



MÁSTER UNIVERSITARIO EN SISTEMAS TELEMÁTICOS E INFORMÁTICOS

Escuela Técnica Superior de Ingeniería Telecomunicación

Curso académico 2013/2014

Trabajo Fin de Máster

Estimación multiobjeto en entornos dinámicos

Autor: Gonzalo José Abella Dago

Tutor: Francisco Martín Rico

A mi familia y Emma

Agradecimientos

Lo cierto es que tengo que agradecer a mucha gente la oportunidad de haber hecho este proyecto. En primer lugar agradecer a Paco, el tutor, todo su esfuerzo y dedicación. He tenido la suerte de trabajar un par de años bajo su tutela y hacer el PFC y el TFM. En todo este tiempo he podido aprender muchísimo de robótica y de muchas otras cosas. ¡Así da gusto trabajar!

Como siempre agradecer infinito a la familia, siempre ayudando, siempre apoyando, siempre ahí, a las duras y a las maduras.

A la gente del Grupo del Robótica de la URJC, compañía indispensable en el despacho. Al final cada uno acaba por un camino distinto, pero estoy seguro que en algún momento en el futuro nos volveremos a cruzar.

Y por último la más importante, Emma. Aguantándome y animándome a diario. Estoy feliz de tenerte.

Resumen

Los robots móviles perciben, miden y *entienden* el entorno que les rodea gracias a sus sensores - láser, ultrasonidos, *bumpers* o cámaras. Esta tarea, que para una persona es básica, para un robot es extremadamente difícil. Uno de los sensores más populares y que más información pueden proporcionar al robot son las imágenes proporcionadas por las cámaras. Analizar estas imágenes es costoso y, normalmente las observaciones no son muy precisas. Por esta razón, es imprescindible filtrar las observaciones para mantener estimaciones fiables y robustas de los estímulos visuales relevantes para las tareas que un robot debe llevar a cabo.

En este proyecto se ha desarrollado un algoritmo para mantener una estimación de múltiples objetos. Para este propósito se ha empleado una colección dinámica de Filtros Extendidos de Kalman. El estado de estos filtros está compuesto por la posición y velocidad del objeto. La dinámica de los objetos se modela mediante una serie de parámetros simples que hacen que este algoritmo pueda aplicarse a un gran rango de problemas.

Como banco de pruebas se ha utilizado en entorno del fútbol de robots. Esta competición es ideal para poner a prueba algoritmos experimentales para, de una forma visual y sencilla, medir su rendimiento y compararlos con el estado del arte actual.

El proyecto se ha realizado dentro del Grupo de Robótica de la Universidad Rey Juan Carlos y la solución se ha implementado dentro del código del equipo de la RoboCup SPL SpiTeam como arquitectura base. Se ha desarrollado una implementación para reemplazar el actual algoritmo de seguimiento de la pelota y las porterías.

Índice general

1. Introducción	1
1.1. Robótica	1
1.2. Visión computacional en robótica	6
1.2.1. Visión 2D y 3D	10
1.3. RoboCup	13
1.4. Estructura del documento	16
2. Objetivos y Metodología	17
2.1. Descripción del problema y requisitos	17
2.2. Objetivo del proyecto	17
2.3. Metodología de desarrollo	18
2.4. Plan de trabajo	20
3. Servidor de mapas dinámico	21
3.1. Costmap_2D	21
3.2. Mapas del servidor de mapas dinámico	22
3.3. Colección de EKFs	24
3.3.1. Inicialización de los filtros	25
3.3.2. Predicción de nuevos estados	26
3.3.3. Incorporación de observaciones	26
3.3.4. Descarte de filtros	30
3.4. Adición de incertidumbre	30
3.5. Objetos complejos	32
4. Experimentos	34
4.1. Seguimiento de un objeto	34
4.2. Descubrimiento de falsos positivos	35
4.3. Seguimiento de objetos con falsos positivos	37
4.4. Seguimiento de objetos complejos	38

4.5. Corrección de la velocidad de un objeto	40
4.6. Tiempos de ejecución	41
5. Conclusiones y trabajos futuros	43
5.1. Conclusiones	43
5.2. Trabajos futuros	44

Índice de figuras

1.1.	Imágenes de distintas representaciones de la obra teatral R.U.R.	2
1.2.	El telar de Jacquard está expuesto en el Museo de la ciencia y la industria en Manchester, Inglaterra.	3
1.3.	Dos de los primeros robots programables en la década de 1970.	4
1.4.	Robot Nao equipado con un láser en la cabeza.	6
1.5.	Detector de personas utilizando una cámara de seguridad en un aeropuerto.	9
1.6.	Imagen de una calle de Bruselas	10
1.7.	Competiciones en la RoboCup	13
1.8.	Competiciones en la RoboCup	14
1.9.	Gary Kaspárov en uno de sus duelos contra la supercomputadora Deep Blue.	15
1.10.	Ajedrez vs Fútbol de robots	15
2.1.	Modelo en espiral.	19
3.1.	Ejemplo visual de un costmap.	21
3.2.	A la izquierda el frame map y a la derecha los frames de odom y base_footprint	22
3.3.	Mapa estático	23
3.4.	Comportamiento de los filtros en la fase de predicción	26
3.5.	Distintas situaciones del emparejamiento entre una observación y un filtro	27
3.6.	Distinguir poste izquierdo de poste derecho	32
4.1.	Distancia a la que se encuentra un objeto haciendo el seguimiento y tomando las medidas del detector directamente	35
4.2.	Gráfico del experimento sobre los falsos positivos	36
4.3.	Seguimiento de la pelota con un falso positivo	37
4.4.	Gráfico del experimento de seguimiento de objetos complejos	39
4.5.	Corrección de la velocidad de un objeto	40
4.6.	Gráfico con tiempos de ejecución	42

5.1. Diferencia de trayectoria con/sin tener en cuenta la velocidad de la bola 45

Capítulo 1

Introducción

La Robótica es uno de los retos tecnológicos más desafiantes a los que se enfrenta el mundo actual. El objetivo principal de esta ciencia es crear máquinas que hagan la vida más fácil, ya sea realizando labores peligrosas o tediosas o simplemente como medio de entretenimiento.

En este primer capítulo se explica el contexto en el que se encuadra el proyecto, la robótica. Muchas técnicas de visión artificial son utilizadas por los robots para medir el entorno en el que se encuentran. Extraer información del mundo no es tarea fácil y las mediciones suelen ser bastante imprecisas. Para reducir y, de alguna manera, minimizar estas imprecisiones se utilizan una serie de algoritmos que se verán más adelante. Todos estos términos se exponen en este capítulo. Por último se explica qué es la RoboCup¹, el porqué de esta competición y la estructura de este documento.

1.1. Robótica

La robótica es la rama de la ciencia que estudia los robots. El término *Robot* es difícil de definir. Fue introducido por el escritor checo Karen Capek en su obra teatral R.U.R. (Robots Universales de Rossum), estrenada en 1921, figura 1.1. Esta palabra deriva del término checoslovaco *Robota* que significa literalmente trabajar y, figuradamente, trabajo duro o penoso.

No hay una única definición que satisfaga a todos los organismos internacionales de estandarización. Por ejemplo, la IFR (International Federation of Robotics) utiliza la definición propuesta por la ISO (Organización Internacional para la Estandarización) incluido en el ISO 8373. En este documento se define como *manipulador programable*

¹<http://www.robocup.org>

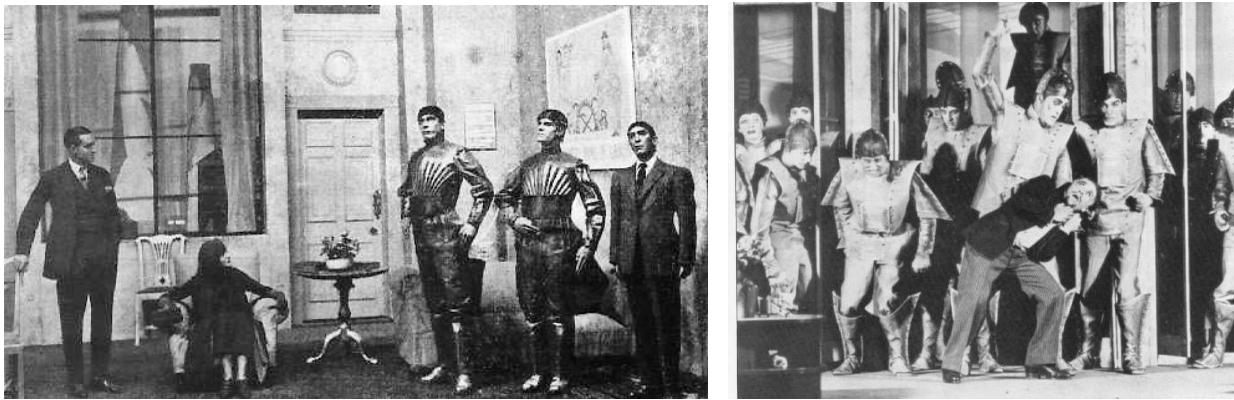


Figura 1.1: Imágenes de distintas representaciones de la obra teatral R.U.R.

en tres o más ejes, multipropósito, controlado automáticamente y reprogramable. Esta definición también es usada por otros comités como el EURON (EUropean RObotics research Network). En cambio, la RIA (Robotic Industries Association) define el término robot como *manipulador reprogramable y multifuncional diseñado para mover materiales, partes, herramientas o dispositivos especializados a través de movimientos programados para la realización de una serie de tareas*. Esta definición es un poco más amplia que la anterior. Por último, la RAE define el término robot como *máquina o ingenio electrónico programable, capaz de manipular objetos y realizar operaciones antes reservadas sólo a las personas*.

Como se puede comprobar, cada una de las definiciones de robot es distinta de las otras. En mi opinión, la definición contenida en la ISO 8373 y la dada por la RIA están muy centradas en la robótica industrial. De hecho, se define al robot como un manipulador. En el lado opuesto, la definición proporcionada por la RAE es demasiado amplia y abstracta. A mi juicio, un robot es una máquina programable capaz de interactuar con su entorno, ya sea modificándolo directamente ya desplazándose por él, para realizar una o varias tareas. Las tareas asignadas suelen ser aquellas que por su peligrosidad, repetición o precisión suponen una liberación para las personas.

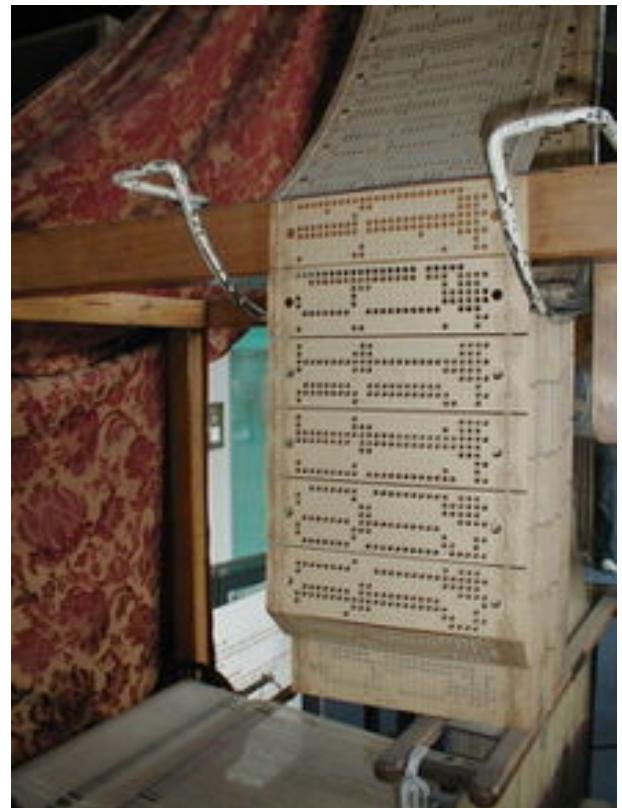
Desde siempre, el ser humano ha intentado construir máquinas que le liberen, simplifiquen o faciliten el trabajo. Los antiguos egipcios unieron brazos mecánicos a las estatuas de sus dioses y los griegos construyeron estatuas con sistemas hidráulicos. Estas estatuas automatizadas eran utilizadas para atemorizar, fascinar e infundir respeto al pueblo. Estos inventos pueden considerarse como los primeros *"robots"* de la historia,

aunque distan mucho de los robots actuales.

El inicio de la robótica, tal y como la vemos en estos momentos, puede fijarse en el siglo XVIII. En 1801 el francés Joseph Jacquard inventó un telar mecánico programable mediante tarjetas perforadas, figura 1.2. Esta máquina permitía elaborar diseños de telares complejos a usuarios inexpertos. Cada tarjeta perforada correspondía a una línea del diseño y la unión de varias tarjetas formaban un patrón con el que se tejería el telar. En esta época también se construyeron algunos artilugios curiosos, como una muñeca mecánica capaz de hacer dibujos de forma autónoma, inventada por Henri Maillardert, o unos músicos de tamaño humano creados por Jacques de Vaucansos. Estos *"robots"* tenían un propósito más de ocio que laboral.



(a) El telar de Jacquard.



(b) Tarjeta programable del telar.

Figura 1.2: El telar de Jacquard está expuesto en el Museo de la ciencia y la industria en Manchester, Inglaterra.

Pero no fue hasta mediados del siglo XX, gracias a los avances en inteligencia artificial, cuando se crearon los primeros robots modernos. El primer robot industrial programable se construye e instala en 1961 en la *Ford Motors Company*, se llamaba *Unimate*. Su cometido era el levantar piezas industriales que estaban a altas

temperaturas. Sin embargo, es en la década de 1970 cuando comienza a desarrollarse completamente la robótica. En 1973 aparece el primer robot con 6 ejes electromecánicos, figura 1.3a y, a los pocos años, el primer brazo mecánico programable, figura 1.3b. Estos primeros robots modernos eran básicamente manipuladores, es decir, brazos mecánicos fijos en el suelo que realizan una tarea concreta.

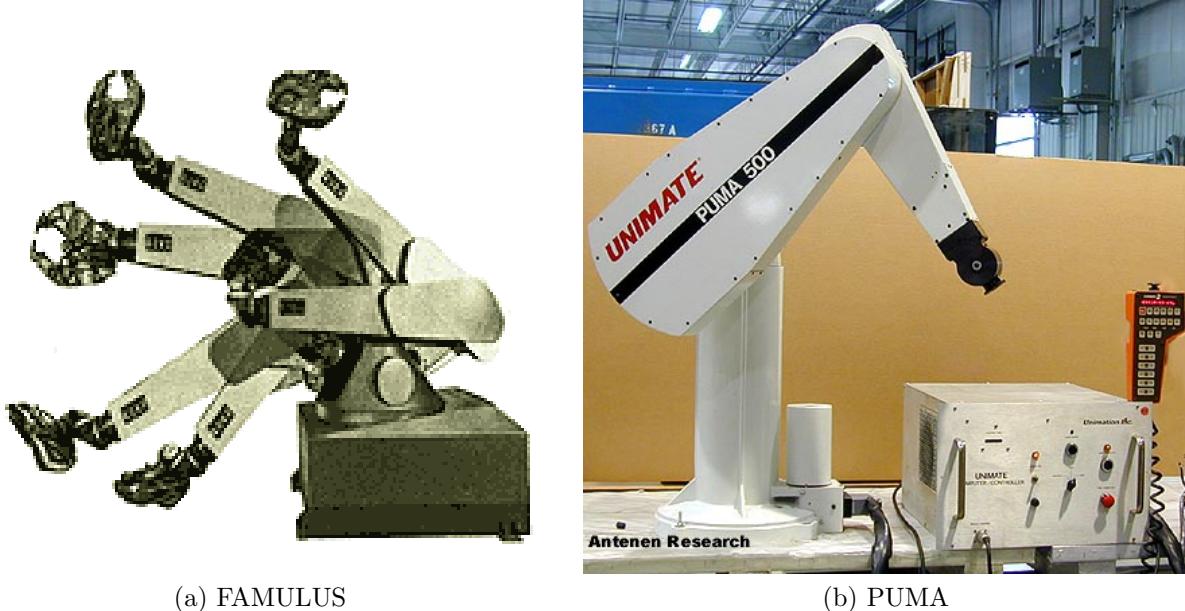


Figura 1.3: Dos de los primeros robots programables en la década de 1970.

Los robots definen su funcionalidad y características en función de una serie de elementos hardware. Estos elementos son los encargados de recibir estímulos, procesarlos y efectuar una respuesta. Los elementos hardware se clasifican en sensores, actuadores y procesadores:

- **Sensores.** Son dispositivos que miden propiedades del entorno tales como distancia, temperatura, intensidad lumínica o señal GPS entre otros. También existen los sensores propioceptivos que miden propiedades del robot, como puede ser la temperatura de los motores, para prevenir que se quemen, o el número de vueltas que da una rueda. Los sensores más utilizados son:
 1. **Ultrasonido.** Es un dispositivo que sirve para medir distancias. La distancia se calcula midiendo el tiempo de vuelo de una onda sonora. En otras palabras, se calcula el tiempo que tarda la onda en ir y volver después de rebotar en un obstáculo. Conocido el tiempo de vuelo y la velocidad de la onda, la distancia se calcula por medio de una sencilla operación matemática.

La onda se emite por encima del espectro audible por el ser humano. Su uso se limita a interiores y el error en la medición suele ser de aproximadamente el 1% de la distancia medida.

2. **Láser.** El láser también sirve para medir distancias y ésta se calcula de la misma manera que lo hace el ultrasonido, sólo que, en vez de utilizar una onda sonora, utiliza una frecuencia de luz del espectro no visible. Este dispositivo es más preciso que el ultrasonido, puede utilizarse tanto en interiores como en exteriores y el error en las medidas suele ser de apenas unos pocos milímetros, pero también es bastante más caro. Un ultrasonido tiene un precio del orden de decenas de dólares, mientras que un láser tiene un precio de miles de dólares.
3. **Cámara.** Este sensor es uno de los más populares. Es un sensor muy rico y barato, el único problema de este dispositivo es la dificultad de interpretar las imágenes por el costo computacional que conlleva dicha tarea. En el siguiente apartado de visión computacional en robots se habla más en profundidad de estos dispositivos.

En la figura 1.4 se puede ver al robot Nao equipado con un láser en la cabeza, aunque este dispositivo es opcional. El robot también dispone de ultrasonidos en el pecho, colocados en la banda azul que rodea el botón central, y de dos cámaras, colocadas una en la frente y la otra en la boca.

- **Actuadores.** Son dispositivos que modifican el estado del robot o su entorno. Los manipuladores o brazos robóticos son un ejemplo de actuadores que pueden interactuar con el entorno, pero también lo son los altavoces o los LEDs. Es importante destacar en este punto la existencia de dos grandes familias de robots: los robots fijos y los robots móviles. Los robots fijos suelen encontrarse en la robótica industrial. Los robots móviles, como su nombre indica, son aquellos capaces de desplazarse por el entorno.
- **Procesadores.** Son elementos encargados de procesar los datos obtenidos de los sensores, analizarlos, comprenderlos y generar una respuesta que ejecutarán los actuadores. Comparando un robot con una persona y haciendo una analogía simple, se puede decir que los sensores, actuadores y procesadores son, respectivamente, los sentidos, los músculos y el cerebro.

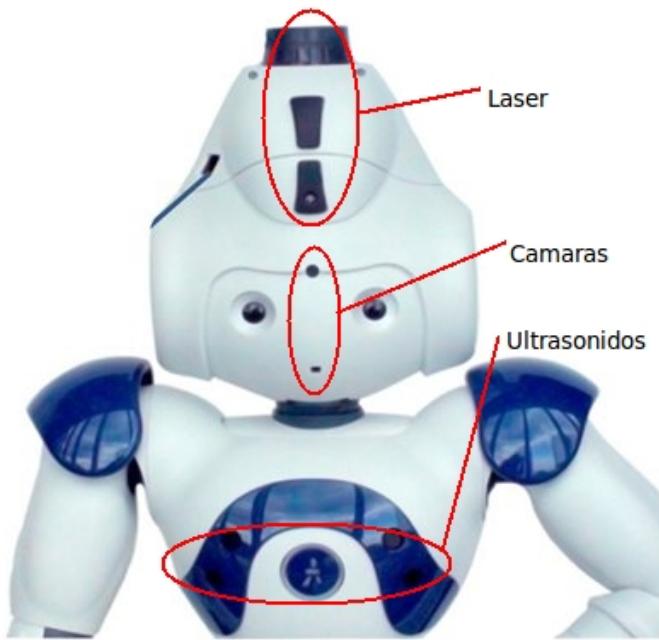


Figura 1.4: Robot Nao equipado con un láser en la cabeza.

1.2. Visión computacional en robótica

La visión computacional es el campo de la inteligencia artificial que pretende obtener información del mundo a partir de una o varias cámaras. La cantidad de información que se puede extraer de las imágenes generadas por las cámaras es enorme. Se pueden reconocer objetos, recrear escenas en 3D, seguir personas, etc. Realizar todas estas tareas para un ser humano es muy sencillo, pero para una computadora, en cambio, es una tarea muy difícil.

El comienzo de la visión computacional fue marcado por Larry Roberts en 1961, cuando creó un programa que podía *ver*. Esta aplicación era capaz de procesar una imagen de una estructura de bloques y reproducir esa misma estructura desde otra perspectiva. Para ello utilizaba solamente una cámara y un ordenador, pero las condiciones del entorno de las pruebas eran muy limitadas. Muchos científicos de la época trataron de solucionar el problema y se dieron cuenta de que no era tarea fácil. Fue entonces cuando se abrió un nuevo campo de investigación: la visión computacional.

Un computador puede tardar años en resolver tareas que para una persona son sencillas y automáticas. A pesar del alto precio computacional que se paga al utilizar las cámaras como fuente de datos, si se consigue analizar correctamente la imagen, es

posible extraer mucha información que no podría obtenerse con ningún otro tipo de sensores.

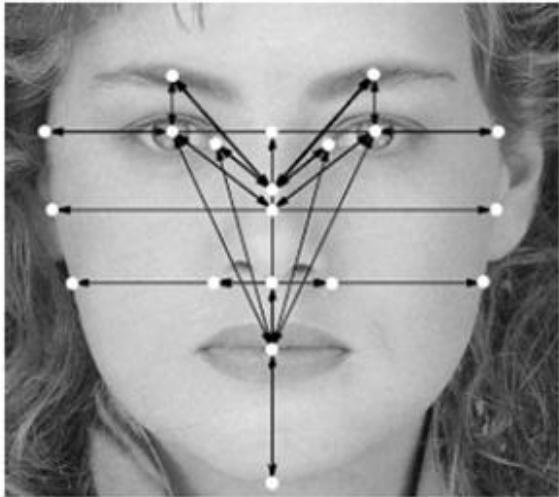
A comienzos de la década de 1990 comenzaron a aparecer ordenadores capaces de procesar suficientemente rápido las imágenes. Esto hizo que grandes problemas de visión comenzaran a dividirse en subproblemas más específicos y, por tanto, más sencillos de resolver. Actualmente la visión computacional se utiliza en muchos procesos científicos, militares o industriales para el reconocimiento de objetos o seguimiento de éstos.

- **Reconocimiento de objetos.** Mediante algoritmos de reconocimiento de objetos se buscan características o patrones de los objetos y se determina si el objeto se encuentra en la imagen o no. Las características más utilizadas son la forma o el color, pero podría utilizarse cualquier otro patrón. En la figura 1.5a se puede ver un patrón de puntos característicos de la cara de una persona. Al convertir este patrón de puntos en una imagen se pueden detectar caras e incluso reconocer a la persona.
- **Seguimiento de objetos.** Una vez detectado un objeto, se pueden realizar tareas de seguimiento de éste. En el siguiente apartado se verá con mayor detalle este tipo de algoritmos. El procesado iterativo de los objetos detectados permite realizar operaciones más complejas. Por ejemplo, en deportes como el tenis o el cricket se utiliza un sistema informático para hacer el seguimiento de la bola. Este sistema genera una imagen de la trayectoria de la bola que puede ser utilizada por los jueces para decidir jugadas dudosas. En la figura 1.5b se puede ver dicha trayectoria. Este sistema es conocido como *Ojo de Halcón*².

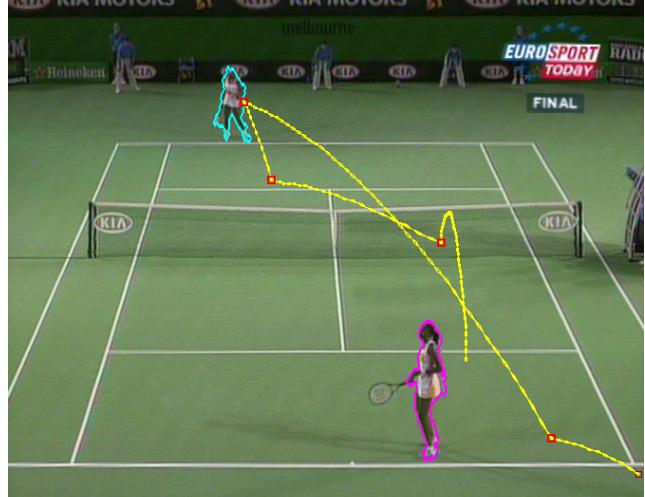
A pesar de la complejidad que supone obtener información a partir de imágenes, las cámaras son uno de los sensores más utilizados en la robótica. Es un sensor bastante barato comparado con el resto de sensores disponibles y la cantidad información que se puede sacar de una sola imagen es enorme.

Los problemas de visión computacional poseen una complejidad bastante elevada. Por ello se han dividido en subproblemas más pequeños, lo que facilita su resolución. Las dos tareas principales para elaborar algoritmos más fiables y robustos son la detección de objetos y el seguimiento o *tracking* de estos.

²[http://es.wikipedia.org/wiki/Ojo_de_Halcón](http://es.wikipedia.org/wiki/Ojo_de_Halc%C3%B3n)



(a) Reconocimiento de cara en imagen



(b) Seguimiento de la pelota durante un partido de tenis

La detección de objetos consiste en localizar un objeto a partir de una imagen. Dependiendo de la resolución de la imagen, esta tarea puede ser muy lenta y consumir muchos recursos. En sistemas de tiempo real, donde se tienen que analizar muchas imágenes por segundo, el tiempo que lleva analizar una imagen es crucial. Si se utilizan sistemas de visión estándar, para conseguir una percepción reactiva del entorno se necesitan un mínimo de aproximadamente 10 imágenes por segundo. Lo ideal sería conseguir una tasa 25 - 30 imágenes por segundo. También hay sistemas más precisos que llegan a analizar 60 o más imágenes en un segundo. En la figura 1.5 se puede ver una captura de pantalla de un sistema de videovigilancia capaz de detectar y seguir personas en una zona de un aeropuerto utilizando las imágenes de las cámaras de seguridad.

Los detectores devuelven información sobre los objetos interesantes de la imagen. Esta información, dependiendo del tipo de sistema, es más o menos precisa. Por ejemplo, no es lo mismo detectar personas con cámaras de vigilancia en un aeropuerto, donde cada persona es totalmente distinta a otra, que detectar objetos simples en una cadena de montaje de una fábrica, donde la diferencia entre los objetos es mínima. Por este motivo las observaciones suelen llevar asociada una incertidumbre o ruido. La incertidumbre depende de la calibración de los parámetros intrínsecos y extrínsecos de la cámara, condiciones de iluminación, etc. Dicho de otra manera, el ruido asociado a una observación indica el grado de fiabilidad de ésta. Cuanto menor es el ruido, más fiable es la observación y más probabilidades tiene el objeto de encontrarse realmente en la posición marcada. Por el contrario, si el ruido es muy grande, la observación no es fiable y seguramente el objeto se encuentre dentro de un área con centro en la posición



Figura 1.5: Detector de personas utilizando una cámara de seguridad en un aeropuerto.

marcada donde según te alejas de él, la probabilidad disminuye. También hay que tener en cuenta la aparición de falsos positivos y falsos negativos, que significa detectar un objeto cuando en realidad no hay nada y no detectar el objeto cuando en realidad sí que se encuentra en la imagen, respectivamente.

En ocasiones, a partir de las observaciones devueltas por los detectores se puede inferir información que no se encuentra en la imagen, no se ha detectado correctamente o es indistinguible. Para ello es necesario tener un modelo del objeto con el que poder comparar. De esta forma se puede sacar información de objetos que están parcialmente ocultos, información que no es medible directamente, o diferenciar objetos totalmente iguales. Por ejemplo, si se detectan tres vértices de un cuadrado y el cuarto no es localizado [?], se puede inferir su posición para en otro momento comprobar si de verdad se encuentra en esa posición; o, por ejemplo, se puede medir la velocidad de un objeto a partir de su posición en distintos instantes de tiempo.

Para mejorar los algoritmos de detección y obtener datos más fiables surgen los algoritmos de seguimiento. Éstos ayudan notablemente a minimizar los problemas que se acaban de exponer: reducir el ruido de las observaciones, protegerse frente a falsos positivos y negativos, inferir datos nuevos a partir de otros o distinguir objetos cuando son exactamente iguales.

1.2.1. Visión 2D y 3D

La visión de un robot puede ser en dos o tres dimensiones. Cuando se usa una sola cámara, lo habitual es tener una visión en 2D, aunque existen técnicas para deducir la tercera dimensión que falta. Las imágenes proporcionadas por las cámaras no tienen profundidad y todos los objetos se encuentran en el mismo plano: tienen dos dimensiones. Le ocurre lo mismo a un ser humano cuando guiña un ojo, no es capaz de medir distancias con la misma facilidad que si lo hace con los dos.

Una persona, ya sea en una imagen o mirando con un solo ojo, utiliza una serie de técnicas instintivas para hacer una representación mental de la imagen y conocer la posición aproximada de los objetos. Por ejemplo, si analizamos la imagen 1.6, a pesar de ser en 2D se puede inferir mucha información de la escena utilizando algunas de estas técnicas.



Figura 1.6: Imagen de una calle de Bruselas

- En la parte derecha de la imagen se ve el morro de un coche y justo a su izquierda una pareja de mujeres, marcados de color rojo. Se sabe que el coche se encuentra en una posición más cercana porque el punto de intersección de la rueda con el suelo está más abajo en la imagen que los pies de las mujeres.

- En la parte izquierda de la imagen hay una mujer con una bolsa en la mano rodeada de azul. Se sabe que se encuentra delante del coche porque oculta la vista de éste.
- De color verde se han resaltado una niña, su padre y una mujer más lejos en la parte izquierda de la imagen. Se sabe que la niña y el padre están más o menos a la misma distancia. De hecho, el padre está un poco más adelantado. Se intuye que es una niña por el color del abrigo y por el tamaño, comparado con el de su padre. En la izquierda de la imagen se puede ver que la mujer tiene el mismo tamaño que la niña; sin embargo, está en una posición más lejana y esto hace que, a pesar de ser más grande en la realidad, su tamaño en la imagen sea menor. Por eso se sabe que se trata de una mujer y no una niña.

Estas son solo algunas de las técnicas que se utilizan. Pueden parecer tontas y un poco obvias porque el ser humano lo hace automáticamente y uno no se para a analizarlas, simplemente se saben. Es una capacidad innata de las personas. Para hacer la representación mental de esta imagen se mezcla muchísima información, desde información física de objetos que se conoce *a priori*, como el tamaño y la forma, con información cultural, como el hecho del abrigo rosa.

Cuando se quiere tener una visión 3D del entorno, se tienen que utilizar un mínimo de dos cámaras. Las cámaras tienen que estar en una posición conocida con respecto a las otras. Lo más habitual en robots es encontrar un par estéreo de cámaras, donde se colocan las cámaras con una disposición similar a la de los ojos humanos, figura 1.7a. Las imágenes de las dos cámaras se pueden ver en la figura 1.7b.



(a) Par estéreo

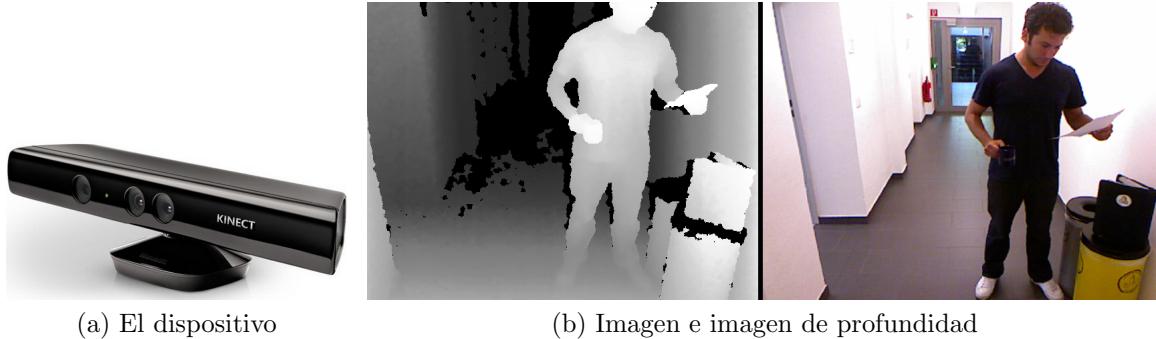


(b) Imágenes de un par estéreo de cámaras

A partir de estas dos imágenes se puede calcular la posición de cualquier punto en la imagen. A partir de un píxel de una de las imágenes hay que descubrir el píxel homólogo en la otra imagen y, mediante un simple cálculo de triangulación, se calcula dicho

punto en 3D. La posición calculada es relativa a la posición de las cámaras. El costo computacional del cálculo de estos puntos es bastante alto, pero no hay restricciones en cuanto a la distancia a la que se encuentre el punto, siempre que sea visible en las dos imágenes.

En los últimos años ha aparecido un nuevo sensor que está cobrando cada vez más protagonismo en el ámbito de los sensores de robots: la *Kinect*, figura 1.7a. Este dispositivo fue inventado por Microsoft para la videoconsola Xbox 360. Permite controlar e interactuar con la consola sin necesidad de tener contacto físico con un controlador de videojuegos tradicional, como un mando o un joystick. Este dispositivo dispone de una cámara y un sensor de profundidad, que no es más que un proyector de infrarrojos combinado con un sensor CMOS monocromo. Básicamente, lo que hace es medir la distancia para cada píxel de la cámara y así obtener una mapa 3D del entorno, independientemente de la luz ambiental.



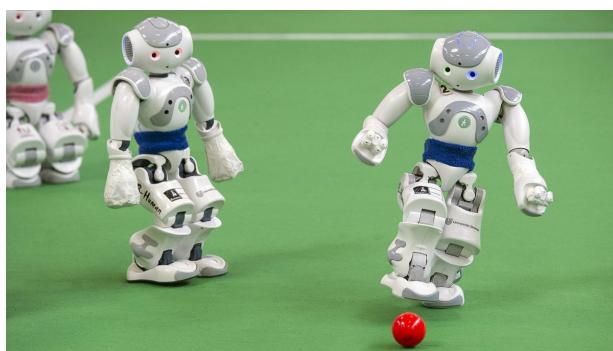
El cálculo de la matriz de distancias se efectúa al mismo tiempo que se captura la imagen y se hace por hardware, liberando la CPU de ese trabajo. En la figura 1.7b se puede ver la imagen capturada y la matriz de profundidad de la imagen. Cuanto menor es la distancia, más claro es el color del píxel y viceversa. Por este motivo este dispositivo se está haciendo muy popular en robótica en los últimos años. Se consiguen los mismos resultados que con una cámara y un láser a un coste mínimo. Uno de los inconvenientes de los pares estéreos de cámaras es el alto costo computacional del cálculo del píxel homólogo. No es una tarea en absoluto sencilla. En cambio, la *Kinect* tiene esta capacidad para todos los píxeles de la imagen sin costo alguno. Pero también tiene sus limitaciones, su uso se limita prácticamente a interiores. La distancia mínima es de unos 60 centímetros, la distancia máxima de unos 4-5 metros y la luz solar directa sobre el sensor aumenta enormemente el error de las mediciones.

1.3. RoboCup

La RoboCup³ es una iniciativa científica a nivel internacional cuyo objetivo es avanzar el estado del arte de la inteligencia en robots. Constituida en 1997, la misión principal era la de crear un equipo de robots completamente autónomos capaz de ganar a la selección de fútbol (humanos) que gane la Copa del Mundo de fútbol en el 2050. Desde que se creó, esta competición se ha expandido y ahora también abarca distintos dominios donde la robótica puede ser aplicada y ser potencialmente útil para la sociedad moderna.

La RoboCup está diseñada para entender y encontrar soluciones a problemas complejos del mundo real en un dominio limitado, para que ello sea asequible y computacionalmente posible. En la RoboCup se cubren muchas áreas de investigación que tienen que ver con la inteligencia artificial y la robótica. Entre ellas se encuentran las siguientes: procesamiento en tiempo real, comportamiento reactivo, aprendizaje, planificación, sistemas multi-agente, visión y muchas más. Las competiciones son:

- **RoboCupSoccer.** Es una competición de fútbol en la que equipos de robots completamente autónomos y cooperativos compiten entre ellos en un partido de fútbol.
- **RoboCupRescue.** Competición dónde los robots asisten a *servicios de emergencia* para salvar gente y llevar a cabo tareas peligrosas. Estos robots tiene una gran movilidad, son semi-autónomos y capaces de cartografiar y moverse a través de entornos complejos.



(a) RoboCupSoccer



(b) RoboCupRescue

Figura 1.7: Competiciones en la RoboCup

³<http://www.robocup.org>

- **RoboCup@Home.** El objetivo de esta competición es realizar típicas tareas domésticas con robots autónomos y una fácil interacción humano-robot.
- **RoboCupJunior.** Esta competición se centra en los más jóvenes. Los participantes tienen que construir y programar robots autónomos sobre los que tienen que aplicar la tecnología, ingeniería y las matemáticas sobre las que se sustenta la robótica.

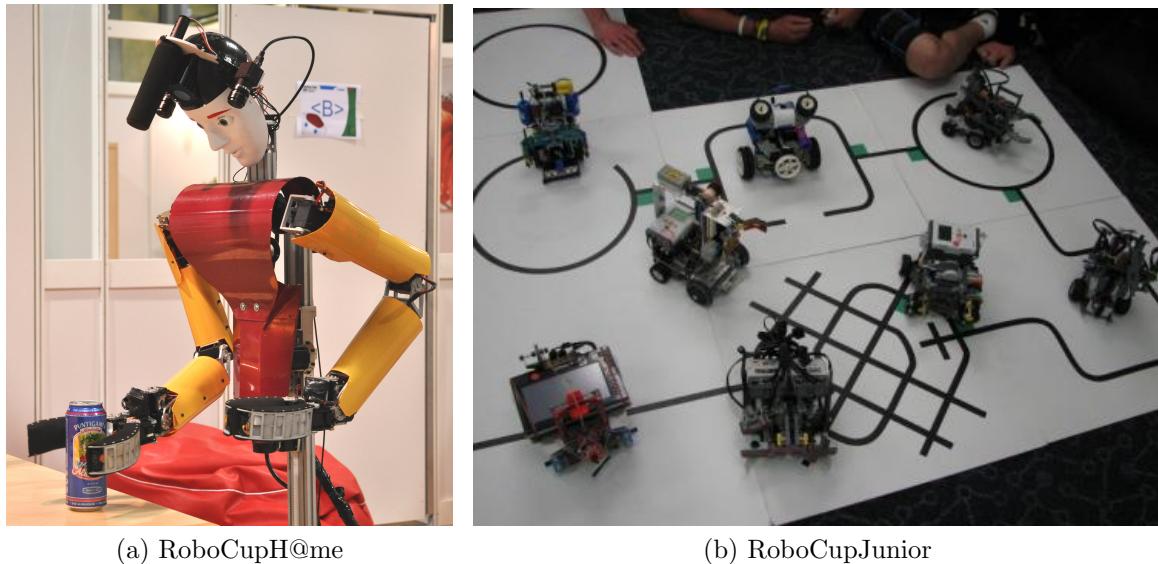


Figura 1.8: Competiciones en la RoboCup

La gran pregunta respecto a esta competición es ¿Por qué fútbol?. El fútbol no es una tarea peligrosa, repetitiva o difícil de hacer por una persona, y no tiene una utilidad concreta. Esto es cierto; sin embargo, el fútbol es simplemente la "excusa". Es un *problema* o juego que conoce todo el mundo. Un banco de pruebas perfecto para comparar y crear nuevos algoritmos, arquitecturas robóticas y desarrollar la autonomía e inteligencia de los robots. También es un juego muy vistoso y atractivo que consigue atraer a más gente.

No es el primer deporte que se toma como banco de pruebas. Hace unos años se investigó y se desarrollaron multitud de algoritmos de búsqueda con la excusa del ajedrez. En 1997 la supercomputadora *Deep Blue* consiguió ganar al campeón del mundo en ese momento, el ruso Gary Kaspárov. En la figura 1.9 se puede ver una instantánea de la partida entre el jugador y la máquina. La forma de interacción con el entorno y las piezas era a través de una persona que introducía en el ordenador los movimientos que realizaba Kaspárov y hacía el movimiento mostrado en la pantalla.



Figura 1.9: Gary Kaspárov en uno de sus duelos contra la supercomputadora Deeper Blue.

El ajedrez y el fútbol son dominios prácticamente opuestos en los que hay que enfrentarse a distintos desafíos para cumplir los objetivos. Las diferencias se pueden ver en la figura 1.10.

	Ajedrez	RoboCup
Entorno	Estático	Dinámico
Cambio de estado	Por turnos	Tiempo real
Visión del problema	Completa	Incompleta
Percepción/Interacción entorno	A través humano	Sensores y motores
Control	Centralizado	Distribuido

Figura 1.10: Ajedrez vs Fútbol de robots

Las diferencias entre los dos deportes saltan a la vista. Un partido de fútbol es un entorno hostil y dinámico donde en unos pocos segundos la situación de juego puede cambiar radicalmente. Las decisiones se tienen que tomar instantáneamente y con los datos disponibles, porque no se tiene una percepción completa del entorno. Toda la información proviene directamente de los sensores. Al ser un juego en equipo también son importantes la información que puedan proporcionar el resto de los robots. En el ajedrez, en cambio, se tiene una visión completa del escenario y el juego es por turnos, por lo que las decisiones no tienen por qué ser instantáneas. Aunque la cantidad de distintos posibles movimientos sea enorme -y de ahí la dificultad del problema-, la interacción y la percepción se hacen a través de una persona y el control es centralizado.

Se pueden encontrar infinidad de artículos científicos que utilizan como banco de pruebas la RoboCup. En [?] se desarrolla un algoritmo para hacer el seguimiento de objetos en general en el contexto de la RoboCup. [?] se centra únicamente en el seguimiento de los otros robots que se encuentran en el campo de fútbol. Y por último, en [?] se presenta un algoritmo para realizar el seguimiento en 2D de objetos en una imagen. Este simplifica la tarea, ya que, al no tener orientación los objetos, no hay rotaciones. Todos estos artículos proponen soluciones al seguimiento de objetos similares a la solución que se ha desarrollado en este proyecto.

1.4. Estructura del documento

En este documento se describen los aspectos más relevantes del desarrollo del algoritmo. La memoria está dividida en seis capítulos. En este primer capítulo se ha presentado el contexto en el que se va a desarrollar el proyecto. En el segundo capítulo se define el problema concreto, los objetivos y requisitos del proyecto. La infraestructura del software utilizado se expone en el tercer capítulo. El cuarto capítulo describe los detalles técnicos del algoritmo desarrollado. Con el fin de demostrar la validez de la investigación, en el quinto capítulo se adjuntan una serie de experimentos. Por último, el sexto capítulo contiene las conclusiones y se proponen nuevas líneas de investigación.

Capítulo 2

Objetivos y Metodología

2.1. Descripción del problema y requisitos

La navegación en entornos dinámicos, como puede ser una casa o una institución pública, supone un gran reto que superar ya que las personas o las mascotas pueden acercarse o cruzarse delante del robot o podríamos encontrarnos objetos del mobiliario que han sido movidos de su posición original y se encuentran en nuestro camino. En este escenario el robot no debería nunca ni chocar ni perderse en el entorno y debe llegar al destino impuesto por la mejor ruta disponible.

Por ejemplo, si nuestro robot está lleno desde el salón a la cocina de nuestra casa pero en su camino habitual y más óptimo se encuentra un mueble, el robot debe darse cuenta rápidamente, esquivarlo, proseguir con su camino y además recordarlo para que cuando volvamos al salón de vuelta podamos esquivarlo más fácilmente. Por otro lado en nuestra casa también habrá personas, estas personas se moverán casi constantemente por la estancia por lo que no será del todo correcto tenerlas en cuenta a la hora de planificar nuestra ruta para navegar de un sitio a otro de la casa pero si que será muy importante no chocar con ellas.

Para analizar el entorno del robot usaremos el sensor laser, este sensor destaca por su alta precisión y su corto tiempo de procesado.

2.2. Objetivo del proyecto

Se quiere diseñar un algoritmo que genere un mapa en tiempo real, el cual será usado por el nodo de navegación de ROS para navegar por un entorno doméstico, ya sea indicando una posición x,y en el mapa o indicándole una estancia a la que navegar.

Este mapa se construirá a partir de la mezcla de 3 mapas, mapa estático, mapa de largo plazo y mapa de corto plazo.

En primera instancia el algoritmo se validará haciendo uso de un simulador en el que se representa una casa con varios tipos de muebles, ya que resulta más fácil de depurar un algoritmo en un entorno virtual, que en un entorno real. Posteriormente el algoritmo se probará en distintas recreaciones de escenarios reales, y se harán las modificaciones oportunas para adaptarlo al entorno real, y por último se llevará a la competición.

Para simplificar la resolución del problema se ha dividido el proyecto en varios subobjetivos:

1. Se usarán las herramientas por defecto que nos ofrece ROS para crear el mapa de corto plazo en referencia a las mediciones tomadas por el laser. En un primer paso solo añadiremos los diferentes objetos que percibamos.
2. Se ampliará el algoritmo anterior para poder añadir y eliminar objetos que aparezcan o desaparezcan del entorno.
3. Se desarrollará un algoritmo para mezclar los mapas entre sí y así generar tanto el mapa de largo plazo como el mapa que usaremos para la navegación.
4. Se creará un servidor de mapas dinámicos que se inicializará con los mapas estático y de largo plazo y que generará el mapa final mezclando los mapas de largo plazo y de corto plazo.
5. Se usará el mapa final como parámetro del paquete de navegación de ROS, *move base*, y del paquete de localización de ROS *amcl*.
6. se generará y usará un mapa semántico, en el que cada nivel de gris se asocie con una etiqueta para después ordenar al robot que navegue a dichas etiquetas.

2.3. Metodología de desarrollo

En el desarrollo del sistema descrito, el modelo de ciclo de vida utilizado ha sido el modelo en espiral basado en prototipos. Este modelo permite desarrollar el proyecto de forma incremental, aumentando la complejidad progresivamente y haciendo posible la generación de prototipos funcionales. Este planteamiento permite obtener productos parciales al final de cada ciclo que pueden ser evaluados, ya sea total o parcialmente. Esto facilita la adaptación a los cambios en los requisitos, circunstancia muy común en

los proyectos de investigación.

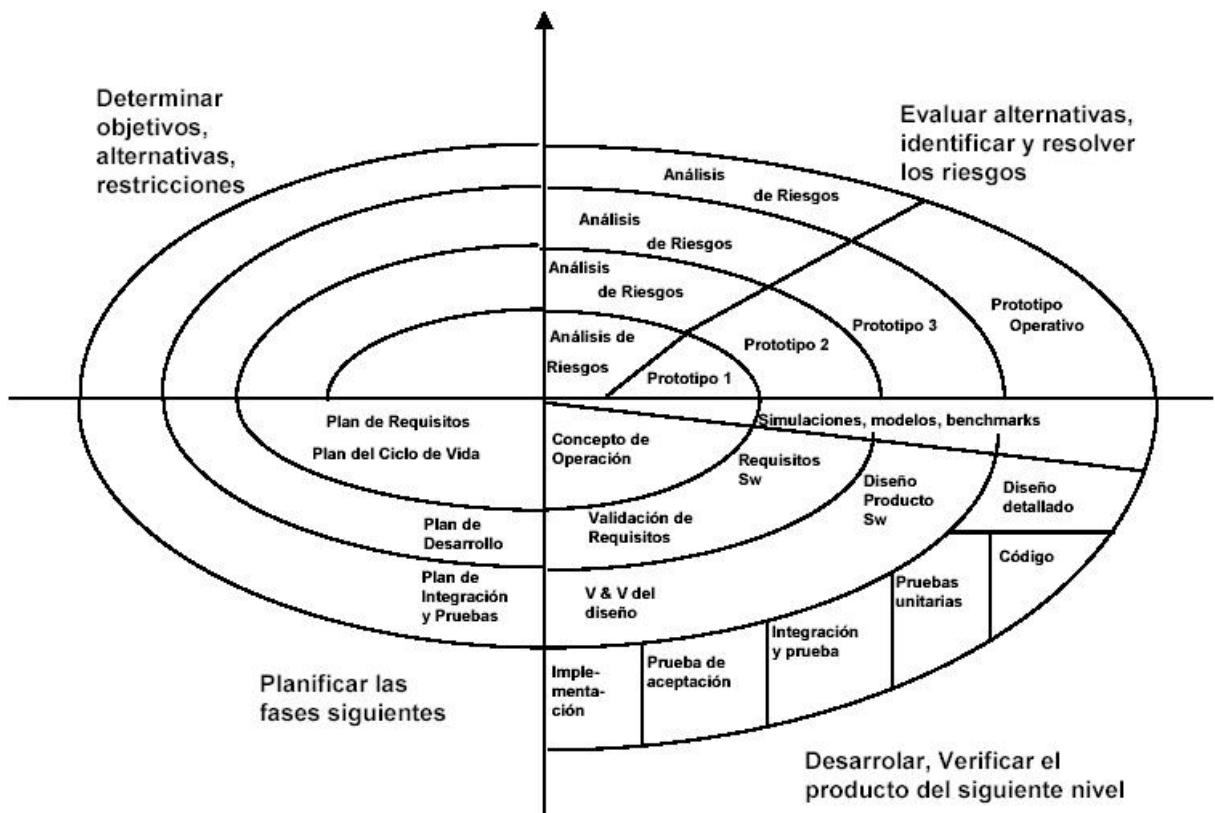


Figura 2.1: Modelo en espiral.

El ciclo de vida de este modelo está dividido en ciclos. Cada ciclo representa una fase del proyecto y está dividido, a su vez, en 4 partes. Cada una de las partes tiene un objetivo distinto:

- **Determinar objetivos.** Se establecen las necesidades que debe cumplir el sistema en cada iteración teniendo en cuenta los objetivos finales. Según avanzan las iteraciones aumenta el coste del ciclo y su complejidad.
- **Evaluar alternativas.** Se determinan diferentes formas de alcanzar los objetivos que se han establecido en la fase anterior. Se aborda el problema desde distintos puntos de vista, como, por ejemplo, el rendimiento del algoritmo en tiempo y espacio. Además, se consideran explícitamente los riesgos, intentando mitigarlos al máximo.

- **Desarrollar y verificar.** Se desarrolla el producto siguiendo la mejor alternativa para poder alcanzar los objetivos del ciclo. Una vez diseñado e implementado el producto, se realizan las pruebas necesarias para comprobar su funcionamiento.
- **Planificar.** Teniendo en cuenta los resultados de las pruebas realizadas, se planifica la siguiente iteración, se revisan los errores cometidos y se comienza un nuevo ciclo de la espiral.

2.4. Plan de trabajo

Para poder abordar el problema se han marcado una serie de subobjetivos ha completar. Dichos hitos son los siguientes:

1. Estudio y comprensión de la composición de un mapa y como construirlo. Nos apoyaremos en las herramientas ofrecidas por ROS y que resultaran básicas para este fin, dichas herramientas son *TF*¹ y *Costmap*².
2. Primer subobjetivo. Una vez conocido como funcionan los *costmap* procederemos a crear un pequeño nodo en el que se cree un mapa con las observaciones instantáneas que percibimos con el laser.
3. Segundo subobjetivo. Extender el algoritmo anterior para poder añadir y eliminar objetos según entren o salgan de la escena.
4. Tercer subobjetivo. Modificar el paquete *map_server* para que acepte varios mapas como entrada y estudiar la manera de mezclar los mapas entre sí.
5. Fase de pruebas. Se le pasará al paquete de navegación de ROS, *move_base*, el mapa resultante y se harán pruebas de navegación en el simulador y en el robot real.
6. Cuarto subobjetivo. Crear el mapa semántico, especificando las etiquetas naturales que tendrá una casa, salón, cocina, habitación... y usarlo para poder ir a la estancia que le indiquemos.

¹<http://wiki.ros.org/tf>

²http://wiki.ros.org/costmap_2D

Capítulo 3

Servidor de mapas dinámico

En este capítulo se explica la construcción y el funcionamiento del servidor de mapas dinámico. En primer lugar se explica que es, para qué se utiliza y cómo se construye un *costmap*. En segundo lugar se explica cómo se construyen los mapas que componen el algoritmo y por último se explica cómo se combinan estos mapas para generar el mapa usado en la navegación.

3.1. Costmap_2D

Un *costmap* es una estructura de datos ofrecida por ROS y compuesta por un grid de ocupación y los metadatos de este grid. Cada celda del grid toma valores entre 0 y 255, donde 0 corresponde a una celda vacía, los valores entre 1 y 254 representan la probabilidad de que una celda está ocupada y el valor 255 se reserva para el desconocimiento. Cada valor se asocia con un nivel de gris, como se puede ver en la imagen.

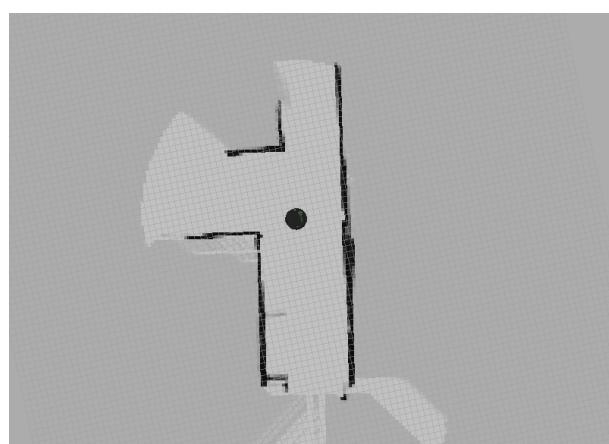


Figura 3.1: Ejemplo visual de un costmap.

Para poder representar la ocupación de un objeto en un *costmap* es necesario hacer uso de las transformadas entre frames que nos ofrece ROS.

tf

Cualquier robot está compuesto por multitud de piezas móviles, como puede ser la propia base del robot o la pinza de un brazo robótico. Cada una de estas piezas se pueden representar con un *frame*. Además existen también otros *frames* que pueden interesarnos, como puede ser el *frame* de world o el *frame* de map.

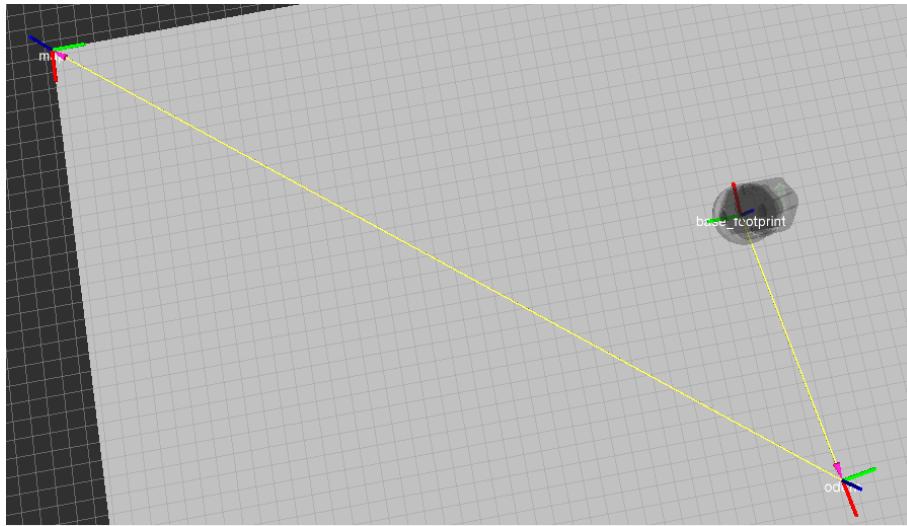


Figura 3.2: A la izquierda el frame map y a la derecha los frames de odom y base_footprint

Usamos las *tf* para poder representar información relativa a uno de estos frames. Esto puede sernos de utilidad, por ejemplo, si queremos conocer la posición de un objeto que hemos cogido con nuestra pinza respecto a la base de nuestro robot, o cuál es la posición relativa de un objeto que estamos percibiendo con el laser respecto a nosotros o respecto al mapa.

Cuando trabajamos con mapas es importante que todo lo que se representa en él sea respecto al frame map. De este modo nuestro mapa puede ser usado por otros nodos, como el nodo de navegación, o en cualquier otro escenario.

3.2. Tipos de mapas

En este apartado se describirá la metodología seguida para la construcción de cada uno de los tres mapas usados por el algoritmo.

Mapa estático

El mapa estático se caracteriza por incluir las partes inmutables del escenario, como son las paredes o las puertas. La mejor manera de construirlo es medir todo el escenario y crear el mapa usando una herramienta de diseño gráfico. En este caso se ha usado *Gimp*. Este mapa nos servirá como base para crear el mapa de largo plazo.

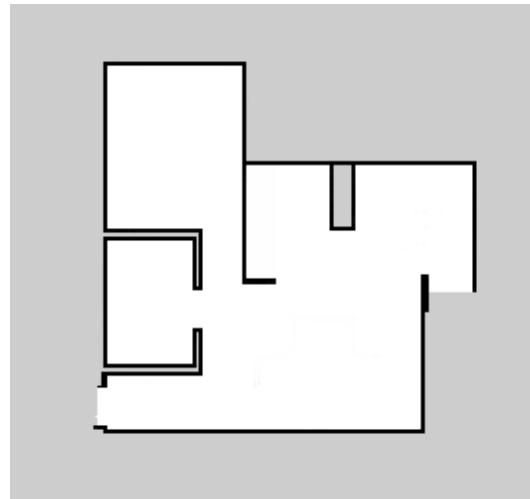


Figura 3.3: Mapa estático

Mapa de corto plazo

El mapa de corto plazo se caracteriza por ser un mapa en el que se representa los objetos que el robot va percibiendo. Este mapa se inicializa con el valor 255, lo que indica una incertidumbre total. En el instante en el que el algoritmo de construcción del mapa comienza a iterar comenzarán a corregirse estos valores iniciales, asignando el valor 0 a las celdas que corresponden con zonas libres e incrementando desde 0 hasta 254 el valor de las celdas que se perciben como ocupadas.

EJEMPLO CODIGO

El algoritmo propuesto destaca por la capacidad de no solo añadir objetos al mapa, si no ademas eliminarlos si los objetos desaparecen del lugar que ocupaban. Para ello se compara cada muestra de datos con el mapa que estamos generando y si en dicha muestra existen celdas libres que en el mapa están ocupadas se decrementa el valor de dicha celda en el mapa. La cuantia del decremento se puede modelar, consiguiendo asi que el robot olvide más lentamente o más rapidamente los objetos que desaparecen del

escenario.

EJEMPLO CODIGO

EJEMPLO GRAFICO

3.3. Colección de EKFs

Una vez explicada la adaptación del Filtro Extendido de Kalman, ahora hay que integrarlo con el resto del sistema. El EKF, por sí solo, únicamente permite hacer el seguimiento de un objeto. Para poder hacer el seguimiento de varios objetos simultáneamente se utiliza un conjunto de EKFs, cada uno de ellos asociado a uno de los objetos. El conjunto de filtros es dinámico pudiendo ajustarse un número mínimo de observaciones que siempre estarán presentes. En el caso de la pelota es uno, mientras que en el caso de las porterías es dos, poste izquierdo y derecho.

Para poder manejar dinámicamente el conjunto de filtros es necesario definir una serie de reglas para añadir un nuevo filtro o para actualizar/eliminar uno de los ya existentes. Hay que tener en cuenta que los filtros creados en el algoritmo puede que no representen un objeto real del entorno, sino que sea un falso positivo. En estos casos, lo ideal es detectarlo cuanto antes y eliminarlo.

Realizar el seguimiento de varios objetos es fácil si los objetos no son homogéneos, es decir, si tenemos un método capaz de distinguir cada uno de ellos. Por ejemplo, una pelota azul y una roja. El problema viene cuando realizar la distinción entre varios objetos no es posible. Como por ejemplo pueden ser las lecturas de un radar o los postes de las porterías. El pseudocódigo del algoritmo que se propone en esta investigación se puede ver en Algoritmo ???. Éste puede dividirse en cinco fases:

- La instrucción 1 representa la inicialización del algoritmo y se ejecutan una sola vez, al iniciar el algoritmo. Las instrucciones entre las líneas 2 y 22 se ejecutarán en bucle durante todo el período de vida del algoritmo.
- Entre 2 y 6 se predice el nuevo estado de los objetos que están siguiendo.
- En las líneas 7 y 14 se incorporan las nuevas observaciones al sistema, ya sea actualizando alguno de los objetos ya existentes o bien, si ninguno satisface las condiciones, creando uno nuevo.
- Las líneas 15 y 17 añaden ruido a los objetos sospechosos de ser falsos positivos.

- Por último, entre las líneas 18 y 22 se eliminan los objetos que tengan demasiada incertidumbre.

Además del estado y la incertidumbre de los objetos de seguimiento se guardan algunos datos más que los enriquecen y permite tomar mejores decisiones. Se guarda la marca de tiempo en la que se predijo por última vez el nuevo estado y la marca de tiempo de la última vez que se vio el objeto. También se guarda el *origen* de la observación, que puede ser local o remota. En este proyecto todas las observaciones que se utilizan son locales, pero ésto deja la puerta abierta a futuras mejoras, que se explicarán en el último capítulo. A continuación se detallan cada una de las fases del algoritmo y las operaciones que se realizan en ellas.

3.3.1. Inicialización de los filtros

La inicialización del algoritmo sólo se produce la primera vez. El algoritmo permite definir un número mínimo de instancias que siempre estarán presentes en la colección. Esta propiedad es útil cuando el sistema donde se integra el algoritmo no permite valores nulos. Un ejemplo sencillo es el caso de la pelota en un partido de la RoboCup. Se sabe que la pelota está siempre presente en el campo. Puede que no se conozca su localización o que esté oculta por el resto de jugadores, pero la pelota está en el terreno de juego. En este caso tiene sentido tener como mínimo una instancia de la pelota en nuestro sistema. En el caso de no haber visto la pelota en mucho tiempo, la instancia tiene almacenada la última posición donde se vio, pero la incertidumbre será tan alta que permite discernir si esa posición es fiable o no.

Es necesario inicializar estas instancias con unos valores arbitrarios. La posición inicial de estas instancias no es importante, lo único que hay que tener cuidado es de inicializar la incertidumbre con un valor suficientemente alto para que la instancia no sea fiable y el robot pueda actuar en consecuencia, lo que sería siguiendo la línea del ejemplo anterior, iniciando la búsqueda de la pelota.

Otro momento en el que hay que inicializar filtros es cuando tenemos una observación nueva que no corresponde a ninguno de los filtros presentes. Esta operación se describe en 3.3.3. En este caso, la inicialización es más sencilla. El filtro se inicializa directamente con los datos de la observación.

3.3.2. Predicción de nuevos estados

En esta fase, instrucciones entre las líneas 2 y 6, se predice el nuevo estado para cada uno de los objetos sobre los que se está realizando el seguimiento. El estado y la incertidumbre se actualizan mediante la fase de predicción del EKF que ya se ha explicado anteriormente. También se actualiza la marca de tiempo en la que se realiza la predicción.

(ab)
 Ellos
 filtros
 está
 siguiendo
 dos
 objetos
 y del
 sebot
 nueve
 ha
 drenado
 hacia
 elos
 incertidumbre

Figura 3.4: Comportamiento de los filtros en la fase de predicción

La figura 3.4 muestra el comportamiento de los filtros en esta fase. En la imagen de la izquierda el robot está realizando el seguimiento de dos objetos. Cada uno tiene una posición y una incertidumbre. El robot se encuentra moviéndose en dirección a ellos. En la imagen de la derecha se puede ver el resultado de la fase de predicción. Los objetos se encuentran ahora en una posición más cercana al robot, hay que tener en cuenta que las posiciones de los objetos son relativas a éste, y la incertidumbre de los filtros aumenta en consecuencia.

3.3.3. Incorporación de observaciones

En esta fase se realizan varias operaciones para decidir si actualizar un filtro ya existente o, si ninguno satisface las condiciones, crear uno nuevo. Para ello se emparejan las observaciones con los filtros y se calcula el valor de similitud de ellos. Seguidamente se escoge la pareja que más se parezca y, si se cumplen las condiciones necesarias, se actualiza el filtro con la observación. En caso de no cumplirse las condiciones, se crea un filtro nuevo a partir de la observación. Se repite este procedimiento con todas las

observaciones.

Emparejamiento

El criterio de emparejamiento utilizado en nuestro algoritmo se basa en la posiciones e incertidumbres tanto de la observación como de cada uno de los filtros. Vincula una observación a uno de los filtros. Para este criterio utilizamos la función de *verosimilitud*, 3.1. Esta función permite realizar inferencias acerca del valor de los parámetros de un modelo estadístico a partir de un conjunto de observaciones. En este paso se compara la observación obtenida con todos los filtros y se escoge el que tenga mayor similitud. s_1 y P_1 son el estado y la matriz de covarianza de un filtro, respectivamente, y s_2 y P_2 representa lo mismo pero de la observación.

$$\text{likelihood} = \frac{e^{\det(-\frac{1}{2}(s_1 - s_2)^T(P_1 + P_2)^I(s_1 - s_2))}}{\sqrt{\det(2\pi(P_1 + P_2))}} \quad (3.1)$$

El estado y la matriz de covarianza de cada uno de los filtros no tiene el tamaño original de 4×1 y 4×4 . Antes de utilizarse en la ecuación se simplifican las matrices eliminando los componentes de velocidad V_x y V_y para que el tamaño de las matrices coincida con los tamaños de las observaciones y pueda aplicarse la ecuación.

(a)
 F
 r
 e
 s
 i
 b
 l
 e
 d
 e
 r
 e
 a
 c
 i
 o
 n
 u
 n
 d
 a
 o
 b
 s
 e
 r
 v
 a
 c
 i
 o
 n
 c
 o
 l
 o
 r
 o
 n
 u
 n
 u
 e
 v
 o
 f
 i
 l
 t
 r
 o

Figura 3.5: Distintas situaciones del emparejamiento entre una observación y un filtro

En la figura 3.5 se puede ver un ejemplo gráfico más fácil de entender. En las imágenes aparecen una colección de tres filtros y dos situaciones distintas al incorporar dos observaciones distintas. En la figura 3.5a se ve claramente que la observación se emparejaría con el filtro 1. En la figura 3.5b la situación no está tan clara. La observación se encuentra prácticamente a la misma distancia de todos ellos. La observación se emparejaría con el filtro 3 por ser el que más incertidumbre tiene.

Matching

Una vez que se ha escogido uno de los filtros existentes hay que decidir si actualizarlo con la nueva observación o crear una filtro nuevo. El método que utilizamos es bastante simple e intuitivo. Se comprueba si las elipses generadas a partir de la media y covarianza de la observación y del filtro intersectan, en cuyo caso se considera que ambas se corresponden con el mismo objeto y se actualiza el filtro con la observación. En caso contrario, se crearía un nuevo filtro.

Al calcular la intersección de las elipses, el tamaño de éstas se multiplica por 4 siguiendo la desigualdad de Chebyshev. Este factor es un resultado que ofrece una cota inferior a la probabilidad de que el valor de una variable aleatoria con varianza finita esté a una cierta distancia de su esperanza matemática. En la siguiente ecuación, se puede ver el valor de la desigualdad de Chebyshev para un valor k veces la desviación típica partiendo de la media. En nuestro caso, hemos escogido el valor 4 y devuelve un resultado muy próximo a 94 %.

Población mínima	# de desviaciones típicas desde la media
50 %	$\sqrt{2}$
75 %	2
89 %	3
94 %	4
96 %	5
97 %	6
$1 - \frac{1}{k^2}$	k
1	$\frac{1}{\sqrt{1-l}}$

Esta tabla se puede leer: para cualquier distribución con una desviación estándar definida, la cantidad de datos que se encuentran a una distancia k veces la desviación típica de la media es, al menos, como se indica en la tabla. Es decir, a una distancia 4 veces la desviación típica partiendo de la media se encuentran el 94 % de los datos.

La intersección de dos elipses no es un cálculo sencillo. Hay que tener en cuenta distintos casos: cuando la distancia entre las elipses es demasiado grande para intersectar, cuando el centro de una ellas está contenida dentro de la otra que intersecta siempre, o cuando simplemente intersectan en uno, dos, tres o cuatro puntos. Las elipses están definidas por 5 parámetros: x e y , que son las coordenadas del centro; a b , que son los semi-ejes mayor y menor respectivamente, y λ que es la rotación.

- Primero se comprueba que las elipses estén a una distancia mínima en la puedan intersectarse o una esté contenida dentro de la otra. En caso de no cumplirse este requisito, se puede afirmar que las elipses no intersectan. Por lo que si la distancia entre los centros es mayor que la suma de los dos semi-ejes de mayor tamaño, las elipses no intersectan.

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} > a_1 + a_2 \rightarrow \text{no intersectan} \quad (3.2)$$

- Si el centro de una de las elipses está contenido dentro de la otra, entonces se puede afirmar que las elipses intersectan o una esté contenida totalmente dentro de la otra, situación igualmente válida. p y q son las coordenadas del centro de la otra ellipse.

$$\frac{((p \cos \lambda + q \sin \lambda) - x)^2}{a^2} + \frac{((p \sin \lambda - q \cos \lambda) - y)^2}{b^2} \leq 1 \quad (3.3)$$

- Por último, si en caso de no poder afirmar nada aún hay que resolver los puntos de intersección de las elipses. A partir de las ecuaciones de la ellipse en su forma

$$\begin{aligned} ax^2 + bxy + cy^2 + dx + ey + f &= 0 \\ a'x^2 + b'xy + c'y^2 + d'x + e'y + f' &= 0 \end{aligned} \quad (3.4)$$

se obtiene una ecuación de cuarto grado que puede resolverse en forma cerrada. Esto quiere decir que se puede resolver en términos de funciones y operaciones matemáticas elegidas de un conjunto limitado. Una ecuación de cuarto grado tiene la siguiente forma

$$ax^4 + bx^3 + cx^2 + dx + e = 0 \quad (3.5)$$

Se pueden hacer una serie de comprobaciones rápidas en la ecuación para averiguar tiene soluciones no triviales.

- Si $e = 0$ una de las raíces es 0, por lo tanto intersectan por lo menos en un punto.
- Si $a = 0, b \neq 0$, se trata de una ecuación cúbica que siempre tiene raíces.
- Si $a = 0, b = 0, c \neq 0$, ecuación cuadrática. Sencillo de resolver mediante la fórmula cuadrática $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
- Si $a = 0, b = 0, c = 0$ y $d \neq 0$, todas las rectas con pendiente distinta de 0 tienen una raíz

En cualquier otro caso hay que resolver la ecuación de cuarto grado, operación más costosa computacionalmente.

3.3.4. Descarte de filtros

Para obtener un rendimiento mayor es necesario tener el menor número de filtros posibles ejecutándose al mismo tiempo. En caso de no borrarse ningún filtro, al ser una colección dinámica, el tamaño podría crecer desmesuradamente en un corto espacio de tiempo perjudicando enormemente el rendimiento del algoritmo. Normalmente se quiere eliminar un filtro cuando se trata de un falso positivo o cuando un objeto sobre el que se estaba realizando el seguimiento desaparece del entorno.

La colección de filtros tiene un número mínimo de objetos sobre los que hace el seguimiento. Este límite es útil para situaciones en las que se puede afirmar que siempre estará presente en el entorno ese mínimo número de objetos, como, por ejemplo, cuando se realiza el seguimiento de una pelota en un partido de la RoboCup. Siempre hay una pelota, aunque el robot no la haya localizado o la haya perdido de vista. En el caso de haber localizado en algún momento la pelota, pero haberla perdido de vista hace un tiempo, el filtro almacena la última posición conocida de la pelota. Al haberse perdido de vista hace tiempo la pelota tendrá una incertidumbre alta. Esta información es muy útil para comportamientos de más alto nivel ya que pueden basar sus decisiones, además de en la posición de la pelota, en su incertidumbre.

El método utilizado es bastante simple. Si hay un número mayor de filtros que el número mínimo especificado, se compara el área de la elipse con un umbral determinado y, en caso de superarse, el filtro se elimina. Este umbral se elige de forma arbitraria. El método usado para que sea sencillo elegir el umbral es crearlo a partir del área de una circunferencia. Dependiendo del modelo del objeto que se utilice se puede tener un umbral más alto o más bajo, pudiendo adaptarse el algoritmo a mayor cantidad de problemas.

$$\underbrace{2ab}_{\text{Área elipse}} > \underbrace{\pi r^2}_{\text{Área circunferencia}} \rightarrow \text{eliminar filtro} \quad (3.6)$$

3.4. Adición de incertidumbre

El algoritmo del Filtro de Kalman está pensado para tener una fuente de datos continua. En cada iteración se actualiza la información del filtro con las nuevas observaciones. El problema es que normalmente un robot tiene una visión parcial del problema y puede no percibir todos los objetos todo el tiempo. El algoritmo básico de

Kalman no contempla esta situación. La única manera de aumentar la incertidumbre es mediante el movimiento del robot o mediante otras observaciones. Pero, ¿qué pasa cuando un robot está parado y no detecta un objeto? La forma más sencilla de entenderlo es mediante un ejemplo.

Durante un partido de fútbol suponemos que el portero tiene localizada la pelota con muy poca incertidumbre. En ese momento se empiezan a juntar jugadores alrededor de ella, provocando que el portero la pierda de vista. Si el portero está estático, al no tener ninguna observación de la pelota, el filtro mantiene el mismo estado y la misma incertidumbre a lo largo del tiempo. En ese período de tiempo lo más probable es que la pelota se haya movido del sitio en el que estaba, pero el portero sigue con una incertidumbre muy baja lo que le lleva a tomar decisiones como si la pelota siguiese en ese sitio. Decisiones probablemente desacertadas.

Otro comportamiento que se ha observado que no favorece el seguimiento de objetos con movimiento se produce cuando se observa un objeto durante un período de tiempo seguido en el mismo sitio. El filtro tiende a disminuir demasiado la incertidumbre provocando que no sea reactivo a nuevos movimientos del objeto. En nuestra implementación en concreto, se generaba un nuevo filtro para el mismo objeto.

En el primero de los casos, el comportamiento buscado es aumentar la incertidumbre del objeto cuando éste no se visualice. De esta manera, el portero, podría reaccionar frente a la situación de perder de vista la pelota entre los jugadores. En todo momento sigue recordando la última posición en la que ésta se encontraba, pero es capaz de reaccionar mucho antes a las nuevas situaciones que se presenten. Por otro lado, en el segundo caso, también se busca introducir un poco de ruido en los filtros para mantenerlos más reactivos, sobre todo aquellos que tienen más movimiento.

Se han definido tres niveles de adición de incertidumbre distintos dependiendo de la situación:

1. El filtro se ha actualizado con una observación. En este caso se añade un mínimo de ruido para mantener el filtro reactivo. La cantidad de incertidumbre depende de los valores con los que se ha modelado el objeto.

2. El filtro no se ha actualizado porque no se encuentra en el *frustum* de la cámara. De un objeto en esta situación no se puede afirmar nada, si sigue en el mismo lugar o se ha desplazado, por lo que se añade una cantidad *estándar* de ruido.
3. Si un filtro no se actualiza y debería estar en la imagen. Esta situación la tomamos como un falso positivo. Puede que en realidad el filtro no sea un falso positivo y sea una ocultación o que el detector no lo localice en la imagen, pero es muy difícil discernir entre estas posibilidades.

En esta fase se decide en cuál de las tres situaciones se encuentra cada uno de los filtros y se suma el ruido correspondiente a la matriz de covarianza del filtro:

$$S' = S + \text{noise}^{\# \text{situación}} I \quad (3.7)$$

3.5. Objetos complejos

En ocasiones se quieren seguir objetos que están compuestos de formas más simples y más sencillas de reconocer. Esto se hace porque el objeto es demasiado grande y no puede ser detectado con una sola imagen, o bien porque es demasiado complejo para resolver el problema de una vez y se divide en problemas más sencillos.

En el caso de las porterías en el fútbol se dan los dos casos expuestos. Detectar una portería al completo es una tarea bastante difícil, ya que habría que detectar los dos postes al mismo tiempo para poder detectar que ese objeto es una portería. Pero los postes están separados por una distancia considerable y dependiendo de la distancia a la que se encuentre el robot de éstos, puede ser imposible ver los dos postes al mismo tiempo. Por esto se ha simplificado el problema inicial en hacer el seguimiento de los postes y, a partir de estos, averiguar los dos postes con más probabilidades de ser una portería.

Figura 3.6: Distinguir poste izquierdo de poste derecho

Para ello se crea una colección de filtros con un mínimo de dos instancias de los postes de la portería. Simplemente se siguen los postes sin tener en cuenta si son de una portería o no. *A posteriori* se realiza una serie de cálculos para, de entre todos los postes de la colección, decidir cuáles son más probables de ser una portería. Se realiza la siguiente operación a cada uno de las combinaciones de postes que haya en

la colección. s es la posición del poste y P su covarianza.

$$\begin{aligned} s' &= f(s_1 - s_2) \\ P' &= A(P_1 + P_2)A^T \end{aligned} \tag{3.8}$$

donde $f()$ es la función para pasar del espacio euclídeo $\mathbb{R}^2 \rightarrow \mathbb{R}^1$ y A es la matriz jacobiana de la función

$$\begin{aligned} f(s) &= \sqrt{s_x^2 + s_y^2} \\ A &= \left(\frac{s_x}{\sqrt{s_x^2 + s_y^2}}, \frac{s_y}{\sqrt{s_x^2 + s_y^2}} \right) \end{aligned} \tag{3.9}$$

Una vez hecho esto, para cada par de postes de la colección se tiene la distancia entre ellos y la covarianza de ésta. Para decidir qué par de postes es el que más se parece a una portería, se utiliza el mismo método que cuando se emparejan las observaciones con los filtros, se calcula de la distancia de *Mahalanobis*.

$$d(x, y) = \sqrt{(x - y)^T S^{-1}(x - y)} \tag{3.10}$$

Esta distancia es un método estadístico descriptivo que aporta una medición relativa de varios puntos a uno común. El valor calculado no tiene unidades de medida, lo que lo convierte en un método muy útil para comparar distancias teniendo en cuenta la incertidumbre. En nuestro caso se calcula el error de distancia entre los dos postes, $1500 - dist(post1, post2)$.

Una vez escogidos los postes que van a representar la portería, hay que diferenciar el izquierdo del derecho. Se calcula mediante el producto vectorial entre los dos postes. Si el producto entre el poste 1 y el poste 2 es positivo, entonces el poste 1 es el derecho y el poste 2 es el izquierdo. Si el producto es negativo, es a la inversa. La figura 3.6 muestra el cálculo.

Capítulo 4

Experimentos

En este capítulo se muestra el resultado de una serie de experimentos que ayudan a validar y verificar el algoritmo implementado. Como se ha comentado anteriormente todas las pruebas se han hecho en un simulador en el entorno de la RoboCup SPL.

A pesar de que muchas de las gráficas muestran únicamente la calidad de los objetos, en estos experimentos se hablará indistintamente de la calidad o de la incertidumbre de estos. Decir que disminuye la calidad o que aumenta la incertidumbre significa lo mismo, que la información del objeto es menos fiable.

4.1. Seguimiento de un objeto

En este experimento se comparan los resultados obtenidos a partir de las observaciones obtenidas del detector y esas mismas observaciones procesadas mediante el algoritmo de seguimiento, figura 4.1. En este experimento se ha colocado una pelota a cierta distancia del robot y se ha teleoperado el robot de forma que vea la pelota en todo momento y se ha golpeado la pelota. Los datos que muestra la gráfica es la distancia a la que se encuentra la observación en cada instante de tiempo. La distancia está medida en milímetros.

Las observaciones sin procesar son las de color verde. Se puede ver que la gráfica esta llena de picos y no es en absoluto estable. Ello se debe al ruido en las observaciones por causa del movimiento del robot e imprecisiones en la detección de la pelota. De color rojo se puede ver la observación procesada por el algoritmo de seguimiento. Se puede apreciar a simple vista como se mitigan las imprecisiones causadas por el ruido de las observaciones. Entre iteración e iteración existe una pequeña variación de la posición, pero bastante más reducida que las observaciones en crudo.

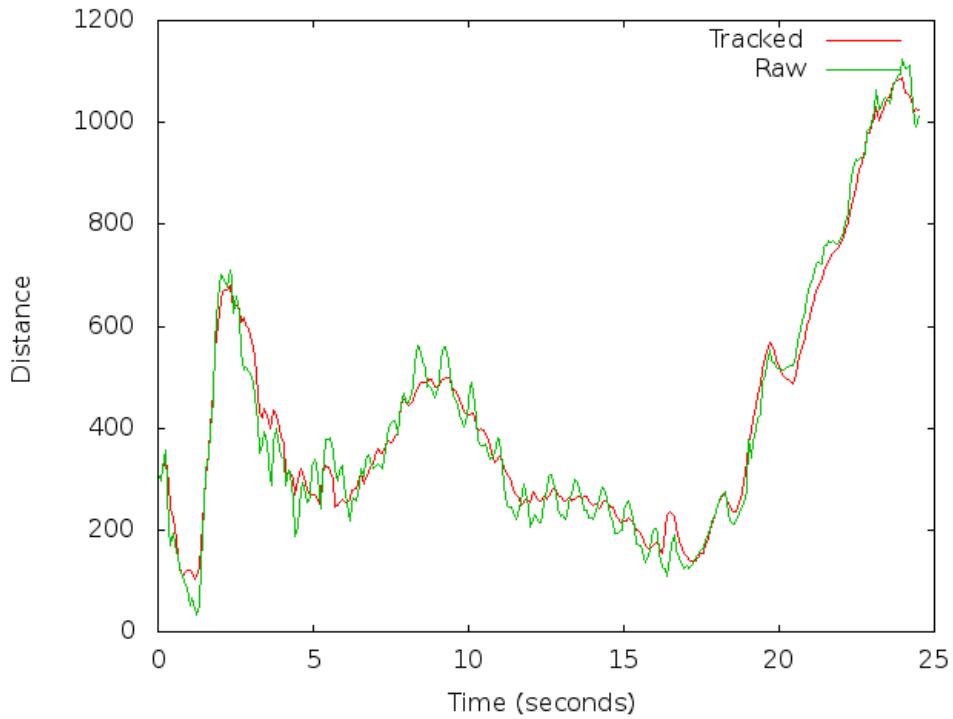


Figura 4.1: Distancia a la que se encuentra un objeto haciendo el seguimiento y tomando las medidas del detector directamente

Uno de los efectos colaterales del procesado de las observaciones es la pérdida de reactividad frente a cambios en la posición del objeto. Se puede ver que los cambios en la observación procesada se producen más tarde en el tiempo que la observación en crudo. Este genera un pequeño desfase entre la posición real y la calculada por el algoritmo. Esta consecuencia es inevitable. Al modelar el objeto que se va a seguir, hay que elegir entre estabilidad y reactividad. Un objeto más estable se verá afectado en menor medida por el ruido de las observaciones, pero tardará más en reaccionar frente a un cambio en la posición del objeto. Si el objeto es más reactivo, ocurre lo contrario.

4.2. Descubrimiento de falsos positivos

En este experimento se muestra como reacciona el algoritmo frente a los falsos positivos. La pelota se ha situado enfrente del robot. Éste hace barridos de un lado a otro con la cabeza de forma que a veces la ve y a veces no. En un momento dado, cuando el robot no está viendo la pelota, se quita de esa posición.

La gráfica de la figura 4.2 presenta la calidad del objeto a lo largo del tiempo. No se aprecia bien en la gráfica, pero en los primeros instantes de ejecución la calidad pasa de 0 a un valor cercano a 1 en cuanto el robot divisa la pelota. Aproximadamente a partir del segundo 5 el robot deja de ver la pelota y la calidad empieza a descender. La curva de descenso es bastante liviana porque no hay ruido añadido causado por el movimiento del robot. Ese descenso de la calidad se debe únicamente a los parámetros con los que se ha modelado la pelota.

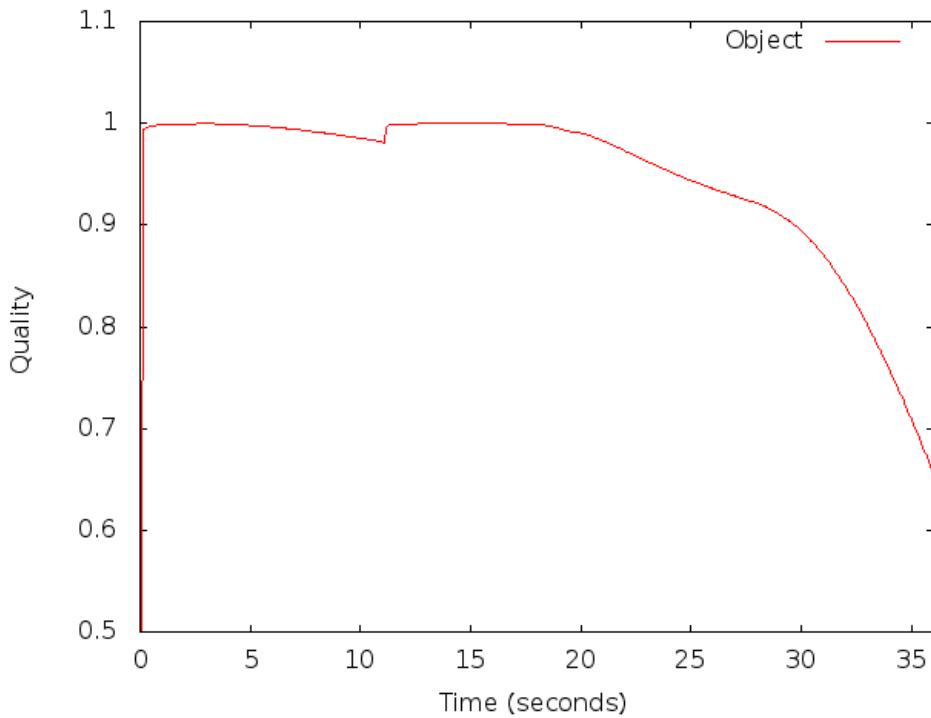


Figura 4.2: Gráfico del experimento sobre los falsos positivos

Un poco después del segundo 10 se vuelve a localizar la pelota y la calidad vuelve a subir. Cuando se pierde otra vez de vista se puede apreciar una curva un poco distinta de la anterior. Ello se debe a que el objeto se encuentra en el límite de la imagen. Lidiar con estas situaciones es bastante difícil. El objeto se encuentra dentro del *frustum* de la cámara, pero el detector no es capaz de reconocerlo. Esta situación hace que se considere el objeto como un falso positivo, lo que causa que la incertidumbre crezca más rápidamente. Esta situación prácticamente no afecta al rendimiento del algoritmo, pues se produce durante un espacio muy corto de tiempo. En cuanto el objeto se encuentra fuera del área de visualización de la cámara, éste deja de considerarse un falso positivo y vuelve a aumentar la incertidumbre al ritmo habitual. En la parte final de la gráfica se puede ver como cae en picado la calidad. Aquí es donde se ha retirado

la pelota del sitio en el que estaba y el robot esta viendo que la pelota no se encuentra donde debería estar.

4.3. Seguimiento de objetos con falsos positivos

En este experimento se demuestra el comportamiento del algoritmo cuando se está haciendo el seguimiento de un objeto y aparece un falso positivo. Éste hay que detectarlo y deshacerse de él cuanto antes para evitar equivocaciones. En esta prueba se ha colocado una pelota a cierta distancia del robot. Éste camina en línea recta hacia la pelota pudiendo verla en todo momento. En cierto momento aparece otra pelota, que simula ser un falso positivo, y desaparece. La gráfica 4.3 recoge la calidad de las dos pelotas a lo largo del tiempo.

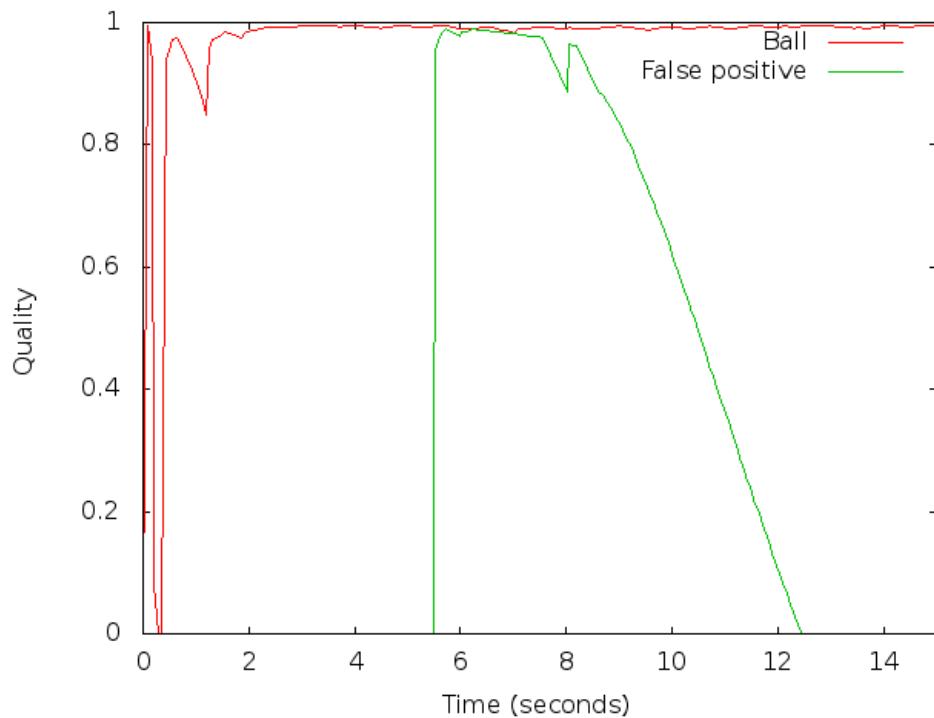


Figura 4.3: Seguimiento de la pelota con un falso positivo

La línea de color rojo representa la calidad de la pelota. Al estar viéndola en todo momento, la calidad es muy alta. Tiene ligeras fluctuaciones que son causadas por el ruido de la observación y el desplazamiento del robot. La línea verde simula el falso positivo. Casi en el segundo 6 se detecta una posible pelota y se incorpora a la colección de pelotas. El falso positivo es detectado durante un breve período de tiempo y luego

desaparece. Durante el tiempo que es detectado, no se llega a apreciar bien, pero la calidad del falso positivo es ligeramente menor que la calidad de la pelota real. Esto se debe a que la pelota real ha sido verificada durante muchas iteraciones anteriormente.

A partir del segundo 8 se pierde de vista por completo el falso positivo. En este momento, al encontrarse la última posición conocida del falso positivo dentro de la imagen, pero no encontrarse una observación en ese lugar, la calidad comienza a descender radicalmente. Este hace que se acabe por eliminar en poco más de 4 segundos.

Como se puede ver con este experimento, el algoritmo es robusto frente a falsos positivos. En el momento que se detecta que un filtro es un falso positivo o un objeto que ya no se encuentra en la imagen, se aumenta la incertidumbre mucho más rápido lo que hace que se acabe eliminando el filtro en un período corto de tiempo.

Existe un inconveniente muy difícil solucionar: las oclusiones. En caso de tener un objeto ocluido, éste se trataría como si fuese un falso positivo. Se tendría que utilizar algún método más elaborado para detectar si se puede tratar de una oclusión o no.

4.4. Seguimiento de objetos complejos

Este experimento muestra el comportamiento del algoritmo cuando se hace el seguimiento de un objeto complejo, es decir, formado por varios objetos simples. El objeto en cuestión es la portería, formada por dos postes. El criterio para identificar una portería es simple: se comprueba la distancia entre los dos postes y las incertidumbres de estos.

Para este experimento se ha colocado un robot en el centro del campo mirando a una de las porterías. En el medio de la portería se ha colocado un poste adicional para que el robot vea tres postes y tenga que decidir cuáles son los que mejor identifican una portería. Durante todo el tiempo de la prueba se ha movido la cabeza de izquierda a derecha. La gráfica 4.4 muestra las calidades de cada uno de los postes.

Las calidades de los postes aumentan y descienden de manera independiente al perderse de vista y volver a detectarse. A pesar de que las calidades varían, en todo momento el algoritmo detecta correctamente los postes de la portería, que son el 1 y el 3.

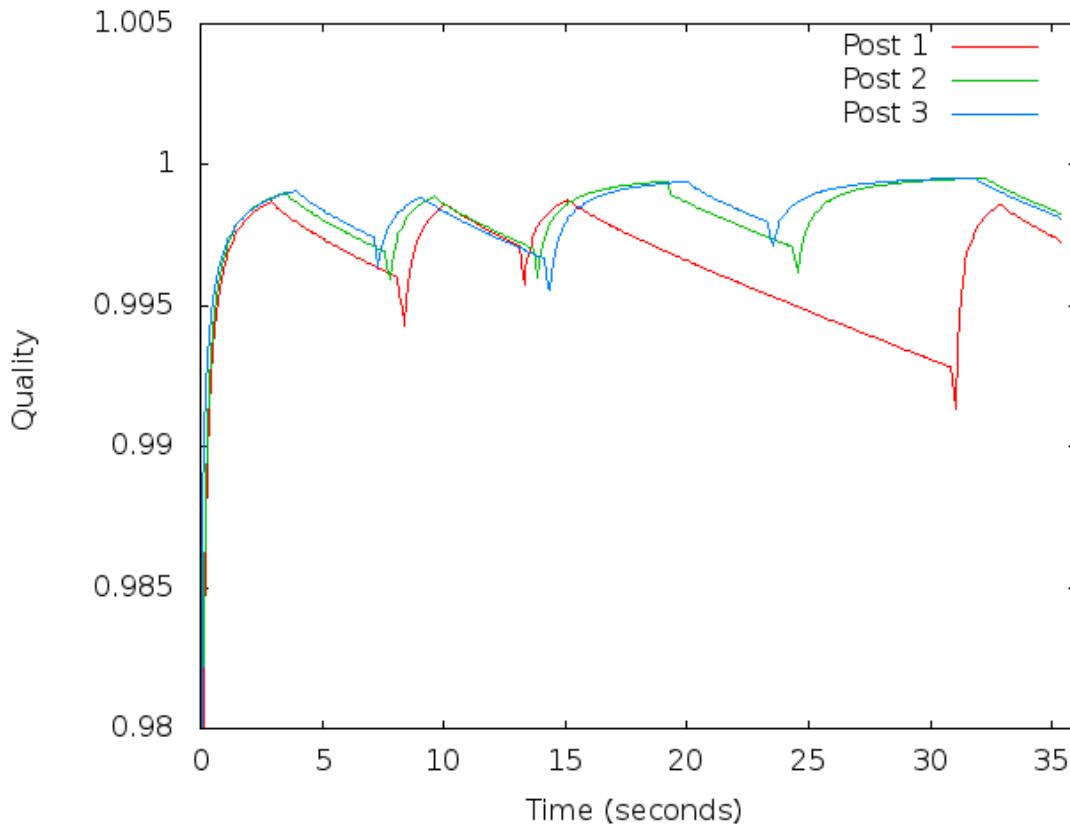


Figura 4.4: Gráfico del experimento de seguimiento de objetos complejos

Solamente en las primeras iteraciones, cuando la incertidumbre de los postes es bastante baja, hay algunos resultados erróneos. Se ha calculado que tiene aproximadamente un 99 % de fiabilidad.

Entre los segundos 25 y 30 se paró la cabeza del robot de forma que sólo viese uno de los postes de la portería y el poste falso. A pesar de disminuir la calidad del otro poste, la detección de la portería sigue siendo correcta.

Como ocurre en el primer experimento, se puede apreciar un comportamiento anómalo justo antes de volver a detectar los postes, una especie de "V". La calidad baja rápidamente y luego vuelve a subir. Al estar el poste en el límite de la imagen, la posición se encuentra dentro del frustum de la cámara, pero el detector no reconoce el objeto como un poste, por lo que se trata como si fuese un falso positivo.

4.5. Corrección de la velocidad de un objeto

En este experimento se compara el resultado de actualizar la velocidad de un objeto cuando se interactúa con él y cuando no. La acción que se realiza es la de golpear la pelota. En este experimento se ha posicionado la pelota a una distancia cercana al robot y se ha activado el comportamiento *Striker2*. Este comportamiento es el que utilizan los delanteros durante un partido. Realiza varias tareas complejas como buscar la pelota, aproximarse a ella y colocarse en posición para chutar entre otras. La figura 4.5 muestra el resultado del experimento, donde se puede ver el error en la posición de la pelota.

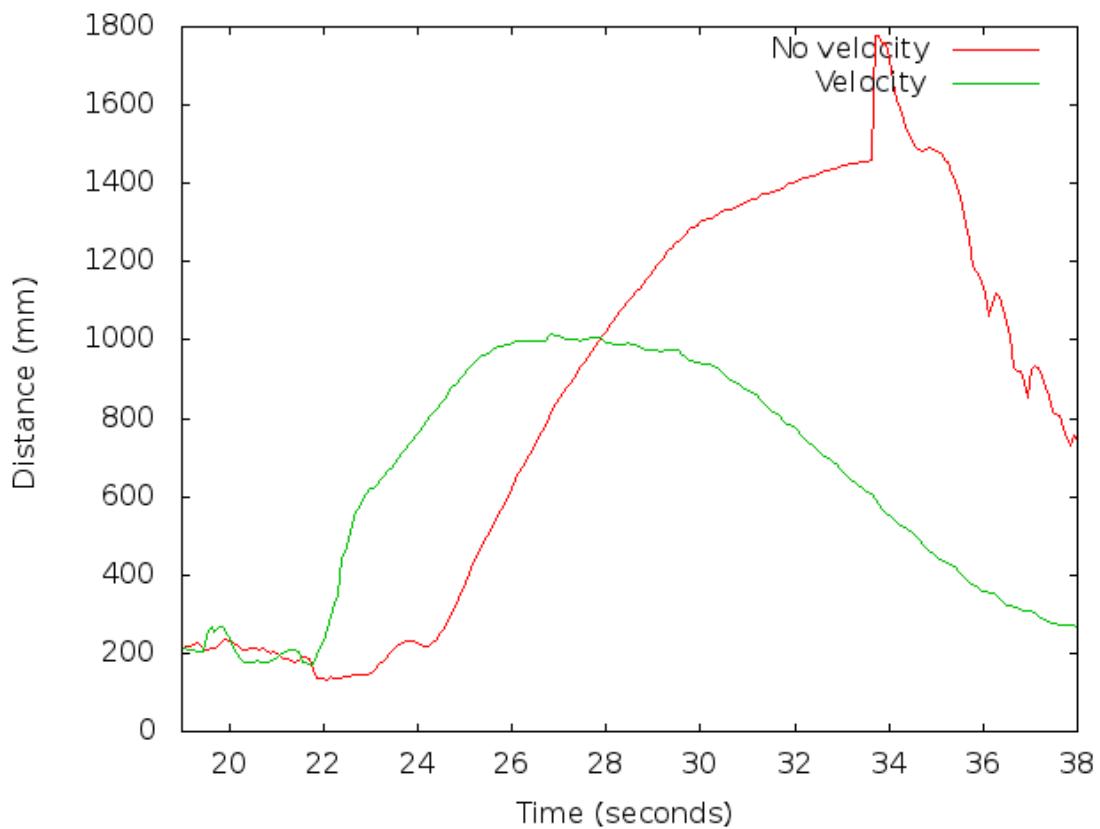


Figura 4.5: Corrección de la velocidad de un objeto

La línea roja representa el experimento en el que no se ha actualizado la velocidad de la pelota cuando se la ha golpeado. En el momento en que se chuta, aproximadamente el segundo 24, se puede ver como empieza a aumentar en gran medida el error de la posición. Ello se debe a que la pelota se pierde de vista demasiado rápido y el robot *cree* que sigue en la misma posición en la que estaba, justo enfrente. Únicamente tras un tiempo prudencial empieza a buscarla de nuevo, encontrándola otra vez en el segundo 34. Durante esos 10 segundos el robot no sólo tenía una posición errónea de la pelota

sino que se quedaba sin hacer nada, esperando encontrar la pelota en una posición que ya no está.

La línea verde representa el experimento en el que sí se actualizó la velocidad de la pelota. En este caso es en el segundo 22 cuando se golpea la pelota, y entonces el error de la posición también aumenta igual que antes, pero lo hace con antelación. Esto se debe a una desincronización entre el momento del golpeo y la actualización de la velocidad. Lo ideal sería que se actualizase la velocidad en el momento exacto del golpeo, pero el sistema robótico utilizado no permite detectar tal situación, por lo que se actualiza la velocidad un poco antes del golpeo real de la pelota, justo en el momento en el que se decide dar una patada a la pelota.

Al actualizar la velocidad de la pelota, ésta comienza a alejarse del robot. El módulo de atención visual que utiliza el comportamiento *Striker2* se ve obligado a levantar la cabeza del suelo para mirar las nuevas posiciones de la pelota y, de ese modo, es capaz de ver la pelota real alejándose del robot. Por esto se detecta mucho antes la pelota y se pueden tomar nuevas decisiones mucho más rápido. El algoritmo predice la nuevas posiciones provocando un mejor comportamiento del sistema. Si antes fueron necesarios 10 segundos para localizar de nuevo la pelota, al actualizar la velocidad de la pelota en poco más de 4 segundos el robot ha localizado de nuevo la pelota y puede seguir jugando.

4.6. Tiempos de ejecución

Por último se mide el tiempo de ejecución del algoritmo cuando se estima la posición de varios objetos, figura 4.6. En este experimento se han introducido uno a uno objetos en el campo de visión del robot hasta un total de 10. Al cabo de un corto período de tiempo, de la misma manera, se han retirado uno a uno los objetos.

El tiempo de ejecución aumenta según aparecen objetos en el campo de visión del robot. Se puede ver que, para 10 instancias, el algoritmo se ejecuta en menos de 1 milisegundo. Cuando se empiezan a retirar los objetos el tiempo vuelve a decrecer al mismo ritmo que subía al principio del experimento. Estos tiempos son suficientes para ejecutar el algoritmo en tiempo real.

Los resultados obtenidos son satisfactorios. Al tener más objetos en el campo de visión, el algoritmo tarda más en ejecutarse, pero el tiempo de ejecución aumenta, no de manera lineal, pero a un ritmo sostenible.

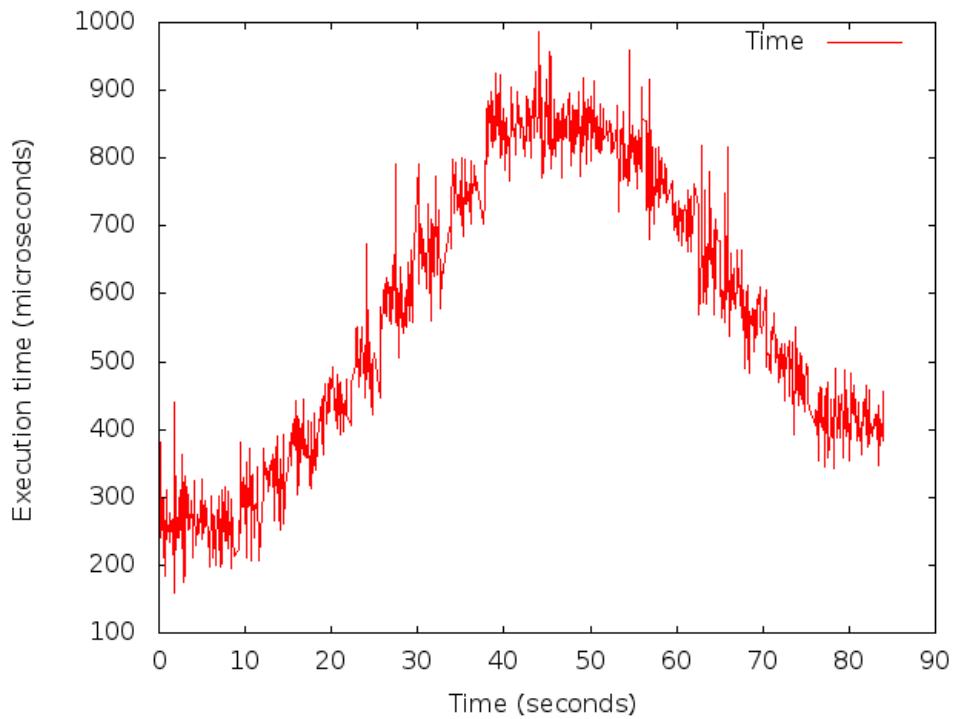


Figura 4.6: Gráfico con tiempos de ejecución

Capítulo 5

Conclusiones y trabajos futuros

Una vez presentados los detalles del algoritmo desarrollado, su validación y verificación, en este último capítulo se exponen las conclusiones y se hace autocrítica del trabajo. Para finalizar, se proponen una serie de mejoras y ampliaciones que mejoren el comportamiento del algoritmo.

5.1. Conclusiones

En este proyecto se ha desarrollado un algoritmo para hacer un seguimiento de los objetos más importantes del entorno de un robot. Como se ha explicado en capítulos anteriores, es muy importante filtrar la información recibida por parte de los sensores. En el caso de la cámara, la información se filtra dos veces. Una primera vez para detectar el objeto en cuestión en la imagen y otra para filtrar esa información y, en la medida de lo posible, reducir al máximo el ruido existente en las observaciones. De esta manera se consiguen resultados más precisos y robustos, lo que hace que el robot tome mejores decisiones.

Como conclusión general del proyecto se puede decir que se han cumplido satisfactoriamente los requisitos y objetivos marcados en el capítulo 2. El estado de las instancias del algoritmo está formado por la posición y la velocidad del objeto. Aunque la velocidad no es directamente medible, ésta se obtiene por inferencia a partir de las posiciones previas, obteniendo una medida bastante acertada. La colección de filtros es dinámica. Puede seguir varios objetos al mismo tiempo y, aunque sean indistinguibles para el detector, el algoritmo es capaz de emparejar las observaciones con su filtro correspondiente. El algoritmo se ejecuta en tiempo real y no consume muchos recursos.

La datos de entrada del algoritmo no están ligados a ningún sensor o detector concreto. El único imprescindible para ejecutarlo es la posición de un objeto en un

instante determinado del tiempo. Igualmente tampoco influye en el funcionamiento del algoritmo si los objetos son dinámicos o estáticos. Si el comportamiento de estos está bien modelado, tendremos unos resultados precisos.

La implementación del algoritmo es independiente de la plataforma, es decir, en el código del núcleo de ésteno hay ninguna referencia a componentes dependientes del sistema donde se ha desplegado. Esto facilita enormemente su exportación y podría, incluso, compilarse como una biblioteca independiente y ser usado en cualquier proyecto. Se ha intentado diseñar el algoritmo de forma que sea flexible y fácilmente extensible, ya que cada situación se resuelve de una manera distinta. Por último, la precisión y robustez de los algoritmos de las observaciones se ha mejorado. Los datos son más fiables y estables que si se operase con ellos directamente desde el detector.

El desarrollo del proyecto se ha llevado a cabo mediante pequeñas iteraciones en las que se han ido implementando cada una de las partes del algoritmo. Se ha puesto especial énfasis en la programación de las operaciones matemáticas. Muchas de las ecuaciones son bastante largas y complejas, hecho que provoca muchos pequeños fallos a la hora de escribir el código. Un simple cambio de signo de un término de la ecuación suele traducirse en comportamientos inesperados y extravagantes.

5.2. Trabajos futuros

El trabajo desarrollado en este proyecto ha dejado la puerta abierta a multitud de mejoras y nuevos trabajos que se pueden hacer partiendo de éste. Esto es sólo la punta del iceberg. En cuanto al algoritmo se puede mejorar lo siguiente:

- Incorporar aceleración al estado de los objetos. El modelo actual sólo contempla la posición y velocidad de los objetos. La aceleración del objeto ayudaría a realizar una mejor predicción del nuevo estado. Esto puede ser de especial interés en entornos más complejos que el del fútbol robótico. Por ejemplo, un coche autónomo podría medir la aceleración de los coches de su alrededor para predecir con mayor precisión futuros estados y tomar mejores decisiones.
- Modelo de movimiento de los objetos más preciso. El modelo actual es completo pero simple. El rozamiento del objeto con el suelo está modelado mediante un simple parámetro que disminuye la velocidad del objeto en cada iteración. A partir de la mejora anterior, una opción es modelar el rozamiento como una aceleración negativa a la velocidad.

- Comparación con UKF. El UKF es otra variante del algoritmo de Kalman para sistemas no-lineales. Utiliza otra técnica distinta de linealización que el EKF. En los últimos años está cobrando bastante popularidad. La linealización es menos engorrosa ya que no hay que calcular las derivadas simples ni Jacobianas de las ecuaciones de predicción y corrección. Sería interesante comparar el rendimiento de ambos y ver cuál tiene mejores resultados.
- Mejorar operación de emparejamiento. El emparejamiento es una operación de complejidad $O(n^2)$. El tiempo de resolución del problema crece exponencialmente con el tamaño de los filtros y observaciones que se tengan. Para valores pequeños de n no se nota mucho, pero según crece el número de instancias a solucionar, la resolución crece enormemente.

En el campo de las aplicaciones directas que tiene el algoritmo se proponen los siguientes proyectos.

- Cálculo de trayectorias en un portero. Teniendo la velocidad de un objeto es bastante sencillo calcular la trayectoria de este. Un comportamiento muy interesante para un portero sería la de calcular esta trayectoria para lanzarse hacia alguno de los lados para parar la pelota. Basta con averiguar si la pelota va a pasar por el lado izquierdo o derecho del robot.
- Cálculo de trayectorias en un jugador. El cálculo de la trayectoria de la pelota también es una función interesante para un jugador. El robot puede calcular la futura posición de la pelota e ir directamente a esa posición para llegar más rápido que los contrarios. En la figura 5.1 se presenta un pequeño diagrama que representa esta idea.

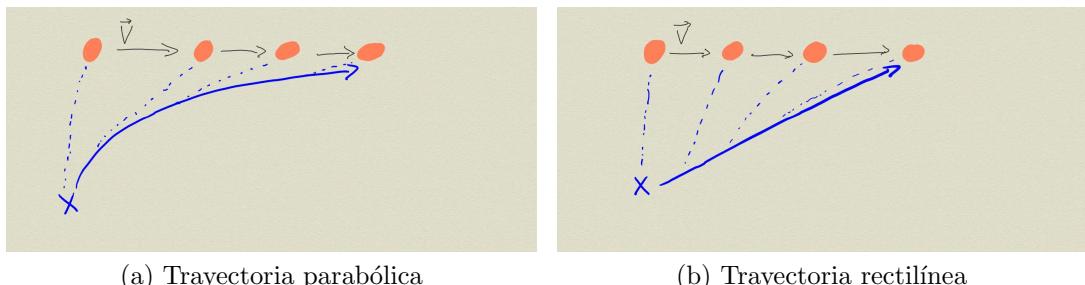


Figura 5.1: Diferencia de trayectoria con/sin tener en cuenta la velocidad de la bola

En la figura 5.1a se puede ver la típica trayectoria que toma un jugador que sólo conoce las posiciones instantáneas de la pelota. Es una trayectoria con forma

de parábola. Según se acerca el jugador a la pelota, ésta se va moviendo y el robot siempre toma las decisiones con retraso. En cambio, si se conoce la posible posición final de la pelota -figura 5.1b- , se puede ir directamente a esta localización y utilizar el sistema de visión, simplemente para comprobar que la hipótesis es correcta y, al mismo tiempo, realizar pequeños ajustes.

- Seguimiento de otros jugadores. El algoritmo se puede utilizar también para realizar el seguimiento de los otros robots que están en el campo. Esto puede ayudar enormemente en el cálculo de caminos para evitar a los jugadores con antelación.
- Estado compartido de la pelota. Se pueden introducir observaciones de otros jugadores para intentar mejorar la percepción de la pelota, llegando incluso al punto de conocer la posición de la pelota sin ni siquiera verla.