadas por los Filtros



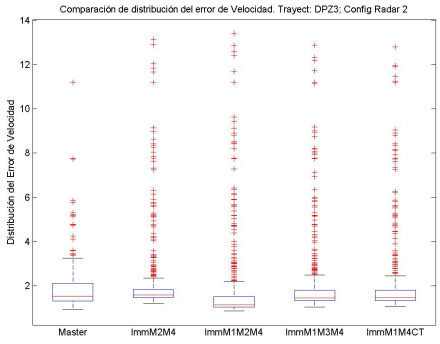
(a) Dist Errores Pos (Config.1)



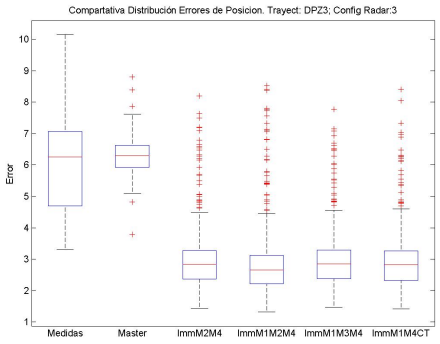
(b) Dist Errores Vel (Config.1)



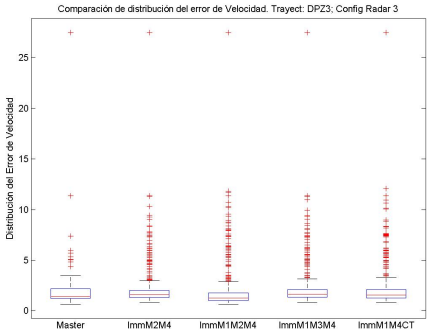
(c) Dist Errores Pos (Config.2)



(d) Dist Errores Vel (Config.2)



(e) Dist Errores Pos (Config.3)



(f) Dist. Errores Vel (Config.3)

Figura 6.40: Tray DPZ3: Distribución del RSME en las Posiciones y Velocidad

6.1.9. Despegue en Pista 1 con Rodaje desde la Terminal

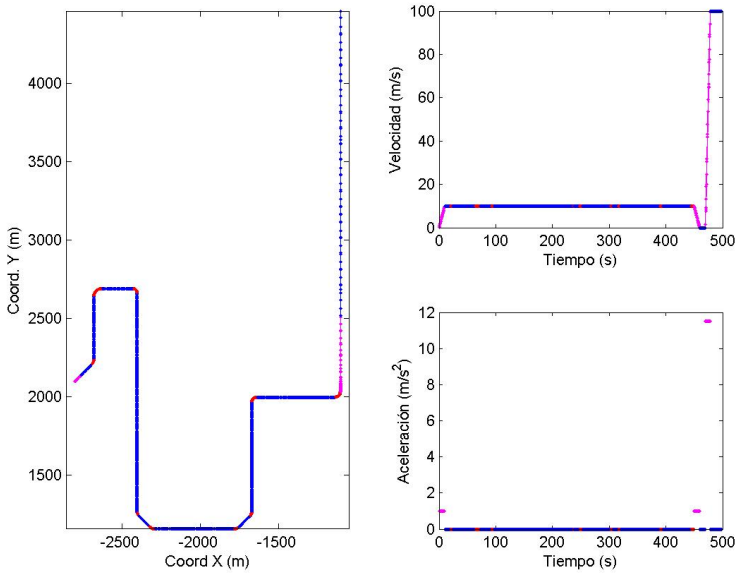


Figura 6.41: Descripción Escenario Despegue en Pista 1

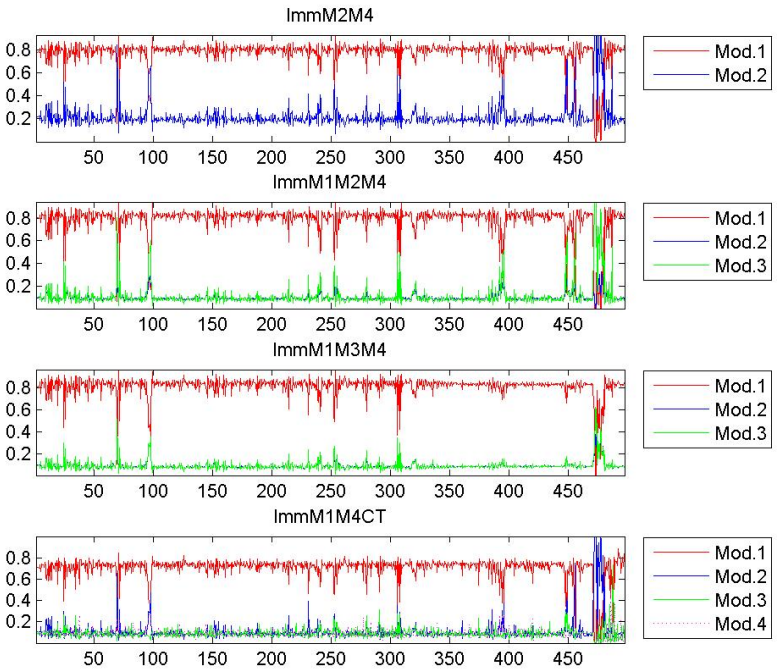


Figura 6.42: Tray Despegue 1: Probabilidades de Modo según cada Filtro IMM

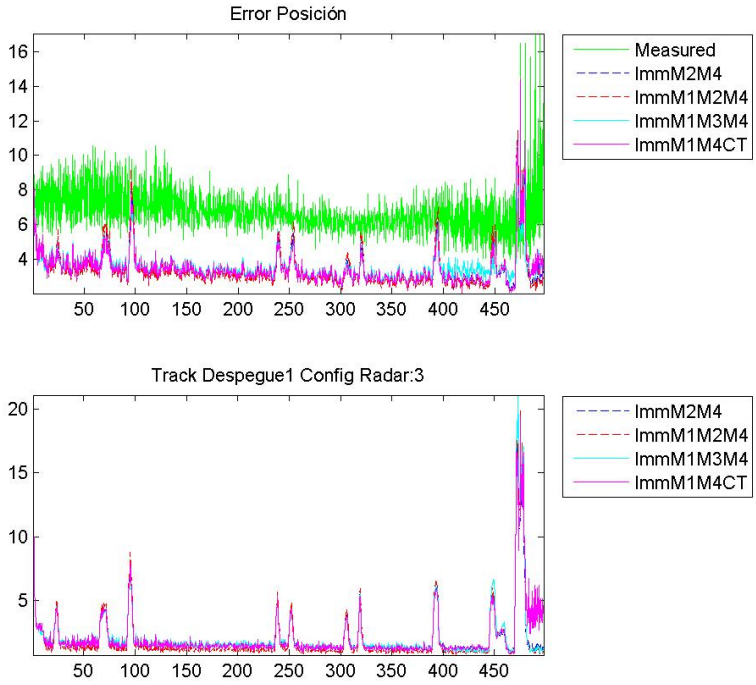
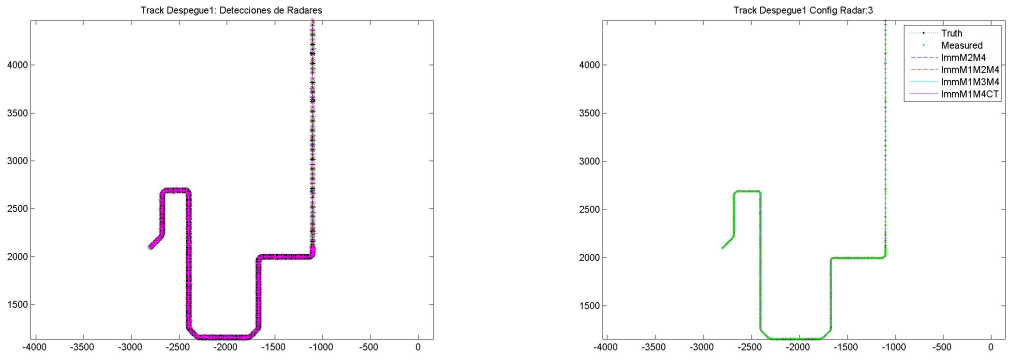


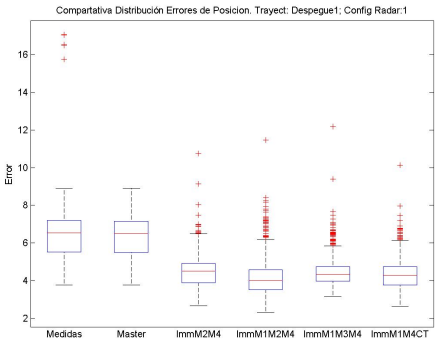
Figura 6.43: Tray Despegue 1: RSME de Posición y Velocidad



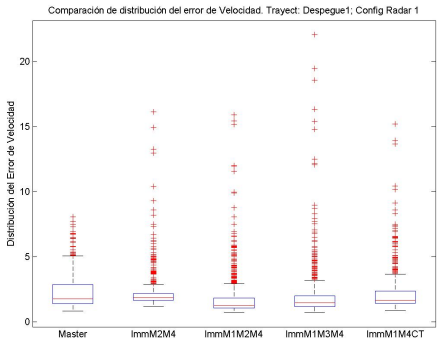
(a) Mediciones Radar

(b) Trayectorias Filtradas

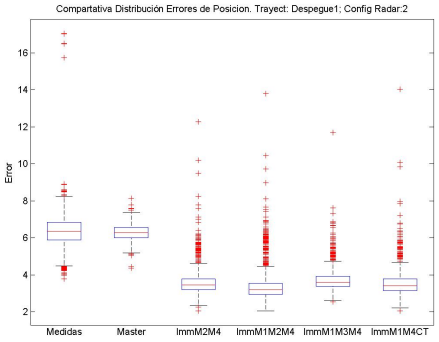
Figura 6.44: Tray Despegue 1: Detecciones Radar y Trayectorias Suavizadas por los Filtros



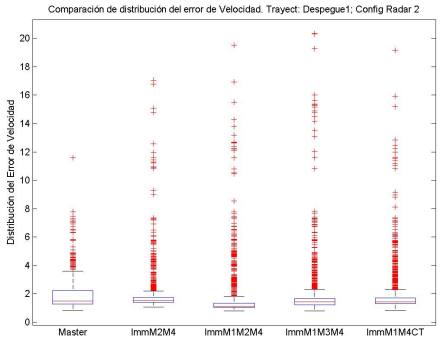
(a) Dist Errores Pos (Config.1)



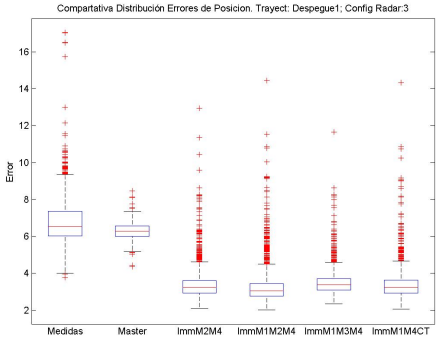
(b) Dist Errores Vel (Config.1)



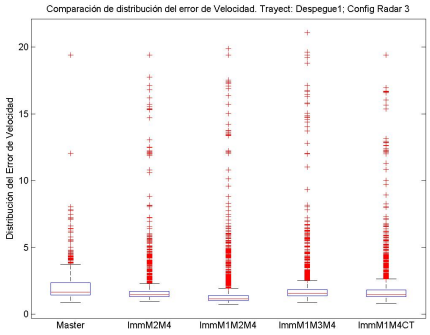
(c) Dist Errores Pos (Config.2)



(d) Dist Errores Vel (Config.2)



(e) Dist Errores Pos (Config.3)



(f) Dist. Errores Vel (Config.3)

Figura 6.45: Tray Despegue 1: Distribución del RSME en las Posiciones y Velocidad

6.1.10. Despegue en Pista 2 con Rodaje desde la Terminal

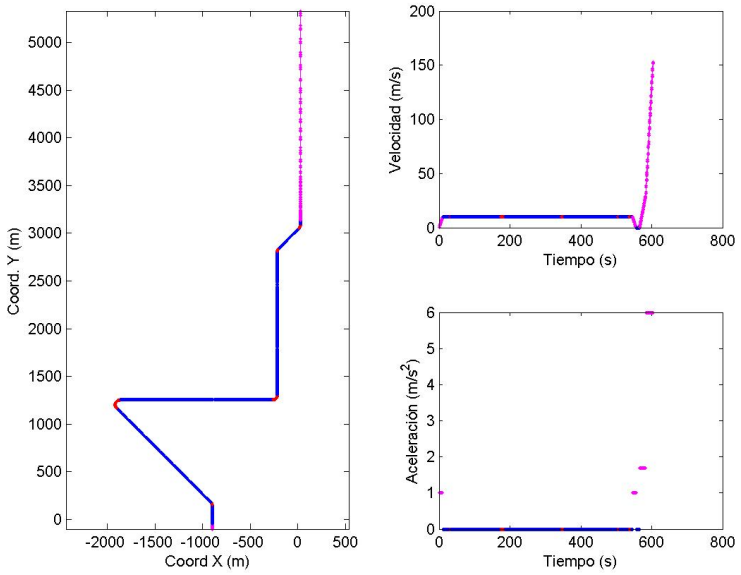


Figura 6.46: Descripción Escenario Despegue en Pista 2

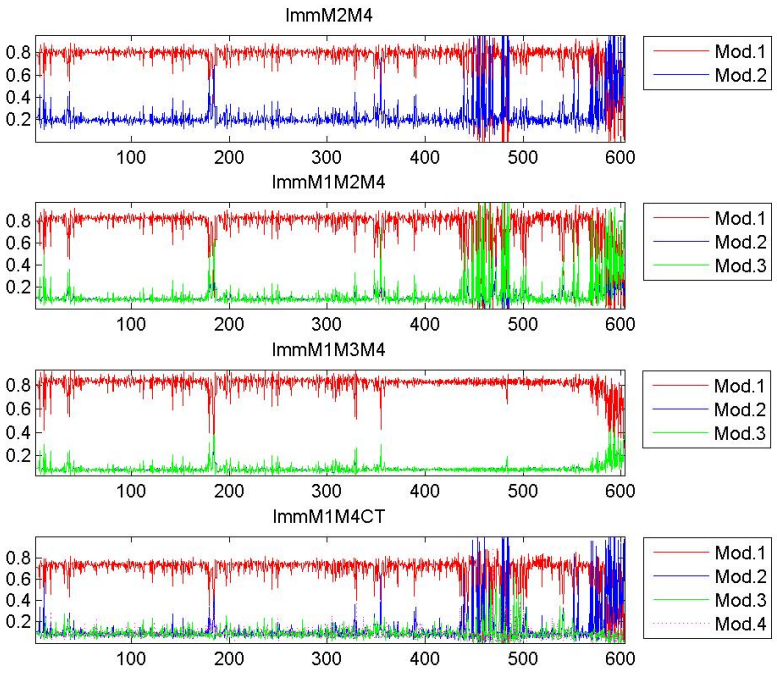


Figura 6.47: Tray Despegue 2: Probabilidades de Modo según cada Filtro IMM

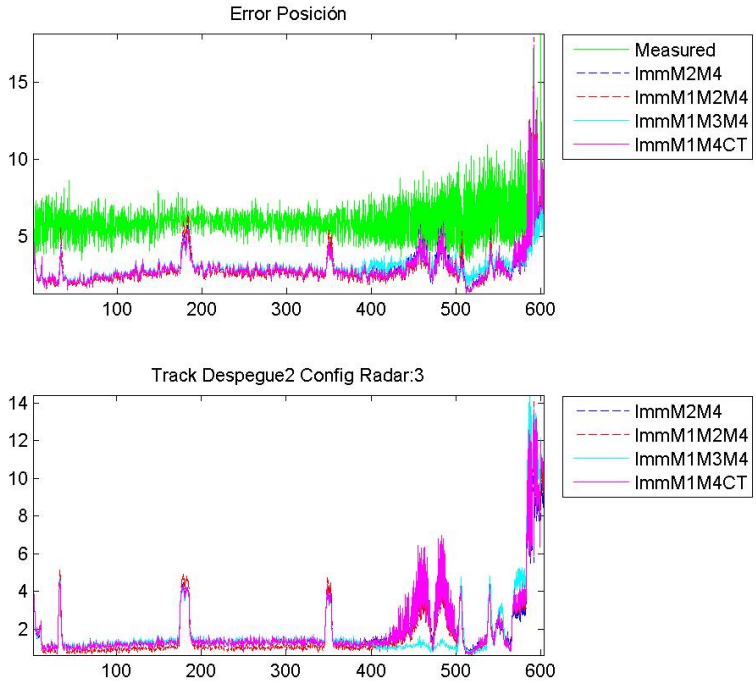
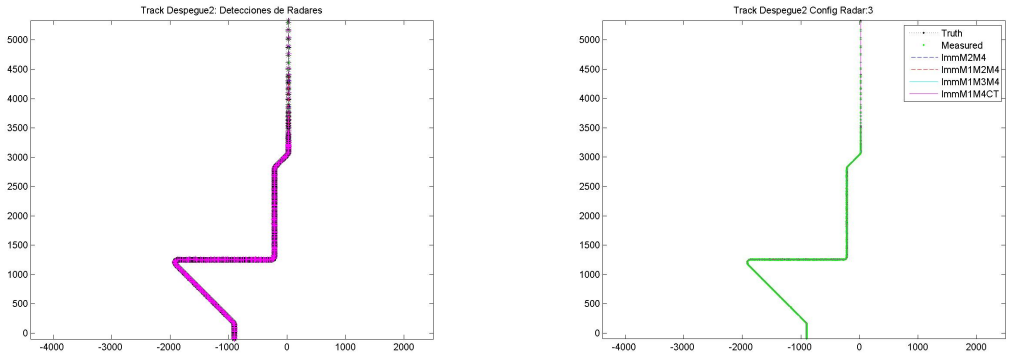


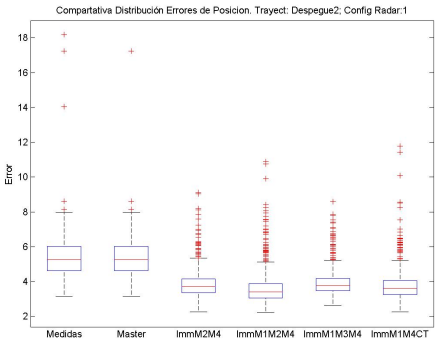
Figura 6.48: Tray Despegue 2: RSME de Posición y Velocidad



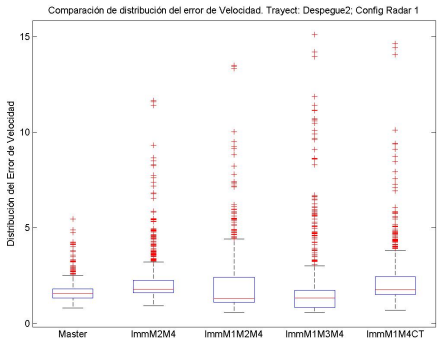
(a) Mediciones Radar

(b) Trayectorias Filtradas

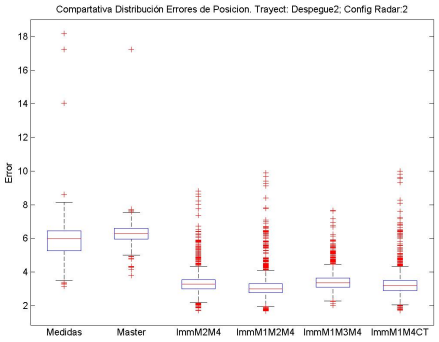
Figura 6.49: Tray Despegue 2: Detecciones Radar y Trayectorias Suavizadas por los Filtros



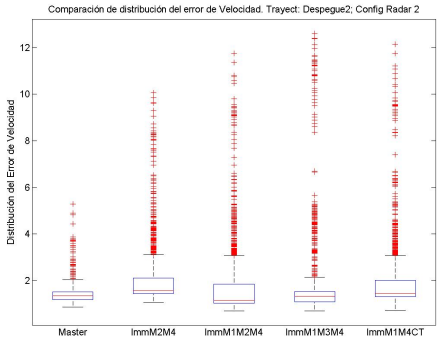
(a) Dist Errores Pos (Config.1)



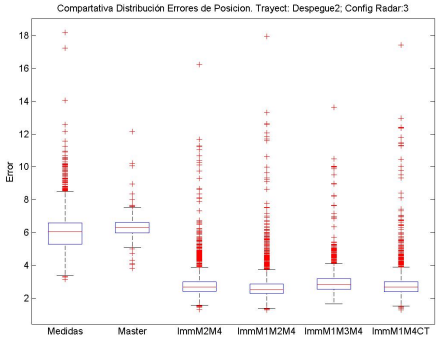
(b) Dist Errores Vel (Config.1)



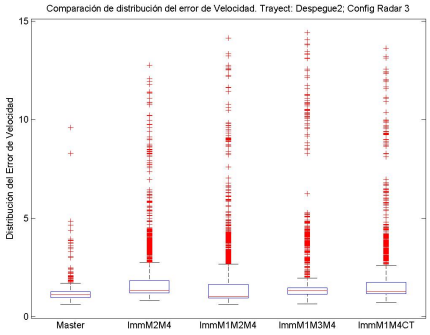
(c) Dist Errores Pos (Config.2)



(d) Dist Errores Vel (Config.2)



(e) Dist Errores Pos (Config.3)



(f) Dist. Errores Vel (Config.3)

Figura 6.50: Tray Despegue 2: Distribución del RSME en las Posiciones y Velocidad

6.1.11. Despegue en Pista 3 con Rodaje desde la Terminal

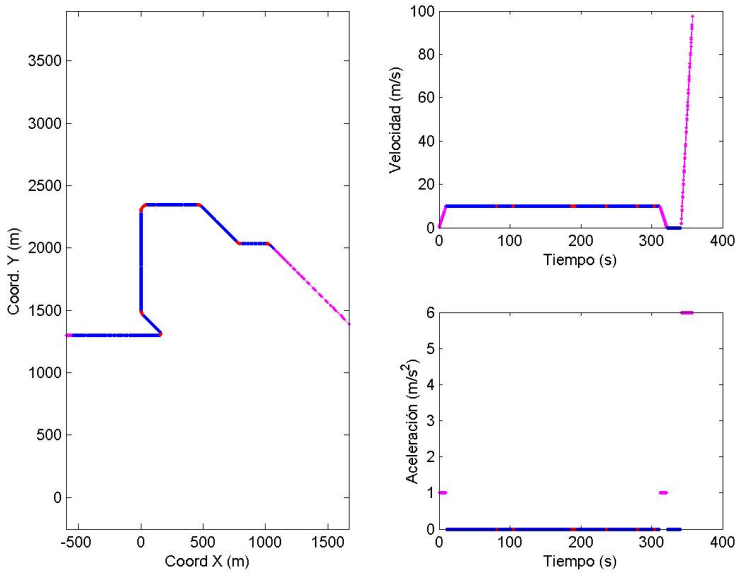


Figura 6.51: Descripción Escenario Despegue en Pista 3

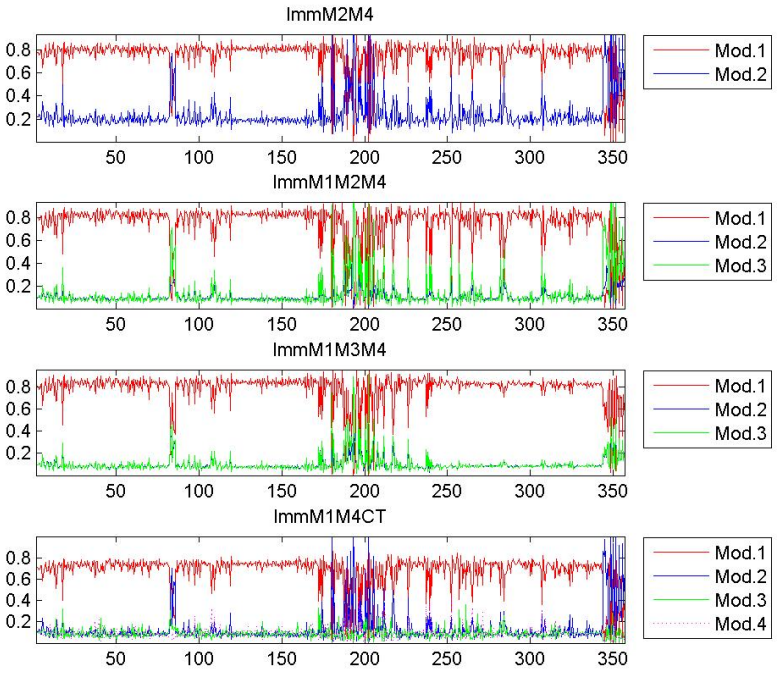


Figura 6.52: Tray Despegue 3: Probabilidades de Modo según cada Filtro IMM

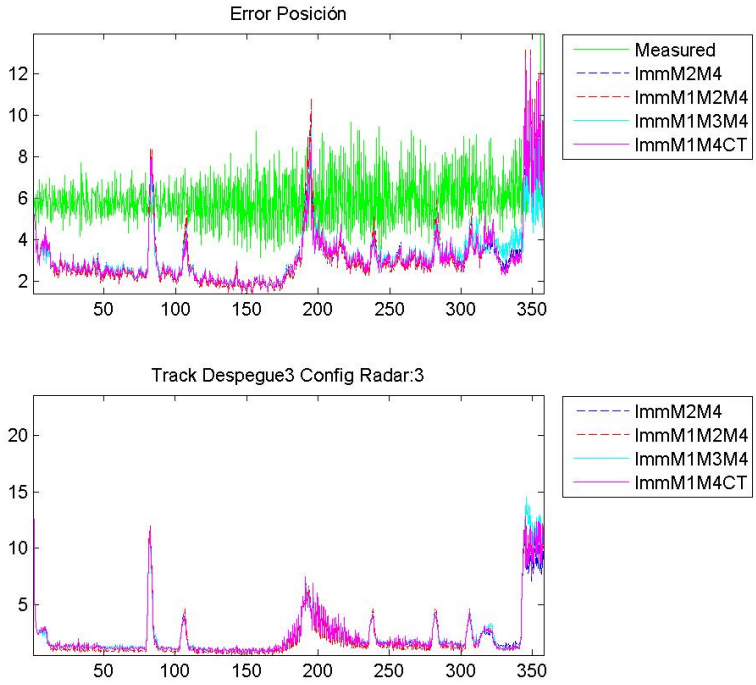
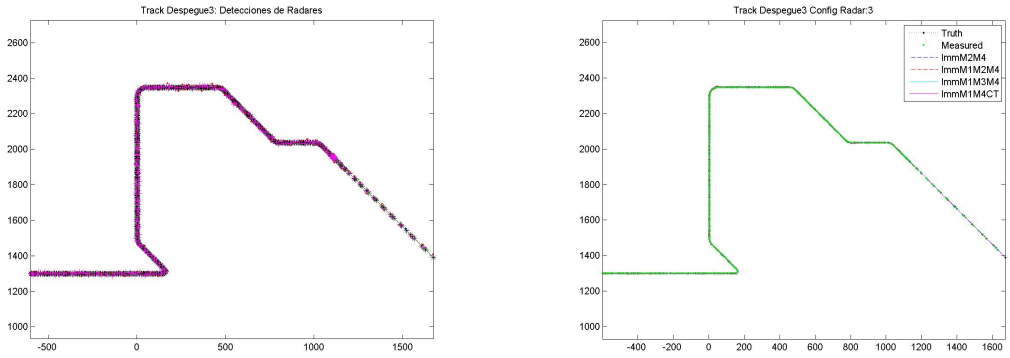


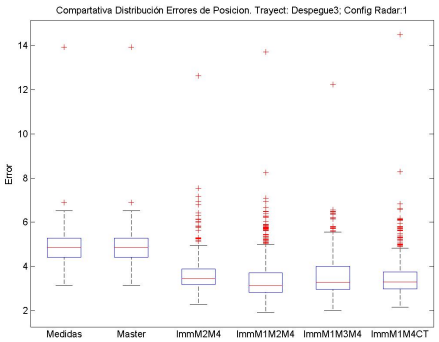
Figura 6.53: Tray Despegue 3: RSME de Posición y Velocidad



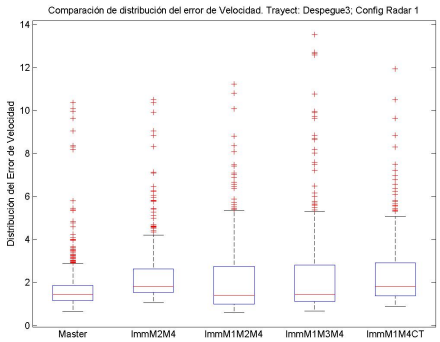
(a) Mediciones Radar

(b) Trayectorias Filtradas

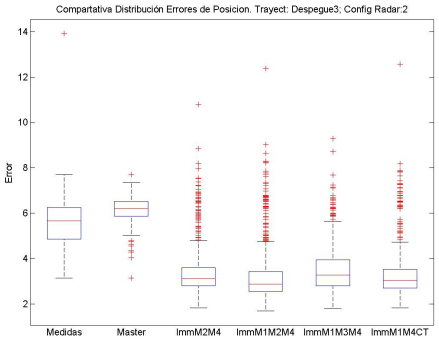
Figura 6.54: Tray Despegue 3: Detecciones Radar y Trayectorias Suavizadas por los Filtros



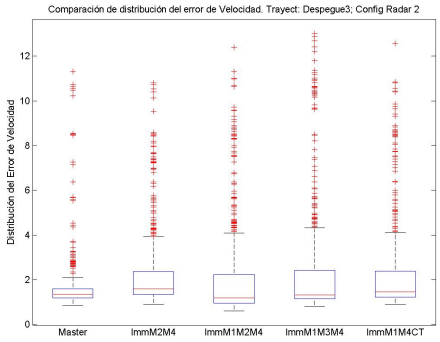
(a) Dist Errores Pos (Config.1)



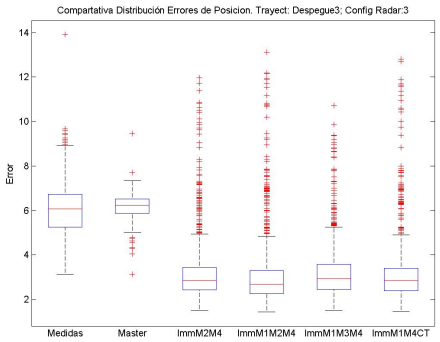
(b) Dist Errores Vel (Config.1)



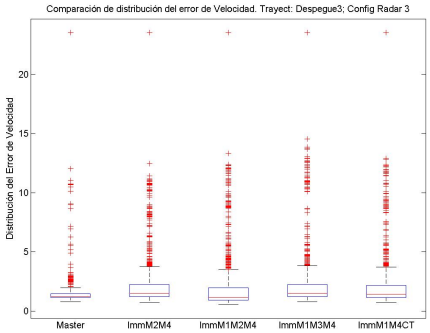
(c) Dist Errores Pos (Config.2)



(d) Dist Errores Vel (Config.2)



(e) Dist Errores Pos (Config.3)



(f) Dist. Errores Vel (Config.3)

Figura 6.55: Tray Despegue 3: Distribución del RSME en las Posiciones y Velocidad

6.1.12. Despegue en Pista 4 con Rodaje desde la Terminal

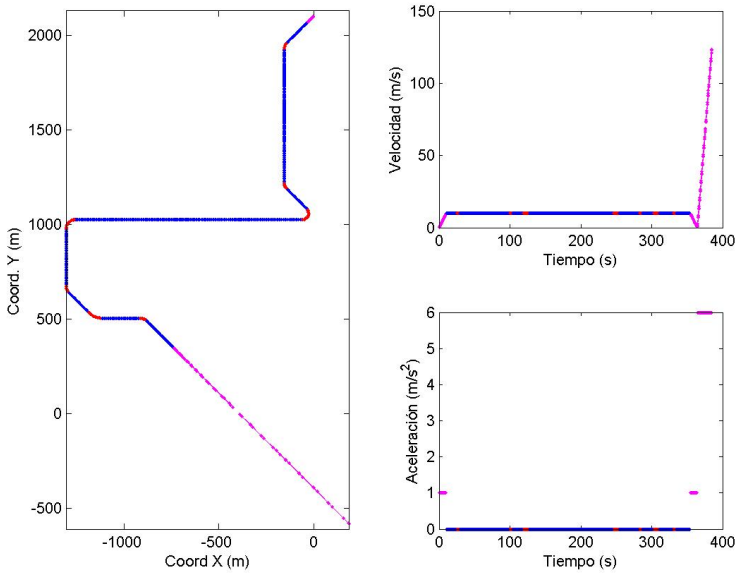


Figura 6.56: Descripción Escenario Despegue en Pista 4

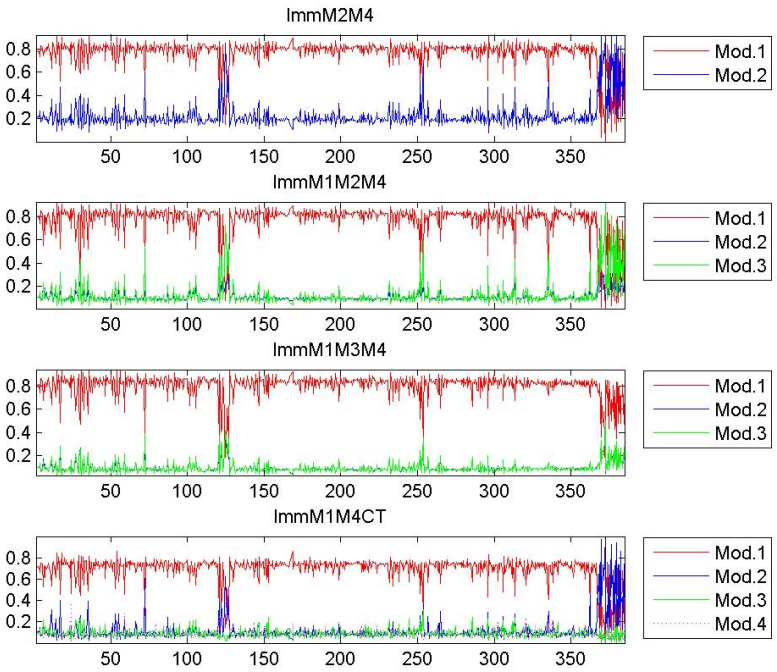


Figura 6.57: Tray Despegue 4: Probabilidades de Modo según cada Filtro IMM

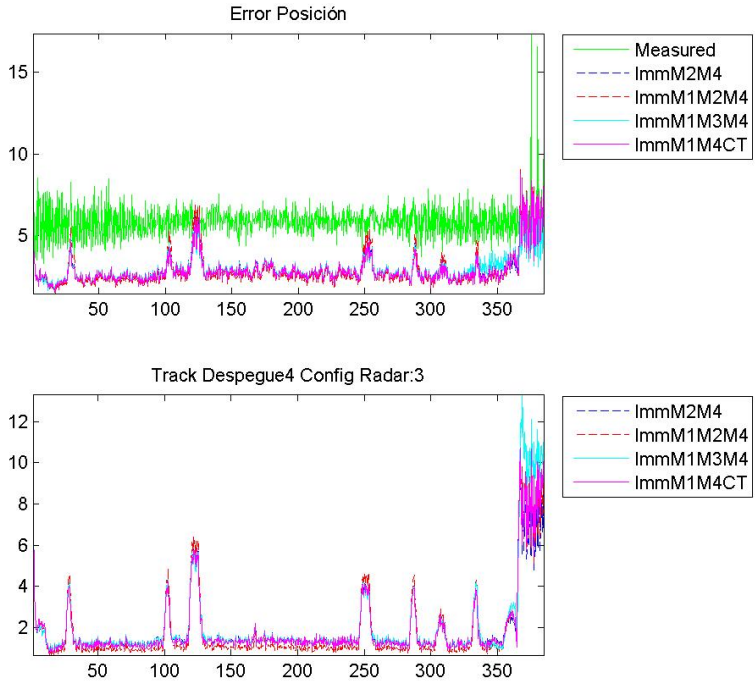
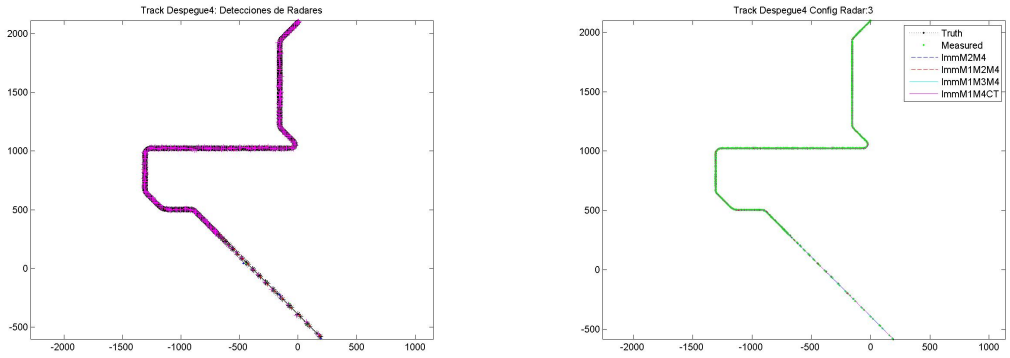


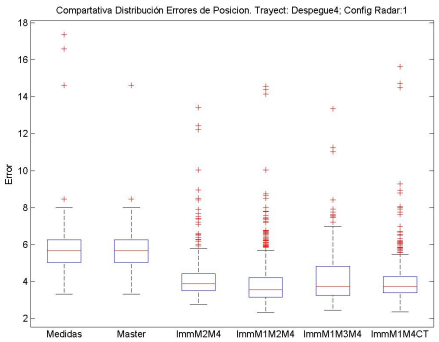
Figura 6.58: Tray Despegue 4: RSME de Posición y Velocidad



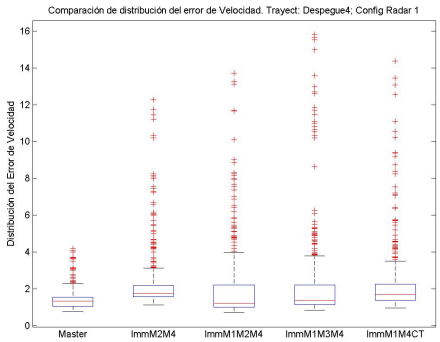
(a) Mediciones Radar

(b) Trayectorias Filtradas

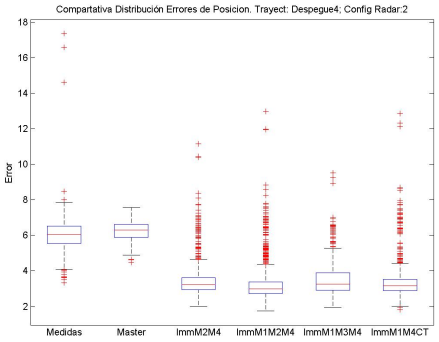
Figura 6.59: Tray Despegue 4: Detecciones Radar y Trayectorias Suavizadas por los Filtros



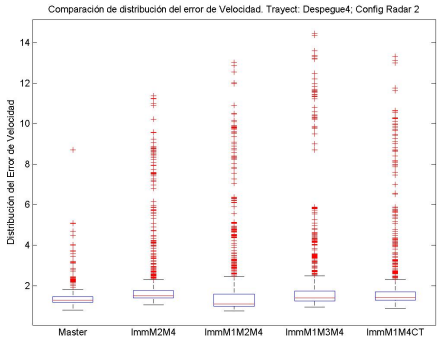
(a) Dist Errores Pos (Config.1)



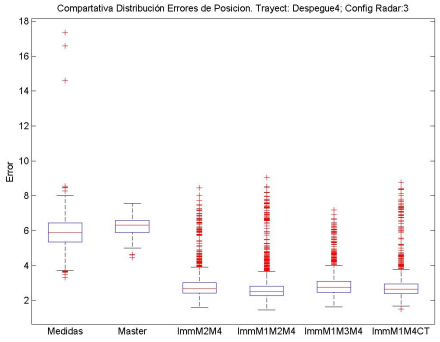
(b) Dist Errores Vel (Config.1)



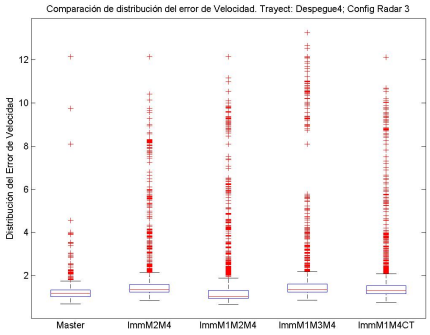
(c) Dist Errores Pos (Config.2)



(d) Dist Errores Vel (Config.2)



(e) Dist Errores Pos (Config.3)



(f) Dist. Errores Vel (Config.3)

Figura 6.60: Tray Despegue 4: Distribución del RSME en las Posiciones y Velocidad

6.1.13. Aterrizaje en Pista 1 aparcando en Terminal

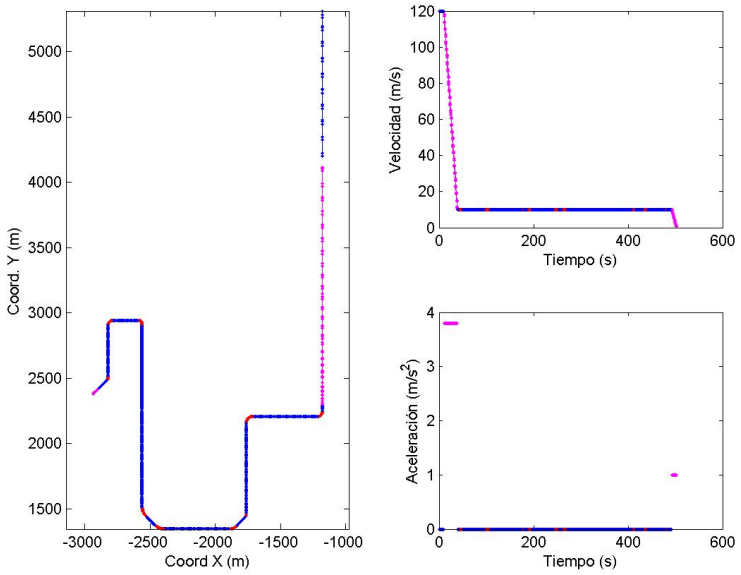


Figura 6.61: Descripción Escenario Aterrizaje en Pista 1

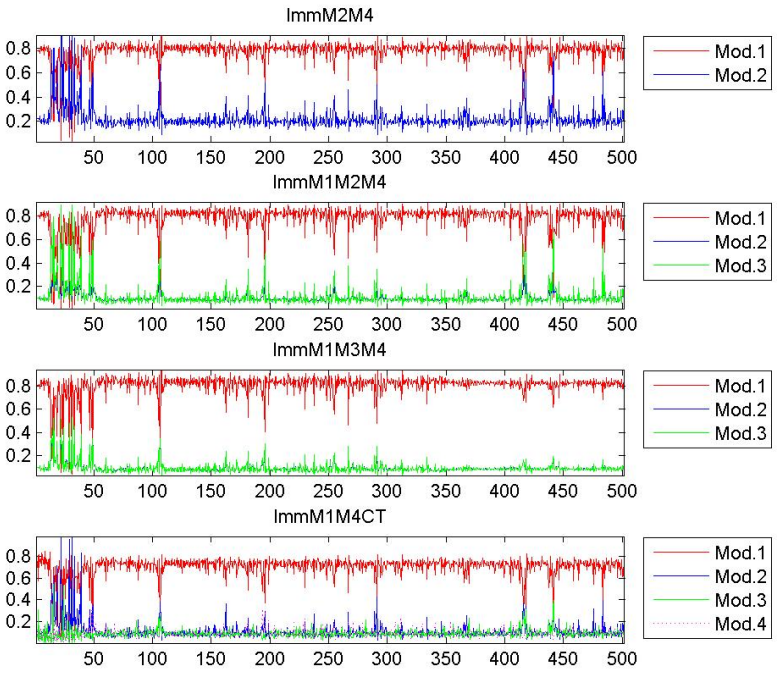


Figura 6.62: Tray Aterrizaje 1: Probabilidades de Modo según cada Filtro IMM

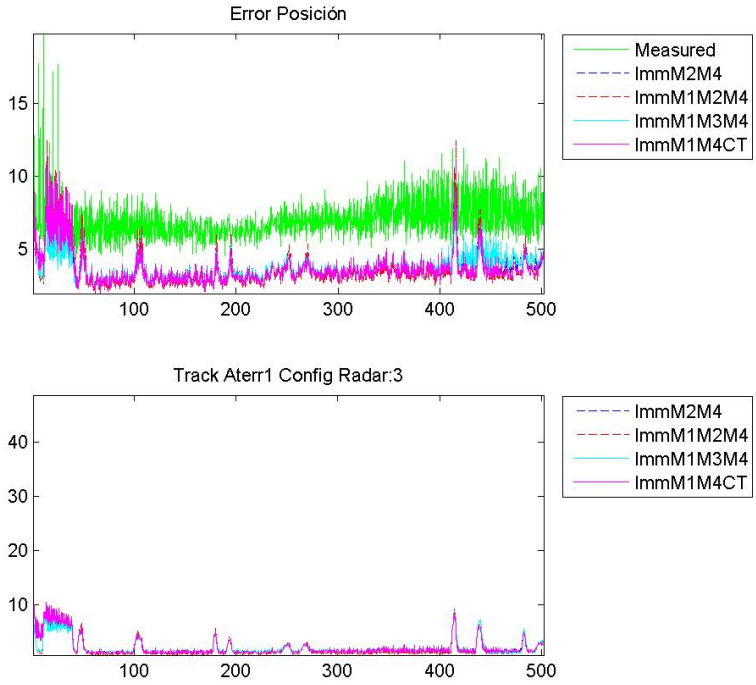
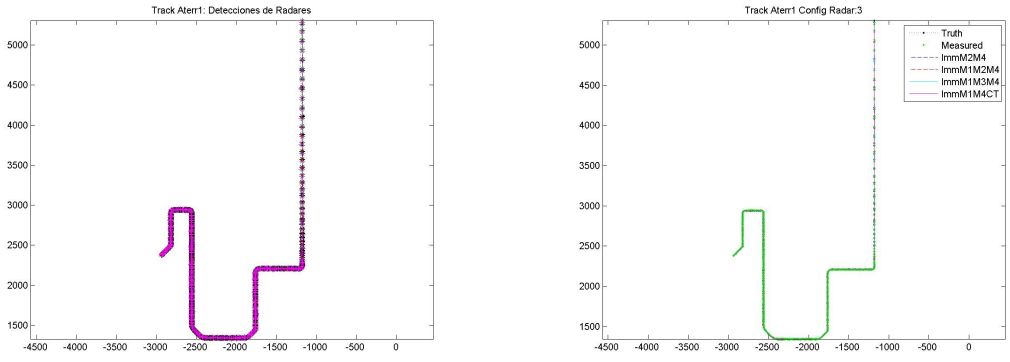


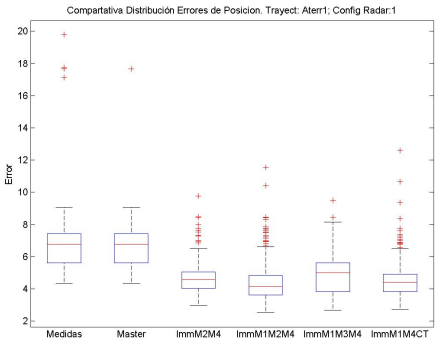
Figura 6.63: Tray Aterrizaje 1: RSME de Posición y Velocidad



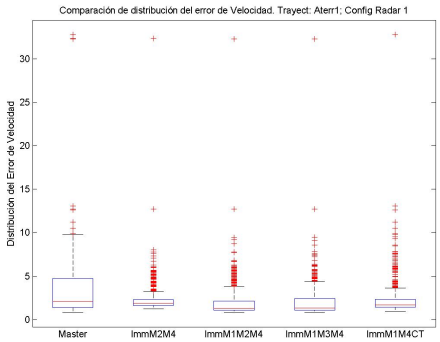
(a) Mediciones Radar

(b) Trayectorias Filtradas

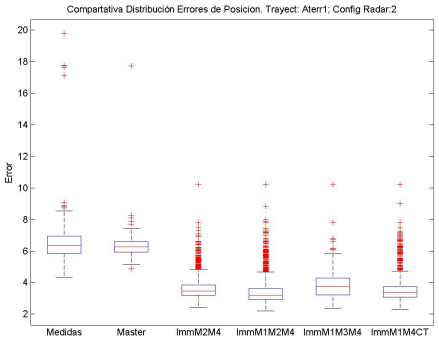
Figura 6.64: Tray Aterrizaje 1: Detecciones Radar y Trayectorias Suavizadas por los Filtros



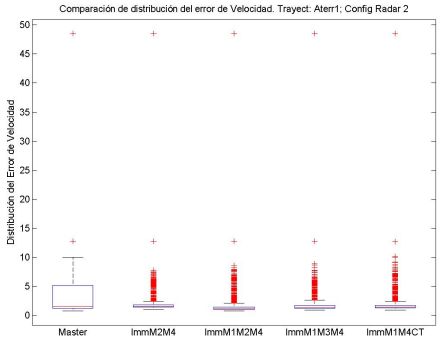
(a) Dist Errores Pos (Config.1)



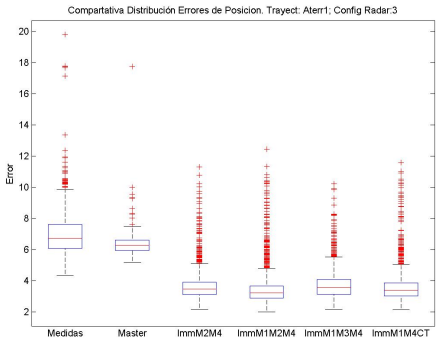
(b) Dist Errores Vel (Config.1)



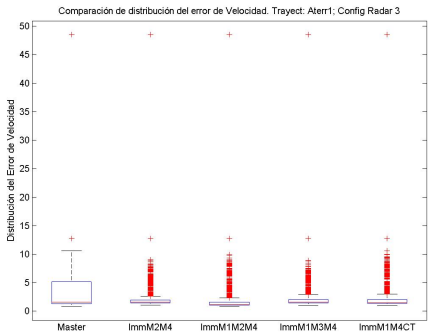
(c) Dist Errores Pos (Config.2)



(d) Dist Errores Vel (Config.2)



(e) Dist Errores Pos (Config.3)



(f) Dist. Errores Vel (Config.3)

Figura 6.65: Tray Aterrizaje 1: Distribución del RSME en las Posiciones y Velocidad

6.1.14. Aterrizaje en Pista 2 aparcando en Terminal

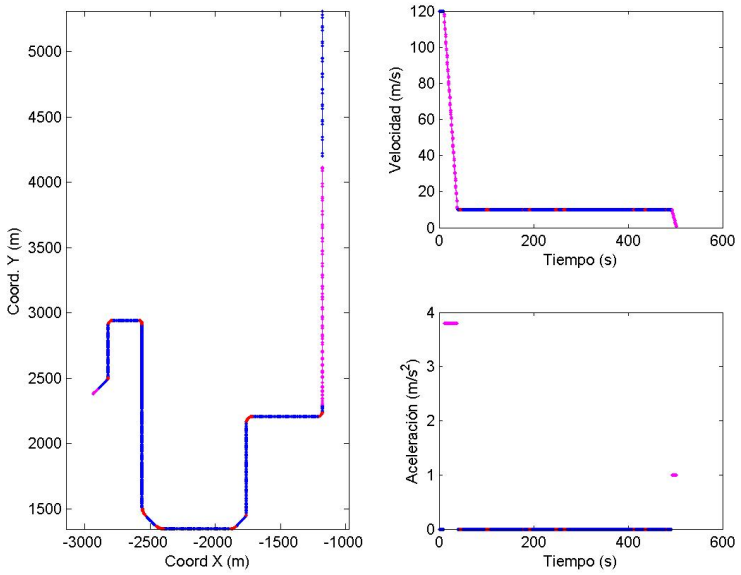


Figura 6.66: Descripción Escenario Aterrizaje en Pista 2

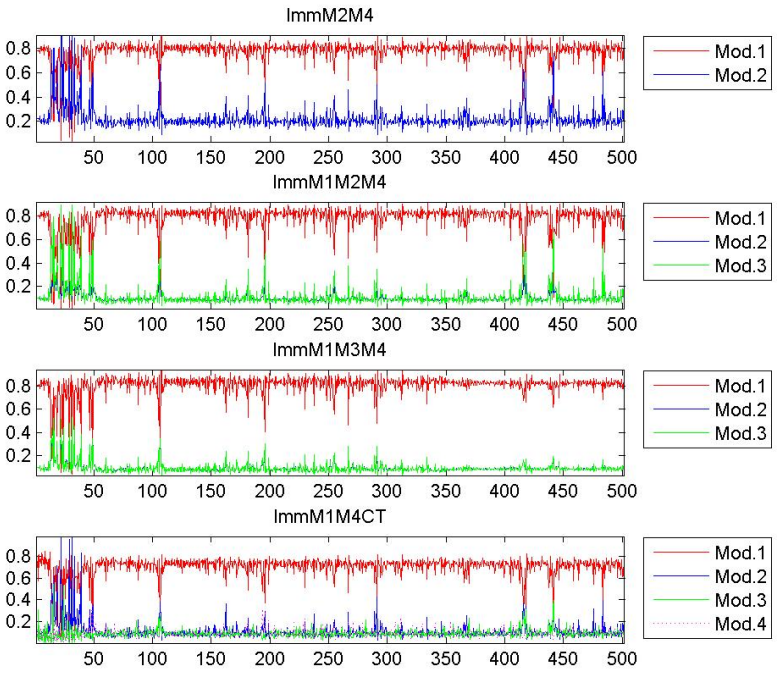


Figura 6.67: Tray Aterrizaje 2: Probabilidades de Modo según cada Filtro IMM

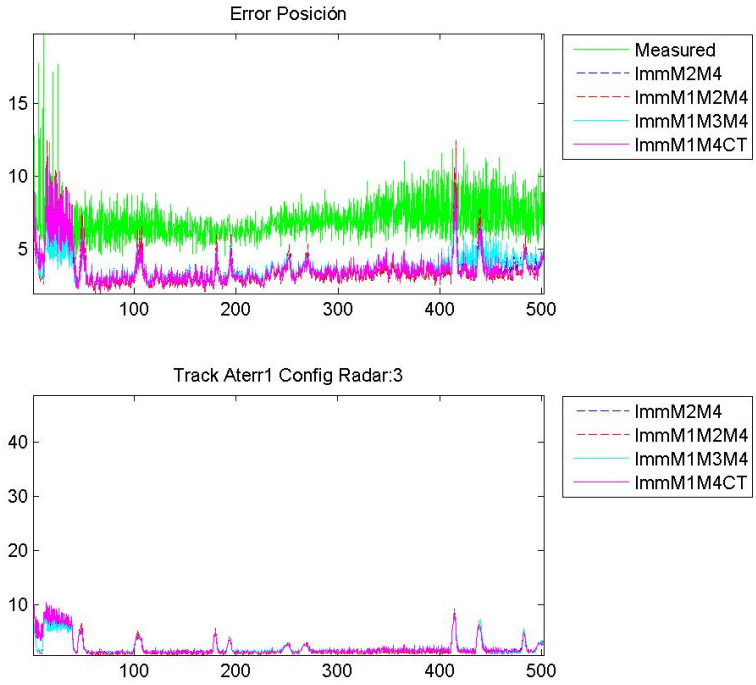
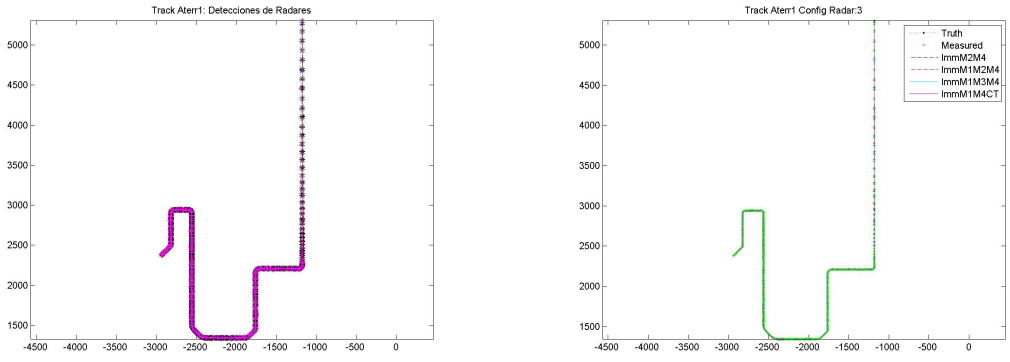


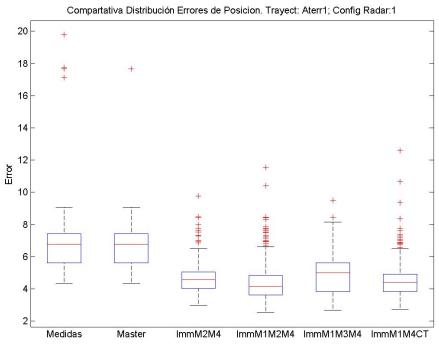
Figura 6.68: Tray Aterrizaje 2: RSME de Posición y Velocidad



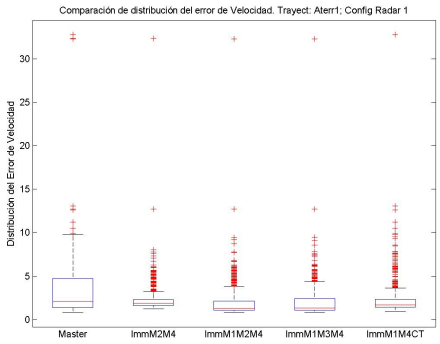
(a) Mediciones Radar

(b) Trayectorias Filtradas

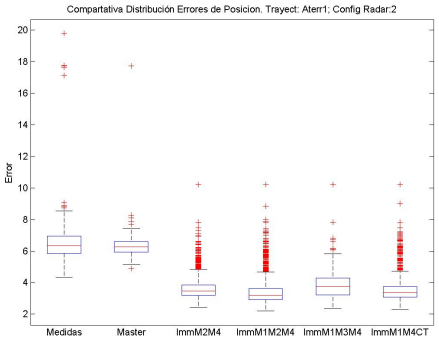
Figura 6.69: Tray Aterrizaje 2: Detecciones Radar y Trayectorias Suavizadas por los Filtros



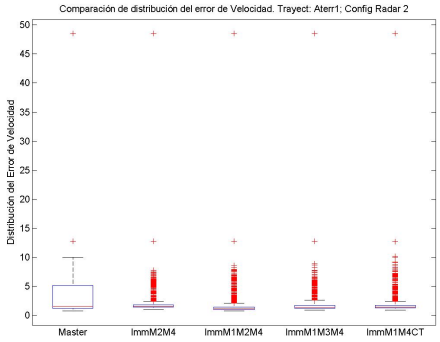
(a) Dist Errores Pos (Config.1)



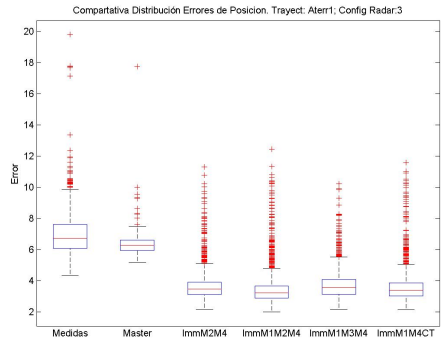
(b) Dist Errores Vel (Config.1)



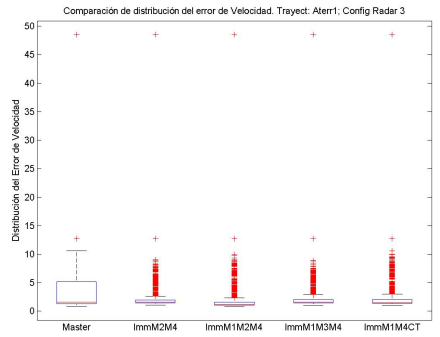
(c) Dist Errores Pos (Config.2)



(d) Dist Errores Vel (Config.2)



(e) Dist Errores Pos (Config.3)



(f) Dist. Errores Vel (Config.3)

Figura 6.70: Tray Aterrizaje 2: Distribución del RSME en las Posiciones y Velocidad

6.2. Análisis de Resultados

Se ha observado que al aplicar cualquiera de los cuatro tipos de filtros IMM diseñados a la hora de localizar los blancos circulando por un aeropuerto se obtienen resultados sensiblemente mejores a los del sistema de vigilancia actual que sólo actualiza la posición y cinemática de los blancos a partir de los datos del radar preferente (actualizando el estado del blanco con el dato denominado maestro o máster).

Esto ocurre con todas las configuraciones de sensores, apreciándose una mejoría notable cuando se incorporan nuevos radares y en especial los del tipo MLAT. Una vez entran en juego los sensores multilateración no se advierte reducción de los errores de localización tan destacada al incluir un segundo SMR El algoritmo ***Imm M1M2M4*** compuesto por dos modelos de segundo orden con distintos niveles de ruido y uno de tercer orden es el que con el que se obtienen valores más cercanos a los reales, logrando reducir los errores en la posición y cinemática de los blancos de forma muy significativa en todos los escenarios, con porcentajes de reducción de la Raíz del Error Cuadrático Medio en la posición que van del 25 % en el peor de los casos y alcanzando el 60 % cuando se utiliza la configuración de radares mas avanzada (ver apéndice C).

Por otro lado, la inclusión de los modelos de giros con velocidad angular prefijada (ej. filtro *Imm M1M2M5M6*) no resulta en estimaciones más precisas.

A todos los algoritmos IMM les cuesta ajustarse tras un cambio de modo de movimiento y es sobre todo palpable en las maniobras de giro (fig. 6.8).

Capítulo 7

Conclusiones

7.1. Conclusiones

En este trabajo de fin de Máster hemos intentado diseñar un filtro de estimación *Múltiples Modelos Interactuantes* (IMM) que fuese adecuado para su aplicación en la función de seguimiento de radar de un *Sistema Avanzado de Guía y Control del Movimiento en la Superficie* (A-SMGCS) y evaluar su comportamiento.

Para conseguirlo se ha llevado a cabo un estudio en profundidad del problema de seguimiento en la superficie de los aeropuertos, sus complicaciones e implicaciones (diversidad de movimientos, heterogeneidad de sistemas de detección) y examinado las técnicas de estimación de múltiples modelos basadas en filtros Kalman. Este proceso de investigación nos ha servido para ser capaces de proponer cuatro configuraciones de IMMs compuestos de distinto número de subfiltros Kalman, adaptados a varios tipos de movimientos, y evaluar la eficacia de cada una a la hora de estimar la posición y la velocidad de vehículos realizando maniobras típicas en los aeródromos.

Se ha implementado una herramienta con la que hemos simulado varios escenarios representativos de estas operaciones aeroportuarias y se ha analizado el comportamiento de los distintos filtros diseñados según tres configuraciones distintas de sensores. Además se ha comparado el desempeño de los filtros entre ellos y con el del sistema de vigilancia implantado en la actualidad en los aeropuertos que no realiza ningún filtrado de los datos.

El comportamiento de todos los filtros diseñados es muy positivo y con todos logramos reducir el error en la localización de manera significativa. Los datos de los experimentos muestran que la Raíz del Error Cuadrático Medio de la posición filtrada se reduce mas de un 25 % respecto a las mediciones sin filtrar en todos los escenarios al utilizar cualquiera del os filtros junto con la configuración de sensor más básica (un solo sensor de superficie más los datos del sistema de vigilancia en aire) y esta reducción llega a mas del 60 % en algunos casos cuando se utiliza la configuración de sensores como la del aeropuerto de Barajas.

El planteamiento con el que mejores resultados hemos alcanzado en todos los escenarios en general es el filtro IMM con un modelo de tercer orden y dos modelos de segundo orden con distintos niveles de ruido, pero habría que realizar un proceso de ajuste de los parámetros del filtros (ruidos de proceso y probabilidades de transición *a priori*) para conseguir una recuperación más rápida en las transiciones entre modos de movimiento y lograr mayor precisión en las maniobras de giro y cambios bruscos de velocidad.

La utilización de algoritmos IMM no solo tiene la ventaja de que reduce el error de localización sino que además, el análisis de la probabilidad de cada modo brinda la posibilidad de predecir el tipo de movimiento del blanco, lo que puede ser utilizado para prevenir colisiones, y en definitiva

para ayudar a conseguir la funcionalidad requerida en los sistemas A-SMGCS.

7.2. Trabajo Futuro

Lo primero que hay que tener en cuenta es que las probabilidades de transición de modos están fuertemente condicionadas por la región del aeropuerto por la que se esté moviendo un blanco, ya que las maniobras que puede realizar y la duración de las mismas dependen de la localización. En este trabajo hemos utilizado probabilidades de transición estáticas, que para una primera iteración de los filtros está bien, pero deberían modificarse para tener en cuenta las circunstancias particulares de cada caso. Además se hace necesario someter a un proceso de ajuste riguroso los demás parámetros de los filtros (como el ruido de proceso de cada modelo de movimiento). Una posibilidad que nos parece interesante es la aplicación de estrategias evolutivas para optimizar los valores de los parámetros ([1]).

Por otro lado, este trabajo se ha centrado en el problema de seguimiento de blancos independientes, así que habría que integrar el filtro en el sistema de vigilancia A-SMGCS que se presentó en la sección 1.2.2 (ver fig. 1.2) y examinar la adecuación del mismo al entrar en funcionamiento las demás funcionalidades del sistema (adquisición, preprocesado y procesado de datos , fusión de datos multi-sensor,...).

Acrónimos

A-SMGCS Sistema Avanzado de Guía y Control del Movimiento en la Superficie.

ACC Centro de Control de Aárea.

ADS-B Vigilancia Dependiente Automática-Broadcast.

ASDE Airport Surface Detection Equipment.

ASM Gestión del Espacio Aéreo.

ATC Control de Tránsito Aéreo.

ATFM Gestión del Flujo del Tránsito Aéreo.

ATM Gestión del Tránsito Aéreo.

ATS Servicios de Tráfico Aéreo.

CA Aceleración Constante.

CT Giro a Velocidad Angular Constante.

CV Velocidad Constante.

EDO Ecuación Diferencial Ordinaria.

EKF Filtro Extendido de Kalman.

IMM Múltiples Modelos Interactuantes.

KF Filtro de Kalman.

MLAT Sistema de Multilateración.

MM Múltiples Modelos.

OACI Organización de Aviación Civil Internacional.

PF Filtro de Partículas.

POO Programación Orientada a Objetos.

PSR Radar de Vigilancia Primario.

SES Cielo Único Europeo (Single European Sky).

SESAR Single European Sky ATM Research.

SJU SESAR SESAR Joint Undertaking (Agency of the European Commission).

SMR Radar de Movimiento en Superficie.

SSR Radar de Vigilancia Secundario.

TMA Área de Control Terminal.

TWR Torre de Control de Aeropuerto.

UKF Filtro de Kalman Unscented.

VS-IMM Filtro IMM de Estructura Variable.

Definiciones

A-SMGCS
Sistema que proporciona vigilancia, control, generación de rutas y guiado para el control de aeronaves y vehículos con el fin de mantener la tasa de movimiento en la superficie, asegurar el espaciado entre aeronaves, vehículos o ambos en el área de movimiento del aeródromo y mantener la capacidad del aeródromo en todas las condiciones meteorológicas text.

ATM
Administración dinámica e integrada (segura, económica y eficiente) del tránsito aéreo y del espacio aéreo, que incluye los servicios de tránsito aéreo, la gestión del espacio aéreo y la gestión de la afluencia del tránsito aéreo, mediante el suministro de instalaciones y servicios sin discontinuidades en colaboración con todos los interesados y funciones de a bordo y basadas en tierra.

ATZ
Zona de responsabilidad de la Torre de Control (TWR). Es un espacio aéreo pequeño de forma cilíndrica, que arranca desde el suelo y de altura dependiente de la visibilidad y que está centrado en un punto llamado ARP (Airport Reference Point), y cuyo radio suele ser de 5 NM..

Azimut
Angulo que con el meridiano forma el círculo vertical que pasa por un punto del globo terráqueo. En relación con el radar, es el ángulo del objetivo con relación al norte, medido en el sentido de las manecillas del reloj. Varía entre 0^o y 360^o y no se requiere indicar el cuadrante en el que se encuentra el objetivo detectado.

Blanco
Aeronave, vehículo u otro obstáculo que se visualiza en un display de vigilancia.

Calle de Rodaje
Vía definida en un aeródromo terrestre, establecida para el rodaje de aeronaves y destinada a proporcionar enlace entre una y otra parte del aeródromo.

Groundspeed
Velocidad absoluta (velocidad respecto a tierra).

MSE
Media aritmética de los cuadrados de las desviaciones del estimador (x1, x2, ..., xn) respecto al valor verdadero del estadístico que se trata de estimar.

Nudo
1 milla náutica por hora (1,852 km/h), es decir aproximadamente 0,5144 metros por segundo.

Pista del Aeropuerto
Área rectangular definida en un aeródromo terrestre preparada para el aterrizaje y el despegue de las aeronaves.

Plataforma
Área definida, en un aeródromo terrestre, destinada a dar cabida a las aeronaves para los fines de embarque o desembarque de pasajeros, correo o carga, abastecimiento de combustible, estacionamiento o mantenimiento.

Rango
Distancia desde el radar hasta el objetivo detectado a lo largo de la línea de visión.

RMSE
Raíz cuadrada de la media aritmética de los cuadrados de los errores.

Ruido Blanco Gaussiano
Señal aleatoria cuyos valores en instantes de tiempo distintos no tienen relación alguna entre sí (no existe correlación estadística entre sus valores), y cuya función de densidad responde a una distribución normal.

Torre de Control
Es una instalación elevada que desde su sala de control a través de un fanal se contempla visualmente el aeródromo y sus inmediaciones (zona ATZ) y donde varios controladores se encargan de gestionar el rodaje, el despegue y el aterrizaje de las aeronaves.

Variables de Estado
Conjunto de funciones del tiempo $x_i(t)$, que constituyen el vector de estado.

Área de Movimiento
Parte del aeródromo que ha de utilizarse para el despegue, aterrizaje y rodaje de aeronaves, integrada por el área de maniobras y las plataformas.

Bibliografía

[1] J. De Miguel G. Berlanga A. Molina M. Casar J. Besada, J. García. Design of imm filter for radar tracking using evolution strategies. *IEEE Transactions on Aerospace and Electronic Systems*, 2005.

[2] Organización de Aviación Civil Internacional. Advanced surface movement guidance and control systems (a-smgcs) manual. Technical report, International Civil Aviation Organization (ICAO), 2004.

[3] Organización de Aviación Civil Internacional. Los procedimientos para los servicios de navegación aérea-gestión del tránsito aéreo (pans-atm). Technical report, International Civil Aviation Organization (ICAO), 2007.

[4] Joana Barbosa Bastos Gomes. An overview on target tracking using multiple model methods. Tesis profesional, Universidad Técnica de Lisboa.

[5] SESAR JU. Improved surveillance for surface management: Phase 1 - technological study report, 2011. This document describes the results of the theoretical and technical study performed in the frame of the WP 12.3.1.

[6] Chih-Chung Ke. Literature on ground target tracking problems. Technical report, Center for Multisource Information Fusion, State University of New York at Buffalo, 1999.

[7] Peter S. Maybeck. *Stochastic models, estimation, and control*, volume 141 of *Mathematics in Science and Engineering*. Academic Press, 1979.

[8] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan. Interacting multiple model methods in target tracking: a survey. *Ieee Trans. On Aerospace and Electronic Systems*, pages 103–123, 1998.

[9] Rong Li X. Jilkov V. P. A survey of maneuvering target tracking. part i: Dynamic models, 2000.

[10] X. Rong Li and Yaakov Bar-Shalom. Design of an interacting multiple model algorithm for air traffic control tracking. In *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY*.

[11] Jilkov V.P. Rong Li X. A survey of maneuvering target tracking. part iii: Measurement models. In *Proceedings of SPIE Conference on Signal and Data Processing of Small Targets*, pages 423–446, Jul-Aug 2001.

[12] SER. Single european sky atm research. `http://www.sesarju.eu/programme/workpackages/wp-12-airport-systems--199`.

[13] SesarJU. Improved surveillance for surface management: Phase1-architecture design, 2011. This document describes the initial architecture of the A-SMGCS Surveillance function and how it interfaces with other components of the A-SMGCS and with external systems.

[14] Semerdjiev T. Simeonova I. Specific features of imm tracking filter design. *I&S*, 9, 2002.

[15] Greg Welch and Gary Bishop. An introduction to the kalman filter, 2001. `http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf` Updated Monday, July 24, 200.

[16] Thiagalingam Kirubarajan Yaakov Bar-Shalom, X.-Rong Li. *Estimation with Applications to Tracking and Navigation.* Wiley-Interscience, 2001.

Apéndices

Apéndice A

Código Fuente

A.1. Modelos de Movimiento

A.1.1. Clase Modelo

../code/Objetos/Model.m

```
classdef Model
    %MODEL Clase base que proporciona las propiedades y métodos comunes a todos los
        tipos de
    %modelos dinámicos y define un interfaz con las funciones que
    %deben proporcionar los modelos concretos
    %   Modelos particulares derivarán de estos (CV, CA, CT...) y deberán
    %   implementar las funciones comunes

    properties
        var_vx; %varianza del error de proceso en la coordenada x
        var_vy; %varianza del  error de proceso en la coordenada x
        stateInds=[]; %array con los índices de posiciones las variables de estado
                      %del modelo en el vector de estado "común"
        color
    end
    %interface, metodos que tienen que ser implementados por todas las clases derivadas
    methods (Abstract)
      % F=getF(T) %retorna la matriz de transición de estados
      %  G=getG(this,T) %retorna la matriz de transición de estados
       % H=getH(T) %retorna la matriz de medida
       %R=getR(T, errorMedida) %retorna la matriz R para para el instante T
    end

    methods(Abstract, Static)
        F=getTransitionMatrix(T);%retorna la matriz de transición de estados del modelo
            concreto
        G=getGananciaErrorMatrix(T);%retorna la matriz de transición de estados
        H=getMeasureMatrix();
    end
    %metodos
    methods
        function m=Model(dimsState,dterrorplanta, numModelVars, color)
            %constructor, necesita el vector de estado del filtro
            %y la desviación típica del error de proceso del modelo dinamico
            %inicializa los valores de los índices del
            m.var_vx=dterrorplanta;%2;
            m.var_vy=dterrorplanta;%2;
            m.stateInds= m.initIndexesArray(dimsState,numModelVars);
            if (nargin==4)
                m.color=color;
            end
        end
        %setter y getters
        function this = set.stateInds(this,value) %Value class
            this.stateInds=value;
        end
```

```
46          function systemMatrixes=getModelMatrixes(this, T, covErrMedida)
                %devuelve una estructura con todas las matrices del modelo de
                %estados
                systemMatrixes.F=this.getF(T);
                systemMatrixes.Q=this.getQ(T);
51              systemMatrixes.H=this.getH(T);
                %R depende de la medida.. me lo pasan
                %systemMatrixes.R=diag([dterrmedida dterrmedida]);
                systemMatrixes.R=covErrMedida;
                systemMatrixes.indices= this.stateInds;
56          end


            function Q=getQ(this,T)
                %devuelve la matriz que modela el ruido de planta
61              qv = diag([this.var_vx this.var_vx]);
                G=this.getG(T);
                Q = G*qv*transpose(G);
            end

66          function F=getF(this,T)
                %retorna la matriz de transición de estados
                F=this.getTransitionMatrix(T);
            end

71          function G=getG(this,T) %retorna la matriz de transición de estados
                G=this.getGananciaErrorMatrix(T);
            end

            function H=getH(this,T)
76              %retorna la matriz de medida. En principio asumimos que
                %las observaciones solo proporcionan la posicion x e y
                %TODO:cambiar para q tambien acepte vx y vy si el radar
                %proporciona las velocidades
                H=this.getMeasureMatrix(T);
81          end

            function [state]=getNextState(this, prevState, incT)
            %calcula el vector de estado en el tiempo T, dado el estado en T-1
                 qv=randn(1)*([this.var_vx ; this.var_vy]);
86               %x(k)=F*x(k-1)+ Gamma*verr
                 state=this.getF(incT)*prevState  +this.getG(incT)*qv;
            end



91
            function varsstate=getNumVarsState(this)
                 varsstate=length(this.stateInds);
            end

96           function plotdata(this, true, observations,filtered)
                xrow=1;
                yrow= floor(this.getNumVarsState()/2)+1;
                plot(filtered(xrow,:), filtered(yrow,:), '*:r',...
                    true(1,:), true(2,:),'+:k',...
101                 observations(1,:),observations(2,:),'o:c');
                axis equal;
                legend('Modelos CV');
            end
        end
106
        methods(Static)
            function stateind=initIndexesArray(statedims,numModelVars)
                halfvars=floor(statedims/2);
                statevars=floor(numModelVars/2);
111             stateind=[];
                for l=1:1:(statevars)
                    stateind([l statevars+l])=[l halfvars+l];
                end
            end
116


             function X = gauss_rnd(M,S,N)
                 if nargin < 3
                     N = 1;
121              end
                 L = chol(S)';
                 X = repmat(M,1,N) + L*randn(size(M,1),size(M,2)*N);
             end
```

```
126

    end
end
```

A.1.2. Clase Modelo Velocidad Constante

../code/Objetos/ModelCV.m

```
classdef ModelCV < Model
    %MODELCV Summary of this class goes here
    %   Detailed explanation goes here

    properties
    end

    methods
        function m=ModelCV(dimsState,errorplanta)
            %constructor, necesita el vector de estado del filtro
            %y la desviación típica del error de proceso del modelo dinamico
            %inicializa los valores de los índices del
            % inferiorto('Model');
            if (nargin==1)
                errorplanta=0.001; %default
            end
            m@Model(dimsState,errorplanta,4,'b.-');
        end
        %{
        function F=getF(this,T)
            %retorna la matriz de transición de estados
            F=this.getTransitionMatrix(T);
        end

        function G=getG(this,T) %retorna la matriz de transición de estados
            G=this.getGananciaErrorMatrix(T);
        end

        function H=getH(this)
            %retorna la matriz de medida. En principio asumimos que
            %las observaciones solo proporcionan la posicion x e y
            %TODO:cambiar para q tambien acepte vx y vy si el radar
            %proporciona las velocidades
            H=this.getMeasureMatrix(T);
        end
        %}
    end

     methods(Static)
         function F=getTransitionMatrix(T)
             if(nargin>0)
                 %retorna la matriz de transición de estados
                 F=[1 T 0 0
                     0 1 0 0
                     0 0 1 T
                     0 0 0 1];
             end
         end

        function G=getGananciaErrorMatrix(T) %retorna la matriz de transición de estados
            if(nargin>0)
                G=[T*T/2 0; T  0; 0  T*T/2; 0  T ];
            end
        end

        function H=getMeasureMatrix(T)
            %retorna la matriz de medida. En principio asumimos que
            %las observaciones solo proporcionan la posicion x e y
            %TODO:cambiar para q tambien acepte vx y vy si el radar
            %proporciona las velocidades
            H=[1 0 0 0; 0 0 1 0];
        end

     end
65
```

```
end
```

A.1.3. Modelo de Aceleración Constante

../code/Objetos/ModelCA.m

```
classdef ModelCA < Model
    %MODELCA Clase que define el Modelo dinámico de movimiento
    %uniformemente acelerado (Aceleración constante)
    %   Clase que deriva de la clase Model e implementa las funciones
    %   definidas en su interfaz abstracto, para proporcionar las matrices
    %   especializadas para este tipo de momimiento concreto.

    properties(Constant)
        modeldims=6;
    end

    methods
        function m=ModelCA(dimsState,errorplanta)
            %constructor, necesita el vector de estado del filtro
            %y la desviación típica del error de proceso del modelo dinamico
            %inicializa los valores de los índices del
            % inferiorto('Model');
            if (dimsState<ModelCA.getStateDims())
                error('Error, la dimension del vector de estado Global no puede ser
                    menor que la del modelo ')
            end
            if (nargin==1)
                errorplanta=0.1; %default
            end
            m@Model(dimsState,errorplanta,ModelCA.getStateDims(),'m.-');
        end

    end

     methods(Static)
         function stateDims=getStateDims()
             %retorna la dimension del vector de estado para este modelo
            stateDims= 6;
         end

         function F=getTransitionMatrix(T)
             if(nargin>0)
                 %retorna la matriz de transición de estados
                 F=[ 1   T   (T^2)/2   0   0   0
                     0   1    T        0   0   0
                     0   0    1   0    0   0
                     0   0    0   1    T    (T^2)/2
                     0   0    0   0    1   T
                     0    0   0   0    0    1];
             end
         end

         function G=getGananciaErrorMatrix(T)
             %         sigm_vx=0.         function G=getGananciaErrorMatrix(T) %retorna
                 la matriz de transición de estados
             if(nargin>0)
                 G=[T*T/2  0
                     T    0
                     1    0
                     0    T*T/2
                     0     T
                     0     1];
             end
        end

        function H=getMeasureMatrix(T)
            %retorna la matriz de medida. En principio asumimos que
            %las observaciones solo proporcionan la posicion x e y
            %TODO:cambiar para q tambien acepte vx y vy si el radar
            %proporciona las velocidades
            H=[1 0 0 0 0 0; 0 0 0 1 0 0];
```

```
65          end


        end

70  end
```

A.1.4. Modelo de Giro

../code/Objetos/ModelCT.m

```
classdef ModelCT< Model
    %MODELCT Clase que define el Modelo dinámico de movimiento
    %de giro con velocidad angular constante
4   %   Clase que deriva de la clase Model e implementa las funciones
    %   definidas en su interfaz abstracto, para proporcionar las matrices
    %   especializadas para este tipo de momimiento concreto.

    properties(Constant)
9       modeldims=4;
    end
    properties
        omega
    end
14
    methods
        function m=ModelCT(dimsState, omega,errorplanta)
            %constructor, necesita el vector de estado del filtro
            %y la desviación típica del error de proceso del modelo dinamico
19          %inicializa los valores de los índices del
            % inferiorto('Model');
            if (dimsState<ModelCT.getStateDims())
                error('Error, la dimension del vector de estado Global no puede ser
                    menor que la del modelo ')
            end
24          if (nargin<3)
                errorplanta=0.001; %default
                if (nargin==1)
                    omega=15*2*pi/360;
                end
29          end
            m@Model(dimsState,errorplanta,ModelCT.getStateDims(),'r.-');
            m.omega=omega;
        end

34      function F=getF(this,T) %retorna la matriz de transición de estados
            F=this.getTransitionMatrix(T, this.omega);
        end
    end

39   methods(Static)
        function stateDims=getStateDims()
            %retorna la dimension del vector de estado para este modelo
            stateDims= 4;
        end
44
        function F=getTransitionMatrix(T,Om)
            if(nargin>1)

                %controlar que Om no sea cero, si lo es, usar
49              %aproximaciones
                %retorna la matriz de transición de estados
                F=[1   (sin(Om*T))/Om   0  -(1-cos(Om*T))/Om ;
                   0       cos(Om*T)    0       -sin(Om*T)  ;
                   0  (1-cos(Om*T))/Om  1    (sin(Om*T))/Om;
54                 0      sin(Om*T)     0        cos(Om*T)  ];
            end
        end

        function G=getGananciaErrorMatrix(T)
59          %         sigm_vx=0.          function G=getGananciaErrorMatrix(T) %retorna
                la matriz de transición de estados
            if(nargin>0)
                G=[T*T/2  0; T    0; 0    T*T/2; 0     T];
```

```
            end
        end

        function H=getMeasureMatrix(T)
            %retorna la matriz de medida. En principio asumimos que
            %las observaciones solo proporcionan la posicion x e y
            %TODO:cambiar para q tambien acepte vx y vy si el radar
            %proporciona las velocidades
            H=[1 0 0 0 ;  0 0 1 0];
        end


    end


end
%{
dt = param{1};
w = x(5);
if w == 0
    coswt = cos(w*dt);
    coswto = cos(w*dt)-1;
    coswtopw = 0;

    sinwt = sin(w*dt);
    sinwtpw = dt;
else
    coswt = cos(w*dt);
    coswto = cos(w*dt)-1;
    coswtopw = coswto/w;

    sinwt = sin(w*dt);
    sinwtpw = sinwt/w;
end

F = [1 0 sinwtpw     coswtopw   0;...
     0 1 -coswtopw   sinwtpw    0;...
    0 0 coswt       -sinwt     0;...
     0 0 sinwt       coswt      0;...
     0 0 0           0          1];
x_k = F*x;
%}
```

A.2. Simulación Trayectorias

A.2.1. Clase Segmento

../code/Objetos/Segment.m

```
classdef Segment
    %SEGMENT
    properties
      mode       % modelo de movimiento q se rige en el tramo.
      tinterval  %  intervalo de tiempos entre los que se da el tramo
      stateini   %  vector estado inicial del tramo
      statefin   %
      omega
    end

    methods
        function segm = Segment(movement,timeinterval,initialstate,finalstate,omega)
            if nargin > 0 %Support calling with 0 arguments
                segm.mode = movement;
                segm.tinterval = timeinterval;
                segm.stateini = initialstate;
                segm.statefin = finalstate;
                segm.omega = omega;
            end
        end %

        function segment= preprocess(segment,segmentcell,initialstate,tini, filetrack)
        %{
```

```
        Constructor que crea un segmento a partir de un cell array y datos de
            inicialización
        Cada tramo/segmento 'segmentcell' es una celda con la siguiente información:
            1.Tipo de movimiento
            2.velocidad inicial     - requerido para modo CV
            3.velocidad final,      - requerido para modo CA
            4.aceleración inicial   - requerido para modo CA
            5.angulo de heading     - requerido para modo CA y CV
            6.distancia             - requerido para modo CV y CT:
                                    CV: metros durante los que se aplica CV
                                    (si la velocidad es 0,entonces aquí
                                    va el tiempo -en segundos- durante el cual
                                        tendrá esta velocidad)
                                    CT: ángulo en radianes
            7.omega(velocidad angular) - requerido para modo CV y CT:
                datos que no son necesarios para un tipo de movimiento o porque para
                    ese segmento no sean precios, aparecerá como NaN en la
    %
    segment.stateini=initialstate;
    segment.mode=segmentcell{1};
    segment.omega=segmentcell{7};
    heading=segmentcell{5}   ;
    distance=segmentcell{6};
    duration=0;
    vini=segmentcell{2};
    aini=segmentcell{4};
    switch segment.mode
        case 'CV'
            if (isnan(vini)) %si la velocidad inicial no me la pasan como
                parámetro,la cojo del estado
                vini= sqrt(segment.stateini(2)^2+ segment.stateini(5)^2);
            end
            if (~isnan(vini) && (vini >=0))
                if (~isnan(distance))
                    if (vini==0)
                        duration=distance; %tengo que hacer esto para el caso en
                            que vct =0
                    else %vini >0
                        duration=distance/vini;
                    end
                end
                segment.stateini([2 5])=[vini*cos(heading) vini*sin(heading)];
            end
            segment.stateini([3 6])=[0 0]; %aceleración a 0
            model=ModelCV(6,0);
        case 'CA'
            vfin=segmentcell{3};
            if (isnan(vini)) %si la velocidad inicial no me la pasan como
                parámetro,la cojo del estado
                vini= sqrt(segment.stateini(2)^2+ segment.stateini(5)^2);
            end
            if (~isnan(vfin) && ~isnan(aini))
                duration=(vfin-vini)/aini;
                segment.stateini([2 5])=[vini*cos(heading) vini*sin(heading)];
                segment.stateini([3 6])=[aini*cos(heading) aini*sin(heading)];
            end
            model=ModelCA(6,0);
        case 'CT'
            if (isnan(distance)||isnan(segment.omega))
                segment.omega=0;
                disp 'error, missing parameter para movtype CT '
            else
                if ((abs(segment.omega)>0))
                    duration=abs(distance/segment.omega);
                end
            end
            Om=segment.omega;
            model=ModelCT(6,Om,0);
    end
    segment.tinterval=[tini tini+duration];
    %segment.statefin(model.stateInds)=model.getF(T)*segment.stateini(model.
        stateInds);
    segment.statefin(model.stateInds)=model.getNextState(segment.stateini(model.
        stateInds),duration);
    fprintf(filetrack,'%s \t & %.2f & %.2f  & %.2f & %.2f & %.2f\t \\\\ \n',...
        segment.mode, tini, duration,vini,aini,segment.omega);
end

function disp(segm)
    fprintf(1,'Type Mov: %s\n ',...
```

```
                segm.mode);
        end % disp
    end
end
```

A.2.2. Clase Trajectory

Clase que implementa las funciones necesarias para simular una trayectoria basandose en segmentos. 4

../code/Objetos/Trajectory.m

```
classdef Trajectory
    %TRAJECTORY Clase trayectoroia

    properties
        id
        posIni %posición inicial
        segments %array de segmentos
    end

    methods
        function traj = Trajectory(trajid,initialpos,listatramos)
            % Constructor de Objeto trayectoria, inicializa la posición
            % inicial ,el id de la trayectoria y los tramos que la forman.
            if nargin > 0 % Support calling with 0 arguments
                traj.id = trajid;
                traj.posIni = initialpos;
                traj.segments = listatramos;
            end
        end %

        function [duration]=getDuration(traj)
            %retorna la duración de la trayectoria perfecta (el tiempo final
            %del último tramo
            duration=traj.segments(length(traj.segments)).tinterval(2);
        end

        function [numtramos]=getNumTramos(traj)
            %retorna la duración de la trayectoria perfecta (el tiempo final
            %del último tramo
            numtramos=length(traj.segments);
        end

        function  [legend]=plot(traj)
            legend=['Tray: ' traj.id];
            legend=[legend ',(' num2str(length(traj.segments)) ' tramos'];
        end %

        function [truetraj]=generateTruth(traj, timeline, plotsegms)
            %genera una la trayectoria "verdadera" para la linea del
            %tiempo dada.
            truetraj=[];
            if (nargin>2)
                plotsegments=plotsegms;
            else
                plotsegments=0;
            end
            for s=1:length(traj.segments)
                %el estado actual se actualiza a apartir del último
                segment=traj.segments(s);
                tinterval=segment.tinterval;
                switch      segment.mode
                    case     'CV' %movimiento rectilineo a velocidad constante
                        model=ModelCV(6,0);
                    case     'CT'
                        model=ModelCT(6,segment.omega,0);
                    case     'CA'
                        model=ModelCA(6,0);
                end
                %obtenemos los índices de las variables del estado en el estado global
                tend= timeline <= tinterval(2);
                tindices=find (timeline(tend)>tinterval(1)); %asi tengo los indices de
                    los times de ese segment
```

```
                tramoState=zeros(6,length(tindices));
                for i=1:length(tindices)
                    % si sigo dentro de la duraccion del segmento calculo el incremento
                    % de tiempo y el estado
                    incT=timeline(tindices(i))-tinterval(1);
                    tramoState(model.stateInds,i)=model.getNextState(segment.stateini(
                        model.stateInds),incT);
                end
                %el los estados del tramo los añadimos al cont de estados del a
                %trayectoria
                truetraj=[truetraj tramoState];
                %aquí podría plot el segment...
                if(plotsegments)
                    traj.plotstate(tramoState,timeline(tindices),model.color, 0);
                    %Sacamos al fichero :
                    %fprintf(trackfile,'(%d)\t& %s \t. & [%.2f,%.2f ] s. & %.2f & %.2f &
                        %.2f\t\n',s,segment.mode,tinterval(1),tinterval(2));
                end
                clear  segment tramoState;
            end
            if(plotsegments)
                hold off;
                % %guardar el gráfico con la trayectoria
                % global Global_PathFigs;
                % print(f, '-djpeg',[Global_PathFigs traj.id '.jpeg'] );
            end
        end

        function plotstate(traj, statev,timeline,s, posonly)
            global GLOBAL_Markers ;
            if(ischar(s))
                col=s;
            else
               col=GLOBAL_Markers{mod(s-1, length(GLOBAL_Markers))+1};
            end
            subplot(2,2,[1 3]);
            plot(statev(1,:),statev(4,:),col)
            axis equal;
            ylabel('Coord. Y (m)');
            xlabel('Coord X (m)');
            if (s==1)
                title(sprintf('Tray: %s (%d tramos)', traj.id,length(traj.segments)));
            end
            hold on;
            if (~posonly)
                subplot(2,2,2);
                v=[statev(2,:);statev(5,:)];
                V_mod=dot(v,v).^(1/2);
                plot(timeline,V_mod,col);
                ylabel('Velocidad (m/s)');
                xlabel('Tiempo (s)');
                hold on;

                subplot(2,2,4);
                a= [statev(3,:);statev(6,:)];
                a_mod=dot(a,a).^(1/2);
                plot(timeline,a_mod,col);
                ylabel('Aceleración (m/s^2)');
                xlabel('Tiempo (s)');
                hold on;
            end
        end
    end

    methods (Static)
        function [trueTrajectory]=preprocessTrajectory(trajDesc,dims, trajDefPath)
            %{
            %% @function [trueTrajectory]=preprocessTrajectory(atray)
            Función que preprocesa los datos de una trayectoria predefinida como una
                posición inicial y  un
            conjunto de tramosen un array de celdas.

            La función realiza un preprocesado generando la definición en forma
            analítica de la trayectoria perfecta (para cada segmento se define:
            vector de estado inicial
            vector de estado final
            tiempo inicial
            tiempo final
            tipo de movimiento
```

```
            %% In:
            trajDesc   -   array de celdas con información de los tramos/segmentos que
                forman la
            trayectoria. Cada tramo/segmento es una celda con la siguiente información:
                1.Tipo de movimiento
                2.velocidad inicial      - requerido para modo CV
                3.velocidad final,       - requerido para modo CA
                4.aceleración inicial    - requerido para modo CA
                5.angulo de heading      - requerido para modo CA y CV
                6.distancia              - requerido para modo CV y CT:
                                           metros durante los que se aplica CV
                                           o ángulo en radianes(para el caso de CT)
                7.omega(velocidad angular) - requerido para modo CV y CT:
                    datos que no son necesarios para un tipo de movimiento o porque para
                        ese segmento no sean precios, aparecerá como NaN en la
                    definición

            %% Out:
            trueTrajectory   -  Trayectoria definida de forma analítica consistente en
                un array de estructuras. Cada estructura es la definición analítica de
                    un
                tramo, con los siguientes campos
                    mode       -   modelo de movimiento q se rige en el tramo.
                    tinterval   -   intervalo de tiempos entre los que se da el tramo
                    stateini    -   vector estado inicial del tramo
                    statefin    -   estado al final det tramo
            %}

            [trajfile, msg]=fopen(trajDefPath,'a+');
            %fprintf(trajfile, '\n\n%s  Trayectoria: %s. (%d tramos) \n ', datestr(now),
                trajDesc.id, length(trajDesc.segments));
            fprintf(trajfile, '\n \\begin{table}[h] \n \\caption{ Trayectoria: %s. (%d
                tramos)} \n',trajDesc.id, length(trajDesc.segments));
            fprintf(trajfile, '\\begin{tabular}{llcccc} \n \\hline \n \\hline \n' );
            fprintf(trajfile, ' \\multicolumn{6}{c}{Trayectoria: %s. (%d tramos)} \\\\ \
                n  \\hline \n \\hline  \n ',trajDesc.id, length(trajDesc.segments));
            fprintf(trajfile, 'Mov & Inicio&Durac (segs) & Vel(m/s) & Acel(m/s2) & Tasa
                Giro(rad)  \\\\ \n \\hline \n');

            %genero un objeto truesegment inicial que es el anterior al actual
            initruesegment.tinterval=[0 0];
            initruesegment.stateini=zeros(dims,1);
            initruesegment.statefin=initruesegment.stateini;
            %estado inicial del segmento e inicialamos la pos
            initruesegment.stateini([1 floor(1+dims/2)])=trajDesc.inipos;
            initruesegment.statefin=initruesegment.stateini;
            %trueTrajectory=cell(1,length(trajDesc)-1);
            segminitstate=initruesegment.stateini;
            tini=0;

            for s=1:length(trajDesc.segments)
                segmentcell=trajDesc.segments{s};
                %    initruesegment=preprocessSegment(segment, initruesegment);
                trajSegment=Segment(NaN,[tini tini],segminitstate,segminitstate,NaN);

                trajSegment=trajSegment.preprocess(segmentcell, segminitstate, tini,
                    trajfile);
                segminitstate=trajSegment.statefin;
                tini=trajSegment.tinterval(2); %tiempo final del segmento anterior
                %   tramos(s)=initruesegment;
                arraytramos(s)=trajSegment;
            end
            trueTrajectory=Trajectory(trajDesc.id,trajDesc.inipos,arraytramos);

            fprintf(trajfile, '\\hline \n \\hline \n \\end{tabular} \n \\label{tab:%s}
                \\end{table}',trajDesc.id);
            fclose(trajfile);
        end
    end
end
```

A.3. Simulación de Radares

A.3.1. Clase Radar

Clase base que implementa las funciones comunes a todos los tipos de radares

../code/Objetos/Radar.m

```
classdef Radar
    %RADAR Clase que incluye la información de un radar
    %   Un radar consiste en tipo, prioridad, precision, refresh-rate
    %dependiendo del radar, detecta x y , range azimut , v...

    properties
        id
        tipo %SMR MLT o AIRE
        prioridad %depende del tipo
        dterror %desviación típica del error [dt1,dt2...]'
        refreshRate

        posRadar %posición del radar ([x,y]'
        alcance %en metros
        minTargetSpeed %minima velocidad detectable, vale 0 por defecto. se usa para el
            caso de
        %radar de aire, consideramos que el blanco está en el aire si se
        %mueve a mas de 70 m/s
    end



    methods (Static)
        function distance=getDistance(p1,p2)
            %devuelve la distancia euclídea entre dos vectores
            vdist=p1-p2;
            distance=sqrt(dot(vdist,vdist));
            %distance2=sqrt((p1-p2)*(p1-p2)')
        end
    end

    methods
        function this=Radar(id, tipo, refreshrate, posicionRadar, alcance, priority,
            paramserror, minDetectableSpeed)
            %Constructor de radar,
            %crea un objeto de tipo radar con lso parámetros que se le
            %pasan.
            %Los datos de los errores de detección, de momento los
            %ponemos fijos.
            this.id=id;
            this.tipo=tipo;
            this.prioridad=priority;
            this.posRadar=posicionRadar;
            this.alcance=alcance;
            this.refreshRate=refreshrate;
            this.dterror=paramserror;
            if (nargin == 8)
                this.minTargetSpeed =minDetectableSpeed;
            else
                this.minTargetSpeed =0;
            end
        end

        function detected=isDetectable(this, position, vel)
        %función que decide si un objeto en la posición 'position'
        %moviendose a velocidad 'vel' es detectable por el radar
        %el radar lo puede detectar si eta dentro del alcance y si la
        %velocidad que lleva es mayor que la mínima
            detected= this.getDistance(position,this.posRadar)<this.alcance;
            %dot(vel,vel).^(1/2)
            detected= detected &&  ( sqrt(dot(vel,vel)) >= this.minTargetSpeed);
        end


        function [measures trueTraj]=generateMeasures(this,plotsegments,
            perfectTracjectory, timeline)
            %% función que genera detecciones de radar para los instantes de
            %tiempo de time line. Devuelve una estructura con los campos:
            %'measure' 'errmeas' 'radar''time',{}
```

```
            % donde
            %   measure es la medida de radar en coordenadas cartesianas (x,y y quizá vx
                y vy)
            %   errmeas es la matriz de covarianza del error de medida (R)
            %   time es el instante de tiempo en el que se realiza la
            %   detección

%old...para generar aquí la traj real, pero me la van a pasar
%             if( nargin == 3) %no me dan la trayectoria real para este timeline, la
    genero
%                 trueTraj=perfectTracjectory.generateTruth(timeline,plotsegments);
%             elseif ( nargin == 4) %me pasan la trayectoria para el timeline como
    parámetro
%                 trueTraj=perfectTracjectory;
%             end
            trueTraj=perfectTracjectory;
            numobservaciones=0;
            %measures={};%struct;%('measure', {}, 'errmeas', {}, 'radar', {},'time',{});
            for i=1:size(trueTraj,2) %generacion de la medida para cada pos
                %% obtenemos la pos en la trayectoria, si es dentro del alcance
                %del radar, generamos medida con el error correspondiente
                state=trueTraj(:,i);
                dimstraj=size(state,1);
                indexDim2=floor(dimstraj/2)+1;
                realpos=state([1; indexDim2]);
                realvel=state([2; indexDim2+1]);
                % Añadir medida si  el radar detecta el blanco
                %indexMeasure=length(measures)+1;
                %if(this.getDistance(realpos,this.posRadar)<this.alcance)
                if(this.isDetectable(realpos,realvel ))
                    %
                    [medidarad covR]=generaMedida(this,realpos);
                    numobservaciones=numobservaciones+1;
                    measures(numobservaciones)=RadarObservacion(timeline(i),medidarad,
                        covR, this.id, state,this.prioridad);
                end
            end
            if (numobservaciones==0)
                measures=[];
            end;
        end

        function [vmedida materror]=generaMedida(this,realpos)
            %% Esta funcion es para los radares por defecto (los que me den la posición
                del blanco
            % en coordenadas cartesianas
            vmedida= realpos+ this.dterror.*randn(2,1);
            materror=diag(this.dterror.^2);
        end

        function [timeline]=generateTimeLine(this,N)
            % Generar Bases de tiempos  N= número de detecciones/segundos
            % Retorna un vector con los tiempos de detección
            %Generación de incertidumbre aleatoria en el dT (no se admiten plots cuyo dT
                sea menor que un umbral)
            dT=this.refreshRate;
            incT=ones(1,N-1);
            sigmadT=dT/100;
            for i=1:(N-1)
                errdT=sigmadT*randn(1);
                if errdT<-dT/2
                    errdT=-dT/2;
                end
                incT(i)=dT+errdT;
            end
            %Creación de la base de tiempos del radar,
            %la primera detección se realiza
            %entre el segundo 0 y refreshRate
            timeline=zeros(1,N);
            timeline(1)=this.refreshRate*mean(rand(1,2));
            timeline(1)=abs( timeline(1)-floor( timeline(1)));

            for i=1:(N-1)
                timeline(i+1)=timeline(i)+incT(i);
            end

            %retornamos solo un 95% (como si se pierden el 5% de las
            %detecciones
            indices=(rand(1,N)<=0.95);
            timeline=timeline(indices);
```

```
        end

    end

end
```

A.3.2. Clase RadarPolares

Clase derivada de la clase radar que sobreescribe las funciones necesarias para la simulacion de los radares que aportan mediciones en forma rango y azimut, según se explicó en 4

../code/Objetos/RadarPolares.m

```
classdef RadarPolares < Radar
    %RADAR Clase que incluye la información de un radar
    %  Un radar consiste en tipo, prioridad, precision, refresh-rate
    %dependiendo del radar, detecta x y , range azimut , v...

    properties
        %id
        %tipo %SMR MLT o AIRE
        %prioridad %depende del tipo
        %dterror %desviación típica del error
        %refreshRate

        %posRadar %posición del radar
        %alcance %en metros
    end


    methods
        function this=RadarPolares(id, tipo, refreshrate, posicionRadar, alcance,
            priority, paramserror, minDetectableSpeed)
            %% Constructor de radar,
            %crea un objeto de tipo radar con lso parámetros que se le
            %pasan.
            %Los datos de los errores de detección, de momento los
            %ponemos fijos.
            this@Radar(id, tipo, refreshrate, posicionRadar, alcance, priority,
                paramserror,minDetectableSpeed)
        end



        function [poscart matR]=tempMeasure(this, truerange, trueazimut)
            %función que serviría para obtener las medidas del radar en
            %cartesianas mediante otro método.
            measrange=truerange+this.dterror(1)*randn(1);
            measangle=trueazimut+this.dterror(2)*randn(1);
            varerr=this.dterror.^2;
            numden=varerr(1)-varerr(2)*measrange^2;
            b=(varerr(1)+varerr(2)*measrange^2)/numden;
            angdoble= 2*measangle;
            matR=  (numden/2)*[b+ cos(angdoble) sin(angdoble); sin(angdoble) b-cos(
                angdoble)];

            %transformamos el angulo y azimut "medidos" a coord cartesianas:
            %habría que convertir el ángulo según el cuadrante para aplicar
            %esta operación:
            poscart=[measrange*cos(measangle)
            measrange*sin(measangle)];
         end

       function [vmedida materror]=generaMedida(this,realpos)
            %% la true_position proviene de la trayectoria real y está en
            %cartesianas, hay q obtener a partir  de ellas el
            %rango y azimut" y generar la posición medida a partir de estos y de los
                errores en la medida.

            %trasladamos la posicion al sistema de ref del radar
```

```
            posrelativa=realpos-this.posRadar;%posición centrada en el radar
            trueazimut=this.getAzimut(posrelativa);
            truerange=sqrt(dot(posrelativa,posrelativa)); %distancia del punto al radar
            %obtener la matriz jacobiana
            Jac=[cos(trueazimut) -truerange*sin(trueazimut)
                  sin(trueazimut) truerange*cos(trueazimut)];

            errorlineal=Jac*this.dterror;
            %z_k= H*x_k + v_k
            vmedida= realpos+ errorlineal.*randn(2,1);
            %genero la matriz de covarianza del error
            R=diag(this.dterror.^2);
            materror=Jac*R*transpose(Jac);
        end
    end

    methods (Static)
        function ang=getAzimut(pr)
            %% devuelve el azimut (ángulo respecto al norte, medido en el
            %sentido de las agujas del reloj
            angref=atan(pr(2)/pr(1));
            if(pr(1)>0)
                if (pr(2)>0) %x>0,y>0 primer cuadrante
                    ang=pi/2-angref;
                else %(pr(2)<0);%cuarto cuadrante
                    ang=pi/2-angref;%ang=pi/2+angref;  %igual pq me da el negativo!
                end
            else % negativo
                    if (pr(2)>0) %x<0  y>0 2C
                        ang=pi*3/2-angref;
                    else %x<0 y<0, 3C
                        ang=pi*3/2-angref;
                    end
            end
        end
            % if (p1(1)<0 && p2(1)>0)
                %los dos en el CI
            % ang=dot(p1,p2)/sqrt(dot(p1,p1)*dot(p2,p2))
    end



end
```

A.3.3. Simulación de Observaciones de radar

Clase para el manejo de cada observación de radar.

../code/Objetos/RadarObservacion.m

```
classdef RadarObservacion
    %MEDIDARADAR Clase Observación de Radar
    %   Detailed explanation goes here

    properties
        time %instante en que se detectó la medida
        measure %array de medidas [x,y,vx, vy..]
        errmeas %array de los errores de medida de tamaño igual al de la medida
        radar %id del radar que generó la medida
        prioridad
        truestate %estado verdadero en el tiempo time
    end

    methods
        function mr=RadarObservacion(time,variablesmedidas,errormed, idradar, truestate,
            prioridadradar)
            mr.time=time;
            mr.measure=variablesmedidas;
            mr.errmeas=errormed;
            mr.radar=idradar;
            mr.truestate=truestate;
            mr.prioridad=prioridadradar;
        end
```

```
    end

    methods (Static)
        function plotRadarObservations(observaciones, codcol)
            if(nargin<2)
                codcol='g:.';
            end
            if (~isempty(observaciones))
                %mirarlo pq igual hacemos que se saque de la informacion el
                %radar al que pertenece y se asigne un color segun el radar.
                medidas=[observaciones.measure];
                truest=[observaciones.truestate];
                plot(truest(1,:),truest(4,:),'k-*', medidas(1,:), medidas(2,:),codcol)
                %               plot(trajstate(1,:),trajstate(4,:),'k*', m1(1,:), m1(2,:)
                    ,'m:o' ,...
                %               trajstate2(1,:),trajstate2(4,:),'ko',m2(1,:), m2(2,:),'g:o
                    ',...
                %               trajstate3(1,:), trajstate3(4,:),'k+', m3(1,:), m3(2,:),'r
                    :+');

                % axis([-5000 5000 -3000 7000]);
                axis equal;
                title('Trayectoria y medidas');
            end
        end

        function [indexMasterObs] = getIndexMasterObservation(observaciones)
            %getIndexMasterObservationx Obtenemos los índices de las
            %observaciones maestras
            obstimes=[observaciones.time];
            endT=ceil(obstimes(length(obstimes)));
            indexMasterObs=[];%creamos el vector de índices de observaciones maestras
            for i=1:endT
                inds= (obstimes>(i-1))&(obstimes<=i);
                [val inds]= find(inds>0);
                %inds=obstimes(inds)<=i;
                if( isempty(inds))
                  % sprintf('no observations in interval (%d,%d)', i-1,i)
                    continue;
                end
                intervObs= observaciones(inds);
                %buscar la observación con mayor prioridad, (es prioridad
                %min) y la guardamos
                [maxval indexmax]=min([intervObs.prioridad]);
                indexMasterObs=[indexMasterObs inds(indexmax)];
             end
        end
    end
end
```

A.4. Función que implementa el algoritmo IMM

../code/filters/vImmFilt.m

```
%funcion que implementa el filtro IMM Interacting Multiple Model (IMM)
%Entrada:
% z = vector con la medida
% x_est_mod -  array de celdas con los vectores de estado estimados para
%              cada modelo en el paso anterior
% P_cov_mod -  Array de Celdas con las matrices de covarianzas predichas por cada
    modelo
% prmodo    - Vector con las probabilidades de modo en el paso anterior
% prTrans   -   Matriz de probabilidad de transición entre modelos.
% NxN prmod_mix Prob de premezclado: prb de que estuviese en modo i en k-1,
%dada la hipotesis de que en k está nen modo j
% ind_est   - Indices of state components for each model as a cell array
% dims      - Total number of different state components in the combined system
% SystModel -    Array de estructuras con las matrices de cada modo Q,R,A,B,u,H
%paso un array de matrices de sistema (hay una por cada modo)
%
%Salida:
% x_est        -   Vector de estado resultado (combinado)
% P_cov        -   Matriz de covarianza asociada al vector de estado
```

```
%                   (combinada)
% prmodo        -   Vector de probabilidades de modo
% x_est_mod     -   Array de Celdas con los estados predichos por cada modelo
% P_mod         -   Array de celdas con las matrices de covarianzas  (cada entrada j del
    array
%corresponde a la matriz de cov de estado de ese modo)

function[x_est,P_cov,x_est_mod, P_cov_mod, prmodo]=...
    vImmFilter(z, x_est_mod, P_cov_mod, TransPr,prmodo,...
    SystModel, indst, dims )

numModos=length(x_est_mod);

%probabilidades de modo predichas para instt k (c_j)
prmod_pred = prmodo*TransPr; %forma matricial

%probabilidad de mezclado (Probabilidad de q el modo en k-1 era i...)
prmod_mix=zeros(numModos);
for i=1:numModos
    for j=1:numModos
        prmod_mix(i,j) = prmodo(i)*TransPr(i,j) / prmod_pred(j); %factor de
            normalización
    end
end
%forma corta: prmod_mix2 = TransPr.*(prmodo'*(prmod_pred.^(-1)));

% Estimadores de estado mezclados  y covarianzas
 xmod_est_mix=cell(1,numModos);
 for j=1:numModos
     xmod_est_mix{j}=zeros(dims,1); %inicializo a 0
     for i=1:numModos
         xmod_est_mix{j}(indst{i})=xmod_est_mix{j}(indst{i})+ prmod_mix(i,j)*x_est_mod{i
             };
     end
 end
%forma corta, op matricial
%xmod_est_mix=x_est_mod*prmod_mix;

Pcov_mod_mix=cell(1,numModos);
for j=1:numModos
    %calculo el término de correccion de desviación
    Pcov_mod_mix{j} = zeros(dims,dims);
     for i=1:numModos
         ind=indst{i};
         estvar=(x_est_mod{i}- xmod_est_mix{j}(ind));
         Pcov_mod_mix{j}(ind,ind)=Pcov_mod_mix{j}(ind,ind)+prmod_mix(i,j)*(P_cov_mod{i}+
             estvar*estvar');
     end
end
%PREDICCION, filtros kalman
verosim = zeros(1,numModos);
for j=1:numModos
    %Estimadores premezclados
    xj=xmod_est_mix{j}(indst{j});
    Pj=Pcov_mod_mix{j}(indst{j},indst{j}); %!!! MIX!!!!! estaba mal!!P_cov_mod)

    [x,Pk,innovk,covS, LH]= vkalmanf(xj,Pj,z,SystModel(j));

    %hay q guardar/actualizar:el vector de estado del modo, matriz de
    %covarianza del mismo, las probabilidades de modo (y mirar si hay q
    %arrastrar las demás variables q devuelve Kalman.
    x_est_mod{j}=x;
    P_cov_mod{j}=Pk;

    aux=(innovk'/covS)*innovk;
   verosim(j)=(((2*pi)^(-1))*(det(covS))^(-1/2))*exp(-0.5*aux);
end

%Probabilidad de los modelos (p de q el vehiculo siga cada uno de los
modelos en el instante de observación)
%pmod_predTotal= prmod_pred*verosim'; %sum_1_nmods(pmod_pred[j]*verosim[j]
pmod_predTotal=sum(verosim.*prmod_pred);

% for j=1:numModos
%     prmodo(j)=(prmod_pred(j)*verosim(j))/pmod_predTotal;
% end
%version matricial
prmodo=prmod_pred.*verosim/pmod_predTotal;

for i=1:numModos, dummy=0;
```

```
    for j=1:numModos
96      if j~=i, dummy=dummy+c(1,j)*exp(-0.5*(c(2,j)-c(2,i))); end
    end
    modePr(i)=1/(1+dummy/c(1,i));
end
prmodo=modePr/sum(modePr);
101
%SALIDA: combinación de los estimadores de los modelos pra generar el estado
x_est = zeros(dims,1);
P_cov = zeros(dims,dims);
%filtrado final y su matriz de covarianza asociada
106 for i = 1:numModos
        x_est(indst{i}) = x_est(indst{i}) + prmodo(i)*x_est_mod{i};
end
%Matriz de covarianza asociada
for j=1:numModos
111     ind= indst{j};
        errEst=(x_est(ind)-x_est_mod{j});
        aumentoIncertid=errEst*errEst';
        P_cov(ind,ind)= P_cov(ind,ind) + prmodo(j)*(P_cov_mod{j}+ aumentoIncertid);
end
116 return
```

A.5. Filtrado de Observaciones Radar

../code/Objetos/FilterImmObj.m

```
classdef FilterImm
    %FILTERIMM Objeto con todo lo necesario para un filtro IMM
3   %   Incluye:
    %   el conjunto de modelos del filtro (con sus respectivos parámetros)
    %   para los filtros kalman
    %   Matriz de transición de modos
    %   Probabilidad de Modo
8
    properties
        Modelos
        matProbTrans %matriz de probabilidades de transición (puede que no sea fija)
        probModo %vecor de probabilidad de cada modo
13      stateVars %estructura con las variables del vector estado del filtro y su orden
            en el vector
        stateDims %numero de variables del vector de estado
    end

    methods
18      function this=FilterImm(modelosMov,matrizTransProb, probsMod, statevars,
            immstatedims)
            %constructor
            this.Modelos=modelosMov;
            this.matProbTrans=matrizTransProb;
            this.probModo=probsMod;
23          this.stateVars=statevars;
            this.stateDims=immstatedims; %podría sacarlo pero ya lo haré;
        end

        function varind=getVarIndex(this, varname)
28          %Devuelve el índice en el vector de estado de la variable "varname" ('x','y
                ',...)
            %si no existe dicha variable en el vector, devuelve 0
            if (isfield(this.stateVars,varname))
                varind=getfield(this.stateVars,varname);
            else
33              varind=0; %no existe
            end
        end


38      function [state_est_mod covP_mod state_est]=initFilterStateData(this,
            radObserv1, radObserv2)
            %inicializa las variables de los filtros del filtro a partir
            %de las dos primeras observaciones recibidas
            %devuelve los estados iniciales y las covarianzas de cada modo
            %en dos arrays de cells
43          state_est=zeros(this.stateDims,1);
```

```
            %calculamos el estimador de estado inicial, que es igual para cada modelo
                del filtro
            %a partir de las dos primeras observaciones.
            intervT=radObserv2.time-radObserv1.time;
            %si me proporcionasen v, inicializaría el filtro con ella
            if(intervT < 0.005)
                intervT=0.005;
            end
            velocidad= (radObserv2.measure-radObserv1.measure)/intervT;
            state_est([this.stateVars.x this.stateVars.y])=radObserv2.measure;
            state_est([this.stateVars.vx this.stateVars.vy])=velocidad;
            %ahora se genera la matriz de covarianza (es igual para cada
            %modelo)
            varx2=radObserv2.errmeas(1,1);
            vary2=radObserv2.errmeas(2,2);
            varxy2= radObserv2.errmeas(1,2);

            varx1=radObserv1.errmeas(1,1);
            vary1=radObserv1.errmeas(2,2);
            varxy1= radObserv1.errmeas(1,2);
            T=intervT;
            %covP=diag(ones(this.stateDims,this.stateDims))
            cov{1}=[varx1   varx1/T
                    varx1/T  (varx2+varx1)/T^2];
            cov{2}=[varxy1 varxy1/T; varxy1/T (varxy1+varxy2)/T^2];
            cov{3}=[vary1   vary1/T
                    vary1/T  (vary2+vary1)/T^2] ;
            %si hay
            if (isfield(this.stateVars,'ax'))
                state_est([this.stateVars.ay this.stateVars.ay])=0;
                %extender las matrices de covarianza:
                for i=1:length(cov)
                    cov{i}(:,this.stateVars.ax)=0;
                    cov{i}(this.stateVars.ax,:)=0;
                end
            end
            %covP=zeros(this.stateDims);
            %genero la matriz de covarianza
            covP=[cov{1} cov{2};transpose(cov{2})  cov{3}];
            %si en el vector de estado se incluye la omega,
            %en su posición,q será despues de todas las demás variables, le añadimos
                ceros
            if (isfield(this.stateVars,'w'))
                state_est(this.stateVars.w)=0;
                covP(:,this.stateVars.w)=0;
                covP(this.stateVars.w,:)=0;
            end
            %para cada modelo utilizo el mismo vector de estado y de
            %covarianzas pero solo las variables del mismo que requiere
            %cada modelo!
            for i=1:length(this.Modelos)
                ind=this.Modelos{i}.stateInds;
                covP_mod{i}=covP(ind,ind);
                state_est_mod{i}=state_est(ind);
            end
        end

        function [pos_err vel_err]=getEstimationError(this, trueState,estimations)
        %devuelve el error en la estimación y en la posición y en el módulo
        %de la velocidad
        varpos=[this.stateVars.x;this.stateVars.y];
        pos_err= trueState(varpos,:)-estimations(varpos,:);

        if (nargout==2)
            posvel= varpos+1;
            v_real=trueState(posvel,:);
            V_mod_r=dot(v_real,v_real).^(1/2);
            v_est=estimations(posvel,:);
            V_mod_est=dot(v_est,v_est).^(1/2);
            vel_err=V_mod_r-V_mod_est;
        end
        end

        function [RMSE_pos RMSE_vel]=getRMSE(this, trueState, estimations)
            %retornta el error cuadrático medio de las estimaciones
            %obtenidas con el filtro
            %@in:
            %estado 'real' (?) nxTimesLength
            %estados filtrados ( dims x Timeslength)
            [poserr velerr]=ErrorsManager.getStateErrors(trueState,estimations);
```

```
            RMSE_pos=ErrorsManager.getRMSE(poserr);
            RMSE_vel=ErrorsManager.getRMSE(velerr);
        end

        function plotFilteredResults(this,observaciones,fstates)
            % Plot the estimates
            %de momento plot las pos only
            ix=this.stateVars.x;
            iy=this.stateVars.y;
            times=[observaciones.time];
            truest=[observaciones.truestate];
            z= [observaciones.measure];
            %para mostrar los errores.. CAMBIARLO
            zerr= abs([observaciones.measure]-truest)    ;
            stateerr(1,:)=abs(fstates(ix,:)-truest(ix,:));
            stateerr(2,:)=abs(fstates(iy,:)-truest(iy,:));

            subplot(3,2,5)
            plot( [observaciones.time],zerr(1,:),'g:.',...
                [observaciones.time],stateerr(1,:),'r:.');
            axis tight;
            subplot(3,2,6)
            plot( [observaciones.time],zerr(2,:),'g:',...
                [observaciones.time],stateerr(2,:),'r:.');
            axis tight;

            subplot(3,2,[1 3])
            plot3(times,truest(ix,:),truest(iy,:),'b:+',...
                [observaciones.time],fstates(ix,:),fstates(iy,:),'r.-',...
                [observaciones.time], z(1,:),z(2,:),'g:o')
            axis tight;%equal;
            xlabel('Tiempo (s)');
            ylabel('Coord X (m)');
            zlabel('Coord Y (m)');

            subplot(3,2,2)
            plot( [observaciones.time],truest(ix,:),'b');
            hold on;
            plot([observaciones.time],fstates(ix,:),'r-.');
            hold on;
            plot([observaciones.time],z(1,:),'g:');
            hold off;
            legend ('true x', 'filtered x', 'observed x');
            xlabel('Tiempo (s)');
            ylabel('X (m)');
            subplot(3,2,4)
            plot(times,truest(iy,:),'b');
            hold on;
            plot([observaciones.time],fstates(iy,:),'r-.');
            hold on;
            plot([observaciones.time],z(2,:),'g:');
            legend ('true y', 'filtered y', 'observed y');
            xlabel('Tiempo (s)');
            ylabel('Y (m)');
            hold off;

            legend('Real','IMM','Observaciones');
            drawnow
        end



        function [results firstFilteredObs] =filterObservations(this, observaciones)
            %% función que realiza el filtrado de todas las observaciones
            % con este filtro imm
            % se le pasa un vector de RadarObservacion y los datos resultantes de pasar
                el filtro imm
            %El primer estado es el observado, ya q no se puede filtrar,
            %el segundo se obtiene combinando las dos primeras
            %observaciones,y a partir del tercero se pasa el filtro
            % inicializamos los estimadores de estado y las covarianzas
            prmodo=this.probModo; %probabilidad de modo inicial
            pr_trans = this.matProbTrans; %mirarlo pq igual es fija o igual no!
            dims=this.stateDims;
            results= struct('vMM',zeros(dims,length(observaciones)-2),'vMU',prmodo');
            %obtenemos el estado inicial de entrada al filtro
            [x_est_mod P_cov_mod initialState]=this.initFilterStateData(observaciones(1)
                ,observaciones(2));
                firstFilteredObs=3;
            for i=firstFilteredObs:length(observaciones)
```

```
                %filtrar todas las observaciones a partir de la
                %segunda.
                resindex=i-2;
                intervT=observaciones(i).time-observaciones(i-1).time;
                %debería comprobar que el intervalo de tiempo es >0
                if (intervT<0)
                    error (['tiempo de la observación ' o 'anterior a observación ' (o
                        -1)]);
                    %                           elseif (intervT < 0.001) %a ignoro porque
                         ha pasado muy poco tiempo!
                    %                               disp 'intervalo muy pequeño'
                    %                               %continue
                end
                %% filtro IMM
                %obtener las matrices del sistema dinámico para cada modelo, para el
                %intervalo de tiempo.
                for m=1:length(this.Modelos)
                    %obtenemos las matrices del sistema
                    modelsMatrixes(m)= this.Modelos{m}.getModelMatrixes(intervT,
                        observaciones(i).errmeas);
                    indicesStateModels{m}=this.Modelos{m}.stateInds;
                end
                [x_est,P_cov, x_est_mod, P_cov_mod, prmod]= vImmFilter(observaciones(i).
                    measure, ...
                    x_est_mod, P_cov_mod, pr_trans,prmodo, modelsMatrixes,
                        indicesStateModels, dims );

                results.vMM(:,resindex)   = x_est;
                results.vPP(:,:,resindex) = P_cov;
                results.vMU(:,resindex)   = prmod;

            end
        end

        function [ RMSE_Master RMSE_Est ] = compareMasterObsWithFiltered( this,
            observaciones, estimaciones )
            %para cada intervalo (de 1 segudno) cogemos la observación del radar con
            %mayor prioridad, calculamos el error respecto a la trayectoria real y
            %luego lo comparamos con el resultado obtenido por el filtro de estimación
            obstimes=[observaciones.time];
            endT=ceil(obstimes(length(obstimes)));

            currSecond=0;
            indexMaster=[];%creamos el vector de índices de observaciones maestras
            for i=1:endT
                inds= (obstimes>(i-1))&(obstimes<=i);
                [val inds]= find(inds>0);
                %inds=obstimes(inds)<=i;
                if( isempty(inds))
                  % sprintf('no observations in interval (%d,%d)', i-1,i)
                    continue;
                end
                intervObs= observaciones(inds);
                intervEstims=estimaciones(:,inds);
                %buscar la observación con mayor prioridad, (es prioridad
                %min) y la guardamos
                [maxval indexmax]=min([intervObs.prioridad]);
                indexMaster=[indexMaster inds(indexmax)];

                trueState=intervObs(indexmax).truestate([this.stateVars.x; this.
                    stateVars.y]);
                errRadar=intervObs(indexmax).measure - trueState;
                errEstim=intervEstims([this.stateVars.x this.stateVars.y], indexmax) -
                    trueState;

                currSecond=currSecond+1;
                comp(currSecond).time=obstimes(inds(indexmax));
                comp(currSecond).errMaster=errRadar.^2;
                comp(currSecond).errEstim=errEstim.^2;

            end
            allMasterErr=[comp.errMaster];
            MSE_Master=mean(allMasterErr,2);
            RMSE_Master = 1/2*(sqrt(MSE_Master(1)) + sqrt(MSE_Master(2)));

            allEstErr=[comp.errEstim];
            MSE_Est=mean(allEstErr,2);
            RMSE_Est= 1/2*(sqrt(MSE_Est(1)) + sqrt(MSE_Est(2)));
            testIndexes=RadarObservacion.getIndexMasterObservation(observaciones);
        end
```

```
    end
273
    %Specific feautres of IMM design V. Vaidehi, K. Kalavidya and S. Indira Gandhi
    methods (Static)

        function objImm=ImmM1M4M5M6(kfmodels)
278         immstatedims=6;
            omega=0.2;
            if(nargin==0)
                %so no me pasan los 4 modelos los hago yo
                M1=ModelCV(immstatedims, 0.01);
283             M4=ModelCA(immstatedims, 2);
                M5=ModelCT(immstatedims, omega,0.01);
                M6=ModelCT(immstatedims, -omega,0.01);
                kfmodels={M1 M4 M5 M6};
            end
288         statevars.x=1;
            statevars.vx=2;
            statevars.y=4;
            statevars.vy=5;
            statevars.ax=3;
293         statevars.ay=6;

            matrizTransProb=[0.955 .015 .015 .015;
                .1 .85 .025 .025
                .15 .05 .75 .05
298             .15 .05 .05 .75];

            probsMod=[0.7 .1 .1 .1];%0.05];
            objImm=FilterImm(kfmodels,matrizTransProb, probsMod, statevars, immstatedims
                );
303     end

        function objImm=ImmM1M2M4()
            M1=ModelCV(6, 0.01);
            M2=ModelCV(6, 2);
308         M4=ModelCA(6, 1.5);

            immstatedims=6;
            kfmodels={M1 M2 M4};

313         statevars.x=1;
            statevars.vx=2;
            statevars.y=4;
            statevars.vy=5;
            statevars.ax=3;
318         statevars.ay=6;
            matrizTransProb=[0.97 0.015 .015;
                .15 .75 .10
                .1 .1 .80];
            probsMod=[0.8 .1 .1 ];
323
            objImm=FilterImm(kfmodels,matrizTransProb, probsMod, statevars, immstatedims
                );
        end

        function objImm=Imm1M1M3M4()
328         M1=ModelCV(6, 0.01);
            M3=ModelCA(6, 0.01);
            M4=ModelCA(6, 2);

            immstatedims=6;
333
            kfmodels={M1, M3, M4};
            statevars.x=1;
            statevars.vx=2;
            statevars.y=4;
338         statevars.vy=5;
            statevars.ax=3;
            statevars.ay=6;

            matrizTransProb=[0.97 0.015 .015;
343             .15 0.8 0.05
                .075 .075 0.85];

               matrizTransProb=[0.97 0.015 .015;
                .15 .70 .15
348             .15 .15 .70];
```

```
            probsMod=[0.8  .1  .1  ];
            objImm=FilterImm(kfmodels,matrizTransProb, probsMod, statevars, immstatedims
                );
        end

        function objImm=ImmM2M4()
            M2=ModelCV(6, 1);
            M4=ModelCA(6, 1);
            immstatedims=6;

            kfmodels={M2, M4};
            statevars.x=1;
            statevars.vx=2;
            statevars.y=4;
            statevars.vy=5;
            statevars.ax=3;
            statevars.ay=6;
            matrizTransProb=[0.95 0.05 ; 0.1  0.90];
            probsMod=[0.8  .2];
            objImm=FilterImm(kfmodels,matrizTransProb, probsMod, statevars, immstatedims
                );
        end

    end
end
```

A.6. Función que implementa un ciclo del Filtro Kalman

../code/filters/vKalmanFilt.m

```
%{
Función Filtro Kalman
% @author Vanessa Ibanez
%
% Sistema tiempo discreto es´tocástico
%         plant equation/Ec Dinámica:      x(k) = F(k-1)*x(k-1) +v(k-1)
%         medida,observación equation: z(k) = H(k)*x(k)      + w(k)
 Esta función implementa un ciclo del algoritmo de Kalman
%}

function [x P innova Sinn] = vKalmanFilter(x, P, z, model)
%Establecemos valores por defecto por si no nos pasan alguno
if ~isfield(model,'u'); model.u=0; end
if ~isfield(model,'F'); model.F=eye(length(x)); end
if ~isfield(model,'B'); model.B=0; end
if ~isfield(model,'Q'); model.Q=zeros(length(x)); end
if ~isfield(model,'R'); error('Observation covariance missing'); end
if ~isfield(model,'H'); model.H=eye(length(x)); end

    % Predicción del estado
    x = model.F*x + model.B*model.u;
    %prediccion de la matriz de covarianza
    P = model.F * P * model.F' + model.Q;
    %covarianza del residuo
    Sinn=model.H*P*model.H'+model.R;
    % Calculo del vector de ganancias
    K = P*model.H'/Sinn;
    %residuo (innovación)
    innova=z-model.H*x;
    % Correction based on observation:
    x = x + K*(innova);
    %actualización de la matriz de covarianza
    P = P - K*Sinn*K';
end
```

Apéndice B

Datos Simulación

En este apéndice se incluye la descripción completa de las trayectorias utilizadas en el proceso de simulación.

Trayectoria: CV. (1 tramos)					
Mov	Inicio	Durac (segs)	Vel(m/s)	Acel(m/s2)	Tasa Giro(rad)
CV	0.00	40.00	10.00	NaN	NaN

Trayectoria: StGo. (5 tramos)					
Mov	Inicio	Durac (segs)	Vel(m/s)	Acel(m/s2)	Tasa Giro(rad)
CV	0.00	20.00	10.00	NaN	NaN
CA	20.00	10.00	10.00	-1.00	NaN
CV	30.00	20.00	0.00	NaN	NaN
CA	50.00	10.00	0.00	1.00	NaN
CV	60.00	10.00	10.00	NaN	NaN

Trayectoria: CVCT(45). (3 tramos)					
Mov	Inicio	Durac (segs)	Vel(m/s)	Acel(m/s2)	Tasa Giro(rad)
CV	0.00	20.00	10.00	NaN	NaN
CT	20.00	5.24	NaN	NaN	0.15
CV	25.24	20.00	10.00	NaN	NaN

Trayectoria: CVCT(135). (3 tramos)					
Mov	Inicio	Durac (segs)	Vel(m/s)	Acel(m/s2)	Tasa Giro(rad)
CV	0.00	20.00	10.00	NaN	NaN
CT	20.00	11.78	NaN	NaN	-0.20
CV	31.78	30.00	10.00	NaN	NaN

Trayectoria: TakeOff. (4 tramos)					
Mov	Inicio	Durac (segs)	Vel(m/s)	Acel(m/s2)	Tasa Giro(rad)
CV	0.00	20.00	0.00	NaN	NaN
CA	20.00	10.00	0.00	0.50	NaN
CV	30.00	20.00	5.00	NaN	NaN
CA	50.00	8.75	5.00	4.00	NaN

Trayectoria: Landing. (5 tramos)					
Mov	Inicio	Durac (segs)	Vel(m/s)	Acel(m/s2)	Tasa Giro(rad)
CV	0.00	20.00	40.00	NaN	NaN
CA	20.00	15.00	40.00	-2.00	NaN
CV	35.00	20.00	10.00	NaN	NaN
CT	55.00	10.47	NaN	NaN	-0.10
CV	65.47	20.00	10.00	NaN	NaN

Trayectoria: DPZ1. (19 tramos)					
Mov	Inicio	Durac (segs)	Vel(m/s)	Acel(m/s2)	Tasa Giro(rad)
CA	0.00	5.00	0.00	1.00	NaN
CV	5.00	32.00	5.00	NaN	NaN
CT	37.00	1.96	NaN	NaN	-0.40
CV	38.96	20.00	5.00	NaN	NaN
CA	58.96	20.00	5.00	0.50	NaN
CV	78.96	6.67	15.00	NaN	NaN
CT	85.63	3.93	NaN	NaN	0.40
CV	89.56	86.67	15.00	NaN	NaN
CT	176.22	1.96	NaN	NaN	-0.40
CV	178.19	6.67	15.00	NaN	NaN
CT	184.85	3.93	NaN	NaN	-0.40
CV	188.78	100.00	15.00	NaN	NaN
CA	288.78	20.00	15.00	-0.50	NaN
CT	308.78	3.93	NaN	NaN	0.40
CV	312.71	140.00	5.00	NaN	NaN
CT	452.71	3.93	NaN	NaN	-0.40
CV	456.63	30.00	5.00	NaN	NaN
CT	486.63	1.96	NaN	NaN	0.40
CV	488.60	30.00	5.00	NaN	NaN

Trayectoria: Despegue 1 (24 tramos)					
Mov	Inicio	Durac (segs)	Vel(m/s)	Acel(m/s2)	Tasa Giro(rad)
CA	0.00	10.00	0.00	1.00	NaN
CV	10.00	10.00	10.00	NaN	NaN
CT	20.00	3.93	NaN	NaN	0.20
CV	23.93	40.00	10.00	NaN	NaN
CT	63.93	7.85	NaN	NaN	-0.20
CV	71.78	20.00	10.00	NaN	NaN
CT	91.78	4.19	NaN	NaN	-0.38
CV	95.97	140.00	10.00	NaN	NaN
CT	235.97	2.62	NaN	NaN	0.30
CV	238.59	10.00	10.00	NaN	NaN
CT	248.59	3.93	NaN	NaN	0.20
CV	252.51	50.00	10.00	NaN	NaN
CT	302.51	3.93	NaN	NaN	0.20
CV	306.44	10.00	10.00	NaN	NaN
CT	316.44	2.62	NaN	NaN	0.30
CV	319.06	70.00	10.00	NaN	NaN
CT	389.06	5.24	NaN	NaN	-0.30
CV	394.30	50.00	10.00	NaN	NaN
CT	444.30	5.24	NaN	NaN	0.30
CV	449.53	0.16	10.00	NaN	NaN
CA	449.69	10.00	10.00	-1.00	NaN
CV	459.69	10.00	0.00	NaN	NaN
CA	469.69	8.70	0.00	11.50	NaN
CV	478.38	20.00	100.00	NaN	NaN

Trayectoria: DPZ2. (16 tramos)					
Mov	Inicio	Durac (segs)	Vel(m/s)	Acel(m/s2)	Tasa Giro(rad)
CV	0.00	50.00	5.00	NaN	NaN
CT	50.00	3.93	NaN	NaN	0.40
CV	53.93	20.00	5.00	NaN	NaN
CT	73.93	2.62	NaN	NaN	-0.30
CV	76.54	30.00	5.00	NaN	NaN
CT	106.54	2.62	NaN	NaN	-0.30
CV	109.16	30.00	5.00	NaN	NaN
CT	139.16	3.93	NaN	NaN	0.40
CV	143.09	16.00	5.00	NaN	NaN
CA	159.09	66.67	5.00	0.15	NaN
CV	225.76	30.00	15.00	NaN	NaN
CT	255.76	1.96	NaN	NaN	-0.40
CV	257.72	16.67	15.00	NaN	NaN
CT	274.39	5.24	NaN	NaN	-0.30
CV	279.62	13.33	15.00	NaN	NaN
CA	292.96	15.00	15.00	-1.00	NaN

Trayectoria: DPZ3. (14 tramos)					
Mov	Inicio	Durac (segs)	Vel(m/s)	Acel(m/s2)	Tasa Giro(rad)
CA	0.00	10.00	0.00	1.00	NaN
CV	10.00	20.00	10.00	NaN	NaN
CT	30.00	1.96	NaN	NaN	-0.40
CV	31.96	35.00	10.00	NaN	NaN
CT	66.96	2.95	NaN	NaN	0.80
CV	69.91	30.00	10.00	NaN	NaN
CT	99.91	3.14	NaN	NaN	-0.50
CV	103.05	40.00	10.00	NaN	NaN
CT	143.05	3.14	NaN	NaN	-0.50
CV	146.19	5.00	10.00	NaN	NaN
CA	151.19	10.00	10.00	0.50	NaN
CV	161.19	53.33	15.00	NaN	NaN
CT	214.53	2.62	NaN	NaN	0.60
CV	217.14	13.33	15.00	NaN	NaN

Trayectoria: Aterr2. (13 tramos)					
Mov	Inicio	Durac (segs)	Vel(m/s)	Acel(m/s2)	Tasa Giro(rad)
CA	0.00	24.35	150.00	-5.75	NaN
CV	24.35	30.00	10.00	NaN	NaN
CT	54.35	7.85	NaN	NaN	-0.10
CV	62.20	30.00	10.00	NaN	NaN
CT	92.20	7.85	NaN	NaN	0.10
CV	100.06	150.00	10.00	NaN	NaN
CT	250.06	10.47	NaN	NaN	-0.15
CV	260.53	160.00	10.00	NaN	NaN
CT	420.53	11.78	NaN	NaN	0.20
CV	432.31	140.00	10.00	NaN	NaN
CT	572.31	11.78	NaN	NaN	-0.20
CV	584.09	20.00	10.00	NaN	NaN
CA	604.09	10.00	10.00	-1.00	NaN

Trayectoria: Despegue2. (16 tramos)					
Mov	Inicio	Durac (segs)	Vel(m/s)	Acel(m/s2)	Tasa Giro(rad)
CA	0.00	10.00	0.00	1.00	NaN
CV	10.00	20.00	10.00	NaN	NaN
CT	30.00	3.14	NaN	NaN	0.25
CV	33.14	140.00	10.00	NaN	NaN
CT	173.14	11.78	NaN	NaN	-0.20
CV	184.92	160.00	10.00	NaN	NaN
CT	344.92	7.85	NaN	NaN	0.20
CV	352.78	150.00	10.00	NaN	NaN
CT	502.78	3.93	NaN	NaN	-0.20
CV	506.70	30.00	10.00	NaN	NaN
CT	536.70	3.93	NaN	NaN	0.20
CV	540.63	5.00	10.00	NaN	NaN
CA	545.63	10.00	10.00	-1.00	NaN
CV	555.63	10.00	0.00	NaN	NaN
CA	565.63	17.65	0.00	1.70	NaN
CA	583.28	21.67	30.00	6.00	NaN

Trayectoria: Despegue3. (17 tramos)					
Mov	Inicio	Durac (segs)	Vel(m/s)	Acel(m/s2)	Tasa Giro(rad)
CA	0.00	10.00	0.00	1.00	NaN
CV	10.00	70.00	10.00	NaN	NaN
CT	80.00	2.95	NaN	NaN	0.80
CV	82.95	20.00	10.00	NaN	NaN
CT	102.95	3.93	NaN	NaN	-0.20
CV	106.87	80.00	10.00	NaN	NaN
CT	186.87	7.85	NaN	NaN	-0.20
CV	194.73	40.00	10.00	NaN	NaN
CT	234.73	3.93	NaN	NaN	-0.20
CV	238.65	40.00	10.00	NaN	NaN
CT	278.65	3.93	NaN	NaN	0.20
CV	282.58	20.00	10.00	NaN	NaN
CT	302.58	3.93	NaN	NaN	-0.20
CV	306.51	5.00	10.00	NaN	NaN
CA	311.51	10.00	10.00	-1.00	NaN
CV	321.51	20.00	0.00	NaN	NaN
CA	341.51	16.67	0.00	6.00	NaN

Trayectoria: Despegue4. (19 tramos)					
Mov	Inicio	Durac (segs)	Vel(m/s)	Acel(m/s2)	Tasa Giro(rad)
CA	0.00	10.00	0.00	1.00	NaN
CV	10.00	15.00	10.00	NaN	NaN
CT	25.00	3.93	NaN	NaN	0.20
CV	28.93	70.00	10.00	NaN	NaN
CT	98.93	3.93	NaN	NaN	0.20
CV	102.85	15.00	10.00	NaN	NaN
CT	117.85	7.85	NaN	NaN	-0.30
CV	125.71	120.00	10.00	NaN	NaN
CT	245.71	7.85	NaN	NaN	0.20
CV	253.56	30.00	10.00	NaN	NaN
CT	283.56	3.93	NaN	NaN	0.20
CV	287.49	15.00	10.00	NaN	NaN
CT	302.49	7.85	NaN	NaN	0.10
CV	310.34	20.00	10.00	NaN	NaN
CT	330.34	3.93	NaN	NaN	-0.20
CV	334.27	20.00	10.00	NaN	NaN
CA	354.27	10.00	10.00	-1.00	NaN
CA	364.27	11.67	0.00	6.00	NaN
CA	375.94	10.00	70.00	6.00	NaN

Trayectoria: Aterr1. (22 tramos)					
Mov	Inicio	Durac (segs)	Vel(m/s)	Acel(m/s2)	Tasa Giro(rad)
CV	0.00	10.00	120.00	NaN	NaN
CA	10.00	28.95	120.00	-3.80	NaN
CV	38.95	5.00	10.00	NaN	NaN
CT	43.95	5.24	NaN	NaN	-0.30
CV	49.18	50.00	10.00	NaN	NaN
CT	99.18	7.85	NaN	NaN	0.20
CV	107.04	70.00	10.00	NaN	NaN
CT	177.04	2.62	NaN	NaN	-0.30
CV	179.66	10.00	10.00	NaN	NaN
CT	189.66	5.24	NaN	NaN	-0.15
CV	194.89	50.00	10.00	NaN	NaN
CT	244.89	7.85	NaN	NaN	-0.10
CV	252.75	10.00	10.00	NaN	NaN
CT	262.75	7.85	NaN	NaN	-0.10
CV	270.60	140.00	10.00	NaN	NaN
CT	410.60	4.19	NaN	NaN	0.38
CV	414.79	20.00	10.00	NaN	NaN
CT	434.79	5.24	NaN	NaN	0.30
CV	440.02	40.00	10.00	NaN	NaN
CT	480.02	2.62	NaN	NaN	-0.30
CV	482.64	10.00	10.00	NaN	NaN
CA	492.64	10.00	10.00	-1.00	NaN

Apéndice C

Resultados Numéricos de los Experimentos

A continuación se presentan los datos relativos a los errores cuadráticos medios obtenidos en las 100 ejecuciones de cada experimento realizado. Para cada configuración de sensores, se incluyen los datos obtenidos con cada uno de los cuatro filtro IMM: el RMSE de los errores en las observaciones de los radares, el RMSE de los datos filtrados, el RMSE de los datos considerados maestros, y el porcentaje de mejora del RMSE (respecto a las medidas radar) que se obtiene al filtrar los datos.

Velocidad Constante.						
Sensores	RMSE Medidas	Filt IMM	RMSE Filtrado	RMSE Master	RMSE Est	Reducción Error(%)
	6.0490	1	3.9867	8.7090	5.7297	34.09
Conf.	6.0490	2	3.6223	8.7090	5.2224	40.12
Radar 1	6.0490	3	3.7651	8.7090	5.4250	37.76
	6.0490	4	3.8112	8.7090	5.4828	36.99
	5.5683	1	2.9098	7.1802	4.2802	47.74
Conf.	5.5683	2	2.7119	7.1802	3.9818	51.30
Radar 2	5.5683	3	2.8698	7.1802	4.2181	48.46
	5.5683	4	2.8491	7.1802	4.1948	48.83
	5.0319	1	2.1820	8.4349	3.3887	56.64
Conf.	5.0319	2	2.0343	8.4349	3.1677	59.57
Radar 3	5.0319	3	2.1840	8.4349	3.3964	56.60
	5.0319	4	2.1406	8.4349	3.3355	57.46

Stop And Go						
Sensores	RMSE Medidas	Filt IMM	RMSE Filtrado	RMSE Master	RMSE Est	Reducción Error(%)
Conf. Radar 1	4.1997	1	2.9711	6.2626	4.4553	29.25
	4.1997	2	2.9328	6.2626	4.4977	30.17
	4.1997	3	2.8982	6.2626	4.4064	30.99
	4.1997	4	2.9869	6.2626	4.5446	28.88
Conf. Radar 2	4.6571	1	2.6125	6.9081	3.8671	43.90
	4.6571	2	2.5695	6.9081	3.8338	44.83
	4.6571	3	2.5986	6.9081	3.8497	44.20
	4.6571	4	2.6120	6.9081	3.8850	43.91
Conf. Radar 3	4.7553	1	2.3321	6.9279	3.4429	50.96
	4.7553	2	2.3125	6.9279	3.4158	51.37
	4.7553	3	2.3456	6.9279	3.4639	50.67
	4.7553	4	2.3566	6.9279	3.4664	50.44

Stop And Go						
Sensores	RMSE Medidas	Filt IMM	RMSE Filtrado	RMSE Master	RMSE Est	Reducción Error(%)
Conf. Radar 1	5.2153	1	3.7794	7.4554	5.4501	27.53
	5.2153	2	3.5759	7.4554	5.1876	31.43
	5.2153	3	3.6337	7.4554	5.2563	30.33
	5.2153	4	3.6094	7.4554	5.2282	30.79
Conf. Radar 2	5.0909	1	3.1781	7.0043	4.6228	37.57
	5.0909	2	3.0891	7.0043	4.4885	39.32
	5.0909	3	3.1292	7.0043	4.5505	38.53
	5.0909	4	3.1222	7.0043	4.5328	38.67
Conf. Radar 3	5.6156	1	3.1710	9.0355	4.6286	43.53
	5.6156	2	3.1265	9.0355	4.5407	44.32
	5.6156	3	3.1347	9.0355	4.5883	44.18
	5.6156	4	3.1496	9.0355	4.5656	43.91

Maniobra de Aterrizaje						
Sensores	RMSE Medidas	Filt IMM	RMSE Filtrado	RMSE Master	RMSE Est	Reducción Error(%)
Conf. Radar 1	3.9685	1	2.9679	5.9955	4.4911	25.21
	3.9685	2	3.0264	5.9955	4.5628	23.74
	3.9685	3	2.9511	5.9955	4.4529	25.64
	3.9685	4	3.1220	5.9955	4.6618	21.33
Conf. Radar 2	4.5963	1	2.6532	6.9319	3.8453	42.28
	4.5963	2	2.6811	6.9319	3.8957	41.67
	4.5963	3	2.6264	6.9319	3.7933	42.86
	4.5963	4	2.7767	6.9319	3.9818	39.59
Conf. Radar 3	5.2361	1	2.5737	7.2386	3.6056	50.85
	5.2361	2	2.5904	7.2386	3.6490	50.53
	5.2361	3	2.5524	7.2386	3.5628	51.25
	5.2361	4	2.6755	7.2386	3.7233	48.90

Maniobra con Giro de 45 grados						
Sensores	RMSE Medidas	Filt IMM	RMSE Filtrado	RMSE Master	RMSE Est	Reducción Error(%)
	3.3634	1	2.5739	5.8744	4.3599	23.47
Conf.	3.3634	2	2.6146	5.8744	4.3158	22.26
Radar 1	3.3634	3	2.5586	5.8744	4.2957	23.93
	3.3634	4	2.5272	5.8744	4.3120	24.86
	4.4862	1	2.5623	6.8949	3.8411	42.89
Conf.	4.4862	2	2.4922	6.8949	3.7310	44.45
Radar 2	4.4862	3	2.5329	6.8949	3.7937	43.54
	4.4862	4	2.5069	6.8949	3.7530	44.12
	5.0334	1	2.3909	8.1969	3.5583	52.50
Conf.	5.0334	2	2.3265	8.1969	3.4557	53.78
Radar 3	5.0334	3	2.3964	8.1969	3.5648	52.39
	5.0334	4	2.3526	8.1969	3.5051	53.26

Maniobra con Giro de 135 grados						
Sensores	RMSE Medidas	Filt IMM	RMSE Filtrado	RMSE Master	RMSE Est	Reducción Error(%)
	4.1702	1	3.1654	6.9167	5.0455	24.10
Conf.	4.1702	2	3.2311	6.9167	4.9754	22.52
Radar 1	4.1702	3	3.2081	6.9167	4.9943	23.07
	4.1702	4	3.0805	6.9167	4.8491	26.13
	4.8117	1	2.9928	7.0593	4.6318	37.80
Conf.	4.8117	2	2.9853	7.0593	4.6259	37.96
Radar 2	4.8117	3	2.9572	7.0593	4.5731	38.54
	4.8117	4	2.9143	7.0593	4.5107	39.43
	5.0915	1	2.7243	6.8424	4.2913	46.49
Conf.	5.0915	2	2.7209	6.8424	4.2912	46.56
Radar 3	5.0915	3	2.7021	6.8424	4.2508	46.93
	5.0915	4	2.6644	6.8424	4.1938	47.67

Desplazamiento por Aeropuerto 1						
Sensores	RMSE Medidas	Filt IMM	RMSE Filtrado	RMSE Master	RMSE Est	Reducción Error(%)
	4.3685	1	3.1215	6.2125	4.4288	28.55
Conf.	4.3685	2	2.9264	6.2125	4.1499	33.01
Radar 1	4.3685	3	3.1685	6.2125	4.4877	27.47
	4.3685	4	3.0024	6.2125	4.2611	31.27
	4.6919	1	2.6669	6.9954	3.8458	43.16
Conf.	4.6919	2	2.5413	6.9954	3.6489	45.84
Radar 2	4.6919	3	2.8176	6.9954	4.0381	39.95
	4.6919	4	2.5997	6.9954	3.7302	44.59
	4.9206	1	2.4174	7.0780	3.5303	50.87
Conf.	4.9206	2	2.3161	7.0780	3.3726	52.93
Radar 3	4.9206	3	2.5550	7.0780	3.7065	48.08
	4.9206	4	2.3678	7.0780	3.4400	51.88

Desplazamiento por Aeropuerto 2						
Sensores	RMSE Medidas	Filt IMM	RMSE Filtrado	RMSE Master	RMSE Est	Reducción Error(%)
Conf. Radar 1	4.5752	1	3.1996	6.8693	4.7506	30.07
	4.5752	2	3.0565	6.8693	4.4695	33.19
	4.5752	3	3.0855	6.8693	4.5505	32.56
	4.5752	4	3.0794	6.8693	4.5313	32.69
Conf. Radar 2	4.8711	1	2.7595	7.0625	4.1611	43.35
	4.8711	2	2.6350	7.0625	3.9517	45.90
	4.8711	3	2.7088	7.0625	4.0756	44.39
	4.8711	4	2.6613	7.0625	4.0016	45.37
Conf. Radar 3	4.7925	1	2.4168	6.6132	3.5043	49.57
	4.7925	2	2.3344	6.6132	3.3632	51.29
	4.7925	3	2.4038	6.6132	3.4824	49.84
	4.7925	4	2.3495	6.6132	3.3923	50.97

Desplazamiento por Aeropuerto 3						
Sensores	RMSE Medidas	Filt IMM	RMSE Filtrado	RMSE Master	RMSE Est	Reducción Error(%)
Conf. Radar 1	5.7540	1	4.0227	8.8153	6.0733	30.09
	5.7540	2	3.8959	8.8153	5.8495	32.29
	5.7540	3	3.9090	8.8153	5.8955	32.06
	5.7540	4	3.9576	8.8153	5.9265	31.22
Conf. Radar 2	5.5208	1	3.0923	7.3427	4.1217	43.99
	5.5208	2	3.0126	7.3427	4.0167	45.43
	5.5208	3	3.0379	7.3427	4.0569	44.97
	5.5208	4	3.0495	7.3427	4.0607	44.76
Conf. Radar 3	5.0389	1	2.4523	5.7342	3.5141	51.33
	5.0389	2	2.4050	5.7342	3.4430	52.27
	5.0389	3	2.4510	5.7342	3.5064	51.36
	5.0389	4	2.4495	5.7342	3.5073	51.39

Despegue en Pista 1 partiendo desde la Teminal						
Sensores	RMSE Medidas	Filt IMM	RMSE Filtrado	RMSE Master	RMSE Est	Reducción Error(%)
Conf. Radar 1	5.4031	1	3.6936	7.7416	5.2923	31.64
	5.4031	2	3.5433	7.7416	5.0511	34.42
	5.4031	3	3.6985	7.7416	5.2801	31.55
	5.4031	4	3.6009	7.7416	5.1213	33.35
Conf. Radar 2	5.2233	1	2.9338	7.2618	4.1581	43.83
	5.2233	2	2.8565	7.2618	4.0396	45.31
	5.2233	3	3.0412	7.2618	4.3375	41.78
	5.2233	4	2.9222	7.2618	4.1124	44.05
Conf. Radar 3	5.6023	1	2.8175	7.5781	3.8682	49.71
	5.6023	2	2.7622	7.5781	3.7911	50.70
	5.6023	3	2.8673	7.5781	3.9483	48.82
	5.6023	4	2.8297	7.5781	3.8572	49.49

Despegue en Pista 2 partiendo desde la Teminal						
Sensores	RMSE Medidas	Filt IMM	RMSE Filtrado	RMSE Master	RMSE Est	Reducción Error(%)
	4.5791	1	3.2096	6.6177	4.6476	29.91
Conf.	4.5791	2	3.0765	6.6177	4.4205	32.81
Radar 1	4.5791	3	3.2744	6.6177	4.6989	28.49
	4.5791	4	3.1748	6.6177	4.5658	30.67
	4.8282	1	2.7463	7.0959	3.9893	43.12
Conf.	4.8282	2	2.6599	7.0959	3.8386	44.91
Radar 2	4.8282	3	2.8134	7.0959	4.0730	41.73
	4.8282	4	2.7371	7.0959	3.9530	43.31
	5.0639	1	2.4378	7.8746	3.3434	51.86
Conf.	5.0639	2	2.4104	7.8746	3.2839	52.40
Radar 3	5.0639	3	2.4665	7.8746	3.4238	51.29
	5.0639	4	2.4601	7.8746	3.3618	51.42

Despegue en Pista 3 partiendo desde la Teminal						
Sensores	RMSE Medidas	Filt IMM	RMSE Filtrado	RMSE Master	RMSE Est	Reducción Error(%)
	3.9514	1	3.0197	5.6918	4.3422	23.58
Conf.	3.9514	2	2.9596	5.6918	4.2408	25.10
Radar 1	3.9514	3	2.9791	5.6918	4.2929	24.61
	3.9514	4	2.9931	5.6918	4.2966	24.25
	4.5193	1	2.8070	6.8979	4.1320	37.89
Conf.	4.5193	2	2.7604	6.8979	4.0330	38.92
Radar 2	4.5193	3	2.8506	6.8979	4.1928	36.92
	4.5193	4	2.7891	6.8979	4.0628	38.29
	5.0263	1	2.7278	6.2148	3.7438	45.73
Conf.	5.0263	2	2.7313	6.2148	3.7064	45.66
Radar 3	5.0263	3	2.6969	6.2148	3.7115	46.34
	5.0263	4	2.7413	6.2148	3.7114	45.46

Despegue en Pista 4 partiendo desde la Teminal						
Sensores	RMSE Medidas	Filt IMM	RMSE Filtrado	RMSE Master	RMSE Est	Reducción Error(%)
	4.8011	1	3.5308	6.8467	4.9770	26.46
Conf.	4.8011	2	3.4643	6.8467	4.8264	27.84
Radar 1	4.8011	3	3.4991	6.8467	4.9559	27.12
	4.8011	4	3.5142	6.8467	4.9115	26.81
	4.9289	1	2.9027	7.1188	4.2110	41.11
Conf.	4.9289	2	2.8693	7.1188	4.1149	41.79
Radar 2	4.9289	3	2.8976	7.1188	4.2613	41.21
	4.9289	4	2.9056	7.1188	4.1673	41.05
	4.8542	1	2.4264	7.3381	3.6944	50.01
Conf.	4.8542	2	2.4123	7.3381	3.6687	50.31
Radar 3	4.8542	3	2.4060	7.3381	3.6764	50.43
	4.8542	4	2.4388	7.3381	3.7056	49.76

Aterrizaje en Pista 1 con llegada a la Teminal						
Sensores	RMSE Medidas	Filt IMM	RMSE Filtrado	RMSE Master	RMSE Est	Reducción Error(%)
Conf. Radar 1	5.6301	1	3.7917	7.9645	5.3717	32.65
	5.6301	2	3.6285	7.9645	5.1317	35.55
	5.6301	3	4.0033	7.9645	5.6553	28.89
	5.6301	4	3.7441	7.9645	5.2555	33.50
Conf. Radar 2	5.3179	1	2.9684	7.3661	4.1923	44.18
	5.3179	2	2.8995	7.3661	4.0901	45.48
	5.3179	3	3.0886	7.3661	4.3803	41.92
	5.3179	4	2.9823	7.3661	4.1625	43.92
Conf. Radar 3	5.7629	1	3.0302	8.6054	3.9686	47.42
	5.7629	2	2.9709	8.6054	3.8728	48.45
	5.7629	3	3.0710	8.6054	4.0484	46.71
	5.7629	4	3.0338	8.6054	3.9325	47.36

Aterrizaje en Pista 2 con llegada a la Teminal						
Sensores	RMSE Medidas	Filt IMM	RMSE Filtrado	RMSE Master	RMSE Est	Reducción Error(%)
Conf. Radar 1	4.7996	1	3.4378	6.9097	4.9584	28.37
	4.7996	2	3.3120	6.9097	4.7223	30.99
	4.7996	3	3.5825	6.9097	5.1915	25.36
	4.7996	4	3.3783	6.9097	4.8027	29.61
Conf. Radar 2	4.9223	1	2.8402	7.1155	3.9152	42.30
	4.9223	2	2.7651	7.1155	3.7933	43.82
	4.9223	3	3.0808	7.1155	4.2295	37.41
	4.9223	4	2.8223	7.1155	3.8579	42.66
Conf. Radar 3	5.0827	1	2.4230	6.9475	3.3367	52.33
	5.0827	2	2.3786	6.9475	3.2479	53.20
	5.0827	3	2.5518	6.9475	3.4855	49.80
	5.0827	4	2.4053	6.9475	3.2863	52.68

Apéndice D

Datos Reales de Sistemas de Vigilancia en Aeropuertos

Cuadro D.1: Datos de 14mins de seguimiento en el aeropuerto de Palma organizados por pistas TDVT

Tiempo(s)	TDVT	Plots	Radares	Aports	Int Vels.	Int Acels.
$[3, 912]$	346	912	SMR	911	$[0, 2]$	$[-1, 1]$
$[3, 178]$	179	157	SMR	157	$[0, 3]$	$[-3, 2]$
$[3, 912]$	424	912	SMR	911	$[0, 3]$	$[-2, 2]$
$[4, 912]$	204	912	SMR	911	$[0, 4]$	$[-3, 3]$
$[4, 912]$	154	912	SMR	911	$[0, 12]$	$[-12, 12]$
$[4, 95]$	219	78	SMR	78	$[0, 5]$	$[-3, 2]$
$[4, 536]$	172	535	SMR	535	$[0, 16]$	$[-5, 12]$
$[4, 912]$	68	912	SMR	911	$[0, 3]$	$[-3, 3]$
$[4, 912]$	433	912	SMR	911	$[0, 7]$	$[-7, 6]$
$[4, 912]$	284	901	SMR	900	$[0, 10]$	$[-10, 4]$
$[4, 366]$	361	334	SMR	334	$[0, 10]$	$[-10, 5]$
$[4, 912]$	30	911	SMR	910	$[0, 6]$	$[-5, 6]$
$[4, 912]$	128	911	SMR	910	$[0, 3]$	$[-2, 3]$
$[4, 912]$	48	911	SMR	910	$[0, 2]$	$[-2, 2]$
$[4, 912]$	365	911	SMR	910	$[0, 2]$	$[-2, 2]$
$[5, 911]$	138	178	TDVM	177	$[191, 242]$	$[0, 0]$
$[5, 230]$	476	45	TDVM	45	$[182, 188]$	$[-1, 1]$
$[5, 287]$	73	56	TDVM	56	$[201, 208]$	$[-1, 1]$
$[5, 261]$	192	51	TDVM	51	$[199, 230]$	$[-1, 0]$
$[5, 225]$	291	44	TDVM	44	$[229, 231]$	$[0, 0]$
$[5, 476]$	474	93	TDVM	93	$[215, 220]$	$[-1, 1]$
$[5, 553]$	72	108	TDVM	108	$[247, 252]$	$[0, 0]$
$[5, 845]$	426	165	TDVM	165	$[205, 215]$	$[0, 1]$
$[5, 742]$	37	145	TDVM	145	$[119, 254]$	$[-1, 0]$
$[5, 911]$	305	178	TDVM	177	$[139, 202]$	$[-1, 1]$
$[5, 911]$	269	178	TDVM	177	$[216, 219]$	$[0, 0]$
$[5, 195]$	124	38	TDVM	38	$[229, 238]$	$[0, 2]$
$[5, 456]$	267	89	TDVM	89	$[123, 206]$	$[-1, 0]$
$[5, 911]$	200	178	TDVM	177	$[190, 259]$	$[-1, 1]$
$[5, 292]$	277	57	TDVM	57	$[193, 222]$	$[-1, 1]$

Time	TDVT	Plots	Radares	Aports	Int Vels.	Int Acels.
$[5, 912]$	393	354	TDVM, MMS, SMR, SMR	555	$[0, 194]$	$[-20, 3]$
$[5, 912]$	452	241	TDVM, MMS, SMR	340	$[14, 228]$	$[-28, 6]$
$[5, 584]$	429	114	TDVM	114	$[248, 252]$	$[-1, 0]$
$[5, 911]$	38	178	TDVM	177	$[209, 215]$	$[0, 0]$
$[5, 712]$	97	139	TDVM	139	$[264, 271]$	$[0, 0]$
$[5, 230]$	58	45	TDVM	45	$[50, 135]$	$[-3, 1]$
$[5, 911]$	385	178	TDVM	177	$[107, 192]$	$[-15, 10]$
$[41, 911]$	317	171	TDVM	170	$[164, 217]$	$[0, 1]$
$[67, 911]$	439	166	TDVM	165	$[107, 228]$	$[-1, 2]$
$[125, 192]$	372	59	SMR	59	$[0, 7]$	$[-6, 2]$
$[155, 205]$	283	37	SMR	37	$[0, 5]$	$[-4, 1]$
$[199, 275]$	490	61	SMR	61	$[0, 3]$	$[-3, 2]$
$[217, 345]$	369	122	SMR	122	$[0, 2]$	$[-2, 2]$
$[266, 911]$	34	127	TDVM	126	$[228, 298]$	$[-8, 1]$
$[276, 911]$	320	125	TDVM	124	$[119, 212]$	$[0, 3]$
$[304, 374]$	323	62	SMR	62	$[0, 4]$	$[-4, 2]$
$[305, 359]$	391	37	SMR	37	$[0, 5]$	$[-5, 1]$
$[374, 414]$	263	39	SMR	39	$[0, 2]$	$[-2, 1]$
$[394, 911]$	352	102	TDVM	101	$[121, 214]$	$[0, 2]$
$[410, 465]$	61	52	SMR	52	$[0, 4]$	$[-2, 2]$
$[423, 460]$	331	35	SMR	35	$[0, 4]$	$[-3, 3]$
$[425, 621]$	70	182	SMR	182	$[0, 2]$	$[-2, 2]$
$[430, 911]$	170	95	TDVM	94	$[166, 244]$	$[-1, 2]$
$[464, 902]$	318	397	SMR	397	$[0, 13]$	$[-12, 5]$
$[466, 911]$	423	88	TDVM	88	$[191, 223]$	$[0, 1]$
$[471, 911]$	89	87	TDVM	86	$[126, 195]$	$[0, 2]$
$[490, 537]$	178	33	SMR	33	$[0, 3]$	$[-3, 1]$
$[497, 535]$	436	39	SMR	39	$[9, 24]$	$[-3, 3]$
$[540, 912]$	85	374	SMR	373	$[0, 5]$	$[-4, 4]$
$[568, 911]$	403	68	TDVM	67	$[74, 201]$	$[0, 13]$
$[614, 911]$	199	59	TDVM	58	$[119, 185]$	$[-2, 7]$
$[616, 687]$	408	63	SMR	63	$[0, 6]$	$[-6, 2]$
$[625, 769]$	370	132	SMR	132	$[0, 3]$	$[-3, 1]$
$[666, 911]$	373	49	TDVM	48	$[198, 227]$	$[-5, 0]$
$[686, 911]$	152	45	TDVM	44	$[246, 261]$	$[-1, 3]$
$[696, 736]$	15	38	SMR	38	$[0, 7]$	$[-7, 1]$
$[745, 786]$	333	32	SMR	32	$[0, 2]$	$[-2, 2]$
$[773, 912]$	400	123	SMR	122	$[0, 4]$	$[-4, 1]$
$[860, 912]$	90	48	SMR	47	$[0, 3]$	$[-3, 2]$
$[861, 912]$	75	40	SMR	39	$[0, 2]$	$[-1, 1]$