UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

TRACKING ROBUSTO DE ROBOTS USANDO RANDOM FINITE SETS

TESIS PARA OPTAR AL GRADO DE MAGISTER EN CIENCIAS DE LA INGENIERÍA MENCIÓN ELÉCTRICA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

PABLO IGNACIO CANO MONTECINOS

PROFESOR GUÍA:
JAVIER RUÍZ DEL SOLAR SAN MARTÍN

MIEMBROS DE LA COMISIÓN:
MARTIN ADAMS
FERNANDO AUAT CHEEIN

Este trabajo ha sido parcialmente financiado por Proyecto Fondecyt 1161500.

SANTIAGO DE CHILE
2018

RESUMEN DE TESIS PARA OPTAR AL GRADO DE
MAGISTER EN CIENCIAS DE LA INGENIERÍA MENCIÓN ELÉCTRICAY AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: PABLO IGNACIO CANO MONTECINOS
FECHA: 2018
PROF. GUÍA: SR. JAVIER RUÍZ DEL SOLAR SAN MARTÍN

TRACKING ROBUSTO DE ROBOTS USANDO RANDOM FINITE SETS

Esta tesis se enfoca en resolver el problema del multi-target *tracking* en ambientes altamente dinámicos, y utilizando robots que poseen baja capacidad de procesamiento y sensores limitados. Para esto se utiliza un nuevo método de realizar *tracking* baso en Random Finite Sets (RFS). La utilización de este método supone diferentes mejoras como la eliminación del *data association problem* o la utilización de la información negativa de los sensores. La hipótesis que se desea probar es que la utilización de este nuevo método puede obtener mejores resultados que los clásicos métodos de *tracking*, como el EKF multi-hipótesis. Además se desea demostrar que es posible realizar un *tracking* de este estilo en robots de poco procesamiento computacional y que este resultado se puede mejorar aún más si un conjunto de robots comparten sus estimaciones para generar una estimación global.

Para esto se utiliza el robot Nao, el cual es utilizado en la competencia RoboCup, la cual corresponde a una competencia de fútbol robótico. Es en este escenario donde se implementa un método de *tracking*, basado en RFS, que se utiliza para ubicar todos los robots de la cancha, con tal de realizar un mapa de obstáculos. La implementación de este método la caracterización de cada sensor del robot, los cuales son utilizados como inputs del sistema. Finalmente, se realizan diversas pruebas, ya sea con robots reales o con simulaciones realistas, la cuales constan de escenarios ficticios y de partidos reales. En cada una de estas, se compara el mapa de obstáculos obtenido con las posiciones reales de los robots con la cancha. Esta comparación se realiza utilizando una medida de distancia especialmente diseñada para comparar conjuntos, llamada OSPA.

Los resultados obtenidos demuestran que el método propuesto en esta tesis supera en general a los métodos clásicos, ya sea tanto cualitativa como cuantitativamente. Se pueden observar claramente las ventajas de la utilización negativa de los sensores, así como la no necesidad de resolver el *data association problem*. También se puede observar la mejora generada al utilizar la información compartida entre robots, lo que genera un mapa de obstáculos más preciso.

Para toda mi gran familia

Agradecimientos

Agradezco a mis padres Rodrigo y Sofía por brindarme todo el apoyo necesario para lograr mis objetivos en la vida y por ser un gran modelo a seguir. A mis hermanos Mauricio y Pedro por acompañarme siempre y enseñarme a ser una mejor persona.

Agradezco también a toda mi familia, en especial a Lorena y Gonzalo, por aceptarme en su casa en mis primeros meses en la universidad.

Agradezco a los amigos que hice durante mis primeros años en la universidad, José Miguel Alvarado, Gerardo Rojas, Catalina Faune, Diego Carvajal, Nicole Salas y Javiera Guevara, por hacer de la universidad un lugar entretenido y acogedor.

Agradezco a los amigos del departamento, principalmente a Matías Mattamala e Ignacio Barrueto, sin quienes hubiera sido imposible terminar esta carrera.

Agradezco a Pablo Guerrero por aceptarme en el Laboratorio de Robotica, lugar donde mejor me desarrolle como estudiante y donde pude conocer a muchas grandes personas. Agradezco especialmente a José Miguel Yañez por permitirme ser parte del proceso de cambio del laboratorio y por todas las enseñanzas entregadas.

Agradezco a todos los amigos de hice durante mi estadía en el laboratorio como Gonzalo Olave, Gabriel Azocar por todos los buenos momentos vividos.

Agradezco a Constanza Villegas y a nuestra gatita Kida por toda la felicidad que traen a mi vida.

Agradezco finalmente a mi profesor guía Javier Ruiz del Solar y a los miembros de la comisión Martin Adams y Fernando Auat por todos sus comentarios y revisiones.

Tabla de contenido

Índice de tablas

Índice de ilustraciones

Capítulo 1

Introducción

1.1. Motivación

1.1.1. Fundamentación General

La teoría del filtrado estocástico es una de las áreas más estudiadas y desarrolladas en el mundo científico, con grandes aplicaciones en diversas áreas como la comunicaciones, aprendizaje de máquinas, neurociencia, economía, finanzas, ciencias políticas, entre otras. Actualmente, una de sus áreas de aplicación, y que también tiene gran repercusión práctica, es el problema del seguimiento o *tracking*. Algunas aplicaciones conocidas, y de uso común, son los programas de seguimiento de caras, presentes es la mayoría de *smartphones* y cámaras digitales, que permiten adecuar una fotografía de manera de mejorar la visibilidad de las caras de las personas. A pesar de existen múltiples soluciones para el problema de *tracking* que alcanzan grandes tasas de rendimiento, esto no se replica en el problema del *tracking* múltiple. Este problema hace alusión a realizar *tracking* a varios elementos de manera simultánea, y donde los elementos son indistinguibles entre sí. Para resolver el problema del *tracking* múltiple, se han realizado diferentes adaptaciones de soluciones de *tracking* simple, los cuales resuelven un problema de *tracking* por cada elemento presente. Sin embargo, esta forma de resolverlo implica solucionar el problema de asociación de datos o *data association*, el cual hace referencia a la dificultad de asociar cada medición (la cual se obtiene de todos los elementos indistinguibles) con alguno de los *tracking* simultáneos que se están realizando. Como se muestra más adelante, existen variadas formas de abordar este problema. Por esto que surge un nuevo método de realizar *tracking* múltiple basado en *Random Finite Sets* (RFS), en el cual se evita por completo el problema del *data association*. Para esto, el principal cambio con respecto a los métodos clásicos se centra en el uso de conjuntos o *sets*, en vez de vectores. Desde la aparición de los RFS, que han surgido variados métodos de *tracking* basados en esta técnica, los cuales han tenido mejores resultados para los mismos problemas resueltos con métodos anteriores, razón por la cual su uso se ha extendido a otros tipos de problemas como el SLAM. Sin embargo su uso en robots en movimiento, y realizando *tracking* de objetos en movimiento es limitado, y solo existen algunos trabajos, como los que se pueden observar en [1, 2]. Pero, estos trabajos cuentan con computadores con grandes capacidades de procesamiento, por lo que no se puede utilizar en sistemas pequeños, como

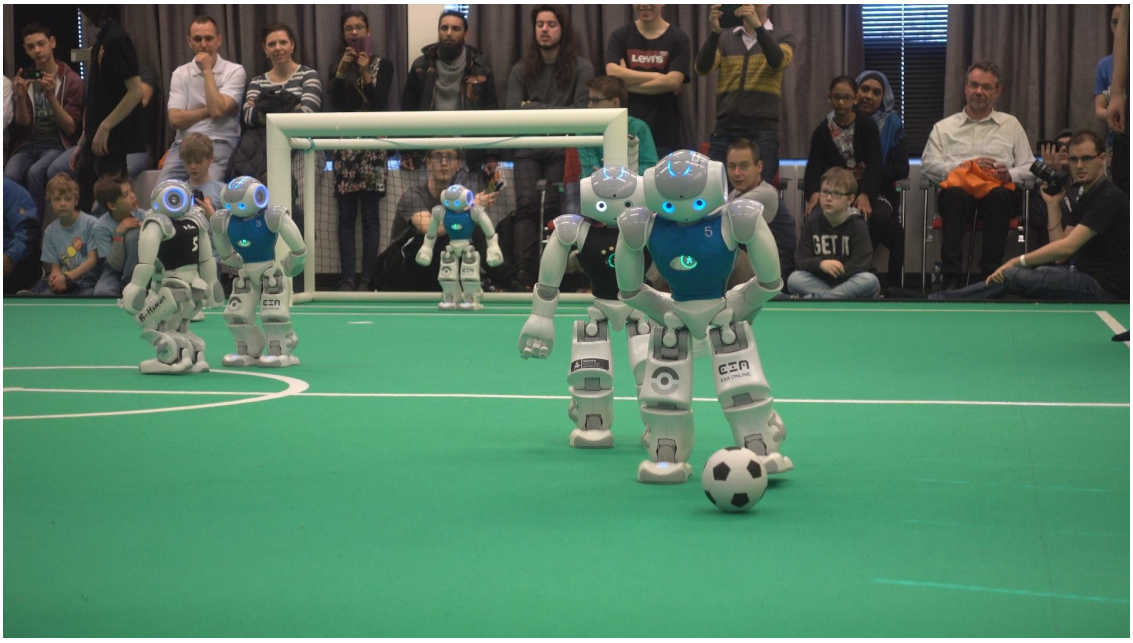


Figura 1.1: Robots Nao jugando fútbol durante la competencia RoboCup 2016

computadores embebidos. Es aquí donde se centra el desarrollo de esta tesis, donde se eligió el futból robótico como caso de estudio y el robot Nao como hardware de desarrollo.

Fútbol Robótico

El fútbol róbotico es un aplicación en la cual se utilizan robots que son capaces de "jugar" fútbol de manera autónoma, es decir, sin que un humano los controle. Esta aplicación fue creada por la Federación Internacional RoboCup[3], la cual es la encargada de realizar una competencia anual, conocida como el mundial de "fútbol robótico". Actualmente, existen diferentes ligas dentro de la RoboCup, de las cuales no todas corresponden a competencias de fútbol, y que buscan mejorar variadas áreas de la robótica. Una de las ligas relacionadas con el fútbol robótico es la liga estándar (SPL), donde todos los equipos utilizan el mismo robot (Nao, de la empresa SoftBank Robotics) para competir, lo que la hace principalmente una competencia de software.

En esta competencia, dos equipos conformados por cinco robots Nao, se enfrentan en un partido de fútbol, con reglas acordes a las capacidad de los robots. Como cada robot debe tomar decisiones de manera autónoma, cada uno de ellos debe realizar el proceso completo de:

1. Adquirir información del ambiente.
2. Procesar y entender esta información, de manera de entender su entorno.
3. Tomar decisiones con respecto a su entendimiento de su entorno.

En este proceso se encuentra el problema planteado por esta tesis. Dentro de las muchas cosas que el robot debe sensar y decidir, una información importante consiste en saber donde se encuentran sus compañeros y oponentes, de manera de tomar la mejor decisión posible. Para saber donde se encuentran los otros robots en la cancha es donde se utiliza el *tracking*

mencionado. Cabe destacar que en este escenario, todos los robots están en constante movimiento, por lo que no solo los objetos a los cuales se les realiza *tracking* se mueven en el escenario, sino que también el robot que esta realizando el *tracking* se mueve por el campo. Además, los robots pueden ser sacados de la cancha por los árbitros humanos si infringen las reglas del juego, lo que hace variable el número de robots que se encuentran en ésta. Por estas razones es que se hace referencia a este problema como uno dinámico.

Cabe mencionar que el Departamento de Ingeniería Eléctrica (DIE) de la Universidad de Chile, en conjunto con el Centro Avanzado de Tecnología para la Minería(AMTC) cuentan con un equipo de fútbol robótico (UChile Robotics Team), donde se desarrolló esta tesis, y el cual participa de la competencia internacional RoboCup desde el año 2001, alcanzando el cuarto puesto durante los años 2014, 2015 y 2016. Esto significa que este trabajo de tesis pudo ser utilizado en un ambiente competitivo, obteniéndose resultados favorables.

Robot Nao

El robot utilizado para desarrollar esta tesis es el robot Nao, creado por la empresa SoftBank Robotics (ex Aldebaran). Este robot humanoide cuenta con variados sensores, como se muestra en la Figura 1.2, los cuales le permiten interactuar con su entorno. Entre ellos se encuentran sonares, detectores infrarrojos, detectores de tacto y dos cámaras HD no estéreo. Este robot cuenta con un computador ATOM Z530 de 1.6 GHz de velocidad de CPU, lo que es una muy baja capacidad de procesamiento para realizar tareas pesadas. Aún así, el robot es capaz de procesar todas las imágenes de las cámaras, las cuales funcionan a 30Hz, modelar su entorno y tomar decisiones. Es por esto que la solución utilizada para la realización del *tracking* no debe ser compleja. De hecho, la ventana de procesamiento que se tiene es de $\sim$ 5[ms], con lo cual no se interfiere con las tareas que le permiten al robot jugar fútbol adecuadamente.

Simulador realista

Otra de las herramientas utilizadas para esta tesis consiste en un simulador realista de robots Naos y de la dinámica del juego de la Robocup. Este programa llamado SimRobot[4] fue creado por el equipo alemán B-Human.

Este software es capaz de simular las cámaras de los robots, generando imágenes que pueden ser analizadas de la misma manera que si fueran imágenes reales, lo que permite probar distintos algoritmos de manera mucho más rápida que en un partido real. Además, puede simular la física de los robots, por lo que los choques entre ellos, o con la pelota tienen un impacto en el simulación. En la Figura 1.3 se puede observar una visualización del software, donde a la izquierda se encuentran las imágenes entregadas por las cámaras simuladas, y a la derecha el escenario simulado completo.

1.1.2. Definición del Problema

El problema a resolver consiste en que cada robot realiza un *tracking* de todos los otros robots en la cancha, ya sean compañeros u oponentes, utilizando sus diferentes sensores, con el objetivo de crear un **mapa local** de obstáculos *online*. También, los robots de un mismo

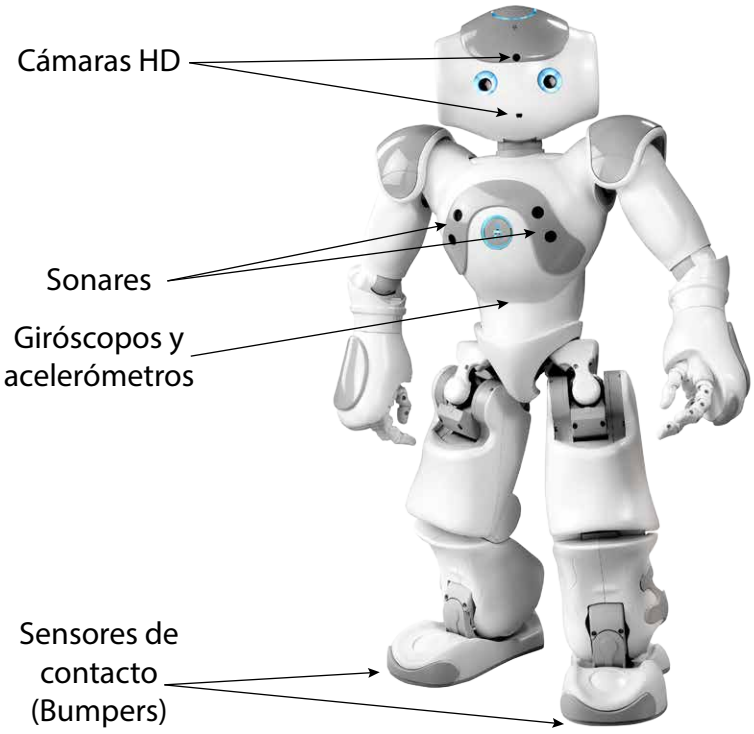


Figura 1.2: Robot Nao y sus principales sensores utilizados en esta tesis.

equipo pueden compartir sus mapas locales, para generar un mapa mejorado, llamado **mapa compartido**.

Para realizar este *tracking* se desarrolla un método basado en *Random Finite Sets*, utilizando el filtro PHD[5], pero basado en gaussianas (GM-PHD) mostrado en [6].

1.2. Objetivos

1.2.1. Objetivos Generales

El objetivo general de esta tesis consiste en realizar *tracking* robusto de obstáculos desde un robot, dentro del contexto de fultbol robótico, un ambiente dinámico donde tanto el robot que realiza el *tracking* como los robots observados están movimiento.

Al tener una buena estimación de la posición de los otros robots en la cancha es posible generar un mapa de obstáculos preciso que permita realizar tareas de alto nivel, como pases, formaciones o estrategias de equipo. Por otro lado, un buen mapa de obstáculos permite que cada robot tome mejores decisiones como el lugar hacia donde dirigir la pelota o donde posicionarse en la cancha.

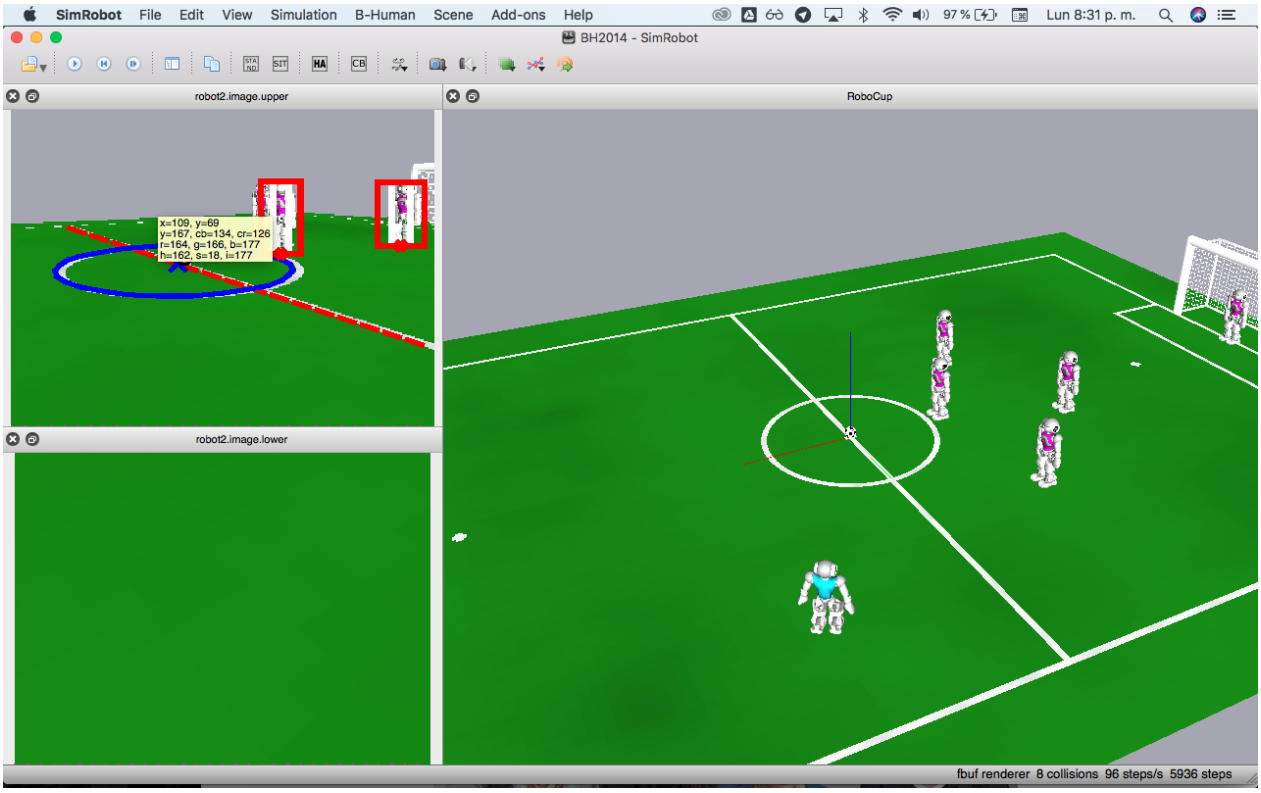


Figura 1.3: Software Simrobot, donde se pueden observar las imágenes creadas por las cámras simuladas, y el escenario 3D completo.

1.2.2. Objetivos Específicos

- Implementar el *framework* de *Random Finite Sets* de manera eficiente en términos del tiempo de procesamiento, de tal manera que pueda ser utilizado en tiempo real en un robot Nao.
- Generar un mapa local de obstáculos, el cual sea de utilidad para realizar tareas más complejas dentro de un partido de fútbol robótico.
- Compartir información entre robots con tal de tener un mapa global de obstáculos.

1.3. Hipótesis

La hipótesis de este trabajo es que se puede realizar *tracking* en tiempo real de manera robusta en un ambiente altamente dinámico (donde tanto el robot como los obstáculos están en movimiento), utilizando un robot que posee poca capacidad de procesamiento. Para esto se implementa un *tracking*, basado en RFS, en robots Nao, y se comparan sus resultados con un método tradicional como el filtro EKF multi-hipótesis.

También se plantea en esta tesis que al utilizar la información de los robots compañeros se puede mejorar el resultado obtenido por el *tracking* normal. Para esto se implementa un método que mezcla la información de los robots compañeros, y el resultado se compara con el *tracking* basado en RFS y el EKF multi-hipótesis.

1.4. Aportes del Trabajo de Tesis

Es esta tesis se presenta un método que utiliza RFS para realizar *tracking* en ambientes altamente dinámicos y que utiliza robots con capacidades computacionales limitadas, algo que no ha sido investigado en la literatura hasta la fecha. En general, se han utilizado los métodos de RFS en problemas de *tracking* donde el observador no se mueve, o aplicado en robots móviles realizando *mapping*, es decir, ubicando en el espacio obstáculos estáticos. Por otra parte, los estudios realizados con obstáculos y robots móviles utilizan computadores con gran capacidad de procesamiento, a diferencia del caso estudiado en esta tesis.

1.5. Estructura de la Tesis

Esta tesis se estructura de la siguiente manera: En el Capítulo 2 se muestran los principios básicos del *tracking*, para luego avanzar al *tracking* de múltiples objetos en el Capítulo 3. Luego, se explica los principios del RFS en el Capítulo 4, junto con los principales trabajos relacionados a esta tesis en esta materia. Así, se llega al desarrollo de esta tesis mostrado en el Capítulo 5, para luego presentar los resultados en el Capítulo 6. En el Capítulo 7 se muestran las conclusiones obtenidas y el trabajo futuro. Finalmente, se presentan las Referencias de las cuales se obtuvo la información necesaria para el desarrollo de este trabajo, junto con los Anexos que muestran los papers asociados a esta tesis.

Capítulo 2

Principios del Tracking

El objetivo de este capítulo es introducir los conceptos necesarios para entender el *tracking*, en particular, de el *tracking* un solo objeto (*single-target*). La Sección 2.1 se comienza mostrando de manera formal el problema general de filtrado[7], ya que el *tracking* es un caso particular de esto. Luego en la Sección 2.2 se muestra el filtro recursivo de Bayes, para luego mostrar aproximaciones de este en las secciones 2.3 y 2.4

2.1. Filtrado Estocástico

Antes de presentar la formulación matemática del problema de filtrado es necesario clarificar algunos conceptos:

- *Filtrar* es la operación de extracción de información acerca de una variable de interés x en el tiempo de interés t usando la información obtenida por mediciones hasta (e incluyendo) el tiempo t.
- *Predicción* es una estimación *a priori*. Su objetivo es inferir como será la variable de interés en algún tiempo $t + \tau$ in el futuro ($\tau > 0$) utilizando las mediciones obtenidas hasta el tiempo t.
- *Corrección* es una estimación *a posteriori* en la que se utilizan mediciones obtenidas después del tiempo de interés.

Así, se considera el siguiente problema general de filtrado estocástico, en un espacio de estados dinámico[8][9]:

$$\dot{x}_t = f(t, x_t, u_t, v_t) \tag{2.1}$$
$$z_t = g(t, x_t, u_t, w_t) \tag{2.2}$$

donde las ecuaciones 2.1 y 2.2 son conocidas como la ecuación de estados y la ecuación de medición respectivamente. Además se definen los siguientes términos:

- x_t representa el vector de estados del sistema en el tiempo t, el cual se encuentra en el espacio de estados $\mathcal{X} \subseteq \mathbb{R}^{n_x}$.

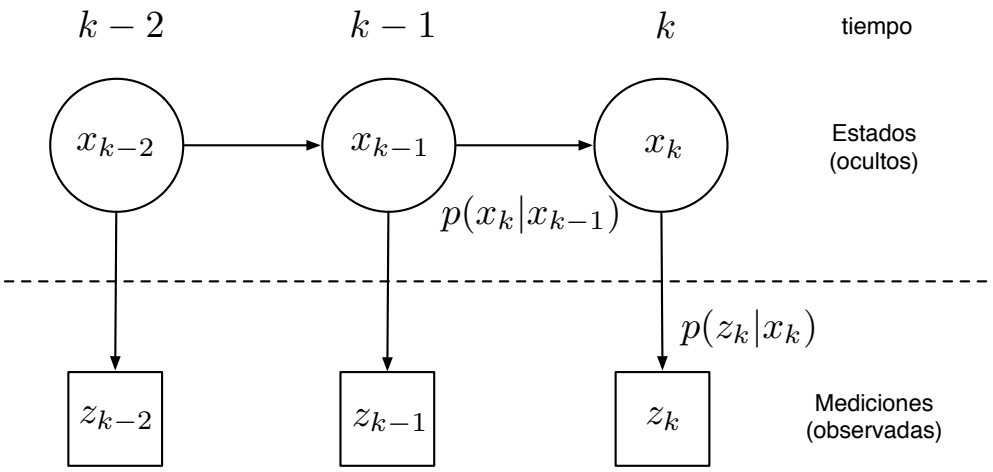


Figura 2.1: Modelamiento general del problema en tiempo discreto

- z_t es el vector de medición obtenida en el tiempo t, la cual se encuentra en el espacio de mediciones $\mathcal{Z} \subseteq \mathbb{R}^{n_z}$.
- u_t representa el vector de entrada al sistema o control.
- $f : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_x}$ es una función vectorial que puede variar en el tiempo, que representa el modelo del proceso.
- $g : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_z}$ es una función vectorial que puede variar en el tiempo, que representa el modelo de medición del proceso
- v_t representa el ruido de proceso.
- w_t representa el ruido de medición.

La formulación mostrada representa un problema general de filtrado en tiempo continuo, pero en esta tesis se aborda el filtrado en tiempo discreto. De todas maneras, cualquier problema continuo se puede transformar en discreto si se muestrea la salida y se utiliza un acumulador de orden cero en la entrada. Así, la derivada se convierte en una diferencia, por lo que el problema general en tiempo discreto corresponde a:

$$x_k = f(x_{k-1}, v_{k-1}) \tag{2.3}$$
$$z_k = g(x_k, w_k) \tag{2.4}$$

donde v_k y w_k pueden ser vistos como secuencias aleatorias de ruido blanco en el dominio de tiempo discreto. La Ecuación 2.3 caracteriza la función de probabilidad de transición de estado $p(x_k|x_{k-1})$ mientras que la Ecuación 2.4 describe la función de probabilidad $p(z_k|x_k)$ de obtener una medición z_k, también llamada **verosimilitud**.

De esta manera, se puede describir un modelo general de filtrado en tiempo discreto de manera gráfica, como se muestra en la Figura 2.1. Dada un densidad inicial $p(x_0)$, una densidad de transición $p(x_k|x_{k-1})$, obtenida a partir del modelo del proceso, y una función de verosimilitud $p(z_k|x_k)$, obtenida del modelo de medición, el objetivo del filtrado es estimar de manera óptima el estado actual en el tiempo k, dadas las mediciones hasta el tiempo k, lo que es en esencia calcular la densidad de probabilidad *a posteriori* $p(x_k|z_{0:k})$ o $p(x_{0:k}|z_{0:k})$.

2.1.1. Caso particular: Tracking

Como se dijo anteriormente, el problema de seguimiento o *tracking* de un objeto (target), es un caso particular del filtrado, donde el estado a estimar x corresponde a la posición y/o velocidad de este objeto y f representa el modelo de movimiento del objeto, al cual se le realiza un seguimiento utilizando mediciones de este estado z, las cuales en general son obtenidas mediante algún sensor, cuyo modelo de medición corresponde a g, y que no necesariamente es capaz de observar todo el estado x, y donde se omite el termino u, ya que en general en el *tracking* no se puede controlar el objeto al cual se le realiza el seguimiento.

2.2. Filtro de Bayes

La teoría Bayesiana[10] es una rama de la matemática probabilística que permite modelar las incertezas del mundo, utilizando información previa y evidencia observacional. De aquí surge el filtro de Bayes, que es un método probabilístico que permite estimar una densidad de probabilidad desconocida (PDF) recursívamente utilizando mediciones obtenidas en el tiempo. Luego, como se mencionó anteriormente, lo que se planea encontrar es una estimación *a posteriori* de la forma:

$$p(x_k|z_{0:k}) \tag{2.5}$$

que representa la distribución de probabilidad del estado x en el tiempo k dada todas las mediciones obtenidas. En realidad este método no busca calcular esta distribución utilizando todas las mediciones, sino que plantea una relación de recursión entre las PDFs tal que

$$p(x_k|z_{0:k}) = f\left[p(x_{k-1}|z_{0:k-1}), z_k\right] \tag{2.6}$$

es decir, una relación donde se pueda obtener la estimación actual utilizando la estimación anterior y la ultima medición. Para encontrar esta relación se realizan dos suposiciones importantes:

1. El estado x sigue un proceso markoviano, es decir, el estado actual x_k solo depende del estado anterior x_{k-1}. Esto implica que:

$$p(x_k|x_{1:k-1}) = p(x_k|x_{k-1}) \tag{2.7}$$

2. Una medición z_k solo depende del estado actual x_k, lo que implica que:

$$p(z_k|x_{1:k}) = p(z_k|x_k) \tag{2.8}$$

Con esto es posible obtener la recursión, ya que por el teorema de Bayes se tiene que:

$$p(x_k|z_{0:k}) = \frac{p(z_k|x_k)p(x_k|z_{0:k-1})}{p(z_k|z_{0:k-1})} \tag{2.9}$$

Con esto se obtiene la recursión buscada, ya que la PDF $p(x_k|z_{0:k})$ queda descrita por los siguientes tres términos:

- **Predicción:** La PDF $p(x_k|z_{0:k-1})$ define el conocimiento que se tiene del modelo

$$p(x_k|z_{0:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{0:k-1})dx_{k-1} \tag{2.10}$$

donde $p(x_k|x_k-1)$ es la PDF de la transición del estado, la cual se puede obtener de la Ecuación 2.3.

- **Verosimilitud:** la verosimilitud $p(z_k|x_k)$ que se determina por el modelo del sensor mostrado en la Ecuación 2.4, y que se denomina como $g(z_k|x_k)$.
- **Evidencia:** el denominador se calcula con la integral

$$p(z_k|z_{0:k-1}) = \int p(z_k|x_k)p(x_k|z_{0:k-1})dx_k \tag{2.11}$$

El cálculo o aproximación de estos tres términos son la base de cualquier filtro Bayesiano y son la base de los métodos mostrados a continuación. En general, se dividen estas ecuaciones en dos etapas que forman la recursión mostrada:

- **Predicción**:

$$p_{k|k-1}(x_k|z_{k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{k-1})dx_{k-1} \tag{2.12}$$

- **Actualización**:

$$p_{k|k}(x_k|z_k) = \frac{g_k(z_k|x_k)p_{k|k-1}(x_k|z_{k-1})}{\int g(z_k|x_k)p_{k|k-1}(x_k|z_{k-1})dx_k} \tag{2.13}$$

2.3. Filtro de Kalman

El filtro de Kalman[11] es un solución cerrada del filtro de Bayes, para el caso donde tanto las distribuciones como los ruidos son Gaussianos, y cuando la función de estados y de medición son lineales. Como una distribución Gaussiana es completamente caracterizada por su primer y segundo momento estadístico (es decir, la media y la covarianza), es suficiente propagar estos momentos en el tiempo en vez de toda la PDF. Para la demostración del filtro de Kalman se utiliza mínimos cuadrados en [11], pero en [12] se encuentran las mismas ecuaciones utilizando solo densidad de probabilidades al igual que en el filtro Bayesiano. Por lo tanto, el filtro de Kalman es una solución óptima según Bayes para un problema de *single-target*, si se cumplen los suposiciones de linealidad y Gaussianidad.

De esta manera el filtro de Kalman permite estimar una función de probabilidad de manera recursiva a través de los siguientes pasos:

1. **Inicialización**: Primero, se asume que la estimación inicial del estado x sigue una distribución Gaussiana dada

$$\mathcal{N}(x; \hat{x}_{0|0}, P_{0|0}) \triangleq \frac{1}{\sqrt{\det(2\pi P_{0|0})}} \exp\left(-\frac{1}{2}(x-\hat{x}_{0|0})^T P_{0|0}^{-1}(x-\hat{x}_{0|0})\right) \tag{2.14}$$

   con media $\hat{x}_{0|0}$ y covarianza $P_{0|0}$.
2. **Predicción**: Como se asume que la ecuación de estados y de medicíon son lineales, las ecuaciones 2.3 y 2.4 se simplifican a

$$x_k = F_{k-1}x_{k-1} + v_{k-1} \tag{2.15}$$
$$z_k = H_{k-1}x_{k-1} + w_{k-1} \tag{2.16}$$

donde F_{k-1} y H_{k-1} son las llamadas matrices de transición de estados y medición, respectivamente. Luego, suponiendo que el ruido de proceso v_{k-1} es un ruido Gaussiano de media cero y covarianza Q_{k-1}, se tiene que la estimación o *predicción* $\hat{x}_{k|k-1}$ del estado x_k según la Ecuación 2.15:

$$\hat{x}_{k|k-1} = F_{k-1}\hat{x}_{k-1|k-1} \tag{2.17}$$

donde la covarianza de la predicción es

$$P_{k|k-1} = F_{k-1}P_{k-1|k-1}F_{k-1}^T + Q_{k-1} \tag{2.18}$$

3. **Corrección**: De la misma manera, se supone que el ruido de medición w_{k-1} es un ruido Gaussiano de media cero y covarianza R_{k-1}. Por lo tanto, según la Ecuación 2.16 se tiene una predicción de la medición $\hat{z}_k$ que se debería obtener, dado el estado estimado $\hat{x}_{k|k-1}$. Esta predicción está dada por la siguente ecuación:

$$\hat{z}_k = H_{k-1}\hat{x}_{k|k-1} \tag{2.19}$$

Con esto, el filtro de Kalman indica que la mejor estimación que se puede tener del estado se obtiene de la siguiente ecuación:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - \hat{z}_k) \tag{2.20}$$

la cual es conocida como la *ecuación de corrección*. Este filtro también entrega la covarianza que se obtiene al realizar esta corrección, la cual viene definida por

$$P_{k|k} = (I - K_kH_{k-1})P_{k|k-1} \tag{2.21}$$

donde se tiene la ganancia de Kalman K_k:

$$K_k = P_{k|k-1}H_{k-1}^T S_k^{-1} \tag{2.22}$$

Y donde S es la llamada matriz de innovación:

$$S_k = H_{k-1}P_{k|k-1}H_{k-1}^T + R_{k-1} \tag{2.23}$$

2.3.1. Filtro de Kalman con ganancia constante

Debido a que el cálculo de la ganancia de Kalman requiere una inversión de una matriz en cada instante, el paso de corrección es el más costoso computacionalmente. Es por esto que asumir una ganancia de Kalman constante aumenta mucho la eficiencia del algoritmo. Al asumir esto, ya no es necesario calcular la covarianza P_k, por lo que solo es necesario propagar la estimación x_k del estado en el tiempo. Es por esto que una filtro con ganancia de Kalman constante puede ser visto como una aproximación del filtro de Bayes, donde solo se propaga el primer momento estadístico de la PDF $p(x)$.

Como la ganancia de Kalman converge para el caso donde el ruido de proceso y de medición son constantes, la supocición de una ganancia constante sirve para problemas que cumplan estas características. El filtro más conocido de ganancia constante es el *filtro alpha-beta*[13], el cual es muy usado en aplicaciones de *tracking* de vehículos aéreos.

2.3.2. Filtro de Kalman para sistemas no lineales

En la demostración de el filtro de Kalman se asume que el sistema, es decir, la ecuación de estados y de medición son funciones lineales. Sin embargo, en muchas aplicaciones reales estas suposiciones no se cumplen. Para estos casos, el filtro de Kalman extendido (EKF)[8], y el filtro de Kalman *unscented* (UKF)[14] permiten utilizar la recursión del filtro de Kalman cuando las ecuaciones de estado y de medición son ligeramente no lineales.

- **Filtro de Kalman Extendido**: En este caso se utiliza una aproximación con serie de Taylor de primer orden para linealizar las funciones f y g dadas en las ecuaciones 2.3 y 2.4. Esto permite el uso de la misma estructura del Filtro de Kalman, ya que la matrices F_k y H_k se pueden calcular con el jacobiano

$$F_{k-1} = \left.\frac{\partial f(x,0)}{\partial x}\right|_{x=\hat{x}_{k-1}}, \qquad H_{k-1} = \left.\frac{\partial g(x,0)}{\partial x}\right|_{x=\hat{x}_{k|k-1}} \tag{2.24}$$

  y así utilizar las ecuaciones 2.15 a la 2.22
- **Filtro de Kalman *unscented***: mientras que el EKF lineariza las funciones no lineales de proceso y medición, la idea del UKF es aproximar directamente la PDF. Al asumir distribuciones Gaussianas, estas pueden ser descritas utilizando *sigma points* $\mathcal{S}$, los cuales son seleccionados utilizando la media y varianza de una gaussiana, y el número de puntos depende de la dimensión de ésta. Luego cada punto es transformado utilizando las funciones no lineales de las ecuaciones 2.3 y 2.4. Los puntos resultantes son luego utilizados para obtener una representación transformada del estado inicial. Para más información revisar [14].

2.4. Filtro de Partículas

El filtro de Kalman necesita que la distribución de la PDF sea parametrizable completamente a través del primer y segundo momento estadístico (distribuciones Gaussianas). Por lo tanto no permite manejar problemas con distribuciones arbitrarias. Además, los filtros EKF y UKF solo funcionan en casos donde el modelo de proceso y medición son medianamente no lineales. Es por esto que surge el filtro de partículas [15], ya que es una implementación no paramétrica del filtro de Bayes que permite manejar distribuciones arbitrarias utilizando un número finito de muestras. Así, un filtro de partículas se puede utilizar para modelar distribuciones de cualquier tipo, incluso multimodales. El filtro de partículas también es conocido como el método secuencial de Monte Carlo (SMC).

En el filtro de partículas, la PDF $p(x|z_{0:k})$ es aproximada por ν partículas

$$x^{(1)}, \ldots, x^{(\nu)} \sim p(x|z_{0:k}) \tag{2.25}$$

donde cada partícula $x^{(i)}$ es una realización de la variable aleatoria x. Se asocia un peso normalizado $w^{(i)} > 0$ a cada partícula tal que $\sum_{i=1}^{\nu} w^{(i)} = 1$. De esta manera la aproximación de la PDF $p(x|z_{0:k})$ usando ν partículas esta dado por la sumatoria ponderada de partículas:

$$p(x|z_{0:k}) \approx \sum_{i=1}^{\nu} w^{(i)} \cdot \delta(x - x^{(i)}), \tag{2.26}$$

donde la función δ representa la función delta dirac. De este modo, si se utiliza una gran cantidad de partículas $\nu \to \infty$ se puede obtener una representación exacta de la PDF.

Al igual que el filtro de Kalman, el filtro de partículas cuenta de tres pasos:

1. **Inicialización**: Si se tiene la información de la distribucion inicial $p(x_0)$ del estado x, entonces el filtro se inicializa extrayendo ν partículas de esa distribución:

$$x_0^{(1)}, \ldots, x_0^{(\nu)} \sim p(x_0) \tag{2.27}$$

   Si no se tiene ninguna información de la distribución inicial del estado x se puede comenzar utilizando una función uniforme en toda la región de interés el espacio de estados. Además se inicializan los pesos de cada partícula $w_0^{(i)} = 1/\nu$.
2. **Predicción**: Asumiendo que en el tiempo $k-1$ se tiene una distribución de la PDF $p(x_{k-1}|z_{0:k-1})$ representada por un conjunto de partículas $\{w_{k-1}^{(i)}, x_{k-1}^{(i)}\}_{i=1}^{\nu}$ de manera que

$$p(x_{k-1}|z_{0:k-1}) \approx \sum_{i=1}^{\nu} w_{k-1}^{(i)} \delta(x_{k-1} - x_{k-1}^{(i)}) \tag{2.28}$$

   Entonces se puede obtener una predicción del estado utilizando el modelo de transición f de la Ecuación 2.3 para cada partícula i

$$x_{k|k-1}^{(1)}, \ldots, x_{k|k-1}^{(\nu)} \sim f(x_{k-1}^{(i)}, v_{k-1}) \tag{2.29}$$

   con lo que se tiene una PDF estimada dada por

$$p(x_{k|k-1}|z_{0:k-1}) \approx \sum_{i=1}^{\nu} w_{k-1}^{(i)} \delta(x_{k|k-1} - x_{k|k-1}^{(i)}) \tag{2.30}$$

   donde los pesos $w_{k-1}^{(i)}$ no varían en este paso.
3. **Corrección**: En este paso, la verosimilitud $p(z_k|x_k)$ obtenida de la Ecuación 2.4 se debe evaluar para cada una de las ν partículas. El peso de cada una de estas se actualiza de la siguiente forma

$$w_k^{(i)} = \frac{p(z_k|x_{k|k-1}^{(i)}) w_{k-1}^{(i)}}{\sum_{e=1}^{\nu} p(z_k|x_{k|k-1}^{(e)}) w_{k-1}^{(e)}} \tag{2.31}$$

   donde el denominador asegura que los pesos actualizados sigan sumando uno. Finalmente se tiene que la estimación para el estado x_k viene dada por la suma ponderada de las partículas actualizadas, tal que

$$p(x_k|z_{0:k}) \approx \sum_{i=1}^{\nu} w_k^{(i)} \delta(x_{k|k-1} - x_{k|k-1}^{(i)}) \tag{2.32}$$

Un problema de esta solución es la degeneración de las partículas, es decir, que los pesos se concentran en solo algunas de ellas. Esto se debe a que, durante la etapa de corrección la varianza entre los pesos aumenta, y en ningún momento disminuye. Para evitar esto, las partículas de muy bajo peso son eliminadas mientras que las partículas con peso alto son elegidas varias veces, en un proceso llamado remuestreo o *resampling*. En la literatura existen muchas estrategias diferentes para realizar esta etapa [16], las cuales difieren en complejidad computacional y exactitud de la estimación.

2.4.1. Distribución Propuesta

El filtro de partículas requiere extraer muestras a partir de distribuciones arbitrarias. Esto se puede realizar de manera eficaz en distribuciones uniformes o gaussianas, pero en general no es posible realizarlo en cualquier distribucíon arbitraria $p(x)$. En este caso, las partículas son extraídas a partir de una distribución propuesta $q(x)$ que sea similar a $p(x)$:

$$p(x) > 0 \implies q(x) > 0 \quad \forall x \in \mathbb{R}^n \tag{2.33}$$

Para la distribución propuesta es común utilizar una densidad uniforme o gaussiana con ciertos parámetros. Mas, las partículas extraídas no aproximan la PDF $p(x)$ sino que la distribución propuesta $q(x)$. Es por esto que en el paso de corrección el peso de las partículas deben ser calculados de la siguiente manera

$$\tilde{w}_k^{(i)} = \frac{p(z_k|x_{k|k-1}^{(i)})w_{k-1}^{(i)}}{q(x_{k|k-1}^{(i)})} \tag{2.34}$$

donde cada peso debe ser normalizado para que todos sumen uno

$$w_k^{(i)} = \frac{\tilde{w}_k^{(i)}}{\sum_{e=1}^{\nu} \tilde{w}_k^{(e)}} \tag{2.35}$$

Así, se puede realizar el procedimiento inducido anteriormente del filtro de partículas para casos donde no es posible extraer partículas directamente de la distribución.

Capítulo 3

Multi-target *Tracking*

Como se mencionó anteriormente, los métodos de *tracking* pretenden estimar el estado x de un target u objeto, que en general se refiere a estimar su posición o también su velocidad. Cualquiera de los métodos mostrados anteriormente son capaces de realizar este *tracking single-target* donde se asume implícitamente que el estado x hace referencia a un objeto y que se tiene una sola medición, completa o parcial, de este estado en cada instante. Sin embargo, en muchas aplicaciones es necesario realizar un *tracking* de varios objetos a la vez, y donde además estos objetos son indistinguibles entre sí. A las técnicas utilizadas para resolver estos problemas se les llama métodos de *tracking* multi-target. Existe una gran cantidad de trabajos sobre este tema en la literatura, la cual es no es posible de describir completamente en esta tesis. Para una vision más acabada del tema se puede recurrir a artículos tutoriales del tema como [17, 18, 19, 20, 21]; estudios como [22]; o artículos elementales como [23, 24, 17].

En general los métodos de *multi-target tracking* emplean la técnica de *dividir para conquistar*, de manera de particionar el problema *multi-target* en un grupo de problemas paralelos de *single-target*. En general, para lograr esto se utilizan las siguientes suposiciones:

- Una sola medición es generada por no más de un objeto y ningún objeto genera más de una sola medición.

- En cada tiempo k se tiene acceso a una tabla de *tracks*, la cual es una lista del estado y la covarianza de cada objeto que se cree que esta presente en el tiempo k.

- El movimiento de cada objeto es estadísticamente independiente del resto de los objetos.

Dadas estas suposiciones, existen variados métodos de dividir el problema, donde el principal desafío consiste en asociar cada medición obtenida en un tiempo dado a alguno de los *track* de la tabla de *tracks*, para poder ocupar correctamente alguno de los métodos de *tracking single-target* mencionados en el Capítulo 2. A esta etapa se le conoce como el *data association*. En este capítulo se explica con detalle este problema y se muestran los distintos métodos que existen para resolverlo en la literatura.

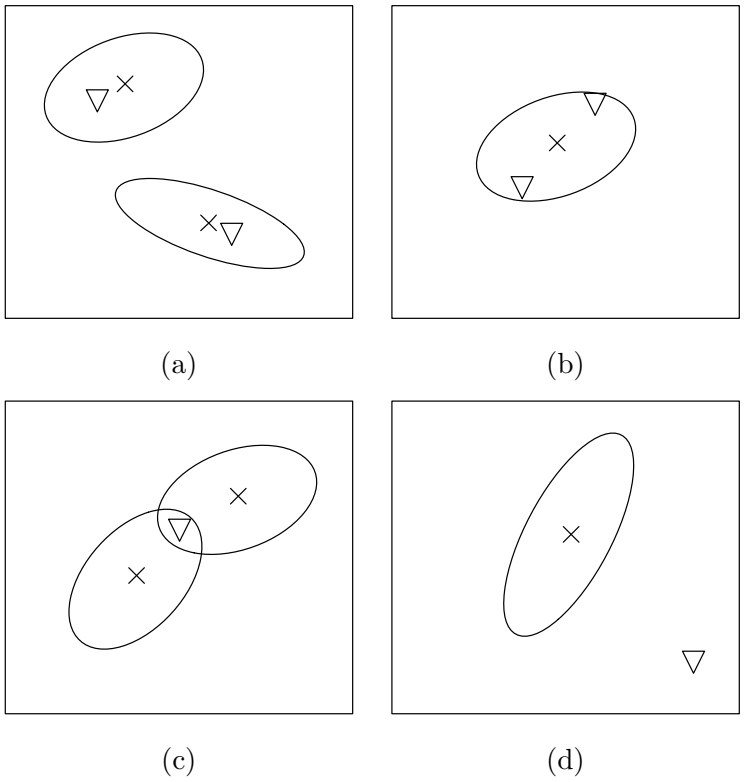


Figura 3.1: Casos de asociación entre *tracks* y mediciones. Los *tracks* se muestran con una cruz, donde la elipse representa su respectiva covarianza, mientras que los triángulos representan mediciones.

3.1. Problema de *Data Association*

El *data associacion problem* constituye un paso fundamental en los algoritmos de *tracking multi-target*. Esto se debe a que, si no se realiza una correcta asociación de las mediciones con sus *tracks*, el *tracking* será deficiente y no logrará estimar el estado del objeto correctamente.

En la figura 3.1 se muestran diferentes casos de asociación entre los *tracks* y las mediciones obtenidas. En el caso 3.1a la asociación es trivial, ya que cada medición esta muy cercana a su respectivo *track*. Esto no ocurre en la caso 3.1b donde, además de tener más mediciones que *tracks*, estos están equidistantes del *track*. Algo similar ocurre en el caso 3.1c donde solo hay una medición para dos *tracks*, y esta se encuentra a la misma distancia de ambos. Y finalmente en el caso 3.1d hay un *track* y una medición, pero ambos están muy alejadas entre sí, por lo que no es claro si la medición hace referencia a un objeto nuevo, es un falso positivo o el objeto se movió significativamente. Todos estos casos representan distintos problemas que se deben resolver para poder realizar un buen *tracking* de los diferentes objetos. A continuación se muestran distintos métodos de abordar este problema.

3.1.1. *Gating*

Para simplificar el paso de asociación es común realizar procedimientos de validación de las mediciones con tal de reducir el número de posibles asignaciones. En aplicaciones donde

se utilice un filtro de Kalman para el *tracking* de los objetos individuales, se puede usar la distancia cuadrática de Mahalanobis (MHD) entre la predicción de la medicion $\hat{z}_k$ y la medición obtenida z_k, la cual viene dada por la siguiente ecuación:

$$d^2_{\text{MHD}} \triangleq (z_k - \hat{z}_k) S_k^{-1} (z_k - \hat{z}_k) \tag{3.1}$$

donde S es la matriz de innovación presentada en 2.23. Con esta distancia es posible crear una cerca o *gate* dentro de la cual es probable que se encuentre la medición con respecto a la predicción de la medición. Esto se realiza utilizando el hecho de que la forma cuadrática del MHD sigue una distribución χ^2 con $\dim(z)$ grados de libertad. De esta manera, el conjunto de mediciones que están dentro de la cerca del i-ésimo *track* queda determinado por

$$Z^{(i)}(\xi) = \left\{ z : d^2_{\text{MHD}} \left(z_k, \hat{z}_k^{(i)} \right) < \xi \right\} \tag{3.2}$$

donde ξ es el umbral del método, el cual se puede calcular usando una tabla de probabilidad de la distribución χ^2. Así, se pueden eliminar todas las posibles asociaciones con mediciones que no se encuentren dentro de la cerca. Aunque este método no representa una solución explícita del *data association problem*, si ayuda a disminuir la complejidad del problema.

3.1.2. Vecino más cercano

El método de vecino mas cercano (NN por su nombre en inglés *nearest neighbor*) simplemente asocia cada medición con el *track* con el cual tenga la menor distancia euclidiana o de Mahalanobis. Si existen muchas mediciones cercanas a un *track* la asociación es ambigua y la probabilidad de asignar una medición de forma errónea es considerablemente alto. Y dado que las asociaciones son irreversibles, el desempeño del algoritmo NN empeora rápidamente en estas situaciones. Además, el algoritmo NN no asegura la clásica suposición de que una medición es generada por máximo un objeto, ya que permite la asociación de una medición con más de un *track*. El vecino más cercano global (GNN o *gloabal nearest neighbor*)[17] asegura la suposición anterior ya que encuentra la mejor asociación usando todos los *tracks* y las mediciones. Una asociación global para m mediciones y n *tracks* se define por el mapeo

$$\theta : \{1, \ldots, n, \text{NO}\} \to \{0, 1, \ldots, m\} \tag{3.3}$$

donde "0" se refiere a los objetos no detectados o fallas de detección, y donde "NO" representa la posibilidad de nuevos objetos. En una asociación global θ, se requiere que una medición z_j esté asociada únicamente con un *track* $x^{(i)}$, es decir, $\theta(i) = \theta(j) > 0$ si y solo si $i = j$. Esto implica que una asociación global no necesariamente asigna a cada medición al *track* con el cual tenga menor distancia de Mahalanobis.

Si se representan todas las posibles asociaciones en una matriz A donde cada elemento a_{ij} representa la asociación $\theta(i) = j$, encontrar la mejor asociación global corresponde a resolver el problema de asignación que originalmente fue investigado en el contexto de asignar un número de trabajos a los trabajadores disponibles. Entonces, algoritmos de asignación óptima como el algoritmo Húngaro [25] o el algoritmo Munkres [26, 27] son usados para obtener la mejor asociación global θ. Otras alternativas de resolver el problema de asignación son el algoritmo Jonker-Volgenant[28] y el algoritmo de subasta[29, 30].

3.1.3. *Data Association* Probabilístico

El desempeño del GNN empeora significativamente en escenarios con un gran número de falsos positivos debido a que la asociación se complica en situaciones ambiguas. El método del *data association* probabilístico (PDA)[31, 32] se basa en la realización una etapa de corrección ponderada de cada *track* con todas las posibles mediciones. Finalmente se intenta aproximar la PDF de la suma ponderada de los resultados anteriores con una sola distribución Gaussiana. Debido a la etapa de corrección ponderada, el PDA evita la difícil y posiblemente errónea decisión de asociación que tiene que realizar el GNN al costo de un incremento en el error de la estimación de la covarianza [33, p. 201].

Para realizar la etapa de corrección ponderada, el método PDA determina la probabilidad de asociación $\beta^{(i,j)}$ para cada *track* i y para todas las mediciones $j = 1, \ldots, m$ del conjunto de mediciones $Z_k = \{z_1, \ldots, z_m\}$. Por lo tanto, la PDF a posteriori del *track* $x^(i)$ esta dada por la suma ponderada de las $m + 1$ posibles asociaciones:

$$p(x^{(i)}|z_1, \ldots, z_m) = \sum_{j=0}^{m} \beta^{(i,j)} p(x^{(i)}|z_j), \tag{3.4}$$

donde $j = 0$ representa la perdida de detección y

$$\sum_{j=0}^{m} \beta^{(i,j)} = 1 \tag{3.5}$$

la PDF $p(x^{(i)}|z_j)$ representa el estado a posteriori del *track* $x^{(i)}$ luego del proceso de correción utilizando la medición asociada z_j. En el caso de una pérdida de detección, el estado a posteriori corresponde a la predicción del estado del *track*. Obviamente la PDF a posteriori (3.4) ya no sigue una distribución Gaussiana, incluso si cada distribución $p(x^{(i)}|z_j)$ es una Gaussiana. Por lo tanto, una aproximación de (3.4) es necesaria para poder utilizar las ecuaciones del filtro de Kalman, para las siguientes etapas de predicción y corrección. Así, para utilizar este método, se calcula el estado a *posteriori* para cada posible asociación, de la misma manera que se realizaba en (2.20):

$$\hat{x}_{k|k}^{(i,j)} = \hat{x}_{k|k-1} + K_k^{(i,j)}(z_j - \hat{z}_k^{(i)}) \tag{3.6}$$

En caso de una pérdida de detección, el estado a *posteriori* esta dado por la predicción anterior correspondiente:

$$\hat{x}_{k|k}^{(i,0)} = \hat{x}_{k|k-1}^{(i)} \tag{3.7}$$

Finalmente, el estado a posteriori del *track* $x^{(i)}$ se obtiene a partir de la media ponderada sobre todas las posibles asociaciones:

$$\hat{x}_{k|k}^{(i)} = \sum_{j=0}^{m} \beta^{(i,j)} \hat{(x)}_{k|k}^{(i,j)} \tag{3.8}$$

La correspondiente covarianza está dada por:

$$P_{k|k}^{(i)} = \sum_{j=0}^{m} \beta^{(i,j)} \left[P_{k|k-1}^{(i)} - K_k^{(i,j)} S_k^{(i,j)} \left[K_k^{(i,j)} \right]^T + (\hat{x}_{k|k}^{(i,j)} - \hat{x}_{k|k}^{(i)})(\hat{x}_{k|k}^{(i,j)} - \hat{x}_{k|k}^{(i)})^T \right] \tag{3.9}$$

donde el último sumando representa la incerteza originada debido a la aproximación por una distribución gaussiana. La contribución de cada diferente asociación entre la medición j y el *track*, que se representa por el peso, es proporcional a la verosimilitud $g(z_j|x^{(i)})$. Ejemplos de como calcular la probabilidad $\beta^{(i,j)}$ se pueden encontrar en [31, 33].

Al igual que el algoritmo de GNN, el desempeño del PDA disminuye en situaciones donde una medición se encuentra cercana a varios *tracks*. Para mejorar el desempeño en tales situaciones, se propuso el algoritmo de *data association probabilístico conjunto* (JPDA)[33, 34]. En este caso, en vez de calcular la probabilidad $\beta^{(i,j)}$ con $j = 1, \ldots, m$ para cada *track* i de manera separada, se calcula $\beta^{(i,j)}$ *conjuntamente* a través de todo el conjunto de *tracks*. Sin embargo, el número de posibles asociaciones es combinacional, lo que implica que la complejidad del método es exponencial. Es por esto que la evaluación de todas las posibles combinaciones de *tracks* con mediciones es solo posible para un número reducido de *tracks*.

Otra extensión del JPDA es el *data association probabilístico integrado* (IPDA)[35] que incorpora inicialización y eliminación de los *tracks* basado en una cadena de Markov sobre la existencia del objeto. En [36] se propone un método de *data association probabilistico integrado conjunto*(JIPDA), el cual utiliza las probabilidades de asociación conjunta en la etapa de corrección y la estimación de existencia adicional. Finalmente en [37] se presenta el algoritmo de *data association probabilístico integrado lineal multi-target* (LM-IPDA) que es una aproximación del JIPDA, logrando así un *tracking multi-target* de complejidad lineal.

3.1.4. Multi-Hipótesis *Tracking*

El algoritmo de Multi-Hipotesis *tracking*(MHT)[38] difiere significativamente de métodos de *tracking* basados en PDA. Mientras que en los métodos de PDA se combinan las múltiples hipótesis obtenidas de la etapa de corrección antes de la siguiente etapa de corrección, en el método de MHT se mantienen todas las hipótesis generadas y se espera que las subsiguientes mediciones obtenidas resuelvan la incerteza de las asociaciones. Esto significa que si en un momento dado existe solo un *track* y se adquieren tres mediciones distintas, entonces, sin contar el nacimiento de nuevos *tracks*, se obtendrán cuatro hipótesis nuevas. La primera representa la pérdida de detección del objeto, y supone que las tres mediciones obtenidas fueron falsos positivos. Las siguientes tres hipótesis corresponden a asociar a el *track* con una de las mediciones, suponiendo que las otras dos son falsos positivos. Esto implica que para casos de asociaciones ambiguas, el problema crece exponencialmente. De este modo, para reducir el costo computacional, se realizan etapas de mezcla y eliminación, más conocidas como técnicas de *merging y pruning*. En [39] se presenta una implementación eficiente del MHT que utiliza el algoritmo de Murty [40] para obtener las l mejores hipótesis sin tener que evaluar todas las posibles hipótesis.

Capítulo 4

Random Finite Sets

Los métodos de multi-target *tracking* introducidos en la sección anterior permiten realizar un *tracking* de varios objetos a la vez, utilizando un conjunto de *tracks* individuales que realizan el *tracking* de cada objeto. Por lo tanto, para utilizar estos métodos es necesaria una asociación explícita entre cada medición y un *track*, la cual tiende a ser ambigua en casos donde se tienen objetos muy cercanos entre sí o en caso de una alta tasa de falsos positivos. Además, cada *track* por separado no tiene ninguna información de la existencia de otros objetos cercanos, y en general la inicialización y eliminación de cada *track* está basada en heurísticas.

Es posible encontrar una generalización del filtro de Bayes de un objeto para el caso de *tracking* de muchos objetos si se modelan tanto los estados del sistema como las mediciones, como conjuntos finitos aleatorios o *random finite sets* (RFSs), el cual es un concepto muy conocido en la teoría de *point-process*[41, 42, 43]. Las *Finite Set Statistics* (FISST)[5, 44, 45] entregan una intuitiva aplicación de la teoría de RFS al problema de multi-target *tracking*, planteando el problema en términos del conocido Filtro de Bayes. Por lo mismo, el Filtro de Bayes *multi-target* resultante es una rigurosa extensión del estándar Filtro de Bayes aplicado a *tracking* multi-target. Los papers "Statistics 101"[46] y "Statistics 102"[47] son una buena forma de comenzar a entender las FISST. Para una fundamentación matemática más detallada referirse a [45]. La relación entre las FISST y la probabilidad teórica de medición es establecida en [48, 49].

El objetivo de este capítulo es resumir los principales conceptos de las FISST e introducir el Filtro de Bayes multi-target, el cual se utiliza durante toda la tesis.

4.1. Formulación del Problema

4.1.1. Espacios de Estado y de Medición

Para iniciar la formulación del problema de *tracking* multi-target, es necesario especificar el espacio de estados y de mediciones que se utiliza en esta caso. Al igual que para el problema mostrado en el Capítulo 3, se supone que en cada momento k se tiene acceso a una tabla de *tracks*, la cual indica el estado de cada objeto al cual se le realiza el *tracking*. Pero, a diferencia de los métodos mostrados en el Capítulo 3, en caso no se utiliza el recurso de *dividir para conquistar*, sino que se planea utilizar toda la tabla como un solo estado, al igual que todas las mediciones como una sola medición. Para esto se cambia el vector de estados por el conjunto finito de todos los *tracks* de la tabla, de manera que

$$X = \{x^{(1)}, \ldots, x^{(n)}\} \in \mathcal{F}(\mathcal{X}) \tag{4.1}$$

como se presentó en la Sección 2.1, $\mathcal{X}$ representa el espacio de estados de cada *track* $x^{(i)}$, por lo que $\mathcal{F}(\mathcal{X})$ representa todas los posibles subconjuntos finitos de $\mathcal{X}$.
De la misma manera, para el caso de las observaciones, se representan todas las observaciones obtenidas como un conjunto, de manera que

$$Z = \{z^{(1)}, \ldots, z^{(m)}\} \in \mathcal{F}(\mathcal{Z}) \tag{4.2}$$

donde $\mathcal{Z}$ representa el espacio de observaciones, por lo que $\mathcal{F}(\mathcal{Z})$ son todos los posibles subconjuntos finitos de $\mathcal{Z}$.

De esta manera, la idea de este método de *tracking* es tomar el conjunto X como el estado multi-target y el conjunto Z como la observación multi-target de un problema de filtrado, donde el espacio de estados está definido por $\mathcal{F}(\mathcal{X})$ y el espacio de observaciones por $\mathcal{F}(\mathcal{Z})$.

4.1.2. Random Finite Sets

En aplicaciones de *tracking single-target* es conveniente utilizar vectores aleatorios para representar la incerteza en el estado de un objeto, al ser este estado una colección de variables aleatorias. De la misma manera, para el caso donde se utilizan conjuntos finitos, la incerteza se representa por *conjunto aleatorios finitos* o Random Finite Sets (RFS). Un conjunto aleatorio se define de la siguiente manera[45, p. 349]: "un conjunto aleatorio es una variable aleatoria que extrae una realización $\Psi = Y$ del hiperespacio $\mathcal{F}(\mathcal{Y})$ el cual consta de todos los subconjuntos finitos Y (incluyendo el conjunto vacío $\emptyset$) de algún espacio $\mathcal{Y}$".
Es decir, un RFS puede representar por completo el estado X o las mediciones Z como una variable aleatoria.

4.2. Notación y abreviaciones

Para diferenciar RFSs de vectores aleatorios, los estados multi-target se representan con letras mayusculas (por ejemplo X), mientras que el estado de cada elemento individual se representa por letras minúsculas (por ejemplo x). Además, para simplificar la notación se utilizan los mismos símbolos para representar un RFS como su realización.

4.3. Distribuciones de Probabilidad Multi-Target

La incerteza de un vector aleatorio x es comúnmente representada por la probabilidad $p(x)$. De la misma manera, una funcíon de densidad de probabilidad multi-target $\pi(X)$ permite una representación de la incerteza acerca del estado multi-target X, la cual incorpora tanto la incerteza del número de objetos presentes en el conjunto y sus estados individuales. La densidad de probabilidad multi-target depende de la cantidad de elementos presentes en X y esta dada por

$$\pi(X) = \begin{cases} \pi(\emptyset) & \text{si } X = \emptyset \\ \pi(\{x^{(1)}\}) & \text{si } X = \{x^{(1)}\} \\ \pi(\{x^{(1)}, x^{(2)}\}) & \text{si } X = \{x^{(1)}, x^{(2)}\} \\ \vdots & \vdots \end{cases} \tag{4.3}$$

En [45, p. 349] se muestran varios ejemplos ilustrativos de densidades de probabilidad multi-target. Uno de los ejemplos hace referencia a una estrella parpadeante en el cielo nocturno, la cual sólo es visible para el observador con probabilidad r y con una distribución espacial $p(x)$. De esta manera, la densidad de probabilidad multi-target de este ejemplo está dada por

$$\pi(X) = \begin{cases} 1 - r & \text{si } X = \emptyset \\ r \cdot p(x) & \text{si } X = \{x\} \\ 0 & \text{si no} \end{cases} \tag{4.4}$$

con la suposición de que una sola estrella implica una probabilidad $\pi(X) = 0$ para todos los conjuntos con cardinalidad $|X| >= 2$.

Debido a la dependencia que tiene la densidad de probabilidad 4.3 de la cantidad de elementos, la evaluación de la integral sobre una densidad de probabilidad multi-target requiere la utilización de una integral de conjunto, la cual se define en [45, p. 361] de la siguiente manera: sea S una región de $\mathcal{X}$ y $f(X)$ una función real de un conjunto finito X, entonces

$$\begin{aligned} \int_S f(X)\delta X &\triangleq \sum_{n=0}^{\infty} \frac{1}{n!} \int_{\underbrace{S \times \cdots \times S}_{n}} f(\{x^{(1)}, \ldots, x^{(n)}\}) dx^{(1)} \cdots dy^{(n)} \\ &= f(\emptyset) + \int_S f(\{x\}) dx + \frac{1}{2} \int_{S \times S} f(\{x^{(1)}, x^{(2)}\}) dx^{(1)} dx^{(2)} + \cdots \end{aligned} \tag{4.5}$$

donde $\pi(\{x^{(1)}, \ldots, x^{(n)}\}) = 0$ si $|\{x^{(1)}, \ldots, x^{(n)}\}| \neq n$. Claramente la evaluación de la integral de conjunto 4.5 para todas las cardinalidades $0, \ldots, \infty$ no es manejable computacionalmente. Sin embargo, densidades de probabilidad como la mostrada en la Ecuación 4.4 si son calculables ya que sólo un subconjunto de todas las posibles cardinalidades debe ser evaluadas. Cabe notar que la integral de conjunto de este ejemplo aún requiere que se calcule una integral multidimensional, lo cual en general puede no ser posible.

La distribución de la cardinalidad de un RFS X, que indica la probabilidad de que X contenga exactamente n vectores, esta dada por

$$\rho(n) = \Pr(|X| = n) = \frac{1}{n!} \int \pi(\{x^{(1)}, \ldots, x^{(n)}\}) dx^{(1)} \cdots dx^{(n)} \tag{4.6}$$

De esta manera, la distribución de la cardinalidad entrega una estimación del número de objetos que están siendo representados por X utilizando por ejemplo el máximo de ρ o una media ponderada.

4.3.1. Random Finite Set con distribución independiente e indénticamente distribuida

La densidad de probabilidad multi-target de un RFS $X = \{x^{(1)}, \ldots, x^{(n)}\}$ con distribución independiente e idénticamente distribuida esta dada por

$$\pi(X) = n! \cdot \rho(n) \cdot p(x^{(1)}) \cdots p(x^{(n)}) \tag{4.7}$$

donde $\rho(n)$ es la probabilidad de distribución de la cardinalidad para $n \geq 0$ y $p(x^{(1)})$ representa la función de densidad de probabilidad del estado del i-esimo objeto.

4.3.2. Random Finite Set con distribución de Poisson

Un RFS con distribución de Poisson se obtiene directamente del proceso i.i.d. al reemplazar la distribución cardinalidad general $\rho(n)$ con la distribución de Poisson

$$\rho(n) = e^{-\lambda} \cdot \frac{\lambda^n}{n!} \tag{4.8}$$

donde λ es el numero esperado de objetos. Por lo tanto, la distribución de Poisson multi-target está dada por

$$\pi(X) = e^{-\lambda} \cdot \lambda^n \cdot p(x^{(1)}) \cdots p(x^{(n)}) \tag{4.9}$$

4.4. Función de Verosimilitud Multi-Target

En single-target *tracking* se utiliza la función de verosimilitud $g(z|\hat{x})$ en el paso de corrección del filtro de Bayes, para evaluar la diferencia entre la medición estimada y la medición recibida. Por el contrario, en el caso de *tracking* multi-target con RFS la función de verosimilitud $g(Z|X)$ tiene que representar todo el proceso de medición, es decir, tiene que incorporar el campo de vision del sensor (FOV), la probabilidad de detección y la tasa de falsos positivos. Para esto se enuncia el modelo general de medición multi-target con RFS, el cual se basa en los siguientes supuestos, los cuales son considerados "standard" en la literatura, como se presenta en [34, 38]:

- El sensor observa una escena que contiene un número desconocido de objetos.
- Una medición solo puede ser generada por a lo más un solo objeto.
- Un solo objeto $x^{(i)}$ puede generar o una medición (detección) con probabilidad $p_D(x^{(i)})$ o no generar una medición (una perdidad de detección) con probabilidad $1 - p_D(x^{(i)})$.
- El proceso de falsas alarmas es uniformemente distribuido en todo el espacio y se distribuye Poisson en el tiempo.
- Las mediciones generadas por los objetos son condicionalmente independientes.
- El proceso de falsas alarmas y los procesos de generación de mediciones por objetos son estadísticamente independientes.

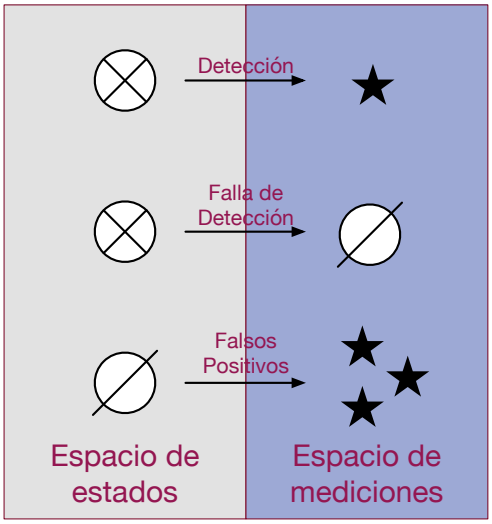


Figura 4.1: Resumen de las posibles mediciones que se pueden obtener en un caso "standar" de *tracking* multi-target. Se pueden tener mediciones acertadas de un objeto presente en la escena, no obtener mediciones de un objeto presente en la escena o obtener falsos positivos.

Un resumen de esto se puede observar en la Figura 4.1. Entonces, suponiendo que en un tiempo k existe un conjunto de *tracks* $X_k = \{x_k^{(1)}, \ldots, x_k^{(n)}\}$, donde cada elemento $x_k^{(i)}$ genera una medición $z_k^{(i)}$ cuya densidad está dada por la Ecuación 2.4, se define un modelo genérico

$$Z_k = C_k \cup \left[\bigcup_{\zeta \in X_k} \Theta_k(\zeta) \right] \tag{4.10}$$

donde $\Theta_k(x_k^{(i)})$ es un RFS que puede tomar los valores $\{z_k^{(i)}\}$ o $\emptyset$ dependiendo si es que el *track* es detectado o no, y donde C_k representa el RFS de las mediciones que corresponden a falsos positivos. De esta manera el modelo toma en cuenta los casos de fallas de detección o falsos positivos.

Luego, así como la función mostrada en 2.4 capturaba la aleatoreidad de la medición obtenida, para el caso multi-target se define la función de verosimilitud $g_k(Z_k|X_k)$ de la siguiente manera. Si $Z_k = \emptyset$ entonces

$$g_k(\emptyset|X_k) = e^{\lambda} \prod_{x \in X_k} (1 - p_D(x)) \tag{4.11}$$

y si $Z_k \neq \emptyset$, entonces:

$$g_k(Z_k|X_k) = e^{\lambda} f_C(Z_k) \cdot g_k(\emptyset|X_k) \cdot \sum_{\theta} \prod_{i:\theta(i)>0} \frac{p_D(x^{(i)}) \cdot g(z_{\theta(i)}|x^{(i)})}{(1 - p_D(x^{(i)})) \cdot \lambda \cdot c_k(z_{\theta(i)})} \tag{4.12}$$

donde la sumatoria se realiza sobre todas las posibles asociaciones θ, de la misma manera en que se presentó en la Subsección 3.1.2, y $c_k(\cdot)$ representa la densidad del RFS $C_k(\cdot)$. Además,

$$f_{C(X_k)}(Z_k) = e^{-\lambda(X_k)} \prod_{z \in Z_k} \lambda \cdot c(z) \tag{4.13}$$

De esta manera se tiene un símil de la función 2.4 pero para el caso multi-target. Una demostración formal de esta ecuación se puede encontrar en [45].

4.5. Funciones de Markov Multi-Target

En el *tracking* single-target, basta solo con predecir el estado de un objeto x para algún tiempo t en el futuro, donde se recibirá la siguiente medición para realizar el *tracking*. Para lograr esta predicción se utiliza una densidad de Markov $f(\hat{x}_k|x_{k-1})$, derivada de la Ecuación 2.3. Pero para el caso multi-target, la ecuación de estados es mucho mas compleja, ya que no sólo debe manejar la evolución del estado de cada uno de los objetos, sino que también debe describir la posible aparición o desaparición de objetos. En la Figura 4.2 se pueden observar las posibles transiciones de estado presentes en un problema multi-target. En el caso general mostrado por Mahler[45] también existe la posibilidad de aparición de nuevos objetos a partir de objetos ya existentes (un ejemplo de esto se da en el *tracking* de células, donde estas pueden separarse para crear dos células a partir de una), pero este caso no es de interés para el problema abordado en esta tesis, por lo cual se omite en adelante.

La densidad de Markov multi-target es matemáticamente similar a la densidad de verosimilitud, donde la aparición y desaparición de objetos corresponden a falsas alarmas y fallas de detección respectivamente. Luego, como es descrito en [45], el modelo "estandar" de transición de estados se basa en las siguientes suposiciones:

- La verosimilitud de que un objeto con estado $\hat{x}$ en un tiempo k si tenia un estado x en un tiempo anterior $k-1$ es descrito por la densidad de Markov $f_{k|k-1}(\hat{x}|x)$, la cual se puede obtener de la Ecuación 2.3.
- Un solo objeto con estado $\hat{x}$ en el tiempo $k-1$ tiene una probabilidad

$$p_S(x) \overset{\text{abbr.}}{=} p_S^{k|k-1}(\hat{x}) \tag{4.14}$$

  de sobrevivir en el tiempo k.
- La densidad de probabilidad de que nuevos objetos X aparezcan en el tiempo k es

$$\Gamma_k(X). \tag{4.15}$$

- La aparición, desaparición y transición son condicionalmente independientes del estado multi-target anterior.

Luego, se define un modelo genérico que representa todas las posibles transiciones de estado mostradas en la Figura 4.2

$$X_k = \left[\bigcup_{\zeta \in X_{k-1}} S_k(\zeta) \right] \cup \Gamma_k(X_k) \tag{4.16}$$

donde $S_k(x^{(i)})$ es un RFS que puede tomar los valores $\{x^{(i)}\}$ o $\emptyset$ y representa la densidad de los objetos que permanecen en la escena en el tiempo k.

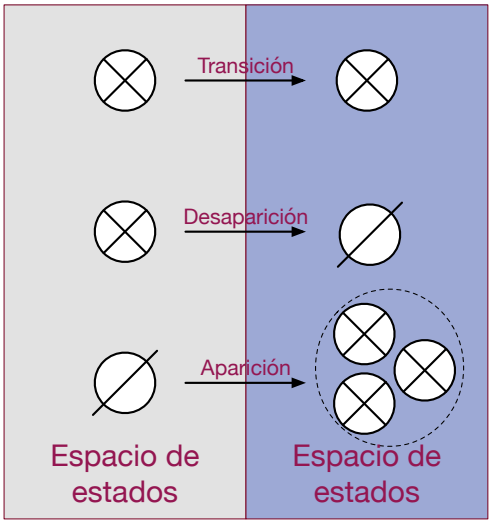


Figura 4.2: Resumen de los posibles cambio de estado existentes en un problema de *tracking* multi-target. El objeto puede permanecer en la escena y evolucionar con respecto la densidad de Markov o desaparecer de la escena. Además nuevos objetos pueden aparecer en la escena.

Luego, la función de densidad que captura la aleatoreidad de la transición de estados para el caso multi-target es la densidad de Markov $f_{k|k-1}(X_k|X_{k-1})$ la cual Mahler[45, p. 472] presenta como

$$f_{k|k-1}(X_k|X_{k-1}) = e^{\mu_0} f_B(X_k) \cdot f_{k|k-1}(\emptyset|X_{k-1}) \cdot \sum_{\theta} \prod_{i:\theta(i)>0} \frac{p_S(x_k^{(i)}) \cdot f_{k|k-1}(x_{\theta(i)}|x_k^{(i)})}{(1-p_S(x_k^{(i)})) \cdot \mu_0 \Gamma(x_{\theta(i)})} \tag{4.17}$$

donde μ_0 es el número esperado de nuevos objetos que aparecerán en el tiempo k y donde la sumatoria se realiza sobre todas las posibles asociaciones θ, de la misma manera en que se presentó en la Subsección 3.1.2. Además

$$f_B(X) = e^{-\mu_0} \prod_{x \in X} \mu_0 \Gamma(x) \tag{4.18}$$

$$f_{k|k-1}(\emptyset|X_{k-1}) = e^{-\mu_0} \prod_{x \in X_{k-1}} (1 - p_S(x)) \tag{4.19}$$

4.6. Filtro de Bayes Multi-Target

La utilización de RFS para capturar incerteza del estado y de las mediciones multi-target, junto con las funciones de verosimilitud y de Markov mostradas anteriormente, permiten extender de manera rigurosa el filtro de Bayes utilizado en el problema de single-target al caso multi-target. Además, como en este caso la variable aleatoria es un RFS, entonces el filtro de Bayes estimará la cantidad de objetos presentes en al escena, además de la incerteza en el estado de cada objeto.

Utilizando la suposición de Markov, toda la información acerca del estado multi-target en un tiempo $k-1$ es capturada por la probabilidad *a posteriori* $p_{k-1}(X_{k-1}|Z_{0:k-1})$. Luego, para

obtener la estimación de la densidad en el tiempo k se puede utilizar la siguiente ecuación de estimación

$$p_{k|k-1}(X_k|Z_{0:k-1}) = \int f_{k|k-1}(X_k|X_{k-1})p_{k-1}(X_k|Z_{0:k-1})dX \tag{4.20}$$

la cual es la homóloga de la Ecuación 2.12, pero con la diferencia de que en este caso la variable aleatoria es un conjunto, por lo que la integral es una integral de conjunto como la mostrada en 4.5.

La densidad *a posteriori* en el tiempo k se obtiene utilizando la ecuación de actualización multi-target

$$p_{k|k}(X_k|Z_{0:k}) = \frac{g_k(Z_k|X_k)p_{k|k-1}(X_k|Z_{0:k-1})}{\int g_k(Z_k|X_k)p_{k|k-1}(X_k|Z_{0:k-1})dX} \tag{4.21}$$

donde se utiliza la función de verosimilitud $g(Z|X)$ mostrada en la Sección 4.4. Nuevamente, esta ecuación es la homóloga de la Ecuación 2.13, pero para el caso multi-target, donde también se utilizan integrales de conjunto.

La formulación del problema en términos del filtro multi-target permite que sea posible encontrar una solución matemáticamente consistente de la densidad de probabilidad multi-target. Pero, la complejidad de calcular integrales de conjunto hacen que el problema intratable desde el punto de vista computacional. Pero, al igual que el filtro de Bayes para el caso single-target, existen muchas aproximaciones en la literatura que permiten el uso del filtro de Bayes en casos multi-target reales.

4.7. Aproximaciones al Filtro de Bayes Multi-Target

Durante la última década, se han propuesto muchas aproximaciones realizables computacionalmente del filtro de bayes para el problema multi-target. Basandose en filtro de Kalman, que aproxima el filtro de Bayes para el caso single-target propagando el primer y segundo momento estadístico, la probabilidad de densidad de hipótesis (PHD)[44] y la probabilidad de densidad de hipótesis cardinalizada(CPHD)[50] son filtros que aproximan el filtro de Bayes para el caso multi-target, propagando solo el primer momento estadístico de la densidad de probabilidad multi-target $p(X|Z_{0:k})$ en el tiempo.

4.7.1. Filtro de Probabilidad de Densidad de Hipótesis

Este filtro es una aproximación del filtro de bayes multi-target, que propaga sólo el primer momento estadístico de la densidad de probabilidad multi-target $p(X)$ en el tiempo. En la teoría de *point process* (PPT), el primer momento estadístico $v(x)$ de una densidad $p(X)$ se denomina *intensidad* de densidad[51], aunque generalmente en el contexto de filtrado multi-target también se le denomina la densidad de probabilidad de hipótesis(*probability hypothesis density* o PHD). Para un RFS X en $\mathcal{X}$ con densidad de probabilidad $p(X)$, su intensidad v en $\mathcal{X}$ cumple que para cada region $S \subseteq \mathcal{X}$

$$\int |X \cap S| p(dX) = \int_S v(x)dx \tag{4.22}$$

En otras palabras, la integral de v sobre cualquier región S entrega el número esperado de elementos de X que están en S. Entones, la integral completa $\hat{N} = \int v(x)dx$ entrega el número esperado de elementos de X. Los máximos locales en la intensidad v son puntos en $\mathcal{X}$ con altas concentraciones locales de número de elementos esperados y por lo tanto pueden ser utilizados para generar la estimación de los elementos de X. El método más simple de filtrado puede ser aproximar $\hat{N}$ a un número entero y elegir esa cantidad de máximos locales de la intensidad [50].

El filtro PHD utiliza la densidad de probabilidad de hipótesis para realizar un proceso de filtrado similar a al filtro de Bayes, en lugar de conjuntos aleatorios (RFS). Es por esto que se utiliza la intensidad de los RFSs mostrados en las secciones 4.4 y 4.5, donde se define

- $\gamma_k(\cdot)$: intensidad del RFS de la aparición de objetos Γ_k en el tiempo k.
- $c_k(\cdot)$: intensidad del RFS de falsos positivos C_k en el tiempo k.

Además, si se consideran las siguientes suposiciones:

- Cada target evoluciona y genera observaciones de manera independiente a cualquier otro target.
- El proceso de falsos positivos es Poisson e independiente de las observaciones generadas por targets reales.
- El RFS de la predicción multi-target dada por $p_{k|k-1}$ es Poisson.

se puede demostrar, utilizando las FISST[44], que la intensidad a posteriori se puede propagar en el tiempo a través de una recursión de PHDs. Entonces, sean $v_{k|k}$ y $v_{k|k-1}$ la intensidades asociadas a las densidades $p_{k|k}$ y $p_{k|k-1}$ mostradas en las Ecuaciones 4.21 y 4.20 respectivamente, se puede generar la siguiente recursión

- **Predicción**:

$$v_{k|k-1}(x) = \int p_S(\zeta) f_{k|k-1}(x|\zeta) v_{k-1}(\zeta) d\zeta + \gamma_k(x) \tag{4.23}$$

- **Corrección**:

$$v_{k|k}(x) = [1 - p_D(x)] v_{k|k-1}(x) + \sum_{z \in Z_k} \frac{p_D(x) g_k(z|x) v_{k|k-1}(x)}{c_k(z) + \int p_D(\xi) g_k(z|\xi) v_{k|k-1}(\xi)} \tag{4.24}$$

De esta manera, al utilizar esta aproximación del filtro de Bayes multi-target se evita toda la complejidad computacional de calcular integrales de conjuntos, ya que esta recursión utiliza la intensidad, la cual es una función en $\mathcal{X}$, y no en $\mathcal{F}(\mathcal{X})$ como las funciones del filtro de Bayes multi-target. Aunque es necesario mencionar que al ser esta una aproximación, no se puede obtener una formula cerrada general.

4.8. Filtro PHD para Modelos Gaussianos Lineales

Como se demuestra en [6], se puede llegar a una fórmula cerrada si los modelos de medición y de actualización asociados son modelos gaussianos lineales. A esta solución se le llama filtro

Gaussian Mixture PHD y se utiliza para crear un algoritmo eficiente y capaz de realizar *tracking* multi-target.

Para realizar esta solución en necesario suponer, además de lo mencionado en la Subsección 4.7.1, las siguientes afirmaciones:

- Cada target tiene un modelo de actualización lineal y gaussiano, y cada sensor tiene un modelo de medición lineal y gaussiano, es decir

$$f_{k|k-1}(x|\zeta) = \mathcal{N}(x; F_{k-1}\zeta, Q_{k-1}) \tag{4.25}$$
$$g_k(z|x) = \mathcal{N}(z; H_k x, R_k) \tag{4.26}$$

  donde $\mathcal{N}(\cdot; m, P)$ denota a una densidad gaussiana con media m y covarianza P, F_{k-1} es la matriz de transición de estados, Q_{k-1} es la covarianza del ruido del proceso, H_k es la matriz de observación y R_k es la covarianza de la observación.
- La intensidad del RFS de las apariciones es una mezcla de gaussianas de la forma

$$\gamma_k(x) = \sum_{i=1}^{J_{\gamma,k}} w_{\gamma,k}^{(i)} \mathcal{N}(x; m_{\gamma,k}^{(i)}, P_{\gamma,k}^{(i)}) \tag{4.27}$$

  donde $J_{\gamma,k}, w_{\gamma,k}^{(i)}, m_{\gamma,k}^{(i)}, P_{\gamma,k}^{(i)}$ para $i = 1, ..., J_{\gamma,k}$ son los parámetros del modelo que determinan la forma de los nuevos targets.

Entonces, dadas estas suposiciones se puede obtener una formula cerrada de la recursión mostrada en las ecuaciones 4.23 y 4.24.

- **Predicción**: Suponiendo que la intensidad *a posteriori* en el tiempo $k-1$ es una mezcla de Gaussianas de la forma

$$v_{k-1}(x) = \sum_{i=1}^{J_{k-1}} w_{k-1}^{(i)} \mathcal{N}(x; m_{k-1}^{(i)}, P_{k-1}^{(i)}) \tag{4.28}$$

  Entonces la predicción de intensidad para el tiempo k es también una mezcla de gaussianas dada por

$$v_{k|k-1}(x) = v_{S,k|k-1}(x) + \gamma_k(x) \tag{4.29}$$

  donde $\gamma_k(x)$ se define en la Ecuación 4.27 y donde

$$v_{S,k|k-1}(x) = p_{S,k} \sum_{j=1}^{J_{k-1}} w_{k-1}^{(j)} \mathcal{N}(x; m_{S,k|k-1}^{(i)}, P_{S,k|k-1}^{(j)}) \tag{4.30}$$
$$m_{S,k|k-1}^{(j)} = F_{k-1} m_{k-1}^{(j)} \tag{4.31}$$
$$P_{S,k|k-1}^{(j)} = Q_{k-1} + F_{k-1} P_{k-1}^{(j)} F_{k-1}^T \tag{4.32}$$

- **Corrección**: Suponiendo que la predicción de la intensidad para el tiempo k es una mezcla de gaussianas de la forma

$$v_{k|k-1}(x) = \sum_{i=1}^{J_{k-1}} w_{k|k-1}^{(i)} \mathcal{N}(x; m_{k|k-1}^{(i)} P_{k|k-1}^{(i)}) \tag{4.33}$$

Entonces la intensidad *a posteriori* para el tiempo k es también una gaussiana dada por

$$v_k(x) = (1 - p_{D,k}(x))v_{k|k-1}(x) + \sum_{z \in Z_k} v_{D,k}(x; z) \tag{4.34}$$

donde

$$v_{D,k}(x; z) = \sum_{j=1}^{J_{k|k-1}} w_k^{(j)}(z)\mathcal{N}(x; m_{k|k-1}^{(j)}(z), P_{k|k}^{(j)}) \tag{4.35}$$

$$w_k^{(j)} = \frac{p_{D,k}(x)w_{k|k-1}^{(j)}q_k^{(j)}(z)}{\kappa_k(z) + p_{D,k}(x)\sum_{l=1}^{J_{k|k-1}} w_{k|k-1}^{(l)}q_k^{(l)}(z)} \tag{4.36}$$

$$m_{k|k}^{(j)}(z) = m_{k|k-1}^{(j)} + K_k^{(j)}(z - H_k m_{k|k-1}^{(j)}) \tag{4.37}$$

$$P_{k|k}^{(j)} = [I - K_k^{(j)}H_k]P_{k|k-1}^{(j)} \tag{4.38}$$

$$K_k^{(j)} = P_{k|k-1}^{(j)}H_k^T(H_k P_{k|k-1}^{(j)}H_k^T + R_k)^{-1} \tag{4.39}$$

Luego, si se tiene una intensidad inicial v_0, entonces se puede obtener la intensidad *a posteriori* para cualquier tiempo k usando la anterior recursión. La demostración de estas ecuaciones se puede encontrar en [6, p. 4096].

4.9. Extensión Filtro PHD para modelos no lineales

Como se mostró en la seccíon anterior, es posible encontrar una formula cerrada del filtro PHD para modelos lineales de medición y actualización. Pero, también es posible extender este filtro para casos donde

$$x_k = \varphi_k(x_{k-1}, \nu_{k-1}) \tag{4.40}$$

$$z_k = h_k(x_k, \varepsilon_k) \tag{4.41}$$

donde φ_k y h_k son funciones no lineales y ν_{k-1} y ε_k son ruido de proceso y ruido de medición Gaussianos de media zero con covarianzas Q_{k-1} y R_k respectivamente. Como φ_k y h_k no son lineales, ya no es posible aproximar estas funciones con una mezcla de gaussianas. Sin embargo, es posible adaptar el filtro PHD para estos casos[6].

Como se mostró en la Subsección 2.3.2, para los casos single-target donde las funciones de actualización y de medición no son lineales se pueden utilizar los filtro EKF[8] o UKF[52], los cuales permiten mantener el esquema de Kalman y la vez manejar funciones no lineales. De la misma manera, se puede extender el filtro PHD a los filtros EK-PHD o UK-PHD si se cambian las funciones presentadas en la sección anterior.

4.9.1. Filtro EK-PHD

Para adaptar el filtro PHD utilizando el mismo enfoque del filtro EKF, es necesario cambiar las siguientes funciones.

- **Predicción**: La ecuaciones 4.31 y 4.32 se deben cambiar por

$$m_{S,k|k-1}^{(j)} = \varphi_k(m_{k-1}^{(j)}, 0) \tag{4.42}$$

$$P_{S,k|k-1}^{(i)} = G_{k-1}^{(j)} Q_{k-1} [G_{k-1}^{(j)}]^T + F_{k-1}^{(j)} P_{k-1}^{(j)} [F_{k-1}^{(j)}]^T \tag{4.43}$$

respectivamente, donde ademas

$$F_{k-1}^{(j)} = \left. \frac{\partial \varphi_k(x_{k-1}, 0)}{\partial \nu_{k-1}} \right|_{x_{k-1}=m_{k-1}^{(j)}} \tag{4.44}$$

$$G_{k-1}^{(j)} = \left. \frac{\partial \varphi_k(m_{k-1}^{(j)}, \nu_{k-1})}{\partial \nu_{k-1}} \right|_{\nu_{k-1}=0} \tag{4.45}$$

.

- **Corrección**: La ecuación 4.37 se cambia por

$$m_{k|k}^{(j)}(z) = m_{k|k-1}^{(j)} + K_k^{(j)}(z - h_k(m_{k|k-1}^{(j)}, 0)) \tag{4.46}$$

y la ecuación 4.39 se cambia por las siguientes ecuaciones

$$S_k^{(j)} = U_k^{(j)} R_k [U_k^{(j)}]^T + H_k^{(j)} P_{k|k-1}^{(j)} [H_k^{(j)}]^T \tag{4.47}$$

$$K_k^{(j)} = P_{k|k-1}^{(j)} [H_k^{(j)}]^T [S_k^{(j)}]^{-1} \tag{4.48}$$

donde

$$H_k^{(j)} = \left. \frac{\partial h_k(x_k, 0)}{\partial x_k} \right|_{x_k=m_{k|k-1}^{(j)}} \tag{4.49}$$

$$U_k^{(j)} = \left. \frac{\partial h_k(m_{k|k-1}^{(j)}, \varepsilon_k)}{\partial \varepsilon_k} \right|_{\varepsilon_k=0} \tag{4.50}$$

Luego, si se tiene una intensidad inicial v_0 es posible encontrar la intensidad *a posteriori* para cada instante k aún cuando las funciones de actualización o medición sean no lineales.

4.9.2. Filtro UK-PHD

Al igual que para el caso EK-PHD es necesario cambiar algunas ecuaciones para manejar la no linealidad de las funciones de actualización o medición.

4.10. Filtro Multi-Sensor Multi-Target

En un escenario distinto al mostrado hasta ahora, existen los casos donde los objetos presentes pueden ser observados más de una vez en el mismo instante de tiempo k. Esto ocurre cuando se tienen varios sensores funcionando de forma paralela, los cuales pueden medir simultáneamente los mismos objetos.

Para resolver este problema Malher[44, p. 1169] demuestra que no es posible extender el filtro PHD de manera directa para el caso multi-sensor, ya que la complejidad de las

ecuaciones no permite una aplicación práctica. Es por esto que el mismo Mahler propone una aproximación del filtro PHD, llamada *corrección iterativa*, en la cual se aplica la Ecuación 4.24 para cada ciclo de mediciones, es decir, para cada sensor.

De esta manera se extienden la Ecuaciones mostradas en la Subsección 4.7.1, de la siguiente manera. Sea el conjunto de observaciones multi-sensor que entregan los s sensores

$$Z_k = Z_k^{(1)} \cup \cdots \cup Z_k^{(s)} \tag{4.51}$$

se tiene que la recursión PHD queda

- **Predicción**:

$$v_{k|k-1} = \int p_S(\zeta) f_{k|k-1}(x|\zeta) v_{k-1}(\zeta) d\zeta + \gamma_k(x) \tag{4.52}$$

- **Corrección**:

$$v_{k|k}(x) \cong G_k^{(1)}(Z_k^{(i)}|x) \cdots G_k^{(s)}(Z_k^{(s)}|x) \cdot v_{k|k-1}(x) \tag{4.53}$$

  donde

$$G_k^{(i)}(Z_k^{(i)}|x) = [1 - p_D^{(i)}(x)] + \sum_{z^{(i)} \in Z_k^{(i)}} \frac{p_D^{(i)}(x) g_k^{(i)}(z^{(i)}|x)}{c_k^{(i)}(z^{(i)}) + \int p_D^{(i)}(\xi) g_k^{(i)}(z^{(i)}|\xi) d\xi} \tag{4.54}$$

Esta aproximación trae algunos problemas, entre ellos que si se cambia el orden de los sensores, el resultado final cambia, algo que no calza con la teoría de *tracking*. Sin embargo se ha demostrado que su aplicación en casos reales no afecta de manera significativa el rendimiento del método[53, 54].

4.11. Evaluación de Desempeño del *Tracking* Multi-target

Para poder evaluar el desempeño de un algoritmo de *tracking* múltiple es necesario tomar variados elementos en cuenta. Mientras que en caso de *tracking* simple solo es necesario medir la diferencia entre la posición estimada y la real, para el caso múltiple también se necesita medir la diferencia en la cantidad de objetos estimados, donde es posible fallar en la generación de obstáculos presentes en la realidad (falsos negativos) y en la generación de obstáculos que no existen en la realidad (falsos positivos). Esta dificultad a la hora de medir el desempeño de un método de *tracking* se ha estudiado en la literatura, siendo el más utilizado el método OSPA (Optimal Subpattern Assignment Metric), debido a una buena respuesta en variados casos clásicos de estudio.

4.11.1. Medición OSPA

Esta métrica fue introducida en [55] para abarcar ciertos problemas en la teoría del *point process* y estadísticas espaciales. Sean X e Y dos conjuntos de puntos con $|X| < |Y|$, se tiene que la distancia OSPA entre ellos se puede obtener como

$$\bar{d}^c(X,Y) = \left(\frac{1}{|Y|} \left(\min_{j \in \{1,\ldots,|Y|\}} \sum_i^{|X|} d^c\left(x^i, y^j\right)^2 + c^2\left(|Y| - |X|\right) \right) \right)^{\frac{1}{2}} \tag{4.55}$$

donde $d^c(x^i, y^i) = \min(c, \|x^i - y^j\|)$, y c es un parámetro de corte que representa la máxima distancia entre dos posiciones asociadas.

En términos prácticos, esta métrica pueda ser calculada realizando los siguientes pasos:

1. Encontrar el subconjunto en Y de $|X|$ elementos que están más cercanos a X. Para encontrar estas asociaciones es necesario usar mecanismos como el método Húngaro, el cual encuentra la asociación óptima entre dos conjuntos de puntos.
2. Para punto y^j de Y, asignar α^j como el valor de corte c si este punto no está asignado a ningún punto en X o al valor mínimo entre c y la distancia con el punto asignado en X.
3. Calcular el promedio cuadrado de las distancias $\left(\left(\frac{1}{|Y|}\right)\sum_{j=1}^{|Y|}\right)^{\frac{1}{2}}$ de los números $\alpha^1, \ldots, \alpha^|Y|$.

Capítulo 5

Tracking de Robots usando Random Finite Sets

Para esta tesis se desarrolló un *tracking* robusto de robots, el cual funciona en el ambiente altamente dinámico del futbol robótico introducido anteriormente. En este capítulo se presenta el problema general a resolver y la implementación realizada para esto. Así, en la primera sección se realiza un descripción del problema, y luego se presentan los sensores utilizados por el robot.

5.1. Descripción del Problema

El principal problema a resolver es el conocer la ubicación de todos los robots en la cancha. Para esto, se utilizan los diferentes sensores de los robots para crear un mapa de robots, en adelante mapa de obstáculos, en la cancha. Planteando esto en términos de *tracking* se tiene que:

- Espacio de estados $\mathcal{M}$: en este problema, el espacio de estados corresponde al plano 2D de la cancha, por lo que cada estado es un vector $m = [x, y]$ que representa la posición de un robot en la cancha. El origen de coordenadas es el centro de ésta. En este caso, no se toma como estado ni la orientación ni la velocidad de los robots, ya que ambas variables son muy difíciles de observar o estimar en este problema.
- Posición del robot X_k: Como ya se ha mencionado, el robot que realiza el *tracking* se encuentra en movimiento, por lo que es necesario obtener su posición en el espacio de estados $\mathcal{M}$, la cual se describe como $X_k = [x_k, y_k]$. A pesar de que el conocer la posición del robot en la cancha es un problema en sí mismo (llamado localización), en esta tesis se asume que esta posición es conocida, ya que en general la localización funciona de manera adecuada. Para esta tesis se utilizó el sistema de localización descrito en [56]. Este sistema de localización se basa en un filtro de Kalman multi-hipótesis, donde la posición inicial de las hipótesis se ubican utilizando la información relacionada a las reglas del futbol robótico.
- Observaciones Z_k: Las observaciones se obtienen de los diversos sensores del robot, por lo que este conjunto esta conformado por las mediciones $Z_k^{(i)}$ de cada uno de los S

sensores. El espacio de observaciones no es el mismo para cada sensor, y se detalla en la sección del modelo de sensor de cada uno de ellos, donde ademas se obtienen la función de verosimilitud $g_k^{(i)}(z|m, X_k)$ y la probabilidad de detección $p_D^{(i)}(m|X_k)$.

- Probabilidad de detección $p_D^{(i)}(m|X_k)$: Este número hace referencia a la probabilidad de que el sensor i pueda realizar una medición de un target en el punto m, si el robot se encuentra en la posición X_k. Esta probabilidad se calcula para cada instante k, y se detalla en la sección del modelo de cada sensor.
- Función de Markov $f_{k|k-1}(m)$: Esta función hace referencia a el modelo de actualización de estados de cada target, es decir, representa al modelo de movimiento de cada robot al cual se le realiza *tracking*. Pero, como los robots Nao se mueven de manera omnidireccional (como una caminata humana), es casi imposible de modelar. Por lo tanto, se asume que los targets están estáticos en la cancha, lo que significa que $f_{k|k-1}(m) = m$. La justificación de por que se utiliza esta suposición y como afecta al funcionamiento final se explican más adelante.

De esta manera, se plantea utilizar el filtro PHD para realizar el *tracking* de problema presentado anteriormente, el cual se reduce a

- **Predicción**:

$$v_{k|k-1}(m) = v_{k-1|k-1}(m) + b_k(m|X_k) \tag{5.1}$$

- **Corrección**:

$$v_{k|k} = \left(\prod_{i=1}^{S} G_k^{(i)}(Z_k^{(i)}|X_k, m) \right) v_{k|k-1}(m) \tag{5.2}$$

donde

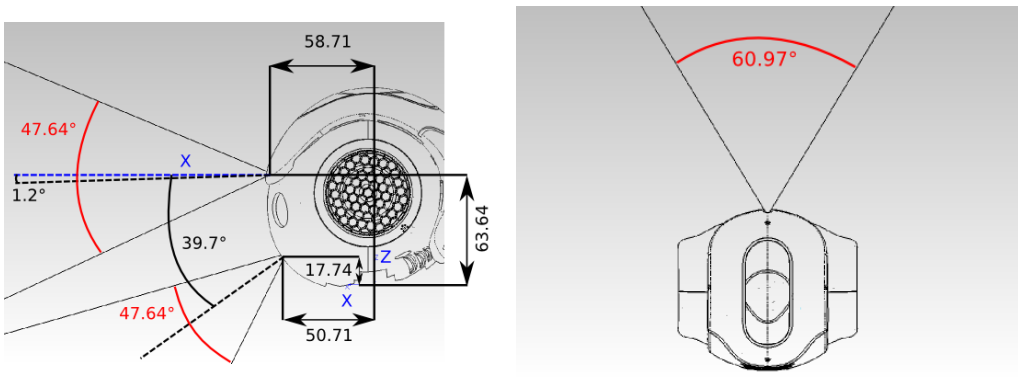
$$G_k^{(i)}(Z_k^{(i)}|m) = [1 - p_D^{(i)}(m|X_k)] + \sum_{z^{(i)} \in Z_k^{(i)}} \frac{p_D^{(i)}(m|X_k) g_k^{(i)}(z|m)}{c_k^{(i)}(z^{(i)}|X_k) + \int p_D^{(i)}(\xi|X_k) g_k^{(i)}(z^{(i)}|\xi) d\xi} \tag{5.3}$$

Esta recursión representa el proceso de *tracking* que realizará cada robot, utilizando la posición del robot X_k y las mediciones de cada sensor $Z_k^{(i)}$.

5.2. Modelo de los Sensores

Las observaciones utilizadas en el problema de *tracking* provienen de los distintos sensores del robot, los cuales son las cámaras de video, el sonar y las posiciones comunicadas de sus compañeros. Para cada uno de los sensores $i \in S$ es necesario obtener su modelo de observación $z^{(i)} = h^{(i)}(m)$, su verosimilitud $g_k^{(i)}(z|m, X_k)$, su probabilidad de detección $p_D^{(i)}(m|X_k)$ y su intensidad de falsos positivos $c_k^{(i)}(z^{(i)}|X_k)$.

Cabe mencionar que todas las mediciones obtenidas por el robot sólo son tomadas en cuenta cuando el robot está parado de manera estable, es decir, que no está recostado en el suelo o caýendose. Esta suposición facilita el cálculo en el modelo de medición de los sensores, además de eliminar las mediciones ruidosas obtenidas mientras el robot se está cayendo.



(a) (b)

Figura 5.1: Cámaras del robot nao, con su ubicación con respecto a la cabeza del robot, y los campos de visión.

5.2.1. Cámara

Las cámaras de video del robot Nao son sus principales fuentes de información. Con estas cámaras el robot detecta líneas, la pelota y otros robots (u obstáculos). Las especificaciones de las cámaras y su ubicación en el Nao se pueden observar en la Figura 5.1.

Para este caso, la detección utilizada es la de obstáculos, para lo cual se utiliza el detector descrito en [57]. Este detector puede encontrar obstáculos en las imágenes obtenidas por ambas cámaras del robot, y su resultado es un punto en la imagen I, lo que representa la ubicación de los pies del robot en la imagen. Por lo tanto, este sensor entrega mediciones I en el espacio de la imagen $\mathcal{I}$.

Modelo de medición

El modelo de medición utilizado para describir el comportamiento de la cámara es conocido como cámara estenopeica, o en inglés como modelo *pin hole*. De manera resumida, este modelo supone que la luz no pasa por un lente hacia el sensor, sino que pasa por un punto.

Como se muestra en la Figura 5.2, esta suposición simplifica en gran medida el cálculo de proyección de un punto en el sensor, lo que se reduce a

$$\begin{pmatrix} I[x] \\ I[y] \end{pmatrix} = -\frac{f}{W[z]} \begin{pmatrix} W[x] \\ W[y] \end{pmatrix} \tag{5.4}$$

donde I es el punto en la imagen resultante y W es el punto en el espacio con respecto a la cámara, donde $W[z]$ es la distancia del objeto. Por lo tanto, si se conoce un la ubicación de un punto $m \in \mathcal{M}$ y se conoce la pose del robot X_k y la pose de la cámara con respecto al robot, entonces es posible calcular la punto donde cualquier punto m se projectará en la cámara, es decir, I.

Para obtener la pose de cámara con respecto al robot se utiliza el modelo del robot, es cual corresponde las posiciones de cada articulación del robot (lo cual esta dado por la construcción

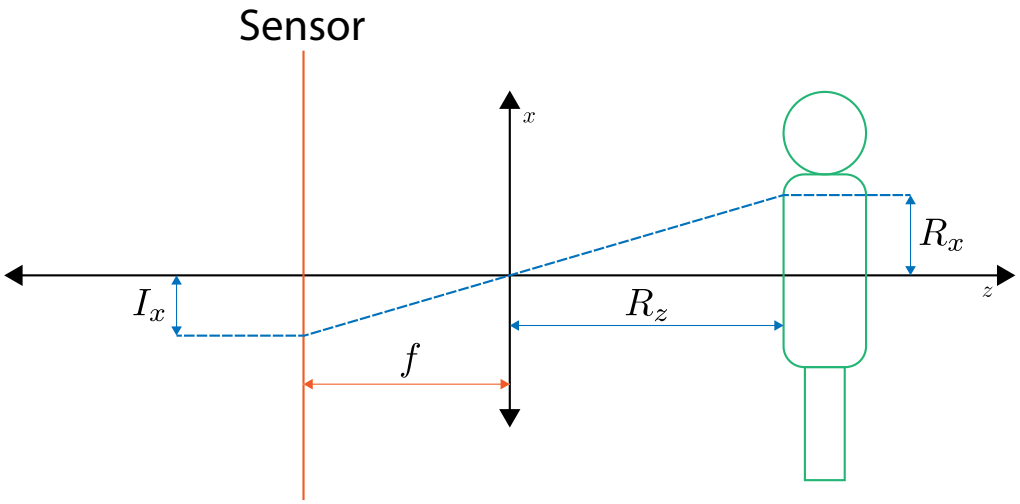


Figura 5.2: Modelo *pin hole* de la cámara. El eje de coordenadas representas el punto por donde pasa la luz, para luego proyectarse en el sensor. En este caso solo se muestra el eje x del sensor, pero el análisis es homólogo para el caso del eje y.

del robot, y se puede observar en la figura 5.1a) y las mediciones de cada articulación del robot. Con esto se puede realizar la cinemática directa de la cámara y así obtener la pose de la cámara C con respecto al robot. Luego, si se tiene un punto en el espacio $\hat{m}$ con respecto al robot, se puede obtener su posición con respecto a la cámara

$$W = C^{-1}\hat{m} \tag{5.5}$$

para luego obtener su posición en la imagen

$$I = \begin{pmatrix} W[y] \\ W[z] \end{pmatrix} \frac{f}{W[x]} \tag{5.6}$$

Entonces, para obtener el punto en la imagen I para cualquier punto $m \in \mathcal{M}$ se debe realizar una transformación de coordenadas globales a las coordenadas del robot, utilizando la pose del robot X_k.

$$\hat{m} = m - X[x, y] \tag{5.7}$$
$$\hat{m} = \text{Rot}(-X[\theta]) * \hat{m} \tag{5.8}$$

Luego, si se siguen los pasos anteriores se tiene que la función buscada

$$z = h(m) \tag{5.9}$$

Verosimilitud

La verosimilitud, que es la densidad de probabilidad de obtener una medición z dado un estado m, se puede obtener utilizando el modelo de medición mostrado anteriormente. Para esto se calcula el error medio de cada observación con respecto a lo que se debería obtener según $h(m)$, en un set de diferentes pruebas. Luego, se descompone este error en los ejes de la imagen para obtener el error medio por eje. Así, suponiendo que la verosimilitud tiene una distribución gaussiana, se tiene que:

$$g_k^{(i)}(z|m, X_k) = \mathcal{N}(z; h(m), R) \tag{5.10}$$

donde $h(m)$ es el modelo de medición y R, es la matriz de covarianza, la cual se obtiene a partir del error medio indicado anteriormente. Para esto se realizó la prueba indica, lo que arrojó que el error de medición en el eje x es de 5 pixeles y en y es de 10 pixeles. Así, se obtiene que la matriz de covarianza es

$$R = \begin{bmatrix} 5^2 & 0 \\ 0 & 10^2 \end{bmatrix} \tag{5.11}$$

Probabilidad de Detección

Para obtener la probabilidad de detección también se utiliza el modelo de medición de la cámara. En esta caso se tiene que

$$P_d(m|X_k) = \begin{cases} 0 & \textbf{si } m \text{ no está en la imagen} \\ f_{p_D,camera}(m) & \textbf{si } m \text{ está en la imagen} \end{cases} \tag{5.12}$$

donde $f_{p_D,camera}(m)$ es una función que varia según el lugar en la imagen en donde se encuentra la medición. Esto ocurre por que, por la naturaleza del detector, es más difícil encontrar un obstáculo si este se encuentra en los bordes de la imagen. Además, esta función toma en cuenta la distancia del obstáculo con respecto al robot, ya que es más difícil medir un obtáculo que se encuentra lejos. Para crear esta función se utilizó un $look - up - table$, o una tabla con valores fijos, que dependen del lugar en la imagen donde se observa el robot. En la Figura 5.3 se observan los valores de p_D para cada sector de la imagen. Con esto se utiliza la función $h(m)$ para obtener la posición en la image y dependiendo de eso se entrega un valor p_D. Para crear esta grilla se utilizaron distintas pruebas empíricas de medición del sensor.

Intensidad de Falsos positivos

Para este sensor se asume que las mediciones están inmersas entre falsas positivos o clutter, el cual es modelado por un RFS Poisson $C_k^{(camera)}$ de intensidad

$$c_k^{(camera)} = \lambda^{(camera)} u(z) \tag{5.13}$$

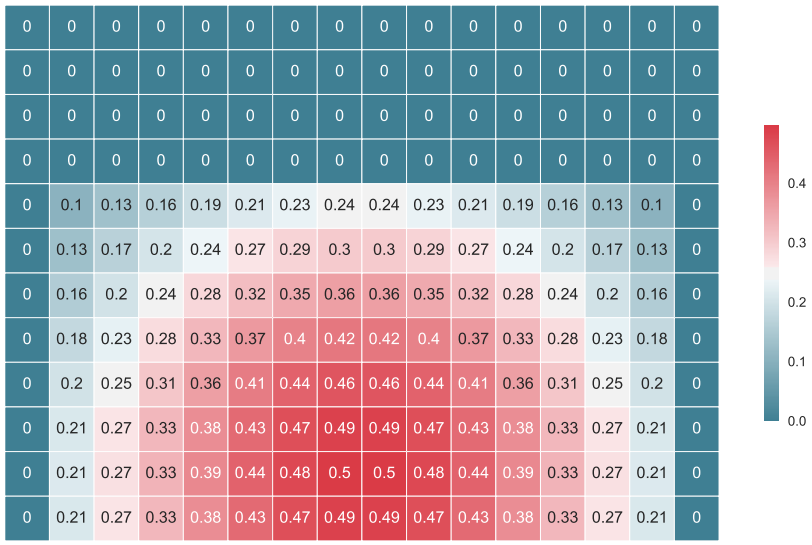


Figura 5.3: Función de probabilidad de detección de los obstáculos, según la posición en la que aparece la base del obstáculo en la imagen. La zona de probabilidad cero superior es la zona de la imagen que se encuentra sobre el horizonte, y se recalcula en cada instante mediante cinemática directa. Los bordes laterales también son llevados a cero, debido a la dificultad de detectar en esa zona.

siendo $u(\cdot)$ la distribución uniforme en toda la cancha y $\lambda^{(camera)} = 5$ el número promedio de falsos positivos en toda la cancha.

5.2.2. Sonar

El sonar es un sensor que tiene el robot en el pecho (mirar Figura 1.2), con el cual es capaz de detectar obstáculos cercanos al robot. El principio de detección del sonar se basa en las ondas mécanicas de sonido, las cuales rebotan en los cuerpos sólidos. Para esto, el robot cuenta con dos parlantes, los cuales emiten sonidos de alta frecuencia, los cuales son captados por dos micrófonos. Ambos pares de parlante y micrófono apuntan en direcciones distintas, para sensar una mayor área. A pesar de que se pueden utilizar ambos sonares de manera combinada, pruebas realizadas con el robot muestran que los resultados de detección empeoran. Es por esto que se eligió tomar cada sonar de manera independiente.

Modelo de medición

La medición obtenida por este sensor corresponde a una distancia entre un objeto y el sonar. Como se puede observar en la Figura 5.4, cada sonar tiene un área de sensado de $\sim 60°$, pudiendo medir objetos entre los $250mm$ y los $800mm$. Cualquier objeto dentro de esa área de trabajo provoca una medición obtenida por la distancia con el objeto. Por lo

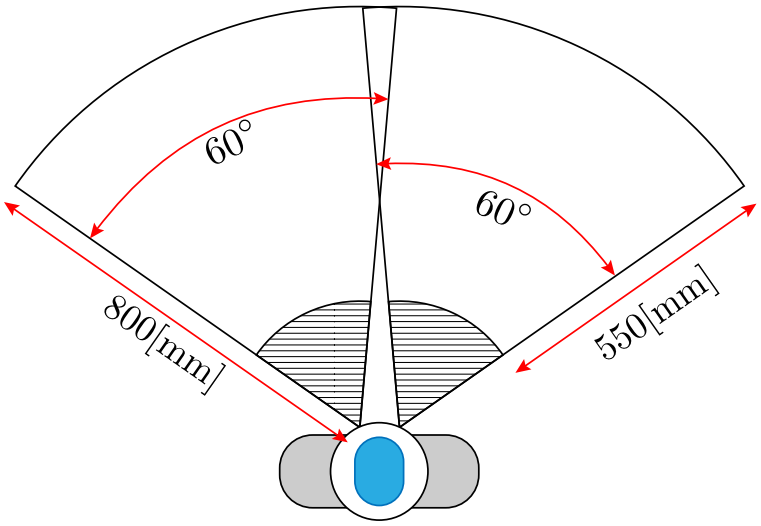


Figura 5.4: Representación de las zonas de medición de los sonares del robot. Ambos conos tienen una apertura de 60°, y una distancia mínima de activación de 250[mm], y una distancia máxima de 800[mm].

tanto, su modelo de medición se puede escribir como:

$$z = \begin{cases} 0 & \textbf{si } dist(s,m) < 250mm \textbf{ o si } dist(s,m) > 800mm \textbf{ o si } |angle(s,m)| > 45° \\ dist(s,m) & \textbf{si no} \end{cases} \tag{5.14}$$

Donde s representa la pose del sensor en $\mathcal{M}$. Esta pose puede conocerse a través de los datos de construcción del robot, los cuales entregan un posición de cada sensor relativa al centro del robot, y con la pose del robot X_k.

Verosimilitud

De la misma manera que para la cámara, se supone que la verosimilitud sigue una gaussiana, en este caso de una dimensión, cuyo centro se obtiene utilizando el modelo de medición obtenido anteriormente y la varianza esta asociada al error de medición, que en el caso de sonares corresponde a $\pm 60mm$. Luego la verosimilitud queda como

$$g_k^{(i)}(z|m, X_k) = \mathcal{N}\left(z; dist(s,m), R\right) \tag{5.15}$$

donde $R = 60^2$.

Probabilidad de Detección

Para obtener la probabilidad de detección de los sonares se realizaron distintas pruebas con el robot, para obtener la tasa de mediciones correctas y de mediciones falsas. Con estos valores se pudo calcular la probabilidad de detección del sensor, la cual esta relacionada con

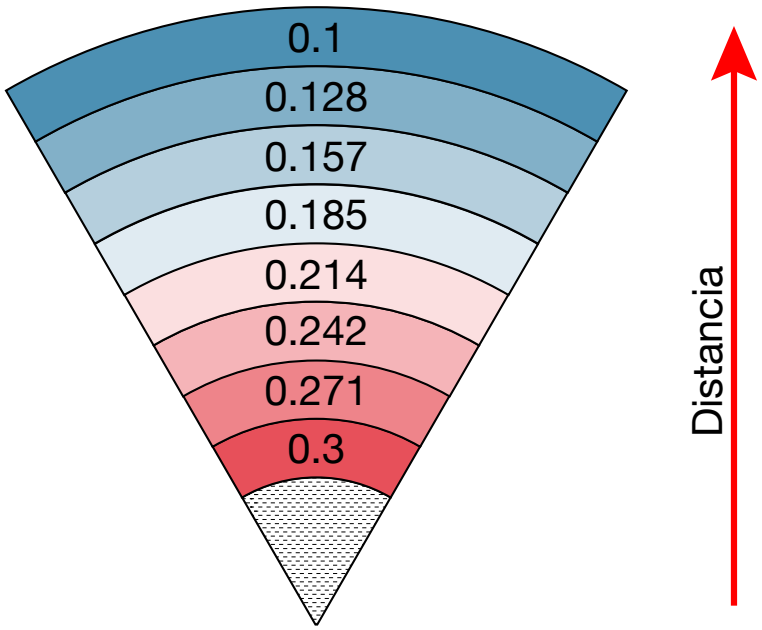


Figura 5.5: Valores de probabilidad de detección de un sonar según la distancia al sensor. Se muestra solo un sonar, ya que ambos tienen los mismos valores de probabilidad. La sección rallada es la zona donde no se puede detectar (por cercanía al sensor).

la distancia del obstáculo al sensor. Con esto, se crea una tabla la cual discretiza la distancia al sensor y asigna una probabilidad de detección a cada celda. En la Figura 5.5 se pueden observar los valores utilizados para los sonares. Luego, la función de probabilidad queda como:

$$p_D^{(sonar)} = \begin{cases} 0 & \textbf{si } dist(s,m) < 250mm \textbf{ o si } dist(s,m) > 800mm \textbf{ o si } |angle(s,m)| > 45° \\ P_D^{(sonar)} & \textbf{si no} \end{cases} \tag{5.16}$$

donde $P_D^{(sonar)}$ es el *look up table* o la función que asigna los valores según la tabla mencionada anteriormente.

Intensidad de Falsos positivos

Para este sensor se asume que las mediciones están inmersas entre falsas positivos o clutter, el cual es modelado por un RFS Poisson $C_k^{(sonar)}$ de intensidad

$$c_k^{(sonar)} = \lambda^{(sonar)} u(z) \tag{5.17}$$

siendo $u(\cdot)$ la distribución uniforme en toda la cancha y $\lambda^{(sonar)} = 100$ el número promedio de falsos positivos en toda la cancha.

5.2.3. Comunicaciones

Como se menciona en la presentación del problema, los partidos constan de cinco robots por lado. Los robots que pertenecen al mismo equipo pueden, por regla, comunicarse entre ellos, compartiendo información. Por ejemplo, los robots se comparten la posición de la pelota en la cancha, la posición de cada uno de ellos en la cancha, etc.

Por otro lado, como tanto para los sonares como para la cámara en la mayoría de los casos, les es imposible discernir si el obstáculo medido corresponde a un robot del equipo contrario o del mismo equipo, es inevitable que el mapa resultante contenga a los robots del mismo equipo. Es por esto que, dado que se conoce la posición aproximada de todos los robots compañeros gracias a las comunicaciones, que se utiliza este dato como un sensor simulado.

Modelo de medición

Este sensor simulado está construido en base a la posición comunicada de cada robot compañero, por lo que el dato comunicado ya se encuentra en el espacio $\mathcal{M}$. Es por esto que el modelo de medición es simplemente la identidad, es decir

$$z = h(m) = m. \tag{5.18}$$

Verosimilitud

Al igual que para los sensores anteriores, la densidad de probabilidad de modela utilizando una gaussiana, en este caso de dos dimensiones (en $\mathcal{M}$). La covarianza de esta gaussiana está dada por la covarianza de la posición comunicada y la covarianza de la propia posición, las cuales se obtienen del método de localización que utilizan los robots. Luego, la verosimilitud está dada por

$$g_k^{(i)}(z|m, X_k) = \mathcal{N}\left(z; m, P^{robotComm} + P^{robot}\right) \tag{5.19}$$

donde $P^{robotComm}$ es la covarianza de la posición del robot que comunicó sus datos y P^{robot} es la covarianza del robot que está realizando el mapa.

5.2.4. Probabilidad de detección

Para la probabilidad de detección asociada a este sensor simulado es necesario extender la metodología utilizada en [6], ya que para calcular esta probabilidad es necesario saber si m es un robot compañero o no. Entonces

$$p_D^{(comm)} = \begin{cases} 0{,}95 & \textbf{si } m \text{ es un robot compañero} \\ 0 & \textbf{si no} \end{cases} \tag{5.20}$$

La forma de saber si m corresponde a un robot compañero se muestra mas adelante.

Intensidad de Falsos positivos

Para este sensor se tiene que no existen falsos positivos, dado que es un sensor simulado, lo que significa que

$$c_k^{(comm)} = 0. \tag{5.21}$$

5.2.5. Mapa estático

Este sensor también es un sensor simulado, el cual agrega al mapa los obstáculos que se sabe *a priori* que existen en la cancha. Estos obstáculos con los cuatro palos verticales de los dos arcos, los cuales son detectados tanto como por el sonar como por el detector visual de robots (esto se debe al algoritmo utilizado para detectar).

Modelo de medición

De la misma manera que para el sensor de comunicación, la posición de los arcos es conocida y se encuentra en el espacio de estados $\mathcal{M}$. Por lo tanto el modelo de observación también es la identidad

$$z = h(m) = m \tag{5.22}$$

Verosimilitud

La verosimilitud se aproxima también con una gaussiana en $\mathcal{M}$, donde el centro está dado por la posición de los postes, y la covarianza esta dada por la covarianza de la posición del robot.

$$g_k^{(i)}(z|m, X_k) = \mathcal{N}\left(z; m, P^{robot}\right) \tag{5.23}$$

donde P^{robot} es la covarianza del robot que esta realizando el mapa.

Probabilidad de detección

Al igual que para el sensor de comunicación, en este caso es necesario saber si m es o no un poste. Luego

$$p_D^{(map)} = \begin{cases} 1 & \textbf{si } m \text{ es un poste} \\ 0 & \textbf{si no} \end{cases} \tag{5.24}$$

La forma en que se diferencia si m es o no un poste se define más adelante.

Intensidad de Falsos positivos

Para este sensor se tiene que no existen falsos positivos, dado que es un sensor simulado, lo que significa que

$$c_k^{(static)} = 0. \tag{5.25}$$

5.3. Implementación

Como se mencionó anteriormente, para desarrollar este problema se utiliza el filtro PHD, implementado con una mezcla de gaussianas, como se muestra en [6]. Con esta metodología, las intensidades se aproximan con una mezcla de gaussianas de la forma

$$\nu_k(m|X_k) = \sum_{j=1}^{J_k} w_k^{(j)} \mathcal{N}(m; \mu_k^{(j)}, P_k^{(j)}) \tag{5.26}$$

Como se puede observar en la implementación del GM-PHD [6], la forma de generar las diferentes etapas del filtro se hacen manejando las gaussianas que corresponden a las intensidades. Sin embargo, dada las diferencias que se introducen al ser este un problema implementado en robots, el desarrollo realizado se acerca más al problema de *mapping* mostrado en [58], donde tanto las funciones de verosimilitud como de probabilidad de detección dependen del estado del robot, en este caso X_k (a diferencia de lo mostrado en [6]).

Pero dado las variaciones que se utilizan en esta tesis, como los sensores simulados explicados anteriormente, es necesario extender la unidad de la gaussiana. Esto significa que cada gaussiana $g_k^{(i)}$ ya no está definida solo por el conjunto de parámetros $\left\{w_k^{(j)}, \mu_k^{(j)}, P_k^{(j)}\right\}$, sino que se extiende al conjunto de parámetros $\left\{w_k^{(j)}, \mu_k^{(j)}, P_k^{(j)}, type_k^{(i)}, player_k^{(i)}\right\}$. Estos parámetros agregan la siguiente información a cada gaussiana:

- *type* : cada gaussiana puede ser uno de los siguientes tipos: *std*, *comm* y *static*. *std* significa un obstáculo estándar, *comm* significa un obstáculo generado por comunicaciones mientras que *static* hace referencia a los obstáculos generados por los objetos estáticos de la cancha. Este parámetro se utiliza para calcular la probabilidad de detección de los sensores, como se muestra más adelante.
- *player* : este parámetro solo tiene validez si la gaussiana es de tipo *comm*, e indica a cual robot compañero hace referencia esta gaussiana. Cada robot tiene un número único del 1 al 5.

Con esto, se tiene que la nueva representación de la intensidad está dada por

$$\nu_k(m|X_k) \coloneqq \left\{v_k^{(j)}\right\}_{j=1}^{J_k} \tag{5.27}$$

donde cada

$$v_k^{(j)} \coloneqq \left\{ w_k^{(j)}, \mu_k^{(j)}, P_k^{(j)}, type_k^{(i)}, player_k^{(i)} \right\} \tag{5.28}$$

Luego, teniendo la nueva representación de una gaussiana v como el conjunto de los parametros mostrado anteriormente, se pueden reescribir las funciones de detección, verosimilitud y probabilidad de detección de todos los sensores mencionados como funciones de una gaussiana, es decir, $f(v)$. Esto es relativamente simple para los modelos de medición y la verosimilitud, ya que el punto m queda representado por el centro de la gaussiana $v[\mu]$, por lo que no es necesario más cambios. Sin embargo, para la probabilidad de detección de los sensores simulados, ahora se puede conocer si una gaussiana es o no un compañero de equipo o un obstáculo estático de la cancha. Es por esto que las ecuaciones 5.20 y 5.24 quedan como

$$p_D^{(comm)}(v|X_k) = \begin{cases} 0{,}95 & \textbf{si } g[type] == comm \\ 0 & \textbf{si no} \end{cases} \tag{5.29}$$

$$p_D^{(map)}(v|X_k) = \begin{cases} 1 & \textbf{si } g[type] == static \\ 0 & \textbf{si no} \end{cases} \tag{5.30}$$

Con estas ecuaciones, más lo mostrado para cada sensor, se realiza la implementación del algoritmo general, aplicado para este problema.

5.3.1. Etapas del Algoritmo

Para realizar la recursión mostrada en la presentación del problema, se dividen las dos etapas en varias subetapas, lo que permite explicar en más detalle como se abordó cada paso.

1. **Intensidad de las apariciones**: Antes de realizar la etapa de la predicción es necesario calcular la intensidad de las apariciones $\gamma_k(m|X_{k-1})$. Para realizar esto se utiliza la idea propuesta en [58], donde se utilizan las mediciones del tiempo $k-1$ para generar la intensidad de las apariciones del tiempo k. Como se tiene que cada sensor $i \in S$ genera un conjunto de mediciones $Z_k^{(i)} = \left\{ z_k^{(i,1)}, \ldots, z_k^{(i,J_{\beta,k}^{(i)})} \right\}$ en el tiempo k, donde $J_{\beta,k}^{(i)} = \left| Z_k^{(i)} \right|$, y como se conoce para cada sensor el modelo de medición h_i y el ruido de medición $R^{(i)}$, entonces se pueden calcular la generación de gaussianas utilizando el algoritmo mostrado en 1.
   En este algoritmo se utiliza la inversa de la función de medición h_i^{-1} y el Jacobiano de la función de medición h_i', con respecto al estado $\mu_{\beta,k}^{(i,j)}$ (que es la matriz H_i en el caso de que el modelo de medición sea lineal). Además, el número $\theta_{weight}^{(i)}$ indica el peso inicial que tienen las gausianas, y es un parámetro del algoritmo. La inicialización de los parametros $type_{\beta,k}^{(j)}$ y $player_{\beta,k}^{(j)}$ de cada gaussiana se realizan utilizando la información de cual sensor es el que se está utilizando para generar la gaussiana, y en el caso del sensor de comunicación, cual fue el robot que entregó esta información.

   Cabe notar que el algoritmo itera sobre cada sensor activo(*active*) y no sobre todos los sensores existentes. Esto se hace debido a que algunos sensores, como los sonares,

Algoritmo 1: Generación de intensidad de apariciones

Datos: mediciones $Z_{k-1}^{(i)}$ con $i = 1, \ldots, |S|$

Resultado: γ_k representando en el conjunto $\left\{v_{\beta,k}^{(j)}\right\}_{j=1}^{J_{\beta,k}}$

begin
- $\gamma_k \leftarrow \emptyset$
- **foreach** *active sensor* $i \in S$ **do**
  - $l \leftarrow 1$
  - **foreach** $j \leftarrow 1$ **to** $J_{\beta,k}^{(i)}$ **do**
    - $v_{\beta,k}^{(l)}[w] \leftarrow \theta_{weight}^{(i)}$
    - $v_{\beta,k}^{(l)}[\mu] \leftarrow h_i^{-1}(z_k^{(i,j)}, X_{k-1})$
    - $v_{\beta,k}^{(l)}[P] \leftarrow h_i'\left(v_{\beta,k}^{(l)}[\mu], X_{k-1}\right) R^{(i)} \left[h_i'\left(v_{\beta,k}^{(l)}[\mu], X_{k-1}\right)\right]^T$
    - **if** $i ==$ *communications* **then**
      - $v_{\beta,k}^{(l)}[type] \leftarrow comm$
      - $v_{\beta,k}^{(l)}[player] \leftarrow RobotComm$
    - **else if** $i ==$ *map* **then**
      - $v_{\beta,k}^{(l)}[type] \leftarrow static$
    - **else**
      - $v_{\beta,k}^{(l)}[type] \leftarrow std$
    - $\gamma_k \leftarrow \gamma_k \cup \left\{v_{\beta,k}^{(l)}\right\}$
    - $l \leftarrow l + 1$
  - $J_{\beta,k} \leftarrow J_{\beta,k} + J_{\beta,k}^{(i)}$

no entregan mediciones a la misma frecuencia que la cámara. Es por esto que todo el algoritmo no debe tomar en cuenta a los sensores que aun no entregan un dato, ya que generaría un error en el calculo de los obstáculos.

2. **Predicción**: Como ya se tiene la intensidad de apariciones $\gamma_k(m|X_{k-1})$, se realiza la actualización mostrada en la Ecuación 5.1. Esta etapa es descrita por el Algoritmo 2.

   Como se dijo en la descripción del problema, en este caso la predicción se realiza suponiendo que la función de actualización f es la identidad. Pero, para poder manejar el movimiento de los robots en la cancha, se aumenta la covarianza de cada gaussiana. Esto es conocido un modelamiento de orden cero. Por lo tanto, exceptuando al aumento de la covarianza, este algoritmo se reduce a juntar las gaussianas del paso anterior y la creadas por cada sensor en un solo conjunto de gaussianas.

3. **Correción**: Luego, se realiza el paso mostrado en las Ecuaciones 5.2 y 5.3, lo que se puede observar en el Algoritmo 3. Para este paso es necesario calcular las variables $P_D^{(i,j)}$, $H^{(i)}$, $S_k^{(j)}$ y $K_k^{(j)}$, utilizando las ecuaciones mostradas para cada sensor en la sección anterior. Como se puede observar, en este paso también se itera solo sobre los sensores válidos, debido a la diferencia de frecuencias que se tiene en la entrega de datos.

   En términos prácticos, esta etapa genera una nueva gaussiana por cada gaussiana exis-

Algoritmo 2: Predicción (Ecuación 5.1)

Datos: ν_{k-1} representado en $\left\{v_k^{(j)}\right\}_{j=1}^{J_{k-1}}$ y $\gamma_k(m|X_{k-1})$ representado en $\left\{v_{\beta,k}^{(l)}\right\}_{j=1}^{J_{\beta,k}}$

Resultado: $\nu_{k|k-1}$ representando en el conjunto $\left\{v_{k|k-1}^{(l)}[w]\right\}_{j=1}^{J_{k|k-1}}$

begin
- $\nu_{k|k-1} \leftarrow \emptyset$
- $l \rightarrow 0$
- **for** $j \leftarrow 1$ **to** J_{k-1} **do** `// iteración sobre cada gaussiana del paso` $k-1$
  - $l \rightarrow l+1$
  - $v_{k|k-1}^{(l)}[\mu] \leftarrow v_{k-1}^{(j)}[\mu]$
  - $v_{k|k-1}^{(l)}[P] \leftarrow v_{k-1}^{(j)}[P] + Q$
  - $v_{k|k-1}^{(l)}[w] \leftarrow v_{k-1}^{(j)}[w]$
  - $v_{k|k-1}^{(l)}[type] \leftarrow v_{k-1}^{(j)}[type]$
  - $v_{k|k-1}^{(l)}[player] \leftarrow v_{k-1}^{(j)}[player]$
- **for** $j \leftarrow 1$ **to** $J_{\beta,k}$ **do** `// iteración sobre nueva gaussiana`
  - $l \rightarrow l+1$
  - $v_{k|k-1}^{(l)}[\mu] \leftarrow v_{\beta,k}^{(j)}[\mu]$
  - $v_{k|k-1}^{(l)}[P] \leftarrow v_{\beta,k}^{(j)}[P] + Q$
  - $v_{k|k-1}^{(l)}[w] \leftarrow v_{\beta,k}^{(j)}[w]$
  - $v_{k|k-1}^{(l)}[type] \leftarrow v_{\beta,k}^{(j)}[type]$
  - $v_{k|k-1}^{(l)}[player] \leftarrow v_{\beta,k}^{(j)}[player]$
- $J_{k|k-1} \leftarrow l$

tente anteriormente, para cada una de las mediciones entregadas en el tiempo k, la que se utiliza para realizar la corrección de la gausiana anterior. Luego, esta nueva gaussiana se agrega al conjunto de gaussianas anteriores.

En este paso es necesario destacar que las gaussianas sólo se utilizan como medición para realizar el update si y solo si el tipo de gaussiana y el *player* (en el caso de gaussianas tipo *comm*) lo permiten, según el sensor que se esta utilizando. Esto no está explícito en el algoritmo, pero se encuentra en el cálculo de la probabilidad de detección P_D. Como se vio en la sección anterior, este cálculo necesita de los parámetros *type* y *player*, y si la asociación no es posible, el resultado es cero. Y como el peso final de la gaussiana resultante esta multiplicado por este valor, esto significa que la nueva gaussiana tendrá peso cero, es decir, que no influye en el cálculo de la intensidad a posteriori.

4. **Mezcla y eliminación**: Este paso final no esta relacionado a las ecuaciones del método original. Como se explica en la etapa anterior, por cada gaussiana existente en el tiempo $k-1$ se generan una cantidad igual al número de mediciones obtenidas en el tiempo k. Esta generación provoca un aumento exponencial en el número de gaussianas, lo que no permitiría el funcionamiento correcto de este método. Para solucionar este problema se

Algoritmo 3: Paso de corrección

Datos: $\nu_{k|k-1}$ representando en el conjunto $\left\{v_{k|k-1}^{(j)}\right\}_{j=1}^{J_{k|k-1}}$.

Resultado: ν_k representando en el conjunto $\left\{v_k^{(j)}\right\}_{j=1}^{J_k}$.

begin
  foreach *active sensor* i **do**
    for $j = 1$ **to** $J_{k|k-1}$ **do**
      `Calculate(`$P_D^{(i,j)}$`)`
      $v_k^{(j)}[w] \leftarrow \left(1 - P_D^{(i,j)}\right) v_{k|k-1}^{(j)}[w]$
    $N \leftarrow J_{k|k-1}$
    foreach z *in* Z_k **do**
      for $j = 1$ **to** $J_{k|k-1}$ **do**
        `Calculate(`$H^{(i)}, S_k^{(j)}, K_k^{(j)}$`)`
        $v_k^{(N+j)}[\mu] \leftarrow v_{k|k-1}^{(j)}[\mu] + K_k^{(j)} \left(z - v_{k|k-1}^{(j)}[\mu]\right)$
        $v_k^{(N+j)}[P] \leftarrow \left[I - K_k^{(j)} H^i\right] v_{k|k-1}^{(j)}[P]$
        $\tau^{(j)} \leftarrow P_D^{(i,j)} v_{k|k-1}^{(i)}[w] \left|2\pi S_k^{(i)}\right|^{-0,5} \times$
        $\exp\left(\left(z - v_{k|k-1}^{(j)}[\mu]\right)\left[S_k^{(j)}\right]^{-1}\left(z - v_{k|k-1}^{(j)}[\mu]\right)^T\right)$
      for $j = 1$ **to** $J_{k|k-1}$ **do**
        $v_k^{(N+j)}[w] \leftarrow \tau^{(j)} \left(c(z) + \sum_{l=1}^{J_{k|k-1}} \tau^{(l)}\right)$
      $N \leftarrow N + J_{k|k-1}$
    $J_k \leftarrow N$

realiza un paso de mezcla y eliminación de gaussianas. Este proceso se puede observar en el Algortimo 4

Como se puede observar en el algoritmo, primero se eliminan todas las gaussianas que tengan un peso menos al parámetro $\theta_{minWeight}$. Luego, se realiza una mezcla de gaussianas, para lo cual se utiliza su distancia de Mahalanobis, siempre y cuando las gaussianas se puedan mezclar según su tipo. Como se puede observar, no se realizan mezclas entre gaussianas del tipo *map* con el tipo *comm*, pero sí se mezcla una gaussiana estándar (tipo *std*) con una de cualquier otro tipo, la gaussiana resultante se transforma en el nuevo tipo. Esto en la práctica se da cuando el robot mide con la cámara o los sonares un obstáculo que resulta ser o un compañero de equipo o un poste de arco.

Finalmente, el conjunto $\left\{\tilde{\mu}_k^{(j)}, \tilde{P}_k^{(j)}, \tilde{w}_k^{(j)}, \tilde{type}_k^{(j)}, \tilde{player}_k^{(j)}\right\}_{j=1}^{\tilde{J}_k}$ representa la intensidad *a posteriori* del tiempo k, y es el conjunto que se utilizará para el siguiente paso en el tiempo $k+1$, con lo que se completa la recursión.

Algoritmo 4: Mezcla y eliminación

Datos: El valor mínimo $\theta_{minWeight}$, la distancia máxima $\theta_{distance}$ y la intensidad ν_k

Resultado: ν_k representado por el conjunto de menos gaussianas $\left\{\tilde{g}_k^{(j)}\right\}_{j=1}^{\tilde{J}_k}$

begin
- $I \leftarrow \left\{j = 1, \ldots, J_k | v_k^{(j)}[w] > \theta_{minWeight}\right\}$
- $l \leftarrow 0$
- **while** $J \neq \emptyset$ **do**
  - $l \leftarrow l + 1$
  - $u \leftarrow \arg\max_{j \in I} v_k^{(j)}[w]$
  - $L \leftarrow \emptyset$
  - **for** $j = 1$ **to** J_k **do**
    - **if** $\left(v_k^{(j)}[\mu] - v_k^{(u)}[\mu]\right)^T \left(v_k^{(j)}[P]\right)^{-1} \left(v_k^{(j)}[\mu] - v_k^{(u)}[\mu]\right) \leq \theta_{distance}$ **then**
      - **if** $v_k^{(u)}[type] == comm$ **and** $v_k^{(j)}[type] == comm$ **and** $v_k^{(u)}[player] == v_k^{(j)}[player]$ **then**
        - $L \leftarrow L \cup j$
      - **else if** $v_k^{(u)}[type] == map$ **and** $v_k^{(j)}[type] == map$ **then**
        - $L \leftarrow L \cup j$
      - **else if** $v_k^{(u)}[type] == std$ **and** $v_k^{(j)}[type] == comm$ **then**
        - $v_k^{(u)}[type] \leftarrow comm$
        - $v_k^{(u)}[player] \leftarrow player_k^{(j)}$
        - $L \leftarrow L \cup j$
      - **else if** $v_k^{(u)}[type] == std$ **and** $v_k^{(j)}[type] == map$ **then**
        - $v_k^{(u)}[type] \leftarrow map$
        - $L \leftarrow L \cup j$
  - $\tilde{g}_k^{(l)}[w] \leftarrow \sum_{j \in L} v_k^{(j)}[w]$
  - $\tilde{g}_k^{(l)}[\mu] \leftarrow \frac{1}{\tilde{g}_k^{(l)}[w]} \sum_{j \in L} v_k^{(j)}[w] v_k^{(j)}[\mu]$
  - $\tilde{g}_k^{(l)}[P] \leftarrow \frac{1}{\tilde{g}_k^{(l)}[w]} \sum_{j \in L} v_k^{(j)}[w](v_k^{(j)}[P] + (\tilde{g}_k^{(l)}[\mu] - v_k^{(i)}[\mu])(\tilde{g}_k^{(l)}[\mu] - v_k^{(j)}[\mu])^T)$
  - $\tilde{g}_k^{(l)}[type] \leftarrow v_k^{(u)}[type]$
  - $\tilde{g}_k^{(l)}[player] \leftarrow v_k^{(u)}[player]$
  - $I \leftarrow I \backslash L$
- $\tilde{J}_k \leftarrow l$

5.4. Mapa Local

Con lo mostrado en la sección anterior, se tiene que se puede calcular una intensidad ν_k para cada instante de tiempo k. Pero, la idea de la realización de este método es encontrar un mapa de obstáculos que el robot pueda utilizar. A este mapa se le llama mapa local, y para crearlo se utiliza el algoritmo mostrado en el Algoritmo 5.

Algoritmo 5: Construcción del mapa local

Datos: La intensidad ν_k

Resultado: Mapa Local, representado en el conjunto $\left\{v_k^{(j)}\right\}$

begin
- $Map \leftarrow \emptyset$
- $I \leftarrow \left\{j = 1, \ldots, J_k | v_k^{(j)}[type] \neq std\right\}$
- $L \leftarrow \left\{j = 1, \ldots, J_k | v_k^{(j)}[type] == std \textbf{ and } v_k^{(j)}[w] > \theta_{minWeightMap}\right\}$
- **forall** $j \in I$ **do**
  - $Map \leftarrow Map \cup \left\{v_k^{(j)}\right\}$
- $l \leftarrow 0$
- **while** $L \neq \emptyset$ **and** $l < 5$ **do**
  - $u \leftarrow \arg\max_{j \in L} v_k^{(j)}[w]$
  - $Map \leftarrow Map \cup \left\{v_k^{(u)}\right\}$
  - $l \leftarrow l + 1$

Como se puede observar, todas las gaussianas del tipo *comm* y *map* son agregadas al mapa. Esto se debe a que los sensores asociados a estos tipos de gaussianas son altamente confiables, al ser sensores simulados. Luego de eso, se agregan las gaussianas que tengan el mayor peso, siempre y cuando su peso sea mayor a $\theta_{minWeightMap}$ (valor mínimo para considerar una gaussiana como obstáculo), y sin exceder los cinco obstáculos. Esto se hace ya que si se sacan los robots compañeros y los postes de los arcos, en la cancha deberían haber a los más cinco obstáculos, que serian todos los robots oponentes.

Finalmente , se tiene un conjunto de gaussianas que representan todos los obstáculos presentes en el mapa, con su respectivo centro y covarianza, donde además se diferencian si son robots compañeros (y cual robot compañero), postes de arcos o robots oponentes.

5.5. Mapa Combinado

Como se indicó en la sección 5.2.3, los robots de un mismo equipo pueden comunicar información entre ellos, lo que se utiliza para generar el sensor simulado de la posición. Sin embargo, también es posible comunicar el resultado del mapa local entre ellos, a modo de generar un mejor mapa combinado. Para esto se realizan los siguientes pasos:

- Comunicación: Cada robot comunica su mapa local a todos los otros robots. Sin em-

bargo, como ya se está comunicando la posición de cada robot, y como todos los robots tienen el mismo sensor simulado que indica la posición de los postes, entonces solo se envían las gaussianas correspondientes a robots oponentes, es decir, la gaussianas que tengan $type == std$.

- Merge: Luego, como cada robot tiene el mapa local de todos los robots compañeros, se procede a generar un único mapa que combine la información de todos, para esto se realiza el procedimiento mostrado en el algoritmo 6.

Algoritmo 6: Construcción del mapa combinado

Datos: Los mapas locales $Map^{(r)}_{Local}$ de cada robot compañero r

Resultado: Mapa Combinado, representado en el conjunto $\left\{\mu^{(j)}_k, P^{(j)}_k\right\}$

```
1  begin
2      u ← Own robot number
3      Map_Combined ← { v ∈ Map^(u)_Local | v[type] == std }
4      forall g ∈ Map_combined do
5          r ← 0
6          forall h ∈ Map^(r)_Local and r ≠ u do
7              if (v[μ] − h[μ])^T (v[P])^−1 (v[μ] − h[μ]) < θ_CombDistance then
8                  Map_Combined ← Map_Combined ∪ Merge(v, h)
9              else
10                 Map_Combined ← Map_Combined ∪ v ∪ h
11         r ← r + 1
```

Como se puede observar en el algoritmo, los mapas son combinados utilizando la misma función **merge** de la creación del mapa local, pero en este caso no se comparan todas las gaussianas, sino que solo las que pertenecen a distintos mapas locales.

Finalmente, se tiene que con los dos pasos mostrados anteriormente, se puede obtener un mapa de obstáculos que utiliza la información de todos los robots compañeros.

Capítulo 6

Resultados y Análisis

En el siguiente capítulo se describen las pruebas realizadas para validar el método propuesto, junto con los resultados obtenidos.

Inicialmente, se efectuó un conjunto de pruebas experimentales, donde se analizó la respuesta del método propuesto frente a diversos escenarios controlados, las cuales se compararon contra el método tradicional de multi tracking que mayormente se utiliza en la SPL, el cual se basa en un EKF multi hipótesis (MH-EKF). Estos resultados se presentaron en la publicación [59] (presente en anexo 1).

En una segunda etapa, se implementó la medición OSPA presentada en la Subsección 4.11.1, la cual se utiliza para estandarizar la diferencia entre dos conjuntos de puntos. Luego, se realizaron simulaciones de partidos completos, donde se utilizó esta medición para calcular el error entre el resultado obtenido por el método propuesto contra las posiciones reales de los robots (*ground truth*). En este caso también se comparó la respuesta del método contra el la medición OSPA del método con MH-EKF. Estos resultados se presentaron en la publicación [60], relacionado a esta tesis (presente en anexo 2).

6.1. Resultados Experimentales

Como se indicó anteriormente, en esta etapa se realizaron pruebas utilizando robots reales, pero en escenarios artificiales (es decir, no un partido real), donde se ubican diversos obstáculos en la cancha (tanto estáticos como dinámicos) con posiciones conocidas, y con un robot que ejecuta ambos métodos a comparar. Luego, se compara la diferencias en mediciones como el error en la estimación de la cantidad de obstáculos presentes, y el error medio de la posición de estos.

El propósito de realizar esta primera prueba consiste en obtener una evaluación cualitativa del método propuesto, ya que en los resultados se pueden observar directamente las posiciones y trayectorias de los obstáculos, además de la estimación de la cantidad de obstáculos presentes.

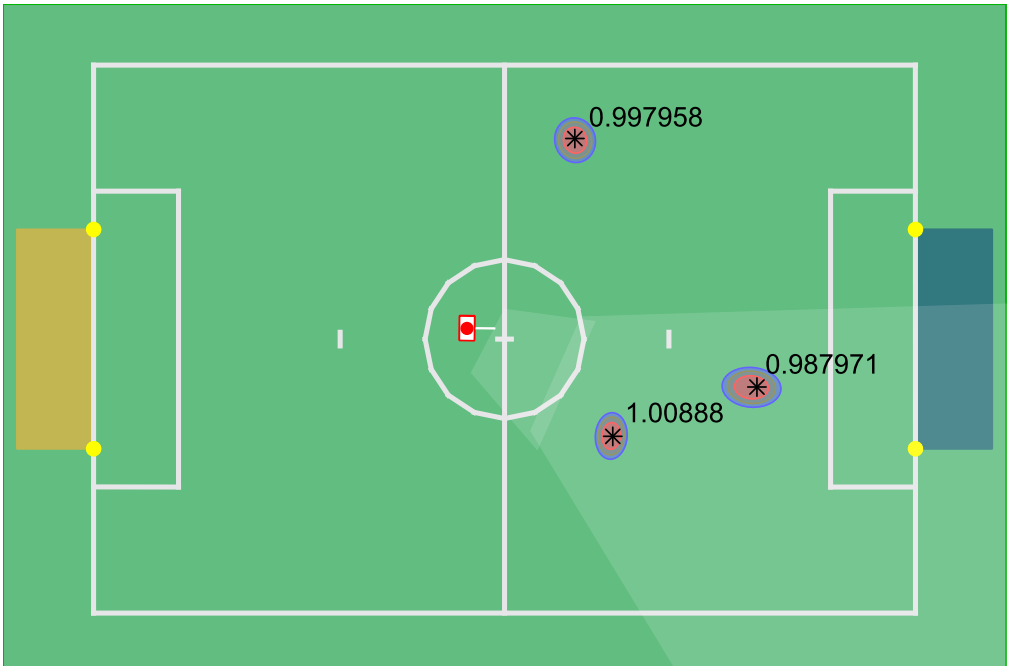


Figura 6.1: Disposición de los objetos en la cancha para el experimento 1. El cuadrado blanco de bordes rojos al centro de la cancha representa al robot que realiza el *tracking*, mientras que los asteriscos negros son los obstáculos estáticos de este experimento.

6.1.1. Configuración de los Experimentos

La etapa de experimentos reales se dividió en dos partes. La primera consta de un robot estático al centro de la cancha, que es el que realiza el mapa, y varios obstáculos ubicados de manera aleatoria. El robot, para observar toda la cancha, mueve constantemente la cabeza, que es donde se ubican las cámaras, por lo que no es capaz de observar todos los obstáculos simultáneamente. En la Figura 6.1 se puede observar las posiciones del robot y de los obstáculos.

Dada la simplicidad del problema, tanto el método propuesto como el metodo de comparación con EKF logran estimar de manera correcta el mapa de obstáculos. El error medio medido para ambos métodos fue de 20[cm], y los tiempos promedio de ejecución de cada ciclo completo fue de 0,13[ms] para el método propuesto y 0,07[ms] para el MH-EKF. Estos tiempos son tiempos reales de procesamiento en el robot, y fueron medidos de manera *online*.

Luego, se ejecutó una serie de pruebas en escenarios que simulaban situaciones de juego reales, ya que tanto el robot que realiza el mapa y algunos obstáculos se mueven en la cancha.

- **Prueba 1**: Como se muestra en la Figura 6.2, el robot que realiza el mapa se mueve desde una orilla de la cancha hacia el centro de ésta, y estima el mapa de obstáculos con ambos métodos. Se puede notar que para cada obstáculo real, el método de MH-EKF genera más de una hipótesis, lo que implica una mala estimación de la cantidad de robots presentes en la cancha. En cambio el método propuesto estima de manera correcta la cantidad de robots en la cancha.
- **Prueba 2**: En esta prueba, además del movimiento que realiza el robot que genera el

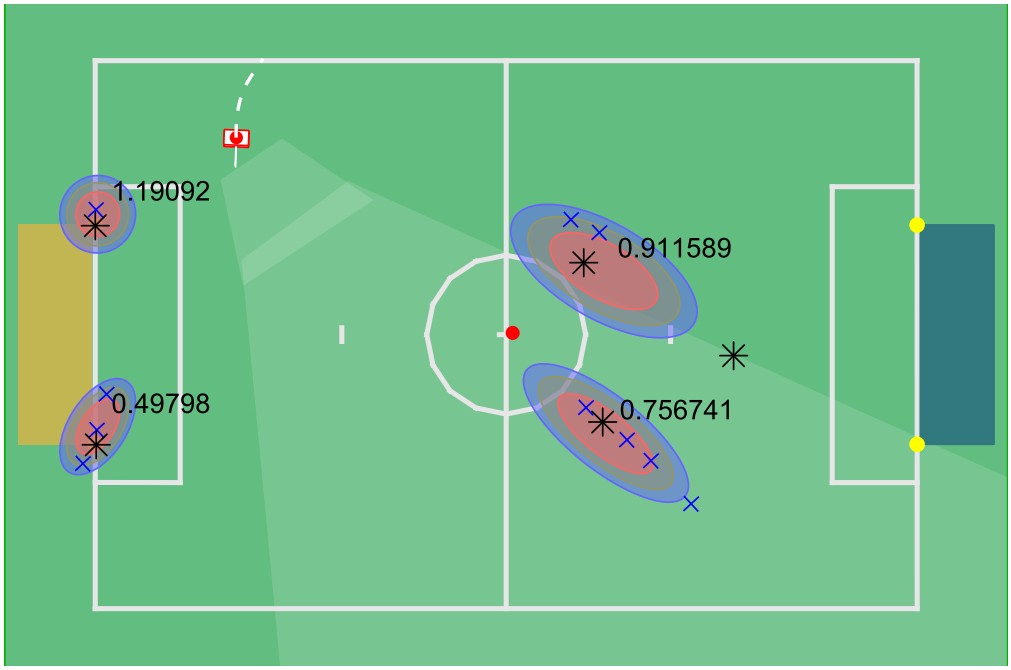


Figura 6.2: En esta caso, el robot realiza una trayectoria, mostrada por una linea segmentada blanca. Los asteriscos negros representan los obstáculos reales, mientras que las cruces azules son las estimaciones entregadas por el método MH-EKF. Además, las elipses de representan la ubicación y la covarianza de las estimaciones entregadas por el método RFS-Mapa-Local.

mapa, se mueve uno de los obstáculos. Como se puede observar la Figura 6.3, el método MH-EKF genera una serie de falsos positivos ubicados en el trayecto del obstáculo que realiza el movimiento. Esto se debe a que el robot que realiza el mapa esta constantemente moviendo la cámara, para poder observar todo el campo, por lo que no puede visualizar al obstáculo durante toda su trayectoria. En cambio el método propuesto en esta tesis estima de manera correcta al obstáculo, combinando todas las observaciones en una sola estimación.

- **Prueba 3**: En esta prueba se realizó un experimento clásico del *tracking* que es el *kidnapping*. Este experimento consiste en que se elimina del mapa un obstáculo, para analizar como se comportan las estimaciones. Este tipo de situaciones ocurre a menudo en el contexto del fútbol robótico, ya que los robots que realizan faltas son removidos de la cancha por un árbitro. La disposición de los robots y obstáculos se puede observar en la Figura 6.4a.

  Como se puede observar en Figura 6.4b, al método MH-EKF le toma aproximadamente $5[s]$ en eliminar por completo la estimación errónea, mientras que al método propuesto le toma $200[ms]$.

- **Prueba 4**: Esta última prueba también consiste en un robot observador que se encuentra en movimiento y en varios obstáculos estáticos. La diferencia es que en este caso dos de los obstáculos se encuentran muy cerca, por lo que la percepción de estos es muy poco precisa. Estos errores dificultan la tarea de estimar y realizar *tracking*.

  Como se puede observar en la Figura 6.5a, el metodo de MH-EKF no logra estimar de manera correcta la cantidad de obstáculos en el lugar donde se encuentran los obstáculos cercanos. Por otro lado, el método propuesto estima los obstáculos con sólo una

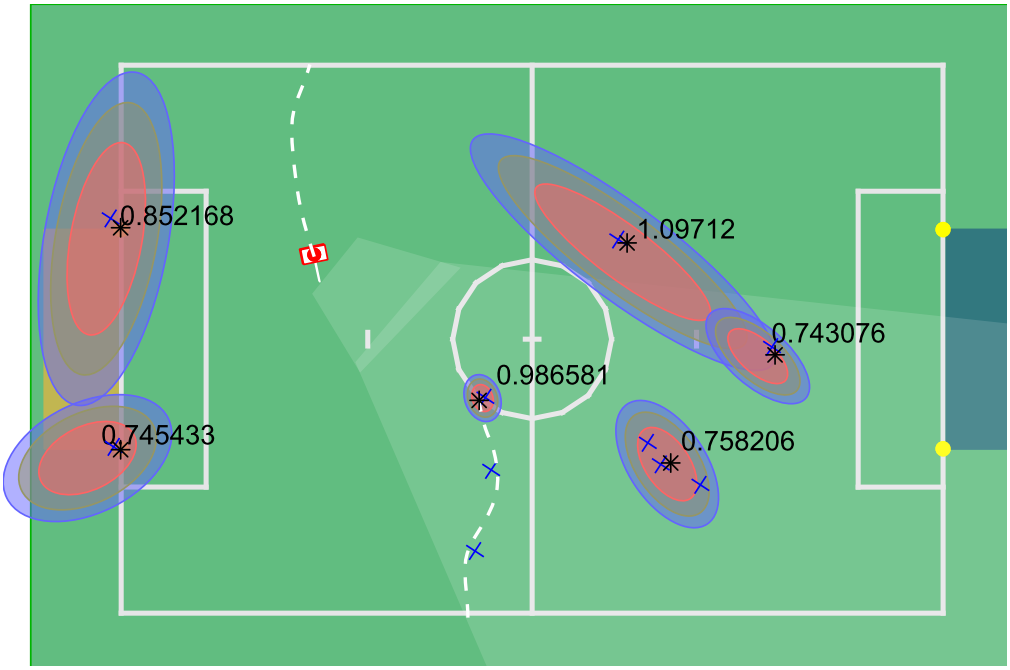
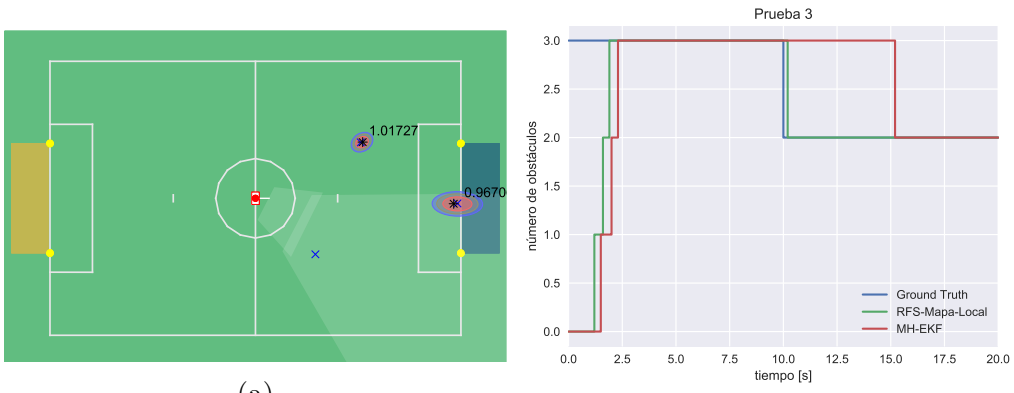
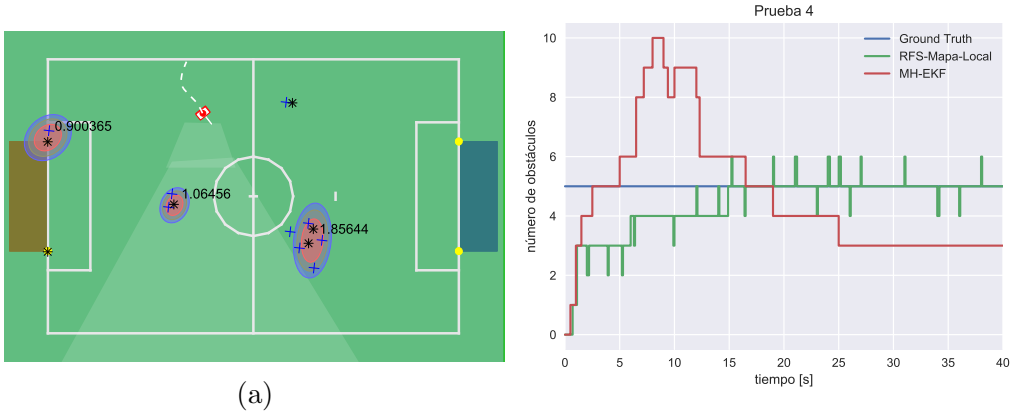


Figura 6.3: En este caso uno de los obstáculos se mueve, realizando una trayectoria mostrada por la linea segmentada blanca. Los asteriscos, cruces azules y elipses representan los obstáculos reales, las estimaciones del método MH-EKF y estimaciones del método RFS-Mapa-Local, respectivamente.



(a) (b)

Figura 6.4: Prueba de una situación de *kidnapping*. En la imagen (a) se puede observar la disposición de los robots en la cancha, donde el robot inferior es el que se remueve. En (b) se puede observar la estimación en el tiempo de la cantidad de obstáculos presentes.



(a) (b)

Figura 6.5: Prueba sobre la dificultad de realizar *tracking* con obstáculos muy cercanos entre sí. Los dos robots de abajo están representado por una sola gaussiana de peso cerca a 2, mientras que el método MH-EKF crea una gran cantidad de falsos positivos.

gaussiana. Sin embargo, el peso de esta muestra que existen dos obstáculos en esa zona. Esta diferencia se puede observar en la Figura 6.5b, que muestra la estimación de la cantidad de obstáculos presentes en la cancha durante la prueba.

Las pruebas mostradas anteriormente muestran de forma cualitativa que el método propuesto en esta tesis supera de manera clara a solución tradicional de MH-EKF, ya que en estas sencillas pruebas, este último tiene variados problemas para estimar los obstáculos de forma correcta. En la sección 6.2 se realizaron distintas pruebas para poder medir de manera cuantitativa el desempeño de ambos métodos.

6.2. Resultados Simulados

Luego de las pruebas experimentales, se realizaron partidos simulados de robots, utilizando el simulador SimRobot[4] mostrado en la Sección 1.1.1. Esta prueba se realizó en un simulador debido a la dificultad de medir la posicíon de todos los robots en la cancha en un partido real. Sin embargo, en el simulador se puede obtener la posición real de cada robot en todo momento, lo que permite la aplicación de la medida OSPA. Esta métrica permite evaluar de forma cuantitativa a ambos métodos.

6.2.1. Configuración de los partidos

Se realizaron dos tipos de partidos. En primer lugar los partidos consisten en duelos de 1 vs 1 robot, donde ambos realizan el mapa local, pero no pueden elaborar el mapa compartido (ya que no existen compañeros en la cancha). La idea de efectuar este tipo de partidos es para mostrar que el método propuesto puede superar de manera correcta el problema del *data association* y que aún sin utilizar la información de los compañeros puede superar a los métodos clásicos. Luego se desarrolló partidos realistas, es decir, de 5 vs 5, donde cada robot elabora el mapa de obstáculos con ambos métodos, utilizando sus sensores y las comunicaciones entre compañeros. En este último caso se puede observar el aporte de compartir información entre los compañeros.

	Prueba	Promedio	Mejor	Peor
RFS-Mapa-Local	1 vs.1	197.8	182.4	212.4
	5 vs. 5	348.6	319.7	381.9
RFS-Mapa-Combinado	5 vs. 5	271.3	225.2	298.7
MH-EKF	1 vs.1	293.6	271.1	305.3
	5 vs. 5	421.5	391.3	453.2

Tabla 6.1: Resultados OSPA obtenidos de las pruebas simuladas

Al igual que para el caso anterior, el método propuesto es comparado contra un EKF multi-hipótesis. En cada uno de estos partidos cada unos de los robots calcula la medida OSPA para ambos métodos, con parámetro de corte $c = 500[mm]$., Para el método de EKF multi-hipótesis los parámetros utilizados fueron los que dieron mejores resultados según la métrica OSPA, los cuales son un distancia de asociación de $500[mm]$ y un tiempo de eliminación de $8[s]$. Para el caso del método propuesto en esta tesis, se utilizaron los parámetros $\theta_{weigth} = 0{,}01$, $\theta_{minWeigth} = 1e-15$, $\theta_{distance} = 18$ y $\theta_{minWeightMap} = 0{,}3$. Estos valores se eligieron realizando pruebas de desempeño, y basándose en valores obtenidos de la literatura. Para los sensores, se utilizaron los parámetros mostrados en la Sección 5.2. Para el mapa combinado, se utilizó el parámetro $\theta_{comDistance} = 20$.

6.2.2. Resultados

Para obtener los resultados se realizaron 10 partidos de cada tipo, de 10 minutos cada uno, donde se aplicaron las mismas reglas que en futbol robótico, y durante los cuales se calculó el parámetro OSPA para cada robot. Los resultados obtenidos se pueden observar en la Tabla 6.1, donde para cada prueba se obtuvieron los valores de error OSPA promedio, el mejor valor y el peor entre todos los robots.

Como se puede observar, los errores OSPA de los métodos de RFS-Mapa-Local y RFS-Mapa-Combinado son menores que para el método de MH-EKF. La reducción del error fue de un 17,29 % y de un 35,73 % para los métodos RFS-Mapa-Local y RFS-Mapa-Combinado, respectivamente, para la prueba de 5 vs. 5. Para el caso de los partidos de 1 vs. 1, la reducción del error obtenido por el método de RFS-Mapa-Local fue de 32,62 %.

6.3. Análisis de Resultados

Los resultados experimentales muestran de manera simple, que el método propuesto en esta tesis tiene una mejor respuesta al estimar los obstáculos presentes en la cancha que el MH-EKF. Principalmente, esta mejora se observa en la cantidad de obstáculos generados por cada método, donde en general el MH-EKF estima una mayor cantidad de falsos positivos. Esta diferencia se observa aún con mayor claridad cuando el problema aumenta en dinamismo, como cuando el robot está en movimiento, o cuando los obstáculos también se mueven. Estos errores se deben a la dificultad de asociar estimaciones con observaciones, lo que lleva a errores como los mostrados en la Figura 6.3, donde se generan varias estimaciones en la trayectoria

de un solo obstáculo real. Este tipo de errores son solucionables para el caso del MH-EKF, si se configura el parámetro de unión de estimaciones. Sin embargo, esta configuración puede llevar a otro tipo de errores, donde varios obstáculos reales sean unidos en una solo estimación solo por el hecho de estar cerca. Otra mejora tangible de observar es lo que se ve en la Prueba 3, donde al eliminar un obstáculo real de la cancha, le toma un tiempo parametrizable al método de MH-EKF eliminar la estimación. Esto se debe a que este método no toma en cuenta la información negativa, es decir, el no obtener una medición del lugar donde existe una estimación, por lo que mantiene esta estimación por tiempo parametrizable. En cambio, los metodos basados en RFS si consideran la información negativa, lo que permite, como se observa en al Prueba 4, eliminar de manera rápida estimaciones no existentes.

De la misma manera, al analizar los métodos de manera cualitativa en las pruebas simuladas, se puede observar una mejora sustancial, al comparar los mapas obtenidos con la medición OSPA. Además, como es previsible, este valor mejora aún más si los robots comparten sus mapas locales, ya que es posible observar conjuntamente mayor parte de la cancha. Así, se muestra que el método propuesto, RFS-Mapa-Compartido, logra combinar de manera acertada los mapas locales de cada robot, resolviendo de manera correcta el problema de asociación de datos.

Capítulo 7

Conclusión y Trabajo Futuro

7.1. Conclusión

El trabajo desarrollado en esta tesis muestra una novedosa aplicación de *tracking* basado en el enfoque de RFS, la cual se desarrolla en un ambiente altamente dinámico, donde tanto los robots que realizan el *tracking* como sus obstáculos están en constante movimiento, y donde todo el procesamiento se realiza en un computador de poca capacidad, en un tiempo promedio de 5[ms]. Además, este nuevo método supera a los métodos tradicionales de multi-*tracking* como el MH-EKF, dado que, como se pudo observar en las pruebas, no tiene los problemas clásicos de un *tracking* múltiple basado en *data association* lo que conlleva mejores resultados en la estimación final.

Por otro lado, se demostró que es posible utilizar de manera conjunta distintas fuentes de información, como son los sensores del los robots, o sus mapas locales compartidos, para así generar una mejor estimación de los obstáculos en la cancha. Cabe destacar que esta solución tuvo aún mejores resultados que su versión local.

7.2. Trabajo Futuro

A pesar de que esta tesis se desarrolló en el ambiente del futbol robótico, el planteamiento mostrado es extensible para cualquier tipo de robots móviles y de baja capacidad de procesamiento. De este modo, un trabajo interesante sería utilizar este método en otra plataforma, y así validar su funcionamiento más genéricamente.

Otras mejoras que se pueden introducir en este trabajo, consisten en incluir el mapa compartido como un sensor simulado más, y así procesar esa información con el algoritmo original del método de RFS multi sensor. La dificultad de este propuesta radica en que sería necesario estimar una probabilidad de detección basado en la información conocida de los robots compañeros (como su posición y la orientación de su cabeza). Adicionalmente añadir un sensor que en promedio obtendrá ~ 6 mediciones por ciclo, implica un gran aumento en el número de gaussianas, lo que incrementaría los tiempos de procesamiento. Es por esto que sería necesario optimizar el algoritmo para ser utilizado en los robots Naos durante un

partido.

Además de las mejoras mencionadas, un gran avance sería utilizar los nuevos métodos de estimación que utilizan los RFS, como el CPHD, el cual no solo estima los estados de los *tracks* sino que también el número de elementos presentes. O también en los filtros basados en Multi-Bernoulli, los cuales se ha demostrado en la literatura que tienen una mejor respuesta que el filtro PHD, así como un mayor gasto computacional.

Bibliografía

[1] P. Dames, P. Tokekar, and V. Kumar, "Detecting, Localizing, and Tracking an Unknown Number of Moving Targets Using a Team of Mobile Robots," pp. 513–529, 2015.

[2] D. Moratuwage, B. N. Vo, and D. Wang, "Collaborative Multi-vehicle SLAM with moving object tracking," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5702–5708, 2013.

[3] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, "Robocup: The robot world cup initiative," in *Proceedings of the first international conference on Autonomous agents*, pp. 340–347, ACM, 1997.

[4] T. Laue and T. Röfer, "Simrobot-development and applications," in *International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*, Citeseer, 2008.

[5] I. R. Goodman, R. Mahler, and H. T. Nguyen, *Mathematics of Data Fusion*. Dordrecht: Springer Netherlands, 1997.

[6] B.-N. Vo and W.-K. Ma, "The Gaussian Mixture Probability Hypothesis Density Filter," *IEEE Transactions on Signal Processing*, vol. 54, pp. 4091–4104, November 2006.

[7] Z. H. E. Chen, "Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond," *Statistics*, vol. 182, no. 1, pp. 1–69, 2003.

[8] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, vol. 64. Academic Press, 1970.

[9] H. Sorenson, "On the development of practical nonlinear filters," *Information Sciences*, vol. 7, pp. 253–270, January 1974.

[10] J. M. Bernardo and A. F. M. Smith, eds., *Bayesian Theory*. Wiley Series in Probability and Statistics, Hoboken, NJ, USA: John Wiley & Sons, Inc., May 1994.

[11] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[12] Y. Ho and R. Lee, "A Bayesian approach to problems in stochastic estimation and control," *IEEE Transactions on Automatic Control*, vol. 9, pp. 333–339, October 1964.

[13] P. Kalata, "The Tracking Index: A Generalized Parameter for α-β and α-β-γ Target Trackers," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-20, pp. 174–182, March 1984.

[14] S. Julier and J. Uhlmann, "Unscented Filtering and Nonlinear Estimation," *Proceedings of the IEEE*, vol. 92, pp. 401–422, March 2004.

[15] N. Gordon, D. Salmond, and a.F.M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proceedings F Radar and Signal Processing*, vol. 140, no. 2, p. 107, 1993.

[16] R. Douc and O. Cappe, "Comparison of resampling schemes for particle filtering," in *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, pp. 64–69, IEEE, September 2005.

[17] S. Blackman and R. Popoli, *Design and analysis of modern tracking systems*. Artech House radar library, Artech House, 1999.

[18] J. K. Uhlmann, "Algorithms for multiple-target tracking," *American Scientist*, vol. 80, no. 2, pp. 128–141, 1992.

[19] J. Uhlmann, "Introduction to the Algorithmics of Data Association in Multiple-Target Tracking," jun 2001.

[20] J. Uhlmann, "An Introduction to the Combinatorics of Optimal and Approximate Data Association," jun 2001.

[21] A. Poore, S. Lu, and B. Suchomel, "Data Association Using Multiple Frame Assignments," jun 2001.

[22] G. Pulford, "Taxonomy of multiple target tracking methods," *IEE Proceedings - Radar, Sonar and Navigation*, vol. 152, no. 5, p. 291, 2005.

[23] Y. Bar-Shalom and X.-R. Li, "Estimation and tracking- principles, techniques, and software," *Norwood, MA: Artech House, Inc, 1993.*, 1993.

[24] S. S. Blackman, "Multiple-target tracking with radar applications," *Dedham, MA, Artech House, Inc., 1986, 463 p.*, 1986.

[25] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, March 1955.

[26] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, pp. 32–38, March 1957.

[27] F. Bourgeois and J.-C. Lassalle, "An extension of the Munkres algorithm for the assignment problem to rectangular matrices," *Communications of the ACM*, vol. 14, pp. 802–804, December 1971.

[28] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, pp. 325–340, December 1987.

[29] D. P. Bertsekas, "The auction algorithm: A distributed relaxation method for the assignment problem," *Annals of Operations Research*, vol. 14, pp. 105–123, December 1988.

[30] D. P. Bertsekas, "The Auction Algorithm for Assignment and Other Network Flow Problems: A Tutorial," *Interfaces*, vol. 20, pp. 133–149, August 1990.

[31] Y. Bar-Shalom, *Tracking and Data Association*. San Diego, CA, USA: Academic Press Professional, Inc., 1987.

[32] Y. Bar-Shalom and E. Tse, "Tracking in a cluttered environment with probabilistic data association," *Automatica*, vol. 11, pp. 451–460, September 1975.

[33] Y. Bar-Shalom, P. K. Willett, and X. Tian, *Tracking and data fusion*. Storrs, CT, USA: YBS Publishing, 2011.

[34] T. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar tracking of multiple targets using joint probabilistic data association," *IEEE Journal of Oceanic Engineering*, vol. 8, pp. 173–184, July 1983.

[35] D. Musicki, R. Evans, and S. Stankovic, "Integrated probabilistic data association," *IEEE Transactions on Automatic Control*, vol. 39, pp. 1237–1241, June 1994.

[36] D. Musicki and R. Evans, "Joint integrated probabilistic data association: JIPDA," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, pp. 1093–1099, July 2004.

[37] D. Musicki and B. La Scala, "Multi-target tracking in clutter without measurement assignment," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 44, pp. 877–896, July 2008.

[38] D. Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, vol. 24, pp. 843–854, December 1979.

[39] I. Cox and S. Hingorani, "An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 2, pp. 138–150, 1996.

[40] B. Lu and P. Rosenbaum, "An algorithm for ranking all the assignments in order of increasing cost," *Journal of Computational and Graphical Statistics*, vol. 13, no. 2, pp. 422–434, 2004.

[41] G. Matheron, *Random sets and integral geometry*. New York: Wiley, 1975.

[42] G. Matheron, *An Introduction to the Theory of Point Processes*. New York: Springer, 1988.

[43] S. N. Chiu, D. Stoyan, W. S. Kendall, and J. Mecke, *Stochastic geometry and its applications*. John Wiley & Sons, 1995.

[44] R. Mahler, "Multitarget bayes filtering via first-order multitarget moments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, pp. 1152–1178, October 2003.

[45] R. Mahler, *Statistical Multisource-Multitarget Information Fusion*. Artech House, Inc., February 2007.

[46] R. Mahler, ""Statistics 101"for multisensor, multitarget data fusion," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, pp. 53–64, jan 2004.

[47] R. Mahler, ""Statistics 102" for Multisource-Multitarget Detection and Tracking," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, pp. 376–389, jun 2013.

[48] B.-T. Vo, *Random Finite Sets in Multi-Object Filtering*. Phd thesis, The University of Western Australia, 2008.

[49] B.-N. Vo, S. Singh, and A. Boucet, "Sequential monte carlo methods for multi-target filtering with random finite sets," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, pp. 1224–1245, oct 2005.

[50] R. Mahler, "PHD filters of higher order in target number," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, pp. 1523–1543, oct 2007.

[51] D. Daley and D. Vere-Jones, *An Introduction to the Theory of Point Processes*. Probability and its Applications, New York: Springer-Verlag, 2003.

[52] S. J. Julier and J. K. Uhlmann, "A New extension of the Kalman filter to nonlinear systems," *Proc. SPIE*, vol. 3068, p. 182, jul 1997.

[53] R. Mahler, "The multisensor PHD filter, II: Erroneous solution via "Poisson magic"," in *Proceedings of SPIE* (I. Kadar, ed.), vol. 7336, pp. 73360D–73360D–12, may 2009.

[54] R. Mahler, "The multisensor PHD filter, I: General solution via multitarget calculus," in *Proceedings of SPIE* (I. Kadar, ed.), vol. 7336, pp. 73360E–73360E–12, may 2009.

[55] D. Schuhmacher, B.-T. Vo, and B.-n. Vo, "A Consistent Metric for Performance Evaluation of Multi-Object Filters," *IEEE Transactions on Signal Processing*, vol. 56, pp. 3447–3457, aug 2008.

[56] G. Jochmann, S. Kerner, S. Tasse, and O. Urbann, *Efficient Multi-hypotheses Unscented Kalman Filtering for Robust Localization*, pp. 222–233. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[57] T. Laue and T. Röfer, "Integrating Simple Unreliable Perceptions for Accurate Robot Modeling in the Four-Legged League," in *RoboCup 2006: Robot Soccer World Cup X. 10th RoboCup International Symposium, June 19-20, Bremen, Germany* (G. Lakemeyer,

E. Sklar, D. G. Sorrenti, and T. Takahashi, eds.), no. 4434, ch. Integratin, pp. 474–482, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[58] M. Adams, B.-N. Vo, R. Mahler, and J. Mullane, "SLAM Gets a PHD: New Concepts in Map Estimation," *IEEE Robotics & Automation Magazine*, vol. 21, pp. 26–37, jun 2014.

[59] P. Cano and J. Ruiz-del Solar, "Robust Tracking of Multiple Soccer Robots Using Random Finite Sets," No. July, pp. 206–217, 2017.

[60] P. Cano and J. Ruiz-del Solar, "Robust Tracking of Soccer Robots Using Random Finite Sets," *IEEE Intelligent Systems*, vol. 32, pp. 22–29, nov 2017.

Anexos

Robust Tracking of Multiple Soccer Robot using Random Finite Sets

Este paper fue presentado en la conferencia de la competencia Robocup 2016, donde ademas en el equipo de esta universidad, donde se implementó parte de lo mostrado en esta tesis, se obtuvo el cuarto lugar.

Robust Tracking of Multiple Soccer Robots using Random Finite Sets

Pablo Cano and Javier Ruiz-del-Solar

Advanced Mining Technology Center & Dept. of Elect. Eng., Universidad de Chile
{pcano,jruizd}@ing.uchile.cl

Abstract. Having a good estimation of the robot-players positions is becoming imperative to accomplish high level tasks in any RoboCup League. Classical approaches use a vector representation of the robot positions and Bayesian filters to propagate them over time. However, these approaches have data association problems in real game situations. In order to tackle this issue, this paper presents a new method for building robot maps using Random Finite Sets (RFS). The method is applied to the problem of estimating the position of the teammates and opponents in the SPL league. Considering the computational capabilities of Nao robots, the GM-PHD implementation of RFS is used. In this implementation, the estimations of the robot positions and the robot observations are represented using Mixture of Gaussians, but instead of associating a robot or an observation to a given Gaussian, the weight of each Gaussian maintains an estimation of the number of robots that it represents. The proposed method is validated in several real game situations and compared with a classical EKF based approach. The proposed GM-PHD method shows a much better performance, being able to deal with most of the data association problems, even being able to manage complex situations such as robot kidnappings.

Keywords: World Modeling, Multi-target tracking, Robot position estimation, Random Finite Sets

1 Introduction

As RoboCup progresses over the years, high-level skills become necessary to maintain a competitive level. Such skills are no longer restricted to the detection of field objects or to the self-localization of the robot players, but include team's skills based on the tracking (position estimation) of teammates and opponents. Examples of these skills are ball passing, adversaries' tracking and team's formation.

When the observability of the game and the players is not an issue to address, as in the case of the Small-size league, high- levels skills based on the tracking of the robot players have been already implemented (e.g. reactive coordination [1], and the analysis and learning of the opponent's strategies [2][3]).

The situation in the Standard Platform League (SPL) is more complex given the

restricted field of view of the robot´s camera and the low computational resources of the Nao robots; in this league the detection of other robots is not robust and the construction of a good map of obstacles/players is a difficult process. Current standard robot tracking systems maintain/update the robot estimations using Kalman filters (e.g. [4][5]). However, in this tracking paradigm there is no clear solution of the data association problem, and several heuristics need to be used in order to eliminate and merge hypothesis.

In this context the main goal of this paper is to propose a new methodology for the robust tracking (position estimation) of multiple soccer robots using the Random Finite Sets (RFS) framework, which allows to overcome the drawbacks of current approaches. The proposed methodology is inspired in [6] where the term Probability Hypothesis Density (PHD) was introduced as the first moment of a point process. Then, the PHD filter is presented in [7] as a way to maintain hypothesis of multiple objects using sets instead of vectors.

There are several works that use this new framework in the literature, concerning all kind of problems and subjects [8]. Principally, it is used in highly complex environments to track large amount of features, which makes it very expensive computationally. But, in the SPL problem the number of features (robots) to detect rise to 10 in the worst case, which make it computationally tractable for a Nao robot's CPU.

The paper is organized as follows: in Section 2 the problem to resolve is described. Section 3 presents a brief introduction to RFS. Section 4 shows the implementation used in this work, and Section 5 the experimental results. Finally, conclusions are drawn in Section 6.

2 Problem Description: Data Association when Tracking Multiple Players in Robot Soccer

As already mentioned, knowing the position of the other robots in the field is relevant for implementing high-level soccer behaviors. In this work we will denote *map of obstacles* to a map that a given player builds, and which includes the positions on the field of every other robot player, teammate or opponent (see Figure 1). We will denote observations to the detections of these robots, and obstacles to the estimated position of these robots in the map.

Most of the existing methods used for estimating the map of obstacles employ a classical approach with a vector representation of the obstacles (robots), which are propagated over time using a Bayesian filter (e.g. an EKF filter). However, it has been demonstrated that the use of a vector representation of the obstacles has numerous drawbacks, mainly related to the data association between new and past observations (obstacles) [9]. Some examples of those problems are illustrated in Figure 2: Fig. 2(a) shows a trivial case where the data association between two new observations (red crosses) and two obstacles (black crosses) is trivial. However, the data associations is not trivial in the cases illustrated en Figs. 2(b) and 2(c). First, Fig. 2(b) shows the case when two new measurements have a similar distance to the obstacle, in addition to be not very close to the obstacle (see the covariance of the obstacle representation). So, depending of the implemented data association strategy, this could end in one, two or

three obstacles in the map. Fig. 2(c) show a case where two obstacles are very close, so the new measurement could be associated with any of them, and leave the other with no update for that frame. For these cases, most of methods use heuristics to associate new measurements to the obstacles, or to create new obstacles if no association is made. But, in a highly dynamic environment as a robot soccer match, these methods may produce several bad associations or missed detections.

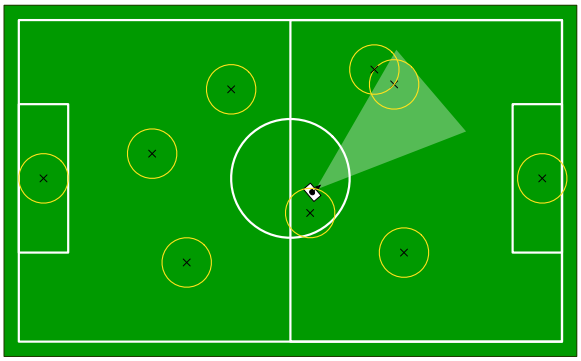


Fig. 1. Example of a map of obstacles. The white rectangle represents the robot which is building the map. The black crosses represent the robots/obstacles positions and the yellow circles the corresponding covariance of each representation. The lighted zone represents the Field of View of the camera.

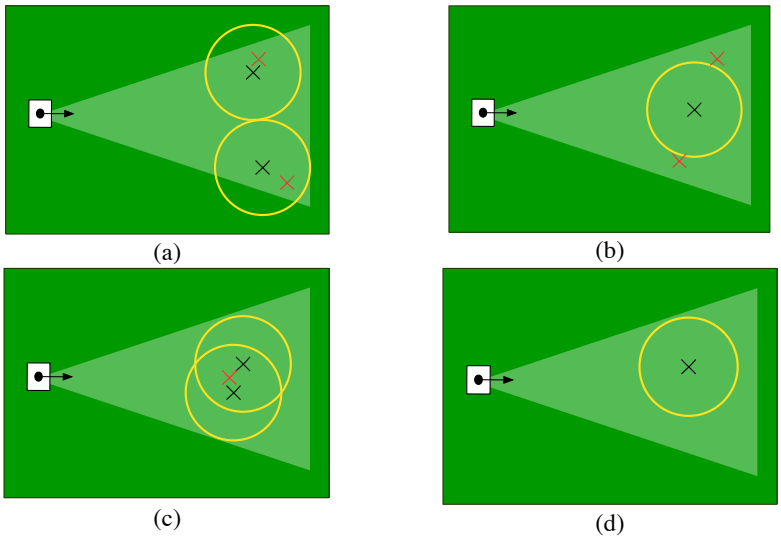


Fig. 2. The red crosses represent measurements of the sensor, black crosses represent the robots/obstacles positions and the yellow circles the corresponding covariance of each

representation. (a) represents an easy case of data association, while (b) and (c) show more complex cases. (d) represents a case when an obstacle that should be detected by the robot is not sensed.

Finally, Fig. 2 (d) describes a situation where no measurement is obtained for an obstacle inside the Field of View (FoV). For the classical approach, this is not different from an obstacle outside the FoV, and its only consequence is that the obstacles' covariance grows. So, depending of the speed of the covariance's growing (which maintain the obstacles outside the FoV), the obstacles inside the FoV will be maintained the same time that the others, although they do not receive any measurements.

3 Multi-target tracking with Random Finite Sets

The main idea of the proposed methodology is to use *finite sets* instead of vectors for representing both observations and obstacles, which can encapsulate positions and quantity uncertainty. As has been widely demonstrated [7][10][9], the first moment of RFS, know as Probability Hypothesis Density (PHD), can be used to construct a filter which propagates the PHD of the map posterior instead of the map posterior itself.

3.1 PHD Filter

The PHD function v at a point represents the density of the expected number of obstacles occurring at that point of the state space (map). Therefore, a property of the PHD is that for any given region S of the map

$$\mathbb{E}[|M \cap S|] = \int_S v(m)dm \tag{1}$$

where M represents the map RFS and $|\cdot|$ denotes the cardinality of a set. This means that, by integrating the PHD on any region S of the map, we obtain the expected number of obstacles in S [7].

The PHD filter considers the following two steps [7]:

- Prediction:

$$v_{k|k-1}(m) = v_{k-1}(m) + b_k(m) \tag{2}$$

where $b_k(m)$ represents the PHD of the new obstacles in time k.

- Update:

$$v_k(m) = v_{k|k-1}(m)\left[1 - P_D(m) + \sum_{z \in Z_k} \frac{P_D(m)g_k(z|m))}{c_k(z) + \int P_D(\xi)g_k(z|\xi)v_{k|k-1}(\xi)d\xi}\right] \tag{3}$$

where $\mathrm{P_D}(m)$ represents the probability of detecting an obstacle at m, $g_k(z|m)$ represents the likelihood that z is generated by an obstacle in m at time k (i.e. the measurement likelihood) and $c_k(z)$ is the clutter intensity at time k.

3.2 Considerations

To adopt this framework to the presented problem, some considerations must be done. As presented before, $P_D(m)$ represents the probability of detecting an obstacle at m, but it does not take into account the capability of the robot to sense at m. So the real probability is represented by $P_D(m|X_k)$ where X_k represent the position of the robot in time k. The same occurs with g_k, c_k and b_k, because they also depend of the robot position.

4 Implementation

There are many implementations of RFS in the literature, but most of them are time consuming, especially for Nao robots with limited computational capabilities [11][12]. Hence we use the Mixture of Gaussian implementation (GM-PHD) [13], because it is very time efficient and it allows to get easily the positions of the obstacles from the PHD.

4.1 GM-PHD implementation

The main idea is to represent any RFS as a mixture of Gaussians. Therefore, both obstacles and detections are represented by Gaussians. But, to represent the position and number uncertainty of the obstacles present in the field, it is necessary to add a weight to every Gaussian. In this way their positions represent the multitude of location of obstacles in the map while their weights represent the number of obstacles in that given region. So, a PHD map is a Gaussian Mixture of the form,

$$v_{k-1}(m|X_{k-1}) = \sum_{j=1}^{J_{k-1}} \omega_{k-1}^{(j)} \mathcal{N}\left(m; \mu_{k-1}^{(j)}, P_{k-1}^{(j)}\right) \tag{4}$$

which is a mixture of J_{k-1} Gaussians, with $\omega_{k-1}^{(j)}, \mu_{k-1}^{(j)}$ and $P_{k-1}^{(j)}$ being their corresponding prior weights, means and covariances, respectively. The same form is used to represent the new obstacles at time k, $b_k(m|Z_{k-1}, X_{k-1})$, as

$$b_k(m|Z_{k-1}, X_{k-1}) = \sum_{j-1}^{J_{b,k}} \omega_{k|k-1}^{(j)} \mathcal{N}\left(m; \mu_{k|k-1}^{(j)}, P_{k|k-1}^{(j)}\right) \tag{5}$$

where $J_{b,k}$ is the number of Gaussians in the new PHD at time k, Z_{k-1} is the vector of measurements at time $k-1$ and $\omega_{k|k-1}^{(j)}, \mu_{k|k-1}^{(j)}$ and $P_{k|k-1}^{(j)}$ determine the shape of the PHD of new obstacles. Therefore, the predicted PHD of the map, shown in (2) is also a Gaussian mixture

$$v_{k|k-1}(m|X_k) = \sum_{j=1}^{J_{k|k-1}} \omega_{k|k-1}^{(j)} \mathcal{N}\left(m; \mu_{k|k-1}^{(j)}, P_{k|k-1}^{(j)}\right) \tag{6}$$

where $J_{k|k-1} = J_{k-1} + J_{b,k}$ are the number of Gaussians representing the union of the

prior map PHD $v_{k-1}(m|X_{k-1})$, and the new obstacles PHD at time k. $\omega_{k|k-1}^{(j)}, \mu_{k|k-1}^{(j)}$ and $P_{k|k-1}^{(j)}$ represents the shape and form of the Gaussians of the prior map PHD if $j < J_{k-1}$ and the shape and form of the Gaussians of the new observations PHD otherwise.

So, the posterior PHD shown in (3) is also a Gaussian mixture of the form

$$v_k(m|X_k) = v_{k|k-1}(m|X_k)\left[1 - P_D(m|X_k) + \sum_{z \in Z_k} \sum_{j=1}^{J_{k|k-1}} v_{G,k}^{(j)}(z, m|X_k)\right] \tag{7}$$

where $v_{G,k}^{(j)}$ corresponds, according to the general PHD Filter update equation, to

$$v_{G,k}^{(j)}(z, m|X_k) = \omega_k^{(j)}(z|X_k)\mathcal{N}\left(m; \mu_{k|k}^{(j)}, P_{k|k}^{(j)}\right) \tag{8}$$

$$\omega_k^{(j)}(z|X_k) = \frac{P_D(m|X_k)\omega_{k|k-1}^{(j)} q^{(j)}(z|X_k)}{c_k(z) + \sum_{i=1}^{J_{k|k-1}} P_D(m|X_k)\, \omega_{k|k-1}^{(i)} q^{(i)}(z|X_k)} \tag{9}$$

where $q^{(i)}(z|X_k) = \mathcal{N}\left(z; H_k \mu_{k|k-1}^{(i)}, S_k\right)$ is the measurement likelihood. The components $\mu_{k|k}^{(i)}$ and $P_{k|k}^{(i)}$ can be obtained from the standard EKF update equations,

$$S_k^{(i)} = R_k + \nabla H_k P_{k|k}^{(i)} \nabla H_k^T \tag{10}$$

$$K_k^{(i)} = P_{k|k}^{(i)} \nabla H_k^T \left[S_k^{(i)}\right]^{-1} \tag{11}$$

$$\mu_{k|k}^{(i)} = \mu_{k|k-1}^{(i)} + K_k^{(i)}\left(z - H_k\left(\mu_{k|k-1}^{(i)}\right)\right) \tag{12}$$

$$P_{k|k}^{(i)} = \left[I - K_k^{(i)} \nabla H_k\right] P_{k|k-1}^{(i)} \tag{13}$$

with ∇H_k being the Jacobian of the measurement equation with respect to the obstacles estimated location.

4.2 Algorithm

In order to use the presented framework, it is necessary to create Gaussians according to the measurements of the robot's sensors. Therefore $b_k(m|Z_{k-1}, X_{k-1})$ is obtained from the measurements Z_{k-1} and the previous robot position X_{k-1}. The components of this Gaussians are determined according to

$$\omega_{b,k}^{(j)} = 0.01, \qquad \mu_{b,k}^{(j)} = h^{-1}(z_{k-1}^j, X_{k-1}),$$
$$P_{b,k}^{(j)} = h'(\mu^{(j)}, X_{k-1}) R \left[h'(\mu^{(j)}, X_{k-1})\right]^T$$

where h^{-1} is the inverse measurement equation, R is the measurement noise covariance and $h'(\mu^{(j)}, X_{k-1})$ is the Jacobian of the measurement model function with respect to the Gaussian state, j. Therefore, the implementation initially considers all detections at time $k-1$ to be potential new features at time k.

Then, as every Gaussian is combined with every measurement to generate a new Gaussian, the numbers of Gaussians grow exponentially in every frame. That is why pruning and merging operations are necessary. Gaussians which are determined

sufficiently close (through a Mahalanobis distance threshold) are merged into a single Gaussian. But this does not represent an elimination of an obstacle because one Gaussian can represent more than one obstacle by its weight; these values are added when two or more Gaussians are merged.

Figure 3 shows the pseudo code of the complete algorithm.

//prediction step
// the parameters of the Map's MoG model ($v_{k-1}(m|X_{k-1})$) are modified
for $i = 1$ to J_{k-1} **do**
  //the obstacles may move -> covariance is increased
  $\mu_{k|k-1}^{(j)} = \mu_{k-1}^{(j)}, P_{k|k-1}^{(j)} = P_{k-1}^{(j)} + Q, \omega_{k|k-1}^{(j)} = \omega_{k-1}^{(j)}$
end for
//birth; new obstacles are added
generateNewGausians(Z_{k-1}, X_{k-1}) // equation (5)
$v_{k|k-1}(m|X_k) = \left\{\mu_{k|k-1}^{(i)}, P_{k|k-1}^{(i)}, \omega_{k|k-1}^{(i)}\right\}_{i=1}^{J_{k|k-1}}$
//update step
for $i = 1$ to $J_{k|k-1}$ **do**
  calculate $P_D^{(i)}$
  $\omega_k^{(i)} = \left(1 - P_k^{(i)}\right)\omega_{k|k-1}^{(i)}$
end for
$N = 1$
for each z in Z_k
  for $i = 1$ to $J_{k|k-1}$ **do**
    calculate $H, S_k^{(i)}$ and $K_k^{(i)}$
    $\mu_k^{(N+i)} = \mu_{k|k-1}^{(i)} + K_k^{(i)}\left(z - \mu_{k|k-1}^{(i)}\right)$
    $P_k^{(N+i)} = \left[I - K_k^{(i)} H\right]P_{k|k-1}^{(i)}$
    $\tau^{(i)} = P_D^{(i)}\omega_{k|k-1}^{(i)}\left|2\pi S_k^{(i)}\right|^{-0.5} \times \exp\left(\left(z - \mu_{k|k-1}^{(i)}\right)\left[S_k^{(i)}\right]^{-1}\left(z - \mu_{k|k-1}^{(i)}\right)^T\right)$
  end for
  for $i = 1$ to $J_{k|k-1}$ **do**
    $\omega_k^{(N+i)} = \tau^{(i)} / \left(c(z) + \sum_{l=1}^{J_{k|k-1}} \tau^{(l)}\right)$
  end for
  $N = N + J_{k|k-1}$
end for
$J_k = N$
//updated map
$v_k(m|X_k) = \left\{\mu_k^{(i)}, P_k^{(i)}, \omega_k^{(i)}\right\}_{i=1}^{J_k}$
prune $(v_k(m|X_k))$

Fig. 3. Pseudo code of the general algorithm that calculates the PHD that represent the map of

obstacles.

With this algorithm, the PHD of the map is obtained. Then to get the position of the obstacles in the map, it is necessary to evaluate every Gaussian's weight. If this values exceeds a given threshold, then the obstacle position is given by the Gaussian's mean vector. Figure 4 shows this algorithm.

$M_k = \emptyset$
for $i = 1$ to J_k **do**
  if $\omega_k^{(i)} > thrld$ **then**
    $M_k = \left[M_k \; \mu_k^{(i)}\right]$
  end if
end for

Fig. 4. Pseudo code of the algorithm that drawn obstacles according to the PHD of the map.

4.3 Application

Using the proposed methodology it is obtained a representation of the obstacle map for the detection of soccer players (Nao robots) in the SPL league. To do this, the methodology is used as follows:

i. State space: in order to describe the obstacles in the field, the state space is a vector $p = (x, y)$ that represents positions on the field according to the center of the field as $(0,0)$ of the coordinate system.
ii. Sensor: the used sensor is the Nao camera. This implies that transformations must be done in order to describe the measurements as positions on the field, using the camera's intrinsic and extrinsic parameters, as well as the position of the robot on the field. The detections are made with the same robot´s detector provided in the B-Human Code Release 2014 [14]
iii. Probability of detection P_D: Given that the sensor is the camera of the robot, the probability of detection is given by the field of view of it and the position of the obstacle relative to the robot. This implies that P_D must be recalculated in every frame for all the Gaussians of the map.
iv. Moving obstacles: The movement of the obstacles is taken into account by growing the covariance of the mixture of Gaussians in every frame instead of adding a movement model into the prediction step.

5 Results

In order to evaluate the proposed methodology several experiments with real Nao robots in a real SPL field were carried out. Given that we needed to measure the accuracy of the obstacle's map (i.e. robots map), a validation system consisting in a

global vision system (camera over the field) for measuring the Ground Truth was implemented.

First, we carried out a very simple experiment in which a robot is placed in the center of the field and it observes three other static robots. The robot is moving its head all the time, and given its reduced field of view, at a given moment it is able to observe just one of the other robots and in some few cases two. The proposed GM-PHD based method is compared with a classical EKF based method. As expected, given the simplicity of the problem, both systems obtained an average error of about 20 cm in the position of the robots. Both methods run in real time, being the processing time of the GM-PHD method 0.13 ms, and the processing time of the EKF method 0.07 ms.

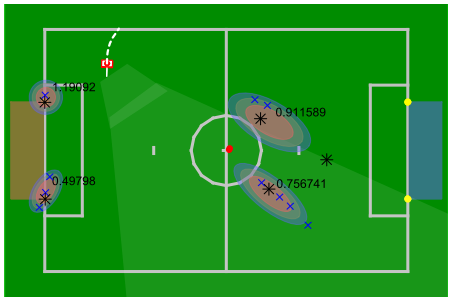
Secondly, the proposed GM-PHD based method and the classical EKF based method were compared under more realistic, dynamic conditions, where the observer robot, i.e. the one that builds the map, moves as well as some of the observed robots. Figure 6 shows this set of experiments.

For the first experiment, five static robots are placed on the field, and the observer robot performs a ready positioning, i.e. the robot walks from its starting position to their legal kick-off position. The observer robot moves its head from left to right all the time, hence the other robots are not inside the FoV in every frame. As can be seen in Fig. 5(a), the differences of the GM-PHD method and the classical EKF method, in term of a multi-tracking criteria, are notorious. While the GM-PHD approach correctly describes the presence of obstacles in most of the positions of the field, the classical one shows an incorrect number of obstacles for each real one. This is because the new observations are not correctly associated with the previous ones, due to the odometry errors and the non constant observations; then new hypothesis are drawn incorrectly by the EKF method.

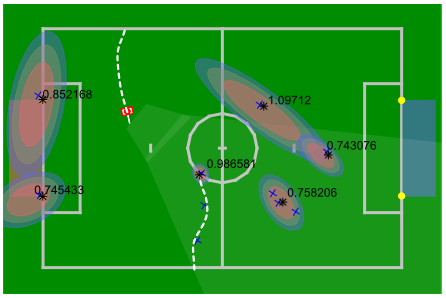
In the second experiment one moving robot observes five other robots; one moving robot and four static ones. The observer robot, while moves, observes the other moving robot occasionally, because it moves its head from left to right all the time. In Fig 5(b) it can be seen that, when using the classical EKF approach, there are two wrongly detected robots placed in the previous path of the moving robot, in addition to the same error that occurs in the last experiment when more than one obstacle in the map is describing each real one. The GM-PHD method correctly relates these observations with the same obstacle. In fact, the GM-PHD method perfectly estimates the number of robots in the field. In the case of the EKF method, bad associations can be corrected by increasing the minimal distance of merging. But this can produce another type of errors, where detections from different robots are associated to the same one.

In the third experiment we analyze a typical kidnapping situation. The observer robot is placed in the center of the field and three static robot are placed in other field positions. The observer robot is looking around when one of the static robots is removed from the field (in a real match, this is very common due to robot penalizations). As can be seen in Fig. 5(c) and Fig. 5(d), the GM-PHD approach deletes very quickly the hypothesis associated with the kidnapped robot, while the classical EKF method keeps the track until the covariance reaches a given threshold value. This can be fixed for the classical EKF method by calculating a different rate of covariance growing when a hypothesis that should be seen is not seen. But, this

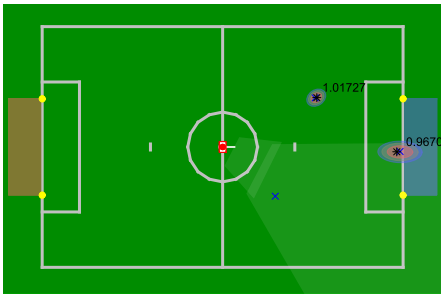
implies including another heuristic to the process, while the GM-PHD method handles this situation naturally.



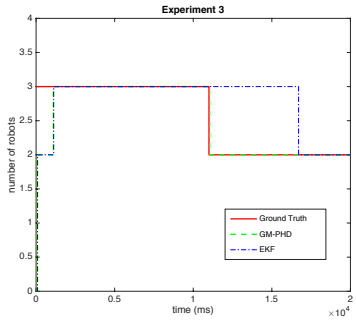
(a)



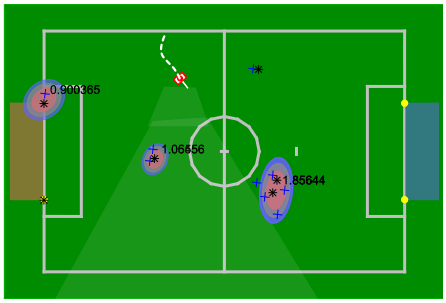
(b)



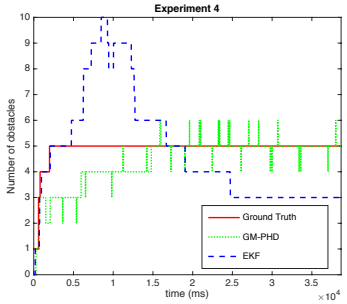
(c)



(d)



(e)



(f)

Fig. 5. Map building experiments under dynamic conditions. Four different situations are described in (a), (b), (c) and (e). In these diagrams the black asterisks represent the real position of the robots, obtained by the Ground Truth system; Blue crosses represent the robots' positions calculated by a EKF tracking method; The colored ellipses represent the robot estimations of the GM-PHD based method, and the associated number represents the weight of each Gaussian. The white dashed lines represent the trajectory of moving robots. (d)/(f) shows the estimated number of robots corresponding to situation (c)/(e).

Finally, in the last experiment the observer robot also realizes a ready positioning while there are some static robots placed in the field. But two of these robots are very close from each other, therefore the perception of these robots is very inaccurate. In Fig. 5(e) it can be seen that even when only one Gaussian is representing these robots, the GM-PHD method can correctly estimate the number of robots in that place (given by the weight of the Gaussian), while the classical EKF approach fails due the odometry and perception errors. In Fig. 5(f) the estimated number of robots given by each method thought the entire experiment is shown. It should be remembered that the estimated number of robots is calculated as the sum of the weight of all Gaussians by the GM-PHD method, and as the number of obstacles created by the classical method.

6 Conclusions

This paper presents a new method for building obstacle maps using a consistent mathematically approach, known as Random Finite Sets. The method is applied to the problem of estimating the position of the robots, teammates and opponents, in the SPL league. Considering the computational capabilities of Nao robots, the GM-PHD implementation is used. In this implementation, obstacles and observations are represented using Mixture of Gaussians, but instead of associating an obstacle or an observation to a given Gaussian, the weight of each Gaussians maintains an estimation of the number of robots that it represents.

The proposed tracking method was validated in several real game situations, with moving robots, and compared with a classical EKF based approach. The proposed GM-PHD method showed a much better performance, being able to deal with most of the data association problems, even being able to manage complex situations such a robot kidnappings. Moreover, the method is able to run in real-time in the Nao robots (mean processing time is 0.13 ms; worse case processing time 0.3 ms).

Acknowledgments

The authors thank Constanza Villegas for her contributions to the development of this publication and the UChile Robotics Team for their general support. We also thank the B-Human SPL Team for sharing their code release, contributing the development of the Standard Platform League. This work was partially funded by FONDECYT Project 1161500.

References

1. Mendoza, J.P., Biswas, J., Cooksey, P., Wang, R., Klee, S., Zhu, D., Veloso, M.: Selectively Reactive Coordination for a Team of Robot Soccer Champions. In: Proceedings of AAAI-16 (2016).
2. Trevizan, F.W.F., Veloso, M.M.M.: Learning Opponent's Strategies In the RoboCup Small Size League. In: Proceedings of AAMAS 2010 Workshop on Agents in Real-time and Dynamic Environments. pp. 45–52. , Toronto (2010).
3. Yasui, K., Kobayashi, K., Murakami, K., Naruse, T.: Analyzing and Learning an Opponent's Strategies in the RoboCup Small Size League. In: Behnke, S., Veloso, M., Visser, A., and Xiong, R. (eds.) RoboCup 2013: Robot World Cup XVII. pp. 159–170. Springer Berlin Heidelberg, Berlin, Heidelberg (2014).
4. Laue, T., Röfer, T.: Integrating Simple Unreliable Perceptions for Accurate Robot Modeling in the Four-Legged League. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., and Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X. pp. 474–482. Springer Berlin Heidelberg, Berlin, Heidelberg (2007).
5. Fabisch, A., Laue, T., Röfer, T.: Robot Recognition and Modeling in the RoboCup Standard Platform League. In: Pagello, E., Zhou, C., Behnke, S., Menegatti, E., Röfer, T., and Stone, P. (eds.) Proceedings of the Fifth Workshop on Humanoid Soccer Robots in conjunction with the 2010 IEEE-RAS International Conference on Humanoid Robots. pp. 65–70. , Nashville, TN, USA (2010).
6. Goodman, I.R., Mahler, R.P.S., Nguyen, H.T.: Mathematics of Data Fusion. Springer Netherlands, Dordrecht (1997).
7. Mahler, R.P.S.: A Theoretical Foundation for the Stein-Winter "Probability Hypothesis Density (PHD)" Multitarget Tracking Approach. In: Sensor and Data Fusion (2000).
8. Mahler, R.: A brief survey of advances in random-set fusion. In: 2015 International Conference on Control, Automation and Information Sciences (ICCAIS). pp. 62–67. IEEE (2015).
9. Mahler, R.P.S.: Statistical Multisource-Multitarget Information Fusion. (2007).
10. Mahler, R.P.S.: Multitarget Bayes Filtering via First-Order Multitarget Moments. IEEE Trans. Aerosp. Electron. Syst. 39, 1152–1178 (2003).
11. Vo, B.N., Singh, S., Doucet, A.: Sequential Monte Carlo methods for multi-target filtering with random finite sets. In: IEEE Transactions on Aerospace and Electronic Systems. pp. 1224–1245 (2005).
12. Vo, B.-N., Ma, W.-K.: The Gaussian Mixture Probability Hypothesis Density Filter. IEEE Trans. Signal Process. 54, 4091–4104 (2006).
13. Random Finite Sets for Robot Mapping & SLAM - New | John Stephen Mullane | Springer, http://www.springer.com/gp/book/9783642213892.
14. Thomas, R., Laue, T., Judith, M., Bartsch, M., Batram, M.J., Arne, B., Martin, B., Kroker, M., Maaß, F., Thomas, M., Steinbeck, M., Stolpmann, A., Taddiken, S.: Team Report and Code Release 2013. 1–194 (2014).

Robust Tracking of Soccer Robots Using Random Finite Sets

Este paper fue presentado en un *special issue* de la revista IEEE Intelligent Systems.

Robust Tracking of Soccer Robots Using Random Finite Sets

Pablo Cano and Javier Ruiz-del-Solar, *Universidad de Chile*

Maintaining a good estimation of the other robots' positions is crucial in soccer robotics. To tackle the data association problem, the proposed approach doesn't require explicit data association and integrates information shared by teammate robots.

Multirobot systems correspond to groups of intelligent robots that perceive and act in a given environment to collectively solve a task. The robots can use different cooperation and coordination mechanisms, which can include cooperative perception, distributed world modeling, planning and control, and distributed decision making, among others. There are many application areas of multirobot systems such as surveillance, exploration, warehouse management, transportation, rescue applications, and soccer robotics, just to name a few.

Soccer robotics corresponds to a very well-known benchmark of multirobot systems, in which two teams of autonomous robots play soccer against each other. In the most advanced soccer leagues (Standard Platform League [SPL] and Humanoid League; www.robocup.org/robocup-soccer), each robot player perceives the environment individually, shares its observations with its teammates, models the environment, which includes the modeling of dynamic objects such as the ball and the other robot players, makes decisions in a centralized or distributed fashion, and acts (that is, it plays soccer). The focus of our research is on soccer robotics, specifically, the robust tracking of robot players by a robot. This is an interesting tracking problem in which both teammates and opponents need to be tracked. Teammates normally share their positions and their estimations of the other robots' positions, but opponents don't share any information.

When playing soccer, one of the main challenges is maintaining a good estimation of opponent and teammate positions to select appropriate behaviors. High-level behaviors such as passing, team formation, and role decisions, just to name a few, need information about players' positions on the field. This problem is particularly complex in leagues that use humanoid robots such as

1541-1672/17/$33.00 © 2017 IEEE
Published by the IEEE Computer Society

the SPL and the Humanoid League: the number of robots in the field changes constantly (robots can be removed from the field due to penalizations or failures), robots of the same team are indistinguishable, the small field of view of the cameras allows only a restricted or partial observation of the field, and the robots have limited computational capabilities. These limitations can be overcome by sharing observations among teammates and by a robust estimation of the robots' positions. The estimation of those positions, which is denoted as robot tracking, has been tackled in a variety of ways within the robot soccer community. Bayesian filtering approaches such as multiple hypothesis tracking (MHT),[1] extended Kalman filters (EKFs),[2] the multihypothesis unscented Kalman filter,[3] and Gaussian mixture models[4] have been proposed to address this problem. However, all these approaches involve an explicit association between measurements and targets, using heuristics or statistics. This step is commonly known as the data association problem.

In this context, the main goal of this article is to propose a new methodology for the robust tracking (position estimation) of multiple soccer robots using the Random Finite Sets (RFS) framework, which doesn't require explicit data association. The proposed methodology is inspired by another work[5] where the term probability hypothesis density (PHD) was introduced as the first moment of a point process. The PHD filter is proposed as a way to maintain hypotheses of multiple objects using finite sets instead of vectors to describe object states.[6] In fact, this paradigm considers the unknown number of objects to be tracked (robots, in our case) as a multi-object set represented by an object and the measurements received by the sensor as a single set of observations, which it models as RFS. RFS is a set of random variables, for which the cardinality is itself a random variable.

The RFS framework has been used to track features in SLAM (Simultaneous Localization and Mapping) applications as well as to track multiple objects and targets.[7,8] However, this is the first application of RFS in soccer robotics, and one of the first that addresses the tracking of robots in a dynamic environment, along with other works[9,10] that track moving objects. The main contributions here are the application of the RFS framework to the tracking of multiple robots in a highly dynamic environment and the incorporation in the tracking process of information shared by teammates robots.

The proposed robot tracking methodology is validated in simulated robot soccer games, using simulated Nao robots, and compared against a classical, state-of-the-art multihypothesis EKF tracking methodology.[3] The specific application of the proposed tracking methodology is soccer robotics, but it can be adapted to track robots in other applications and environments.

Problem Description: Data Association when Tracking Multiple Players in Robot Soccer

As already mentioned, knowing the position of the other robots in the field is relevant for implementing high-level soccer behaviors. In this work, we will use *local map of robots* for a map that a given player builds and that includes the positions on the field of every other robot player (teammate or opponent). We denote observations of robot detections and hypotheses for the estimated positions of the robots on that map.

Most existing methods for estimating the local map of robots employ a classical approach with a vector representation of the robot hypotheses, which are propagated over time using a Bayesian filter (such as an EKF filter). However, it has been demonstrated that the use of a vector representation of hypotheses has numerous drawbacks mainly related to the data association between new and past observations (hypotheses).[11]

Multitarget Tracking with RFS

The main idea of the proposed methodology is to use finite sets instead of vectors to represent both observations and hypotheses. As has been widely demonstrated,[6,11] the first moment of RFS, known as probability hypothesis density (PHD), can be used to construct a filter that propagates PHD to the map posterior instead of the map posterior itself.

PHD Filter

The PHD function $v(m)$ at position m represents the density of the expected number of objects (robots) occurring at that position of the state space (map). Therefore, a property of the PHD is that for any given region S of the map:

$$\mathbb{E}\left[\left|M \cap S\right|\right] = \int_S v(m)dm, \tag{1}$$

where M represents the RFS map and $|\cdot|$ denotes a set's cardinality. This means that, by integrating the PHD on any region S of the map, we obtain the expected number of objects in S.[6]

The PHD filter considers the following two steps:[6]

- Prediction:

$$v_{k|k-1}(m) = v_{k-1}(m)\, b_k(m), \tag{2}$$

where $v_{k-1}(m)$ is the previous estimate of the PHD, $v_{k-1}(m)$ is its prediction at time k, and $b_k(m)$ represents the PHD of the new objects

at time k. This equation is a simplification of the prediction step shown elsewhere,[6,12] where the targets are supposed to be stationary and don't spawn new targets. This is done because the omnidirectional movement of the robots is very difficult to model from the observational point of view. That is why a zero-order modeling is used (zero mean plus covariance).

- Update:

$$v_k(m) = v_{k|k-1}(m)\left[1 - P_D(m)\right] + v_{k|k-1}(m) \left[\sum_{z \in Z_k} \frac{P_D(m)\, g_k(z \mid m))}{c_k(z) + \int P_D(\xi)\, g_k(z|\xi)\, v_{k|k-1}(\xi)\, d\xi}\right], \quad (3)$$

where z represents a measurement, Z_k is the set of all measurements obtained at time k, $P_D(m)$ represents the probability of detecting z at m, $g_k(z|m)$ represents the likelihood that z is generated by an object at m (that is, the observation likelihood), and $c_k(z)$ is the PHD of the clutter intensity.

Thus, when computing the posterior PHD $v_k(m)$, the first term corresponds to the predicted objects weighted by their probabilities of missed detection and the second to the predicted objects, updated by the spatial locations of all the new observations, and their probabilities of detection.

Using Multiple Sensors

The filter mentioned above works for single-sensor scenarios, but for two or more sensors, the PHD corrector equation become computationally intractable.[13] Given this limitation, Ronald Mahler proposed an *iterated-corrector approximation*:[13] during each measurement cycle, iterate the PHD filter equations once for each sensor. This approximation has been proved to be stable and doesn't result in noticeable differences in performance.[13] So, with these changes, the PHD filter equations are

- Prediction:

$$v_{k|k-1}(m) = v_{k-1}(m) + b_k^{(i)}(m). \quad (4)$$

- Update:

$$v_k(m) = v_{k|k-1}(m) \left[1 - P_D^i(m) + \sum_{z \in Z_k^{(i)}} \frac{P_D^{(i)}(m)\, g_k^{(i)}(z \mid m))}{c_k^{(i)}(z) + \int P_D^i(\xi)\, g_k^i(z|\xi)\, v_{k|k-1}(\xi)\, d\xi}\right], \quad (5)$$

where i is the sensor index, and $Z_k^{(i)}$ are the observations obtained by the ith sensor.

To adapt this framework to the multiple object tracking by a mobile robot problem, some considerations must be made. As presented earlier, $P_D^{(i)}(m)$ represents the probability of detecting object at m by the ith sensor, but it doesn't consider its dependence on the current robot position. So, the real probability is represented by $P_D^{(i)}(m|X_k)$, where X_k represent the position of the robot at time k. The same dependence on X_k applies to $g_k^{(i)}$, $c_k^{(i)}$, and $b_k^{(i)}$ because they also depend on robot position.

Tracking Robots Using RFS

The proposed robot tracking system is based on the PHD filter, and it considers the robots' observations of their own cameras and robot poses received from teammate robots to build the local map of robots (see Figure 1). In addition, the system can merge local maps received from teammate robots to build a combined map of robots.

The tracking system uses a *mixture of Gaussians* implementation of the PHD filter (GM-PHD)[12] because it's time efficient and able to run in real time in robots with limited computational resources.

GM-PHD Implementation

We want to represent the likelihood of any RFS, that is, the PHD, as a mixture of Gaussians, so both hypotheses and new observations are represented by Gaussians. But to allow Gaussians to represent the position and number of objects (robots) in the field, it's necessary to add a weight to every Gaussian. In this way, the Gaussian mean values represent the different locations of the hypotheses on the map, while their weights represent the number of hypotheses in a given region. So, a PHD map is a Gaussian mixture of the form

$$v_{k-1}(m|X_{k-1}) = \sum_{j=1}^{J_{k-1}} \omega_{k|k-1}^{(j)} \mathcal{N}\left(m; \mu_{k|k-1}^{(j)}, P_{k|k-1}^{(j)}\right), \quad (6)$$

which is a mixture of J_{k-1} Gaussians, with $\omega_{k|k-1}^{(j)}$, $\mu_{k|k-1}^{(j)}$, and $P_{k|k-1}^{(j)}$ being the weight, mean, and covariance, respectively. The same form is used to represent the new hypotheses at time k, $b_k^{(i)}(m \mid X_k)$:

$$b_k^{(i)}(m|X_k) = \sum_{j-1}^{J_{b,k}} \omega_{b,k}^{(j)} \mathcal{N}\left(m; \mu_{b,k}^{(j)}, P_{b,k}^{(j)}\right), \quad (7)$$

where $J_{b,k}$ is the number of Gaussians in the new PHD at time k, and $\omega_{b,k}^{(j)}$, $\mu_{b,k}^{(j)}$, and $P_{b,k}^{(j)}$ are the weight, mean, and

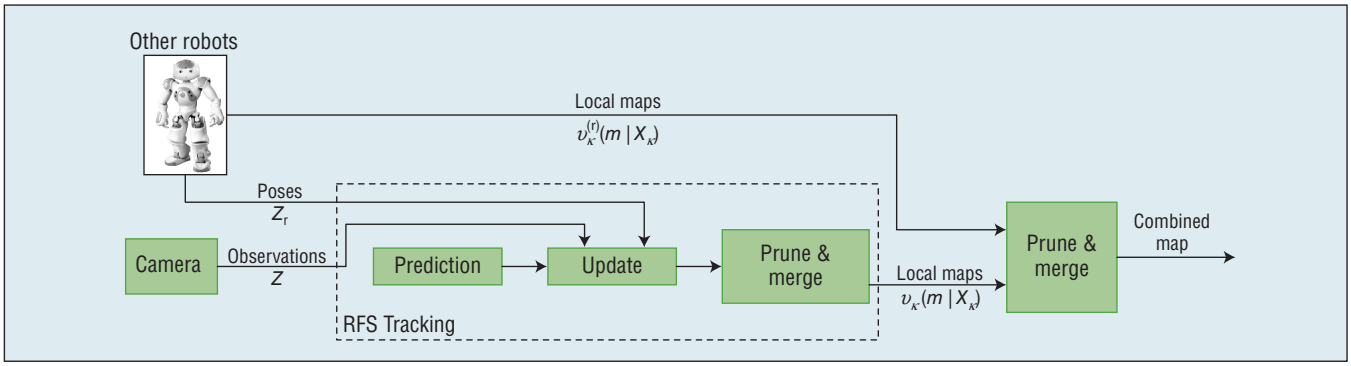


Figure 1. General representation of the proposed method.

covariance, respectively. The calculation of this PHD is made by using the previous observations of each sensor, where each observation creates a Gaussian of the GMM, implying that $J_{b,k} = |Z_{k-1}|$. Therefore, the predicted PHD of the map is also a Gaussian mixture given by

$$v_{k|k-1}(m|X_k) = \sum_{j=1}^{J_{k|k-1}} \omega_{k|k-1}^{(j)} \mathcal{N}\left(m;\ \mu_{k|k-1}^{(j)}, P_{k|k-1}^{(j)}\right), \tag{8}$$

where $J_{k|k-1} = J_{k-1} + J_{b,k}$ is the number of Gaussians representing the union of the prior PHD map $v_{k-1}(m|X_{k-1})$, and the new PHD hypotheses at time k. Note that $\omega_{k|k-1}^{(j)}$, $\mu_{k|k-1}^{(j)}$, and $P_{k|k-1}^{(j)}$ are the weight, mean, and covariance of the Gaussians of the prior map PHD if $j \leq J_{k-1}$, and the weight, mean, and covariance of the Gaussians of the new observations PHD otherwise.

So, the posterior PHD given by Equation 5 is also a Gaussian mixture of the form

$$v_k(m|X_k) = v_{k|k-1}(m|X_k)\left[1 - P_D^{(i)}(m|X_k)\right] + \sum_{z \in Z_k} \sum_{j=1}^{J_{k|k-1}} v_{G,k}^{(i,j)}(z,\ m|X_k), \tag{9}$$

where, according to the general PHD filter update equation, $v_{G,k}^{(i,j)}$ corresponds to

$$v_{G,k}^{(i,j)}(z,\ m|X_k) = \omega_k^{(i,j)}(z \mid X_k)\mathcal{N}\left(m; \mu_{k|k}^{(j)},\ P_{k|k}^{(j)}\right) \tag{10}$$

with

$$\omega_k^{(i,j)}(z|X_k) = \frac{P_D^{(i)}(m|X_k)\omega_{k|k-1}^{(j)} q^{(i,j)}(z \mid X_k)}{c_k(z) + \sum_{l=1}^{J_{k|k-1}} P_D^{(i)}(m|X_k)\omega_{k|k-1}^{(l)} q^{(i,l)}(z \mid X_k)}, \tag{11}$$

where $q^{(i,j)}(z \mid X_k) = \mathcal{N}\left(z;\ H_k^{(i)}\mu_{k|k-1}^{(j)},\ S_k\right)$ is the observation likelihood, i is the sensor index, and j the Gaussian index.

The components $\mu_{k|k}^{(j)}$ and $P_{k|k}^{(j)}$ can be obtained from the standard EKF update equations:

$$S_k^{(l)} = R_k^{(i)} + \nabla_x H_k^{(i)} P_{k|k}^{(l)} \left(\nabla_x H_k^{(i)}\right)^T \tag{12}$$

$$K_k^{(l)} = P_{k|k}^{(i)} \left(\nabla_x H_k^{(i)}\right)^T \left[S_k^{(l)}\right]^{-1} \tag{13}$$

$$\mu_{k|k}^{(l)} = \mu_{k|k-1}^{(l)} + K_k^{(l)}\left(z - H_k^{(i)}\left(\mu_{k|k-1}^{(l)}\right)\right) \tag{14}$$

$$P_{k|k}^{(l)} = \left[I - K_k^{(l)} \nabla_x H_k^{(i)}\right] P_{k|k-1}^{(l)}, \tag{15}$$

with $\nabla_x H_k^{(i)}$ representing the Jacobian of the observation model with respect to the estimated location for the ith sensor.

The parameters used for the creation of the $b_k^{(i)}(m|Z_{k-1}, X_k)$ Gaussians are

$$\omega_{b,k}^{(j)} = \theta_{weight}, \qquad \mu_{b,k}^{(j)} = h_i^{-1}\left(z_{k-1}^j, X_k\right), \qquad P_{b,k}^{(j)} = \nabla_m^j H_k^{(i)}\ R^{(i)}\left[\nabla_m^j H_k^{(i)}\right]^T \tag{16}$$

where h_i^{-1} is the inverse observation model of the ith sensor, $R^{(i)}$ is the observation noise covariance of the ith sensor, $\nabla_m^j H_k^{(i)}$ is the Jacobian of the observation model of the ith sensor, with respect to the Gaussian state j, and θ_{weight} is the initial weight.

Then, as every Gaussian hypothesis is combined with every observation to generate new Gaussian hypotheses,

```
//the parameters of the Map's MoG model (v_{k-1}(m|X_{k-1})) are modified
for i = 1 to J_{k-1} do
  //the robots may move -> covariance is increased using the movement noise
covariance Q
  μ^(i)_{k|k-1} = μ^(i)_{k-1}, P^(i)_{k|k-1} = P^(i)_{k-1} + Q, w^(i)_{k|k-1} = w^(i)_{k-1}
end for
//birth; observations are added; robots' position received from other robots are
treat as observations from additional sensors
for each sensor i
  generateNewGausians(Z_{k-1}, X_{k-1}) //use eq.(7) and (16)
  v_{k|k-1}(m|X_k) = {μ^(j)_{k|k-1}, P^(j)_{k|k-1}, w^(j)_{k|k-1}}_{j=1}^{J_{k|k-1}}
  //update step
  for j = 1 to J_{k|k-1} do
    calculate P_D^(i,j)
    w_k^(j) = (1 - P_D^(i,j)) w^(j)_{k|k-1}
  endfor
  N = 0
  for each z in Z_k
  for j = 1 to J_{k|k-1} do
    calculate H^i, S_k^(j) and K_k^(j)
    μ_k^(N+j) = μ^(j)_{k|k-1} + K_k^(j)(z - μ^(j)_{k|k-1})
    P_k^(N+j) = [I - K_k^(j) H^i] P^(j)_{k|k-1}
    τ^(j) = P_D^(i,j) w^(i)_{k|k-1} |2πS_k^(i)|^-0.5
                          × exp((z - μ^(j)_{k|k-1})[S_k^(j)]^-1 (z - μ^(j)_{k|k-1})^T)
  end for
  for j = 1 to J_{k|k-1} do
    w_k^(N+j) = t^(j)/(c(z) + Σ_{l=1}^{J_{k|k-1}} t^(l))
  end for
  N = N + J_{k|k-1}
  end for
  J_k = N
  //updated map
  v_k(m|X_k) = {μ_k^(j), P_k^(j), w_k^(j)}_{j=1}^{J_k}
  merge_and_prune (v_k(m|X_k)) // use eq. (18)
end for
//combined map building; local maps received from other robots are merge with the
local map
for each received local map i
  merge_maps (v_k(m|X_k), v_k^(i)(m|X_k)) //use eq. (19)
end for
```

Figure 2. Pseudocode of the general algorithm that calculates the PHD that represent the map of robots.

the numbers of Gaussians may grow exponentially in every frame. That is why pruning and merging operations are necessary.[13] First, all Gaussians are sorted by weight. Then, for each Gaussian g_i from $v_k(m|X_k)$,

$$\text{merge}(g_i, g_j) \text{ if } dM(g_i, g_j) < \theta_{local}, \forall g_j \in v_k, j > i, \quad (17)$$

where $dM(g_i, g_j)$ is the Mahalanobis distance between g_i and g_j, and θ_{local} is the local distance threshold. Note that this merge doesn't represent an elimination of a hypothesis because one Gaussian can represent more than one hypothesis by its weight, which are added when two or more Gaussians are merged. The Mahalanobis distance is used because it considers the covariance of the Gaussians being compared. Also, if a Gaussian has a weight below a given threshold $\theta_{minWeight}$, then it's erased from the map.

This process is called merge_and_prune in Figure 2.

Construction of the Local Map of Robots

Multiple aspects are needed to construct the local map of robots using the presented framework (GM-PHD).

Observation process. The robot that builds the map has two sources of information: its own camera and the information sent by its teammates. To detect the other robots in the input images, color blobs are built using scan lines. Blobs surrounded by green pixels are robot candidates (because the field is always green as a rule). Additional details of this detector could be found elsewhere.[3] Then, if one or more robots are detected in the image, their positions are projected on the field coordinates by using the camera's forward kinematic, its intrinsic and extrinsic parameters, and the robot's pose. Also, depending on the position of the detected robots relative to the observing robot, a covariance is calculated, which is also projected on the field coordinates.[3] This covariance is used as $R^{(1)}$ (the covariance of the first sensor). The probability of detection p_D and the clutter intensity are calculated empirically, using statistics of the false and true positives in several situations. The p_D of each Gaussian of the map doesn't have a constant value and must be recalculated in every frame by using the inverse kinematics of the camera, that is, the information of where the camera is observing. If the robot is expected to appear in the image, then it has the p_D calculated before. But if not, the p_D is set to zero.

The second source of information corresponds to the poses of all the teammates, each one treated as a simulated sensor. This information is available through wireless communication. Each robot shares not only its pose but also its player number (every robot on a team must have a number between 1 and 7). So, to use this information in the GM-PHD framework, two other variables are added to each Gaussian. The first is a Boolean (called g^c) that indicates if this Gaussian has ever been updated using the communication information (that is, the other robot poses), and the second is the player number that was used to update this Gaussian (called g^p). This second variable is considered valid only when $g^c == true$. So, when the pose of a teammate is added to the map as defined in Equation 7, this Gaussian starts with $g^c == true$ and g^p equal to the number of the sharing teammate. For these cases, the merge operation defined by Equation 17 is modified as

$$\begin{aligned}&\text{merge}\left(g_i, g_j\right) \quad \text{if } [\{\sim\left(g_i^c \text{ and } g_j^c\right) \text{ and } \text{dM}\left(g_i, g_j\right)\\ &< \theta_{\text{local}}\} \text{ or } \left\{g_i^c \text{ and } g_j^c \text{ and } g_i^p \;\; == \;\; g_j^p\right\}]\end{aligned}. \tag{18}$$

This implies that Gaussians made from poses of two different teammates are never merged, but Gaussians made from the camera sensor can be merged with any other Gaussian that's close enough. This also lets us calculate a detection probability that depends on whether the Gaussian was communicated one or not, that is, the p_D is set on zero for this simulated sensor if $g^c == false$ and near 1 if $g^c == true$.

So, when the final local map is constructed, each target can be associated with one of the teams (their own team or the rival team) by using the Boolean g^c. This information is critical if the local map is going to be used for a high-level task such as passing, and it can't be accomplished without information about the teammates' positions.

Modeling. The state space used in this approach is a vector $p = (x, y)$ that indicates the robot's position in the field coordinates, whose origin is the center of the field. To use this state space, it's necessary to assume that the robot is always localized. In the RoboCup competition, this is generally true because this is the most important subject for a robot to play properly. Particularly in this case, the robots use a multihypothesis Kalman filter for self-localizing.[14,15]

Given that both sensors already give their observations in the field coordinates, H is the identity matrix for both sensors. Also, the use of this state space implies that the speed of the robots isn't used as part of the state. This is the reason for using the simplified version showed in Equation 2.

Construction of the Combined Map of Robots

To build a combined map of robots for each robot, the robots must share their local maps, that is, the GM representation of the map. Thus, the local maps are combined by using the weight and positions of every Gaussian using a pruning and merging process.[12] This process orders every Gaussian by its weight and then uses the Mahalanobis distance to merge similar hypotheses into one. So, for each Gaussian g of every local map $v_k^{(i)}(m|X_k)$ of each other robot i,

$$CMap \leftarrow \begin{cases} g, & dM\left(g, g_{j\in LMap}\right) > \theta_{combined} \\ merge\left(g, g_j\right), & dM\left(g, g_{j\in LMap}\right) < \theta_{combined} \end{cases}, \tag{19}$$

where $CMap$ is the combined map, $LMap$ is the local map of the current robot, that is, $v_k(m|X_k)$, $\text{dM}(g_i|g_j)$ is the Mahalanobis distance between g_i and g_j, and $\theta_{combined}$ is the distance threshold. This process is called merge_maps in Figure 2.

Results

We evaluated the proposed tracking methodology in simulated soccer matches of Nao robots (1 versus 1 and

Table 1. Summary of the obtained results. The numbers correspond to the OSPA values of each experiment. All values are in millimeters (mm).

	Test	Average	Best	Worst
RFS-Local-Map	1 vs. 1	197.8	182.4	212.4
	5 vs. 5	348.6	319.7	381.9
RFS-Combined Map	5 vs. 5	271.3	225.2	298.7
MH-EKF	1 vs. 1	293.6	271.1	305.3
	5 vs. 5	421.5	391.3	453.2

5 versus 5). The main reason for using simulations is to repeat experiments and get an accurate measurement of the ground truth of the robots' positions. Nevertheless, it's important to mention that the proposed tracking system is used in our soccer team of Nao robots, where it can run in real time (approximately 30 fps).

The Nao robot has two cameras of 60° horizontal field of view and 47° vertical field of view, and uses a single-core ATOM processor running at 1.6 GHz. The B-Human Simulator,[16] a 3D tool that accurately simulates the physics and sensors in the Nao robot, is used in the experiments. Note that this tool simulates the images that the robot receives, therefore perception procedures are made using these images exactly as in a real robot, meaning that the simulation has a very similar noise process of the detected robots.

Two variants of the proposed method are analyzed, RFS-Local-Map, which doesn't consider maps received from other robots, and RFS-Combined-Map, which corresponds to the proposed method. The methods are tested against a classical multihypothesis EKF (MH-EKF) tracking method,[3] where each robot position is estimated via an independent EKF. In the method, each observation is associated with an existing hypothesis using a distance threshold D_{EKF}; a new hypothesis is created if the observation isn't associated. If a hypothesis has no observations in a certain period T_{EKF}, then it's eliminated.

To evaluate the performance of each method, we used the OSPA metric.[17] This metric allows us to compare the obtained maps of robots with the ground truth map of robots by calculating a distance $\bar{d}^c$ between the two sets (maps) as

$$\bar{d}^c(X,Y)=\left(\frac{1}{|Y|}\left(\min_{j\in\{1,..,|Y|\}}\sum_i^{|X|}d^c\left(x^i,y^j\right)^2+c^2\left(|Y|-|X|\right)\right)\right)^{1/2}, \quad (20)$$

where $|X| < |Y|$, $d^c(x^i, y^j) = \min(c, \|x^i - y^j\|)$, and c is a cut-off parameter that represents the maximum distance to associate two positions.

The first experiments correspond to 10 simulated soccer matches of 5 versus 5 robots. In the experiments, all the robots calculate the OSPA metric for each method under evaluation, with a cut-off value c = 500 mm. The MH-EKF parameters used are D_{EKF} = 500 mm and T_{EKF} = 8*s*, which are selected to have the best OPSA results. In the case of the RFS-Local-Map, the parameters are θ_{weight} = 0.01, θ_{local} = 18, and θ_{object} = 0.3. For the sensors, $P_D^{(1)} = 0.35$ if the robot must appear in the image and zero if not, and $P_D^{(2)} = 0.98$ if the Gaussian has g^c = *true* and zero if g^c = *false*. In addition, $R^{(1)}$ is calculated for the perceptor[4] and $R^{(2)}$ = [10000,0; 0,10000]. As explained earlier, $H^{(i)} = I$ for i = {1,2}, that is, for both sensors. For the RFS-Combined-Map, $\theta_{combined}$ = 20 is chosen.

During the matches, which were 10 minutes each, every robot played normally, applying the same rules of penalization as in a real RoboCup competition. The results of these matches can be seen in Table 1 (best, average, and worst OSPA errors for each case). The results show that the OSPA errors of the RFS-Local-Map and the RFS-Combined-Map methods are lower than the error of the MH-EKF method. Error reductions of 17.29 percent and 35.73 percent are obtained by RFS-Local-Map and the RFS-Combined-Map, respectively.

In the second set of experiments, 10 matches of 1 versus 1 were performed. In this case, only the RFS-Local-Map is compared against the MH-EKF method (there is no shared information to use). These experiments were performed to show that the proposed method overcomes the difficulties of data association and has a better performance even though no information is shared by other robots. As can be seen in Table 1, in this case, RFS-Local-Map obtains an error reduction of 32.62 percent when compared to the baseline method.

Our proposed methodology can integrate information shared by teammate robots, their positions, and their estimations of other robots' positions (that is, their local maps). We validated our methodology in several soccer matches and compared with a classical multihypothesis EKF tracking methodology. Although the specific application of the proposed tracking methodology is soccer robotics, its adaptation to other robot tracking applications/environments is straightforward. ■

THE AUTHORS

Pablo Cano is an MS student in the Department of Electrical Engineering at the University of Chile. His research interests include robotics, computer vision, and decision-making systems. Cano was a member of the UChile Robotics Team, which participates in the RoboCup Competition, between 2011 and 2016, leading the team in his last year. Contact him at pcano@die.uchile.cl.

Javier Ruiz-del-Solar is director of the Advanced Mining Technology Center at the Universidad de Chile His research interests include mobile robotics, autonomous systems, and human-robot interaction. Ruiz-del-Solar received a Doctor-Engineer from the Technical University of Berlin. He's recipient of the IEEE RAB Achievement Award 2003, RoboCup Engineering Challenge Award 2004, RoboCup @Home Innovation Award 2007, and RoboCup @Home Innovation Award 2008. Ruiz-del-Solar is a senior member of IEEE. Contact him at jruizd@ing.uchile.cl.

Acknowledgments

This work was partially funded by the Advanced Mining Technology Center of the Universidad de Chile and by FONDECYT Project 1161500.

References

1. T. Schmitt et al., "Probabilistic Vision-Based opponent Tracking in Robot Soccer," *Robocup 2002: Robot Soccer World Cup VI*, LNAI 2752, Springer, 2003, pp. 426–434.
2. R. Marchant, P. Guerrero, and J. Ruiz-del-solar, "Cooperative Global Tracking Using Multiple Sensors," *Robocup 2012: Robot Soccer World Cup XVI*, LNAI 7500, G.A. Kaminka et al., eds., Springer, 2013, pp. 310–321.
3. A. Fabisch, T. Laue, and T. Röfer, "Robot Recognition and Modeling in the RoboCup Standard Platform League," *Proc. 5th Workshop Humanoid Soccer Robots*, 2010, pp. 65–70.
4. J. Santos and P. Lima, "Multi-Robot Cooperative Object Localization: Decentralized Bayesian Approach," *RoboCup 2009: Robot Soccer World Cup XIII*, LNCS 5949, J. Baltes et al., eds., Springer, 2010, pp. 332–343.
5. I.R. Goodman, R.P.S. Mahler, and H.T. Nguyen, *Mathematics of Data Fusion*, Springer, 1997.
6. R.P.S. Mahler, "A Theoretical Foundation for the Stein-Winter 'Probability Hypothesis Density (PHD)' Multitarget Tracking Approach," *Proc. MSS Nat'l Symp. Sensor and Data Fusion*, vol. I, 2000, pp. 99–117.
7. R. Mahler, "A Brief Survey of Advances in Random-Set Fusion," *Proc. Int'l Conf. Control, Automation and Information Soc.*, 2015, pp. 62–67.
8. P. Dames and V. Kumar, "Autonomous Localization of an Unknown Number of Targets without Data Association Using Teams of Mobile Sensors," *IEEE Trans. Automation Science and Engineering*, vol. 12, no. 3, 2015, pp. 850–864.
9. P. Dames, P. Tokekar, and V. Kumar, "Detecting, Localizing, and Tracking an Unknown Number of Moving Targets Using a Team of Mobile Robots," *Proc. Int'l Symp. Robotics Research,* 2015, pp. 513–529.
10. D. Moratuwage, B.-N. Vo, and D. Wang, "Collaborative Multi-Vehicle SLAM with Moving Object Tracking," *Proc. IEEE Int'l Conf. Robotics and Automation*, 2013, pp. 5702–5708.
11. R.P.S. Mahler, Statistical Multisource-Multitarget Information Fusion, Artech House, 2007.
12. B.-N. Vo and W.-K. Ma, "The Gaussian Mixture Probability Hypothesis Density Filter," *IEEE Trans. Signal Process.*, vol. 54, no. 11, 2006, pp. 4091–4104.
13. R. Mahler, "The Multisensor PHD Filter, I: General Solution via Multitarget Calculus," *Proc. SPIE,* 2009, pp. 73360E–73360E–12.
14. J. Mullane et al., Random Finite Sets for Robot Mapping and SLAM, Springer, 2011.
15. G. Jochmann et al., "Efficient Multi-Hypotheses Unscented Kalman Filtering for Robust Localization," *RoboCup 2011: Robot Soccer World Cup XV*, LNCS 7416, T. Röfer et al., eds., Springer, 2011, pp. 222–233.
16. T. Laue and T. Röfer, "Simrobot-Development and Applications," *Proc. Int'l Conf. Simulation, Modeling and Programming for Autonomous Robots*, 2008, pp. 143–150.
17. D. Schuhmacher, B.-T. Vo, and B. Vo, "A Consistent Metric for Performance Evaluation of Multi-Object Filters," *IEEE Trans. Signal Processing*, vol. 56, no. 8, 2008, pp. 3447–3457.