Universidad Nacional Autónoma de México

Posgrado en Ciencia e Ingeniería de la Computacion

# "Implementación de Modelos Ocultos de Markov en la navegación de un robot móvil autónomo"

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
MAESTRA EN INGENIERÍA
(COMPUTACIÓN)

PRESENTA:
MARÍA DEL ROCÍO ORTIZ ALBARRÁN

DIRECTOR DE TESIS:
DR. JESÚS SAVAGE

2011

∴

*«La vida debe residir en el hombre con toda su riqueza instruyéndose en la verdadera sabiduría, corrigiendo a su hermano, cantando con gratitud y de todo corazón.»*

פחות

*La energía capaz de crear, mover, fascinar, iluminar, cuidar es la misma que llevo dentro y que está en todos lados*

*Conocí el azul más brillante en el reflejo divino de una estrella que centelleó ante mi desde aquel primer suspiro:* ***Estelita****, que el cielo entero brille como lo haces tu.*

*Bebí constantemente de donde el agua brota con la fuerza de la calma, del río que sólo lleva una sabiduría dorada que ha compartido conmigo dia con día:* ***Luis Albarrán*** *contágiale al sol tu forma de dar, de combatir.*

*Fui de la mano de un ángel deslumbrante que me cobija así sin palabras,* ***Maria Estela****, nunca me abandones, ni a Luis.*

*Sentí el viento aquel que siempre me custodia, rápido, cálido que me abre los ojos a los tesoros de esta mañana* ***José****, enséñale a la noche lo que logran juntos la mente y el corazón.*

*Y cada mañana miro al cielo lleno de combinaciones increíbles y noto que su bondad es inmensa:* ***Luis,*** *sin el cielo nada se puede percibir, eres la base, eres impresionante, no te dejes caer, no dejes que el mundo vano te tire. Contigo hasta el fin del mundo.*

*Y encontré una mirada que era mi reflejo, que ve mi alma y contrarresta mis sentidos y solo en ondas sonoras llega a él mi voz... «****Lalo****, quiero compartir mi vida contigo»*

*Vivo cuidando a la luna, pequeña, a veces roja, consistente, de alma blanca, cada noche, esperando que sus pensamientos nunca dejen de brillar, y embelesen a todos,* ***Ana Laura*** *nunca dejes de compartir tus palabras.*

*Caminé al lado de contrucciones imponentes, fuertes, llenas de recuerdos e ideas del futuro que me inspiran, me acompañan, como tus palabras,* ***Mary Carmen****, hoy y siempre por ahi caminaremos en senderos cercanos.*

***Joaquis, Paco, Alonso, Emi*** *que su escencia mantenga quieto el mar, por que con la felicidad que me han dado pueden hacer eso y más,*

*Y cuando no salía el sol siempre hubo una mirada luminosa que me ayudó a ver que el camino estrecho lo paso fácil por que camino de puntas :D gracias Lalo,* ***Adrián****,* ***Rodri****,* ***Ger****,* ***Noe****, Jimmy,* ***Juárez****,* ***Memo****,* ***Tanya****,* ***Pepe****...*

***Como no te voy a querer UNAM!***

***Gracias a las instituciones!***
*Consejo Nacional de Ciencia y Tecnología (Conacyt)*
*División de Ingenierías Civil y Geomática (DICyG)*
*Posgrado en Ciencia e Ingeniería de la Computación (PCIC)*
*de las cuales recibí siempre una respuesta positiva ante la búsqueda de opciones antes inalcanzables.*

*Un **agradecimiento muy especial** a **Amalia**, **Diana** y **Lulú** quienes siempre nos han apoyado en todo, siendo parte muy importante de la comunidad del PCIC.*

*Al proyecto **DGAPA-PAPIIT IN-107609** parte de los apoyos de la UNAM a sus alumnos y docentes y con el cual se construyó el robot utilizado en este trabajo.*

*Un **sincero agradecimiento** a mis sinodales*
*Dr. Boris Escalante*
*Dr. Fernándo Aármbula*
*Dr. Ivan Meza*
*Dr. Miguel Moctezuma*
*y en especial al Dr. Jesús Savage, por su tiempo, interés y sus valiosos comentarios que enriquecieron el trabajo final.*

# Índice general

# Resumen

Ante la tendencia de la convivencia de agentes móviles con el ser humano, y la búsqueda de que estos agentes realizen las tareas cotidianas que el hombre hace día con día, se tiene el problema de la navegación autónoma, una actividad compleja en la cual participan los sentidos transformando la información percibida en información útil para cumplir con éxito la tarea de transladarse de un punto a otro evitando las colisiones.

Para lograr que un robot pueda transladarse a pesar de los errores de *hardware*, los errores en el manejo de números decimales y circunstancias de errores aleatorios ocasionados por el ambiente o el suelo por el cual se mueve ocasionando que el robot no cumpla con las tareas que implican la traslación de un punto a otro en un escenario, se buscó un método de retroalimentación que permite a través de sensores darle al robot una posición estimada que hace más fácil su arribo al destino deseado: el uso de modelos ocultos de Markov (*Hidden Markov Models*, HMM) que permiten relacionar dos variables como las que se tienen en este problema: el sensado y la posición del móvil, una de las variables es conocida (valor de los sensores) y la otra oculta (posición real del robot).

Para crear un modelo HMM $\lambda$ que ayude a la navegación se dividió el escenario en regiones las cuales representan los estados del modelo y se obtuvieron muestras de los sensores a lo largo de todo el ambiente por el cual se desea navegar. El modelo fue creado con el algoritmo Baum-Welch.

Para reducir la cantidad de información utilizada en la muestra que da origen al modelo $\lambda$ se utilizó un clasificador, el cual da como resultado un conjunto de vectores representados números enteros que facilita la cuantificación de las observaciones que realiza el agente en una posición determinada. Se compararon dos clasificadores, la red neuronal fuzzy ART(*Adaptive Resonance Theory*) y la cuantización vectorial (VQ, *Vector quantization* ), siendo muy general la red ART y dando mejores resultados en ambientes complejos la VQ.

A lo largo del desarrollo de la tesis se hizo necesaria una caracterización del robot (estudio de sus movimientos de traslación y rotación) para facilitar el análisis de los resultados y el origen de las fallas.

Finalmente con el uso del modelo $\lambda$ obtenido se aumentó el número de ocasiones que el robot llega a su destino en un 14.5 porciento y se obtuvieron altos porcentajes de reconocimiento de región por parte del móvil.

# Capítulo 1

# Antecedentes

*«Entonces, no puede recordar los tiempos en que no había robots. La humanidad tenía que enfrentarse con el universo sola, sin amigos. Ahora tiene seres que la ayudan; seres más fuertes que ella, más útiles y de una devoción absoluta.»*

***Isaac Asimov***

La participación de la robótica en las áreas industriales, docentes y cotidianas es un fenómeno que solía describirse únicamente en escenas de fantasía futurista surgidas de la imaginación de grandes escritores que jugaban mezclando la realidad de las construcciones industriales con una máscara de procesos no viables que la ciencia no podía concebir, entre otras cosas, por sus limitaciones computacionales. En la actualidad, el hombre trabaja con mayor frecuencia en procesos automatizados que perfeccionan o facilitan sus tareas haciendo cada vez más cercano el mundo en el que la interacción con la tecnología se da en la misma medida que la interacción ya existente entre seres humanos.

## 1.1. Introducción

La computación aún no tiene el poder de resolver la mayoría de los problemas que se desearía solucionar, sin embargo se han creado ingeniosos algoritmos que brindan soluciones útiles basados en dos cosas: en las teorías de las ciencias puras (como la física y las matemáticas) y en las más curiosas observaciones. La combinación de éstas permiten establecer soluciones aproximadas a los problemas que se presentan cuando se desea automatizar algún proceso que requiere un alto nivel de computabilidad.

El ser humano realiza de manera natural los procesos cotidianos más complejos: clasificar y ordenar (acciones, objetos, ambientes, situaciones), comprender (lo que ve, lo que escucha, lo que lee), interactuar, razonar, etc. Los procesos anteriores son aprendidos por el hombre a lo largo de un lapso de tiempo y luego los ejecuta intuitivamente en los momentos necesarios. Para lograr esto la maquinaria humana es capaz de percibir e interpretar su entorno, mediante los sentidos recibe la información necesaria para reaccionar a su ambiente

de la manera correcta según su experiencia y según la información obtenida. Además todos los datos recibidos por los sentidos encajan en un proceso determinado, el cual dirige el tratamiento de la información para brindar una reacción primero interna al cuerpo humano y luego externa para la interacción con el medio ambiente. Por ejemplo cuando un ser vivo desea esquivar un obstáculo, generalmente debe observarlo, encontrar un camino alterno y seguir el nuevo camino que está sometido al análisis constante para evitar posibles colisiones. La descripción indica, a este nivel, que el proceso es corto y no muy complicado, sin embargo depende de muchos factores externos como el nivel de luz existente en el ambiente para determinar los obstáculos o la manera en la que el ser vivo puede trasladarse, por lo que con este simple planteamiento se puede concluir que la secuencia anterior no es un proceso sencillo, además la cantidad de información procesada para determinar si existe un obtáculo o no, es muy grande. Lo cierto es que el cerebro humano realiza este procesamiento en muy poco tiempo y con resultados muy eficaces.

Una de las finalidades de realizar horas de trabajo e investigación en prototipos de automatización de tareas es buscar que un agente independiente de los seres humanos realice las tareas encomendadas actualmente al hombre y se desea igualar o superar la velocidad y los buenos resultados que alcanza una persona habituada a la ejecución de estas actividades. Para lograr este objetivo uno de los caminos que se siguen en la investigación es la aplicación de las ciencias exactas a temas delimitados, pero cuando los problemas que se busca resolver estan ligados con procesos complejos se recurre además a la observación del medio, del comportamiento animal, de los procesos humanos; esta observación ha inspirado al hombre para crear métodos novedosos en busca de una solución óptima, por ejemplo la Inteligencia Artificial se ha inspirado en la observación de los mecanismos cerebrales para plantear algoritmos que resuelvan problemas de clasificación o interpolación a través de redes neuronales, estos problemas planteados de la manera tradicional requieren de una cantidad de recursos computacionales exponencialmente proporcional al número de variables de entrada, además, obtener una generalización es un objetivo complejo pero necesario en la mayoría de los casos cuando se desean buenos resultados en aplicaciones con un nivel alto de variabilidad en los valores de entrada, las redes neuronales no se valen de una arquitectura tan compleja como la anatomía humana sino del análisis estadístico de los datos para su simplificación y de una versión compacta de la organización y almacenamiento de la información. La Inteligencia Artificial es una parte importante de la robótica pues ha aportado formas eficientes de representar el conocimiento, de tratar con el concepto de aprendizaje, de planificar y trazar rutas, de inferencia, búsqueda y procesamiento de información.

## 1.2. Problemática

Además de favorecer el incremento de la productividad de alimentos, ropa, calzado y aparatos electrónicos, la robótica puede ayudar al hombre a realizar una enorme cantidad de tareas en áreas como medicina, docencia e investigación. Algunos procesos requieren de movimientos específicos y repetitivos que son programables y que se realizan con mucho éxito mientras que algunos

otros necesitan desenvolverse en contextos de alto dinamismo con frecuente retroalimentación. En particular una de las tareas más explotadas por la ficción es lograr que los robots suplan a los trabajadores domésticos, los cuales realizan múltiples tareas de aseo, pero para lograr que un robot móvil realice con éxito estas tareas y que cubra con eficiencia las necesidades del ser humano es indispensable el buen funcionamiento del proceso de navegación, éste requiere de información acerca del ambiente próximo, de la ubicación en un ambiente actual y del control del movimiento.

Estos tres factores conforman el problema principal de la traslación de un robot móvil de un punto a otro en un ambiente determinado y serán el objeto de análisis de este trabajo.

### 1.2.1. Situación actual de la navegación autónoma

La navegación intenta conducir un móvil mientras atraviesa un entorno para alcanzar un destino sin chocar con algún obstáculo; alrededor del mundo se han implementado algorimos que permiten la navegación de un robot móvil en ambientes pasivos y activos, estos algoritmos se basan tanto en las técnicas más antiguas de posicionamiento como en algunos de los conceptos más avanzados de la ciencia e ingeniería.

Los algoritmos de navegación se pueden clasificar de diferentes maneras: por las necesidades que cubren, por las herramientas que utilizan o por la complejidad del proceso. La clasificación más conveniente a este análisis se define por las herramientas utilizadas. De acuerdo a este criterio, se tienen dos principales grupos, los algoritmos que se basan en un mapa y los que no lo hacen; los primeros algoritmos son denominados de planificación global y los segundos de planificación local, algunos ejemplos de estos algoritmos se muestran en la figura 1.1[7].

En la planificación local se navega sin usar un mapa, sus principales preocupaciones son:

- Evitar la colisión.
- Seguir una dirección determinada
- Moderar entre varios comportamientos.

En la figura 1.1 se muestran los siguientes ejemplos de planificación local (paradigma reactivo):

- El método de velocidad y curvatura (*Curvature-velocity method* CVM), basado en representar los obstáculos como circulos o focos de una parábola la cual será el trayecto del robot manipulando la posición actual y la curvatura del trayecto necesaria para llegar al destino y evaluando cada trayecto en funcion de los obstáculos que se ubican en la parábola y en la fluidez de los trayectos.[24]
- El método de carriles y velocidad (*Lane-curvature method* LCM) combina el método CVM con el uso de carriles, estos resultan de la division del ambiente en diversos trayectos eligiendo el óptimo de acuerdo a la posición actual y al nivel de libertad con la cual el movil puede navegar sin chocar.[13]

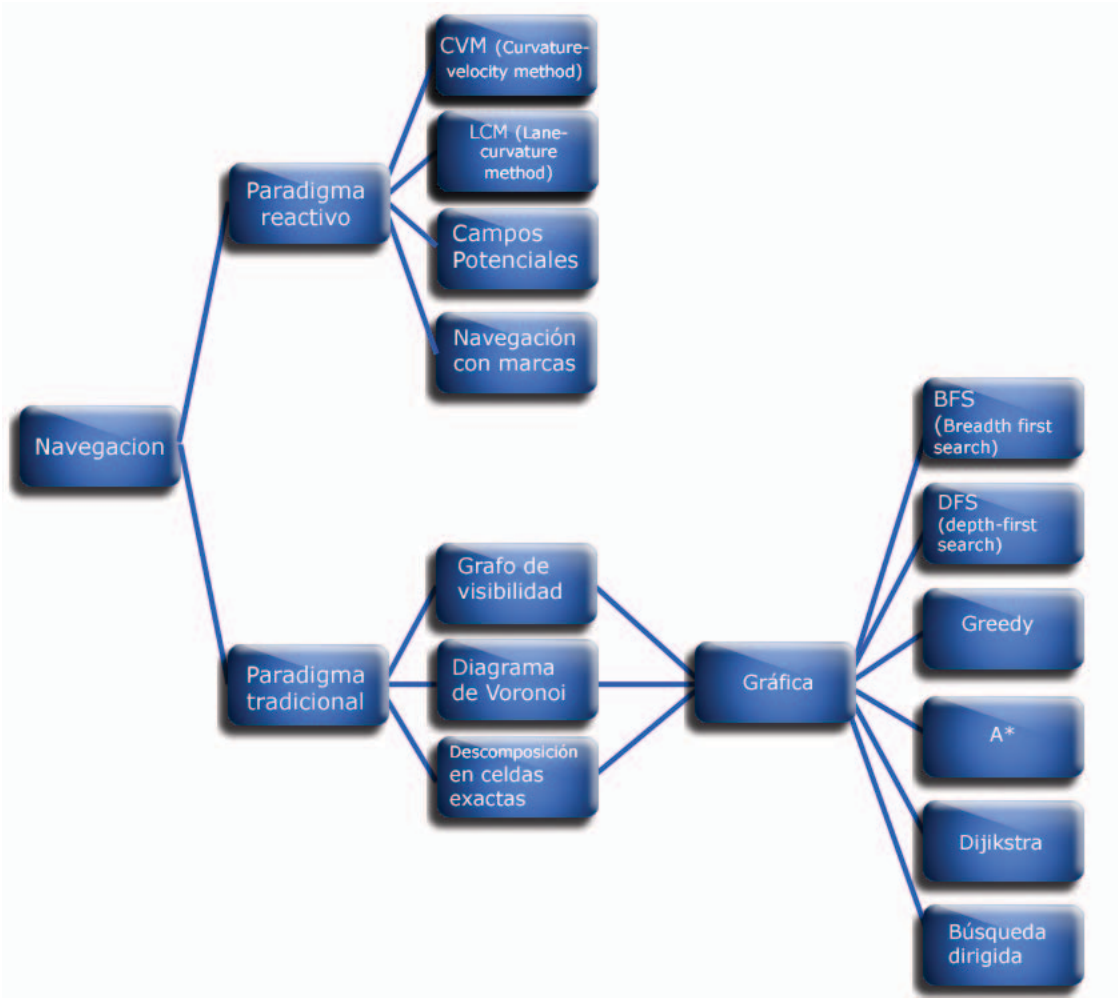



Figura 1.1: Clasificación de los algoritmos de navegación

- Los campos potenciales que representan los obstáculos como fuerzas repulsivas y a los puntos finales como fuerzas de atracción del móvil buscando una función que permite el movimiento del móvil por los trayectos libres.

- La navegación con marcas que a través de visión ubica al móvil en su ambiente gracias a marcas que se encuentran en éste, evitando colisiones y brindándole información acerca de los trayectos libres.

Como ejemplo de los algoritmos de planificación global se tienen diferentes métodos para obtener una gráfica del ambiente, una vez que se tiene la gráfica se utilizan diferentes algoritmos para buscar un camino entre los nodos se ponen como ejemplo los métodos : búsqueda a lo ancho (*breadth first search* - BFS), búsqueda en profundidad (*depth-first search* - DFS), algoritmo voraz (*greedy*), búsqueda A estrella (A*), algoritmo de Dijikstra y búsqueda dirigida.

En la planificación global existe un mapa del entorno lo suficientemente bueno para realizar la navegación, el mapa puede ser topológico, métrico o un híbrido. La navegación inicia en un punto conocido ubicado en el mapa, después se construye el camino en base a segmentos rectos y ángulos. La construcción de este camino requiere conocer estrategias de búsqueda en gráficass y planos o algunos métodos que permiten encontrar una ruta libre entre el punto de origen y el objetivo.

Lo ideal es que los paradigmas y arquitecturas tengan la capacidad de dotar a los comportamientos con herramientas para sobrevivir en un entorno abierto y de condiciones cambiantes. Es por ello que se considera de tanta importancia el poder proporcionar a los robots mecanismos de aprendizaje.

De entre los métodos de aprendizaje existe uno que empalma sus características con la robótica, el aprendizaje por refuerzo (AR) definido como el problema de conseguir que un agente actúe en un entorno de manera que maximice la recompensa que obtiene por sus acciones.

El móvil en estudio maneja un esquema híbrido:

- Tiene un algoritmo tradicional, el robot navega mediante la búsqueda del camino más corto en un grafo generado por los nodos de un mapa topológico que es realizado de manera manual.

- Tiene una implementación de navegación por campos potenciales.

Ambos comportamientos son regidos por un arbitro que “decide“ cual es el método que guiará al móvil en su búsqueda por alcanzar el punto destino.

### 1.2.2. Estado del arte de la auto-localización

La investigación en el campo de la navegación autónoma ha originado métodos ingeniosos y curiosos que son la base para las nuevas ideas, la mayoría de estos trabajos han sido desarrollados en el campo de localización a través de visión[20] por medio de algoritmos EKF (*extended Kalman filter*), el robot construye un mapa mientras se auto-localiza[29]. Este método funciona adecuadamente cuando se tienen marcas artificiales en el ambiente o cuando se cuenta con un grupo de robots. Otro enfoque está basado en la visión global del ambiente, se almacenan todas las escenas que genera el móvil a través del ambiente sin ningún tipo de extracción de características y el reconocimiento se logra cuando concuerda toda la imagen, este enfoque es usado para escenarios muy complejos cuando es muy difícil establecer los patrones a reconocer; con el fin de contrarrestar la cantidad de procesamiento se han realizado pruebas con imágenes de baja resolución[18], otro enfoque utiliza el histograma de color para llevar a cabo la auto-localización extrayendo otras características como la textura o la densidad de los bordes[30].

Tambien se ha utilizado la técnica de SURF (*Speeded Up Robust Feature*)[27], que en su versión estándar es más rápida que SIFT(*Scale-invariant feature transform*) que es también una lgoritmo muy utilizado en la búsqueda de características inmersas en las imágenes, además se ha reducido la complejidad del problema recurriendo al análisis de componentes principales, obteniendo un subespacio con el que es más fácil trabajar[14, 17].

### 1.2.3. Evaluación del proceso actual

El método actual es, en teoría, efectivo en tiempo real debido a que utiliza el mapa topológico para calcular la trayectoria y de vez en cuando ejecuta el comportamiento reactivo de los campos potenciales.

El comportamiento pasivo gasta poco tiempo de procesamiento por ser más rápido que los métodos que necesitan sensar su medio ambiente durante la trayectoria, sin embargo es un método con muy baja tolerancia a errores de *hardware* y de representación del medio ambiente

El comportamiento reactivo no es eficiente si es utilizado en trayectorias largas, está atenido al buen funcionamiento de los sensores y a la posición de los puntos objetivo y de los obstáculos.

Si tomamos en cuenta que la mayoría de los ambientes no son estáticos existen dos soluciones, actualizar constantemmente el mapa que guía al robot en su navegación o implementar un módulo que contenga las características del paradigma reactivo que permita obtener información de su entorno de una manera eficiente para que se pueda ejecutar en tiempo real y que brinde al robot una especie de memoria evitando procesamiento innecesario. Además debe ser un método que sea tolerante a los errores que ocasionan las lecturas odométricas que ofrece el *hardware*.

## 1.3. Planteamiento general de la solución

El entorno por el que puede navegar un robot puede ser descrito de manera muy abstracta, cuando el medio es estático los caminos que puede tomar un robot móvil autónomo no se modifican y la navegación no presenta problemas de trazado de ruta, sin embargo se presentan errores de ubicación ocasionados por los codificadores rotatorios y las fallas inherentes al *hardware* de control. Aunque el entorno sea estático existe la posibilidad de no llegar al destino indicado, es por esto que aún en los entornos sin movimiento es necesario adecuar al sistema de navegación de un robot móvil un módulo de retroalimentación que permite a través de sensores representar el entorno.

Los sensores brindan información de los objetos que se ubican en el ambiente del robot, son capaces de transformar magnitudes físicas o químicas en magnitudes eléctricas, ejemplos de un sensor son las imágenes obtenidas mediante cámaras, lecturas de sonares y lecturas de láser. Mediante un análisis de la información que estos proporcionan se pueden desarrollar algoritmos que faciliten la orientación y navegación de los agentes móviles de la misma manera como el ser humano identifica objetos del ambiente dándose una idea de su ubicación.

Sin embargo la cantidad de información que procesa el ser humano proveniente de los sentidos es muy grande, computacionalmente no se ha alcanzado esa capacidad de procesamiento por lo que se necesita comprimir los datos o buscar una manera de reducir la información que se alimenta al sistema a manera de simplificar el procesamiento en tiempo real. Es válido pensar en la compresión de información por que la redundancia está presente en la naturaleza de los datos. Los eventos independientes en los procesos mecánicos, físicos o biológicos son menos probables que los eventos dependientes lo que significa que existe una relación entre un suceso y otro; es común que sabiendo la ocurrencia de un evento se pueda inferir el otro con una probabilidad definida

en base a la observación de los fenómenos dependientes. Una menor redundancia en sistemas preceptuales hace mas económicos los problemas de codificar y describir la información contenida.

Los valores de sensado son extraídos en vectores de tamaño $n$, donde $n$ es el número de lecturas tomadas de los mecanismos de sensado considerados.

En el laboratorio de biorobótica se cuenta con un láser R283 de *Hokuyo* y una brújula HM55B de *Parallax*, este *hardware* puede ser tomado como equipo básico de sensado y obtener con esto datos que permitan representar el medio ambiente por el que se desea navegar. Se deben estudiar las características de precisión y resolución de los datos que brindan estos dispositivos para hacer el análisis de las entradas del modelo que ayudará al robot a ubicar en qué posición se encuentra.

Se debe analizar cuál es el número óptimo de lecturas de los sensores que permiten obtener la información necesaria y que no afectan el cómputo de los datos. Además es necesario establecer cuál es la estructura base de los sensores, se tiene una propuesta de numerar cada muestra de sensores, y que cada una de estas muestras tenga la posicion del robot definida como (x,y,$\theta$) y las lecturas de los sensores.

La cantidad de información que se recibe es grande y una manera de simplificar el procesamiento en tiempo real es encontrar vectores representativos del total de los vectores que se tienen con la finalidad de agilizar los procesos.

Para hacer un reconocimiento de las zonas que componen el ambiente se propone utilizar los modelos ocultos de Markov (HMM, por sus siglas en inglés *Hidden Markov Models*) debido a que ofrecen la posibilidad de usar dos variables aleatorias una conocida y una desconocida, la variable aleatoria conocida es el sensado y la desconocida la posición. Se tienen además todos los elementos necesarios para obtener el modelo formado por matrices de probabilidades en las cuales se contempla la probabilidad de ver un patrón en una posición determinada, la probabilidad de ir de una posición a otra y la probabilidad de iniciar la secuencia en una posición dada.

Los vectores representativos de la muestra funcionarán como observaciones que se utilizan en los modelos ocultos de Markov ya que, durante la navegación, el robot sensará su entorno y las lecturas obtenidas se clasificarán según los vectores característicos definidos obteniendo así una observación en el estado que se encuentre. Mediante esta observación se hará un análisis para determinar la posición del robot disminuyendo los errores de ubicación que perjudican la navegación.

Se tienen tres procesos importantes para cumplir con el objetivo de ubicar al robot con respecto a su entorno: la cuantización vectorial de las muestras, los modelos ocultos de Markov correspondientes a un ambiente y el algoritmo de Viterbi que permitirá, a través de la secuencia de estados arrojada, obtener la posición (x,y,$\theta$) más probable.

El proceso en general será el siguiente:

Se clasifican las muestras en vectores característicos (o *codebooks*) que son vectores surgidos del numeroso grupo de las muestras, estos vectores generalizan la información contenida en el total de las muestras, son vectores guía que permiten agrupar de una manera razonable a todas las demás muestras, como se ilustra en la figura 1.2.

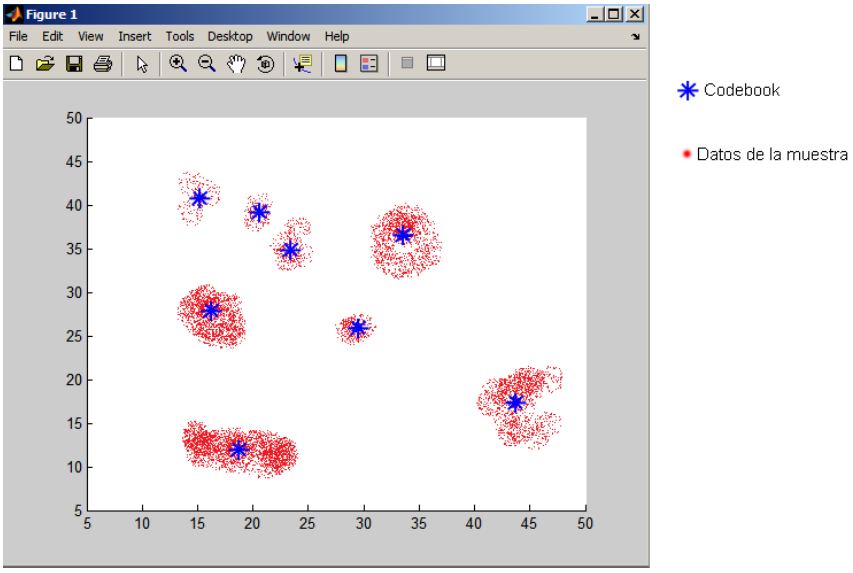



Figura 1.2: Ejemplo de centroides encontrados por cuantización vectorial

Una vez que se tienen los vectores característicos se encuentra el modelo de Markov de un ambiente, cuya representación gráfica se asemeja al modelo mostrado en la figura 1.3.

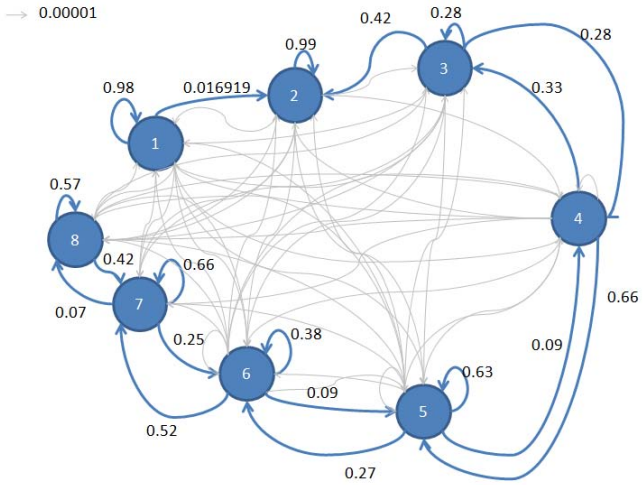



Figura 1.3: Ejemplo gráfico de las transiciones en un modelo de Markov oculto de 8 estados

El modelo se busca tomando como observaciones los vectores resultantes del proceso anterior de manera que cada una de las muestras deberá ser representada por alguno de los *codebooks* obtenidos en la cuantización vectorial, aquel que más se asemeja a las características del vector de muestra. En la figura 1.4 se muestran 5 codebooks y un vector de la muestra, este es asociado al vector

característico que más semejanza tiene con él.

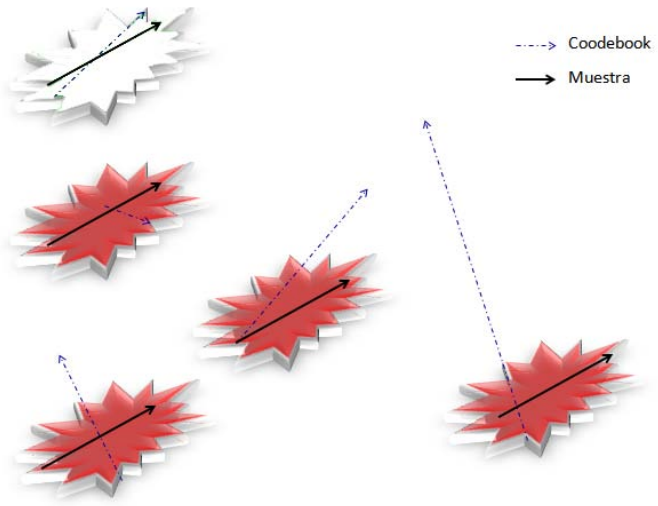



Figura 1.4: Concepto gráfico de la clasificación

## 1.4. Alcances de la solución

Basar los comportamientos de un robot en una teoría matemática como la teoría de decisiones logrará estabilidad en las acciones realizadas a diferentes niveles: simulación, pruebas y situaciones reales. Además de brindar un mayor nivel de concordancia entre las simulaciones y las pruebas.

Los resultados obtenidos al procesar los diferentes valores de las variables aleatorias involucradas serán más cercanos a los resultados previamente calculados teniendo así un mayor control de las decisiones siguientes y en general de toda la cadena de acciones que se ejecuten en una escena dada.

Se proporcionará un método alternativo de localización para no depender de los métodos pasivos como los mapas topológicos y la odometría, por lo que se espera mejores resultados en la ubicación y el alcance de los puntos objetivo.

Se espera mejorar la rapidez de navegación ya que la implementación del proceso de Markov será un algoritmo eficiente de localización y arrojará resultados que pueden aportar información útil para los agentes de control de movimientos.

Con todo lo anterior se busca:

- Mejorar el método actual de navegación utilizado en los robots móviles autónomos del laboratorio de biorobótica.
- Dar a los agentes de control un método de localización por regiones.
- Iniciar el proceso de desarrollo de más comportamientos reactivos para un robot móvil.

- Mostrar la utilidad de los modelos ocultos de Markov en la navegación de un robot móvil haciendo uso de los sensores como variables aleatorias ocultas.
- Analizar el método de navegación reactivo ya implementado de campos potenciales y compararlo con los resultados del proceso markoviano.
- Comprobar que los clasificadores encontrados por cuantización vectorial realmente obtienen los vectores característicos del grupo de datos y permiten encontrar la secuencia de estados correcta.
- Probar un funcionamiento eficiente de los procesos de Markov en el reconocimiento de patrones.
- Comprobar que el algoritmo de Viterbi al proporcionar la secuencia de estados más probable funciona como punto de referencia para ubicar un robot móvil en un ambiente de navegación.

## 1.5. Metodología

Fase teórica

1. Comprender a profundidad el funcionamiento y base de los modelos ocultos de Markov.
2. Estudiar las aplicaciones de robótica que utilizan este método para obtener comportamientos específicos.
3. Analizar los fundamentos matemáticos y el funcionamiento de los clasificadores.

Fase de implementación

1. Analizar las variables involucradas en una instancia determinada: lecturas de sensores, estado anterior, lecturas de odometría, orientación, lecturas de cámaras; a manera de determinar cuáles son relevantes y establecer para estas una definición correcta de variable aleatoria, lo que facilita su estudio con el fin de obtener el comportamiento deseado a través de un proceso controlado.
2. Realizar lecturas de los sensores para un ambiente, cada lectura estará representada por un vector $n$ dimensional. Las lecturas serán tomadas manipulando al robot a través de un programa exclusivo de toma de lecturas de los sensores, estos pueden ser: láser, sonar, brújula ó cámara y se almacenarán para su procesamiento.
3. Procesar las lecturas obtenidas a través de cuantización vectorial para obtener los vectores característicos del grupo de datos recopilados, estos vectores brindan las observaciones para el proceso markoviano.
4. Verificar la veracidad de las observaciones en pruebas con el robot.

5. Implementar el algoritmo de Viterbi para encontrar la secuencia de estados de navegación más probable por medio de las observaciones reportadas.

6. Verificar que el proceso de cadenas ocultas de Markov funciona para ubicar al robot en una región determinada por las observaciones y la probabilidad de estar en un estado y de las transiciones.

7. Hacer pruebas y ajustes al *software* y *hardware*.

8. Analizar los resultados obtenidos en la localización y navegación autónomas del robot.

## 1.6. Objetivo del Trabajo

Elaborar un programa basado en los modelos ocultos de Markov que de acuerdo a los valores de diversos sensores permita ubicar a un agente en un ambiente, implementar e integrar el *software* en un robot móvil real y comparar los resultados con otros métodos de navegación ya utilizados.

## 1.7. Organización de la Tesis

Capítulo 1. **Antecedentes**. Se describe el estado del arte del tema de análisis, se delimita el problema que se desea resolver y se plantea un objetivo.

Capítulo 2. **Marco teórico**. Se presentan las herramientas teóricas en las que se basa el estudio del problema y el planteamiento de la solución.

Capítulo 3. **Desarrollo de *sofware***. Se detalla la forma en la que se implementa la solución al problema.

Capítulo 4. **Implementación en el robot**. Se muestra la manera en la que se implementa el *software* desarrollado en el robot real.

Capítulo 5. **Resultados**. Se describen los resultados del *software* tanto en las pruebas en el simulador como en el robot real.

Capítulo 6. **Conclusiones**. Se evalúa la solución planteada con base en los resultados obtenidos.

# Capítulo 2

# Marco Teórico

*«Cuando necesite una información sobre la ciencia robótica te la pediré por escrito y por triplicado»*

***Isaac Asimov***

El posicionamiento y la noción de ubicación son conceptos intuitivos para el ser humano, se obtiene información a través de lo que se puede ver, oir, oler, tocar; pero cuando se piensa en transmitir la capacidad de ubicación a un agente de *hardware* y *software* la noción pierde su lado intuitivo y debe ser formalizado para lograr su implementación y obtener buenos resultados. Para esto es imprescindible dotar al agente de dispositivos que le permitan escanear el medio: cámaras, sonares ó láser y procesar la información que se recibe de estos para generar una posible ubicación. El proceso parece sencillo pero involucra gran cantidad de información y de análisis de ésta, es por eso que se requieren métodos específicos para lograr resultados semejantes a los que lograría el procesamiento natural de los datos.

## 2.1. Inteligencia Artificial

La Inteligencia Artificial(IA) brinda al ser humano una serie de herramientas que permiten atacar un problema determinado desde un punto de vista muy particular que engloba varias ramas de estudio de la ingeniería, la computación, la psicología, la física, la medicina, la filosofía y la teología. La teoría de la IA nace como una disciplina que intenta responder a preguntas relacionadas con las máquinas y sus alcances: ¿En un futuro, las máquinas serán capaces de pensar? ¿Cual es el proceso que necesitamos para lograr que una serie de metales piense?¿Cual es la probabilidad de lograr que algun día sea realmente difícil diferenciar a un hombre de un robot? pero actualmente ya se define como un área de las ciencias computacionales.

Existen muchos campos de aplicación para esta rama, por ejemplo la IA es la encargada de la creación de *hardware* y *software* fuera de lo común que tenga rasgos de comportamiento inteligentes, sin embargo, debido a la riqueza del área computacional y a su capacidad de interactuar con otras áreas es difícil encontrar un punto de acuerdo para la definición de *inteligencia*, esto causa muchos

conflictos por que se trabaja con conceptos que representan cosas distintas para un filósofo, para un médico, para un ingeniero o para un investigador, definir lo que es la inteligencia es el primer problema pero no obstaculiza proponer y probar avances en el área. Como la sola definición de comportamiento inteligente es compleja se delimitó a lo siguiente: un comportamiento que permita percibir, razonar y actuar con el fin de conseguir un objetivo planteado.

La IA también estudia cómo lograr que las máquinas realicen tareas que, por el momento, son mejor realizadas por los seres humanos.

La unión de la IA con la ingeniería es capaz de resolver problemas reales, actuando como un compilador de ideas acerca de cómo representar y utilizar el conocimiento, y de como ensamblar sistemas en favor de la optimización de procesos.

Los algoritmos que engloba la IA se pueden implementar computacionalmente y resolver problemas o aproximar alguna solución de estos con buenos resultados en tiempos eficientes. Es por esto que varios de los problemas complejos de la industria y la investigación pueder ser atacados mediante el enfoque de la IA, sin importar si son problemas con un algoritmo de resolución determinista o no determinista, ya que con el uso de técnicas evolutivas o probabilísticas se encuentran soluciones viables y que cumplen con las necesidades que hicieron surgir el planteamiento del problema.

## 2.2. ViRbot

Es una herramienta de *software* realizada por desarrolladores de la UNAM. Permite simular el comportamiento de un robot móvil de acuerdo a una trayectoria calculada en base a un punto origen y un punto destino, es capaz de simular la recepción de información de su ambiente a través de sensores, procesar esta información y modificar el comportamiento del móvil conforme a las entradas que recibe. Además permite la manipulación de un móvil real enviando las instrucciones necesarias para su movimiento de acuerdo a la planeación calculada[22].

ViRbot está compuesto por diversos módulos mostrados en la figura 2.1, el principal módulo de interés para este trabajo es el módulo de percepción y su correcta comunicación con el modelo del mundo y el sensado externo. El modulo de percepción contiene el subsistema de visión y junto con otros subsistemas obtiene una representación simbólica de los datos externos ya sea los que el usuario proporciona o los valores obtenidos de los sensores utilizados. La representación simbólica es generada aplicando algoritmos de procesamiento digital a los datos proporcionados por el módulo de sensado.

El *software* está realizado en lenguaje C++ pero tiene una implementación de comunicación por sockets que permite obtener y enviar los resultados a cualquier programa conectado al host y puerto correctos. Por esta razón los módulos en desarrollo pueden ser codificados en cualquier lenguaje de programación que logre la interconectividad por red con el sistema ViRbot.

El sistema utilizado no respeta todos los estándares de programación en C pero funciona sin problemas bajo unix, para su funcionamiento en *windows* son necesarias varias modificaciones principalmente en los archivos donde se utiliza memoria dinámica.

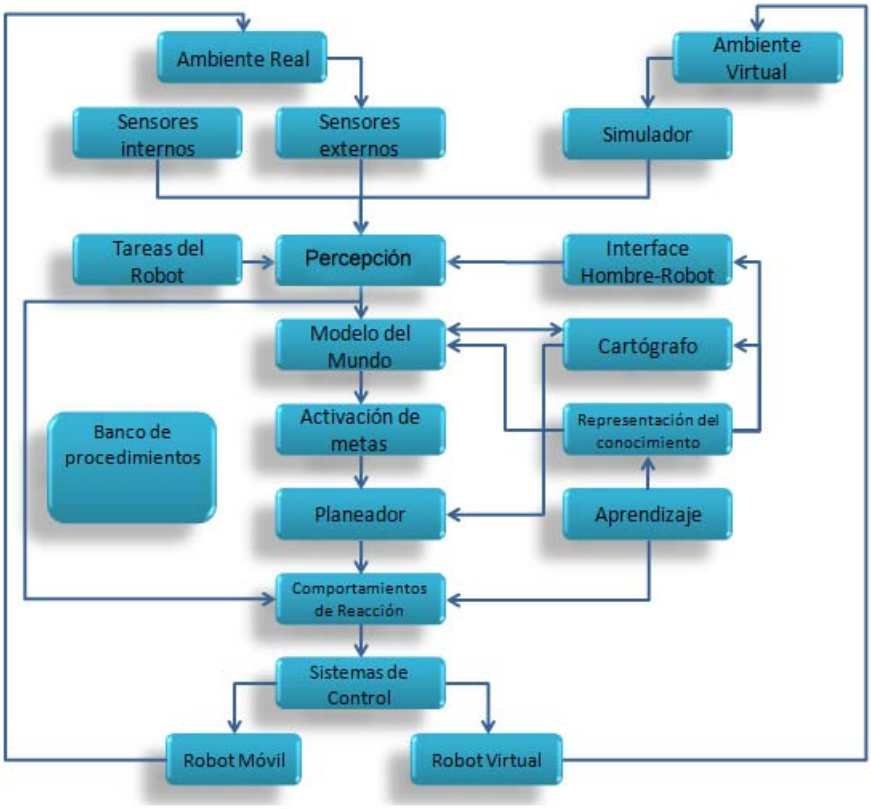



Figura 2.1: Módulos del sistema ViRbot

## 2.3. Fundamentos de procesamiento digital de señales

Si el proceso de sensado se aborda fuera de la simulación se debe considerar que la interacción entre el medio y el agente requiere un procesamiento de señal para poder trabajar adecuadamente con los valores sensados elegidos, el primer problema es la representación del conocimiento brindado por las lecturas de los sensores y la interpretación que debe efectuar el programa que recibe los datos, el segundo es el análisis de la veracidad de los datos y el tercero es buscar una forma adecuada de utilizar la información con el fin de cumplir los objetivos planteados. Para lograr buenos resultados se requiere:

- Delimitar el problema
- Analizar qué magnitud física trabajar (distancia, temperatura, intensidad, densidad).
- Establecer qué sensores se requieren

- Determinar cuál es el tratamiento que se dará al conjunto de lecturas recibidas de los sensores con el fin de obtener las características que se buscan al momento de implementar el sensado.

Una manera de lograr lo anterior es a través del procesamiento digital de señales, este puede ser definido en sentido amplio como la aplicación de técnicas digitales para mejorar la utilidad de una cadena de datos. Para lograr modificar los datos de la manera adecuada con el fin de resaltar las características de interés o para eliminar información no necesaria se requiere conocer la estructura de los datos, el comportamiento de estos bajo diferentes espacios, grado de entropía, redundancia y cantidad de información contenida. En los sistemas complejos la extracción de características dista de ser un problema trivial. Lleva normalmente a analizar el funcionamiento del sistema bajo distintos estímulos. El análisis de las señales a veces con elaboradas técnicas de procesado, permiten culminar con mejores resultados los objetivos generales. Algunas de las operaciones realizadas en el procesamiento involucran transformaciones de los vectores de entrada que representan la señal, a menudo es necesario usar distintos tipos de transformadas para que características muy importantes de las señales se hagan evidentes.

Después del procesamiento, se tiene ya un grupo de datos comprensibles a la computadora y que contienen la información necesaria para lograr el objetivo de ubicar al robot en el medio.

## 2.4. Modelos de Markov

Una vez que se tiene la información con las características adecuadas se puede pensar en un modelo que permita extraer los componentes principales de los valores de sensado y que con base a esto ubique al robot en una posición determinada dentro de su ambiente. Este modelo puede crearse con odometría pura, con el uso de marcas artificiales ó utilizando los modelos ocultos de Markov (HMM). La odometría requiere una impresionante coordinación entre el *hardware* y el *software*, además de ofrecer una baja tolerancia a fallos. El uso de marcas requiere de buena calidad de *hardware* de visión y que el ambiente tenga un determinado número de signos no naturales que el robot debe reconocer. Los HMM requieren precisión de los sensores y un buen preproceso de datos para poder encontra un patrón a seguir.

Para implementar la ubicación de un robot móvil se ha elegido utilizar métodos basados en procesos de Markov debido a que ya fueron probados a través de un simulador y se tienen sensores láser de buena calidad lo que permite realizar un preproceso de datos adecuado.

Los modelos de Markov estan basados en los procesos estocásticos y la propiedad de Markov.

Un proceso estocástico es una familia de variables aleatorias (v.a.) denotadas por $X$ y definido como $\{X_t|\ t\in T\}$, donde $X_t$ denota a la variable aleatoria *t-ésima* y $t$ es un número entero o real que define la naturaleza del proceso estocástico en la clasificación de continuo o discreto, es decir:

Si $T = \{0, 1, 2, \dots\}$ entonces el proceso se llama proceso estocástico con parámetro de tiempo discreto.

Si $T$ es un intervalo de la recta real entonces el proceso se denomina proceso estocástico con parámetro de tiempo continuo.

Si $T \subseteq R_n$ con $n > 1$ entonces el proceso se denomina campo aleatorio

Las variables aleatorias deben tener un espacio de probabilidad común, esto no sólo implica conocer el dominio de las variables aleatorias sino definir formalmente los conjuntos involucrados en los procesos estocásticos como se presenta a continuación:

$\Omega$: un conjunto no vacío.

$\digamma$: una colección de subconjuntos de $\Omega$.

$\sigma$-álgebra: una colección de subconjuntos de que cumple con lo siguiente:

1. $\Omega \in \digamma$.
2. Si A $\in \digamma$ entonces A' $\in \digamma$.
3. Si $A_1$, $A_2$,... $\in \digamma$ entonces:

$$\bigcup_{n=1}^{\infty} A_n \in \digamma$$

**Espacio Medible**: está formado por la pareja ($\Omega$, $\digamma$) donde $\Omega \neq 0$ y $\digamma$ es una $\Omega$- álgebra en $\Omega$.

**Medida**: es una función $\mu$ definida sobre $\digamma$ con valores en $\mathbb{R}$

**Medida de Probabilidad**: es toda medida $\mu$ sobre $\Omega$ tal que $\mu$ ($\Omega$ )=1.

**Espacio de Probabilidad**: Está definido por la tripla ($\Omega$, $\digamma$, P) donde el conjunto ($\Omega$, $\digamma$) es un espacio medible y P una medida de probabilidad sobre ($\Omega$, $\digamma$).

**Proceso de Markov**: es un proceso estocástico $(X_t)_t$ con la probabilidad de que dado el valor de $X_t$, los valores de $X_u$, para $u > t$, no dependen de los valores $X_s$ para $s < t$. De tal manera que si se tiene una correcta descripción del estado presente entonces el conocimiento del pasado no tiene ninguna influencia en la estructura probabilística del futuro.

De manera formal se tiene lo siguiente[12]:

El proceso estocástico real $\{X_t,\ t \in T\}$ es un proceso de Markov si para todo $0 \leq t_1 < \ldots < t_2 < t_n$ y B $\in \mathcal{L}$ se tiene que:

$$P(X_t \in B | X_{t_1}, X_{t_2}, \ldots, X_{t_n}) = P(X_t \in B | X_{t_n})$$

**Cadena de Markov**: es una clase especialmente sencilla de proceso de Markov que cumple con lo siguiente:

Se tiene que el conjunto ($X_0$, $X_1$, ... ) es una secuencia de variables aleatorias que toman valores en un conjunto numerable S, es decir, el rango de X es un conjunto discreto de estados.

La variable temporal es discreta y toma solo valores enteros : $t$=... , -2, -1, 0, 1, 2,...

El proceso es estacionario o al menos homogéneo, de forma que la probabilidad de transición sólo depende de la diferencia de tiempos.

**Probabilidades estacionarias**: Probabilidad de hallar el sistema en un estado determinado cuando lleva funcionando un tiempo indefinidamente largo. Dichas probabilidades se denotan como $\pi_{ij}$ y la matriz de probabilidades de estado estable como **P***.

**Cadenas Ergódicas**: cadena de Markov regular cuyas probabilidades estacionarias no dependen del estado inicial y ninguna de estas probabilidades vale cero[26].

### 2.4.1. Procesos de Markov controlados

Los procesos de Markov controlados generalizan las cadenas de Markov. Se tiene una sucesión de variables aleatorias $x_n$ que indican el estado de una partícula en el paso $n$. La variable $x_n$ toma valores en $X$ que es el conjunto de estados posibles[12].

La dinámica es la siguiente:

Si en el instante $i$ la partícula está en el estado $x$ se debe tomar una acción $a$, entre el conjunto de acciones posibles para ese estado($A(x)$). La acción tomada determina la distribución de probabilidad en $X$ para el siguiente estado, además de tener asociado un costo $c_i(x, a)$.

La diferencia sustancial con una cadena de Markov, es que cuando la partícula se encuentra en un estado dado, no está definida la distribución del siguiente estado hasta que no se toma una acción, de ahí la idea de control. Se obtiene una cadena de Markov en el caso particular en que sólo hay una acción posible por estado, de modo que no hay decisión alguna.

Un problema que se plantea es encontrar un procedimiento de control, es decir, un criterio para decidir las acciones a tomar, de modo que el valor esperado del costo total (suma de los costos de las acciones tomadas), a lo largo de un intervalo de tiempo, sea mínimo.

Existen tres procesos controlados de Markov principales:

- MDP( Markov decision processes): procesos de decisión de Markov
- SMDP(semi-Markov decision processes): procesos de decisión semi-Markov
- POMDP(partially observable Markov decision processes): proceso de decisión de Markov parcialmente observable

### 2.4.2. Procesos de Decisión de Markov

**Procesos de decisión de Markov**

Una tarea de aprendizaje por refuerzo que satisface la propiedad de Markov se denomina proceso de decisión de Markov, ó MDP. Siguiendo uno de estos procesos de decisión, un agente puede escoger la acción más adecuada de entre todas las posibles [12].

Un MDP se define como una tupla $< S, A, T, R >$, tal que:

$S$ es un conjunto de estados

$A$ un conjunto de acciones

$T : SxA \longrightarrow P(S)$, donde un miembro de $P(S)$ es una distribución de probabilidad sobre el conjunto $S$; es decir, transforma estados en probabilidades. Se dice que T(s; a; s') es la probabilidad de que se realice una transición desde s hasta s' ejecutando la acción $a$ .

R : S x A$\longrightarrow$R, que para cada par estado-acción proporciona su refuerzo (ó recompensa). Se dice que R(s, a) es el refuerzo recibido tras ejecutar la acción a desde el estado s.

**Procesos de decisión semi-Markov**

Se puede describir como una generalización de una cadena de Markov de tiempo continuo. Definimos la v.a continua $H_{ij}$ igual al tiempo de permanencia en el estado $i$ antes de saltar al estado $j$.

En un proceso semi-Markov dejamos que la distribución de $H_{ij}$ sea arbitraria. Si $H_{ij}$ está distribuida exponencialmente, tenemos una cadena de Markov de tiempo continuo.

Si $H_{ij}$ no está distribuido exponencialmente, considerar sólo el estado actual no cumple la propiedad de Markov (de ausencia de memoria) pues la evolución del proceso depende del estado actual y el tiempo de permanencia en este estado $(i, t_i)$.

Considerar $(i, t_i)$ si cumpliría la propiedad de Markov, pero tendríamos un proceso de Markov (pues $t_i$ no es una variable discreta)[12].

**Proceso de decisión de Markov parcialmente observable**

En las ocasiones en las que no se dispone de información acerca del estado actual, la tarea anterior se convierte en un proceso de decisión parcialmente observable. Este modelo sigue una aproximación teórica distinta y proporciona una manera adecuada de razonar acerca de los compromisos entre acciones para ganar recompensa y acciones para acceder a más información.

Existen problemas reales en donde no es posible determinar el estado del proceso fácilmente o conseguirlo resulta ser muy costoso.

En un modelo oculto de Markov el estado es parcialmente observable (no se conoce, pero existen observaciones o $\in$ O), y no existen acciones (no hay toma de decisiones la transición equivale a una sola acción) [12].

### 2.4.3. Modelos Ocultos de Markov (HMM)

Un modelo oculto de Markov o HMM (*Hidden Markov Model*) es un proceso estocástico doble en el cual uno de los procesos no es observable (es oculto) la secuencia es un modelo estadístico obtenido del análisis de un sistema que posee la propiedad de Markov en el cual existe una variable aleatoria conocida y una desconocida.

El objetivo es determinar el valor de las variables aleatorias desconocidas a partir de las variables aleatorias observables.

Una vez que el modelo oculto ha sido establecido a través de conteo estadístico se puede utilizar para encontrar los valores de las v.a. desconocidas para cada secuenci*hardware*a de valores conocidos, esto se puede aplicar en áreas como el reconocimiento de patrones. Un HMM se puede considerar como la red bayesiana dinámica más simple.

En un modelo de Markov normal, el estado es visible directamente para el observador, por lo que las probabilidades de transición entre estados son los únicos parámetros. En un modelo oculto de Markov, el estado no es visible directamente, sino que sólo lo son las variables influidas por el estado. Cada estado tiene una distribución de probabilidad sobre los posibles símbolos de salida.

Consecuentemente, la secuencia de símbolos generada por un HMM proporciona cierta información acerca de la secuencia de estados. Una notación habitual de un HMM es la representación como una tupla:

$HMM = \lambda = (Q, V, \pi, A, B)$

Donde:

- El conjunto de estados es $Q$=$\{q_1$,$q_2$,$\ldots q_N\}$. El estado actual se denota como $q_t$.

- El conjunto$V$ de posibles valores observables en cada estado.
- Las probabilidades iniciales $\pi = \{\pi_i\}$, donde $\pi_i$ es la probabilidad de que el primer estado sea el estado $q_i$.
- El conjunto de probabilidades **A** (|A|=$i$ x $j$) de transiciones entre estados. $a_{ij} = P(q_t = j|q_{t-1} = i)$ , es decir, $a_{ij}$ es la probabilidad de estar en el estado$j$ en el instante $t$ si en el instante anterior $t-1$ se estaba en el estado $i$.
- El conjunto de probabilidades $\mathbf{B} = \{b_j(v_k)\}$ de las observaciones. Donde$b_j(v_k)$ = P($o_t = v_k \mid q_t = j$), es decir, la probabilidad de observar $v_k$ cuando se está en el estado$j$ en el instante $t$. La secuencia de observaciones se denota como un conjunto O= $(o_1, o_2, \ldots o_T)$. Donde $T$ es la longitud de la secuencia de observaciones.

Además se tiene:

- $M$ número de símbolos de observación.

Para trabajar con un modelo $\lambda$ se tienen identificados tres problemas fundamentales cuyo estudio y planteamiento de solución permiten obtener y usar un modelo HMM en aplicaciones reales[21].

### 1. Problema de evaluación

Dada la secuencia de observaciones $O = o_1, o_2, ...o_T$ y el modelo $\lambda = (A, B, \pi)$, ¿Cómo se puede calcular $P(O|\lambda)$?, es decir la probabilidad de obtener la secuencia $O$.

**Solución:**

Existe un cálculo intiutivo que arroja una complejidad computacional elevada ($2TN^T$), lo cual lo hace inviable, pero se tiene el algoritmo *forward - backward*. Este utiliza dos variables intermedias ($\alpha$ y $\beta$ ) obteniendo una complejidad computacional de $N^2T$.

La variable $\alpha$ representa la probabilidad de obtener una secuencia de estados $O$ en el estado actual $S_i$ dado el modelo $\lambda$ que se tiene.

$$\alpha_t(i) = P(O_1O_2...O_t, q_t = S|\lambda)$$

La variable $\beta$ busca la probabilidad de observar una secuencia determinada a partir del estado actual ($S_i$) y en los siguientes estados hasta finalizar el número de observaciones.

$$\beta t(i) = P(O_{t+1}O_{t+2}...O_T|q_t = S_i, \lambda)$$

A grandes rasgos se busca iterar en cada elemento de la lista de observaciones $O$ calculando $\alpha$y $\beta$y operando estos hasta obtener, en la última observación el valor de $P(O|\lambda)$. El proceso se detalla en la sección 3, algoritmo 3.1.

**2. Problema de la aplicación a estados**

Dada la secuencia de observaciones $O = o_1, o_2, ...o_T$ ¿Cómo se elige la secuencia de estados más probable que dio origen a la secuencia $O$ ?

**Solución:**

Se utiliza una estrategia de programación dinámica que requiere definir las siguiente variable:

$\delta_t(i) = max_{q_1 q_2 ... q_{t-1}} P[q_1 q_2 ... q_t = i, o_1, o_2 ... o_T | \lambda]$

esta representa la máxima probabilidad de un camino en el tiempo $t$ estando en el estado $S_i$

Quien propuso esta estrategia fue Andrew Viterbi por lo cual el proceso de inducción que da pie al algoritmo de cálculo del camino de estados más probable lleva su apellido. El proceso se describe en la sección 2.7 de este capítulo.

**3. Problema del entrenamiento.**

Dada la secuencia de observaciones $O = o_1, o_2, ...o_T$, ¿Cómo se ajustan los parámetros del modelo $\lambda$ para maximizar $P(O|\lambda)$?

**Solución:**

Es el problema con menor grado de intuición por lo que se trabaja con algoritmos bien definidos que utilizan el principio de máxima verosimilitud, métodos de gradientes u optimización con multiplicadores de Lagrange.

La mayoría de los métodos requieren una estimación inicial obtenida del conteo simple de ocurrencia de las observaciones de una muestra en cada estado observable.

Posteriormente se aplica un algoritmo iterativo que utiliza el cálculo de $P(O|\lambda)$, $\alpha$ y $\beta$ para generar nuevas variables: $\xi$ y $\gamma$ las cuales se detallan en la sección 3.2.3 del capitulo 3.

## 2.5. Análisis de componentes principales (PCA)

El análisis de componentes principales (*Principal Components Analysis*, PCA) es una técnica estándar comunmente usada para reducir un set de datos (dimensión o cantidad de información) en reconocimiento de patrones y en procesamiento de señales[10].

El método surge intuitivamente del concepto *redundancia,* presente en los datos originados por la mayoría de las fuentes de información. La redundancia es un intento natural por preservar la información manejada en ambientes hostiles, llenos de ruido y su presencia crea dependencia entre las variables que se utilizan cuando se está estudiando un fenómeno. Un gran número de eventos de interés para la ciencia y la ingeniería dependen no solo de una sino de múltiples variables lo cual hace difícil su análisis. El método PCA busca reducir la dimensión del problema recurriendo al cálculo de la correlación de las variables estudiando la covarianza de los valores de una muestra. Por éste método se detectan de manera realativamente sencilla aquellos componentes que abarcan el mayor número de variables.

## 2.6. Cuantización vectorial

Sería computacionalmente muy costoso que se tomaran en cuenta todas las muestras de sensado como observaciones del modelo HMM, de manera que se requieren agrupar todos los datos en grupos representativos según sus características. Como no se conoce *a priori* ninguna de estas características es necesario utilizar un algoritmo de aprendizaje no supervisado que encuentre de manera autoorganizada la división más conveniente.

Entre los métodos más conocidos se encuentran los clasificadores neuronales ART (*Adaptive Resonance Theory*) y el método de cuantización vectorial, las redes ART se componen de diversos parámetros internos que permiten moldear la estructura final del grupo de clasificaciones, existen versiones para valores binarios y analógicos, se basan en la teoría de la resonancia adaptativa, formulada por Stephen Grossberg y Gail Carpenter[4]. Estas redes tienen un buen nivel de eficiencia, generalmente constan de 2 capas únicamente, la de entrada y la de salida, y su arquitectura no es rígida, es decir el proceso varía el número de neuronas utilizadas en cada capa al momento de la ejecución[3].

La cuantización vectorial (CV ó VQ, por *vector quantization*) es un proceso mediante el cual se puede obtener un subconjunto de vectores según las características representativas de un grupo de datos, al inicio del algoritmo se decide cuántos vectores representativos se desea obtener pero no se conoce cual será la característica que se encargará de distinguir un grupo final de otro por lo que se considera parte de los algoritmos de aprendizaje no supervisado. El algoritmo, a grandes rasgos, es el siguiente:

1. Se tiene un grupo numeroso de vectores que se desea clasificar y se define un número máximo de grupos a los que debe llegar el cuantizador vectorial.

2. Se calcula el centroide del grupo de vectores.

3. Se le suma y resta al centroide una cantidad pequeña que permite obtener 2 centroides ligeramente separados.

4. Se agrupan los vectores según el centroide que les quede más cerca.

5. Se recalculan los centroides, se repite el paso anterior y el actual hasta que ya no cambien de grupo los vectores.

6. Una vez que se tienen los grupos bien definidos verificar si se ha alcanzado el número de vectores característicos deseados para finalizar el algoritmo, si no es así, regresar a 3.

## 2.7. Viterbi

El algoritmo de Viterbi permite encontrar las secuencia de estados más probable en un modelo oculto de Markov, $S = (q_1, q_2, \ldots, q_T)$, a partir de una observación $O = (o_1, o_2, \ldots, o_T)$ es decir, obtiene la secuencia de estados óptima que explica la lista de observaciones.

En general el algoritmo de Viterbi consta de los siguientes procesos:

1. Inicialización de los deltas que se obtienen como el producto de la probabilidad inicial con la probabilidad de ver la observación 1 ($o_1$) en el estado 1 $\delta_{1i} = \pi_i b_i(o_1)$

2. Recursión. Para obtener los siguientes valores de $\delta$ se busca obtener el máximo valor de los obtenidos al multiplicar el $\delta$ anterior por la probabilidad de transición de un estado $i$ a otro $j$ y el valor obtenido se multiplica por el valor de ver la observación siguiente en el estado $j$.

$$\delta_{t+1}(i) = [\max_{1 \leq i \leq N} \delta_i(a_{ij})] b_j(o_{(t+1)}) \tag{2.1}$$

donde $N$ es el número de estados definidos en el análisis del proceso y $t$ es en número de vectores de observaciones que se tiene y por lo tanto es igual al número de estados totales que se obtendrá en la secuencia y va de 1 a $T-1$; $j$ es el estado siguiente y va de 1 a $N$.

En $\psi$ se almacena el estado que corresponde a $\delta$ para cada $t$ mediante la función $argmax()$ . Finalmente al terminar el cálculo para todas las observaciones, se obtiene de $\psi$ la secuencia de estados más probable.

# Capítulo 3

# Desarrollo del software

*«Esto marcaría la fecha más importante en el avance de la ciencia robótica de nuestra era si supiésemos por qué sucede... No lo sabemos, y tenemos que averiguarlo... Antes de que hubiese transcurrido un mes, nada de lo que podía hacerse había dejado de ser hecho.»*

***Isaac Asimov***

El objetivo de la tesis se cumple mediante el diseño de *pathHmm*, un *software* que integra a la navegación un módulo de localización basado en modelos ocultos de Markov siendo la variable observable el sensado proporcionado por el móvil y la variable oculta la posición real del robot en este apartado está descrito el desarrollo e implementación del mismo para conocer cómo funciona y para facilitar su escalabilidad y modificación.

## 3.1. Análisis

El uso adecuado de los sensores facilita la navegación de los agentes autónomos, se pueden utilizar desde sencillas técnicas de comparación hasta procesos estadísticos que mejoren la ubicación y el traslado del robot en un ambiente con el fin de lograr una tarea determinada. El procesamiento que tendrán las lecturas que brinda el proceso de sensado en este trabajo ayudará al módulo de *Percepción* del sistema ViRbot a tener información más útil en la creación del modelo del mundo en el que se encuentra, como se ilustra en la figura 3.1.

Se tomará una muestra de los sensores del robot en una posición determinada y el algoritmo deberá proporcionar la ubicación más cercana a la real solo con la información de los sensores, buscando dotar al sistema de una retroalimentación al momento de navegar y combatir de esta manera los errores de *hardware* y medición inherentes al movimiento mecánico y eléctrico.

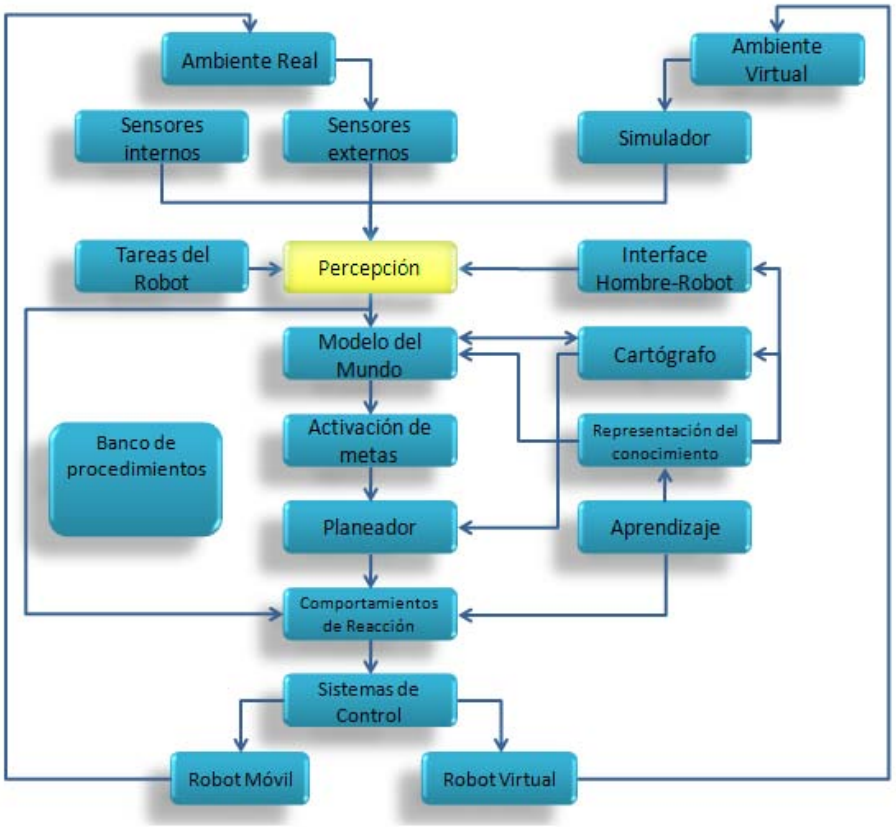



Figura 3.1: Ubicación del módulo desarrollado en el sistema ViRbot

El proceso global de navegación se verá altamente beneficiado con el *software* propuesto debido a que ayuda a cumplir con la ruta generada por el planeador, es necesario trabajar en esta área debido a que las técnicas tradicionales no proporcionan retroalimentación mientras se recorre el trayecto y durante este se presentan errores tanto sistemáticos como aleatorios que perjudican el movimiento del móvil ocasionando que no alcance el destino planeado por mala ubicación o incluso por que el error acumulado en los subtrayectos causa colisiones con los objetos que se encuentran en el ambiente.

El objetivo del *software* en desarrollo es proporcionar coordenadas $(x,y,\theta)$ de ubicación del robot autónomo con el fin de ayudar al móvil en la navegación cada vez que el sistema de control lo solicite. Para lograr el objetivo se utilizó el sistema ViRbot que permite simular los comportamientos y ejecutarlos en el robot estableciendo una comunicación modular entre cada programa ya integrado y el *software* que se desarrolla con el fin de cubrir nuevas necesidades. De esta forma los principales requerimientos del programa *pathHmm* dependen en gran medida de las necesidades del sistema ViRbot, los requerimientos propios de la implementación en el robot serán descritos más adelante.

Se necesita captar en tiempo real las lecturas que arroje el *hardware* de sensado y relacionarlas con la posición en la que se ubica. Si no se obtienen los

resultados deseados se requiere de un preproceso de datos que resalte algunas características de los vectores que funcionan como variables observables a fin de facilitar la tarea del HMM y lograr un mejor reconocimiento de la variable oculta.

### 3.1.1. Requerimientos de hardware

Se requiere una representación del medio por el cual el robot va a navegar, esta representación será formada por las lecturas de los sensores que se tienen.

En particular se utiliza un láser R283 de Hokuyo y una brújula HM55B de Parallax.[1]

El campo de vista del láser es de 240° y la resolución angular es de 0.36° lo que brinda más de 600 lecturas posibles al momento del sensado se pueden obtener datos del ambiente cuando los objetos contenidos en él se encuentran entre 2 y 50 [dm].

La brújula permite una lectura de la posición angular del dispositivo con respecto a una calibración inicial, la lectura tiene una exactitud aceptable de acuerdo a los valores reales, la precisión no es detallada y la brujula se ve afectada por campos externos generados por el mecanismo del robot móvil. Si se utilizan menos lecturas del láser se puede facilitar el cómputo y el análisis de los datos pero se pierde información, saber cuál es el número óptimo de lecturas no es un proceso sencillo por esta razón se determinará este valor de manera experimental a través del análisis de resultados y con la ayuda del PCA.

### 3.1.2. Requerimientos del sistema

De manera global el sistema debe ser capaz de obtener vectores de sensado de diferentes tamaños, obtener la clasificación más adecuada para estos vectores para brindar una secuencia de observaciones que se utilizará en el entrenamiento del modelo $\lambda$, una vez obtenido el modelo debe quedar almacenado para ser utilizado en tiempo real por el robot y debe brindar el estado actual más probable.

En la primera fase el simulador se encarga de almacenar en un archivo de texto cada una de las lecturas que obtiene el robot a lo largo de su trayecto. En base a este archivo se realiza el entrenamiento para obtener en primer lugar los símbolos de observación y posteriormente el HMM más probable para las diferentes secuencias de observación. El archivo mencionado debe ser procesado para presentar la información con el formato de entrada que requiere el cuantizador vectorial.

### 3.1.3. Extracción de características

Cuando el ser humano intenta reconocer un ambiente hace uso de sus sentidos identificando características peculiares de un lugar. Esto le permite ubicarse en un entorno y saber dónde se encuentra. La cantidad de información utilizada en el proceso de reconocimiento que realiza el ser humano es muy grande

[1] La brújula se descartó en el transcurso de la implementación del proyecto debido a que las lecturas arrojadas presentaban una precisión de menos del 50 % en los ángulos requeridos debido a los campos magnéticos presentes en los ambientes utilizados.

y computacionalmente compleja, por lo que el uso de mucha información en un algoritmo solo garantiza que el proceso sea lento y que ocupe una gran cantidad de recursos no asegurando un buen reconocimiento de aquello que buscamos ubicar o clasificar. Es por esto que es necesario analizar cuáles son las características relevantes que se pueden obtener con el *hardware* disponible, qué cantidad de información es necesaria para un buen procesamiento y cómo presentar la información para optimizar el proceso algorítmico y sobre todo los resultados.

Para lograr la reducción del conjunto de datos que se manejará en el entrenamiento y en el reconocimiento en tiempo real se debe realizar un estudio estadístico para determinar los componentes principales del vector de sensado. Estos vectores se pueden utilizar principalmente en la fase de prueba donde es necesario realizar ajustes y comprender cada paso en la manipulación de datos y sirven para evitar procesamiento innecesario que no mejora de manera significativa los resultados. El análisis estadístico incluye el cálculo de las covarianzas entre los elementos de los vectores de muestra disminuyendo la dimensión de los elementos trabajados.

Posteriormente para evitar una carga innecesaria a las variables observables en el modelo oculto de Markov se deberá realizar el proceso de cuantización vectorial reduciendo el número de observaciones disponibles en cada estado.

### 3.1.4. Requerimientos de Interfaces

El único requisito es que el programa desarrollado se pueda integrar al sistema ViRbot, no es necesario que se programe en el mismo lenguaje ni en el mismo sistema operativo pero sí se debe asegurar que la comunicación entre módulos se realice sin problemas.

El programa debe ser ligero ya que al ser ejecutado en tiempo real no se deben perder recursos en gráficos innecesarios ni en algoritmos mal diseñados, se debe poner atención en la complejidad de los algoritmos y en que su ejecución sea automática, si los programas necesitan parámetros se deben especificar de manera muy clara para que cualquier persona involucrada en el manejo del robot pueda utilizar el *software* en desarrollo.

## 3.2. Diseño

### 3.2.1. Representación del ambiente

Las lecturas representan una nube de puntos que delimita la zona por la cual el móvil puede trazar un trayecto.

Se requiere de un módulo de toma de lecturas mientras se cumple con un trayecto, estas muestras servirán de entrenamiento, de prueba y en la fase de ejecución en tiempo real.

En la primera fase las muestras se obtienen con el simulador ViRbot, se guardan las lecturas de los sensores generadas a lo largo de varias trayectorias, de esta forma se obtiene una muestra significativa que sirve para representar el ambiente en el que navegará el robot.

Una lectura es un vector unidimensional de $n$ elementos como se muestra en la figura 3.2, el primero es el elemento $x$ de la posición, el segundo es el

elemento $y$, el tercero es el ángulo y los siguientes valores corresponden a los sensores por lo cual se tiene $n-3$ valores de sensado, el valor de $n$ depende del dispositivo utilizado para obtener las lecturas (sonar o láser). Cada lectura es almacenada en un archivo mientras se ejecuta el simulador, de esta manera se obtienen las muestras necesarias para la implementación de los HMM.

En la segunda fase se modificaron las lecturas obtenidas en la primera fase añadiendo ruido a los valores obtenidos, el ruido es generado con distribución normal. Una vez que las lecturas son modificadas están listas para el proceso de implementación de los HMM.

Para la tercera fase se obtienen lecturas reales, en lugar del simulador se utilizan los sensores del robot a través de *software* que permite adquirir los datos necesarios. En el caso de las lecturas de láser se utiliza un dispositivo de entrada que ayuda a guardar los datos necesarios: un joystick, a través de la manipulación de este accesorio y un programa se guardan las lecturas en el formato mostrado en la figura 3.3.

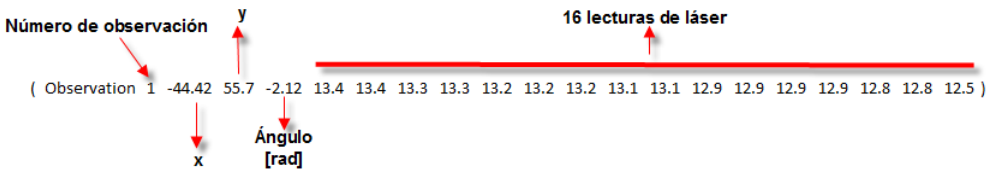



Figura 3.2: Formato de las muestras obtenidas de los sensores del robot

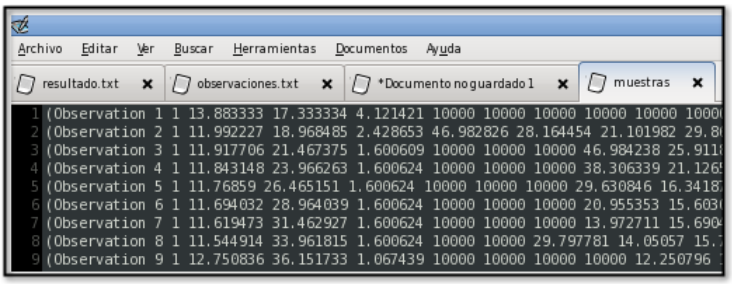



Figura 3.3: Archivo que ejemplifica el almacenamiento de las muestras

Además los valores de las lecturas dependen del lugar y la distribución de los objetos que impiden el paso libre del robot móvil, es por esto que se toman en cuenta diferentes ambientes, algunos se muestran en la siguiente figura:

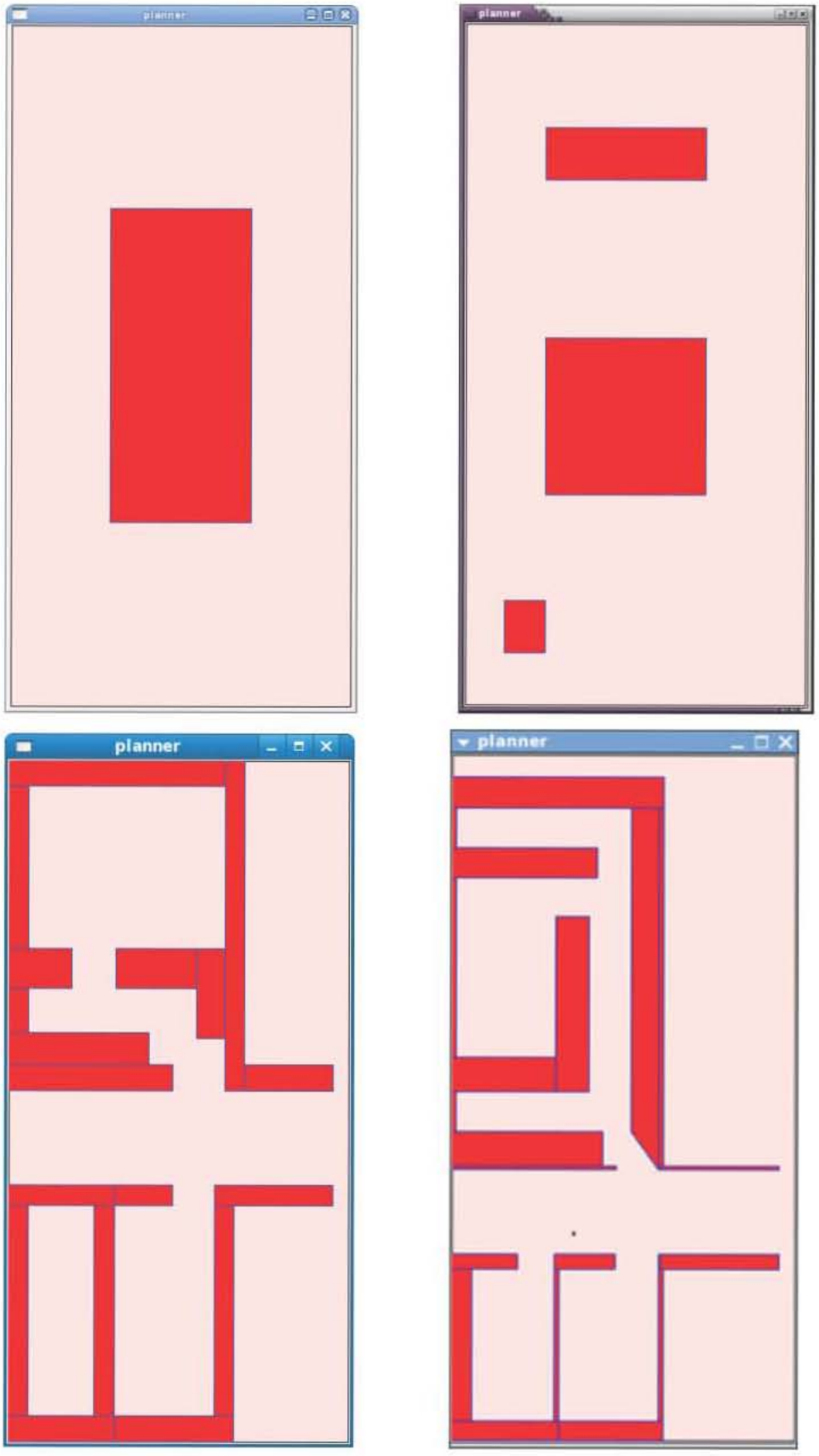



Figura 3.4: Mundos de navegación utilizados en el simulador ViRbot

Los medios ambientes reales son los más complejos y su representación se realiza describiendo la distribución del espacio en un plano a través de coordenadas cartesianas a partir de un punto de referencia. En la figura 3.5 se muestra la representación del laboratorio de biorobótica ubicando las mesas, puertas y pasillos a escala y posteriormente el mapa que se utiliza en el simulador.

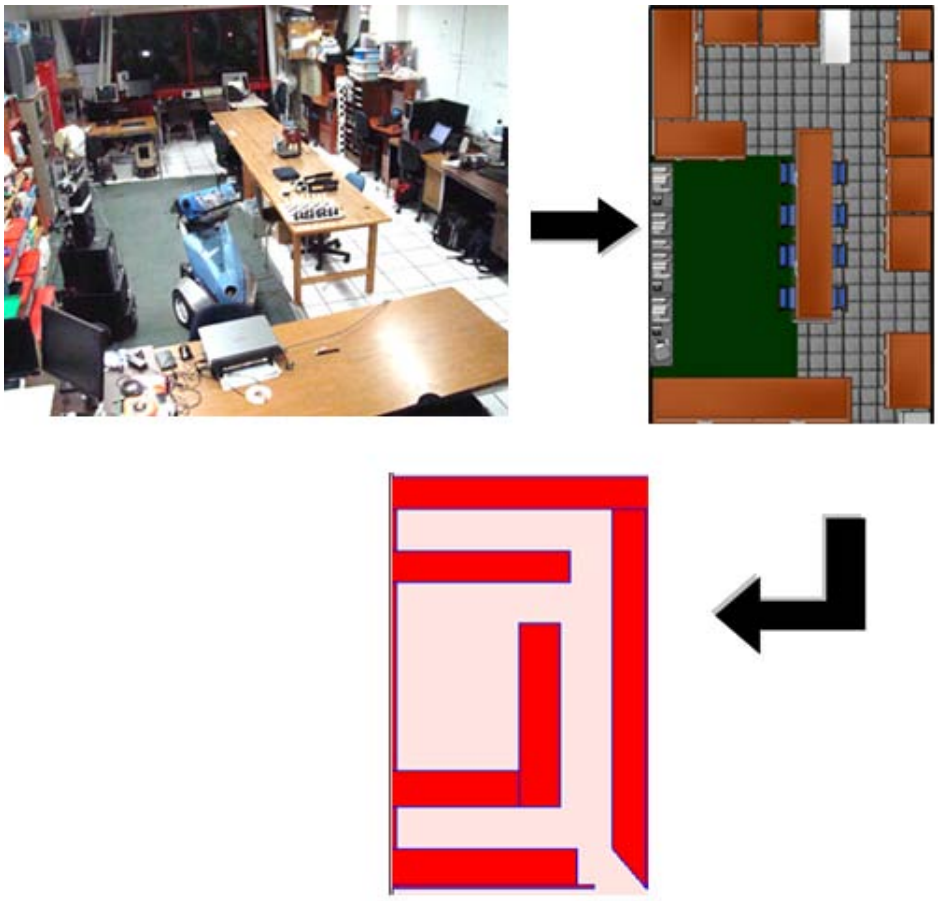


Figura 3.5: Proceso de clasificación

### 3.2.2. Observaciones

Para hacer una síntesis de la información obtenida y para facilitar el procesamiento computacional se realiza una reducción del espacio de las posibles observaciones en las cuales se pueden clasificar las lecturas de los sensores del robot en una determinada posición, es decir cada una de las lecturas posibles es clasificada dentro de un grupo representado por un vector con características semejantes a las muestras, de manera general este vector representa a aquellos que agrupa a través de un número entero. Para obtener estos vectores representativos se pueden utilizar clasificadores como los mapas autoorganizados, las redes neuronales artificiales basadas en la teoría de resonancia adaptativa y la

cuantización vectorial, cada uno de estos procesos arroja, a grandes rasgos, los mismos resultados, una serie de números enteros que enumeran a los vectores característicos de las categorías contenidas en el grupo de datos de la muestra.

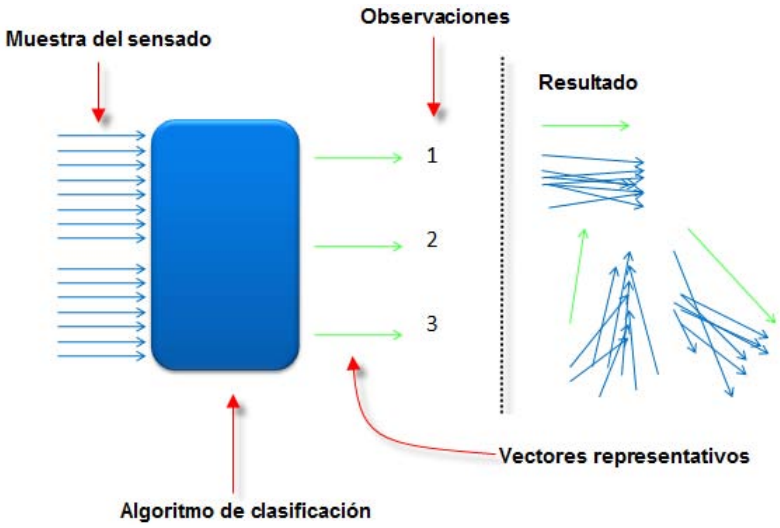



Figura 3.6: Proceso de clasificación

Este proceso se muestra gráficamente en la figura 3.6 donde se tiene una muestra de vectores que al ser ingresados a un algoritmo de clasificación arrojan vectores característicos, estos son una generalización de la muestra, a través de ellos se puede agrupar la muestra en $n$ conjuntos donde $n$ es el número de vectores característicos obtenidos por el clasificador,

Es importante decidir cuál de las técnicas es la más apropiada para esta aplicación, las redes neuronales ART ofrecen versiones para trabajar con números reales entre 0 y 1 de manera que para utilizar este algoritmo es necesario normalizar los datos de entrada al clasificador, para utilizar los mapas autoorganizados se requiere conocer un contexto amplio del programa y versatilidad para aceptar diferentes resultados con las mismas muestras, como la cuantización vectorial no tiene estos inconvenientes se decidió utilizar esta técnica para obtener las observaciones.

Una vez que se tienen los vectores característicos (número entero que representa la observación) se clasifican cada uno de los vectores que almacenan la información del sensado.

Con los vectores de la muestra ya representados por las observaciones del cuantizador vectorial se formará el modelo oculto de Markov.

### 3.2.3. HMM

Para generar una secuencia de estados válida basada en los modelos ocultos de Markov se requiere definir cuántos estados se utilizarán para representar correctamente las transiciones que se dan de una ubicación a otra en el mundo donde se está navegando, además es necesario definir un modelo inicial, este depende del ambiente que estemos utilizando (mundo 1 ó mundo 2) y es

obtenido directamente de los datos muestreados, tal como se describe en la sección de representación del ambiente, además es necesario utilizar las siguientes definiciones:

- $b_j(k)$ Número de observaciones con símbolo $k$ en el estado $j$ dividido entre el número de observaciones en el estado $j$.
- $a_{ij}(k)$ Número de transiciones del estado $i$ al estado $j$ dividido entre el número de transiciones de $i$ a cualquier estado.
- $\pi_i$ Número de veces que se empieza en el estado $i$ dividido entre el número de entrenamientos.

Después de calcular el primer modelo se realizan una serie de iteraciones que ajustan el modelo inicial de acuerdo a la probabilidad que brinda el algoritmo de Viterbi. Se calcula la secuencia de estados más probable con el modelo inicial y se recalcula este para que la secuencia de estados de la muestra tenga una gran probabilidad de ocurrencia, el proceso iterativo de ajuste de los valores del modelo oculto $h\,(A, B, \pi)$ continua hasta que existe convergencia en estos valores.

Una vez que se tiene el modelo oculto de Markov $\lambda$ se podrán probar vectores de sensado de trayectorias diferentes.

#### 3.2.3.1. Obtención del modelo

Para obtener un modelo a partir de las observaciones experimentales es necesario analizar las frecuencias de ocurrencia de cada observación en un estado determinado, pero no está basado únicamente en las estadísticas simples de muestreo sino en las probabilidades de observación de un símbolo en un estado determinado y en las probabilidades de transición de un estado a otro, de manera que es necesario evaluar que tan probable es obtener la secuencia de observacion obtenida a partir del modelo estadístico inicial y modificarlo iterativamente para maximizar la probabilidad de obtener la secuencia de observación experimental dado el modelo calculado. Formalmente se tiene lo siguiente:

$$\mathbf{O} = \{o_1, o_2, ...o_T\}$$

$$1 \leq i \leq N$$

$$1 \leq j \leq N$$

$$1 \leq t \leq T$$

$T$ número de observaciones en la secuencia

Para resolver el problema se puede definir una variable que represente la probabilidad de observar la secuencia $\boldsymbol{O}$ en el estado $i$ dado el modelo $\lambda$:

$$\alpha_t(i) = P(o_1 o_2 ... o_t, q_t = S_i|\lambda) \tag{3.1}$$

Calcular el valor de $\alpha$ implica evaluar la probabilidad de observación del símbolo actual para cada estado tomando en cuenta la probabilidad de transición de un estado a otro.

Para el primer elemento del vector ($t = 1$) no es necesario tomar en cuenta la transición de un estado a otro puesto que es el estado inicial, pero a partir del segundo elemento se debe evaluar la probabilidad de transición para evitar secuencias de estados no factibles por lo que utilizando inducción se obtienen las siguientes fórmulas.

**Algoritmo 3.1** Procedimiento hacia adelante (*forward*)

1. Inicialización

$$\alpha_1(i) = \pi_i b_i(O_i), \qquad \text{Para i=1,2, ... N}$$

2.Inducción

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}\right] b_j(O_t), \qquad \text{j=1,2,...,N}, \quad \text{t=2,...,T}$$

3.Terminación

$$\text{P(O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Una vez que se tiene una medida para evaluar el modelo actual es necesario actualizarlo para que sea óptimo, la actualización de los valores de la matriz de transición, la matriz de observación y el vector de pesos iniciales se realiza tomando en cuenta el concepto de $\alpha$ pero incorporando 3 probabilidades auxiliares que evalúan y normalizan los nuevos valores del modelo.

Se requiere en primer lugar calcular la probabilidad de observar una secuencia posterior al estado actual ($i$) dado el modelo $\lambda$, esto se realiza calculando el valor de esta probabilidad en el instante $t = T$ y luego propagando este valor hacia atrás para obtener el valor en el instante $t = 1$ a través de una expresión que relaciona ambos cálculos. Este concepto de procedimiento hacia atrás se almacena en una variable $\beta$ y se define en 2 partes: en el paso (1) se establece $\beta_T(i) = 1$ para todas las $i$. En el paso (2), se muestra que para estar en el estado $S_i$ en el tiempo $t$ y considerando la secuencia de observación en el tiempo $t+1$ hacia adelante, se toman todos los posibles estados $S_j$ en el tiempo $t+1$, tomando en cuenta la transición del estado $S_i$ al estado $S_j$ (el término $a_{ij}$), así como la observación $O_{t+1}$ en el estado $j$ (el término $b_j(O[t+1])$ y entonces se considera para la secuencia de observación parcial restante, desde el estado $j$ (el término $\beta_{t+1}(i)$ [19].

**Algoritmo 3.2** Procedimiento hacia atrás (*backward*)

1. Inicialización

$$\beta_T(i) = 1, \qquad \text{Para i=1,2, ... N}$$

2.Inducción

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), \qquad \text{i=1,2,...,N}, \quad \text{t=T-1,T-2,...,1}$$

Una vez que se tienen los valores de $\alpha$ y $\beta$ es necesario calcular la probabilidad de que el estado actual sea $i$ y que el estado siguiente sea $j$ dada la secuencia de observaciones y el modelo$\lambda$, el valor de esta probabilidad se almacena en la variable $\xi$ y depende de la probabilidad de transición entre un estado y otro y las observaciones en el estado $t$ y $t + 1$. De manera formal:

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j|O,\lambda \tag{3.2}$$

Tomando $\alpha$ y $\beta$ como auxiliares el cálculo de $\xi$ se realiza como lo muestra el algoritmo 3.3.

**Algoritmo 3.3** Cálculo de $P(q_t = S_i, q_{t+1} = S_j|O,\lambda)$

**for**($t$=1;$t$<numObservaciones;$t$++)
  **for**($i$=1;$i$<numEstados;$i$++)
    **for**($j$=1;$j$<numEstados;$j$++)

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{t=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}$$

    **end for**
  **end for**
**end for**

Además es necesario saber la probabilidad de que el estado actual sea $i$ dada la secuencia de observaciones presentada y dado el modelo $\lambda$. Los valores de esta probabilidad quedan almacenados en la variable $\gamma$. Basándose en $\alpha$ y $\beta$ el cálculo de $\gamma$ se realiza como lo muestra el algoritmo 3.4.

**Algoritmo 3.4** $P(q_t = S_i|O,\lambda)$

**for**($t$=1;$t$<numObservaciones;$t$++)
  **for**($i$=1;$i$<numEstados;$i$++)

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{t=1}^{N}\alpha_t(i)\beta_t(i)}$$

  **end for**
**end for**

Finalmente la actualización de las matrices del modelo $\lambda$ se puede efectuar con las siguientes ecuaciones:

$$\overline{\pi}_i = \frac{\alpha_0(i)\beta_0(i)}{\sum_{j=1}^{N}\alpha_T(j)} \tag{3.3}$$

$$\overline{a}_{ij} = \frac{\sum_{t=1}^{T}\xi_{t-1}(i,j)}{\sum_{t=1}^{T}\gamma_{t-1}(i)} \tag{3.4}$$

$$\overline{b}_i(k) = \frac{\sum_{t=1}^{T}\gamma_t(i)\delta(O_t,o_k)}{\sum_{t=1}^{T}\gamma_t(i)} \tag{3.5}$$

$$\delta(O_t,o_k) = \begin{cases} 1 & si \quad O_t = o_k \\ 0 & en \quad otro \quad caso \end{cases}$$

**Algoritmo 3.5** Probabilidad de una secuencia de observaciones dado un modelo

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

for(k=1;k<numeroDeSecuenciasDeObservacion;k++)

$$P_k = \sum_{i=1}^{N} \alpha_{T^{(k)}}(i)$$

end for

$$P(\mathbf{O}|\lambda) = \prod_{k=1}^{K} P_k$$

**Algoritmo 3.6** Reestimación del modelo $\lambda$ con múltiples secuencias de observación

$$\overline{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_{k-1}} \alpha_t^k(i) a_{ij} b_j(O_{t+1}^k) \beta_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_{k-1}} \alpha_t^k(i) \beta_{t+1}^k(j)}$$

$$\overline{b}_i(\ell) = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_{k-1}} \alpha_t^k(i) \beta_t^k(j) \delta(O_t, o_\ell)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_{k-1}} \alpha_t^k(i) \beta_t^k(j)}$$

$$\overline{\pi} \begin{cases} \pi_1 = 1 \\ \pi_i = 0 \quad i \neq 1 \end{cases}$$

#### 3.2.3.2. Ejecución del modelo

Se requiere un programa que reciba una muestra de las lecturas del robot obtenidas a través de una trayectoria en el mundo donde se entrenó el modelo **h**. Después deberá clasificar cada vector sensado para obtener el vector de observaciones y ejecutar el algoritmo de Viterbi para proporcionar la secuencia de estados más probable de acuerdo a las observaciones presentadas.

#### 3.2.3.3. Pruebas

Los resultados de cada ejecución del programa descrito en la sección anterior se almacenan en archivos de texto, de manera que se puede evaluar que tan certero es el resultado que brinda el *software* si se comparan los estados que presenta el algoritmo de Viterbi y los estados reales. Con este análisis se determina cuantas veces coincide la ubicación real con la ubicación calculada con el programa desarrollado.

### 3.2.4. Diseño modular

El proceso de clasificación por cuantización vectorial, mostrado en la figura 3.7, permite ingresar al programa un archivo de texto con vectores de sensado

como el mostrado en la figura 3.2 pero con más de 16 valores de los sensores, una vez q se ejecuta el proceso la salida es un conjunto de vectores mucho más pequeño que el conjunto muestra, estos vectores arrojados por la clasificación representan las características más significativas del conjunto de entrada por lo cual son llamados vectores característicos y en el desarrollo de este trabajo son usados como observaciones del móvil en un determinado estado.


Figura 3.7: Diagrama de bloques del proceso de cuantización vectorial

Cuando se ha finalizado la búsqueda de los vectores característicos el cuantizador vectorial puede trabajar con nuevos valores de sensado y asignar a estos el número entero que representa el *codebook* al cual se parece más el vector de sensado como se muestra en la figura 3.8, este proceso ayuda a simplificar la cuantificación de las observaciones que se pueden registrar en una posición determinada del ambiente.


Figura 3.8: Diagrama de bloques del proceso de codificación

La búsqueda del modelo $\lambda$ tiene como entrada una muestra de números enteros (obtenidos a través de cuantización vectorial) que representan los valores de sensado almacenados cuando el robot navega por el ambiente que se desea caracterizar. La salida de este módulo es una cuantificación del número de veces que se ve una observación a través de los estados definidos en el programa entre el número total de observaciones que son posibles estando en ese estado.

Además se arroja una relación que permite establecer la posibilidad de ir de un estado $n$ a otro $m$ con cierta probabilidad gráficamente la entrada y la salida de este proceso se muestran en la figura 3.9.

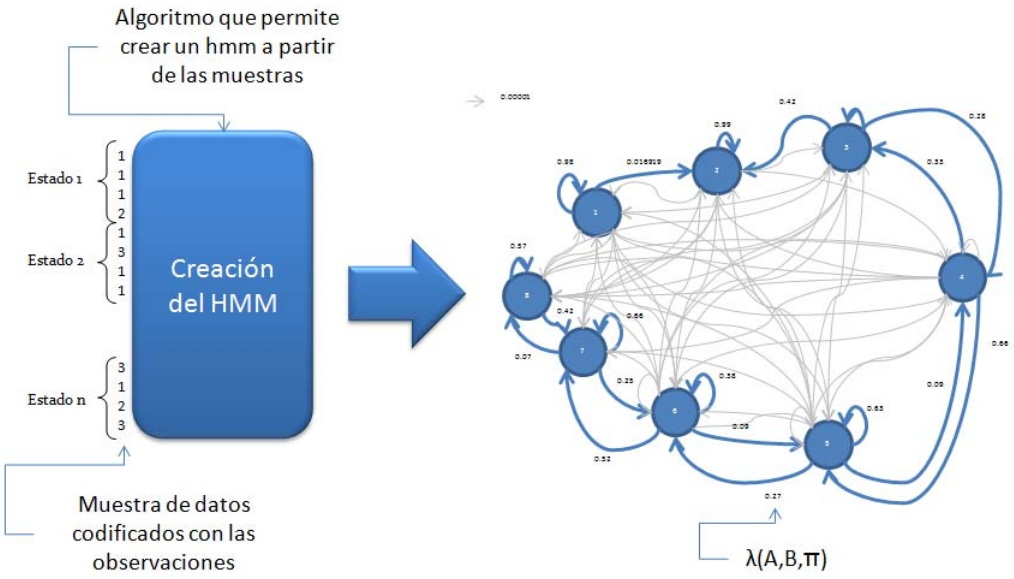



Figura 3.9: Diagrama de bloques de la creación del HMM

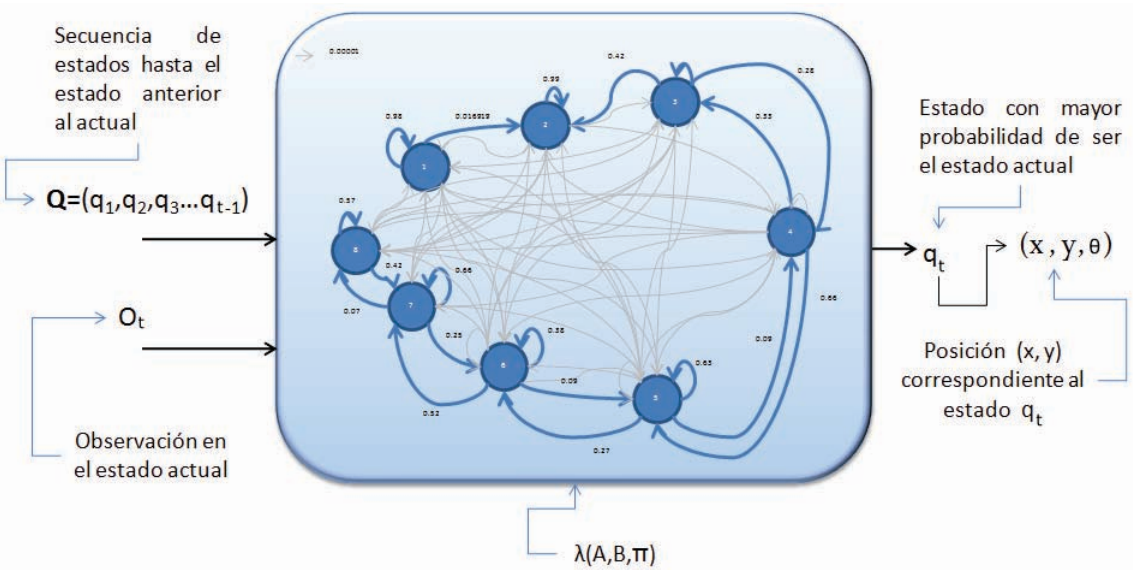



Figura 3.10: Diagrama de bloques de la ejecución de un HMM

La busqueda de la secuencia de estados más probable es la etapa final del trabajo y su resultado es ligado directamente al proceso de navegación, sus

entradas y salidas se muestran en la figura 3.10, se ingresa al proceso del algoritmo de Viterbi la secuencia de estados que se ha registrado hasta el momento $t-1$ $(Q = q_0, q_1 ... q_{t-1})$ y la observación en el momento actual $O_t$, lo que se obtiene es un número entero que representa al estado actual$q_t$ con el cual se puede estimar la posición $(x, y, \theta)$ del móvil.

### 3.2.5. Diagramas de flujo

El diagrama de flujo del proceso descrito en la sección 3.2.3.1 comienza con la inicialización de los parámetros la cual solo cuantifica la ocurrencia de cada observación en los estados que describen el medio ambiente, después evalua el modelo $\lambda$ compuesto por las matrices $a$, $b$ y $\pi$ y cuando no se obtienen los resultados deseados se recalculan los valores de los elementos del modelo $\lambda$ con las ecuaciones 3.3 para actualizar $\pi$, 3.4 para $a$ y 3.5 para $b$, cuando el modelo es convergente se han encontrado los parámetros adecuados..

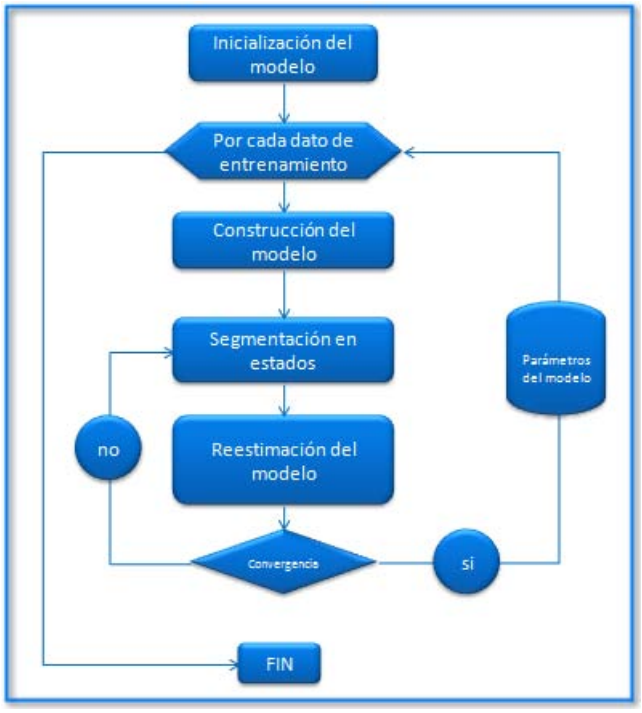



Figura 3.11: Cálculo de los parámetros de un HMM

Los valores de sensado deben pasar por diferentes procesos en el simulador con el fin de realizar y evaluar la funcionalidad de los módulos desarrollados, éstos se muestran en la figura 3.12. El simulador genera valores de sensado correspondientes al trayecto que va realizando y estos se almacenan en un archivo de acuerdo al formato ya mencionado, se ejecuta el programa *pathHmm* con los datos del simulador y el modelo $\lambda$ ya encontrado, además para visualizar de manera más sencilla el trayecto generado indirectamente por el algoritmo de Viterbi se escribe un archivo que permite mostrar gráficamente en el simu-

lador la trayectoria que se aproxima de acuerdo a las observaciones realizadas anteriormente con la finalidad de medir qué tan semejantes son los trayectos, evaluar la funcionalidad del *software* programado y determinar si el sistema podía ejecutarse con el robot real.

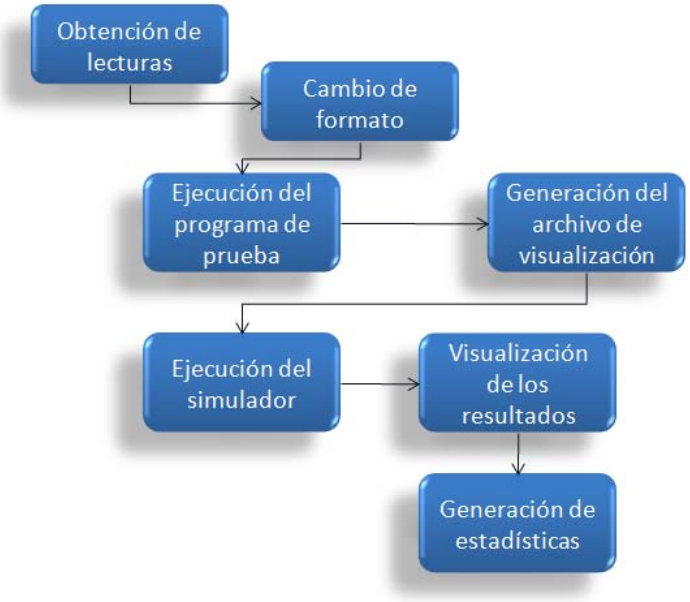



Figura 3.12: Trayecto de los datos del simulador

El diagrama de flujo del algoritmo de cuantización vectorial se muestra en la figura 3.13, se requiere obtener un conjunto de vectores generados por los sensores del robot (real o simulado) a lo largo de todo el ambiente que se desea representar. Se hace la sumatoria de ada componente y se divide entre el tamaño de la muestra para obtener un centroide al cual se le suma y resta una cantidad $\epsilon$ con el fin de obtener dos centroides, estos serán la base para agrupar al conjunto de datos en dos grupos, dependiendo del centroide al cual se asemejen más. Posteriormente se recalculan los centroides de cada grupo y reagrupa el conjunto completo de acuerdo a los nuevos centroides, este proceso se repite hasta alcanzar la estabilidad deseada. Cuando los vectores ya no cambian de grupo, ni se modifican los centroides cuando se recalculan, se verifica si ya se alcanzó el n´pumero de centroides deseado, si no es así se vuelve a sumar $\epsilon$ a cada centroide obteniendo el doble de centroides que ya se tenían y repitiendo el proceso de reagrupación y recalculo hasta la estabilidad.

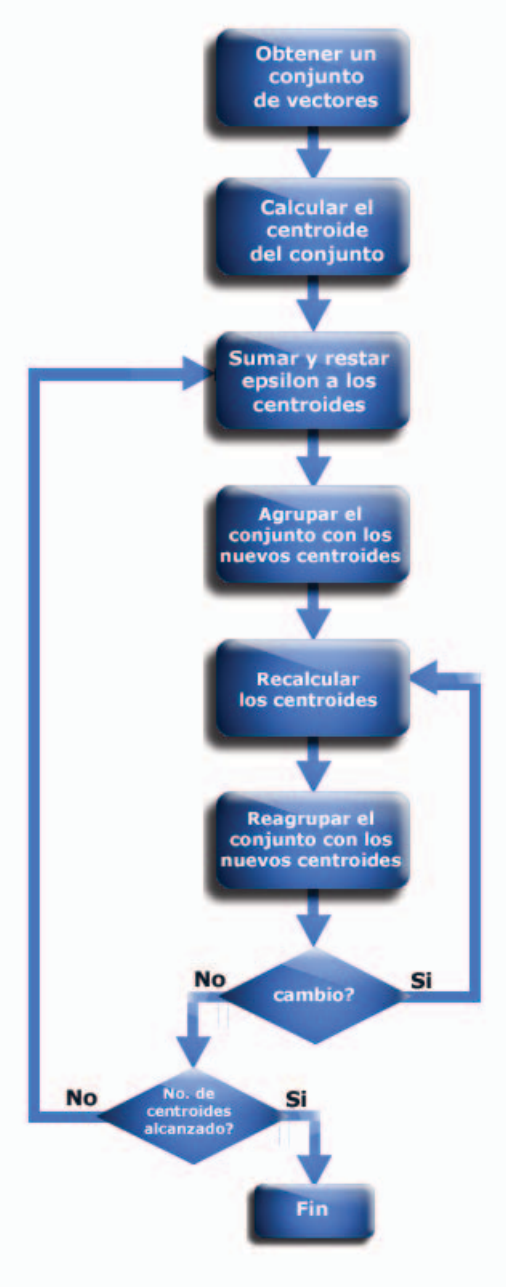



Figura 3.13: Proceso de cuatización vectorial

## 3.3. Codificación

### 3.3.1. Variables

***observations***: vector de M componentes que se obtiene de sensar el ambiente real o en el simulador.

Para HMM:

***ais***: matriz que alberga la probabilidad de transición de un estado $i$ a un estado $j$, donde $1 < i < N$, $1 < j < N$, $N$ es el número máximo de estados.

***bis***: matriz de observación, muestra la probabilidad de observar el símbolo $O_t$ en el estado $i$, donde $1 < i < N$.

***pis***: vector que define las probabilidades iniciales, es decir la probabilidad de iniciar en el estado $i$, donde $1< i < N$, siendo $N$ el número máximo de estados.

***delta***: almacena la probabilidad de la secuencia de estados analizada.

***psi***: almacena la secuencia de estados analizada.

$\alpha$: almacena la probabilidad de observación del símbolo actual para cada estado tomando en cuenta la probabilidad de transición de un estado a otro.

$\beta$: almacena la probabilidad de observar una secuencia posterior al estado actual $(i)$ dado el modelo $\lambda$.

$\xi$: almacena la probabilidad de que el estado actual sea $i$ que el estado siguiente sea $j$ dada la secuencia de observaciones y el modelo$\lambda$.

$\gamma$: almacena la probabilidad de que el estado actual sea $i$ dada la secuencia la secuencia de observaciones presentada y dado el modelo $\lambda$.

Para Viterbi:

Se utilizan *ais*,*bis* y *pis* que llevan los datos del modelo de Markov descritos anteriormente.

$q$: apuntador en el que se almacena la secuencia de estados.

$\delta$: matriz que almacena la probabilidad del mejor camino de estados hasta el estado actual habiendo visto las observaciones anteriores.

$\varphi$:matriz en la que almacena para cada tiempo el valor del estado que maximiza la probabilidad de observación.

## 3.4. Experimentos

Para realizar pruebas con cada uno de los modelos obtenidos y obtener los resultados de la implementación se requieren muestras de las lecturas de los sensores del robot a través del sistema ViRbot y a través de los sensores reales. Se ejecutan diferentes trayectorias que permiten obtener varias secuencias de vectores, estos se codifican con los vectores representativos que arrojó el cuantizador vectorial y se obtienen las secuencias de observaciones que sirven para probar los modelos ocultos de Markov obtenidos. En la figura 3.14 se observa un trayecto sencillo del robot en el simulador y las observaciones registradas en cada paso con las cuales se puede obtener la estimación del estado y de la posición del móvil.

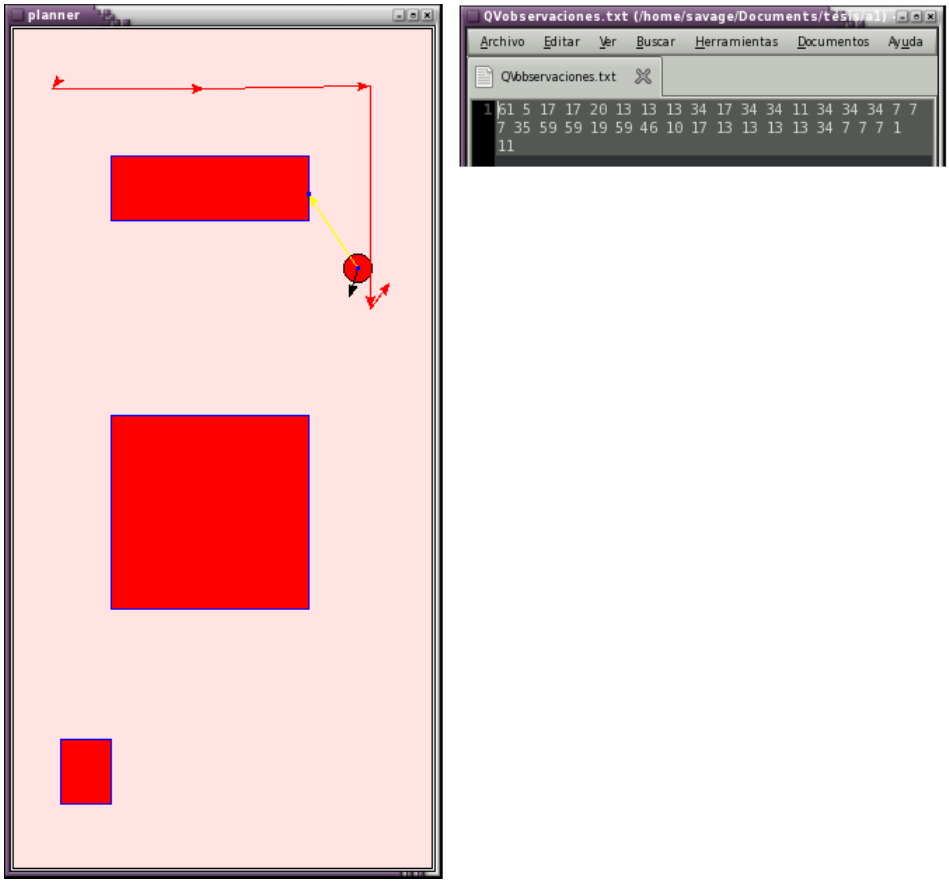



Figura 3.14: Ejemplo de un trayecto con 37 pasos basado en 64 tipos de observaciones

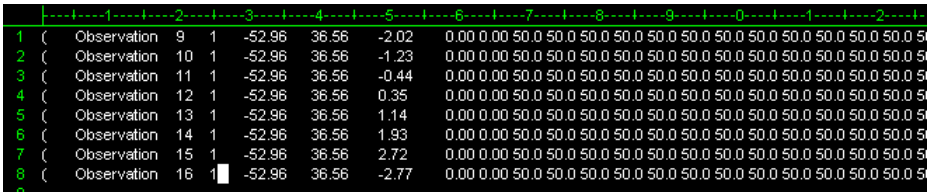
La secuencia de observaciones se procesa con un programa que obtiene el estado actual más probable basado en la secuencia de estados anteriores más probable, al conocer el estado actual se puede identificar la posición del robot (x,y,$\theta$). Estos datos permiten generar una trayectoria teórica que al ser comparada con la trayectoria real permite evaluar el resultado arrojado por el modelo oculto de Markov. En la medida en la que los resultados sean cercanos a la trayectoria real se puede utilizar este método para ubicar al robot mejorando su auto-localización y facilitando su llegada al destino requerido.

Para observar la trayectoria teórica en el simulador se crea un archivo de texto que contiene las lecturas correspondientes a los estados que conforman la trayectoria, el archivo es leído por el simulador y muestra la trayectoria de acuerdo a la posición $x$ y $y$ de cada estado.

La realización de las pruebas en el robot real es un proceso más complejo dividido en dos fases: la primera que verifica que los resultados del modelo funcionen en las posiciones reales y la segunda que prueba la correcta comunicación entre el proceso de navegación y la estrategia de los modelos ocultos para la autolocalizacion por medio de las observaciones.

Ambas fases requieren la toma de lecturas, para realizar este proceso se

utilizó un dispositivo de control de movimiento y toma de lecturas manual. Se utilizó un joystick y el *software* de muestreo del laboratorio de biorobótica, este permite a través de los movimientos de la palanca, el envio de comandos al móvil sin necesidad de especificar distancias, únicamente con el movimiento manual; se toman las lecturas del sensor láser al presionar un botón y se almacenan en archivos de texto fácilmente manipulables con un formato específico mostrado en la figura 3.15.



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | ( | Observation | 9 | 1 | -52.96 | 36.56 | -2.02 | 0.00 0.00 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 5 |
| 2 | ( | Observation | 10 | 1 | -52.96 | 36.56 | -1.23 | 0.00 0.00 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 5 |
| 3 | ( | Observation | 11 | 1 | -52.96 | 36.56 | -0.44 | 0.00 0.00 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 5 |
| 4 | ( | Observation | 12 | 1 | -52.96 | 36.56 | 0.35 | 0.00 0.00 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 5 |
| 5 | ( | Observation | 13 | 1 | -52.96 | 36.56 | 1.14 | 0.00 0.00 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 5 |
| 6 | ( | Observation | 14 | 1 | -52.96 | 36.56 | 1.93 | 0.00 0.00 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 5 |
| 7 | ( | Observation | 15 | 1 | -52.96 | 36.56 | 2.72 | 0.00 0.00 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 5 |
| 8 | ( | Observation | 16 | 1 | -52.96 | 36.56 | -2.77 | 0.00 0.00 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 50.0 5 |

Figura 3.15: Formato con el cual se almacenan las lecturas del láser real utilizando joystick

Las lecturas necesarias para el entrenamiento del modelo deben abarcar todo el escenario del mundo con el que se desea trabajar, y debe tener una secuencia determinada, para los mundos contemplados en este trabajo se utilizaron un mínimo de 600 lecturas y un máximo de 1500. Con esta muestra se generó el cuantizador vectorial y el modelo oculto.

**Primera fase**

Se llevó al robot a través de una trayectoria cualquiera manipulado por el *joystick*, se utilizaron trayectorias de prueba cortas de alrededor de 4 [m] se almacenan las lecturas obtenidas en un archivo, estas lecturas se cuatizan, se procesan con el modelo oculto y se analizan los estados arrojados, se comparan con la trayectoria real y se analizan los resultados.

**Segunda fase**

El objetivo de esta fase es comprobar si el saber la posición actual de una manera aproximada ayuda al móvil a alcanzar la posición objetivo. Se programó la comunicación entre el *software* que mueve al robot y el que brinda la posición de acuerdo al modelo de Markov y se comparan ambas trayectorias con los mismos puntos de origen y destino analizando las diferencias entre un camino y otro , las colisiones, la velocidad y la cercanía al destino final.

Mantenimiento

El movimiento del robot cambia constantemente debido a situaciones de *hardware*, rigidez de los motores, peso del robot, centro de masa y desgaste de piezas (engranes, llantas, cintas de movimiento[2]) por lo cual es recomendable que se realice la caracterización del móvil cuando se navegue en escenarios nuevos y cuando los movimientos sencillos no se concreten, es decir que ante dos o tres comandos no se llegue al punto objetivo, el proceso de caracterización se describe en el apéndice.

El modelo $\lambda$ depende completamente del escenario en el cual se va a navegar por lo cual ante un nuevo ambiente es necesario volver a ejecutar el proceso mostrado en el apéndice A-3.

[2]Las bandas fueron utilizadas en el robot que participó en el torneo Mexicano de Rrobótica en el 2009, el robot actual no las utiliza

# Capítulo 4

# Implementación en el robot

*«El calor no significa nada para él y está construido para poca gravedad y suelo accidentado. Está a prueba de averías..., o por lo menos, debería estarlo.»*

***Isaac Asimov***

El punto clave de este trabajo es la implementación del *software* desarrollado en un agente autónomo buscando una mejora en el proceso de navegación, para este fin se utilizó el robot del laboratorio de Biorobótica mostrado en la figura siguiente:

Figura 4.1: Robot Pac-ito. Laboratorio de Biorobótica FI-UNAM

Como ya se ha mencionado la implementación en el robot consta de dos fases una de verificación del funcionamiento del *software* ya probado en el simulador y otra de la integración del *software* con el proceso de navegación, para la primera no es necesario ningún programa adicional al ya codificado debido a

que únicamente se toman muestras de sensores reales, se realiza el proceso de cuantización vectorial, generación del modelo oculto y se prueba la eficacia del *software* en base a la posición real y a la posición arrojada por la secuencia de estados más probable.

Esta fase no arroja buenos resultados cuando las regiones contempladas en el modelo son grandes por lo que es necesario dividir el escenario en un mayor número de secciones. Además se requiere un mayor número de muestras registrando con especial cuidado el valor de los ángulos utilizados.

Se utilizó un *joystick* para guiar el móvil en el escenario y un módulo de *software* disponible en el laboratorio que permite manipular la base del robot y el sensor láser, conforme las pruebas avanzaron y el número de lecturas necesario para el entrenamiento fue mayor se utilizó un *script* que contiene las instrucciones necesarias para mover al robot en sustitución al dispositivo manual (*joystick*), esto agiliza la obtención de lecturas y aumenta la exactitud de los datos utilizados en el entrenamiento del modelo. Las lecturas son almacenadas en archivos que tienen el mismo formato del simulador lo que permite ingresar los archivos directamente al programa que realiza la cuantización vectorial en el proceso de entrenamiento y al programa de pruebas cuando se busca reconocer alguna trayectoria.

El proceso para completar una fase de verificación del *software* en el robot, con sensado real y trayectoria real, es el siguiente:

**Obtener lecturas de entrenamiento**. Se divide el escenario en regiones, se deben conocer los límites de cada región y las medidas de estas. Cada región representa un estado del HMM. El robot es colocado estratégicamente en cada una de las regiones, de preferencia iniciando siempre en ángulo 0, la posición del robot en la región debe facilitar el reconocimiento, debe ser lo más general posible. Se establece un desplazamiento angular fijo y se realizan las lecturas correspondientes a 360 °.

**Cambio en el formato de lecturas**. El archivo de lecturas es procesado con un primer programa (processLocHmm) para cambiar su formato de almacenamiento y eliminar datos innecesarios para la cuantización vectorial.

**Obtener los vectores representativos**. Se ingresa el archivo de lecturas al cuantizador vectorial indicando el número de centroides deseado $n$. La salida es otro archivo de texto que contiene $n$ vectores $m$ dimensionales donde $m$ es el tamaño de los vectores de la muestra. Cada vector es identificado por un número entero (en este contexto llamado observación) utilizado para determinar la probabilidad de observar un patrón en una región específica.

**Obtener el modelo oculto de Markov**. Cada una de las lecturas de la muestra se compara con los $n$ vectores arrojados por el cuantizador vectorial. El vector más parecido es el que le proporcionará su identificador o número de observación, lo que simplifica el conteo computacional de las probabilidades de observación en cada estado. La salida de este proceso es un archivo .dat que contiene la probabilidad inicial (en este caso la probabilidad de iniciar en un estado determinado es la misma para todos los estados), una matriz que muestra la probabilidad de ir de un estado a otro y una matriz que presenta la probabilidad de sensar una observación en un estado determinado.

**Obtener una secuencia de prueba**. Se manipula el robot con el joytick, el sistema ViRbot o comandos directos a través de una trayectoria y se almacenan las lecturas obtenidas a lo largo del recorrido.

**Ejecución de la prueba en la muestra**. El archivo obtenido se procesa en un *software* de prueba devolviendo una secuencia de estados (la secuencia con mayor probabilidad) y una secuencia de coordenadas ($x$,$y$,$\theta$) que describen la region en la que se ubica el robot de acuerdo a los sensores.

**Visualización de los resultados**. Uno de los archivos generados por el *software* de prueba puede ser utilizado por el simulador ViRbot para observar directamente la trayectoria del móvil.

De la visualización se puede determinar rápidamente y de manera muy superficial si la trayectoria real es la misma que la indicada por el reconocimiento de los modelos ocultos de Markov, para una verificación más formal se obtiene una diferencia de posiciones en cada movimiento, la diferencia entre la posición real y la posición calculada por los HMM indica qué tan efectivo es el reconocimiento de la posición.

Cabe mencionar que para tomar las lecturas de prueba y para efecuar el reconocimiento por zonas se coloca al robot en una posición inicial perfectamente determinada ($x$,$y$,$\theta$) y esa información debe considerarse en el programa utilizado para almacenar las lecturas de los sensores.

Se considera que la fase de verificación de *software* brinda buenos resultados si el reconocimiento es correcto en el 90 % de las regiones que forman la trayectoria.

Cuando la fase de verificación ha resultado satisfactoria se pasa a la fase de integración; en esta, si bien no es necesario desarrollar un sistema entero son necesarias varias interfaces de acoplamiento entre el simulador ViRbot y el sistema programado en este trabajo.

## 4.1. Interfaces de acoplamiento

**Posición y lecturas al sistema**

Se encarga de transmitir adecuadamente la posición que aprentemente tiene el robot de acuerdo al sistema ViRbot y a los comandos de movimiento brindados además de las lecturas que ha sensado el robot en esa posición,

Para obtener la posición se recurre a los atributos ($x$,$y$,$\theta$) que se manejan en la estructura de datos correspondiente, las lecturas de los sensores se almacenan en un archivo de texto donde se encuentran disponibles para que los utilice el programa de HMM.

Una vez que estos datos son procesados por el sistema desarrollado, se obtiene la secuencia de estados más probable, esta arroja el estado actual del móvil según la lectura obtenida, una vez que se tiene el estado actual se ubica cual es la posición ($x$,$y$,$\theta$) correspondiente a esa región y se devuelve como salida para la ponderación.

**Ponderación de resultados.**

Debido a que el reconocimiento de la región tiene un error es necesario ponderar la posición que reconoce el sistema HMM y la posición que registra el sistema ViRbot. Debido a que los experimentos muestran un mayor error en la posición brindada por ViRbot ésta tendrá un menor valor de ponderación.

Si la posición arrojada por ViRbot y por el HMM tienen una cercanía de 2 [dm] y su diferencia en ángulo es menor a 20° se considera una diferencia aceptable y se tomará el promedio de las tres medidas.

Si la diferencia entre una posición y otra no es aceptable se debe aplicar la siguiente consideración:

$$(X_v*0.4+X_h*0.6)/2=X_v$$
$$(Y_v*0.4+Y_h*0.6)/2=Y_v$$
$$(\theta_v*0.30+\theta_h*0.70)/2=\theta_v$$

Figura 4.2: Ponderación utilizada

Esto en virtud de 2 cosas:

El error experimental que se ha observado en ViRbot es de 30.6 % mientras que con los HMM se tiene en promedio 16.12 % de fallos . Estos porcentajes se obtienen de la siguiente forma:

Se cuantifica el número de veces promedio que ViRbot tiene una diferencia aceptable con la posición real después de una secuencia mayor a 10 instrucciones de movimiento de igual manera se contabilizan los aciertos de aproximación a la posición usando la implementación de HMM. Si se hacen las respectivas proporciones con respecto a cada uno de los parámetros se obtienen las consideraciones de ponderación siguientes:

0.42 y 0.58 para $x$

0.43 y 0.57 para $y$

0.38 y 0.62 para $\theta$

Donde el primero número es la ponderación para el elemento de la posición que calcula ViRbot y el segundo es para ponderar el elemento que se obtiene de la estimación a través del algoritmo de Viterbi.

Estas proporciones funcionan como base para contemplar diferentes opciones y probar la que mejor funcione en el módulo de ponderación de resultados. Las opciones que se probaron son:

1)
$$(X_v*0.2+X_h*0.8)/2=X_v$$
$$(Y_v*0.2+Y_h*0.8)/2=Y_v$$
$$(\theta_v*0.15+\theta_h*0.85)/2=\theta_v$$

2)
$$(X_v*0.4+X_h*0.6)/2=X_v$$
$$(Y_v*0.4+Y_h*0.6)/2=Y_v$$
$$(\theta_v*0.30+\theta_h*0.70)/2=\theta_v$$

3)
$$(X_v*0.41+X_h*0.59)/2=X_v$$
$$(Y_v*0.38+Y_h*0.62)/2=Y_v$$
$$(\theta_v*0.26+\theta_h*0.74)/2=\theta_v$$

4)
$$(X_v*0.45+X_h*0.55)/2=X_v$$
$$(Y_v*0.45+Y_h*0.55)/2=Y_v$$
$$(\theta_v*0.4+\theta_h*0.60)/2=\theta_v$$

5)
$$(X_v*0.5+X_h*0.5)/2=X_v$$
$$(Y_v*0.5+Y_h*0.5)/2=Y_v$$
$$(\theta_v*0.5+\theta_h*0.5)/2=\theta_v$$

Obteniendo los siguientes resultados :

Primera ponderación 87.9 % de reconocimiento.
Segunda ponderación 92 % de reconocimiento.
Tercera ponderación 87 % de reconocimiento.
Cuarta ponderación 88.6 % de reconocimiento.
Quinta ponderación 72.2 % de reconocimiento.
Estos resultados justifican la poderación propuesta

**Uso de la posición ponderada**

Una vez que el módulo anterior obtiene una posición ($x$,$y$,$\theta$) estimada se utiliza como punto de partida para alcanzar el nodo siguiente en la trayectoria.

## 4.2. Pruebas de integración

Para que la prueba se catalogue como exitosa se deben concretar adecuadamente los siguientes procesos:

- Generar el archivo de texto con las observaciones
- Verificar que las observaciones correspondan a cada «paso» dado por el móvil (cada instrucción enviada, por ejemplo mv 2.4 1.5).
- Verificar que se ejecute correctamente el programa de HMM.
- Verificar que se obtenga un estado adecuado o realista.
- Obtener la estimación de la posición correspondiente al estado arrojado por el algoritmo de Viterbi.
- Ponderar las posiciones que se tienen
- Establecer la ponderación como la nueva posición real del robot
- Utilizar la nueva ubicación en los cálculos necesarios para alcanzar la posición objetivo.

En la fase final se cambio el formato del archivo generado en tiempo real que contiene las lecturas para eliminar funciones inecesarias al momento de la navegación ahorrando tiempo de procesamiento al costo de perder legibilidad en el archivo.

## 4.3. Caracterización

El agente manejado en el laboratorio de biorobótica es un sistema complejo que emplea diversos elementos mecánicos y eléctricos, al igual que las herramientas que utilizamos para medir magnitudes físicas (vernier, amperímetro) no son sistemas ideales sino reales y su uso implica un error inherente a sus componentes y su funcionamiento, esto hace necasario un análisis integral del sistema actual para determinar el comportamiento del sistema en ejecución, una vez que este comportamiento es correctamente descrito (cuantitativa y cualitativamente) se puede manipular con certeza el escenario en el que se trabaja y y por lo tanto hacer las correcciones necesarias para obtener el comportamiento

deseado a través de estimaciones de parámetros o regresiones no lineales evitando que se afecte la veracidad del muestreo y sobre todo acoplando los valores teóricos con los brindados en particular por el sistema utilizado.

Para caracterizar el mecanismo de giro del robot se enviaron comandos al robot con un angulo determinado y se repitió la petición hasta completar un giro de 360°, se registró el ángulo enviado en el comando y el ángulo que el móvil realmente rotó.

Los valores usados para caracterizar el mecanismo se muestran en la figura siguiente:

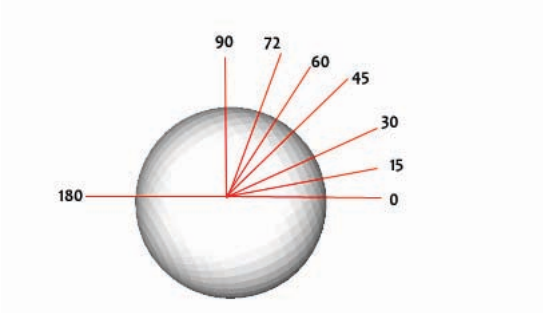



Figura 4.3: Valores de muestreo

Los datos obtenidos son los siguientes:

| Grados | Radianes | Muestra |
|---|---|---|
| 0 | 0.000 | 0 |
| 15 | 0.262 | 0.24 |
| 30 | 0.524 | 0.43 |
| 45 | 0.785 | 0.68 |
| 60 | 1.047 | 0.91 |
| 72 | 1.257 | 1.15 |
| 90 | 1.571 | 1.45 |
| 180 | 3.142 | 3.025 |
| 360 | 6.283 | 6.05 |

Figura 4.4: Datos de muestreo

Con este muestreo se utilizó un modelo de regresión no lineal para obtener la función correspondiente al comportamiento del ángulo, la regresión logarítmica. Ésta arroja la siguiente fórmula:

$$\theta = 0.45318 * 1.60106^{\theta C}$$

que es de la forma $y = b * m^x$

La funcion calculada a través de la estimación logarítmica tiene un error de 0.088713458 para el cálculo de $m$, y un valor de 0.233559824 para el cálculo de $b$, si se considera que los ángulos utilizados en su mayoría son menores a 1.57 el error de $b$ no es óptimo para ser utilizado en la planeación de rutas. Las regresiones polinomiales tampoco arrojan buenos coeficientes de error, incluso se utilizaron *splines* para obtener los valores de los ángulos no muestreados y no se obtuvieron buenos resultados. La solución a este punto es una interpolación

más simple, cada dos puntos se determinan los valores del intervalo únicamente en base a la cantidad de muestras que se desea obtener y al rango de los valores conocidos, por ejemplo:

| Grados | Radianes | Muestra[Rad] |
|---|---|---|
| 15 | 0.26179939 | 0.24 |
| 30 | 0.52359878 | 0.43 |

Figura 4.5: Datos de muestreo

Se tienen 15 grados de diferencia entre ambos valores muestreados, para obtener un valor entre el intervalo $[15, 30]$ se ocupa la siguiente expresión:

$$nuevoAngulo = (((M_s \quad M_i)/G_d) * ((angulo \quad RAD_i) * 180/\pi)) + M_i$$

donde
$M_s$ : muestra del intervalo superior
$M_i$: muestra del intervalo inferior
$G_d$: Número de elementos enteros en el intervalo de la muestra en grados
$angulo$: Radianes que se desea rotar al robot
$RAD_i$: El valor teórico que corresponde con $M_i$ en la tabla de datos de muestreo

## 4.4. Pruebas generales de funcionamiento

Cada prueba corresponde a una incógnita generada al momento del diseño del *software*, al finalizar los experimentos se podrá dar respuesta a cada pregunta de acuerdo al análisis realizado.

### 4.4.1. Reconocimiento

**¿Mientras mayor es el número de observaciones usado el porcentaje de reconocimiento de región es mayor?**

**Prueba 1. Reconocimiento de región en base al número de observaciones.**

Para cada uno de los 4 ambientes utilizandos en el simulador (figura 4.6) , se realizan 2 ejecuciones en las cuales el trayecto es superior a 30 instrucciones, y se toman en cuenta 20 pasos en los que se ejecuta el programa de HMM, el paso es la ejecución de una de las 30 instrucciones que contiene el *script*, en cada paso se obtiene un estado a través de Viterbi por lo cual existe una muestra de 40 corridas en las que se verifica que el estado proporcionado sea igual al estado simulado, si no es así se cuantifica un error. Este experimento se realiza con 8, 12, 14 y 24 estados y para cada uno de estos se prueba con 8, 16, 17, 32, 64, 112, 128, 184 y 256 observaciones. De forma que para cada valor del número de estados se obtiene como variable independiente el número de observaciones y como variable dependiente el porcentaje de reconocimiento de la región, este porcentaje es la proporción del número de veces en las que el

estado mostrado por el sistema ViRbot es igual a la región en la que se ubica el móvil entre las 40 ejecuciones del algoritmo de Viterbi que se tienen por cada valor del número de observaciones.

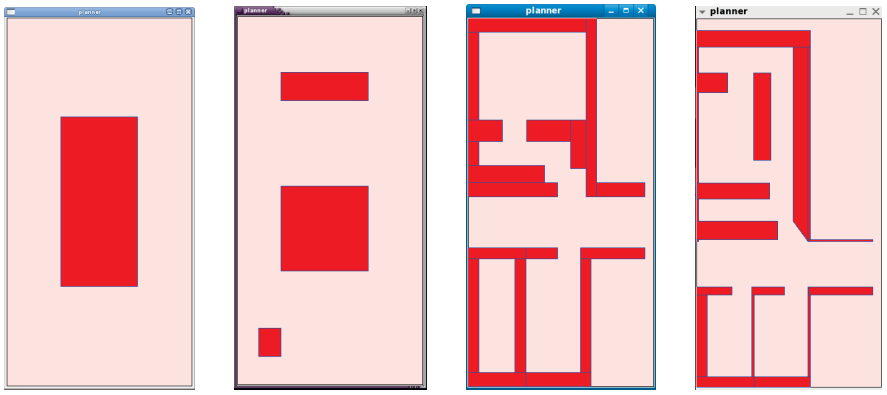



Figura 4.6: Medios ambientes utilizados en el simulador

**¿Cual algoritmo de clasificación es mejor para este trabajo ART o CV? ¿El algoritmo elegido afecta el resultado final?**

**Prueba 2. Reconocimiento de región en base al algoritmo de clasificación**

Se busca comparar dos algoritmos en un mismo medio ambiente con un número muy semejante de observaciones, a través de la prueba anterior se determina cuál es el número de observaciones óptimo para cada medio ambiente utilizado por lo que se toma la configuración que ofreció mejores resultados para cada escenario. De esta forma por cada medio ambiente se realiza lo siguiente:

Se obtiene a través de cuantización vectorial los vectores característicos que actúan como observaciones en el cálculo del HMM y del algoritmo de Viterbi, se realizan 5 recorridos con el simulador de 30 instrucciones cada uno, tomando en cuenta que el algoritmo de Viterbi sólo se ejecuta en 20 de los 30 pasos y se cuantifica el número de veces que no coicidieron el estado simulado con el estado proporcionado por el algoritmo de Viterbi. El porcentaje de reconocimiento obtenido es la proporción de esta cuantificación de errores entre 100.

El procedimiento se repite pero obteniendo los vectores característicos a través de la red neuronal *fuzzy* ART

Además se añade ruido a las lecturas obtenidas y se realiza el mismo proceso con el fin de evaluar la semajanza de estos resultados con el comportamiento en el móvil real.

### 4.4.2. Tiempos de ejecución

La relevancia de estas pruebas es conocer el promedio de tiempo que se requiere para lograr un resultado cuando el móvil se encuentre navegando por lo que en la realización de estos recorridos se utiliza el robot y se deja de lado el simulador, tanto para el movimiento como para la toma de lecturas de los sensores.

**¿El algoritmo de CV se ejecuta en tiempo viable?**

**Prueba 3. Tiempos de ejecución de la cuantización vectorial.**

Se tomaron 1649 lecturas con los sensores a través del robot y se busca obtener de 16 a 256 vectores característicos con el fin de detectar algún problema.

Si por algún valor de los parámetros el tiempo de ejecución es demasiado largo o se generan errores se deben variar los valores buscando una configuración para el buen desempeño del algoritmo.

Se debe medir el tiempo que tarda el algoritmo en arrojar el número de observaciones pedidas por el usuario.

**¿La búsqueda del modelo $\lambda$ se ejecuta en tiempo razonable?**

**Prueba 4. Tiempo de ejecución de entrenamiento del HMM**

Para evaluar la cantidad de tiempo que necesita el programa que busca el modelo de Markov óptimo de acuerdo a l escenario y lecturas se decidió únicamente utilizar el medio ambiente más complejo, se podrían evaluar los resultados de la búsqueda en ambientes distintos, sin embargo para simplificar el trabajo se utilizará únicamente el cuarto ambiente debido a que es el más similar a los escenarios de interés.

Se utiliza el número de estados que proporcionó los mejores resultados de reconocimiento y se prueba con distinto número de centroides, con el fin de asegurar que el algoritmo arroje un modelo $\lambda$ con el número de observaciones deseado.

Se mide el tiempo que tarda el programa en encontrar el modelo y finalizar su ejecución. La variable independiente es el número de centroides utilizado y la variable dependiente es el número de segundos que tardó el programa en arrojar el modelo $\lambda$ que mejor se ajusta a los datos proporcionados.

**¿El algoritmo de Viterbi se puede ejecutar en tiempo real mientras el robot navega?**

**Prueba 5.Tiempo de ejecución del algoritmo de Viterbi**

Se toma en cuenta el mundo 4 con 24 estados y el número de centroides se varía con el fin de asegurar que esta variable no afecte la ejecución del algoritmo en tiempo real, el estudio del tiempo de este programa es el más relevante para el proceso de navegación final. Se cuantifica cuánto tiempo tarda en brindar un estado con diferente número de observaciones, estás van de 16 a 256 observaciones.

### 4.4.3. Clasificación

**¿Existe pérdida de información valiosa al utilizar un método de clasificación?**

**Prueba 6. Clasificacion**

Para realizar esta prueba se utiliza el sensado del robot real. Se ejecuta el cuantizador vectorial con $\varepsilon = 0.01$, una muestra de 1649 lecturas y pidiendo

8, 16, 17, 32, 64, 112, 128, 184 y 256 observaciones. Se realiza la búqueda del modelo $\lambda$ con las observaciones obtenidas. Una vez que se tienen los vectores arrojados por la cuantización se realiza un *script* de 200 comandos que guíe al robot a través de las 24 regiones consideradas en la realización del modelo $\lambda$ y se almacena el valor del sensado en cada paso. Después se procesa la muestra obtenida con el cuantizador y se obtiene una clasificación, se verifica que la observación arrojada por el cuantizador corresponda a alguna de las que representan la región del medio en la que el móvil obtuvo esa muestra. Al finalizar se cuantifica el número de veces que la observacion fue clasificada dentro de la región correcta.

Para realizar la prueba con el algoritmo de la red neuronal se ejecuta la red neuronal, se obtienen las observaciones que se utilizarán para buscar el modelo $\lambda$; se utiliza el mismo *script* que el usado para la cuantización vectorial y las mismas 24 regiones, se obtiene la muestra del valor de los sensores en el trayecto, se preprocesan los datos, se clasifican de acuerdo a la red neuronal y se cuantifica el número de ocasiones que la observación coincide con alguna de las observaciones que representan una región.

### 4.4.4. Caracterización

**¿Es necesaria la caracterización del hardware?**

**Prueba 7. Caracterización**

Se realiza un *script* en el cual se gire al robot un paso hasta completar una vuelta de 360° el paso es un ángulo constante. Después se varía el valor del paso utilizado de forma que se inicia con comandos de rotación de 15°, luego 30°, 45°, 60°, 90°, 180° y 360°, con estas repeticiones se tendrá un total de 57 giros en los cuales se evlúa si el giro realizado por el móvil es igual al ángulo deseado, se tiene una incertidumbre de $\pm 0.34$[rad]. El proceso se repite después de implementar la caracterización y se cuantifica el número de veces que el ángulo es igual al ángulo deseado con la incertidumbre ya mencionada.

### 4.4.5. Uso de HMM

Se realizaron 20 trayectos diferentes compuestos por 5, 10, 20, 30 y 50 instrucciones generando 100 ejecuciones a lo largo de una sección del laboratorio de Biorobótica y se cuantificó el número de ocasiones que el destino no fue alcanzado, esto sucede cuando el elemento $x$ o $y$ calculado en ViRbot tiene una diferencia mayor a 1.5[dm] con respecto a la posición real de $(x, y)$ o una diferencia mayor a 20° (0.34 [rad]) para el caso de $\theta$.

El número de instrucciones manejado actúa como variable independiente y la variable dependiente es la cuantificación del número de veces que la posición real no coincide con la calculada.

Este ejercicio se realiza 3 veces una con el sistema original, otra con la caracterización del robot y otra con el uso de modelos de Markov como auxiliar en el cálculo de la posición para en análisis de cada elemento de la posición y 2 veces en el análisis de navegación global.

**¿Existe algun componente de la posición que genere mayores problemas en la estimación de la posición?**

**Prueba 8. Posición**

La realización de la prueba requiere de trayectos más específicos debido a que se medirá la diferencia entre la posición real $P_r = (x_r, y_r, \theta_r)$ con respecto a la posicion calculada $P_c = (x_c, y_c, \theta_c)$. Se realizaron 100 secuencias de comandos a través del trayecto mostrado en la figura 5.11 y se midió la posición final que el móvil alcanzó al finalizar los movimientos mandados. Se registró esa posición así como la posición a la que se deseaba llegar. Se hace la cuantificación de la diferencia entre $P_r$y $P_c$ y se enumeran las ocasiones en las cuales la posición real es igual a la posición calculada con una incertidumbre de $\pm 1.5$[dm] en el caso de $x$ ó $y$, y de $\pm 0.34$ [rad] para el valor de $\theta$.

Cada secuencia de comandos se repitió 3 veces partiendo de puntos distintos a lo largo de la trayectoria, teniendo un total de 120 repeticiones de las cuales se obtuvieron los porcentajes de estimación de posición.

**¿Existe una mejora en la auto localización al estimar el estado con el algoritmo de Viterbi?**

**Prueba 9. Navegacion Global**

Con 5 repeticiones del experimento anterior también se estableció que, independientemente de las fallas en cada elemento de la posición, se debe cuantificar también qué tan cercana es la posición real del móvil a la posición enviada como destino, para lograrlo se estableció que el móvil alcanza la posición indicada cuando la distancia entre su posición real y la posición deseada es de 2.1[dm] y cuando el ángulo real difiera al deseado en 0.34[rad] como máximo.

# Capítulo 5

# Resultados

*«El final de un problema no había hecho más que dar nacimiento a otro.»*

***Isaac Asimov***

Los promedios mostrados en las gráficas y porcentajes de este capítulo fueron calculados en base a trayectos del agente en el simulador por los medios ambientes mostrados en la figura 4.6 y para los trayectos en el robot real se utilizó el laboratorio de biorobótica (mostrado en la figura 5.1).

Figura 5.1: Medio ambiente real

En el simulador, los resultados son muy buenos cuando las muestras son tomadas sin ruido, se tienen altos porcentajes de reconocimiento del estado

actual y por lo tanto de la posición del móvil. El comportamiento mostrado en el simulador cuando se utiliza ruido se asemeja al comportamiento del robot real en donde la implementación obtiene porcentajes de reconocimiento más bajos que en el simulador pero manteniedo una mejora en el proceso de navegación en comparación con la navegación sin el uso de modelos ocultos.

Son diversos los enfoques que se pueden tomar para el análisis de los resultados, para este trabajo se tomaron en cuenta 4 cosas principales, la capacidad de reconocimiento de patrones, los tiempo de ejecución, el desempeño del proceso de clasificación, los resultados de la caracterización y el funcionamiento integral de los modelos ocultos de Markov.

## 5.1. Integración del software

No existió ningún problema de comunicación entre el *software* realizado y ViRbot, se integró al módulo de sensado el cuantizador vectorial para generar el archivo de texto con las observaciones que va registrando el móvil a cada paso de rotación y/o traslación.

En 50 ejecuciones con 15 comandos por ejecución se verificó que la observación registrada fuera alguna de las observaciones correspondientes a la región, en el 90 % de los comandos se obtuvo una observación adecuada. Posteriormente se corrió el proceso de los modelos ocultos al momento de la navegación del móvil monitoreando y verificando la correcta carga del modelo $\lambda$ encontrado y revisando el resultado de la ejecución del algoritmo de Viterbi. Los estados que se obtienen fueron registrados por trayecto, se realizaron 50 trayectos a lo largo del laboratorio cada trayecto realizó 15 reconocimientos de región, y se obtuvieron buenos resultados en el 92 % de las ejecuciones, por lo tanto el estado arrojado brinda una ubicación adecuada al robot brindando la región en la que se encuentra el móvil al momento de correr el algoritmo de Viterbi, sin embargo aunque el estado es correcto se tienen posiciones estimadas demasiado distantes a la posición actual, con el criterio de determinar una posición aceptable cuando esta dista de la real en menos de 2.1 [dm] y su ángulo es diferente en 0.36[rad] como máximo, al integrar el módulo de poderación de la posición calculada por ViRbot y la obtenida con el uso de HMM la posición estimada se acerca más a la posición real, los porcentajes de error indicados no son generados por problemas de comunicación ni por un mal procesamiento de los resultados de módulos anteriores, son problemas inherentes al planteamiento de la solución (como el número de regiones utilizadas o la cantidad de observaciones contempladas) y a las características de los algoritmos utilizados (como los parámetros y complejidad).

En la fase final se cambio el formato del archivo generado en tiempo real que contiene las lecturas para eliminar funciones inecesarias al momento de la navegación ahorrando tiempo de procesamiento al costo de perder legibilidad en el archivo.

## 5.2. Porcentajes de reconocimiento de región

El patrón que se desea reconocer es el vector de sensado y en base al reconocimiento que se realiza de este se brinda la ubicación del robot que está

asociada al estado actual, esta ubicación puede ser vista como una estructura de datos la cual está etiquetada con un estado, representado como un número entero y podría ser equivalente a la instancia de la estructura, sus atributos serían los elementos $(x, y, \theta)$ de una posición en el plano cartesiano. El estado es la salida del algoritmo de Viterbi y a partir de este se obtienen los valores de los atributos, por ello se decidió realizar el análisis del porcentaje de reconocimiento en base al estado ya que a partir de este se obtienen las variables de interés.

### 5.2.1. En base al número de observaciones

El estudio del número de observaciones buscó determinar si la mejora en el reconocimiento de la región del escenario depende directamente del número de observaciones obtenidas en la cuantización vectorial. Este estudio se realizó en el simulador debido a la facilidad de ejecución y a que no depende del movimiento del móvil directamente, para saber el porcentaje de reconocimiento de un modelo $\lambda$ se ejecutó el programa con distintas secuencias de observaciones conociendo *a priori* el estado en el cual se generaron comparando con el estado que se obtiene según el algoritmo de Viterbi.

Se probaron modelos generados con 8, 12, 14 y 24 estados y 8, 16, 17, 32, 64, 112, 128, 184 y 256 observaciones en los 4 ambientes del simulador mostrados en la figura 4.6.

El primer ambiente es un escenario sencillo en el cual la espectativa es de una fácil navegación, sin embargo nunca se obtiene un alto porcentaje de reconocimiento de región y por lo tanto la retroalimentación de la posición con el robot no es buena, esto es debido a que los patrones extraídos son muy similares, el ambiente es completamente simétrico y esto genera observaciones parecidas en diferentes puntos del escenario.

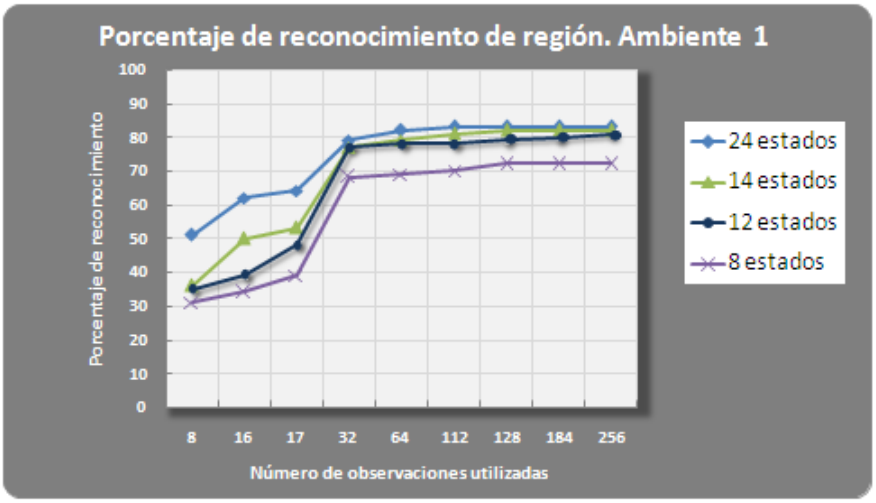



Figura 5.2: Porcentaje de reconocimiento de región con el HMM en lecturas **sin** ruido en función del numero de observaciones utilizado

El reconocimiento máximo obtenido es alrededor del 80 %, en la figura 5.2 se observa que no se obtienen mejoras significativas al aumentar arbitrariamente el número de vectores característicos 32 a 256. El número de estados no es un factor determinante pues al ser un mundo simple se obtienen resultados similares utilizando 24, 14 ó 12 estados.

Como ya se ha mencionado el proporcionar al programa lecturas con ruido tiene el objetivo de mostrar en el simulador resultados muy semejantes a los obtenidos en el móvil real. Graficando el porcentaje de reconocimiento en función de las observaciones utilizadas en el ambiente 1 y lecturas con ruido se tienen menores porcentajes de reconocimiento pero comportamientos similares a los obtenidos con las lecturas libres de ruido como se observa en la figura 5.3.

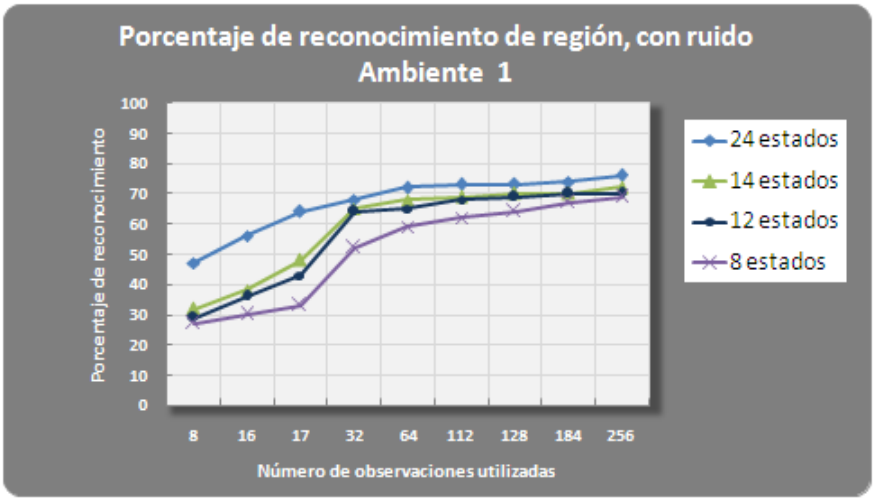



Figura 5.3: Porcentaje de reconocimiento de región con el HMM en lecturas **con** ruido en función del numero de observaciones utilizado

Al analizar ambos casos se determinó que la mejor configuración para este ambiente es utilizar 12 estados y 32 observaciones ya que el aumentar alguna de las dos variables se requiere de una mayor cantidad de procesamiento y no se obtiene una ganancia significativa en el reconocimiento.

Para el ambiente 2 se probó en un ambiente con menor simetría, el punto de equilibrio es alcanzado al usar 14 estados y 64 observaciones obteniendo un porcentaje de reconocimiento del 74 % usando lecturas sin ruido y de 62 % al utilizar lecturas con ruido. En la figura 5.4 se puede ver gráficamente que la elección anterior es la adecuada, además se observa que el modelo con 8 estados no brinda un porcentaje de reconocimiento competitivo.

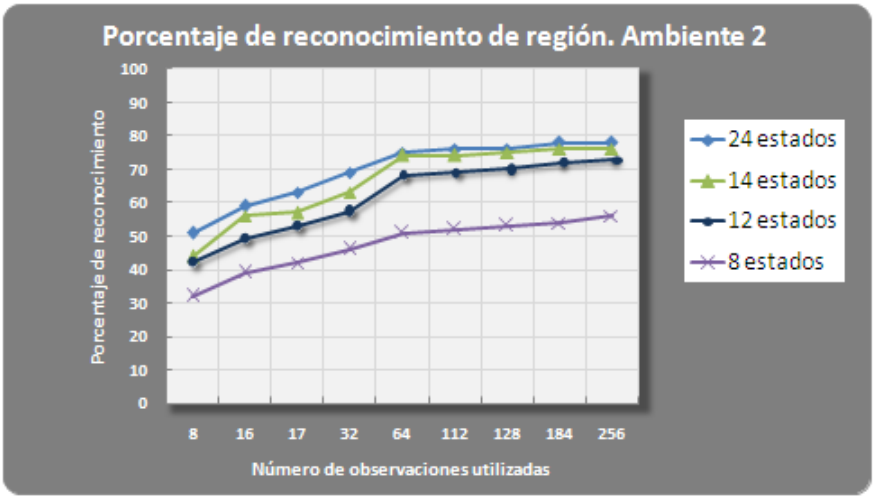



Figura 5.4: Porcentaje de reconocimiento de región con el HMM en lecturas **sin** ruido en función del numero de observaciones utilizado

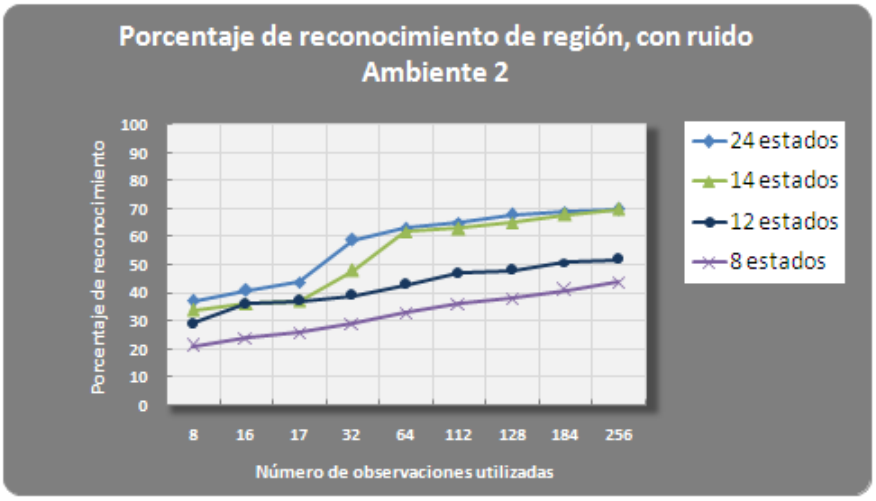



Figura 5.5: Porcentaje de reconocimiento de región con el HMM en lecturas **con** ruido en función del numero de observaciones utilizado

Al utilizar lecturas con ruido en el ambiente 2 se muestra en la figura 5.5 una baja en el porcentaje de reconocimiento pero el comportamiento y el ´punto de equilibrio es similar al obtenido con el uso de las lecturas libres de ruido.

Para el ambiente 3, que ya es la representación de un ambiente real y más complejo, se observaron los mejores resultados (ponderando cantidad de procesamiento y porcentaje de reconocimiento) cuando se utilizaron 24 estados y 128

observaciones. En la figura 5.6 se observa un mayor reconocimiento conforme aumentan los estados utilizados, para esta representación el uso de 8 ó 12 estados no es viable y se requiere un mayor número de observaciones para obtener el equilibrio.

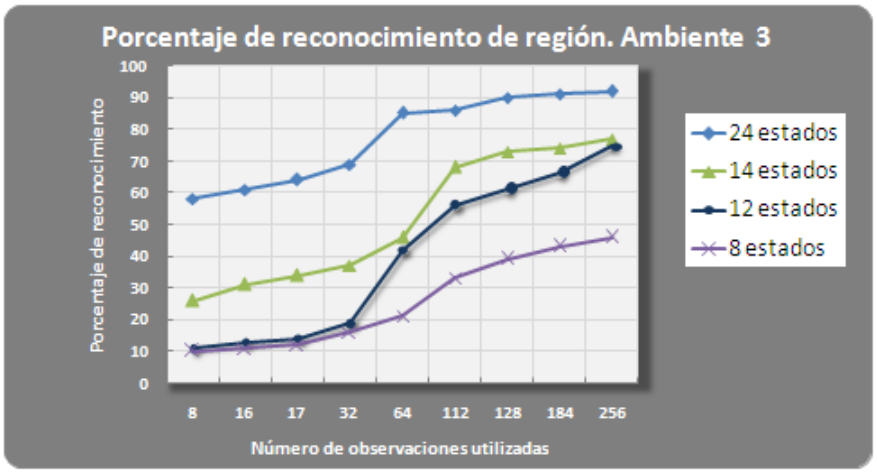



Figura 5.6: Porcentaje de reconocimiento de región con el HMM en lecturas **sin** ruido en función del numero de observaciones utilizado

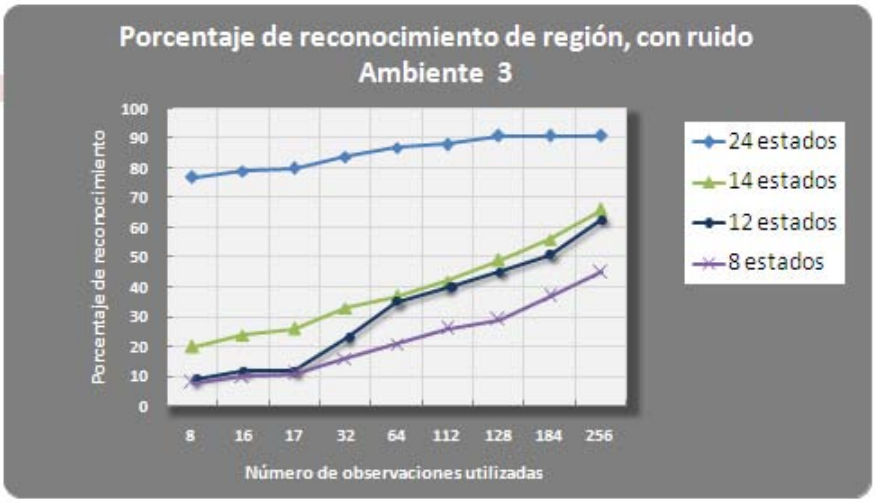



Figura 5.7: Porcentaje de reconocimiento de región con el HMM en lecturas **con** ruido en función del numero de observaciones utilizado

Al utilizar muestras con ruido como se muestra en la figura 5.7, solo el uso de 24 estados da buenos resultados y como esta representación es la más cercana al comportamiento real se tomó este número de estado para realizar el modelo $\lambda$.

Para el ambiente 4, que es la representación del ambiente real en el cual se realizaron las pruebas con el robot, se tuvieron buenos resultados al utilizar 24 estados y 128 observaciones. En la figura 5.8 se observa que el uso de 24 estados es la única curva que brinda resultados aceptables tanto en lecturas con ruido como en muestras libres de éste (figura 5.9).

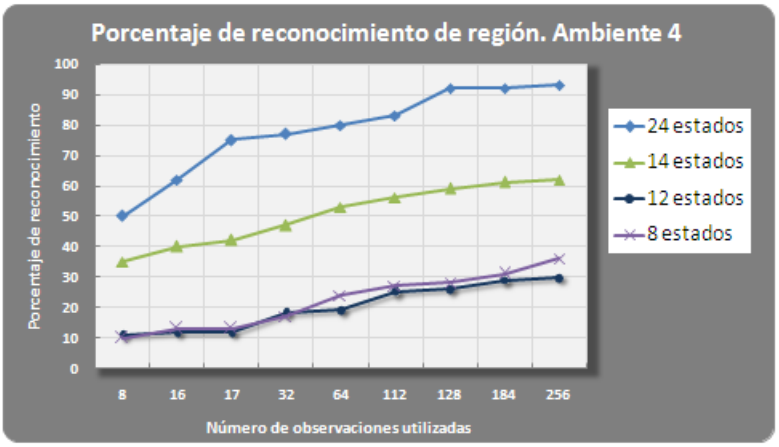



Figura 5.8: Porcentaje de reconocimiento de región con el HMM en lecturas **sin** ruido en función del numero de observaciones utilizado

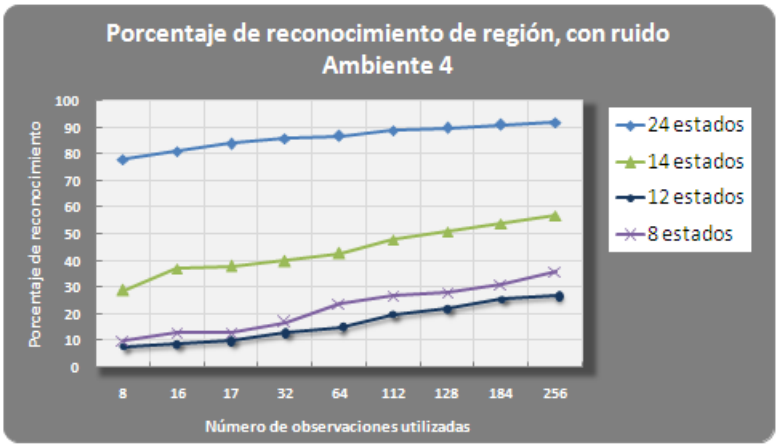



Figura 5.9: Porcentaje de reconocimiento de región con el HMM en lecturas **con** ruido en función del numero de observaciones utilizado

La figura 5.10 muestra el comportamiento real del robot al momento de reconocer las regiones utilizando un número de observaciones diferente, los resultados son similares a los que se describen en la figura 5.9 reconociendo las regiones un 86 % de las ocasiones.

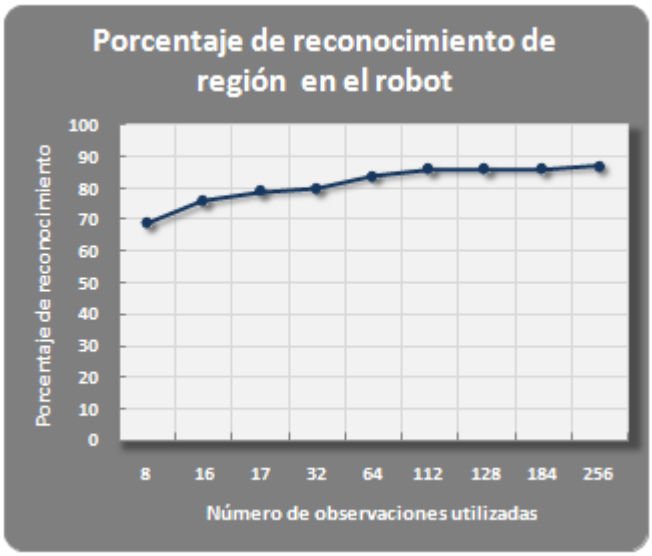



Figura 5.10: Porcentaje de reconocimiento de región con el HMM en lecturas **reales** en función del numero de observaciones utilizado

Con respecto al número de estados se observa un mayor porcentaje de reconocimiento cuando se tienen un mayor número de estados, esta mayoría es observada de mejor manera cuando el ambiente utilizado es complejo. Como los medios utilizados en las pruebas reales son complejos el comportamiento es mejor utilizando un mayor número de estados. Sin embargo es lógico observar un aumento en la probabilidad de error cuando el ambiente utilizado contiene muchos obstáculos. Con el fin de determinar el grado de complejidad de un lugar en el cual navega el robot se analizó el número de patrones semejantes a lo largo de un recorrido y la cantidad de obstáculos que se tienen.

Dentro del simulador y tomando la representación de un ambiente complejo como el mostrado en la figura 5.1 se observa una tendencia de reconocimiento proporcional al número de observaciones generadas en la cuantización vectorial hasta cierto punto, después se cae en un límite a pesar de aumentar el número de observaciones. Es importante este resultado debido a que los espacios en los cuales se debe mover el robot son semejantes a esta distribución y la gráfica de la figura 5.10 nos permite identificar que a partir de un valor elevado de

observaciones ya no se logra mejorar el porcentaje de reconocimiento aumentando el número de vectores característicos, es decir si se utilizan 256 vectores característicos se tiene un reconocimiento de 87 % el cual no proporciona una ganancia muy superior a la obtenida utilizando 128 vectores con los cuales se tiene 86 % de reconocimiento de la región.

Las curvas anteriores, en general muestran una relación proporcional entre la variable dependiente y la independiente, es decir, a mayor número de observaciones se tiene mayor porcentaje de reconocimiento, sin embargo existen caidas y puntos de estabilidad. Las caídas ocurren en el punto en el cual el número de observaciones es muy cercano al número de regiones utilizadas, de forma que se está utilizando un nivel de detalle que no ayuda a la correcta clasificación de las regiones, no se tiene un nivel alto de generalidad (conseguido con pocas observaciones en proporción al número de regiones) ni el nivel suficiente de detalles para diferenciar una región de otra (conseguido al utilizar al menos $2 * |r|$ observaciones[1]).

Por otro lado la estabilidad en el porcentaje de reconocimiento se alcanza cuando el número de observaciones utilizado es demasiado grande comparado con el número de regiones en las cuales se divide el escenario por el cual navegará el móvil, para el caso en estudio esto indica que la complejidad del escenario no requiere un nivel de detalle mayor al conseguido con 128 observaciones.

### 5.2.2. En base al algoritmo de clasificación

Con el fin de comparar dos métodos de generación de los vectores característicos se realizó el proceso con una red *fuzzy* ART y con el método de cuantización vectorial. Los resultados se muestran en la figura 5.11 el funcionamiento de la red neuronal es regular, es decir, sin importar la complejidad del ambiente, la red ofrece alrededor del 90 % de reconocimiento de región. El comportamiento de la cuantización vectorial es menos regular y sí depende del ambiente, ofrece mejores resultados en ambientes complejos.

Las pruebas fueron realizadas en el simulador, por lo que, en una segunda fase del experimento, se le agregó ruido para obtener resultados más cercanos a los obtenidos en una implementación, los resultados se muestran en la figura 5.12, donde la red ART obtiene buenos resultados en mundos sencillos pero el porcentaje de reconocimiento es prácticamente el mismo en el ambiente más complejo que es el de interés debido a que es la representación del mundo real donde se busca probar al robot en diferentes trayectos.

[1] $r$ es el número de regiones utilizadas

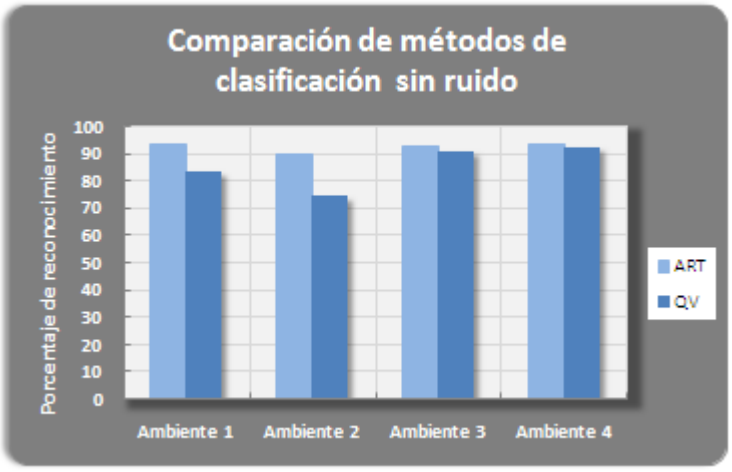



Figura 5.11: Comparación del funcionamiento del HMM de acuerdo al algoritmo de clasificación usado sin ruido

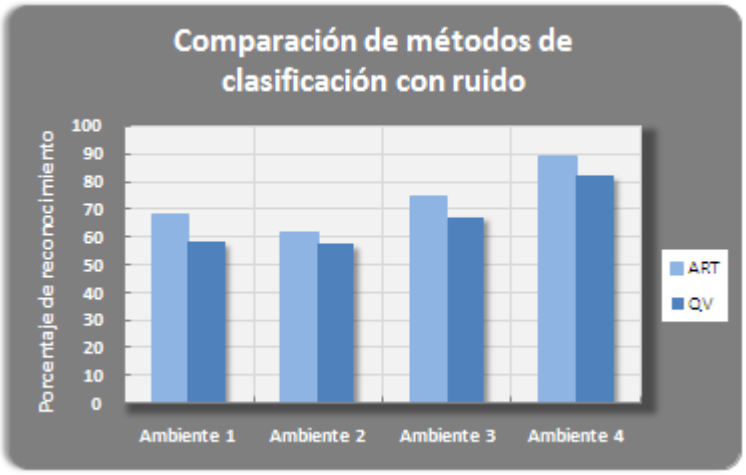



Figura 5.12: Comparación del funcionamiento del HMM de acuerdo al algoritmo de clasificación usado con ruido

## 5.3. Tiempos de ejecución

La evaluación del tiempo de ejecución es relevante debido a que la navegación del robot es un proceso ejecutado en tiempo real, no se tiene un límite

de tiempo para desplazarse de un lugar a otro, pero es importante asegurar que los programas se ejecuten en un tiempo razonable (tanto para las pruebas como para la ejecución final) y que existan recursos disponibles para culminar con éxito la navegación desde el punto origen al destino pedido. Si el tiempo de ejecución no es apropiado es necesario realizar modificaciones en el *software* buscando disminuir la complejidad computacional de los algoritmos simplificando el proceso.

### 5.3.1. Tiempo de ejecución de CV

El algoritmo de clasificación y la búsqueda del modelo oculto de Markov no se ejecutan en tiempo real pero se evaluará su tiempo de ejecución para determinar sus alcances y de qué manera afecta en la ejecución del modelo $\lambda$ en la nevagación del móvil.

El desempeño de la mayoría de los algoritmos se ve afectado si cambian sus parámetros, en particular en el algoritmo de cuantización vectorial es importante el valor de $\varepsilon$ con el cual se generan los nuevos centroides. El primer valor de $\varepsilon$ es 0.0001, si se desean obtener 8, 16 ,32 o 64 centroides se tienen los siguientes resultados:

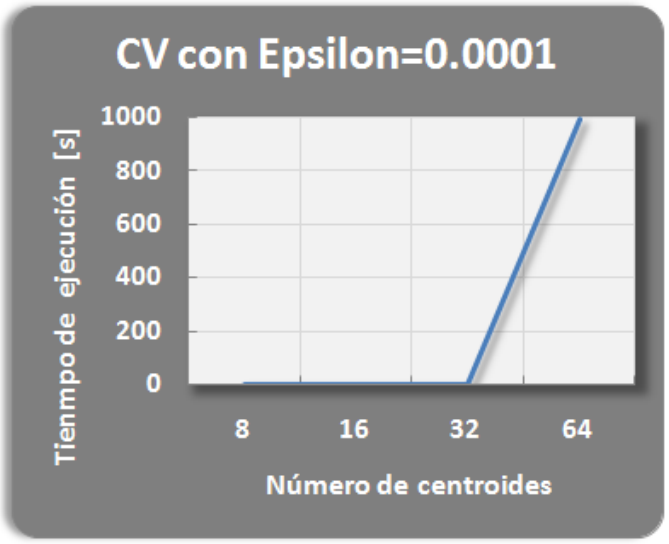



Figura 5.13: Tiempo de ejecución de la cuantización vectorial con $\varepsilon = 0.0001$

El tiempo de ejecución del algoritmo cuando se desean 64 centroides es muy largo, no se tienen un resultado satisfactorio pero si se cambia el valor a $\varepsilon = 0.01$ se obtienen los siguientes tiempos de ejecución:

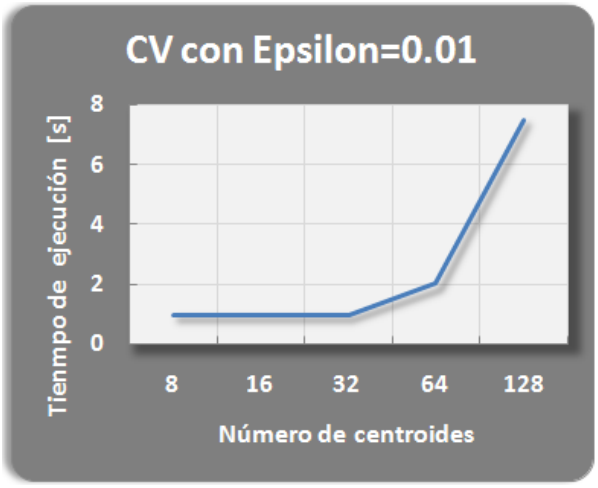



Figura 5.14: Tiempo de ejecución de la cuantización vectorial con $\varepsilon = 0.01$

### 5.3.2. Tiempo de ejecución de búsqueda de $\lambda$

La fase de entrenamiento del modelo se ejecutó únicamente buscando modelos con 24 estados, esto depende del ambiente utilizado de esta forma cada estado corresponde a una región del entorno de navegación de manera directa. Se probaron modelos con 16, 32, 64, 128 y 256 observaciones. Los tiempos de ejecución son los siguientes:

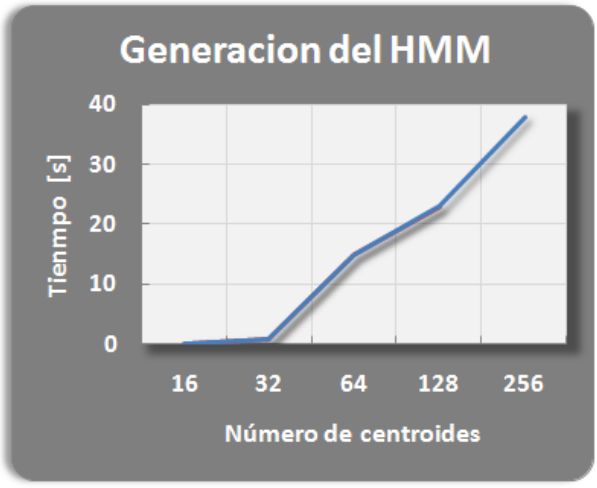



Figura 5.15: Tiempo de ejecución del entrenamiento de **h** utilizando un número de observaciones diferentes $\varepsilon = 0.01$

### 5.3.3. Tiempo de ejecución del algoritmo de Viterbi

El algoritmo de Viterbi es la fase que tiene mejor tiempo de ejecución y es el proceso en el cuál se debe verificar que los tiempos de ejecución sean óptimos debido a que se lleva a cabo en tiempo real por lo que es muy importante que el tiempo de respuesta sea corto.

Al obtener buenos resultados en este proceso califica como factible el uso del modelo obtenido para ayudar al robot a mejorar la percepción que tiene de su posición.

En la figura 5.16 se grafica el tiempo de ejecución del algoritmo cuando se utilizaron 16, 32, 64, 128 y 256 observaciones en las cuales se clasificaron las muestras del sensado a lo largo del trayecto que se desea estimar.

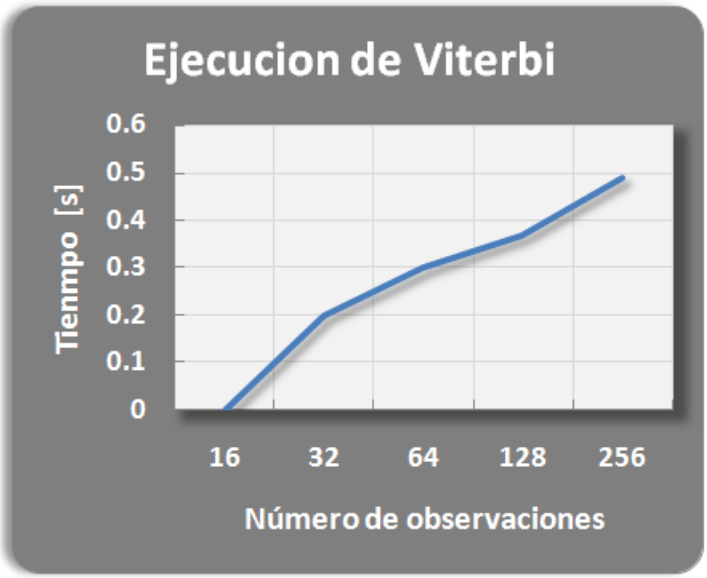



Figura 5.16: Tiempo de ejecución del algoritmo de Viterbi utilizando un número de observaciones diferentes

## 5.4. Clasificación

En la búsqueda de evaluar y validar el uso de la clasificación como un método válido para reducir la cantidad de procesamiento sin perder una cantidad de información significativa se cuantificó el número de veces que coincide la observacion asignada a la muestra con alguna de las observaciones del estado correspondiente de acuerdo a la posición real del robot. En la figura 5.17 se muestran los resultados donde se observa que, en el ambiente más complejo que se utilizó, la clasificación es adecuada cuando se utilizan más de 100 centroides.

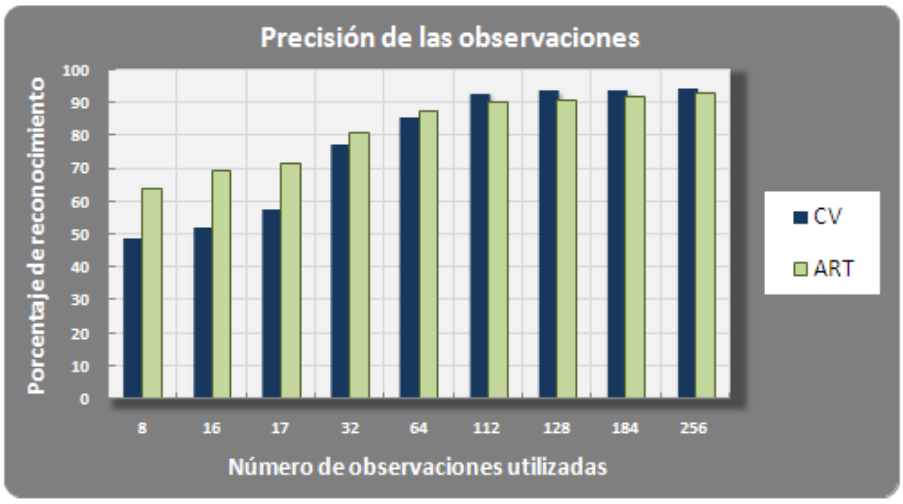



Figura 5.17: Precisión de la clasificación de observaciones

## 5.5. Navegación

Finalmente el proceso de mayor interés es la navegación del móvil, en esta sección se presentan los resultados de la caracterización y del uso del modelo $\lambda$ encontrado.

### 5.5.1. Caracterización

Después de haber realizado la caracterización del móvil, se ejecutaron 20 secuencias de comandos con 10 instrucciones cada una, que dirigen al robot por segmentos del trayecto mostrado en la figura 5.18, ViRbot brinda una posición esperada la cual es muy relevante en el cálculo de los movimientos necesarios para llegar a un destino, por lo cual es necesario determinar si la caracterización realizada mejora la navegación del móvil. Se compara la posición arrojada por ViRbot con la posición real, cuando se utiliza la cuantización existe un parecido mayor entre las posiciones calculadas y las reales que cuando no se utiliza este proceso.

Con el uso de la ecuación propuesta para mejorar la estimación del ángulo real en la sección 4.3 el movimiento del robot es más exacto y se alcanzan los puntos destino en un mayor número de ocasiones.

En la figura 5.19 se muestra la comparación del comportamiento del ángulo antes de la caracterización y después de ésta, y se observa una ligera mejoría, sin embargo al utilizar mas de 20 comandos el ángulo real difiere (en más de 0.34 [rad]) con el calculado en más del 50 % de las ocasiones.

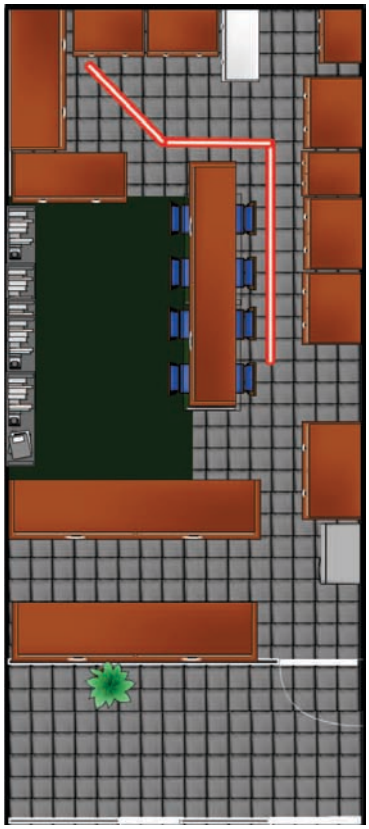


Figura 5.18: Trayectoria utilizada para analizar el porcentaje de reconocimiento de región

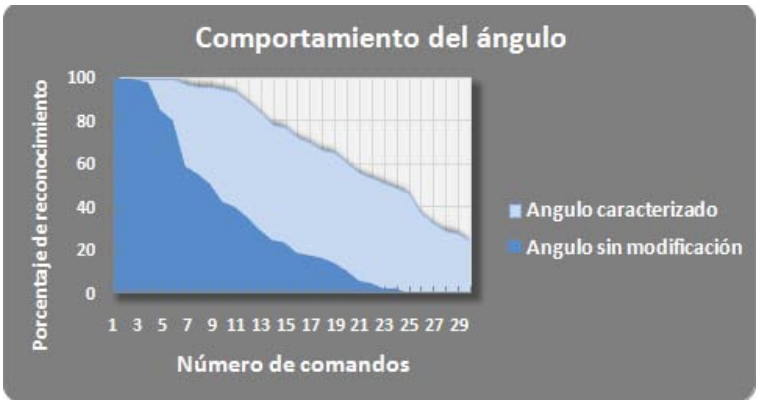



Figura 5.19: Comportamiento del ángulo caracterizado

### 5.5.2. Uso de HMM

**Resultado 8. Posición**

Cada vez que se pide al robot ejecutar el algoritmo de Viterbi y obtener el estado más probable según su observación actual y sus posiciones anteriores lo que se desea saber es la aproximación de la posición actual buscando recalcular el trayecto o valorar si la planeación realizada no requiere modificación, como este valor está compuesto por la tripleta $(x, y, \theta)$ es interesante cuantificar el número de veces que esta aproximación de la posición es cercana al valor real. Para lograrlo se ejecutó un *script* de movimiento que lleva al robot por el ambiente del laboratorio de Biorobótica en una trayectoria máxima de 20 [dm] y un dezplazamiento máximo de alrededor de 16 [dm] a través del segmento mostrado con una línea roja en la figura 5.18.

Repitiendo la ejecución de la trayectoria iniciando de diferentes puntos y tomando únicamente en cuenta la última posición simulada se puede calcular el error entre el valor que brinda el *software* y la posición real del robot.

El porcentaje mostrado en las gráficas 5.20, 5.21 y 5.22 muestra el número de veces que la posición real y la posición dada por el algoritmo de Viterbi tienen una diferencia significativa.[2]

El elemento $x$ calculado por HMM y el valor real concidieron, con la debida incertidumbre, en promedio 72 de 100 veces que se realizó la ejecución, lo cual mejora en 19 % la ejecución de la trayectoria sin el reconocimiento de región con HMM.

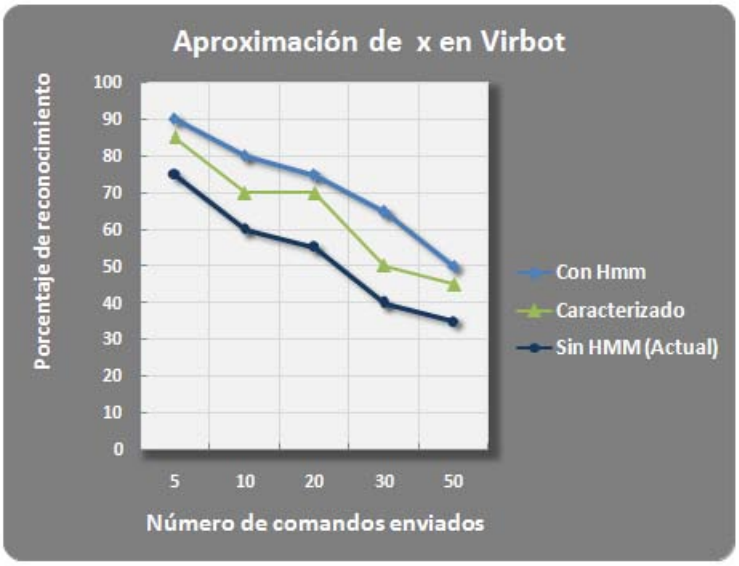



Figura 5.20: Porcentaje de reconocimiento de $x$

[2]Los valores significativos se establecen enla seccion **4.4** del capitulo 4

Para el elemento $y$ se tiene un porcentaje de acierto de 66 %, mejorando en un 16 %.

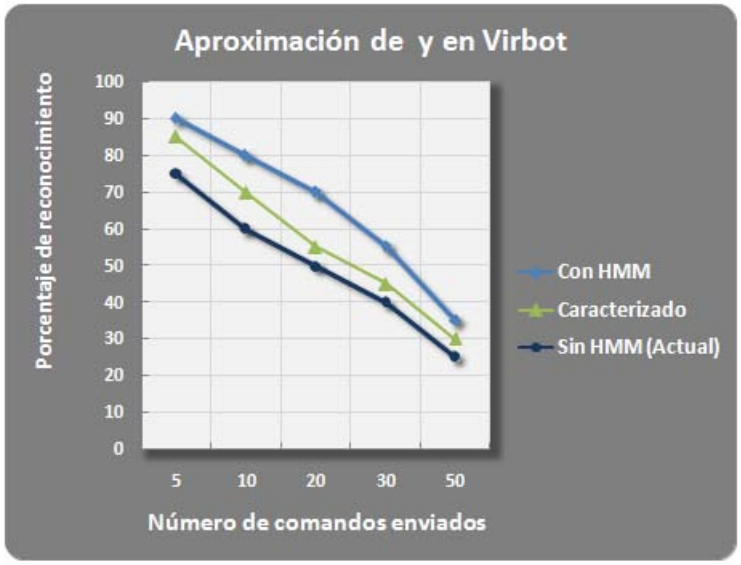



Figura 5.21: Porcentaje de reconocimiento de $x$

Para el elemento $\theta$ se tiene un porcentaje de acierto de 52 %, obteniendo una mejora del 20 %

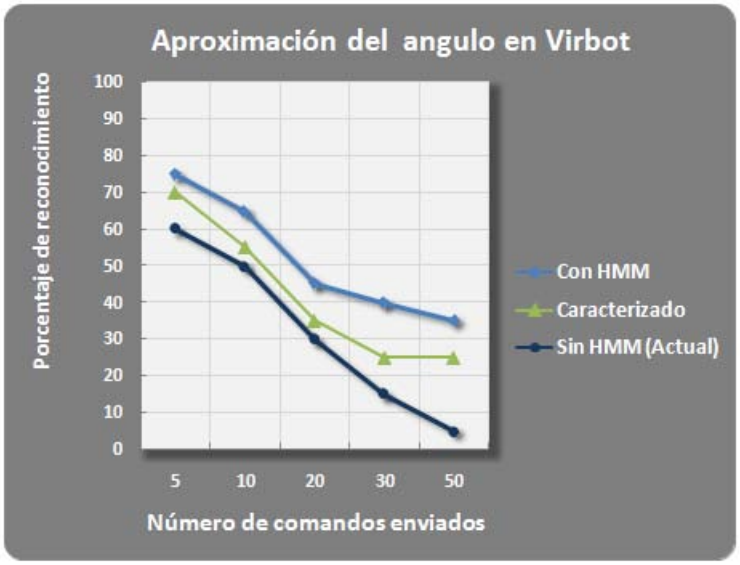



Figura 5.22: Porcentaje de reconocimiento de $x$

**Resultado 9. Navegacion Global**

Para evaluar si el proceso propuesto realmente ayuda a alcanzar un objetivo se realizaron numerosas pruebas con diferentes trayectos, se agruparon los trayectos de acuerdo al número de instrucciones enviadas al robot para llegar a su destino y se cuantificó el número de ocasiones que el móvil llegó a su destino. El cambio en el proceso de navegación es más notorio en los trayectos largos, a partir de 20 comandos. Sin el uso de los HMM en 100 experimentos, el número de veces que el móvil llega a su destino después de un trayecto de 50 comandos es de 34 y cuando utiliza los HMM llega 60 veces a su destino. Se obtiene en promedio una mejora del 14.5 %

En la figura 5.23 se observe la comparación del comportamiento de la navegación con el uso de los HMM y sin el uso de estos, los trayectos fueron de 5, 10, 20, 30 y 50 secuencias de comandos y en todas las trayectorias se tienen mejoras, sin embargo la ganacia es más notoria cuando se probaron trayecctos largos (mayores de 30 instrucciones).

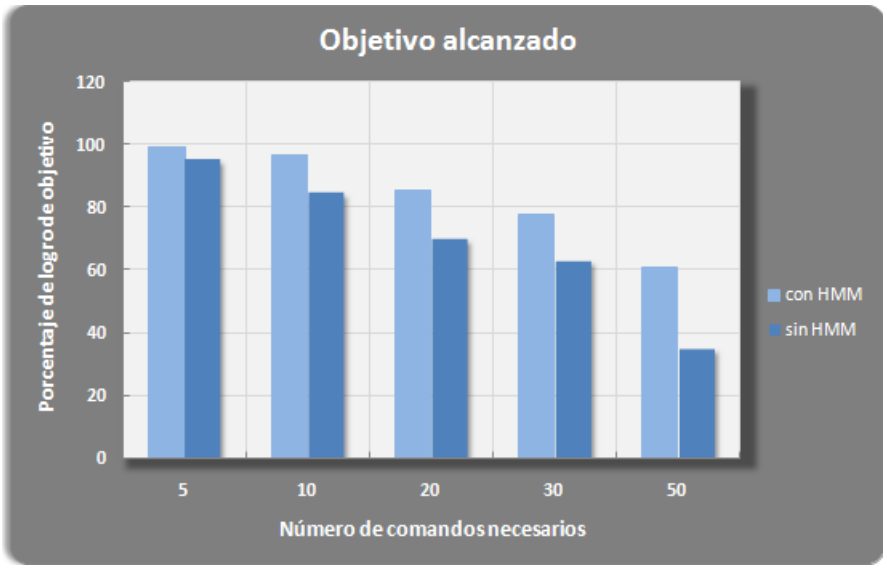



Figura 5.23: Número de veces que el móvil llega a su destino

# Capítulo 6

# Conclusiones

*«Acoplando sus declaraciones, sólo hay una posibilidad a la que podría usted hacer referencia... y ésta es imposible.»*

***Isaac Asimov***

El análisis de los datos y la inmersión en un problema ayudan a identificar de manera más sencilla las fallas en los flujos iniciales y se descubren interesantes relaciones que pueden dar pie a nuevos caminos de estudio.

Al estudiar cuidadosamente los datos recopilados se obtuvo información valiosa acerca del preprocesamiento de datos, tanto de los valores que sirven como muestra para crear los distintos módulos del proyecto (el cuantizador, el modelo \lambda) como de los parámetros de cada uno de estos módulos:

- La naturaleza de las cosas parece ser la redundancia, gracias a esto muchos datos obtenidos a lo largo de los experimentos pueden ser reducidos sin afectar los comportamientos generales facilitando su computabilidad y aunque la mayoría de los fenómenos aun no puede representarse a través de modelos aproximaciones son herramientas muy valiosas que el ser humano puede ajustar a sus necesidades buscando resolver de manera viable los problemas que se le presentan.
- Debe haber un equilibrio entre aquello que se desea representar (en este caso el ambiente) y los recursos utilizados para lograrlo (como el número de estados utilizado) ya que en muchas ocasiones se desperdician herramientas buscando representaciones fieles a un fenómeno que, mejor estudiado, puede tener una descripción adecuada al problema a un menor costo (computacional o económico).

## 6.1. Porcentaje de reconocimiento

El uso de HMM en el reconocimiento de patrones arroja muy buenos resultados en la ubicación del móvil usando el sensado como variable conocida y la posición como variable oculta, los porcentajes de reconocimiento son elevados obteniendo una ubicación hasta del 90 % de los estados en el ambiente del laboratorio de biorobótica.

## 6.2. Tiempo de ejecución

En la fase de búsqueda del modelo $\lambda$ fueron necesarios varios ajustes de los parámetros de los algoritmos buscando disminuir el tiempo de respuesta, debido a que la complejidad computacional de los procesos aumenta de manera polinomial con respecto al número de muestras utilizadas y al tamaño de cada vector de la muestra, con los ajustes realizados los algoritmos arrojan resultados en cuestión de segundos lo cual es favorable para la cantidad de pruebas realizadas y para el trabajo de analizar nuevos ambientes. El tiempo de respuesta del módulo que contiene al algoritmo de Viterbi es lo suficientemente bajo como para utilizarse en tiempo real, sin embargo la navegación en el robot no es muy veloz debido a la velocidad de comunicación con la base móvil y a la constante comunicación con otros módulos de ViRbot.

## 6.3. Clasificación

Se logró comparar dos métodos de clasificación brindando mejores resultados el uso de la red de resonancia adaptativa que el uso de cuantización vectorial. El cuantizador vectorial es un método con un alto grado de complejidad computacional y no brinda mejores resultados que la red neuronal artificial de resonancia adaptativa fuzzyART, sin embargo es un método estandar que no requiere un preprocesamiento de datos lo que lo hace más sencillo de usar de manera repetitiva y sistemática. Las ventajas y desventajas mencionadas se analizaron teoricamente y se visualizaron experimentalmente.

## 6.4. Navegación

Se mejora relativamente el método de navegación, es decir, aumenta el porcentaje de trayectos que alcanzan el destino deseado, sin embargo no es aún un porcentaje significativo en trayectos complejos.

Se lograron los siguientes aspectos:

- Mejorar el método actual de navegación utilizado en los robots móviles autónomos del laboratorio de biorobótica.
- Dar al agente de control un método de localización por regiones.
- Retroalimentar a un robot móvil a través del sensado sin lograr que el comportamiento sea rectivo a un medio distinto para el cual fue entrenado.

El reconocimiento de trayectoria que realiza el robot real es efectivo, sobre todo para ubicar la posición (x, y) en el ambiente de navegación desafortunadamente los resultados no son muy precisos al dar una aproximación de la orientación del robot(ángulo), lo que perjudica el cálculo adecuado de la posición y no se obtienen el resultado deseado en el proceso de navegación, a pesar de mejorar los resultados actuales gracias a la correcta realimentación del sensado.

Con esta implementación se logra minimizar el número de trayectorias en las que el robot se pierde durante el recorrido y se evita el uso de dispositivos

como brújulas que pueden ser alteradas con campos magnéticos ajenos al robot en un ambiente no conocido.

Para poder utilizar HMM en el proceso de navegación se requiere de entrenamiento previo, toma de muestras y determinación de regiones lo cual es un trabajo laborioso por el momento pero que se puede automatizar para obtener mejores resultados en el tiempo de entrenamiento.

El análisis matemático permitió evaluar los resultados obtenidos de manera teórica y cuantificar el grado de similitud entre teoría y práctica. A través del análisis del resultado de la implementación se evaluó si el error manejado es útil para la navegación dando por el momento resultados conservadores ya que la mejora del 30 % no garantiza la llegada del robot al punto destino.

El buen funcionamiento del modelo obtenido depende en gran medida de la calidad de los datos de la muestra.

La implementación de *pathHmm* con ruido es necesaria ya que permite observar desde la simulación los problemas que se deberán resolver en la implantación del *software* en el robot real.

Se comprueba que el medio es correctamente representado por las lecturas del sensor láser y que su preprocesamiento facilita el uso de las muestras en los algoritmos con mayor complejidad computacional.

## 6.5. Uso general de HMM

Se comprobó que el punto clave del correcto reconocimiento de patrones es el proceso markoviano, ya que además del sensado permite contemplar la trayectoria conocida hasta el punto en el cual se realiza el sensado, mejorando la aproximación de posición dada por el proceso de sensado.

Se probó un funcionamiento eficiente de los procesos de Markov en el reconocimiento de patrones.

Se comprobó que el algoritmo de Viterbi al proporcionar la secuencia de estados más probable funciona como punto de referencia para ubicar un robot móvil en un ambiente de navegación.

# Glosario

## -A-

**algoritmo**. Es un método para resolver un problema, debe presentarse como una secuencia ordenada de instrucciones que siempre se ejecutan en un tiempo finito y con una cantidad de esfuerzo también finito.

**ambiente**. Lugar físico a través del cual se desplaza el móvil.

## -C-

**clasificador**. Es un elemento que proporciona una clase etiquetada como salida a partir de un conjunto de características tomadas como entradas

**complejidad computacional**. Es un elemento que proporciona una clase etiquetada como salida a partir de un conjunto de características tomadas como entradas.

**comportamiento**. Serie de acciones que realiza un agente.

**comportamiento reactivo**. Comportamiento que depende de una retroalimentación del ambiente.

**comportamiento pasivo**. Comportamiento predefinido antes de la ejecución sin ningún tipo de retroalimentación

## -E-

**estado**. Representación numérica de una posición en la que se encuentra el agente.

**estado anterior**. Representación numérica de la posición de un agente en el tiempo $t-1$

**estado actual**. Representación numérica de la posición de un agente en el tiempo $t$

## -N-

**navegación**. Traslado de un móvil a través de un ambiente auxiliado por un comportamiento.

## -M-

**modelo tradicional**. Clasificación de las técnicas de navegación en las cuales se tiene una representación del medio ambiente, se planean las acciones y los movimientos del robot.

**modelo reactivo**. Clasificación de las técnicas de navegación en las cuales no es necesaria una representación del medio ambiente y no utiliza planeación de acciones y movimientos. Es adecuado para entornos dinámicos y en los cuales los robots tienen errores de sensado.

## -O-

**observaciones**. Numeros enteros que permiten clasificar un vector a manera de simplificar la información.

## -P-

**posición**. Lugar definido por la ubicación del agente optimamente descrito con una cantidad $x$ y otra $y$ que permiten localizar un punto en un plano predefinido.

**programación dinámica**. Método que para solucionar un problema realiza la siguiente secuencia de pasos:

1. Divide el problema en subproblemas con menor complejidad.
2. Establecer una relación entre las soluciones y la solución principal
3. Resolver el problema planteado de la relación y obtener las soluciones a los subproblemas.
4. Solucionar el problema principal a través de las subsoluciones gracias a la relación establecida.

## -R-

**redes neuronales**. Conjunto de neuronas interconectadas entre si cuya inhibición y exhitación desencadena reacciones determinadas en el organismo.

**redes neuronales artificiales**. Es una serie de neuronas interconectadas entre si que reciben un vector de entrada de datos y ofrecen una salida. Cada conexión modifica los datos que recibe de acuerdo a un valor real o entero que la caracteriza. Las neuronas también modifican los valores de entrada de acuerdo a sus funciones de activación. El concepto de red neuronal artificial se origina de la analogía con el cerebro de los seres humanos y de algunos otros seres vivos.

**representación del ambiente**. Codificación del entorno a través de números

**robot**. Obras de ingeniería, concebidas para producir bienes y servicios a través de una compleja combinación de *hardware* y *software*. El término se usó por primera vez en una novela checa donde la palabra «robota» significa fuerza de trabajo.

**robótica**. Combinación de investigación y aplicación de mecánica, control y programación.

## -S-

**sensor**. Dispositivo de *hardware* que permite obtener información acerca de una característica externa al agente para almacenarla o procesarla.

**sistema**. Conjunto de elementos funcionales interconectados entre si que buscan un objetivo en común. Es un concepto multidisciplinario que ha permitido plasmar ideas en estructuras bien organizadas tanto en áreas de la industria como en los campos de las ciencias puras.

# Apéndices

## A-1. Toma de lecturas

Para obtener los datos de entrenamiento del **HMM** es necesario ejecutar el programa *RobotP* el cual envía los comandos adecuados a la base para mover el robot y obtener las lecturas del láser utilizado, generalmente se encuentra ubicado en la carpeta *biorobo\control\bin* del *home* de usuario de las computadoras del laboratorio de biorobótica. Los parámetros que maneja este programa son los siguientes:

*puerto_com.* Es un número entero referente al puerto de comunicación serial con la base del robot, puede ser USB o serial, sólo se manejan los siguientes números posibles:
0 para */dev/ttyUSB0*
1 para */dev/ttyUSB1*
2 para */dev/ttyUSB2*

*host* Es una dirección IP del host que se va a utilizar como manipulador del robot.

Ejemplo de inicio de ejecución:

```
#./RobotP 0 127.0.0.1
#pwd
/home/savage/biorobo/control/bin
```

*Figura 6.1: Ejecución del programa de toma de muestras y manipulación del robot*

En la carpeta *biorobo* se encuentra un archivo *.sh* que permite la ejecución de la instrucción anterior sin cambiar de ubicación lo cual es más práctico al momento de tomar las lecturas.

```
#./control.sh
#pwd
/home/savage/biorobo
```

*Figura 6.2: Ejecución alterna del programa de toma de muestras y manipulación del robot*

Al iniciar, el programa muestra si existe algún error de comunicación con el puerto deseado para que la comunicación sea correcta el robot debe estar conectado a la computadora en el puerto indicado en los parametros y el sensor láser debe estar encendido, si no existe ningún problema de comunicación se muestra la pantalla siguiente:

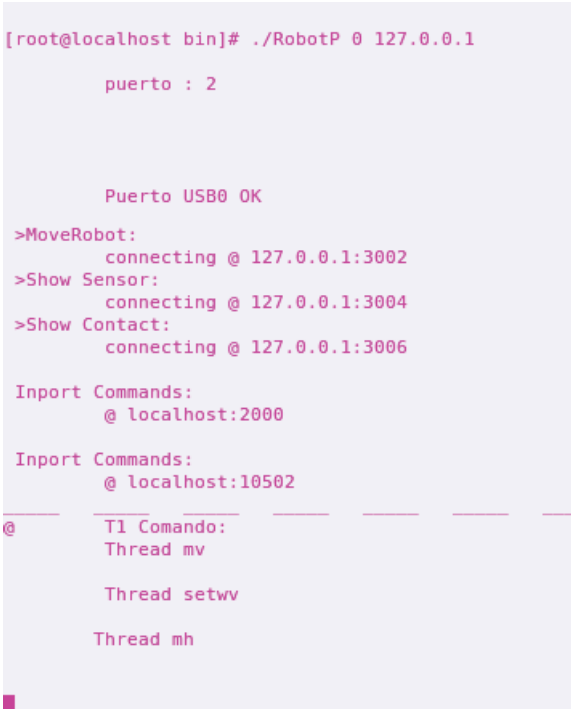



Figura 6.3: Ejecución de RobotP sin errores de comunicación

El programa esta listo para recibir comandos de maipulación del robot, la lista de comandos que recibe se muestra a continuacion:

***getpos*** Situa el robot en una posición inicial diferente a la establecida por default (0,0,0°)

*getpose x y θ*

***x*** número flotante que indica la coordenada en x de la posición inicial.

***y*** número flotante que indica la coordenada en y de la posición inicial.

$\theta$ número flotante que indica el ángulo de la posición inicial.

*>> getpose 10.0 11.5 1.57* ↩

*Figura 6.4: Ejemplo del uso del comando getpose*

***setpos*** Regresa la posición ($x$,$y$,$\theta$) en la que está el robot de acuerdo a los cálculos de odometría y comandos de movimiento.

***make_ map*** Crea un archivo en la ubicación *biorobo\control\files* , el nombre del archivo por default es *laserA_ 1.raw* en el cual se pueden almacenar las

lecturas de los sensores y la posición del robot.

*make_map n*

***n*** número entero que identifica al archivo en el que se almacenarán las lecturas tomadas

$$>> make\_map\ 100 \hookleftarrow$$

*Figura 6.5: Ejemplo del uso del comando make_map*

la ejecución anterior genera un archivo llamado *laserA_100.raw* en la ruta mostrada en la figura 6.6.

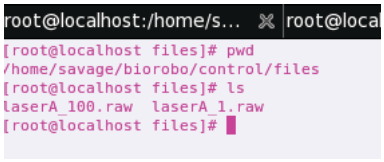



Figura 6.6: Archivo de almacenamiento de los valores sensados

El archivo se crea con el siguiente formato:

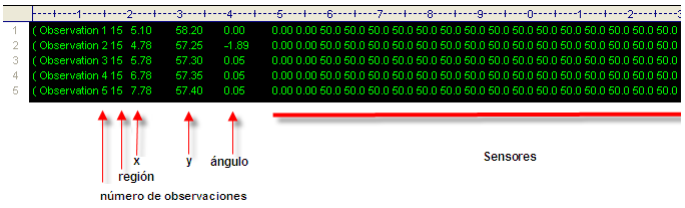



Figura 6.7: Formato de almacenamiento

***closef*** Cierra el archivo creado con el comando make_map

***save*** Almacena los valores de las lecturas enviadas por el sensor láser. Si no existe una ejecución previa de make_map no se almacena ningún valor.

***stop*** Detiene los motores.

***mv*** Permite rotar al robot $\theta$ grados y trasladarlo una distancia ***de d*** [dm]

*mv d θ*

***d*** número flotante que indica cuántos decímetros se desplazará el robot

$\theta$ número flotante que indica cuántos grados rotará el robot antes del desplazamiento

Para enviar estos comandos al programa *RobotP* es necesario establecer una comunicación por sockets, en las computadoras del laboratorio se cuenta con el programa *server* que permite enviar cadenas a un host determinado. La ubicación en general es en la carpeta *sockets* dentro del *home*. Al iniciar pide el *host* al cual enviará los datos escritos, es decir la dirección *ip* en la cual se está ejecutando el programa *RobotP*. En seguida el programa espera un puerto

de comunicación, debe ser el puerto 2000 a menos que esto sea cambiado en el programa *RobotP*. Un ejemplo de esta ejecución se muestra en la figura 6.8.

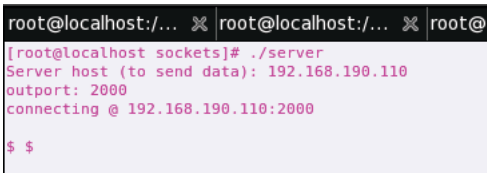



Figura 6.8: Ejecución del programa que permite el envío de comandos

Si no hay ningun problema en la ejecución ambos programas están listos para manipular el robot, en la siguiente figura se muestra una secuencia de comandos que logran que el robot se desplace y guarde los valores sensados en el trayecto, además se muestra la respuesta del programa *RobotP* a cada petición enviada.

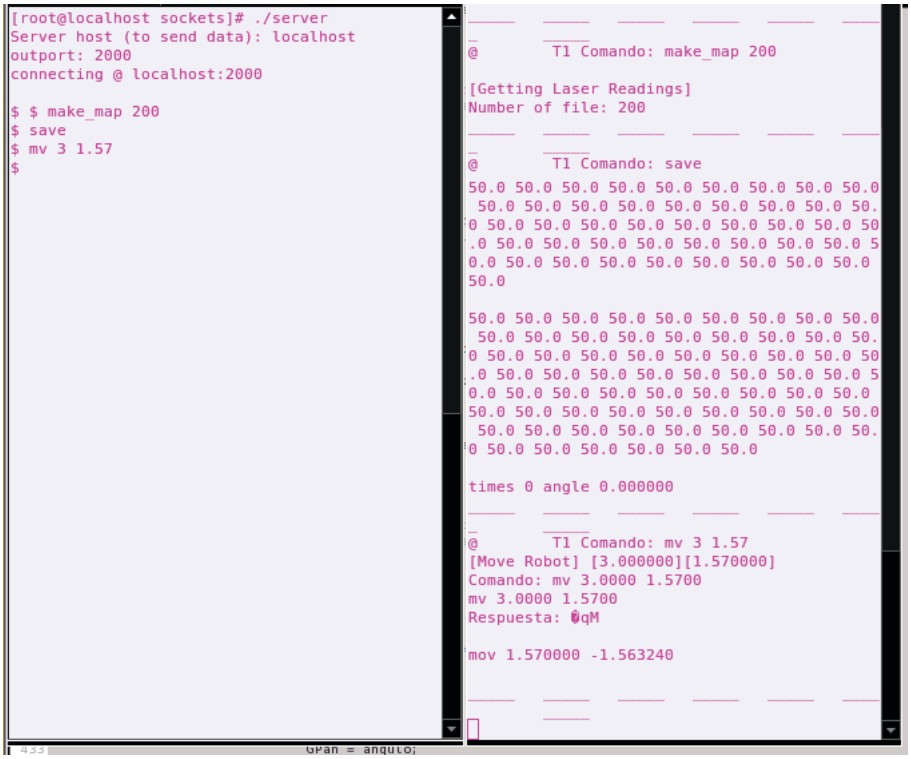



Figura 6.9: Ejecución del programa que permite el envío de comandos

Para la toma de muestras se establece una ruta, se calculan los movimientos que deberá realizar el móvil y se elabora el *script* que envie constantemente los comandos para automatizar la toma de muestras. La manera manual implica el uso de un *joystick* que envía los comandos adecuados de acuerdo a los movimientos que hace el usuario con el dispositivo. El traslado y rotación del

robot se realiza a través del movimietno intuitivo de la palanca del joystick mientras que las lecturas se realizan cuando se presiona el botón 0 (envia comando *save*) y el archivo se cierra con el boton 5 (envia comando *closef*). Para utilizar el *joystick* es necesario ejecutar los programas antes mencionados y además ejecutar el *script* joystick.sh que se encuentra en la carpeta biorobo. Se teclea *make_map* en el server y se utiliza el *joystick* para mover y sensar. Un ejemplo de la ejecución de los 3 programas de manera simultanea se muestra en la figura 6.10.

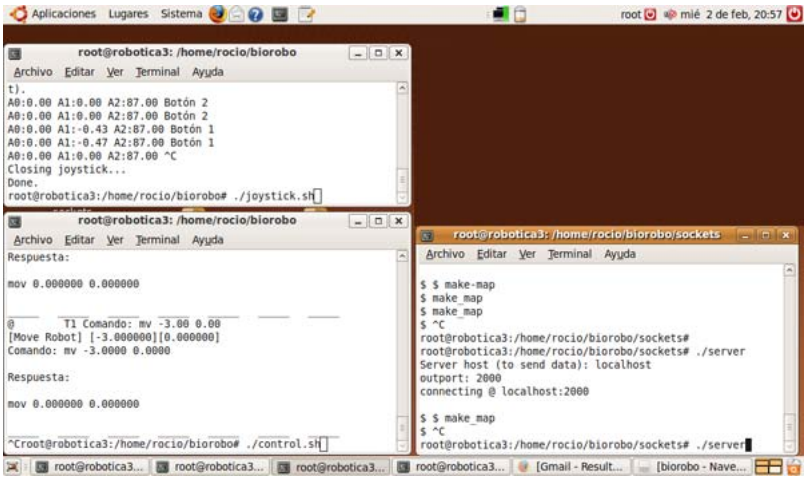



Figura 6.10: Ejecución con el *joystick*

## A-2. Cuantización vectorial

Para obtener una secuencia de centroides de una serie de datos se utiliza el programa vq_vision, para que funcione correctamente con los datos que este problema necesita es necesario dar el valor adecuado de los parámetros establecidos en el programa, los parámetros son los siguientes:

-*m* número entero que indica el número de centroides en los que se agrupan los vectores.

-*e* el número pequeño que se suma a un centroide en el algoritmo de cuantización a fin de obtener 2 centroides diferentes del primero en *e* unidades.

-*f* etiqueta que contiene el archivo base que almacena el identificador de los vectores de agrupamiento.

-*p* ruta en la que se encuentra el archivo base y el que contiene los vectores a agrupar.

-*g* nombre del archivo base.

Por ejemplo si el archivo base es *vision_objectsr* es necesario escribir en el archivo *vision_objectsr.txt* un identificador del archivo fuente que contiene todas las lecturas que se desean agrupar. El archivo fuente tiene el nombre siguiente *images_training_X_1.raw*, donde *X* es el identificador mencionado, ambos archivos deben encontrarse en la dirección indicada en la bandera -*p*.

Si dentro del archivo base *vision_objectsr.txt* se escribe el identificador «*rocio*», el archivo que contiene los vectores a agrupar se buscará con el nombre

*images_training_rocio_1.raw*. En este archivo se tienen los vectores con la estructura mostrada en la figura 6.7.

Mediante el programa de cuantización vectorial ***vq_vision*** se agrupan los vectores del archivo base de observaciones. La línea de ejecución del programa, cuando se desean 128 centroides, es la siguiente:

*# ./vq_vision -f all -m 128 -e 0.0001 -p ../observations/cmucam/training_hsi/ -g vision_objectsr*

Al finalizar la ejecución se obtiene el archivo de salida *vq_images_all_128.dat*, en donde se almacenan los centroides encontrados.

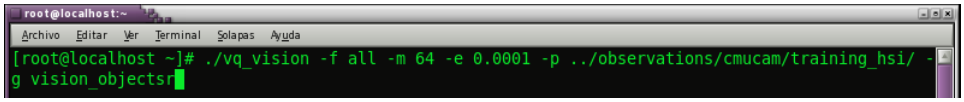



Figura 6.11: Ejemplo de ejecución del programa de cuantización vectorial

## A-3. Generación del modelo $\lambda(A,B,\pi)$

Genera el modelo de Markov

*#./get_hmm_vision -f rocio -n 8 -p /home/savage/observations/cmucam/training_hsi/ -v 32 -q all*

## A-4. Búsqueda del estado actual a través del algoritmo de Viterbi

Búsqueda del estado actual de acuerdo al estado anterior y a la observación actual.

*#./testViterbi -f rocio -n 8 -p /home/savage/observations/cmucam/training_hsi/ -v 32 -q all*

## A-5. Caracterización del robot

Por el momento caracterizar el robot es un proceso muy laborioso y manual, se requiere tiempo y exactitud en las medidas utilizadas como parámetros para obtener ecuaciones correctas y ayudar a mejorar el proceso de navegación

Para obtener buenos resultados se debe realizar un número considerable de repeticiones y tomar los promedios de cada valor, la caracterización esta dividida en dos partes:

Caracterización del ángulo

Caracterización de la traslación

Para la primera fase se deben especificar los ángulos de interés. Por ejemplo 15, 30, 45, 60, 75, 90, etc. y obtener los radianes correspondientes.

Se ejecuta el programa ***characterized*** ubicado en la carpeta *~home\biorobo*, el *software* recibe dos parametros, el *host* y el puerto en donde recibe comandos el programa ./control que manipula la base del robot y permite su movimiento[1]

[1] El manejo del programa *control* se describe en el apéndice A-1

```
# ./characterized ↩
```

*Figura 6.12: Ejecución del programa de caracterización del* hardware *de rotación*

Para la segunda fase se estudia el movimiento del robot, se realizan trayectos de diferente magnitud y se almacena el valor deseado y el valor que el móvil realmente avanzó, en base a esto se determina un modelo de regresión lineal donde la variable independiente es el valor deseado del traslado y la variable dependiente es la magnitud del traslado real.

# Referencias


[1] Baturone, A. O. (2001). *Robótica: manipuladores y robots móviles.* Marcombo, S.A.

[2] Blanco, L. (2009). *Procesos estocásticos I.* [Consulta: 1 de agosto de 2010], http://www.virtual.unal.edu.co/cursos/ingenieria/2001869/index.html.

[3] Carpenter, G. ; Grossberg, S. (1991). *Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system,. Neural Networks (Publication)*, 4:759–771. 32

[4] Carpenter, G. ; Grossberg, S. (2003). *Adaptive Resonance Theory. The Handbook of Brain Theory and Neural Networks*, Segunda Edicion:87–90. 32

[5] Charniak, E. (1993). *Statistical Language Learning.* MIT Press, Cambridge, Massachusetts.

[6] Dean, T. (1998). *Hidden Markov Models.* [Consulta: 10 de diciembre de 2010], http://www.cs.brown.edu/research/ai/dynamics/tutorial/Documents/HiddenMarkovModels.html.

[7] Gabrys, B., Howlett, R. J., ; Jain, L. C. (2006). *Knowledge-based intelligent information and engineering systems*, vol. Part 2. 13

[8] Gowitzke, B. A. ; Milner, M. (2000). *El cuerpo y sus movimientos: bases científicas.* Paidotribo.

[9] Gupta, R. K. ; Senturia, S. D. (1997). *Pull-in Time Dynamics as a Measure of Absolute Pressure.* In *Proc. IEEE International Workshop on Microelectromechanical Systems (MEMS'97)*, pp. 290–294, Nagoya, Japan.

[10] Haykin, S. (1999). *Neural Networks, A comprehensive Foundation*, segunda ed. Prentice Hall. 31

[11] Hidalgo, J. M. G. (2010). *Programación Dinámica.* [Consulta: 12 de agosto de 2010], http://www.esi.uem.es/ jmgomez/alg/04.ProgramacionDinamica.pdf.

[12] Kijima, M. (1997). *Markov Processes for Stochastic Modeling.* Chapman & Hall. 27, 28, 29

[13] Ko, N. Y. ; Simmons, R. (1998). *The lane-curvature method for local obstacle avoidance. Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, 3:1615–1621. 13

[14] Kröse, B., Bunschoten, R., Hagen, S., Terwijn, B., ; Blassis, N. (2004). *Household robots: Look and learn.* In *IEEE Robotics & Automation Magazine*, vol. 4, pp. 45–52. 15

[15] Li, X., Parizeau, M., ; Plamondon, R. (2000). *"Training Hidden Markov Models with Multiple Observations – A Combinatorial Method",. IEEE transactions on Pattern Analysis and Machine Intelligence*, 22(4):371–377. http://vision.gel.ulaval.ca/fr/publications/IdPersonne_17/start_40/PublList.php.

[16] Machuca, F. R. V. (2007). *Procesos estocásticos y cadenas de Markov.* [Consulta: 1 de agosto de 2010], http://www.lcc.uma.es/~villa/mmtc/tema12.pdf.

[17] Maeda, S., Kuno, Y., ; Shirai, Y. (1997). *Active navigation vision based on eigenspace analysis.* In *IEEE Int. Conf. on Intelligent Robots and Systems*, vol. 2, pp. 1018–1023. IEEE Computer Society Press. 15

[18] Matsumoto, Y., Ikeda, K., Inaba, M., ; Inoue, H. (1999). *Visual navigation using omnidirectional view sequence.* In *Intelligent Robots and Systems, 1999. IROS '99. Proceedings. 1999 IEEE/RSJ International Conference*, Kyongju , South Korea. IEEE. 15

[19] Nieto Crisóstomo, O. (2006). *Diseño de un reconocedor de comandos de voz para el DSP TMS320C6711.* Tesis, UNAM. Tutor: Dr. Jose Abel Herrera Camacho. 44

[20] Payá, L., Reinoso, O., Gil, A., Pedrero, J., ; Ballesta, M. (2007). *Appearance-Based Multi-robot Following Routes Using Incremental PCA.* In Apolloni, B., Howlett, R., ; Jain, L., editors, *Knowledge-Based Intelligent Information and Engineering Systems*, vol. 4693 de *Lecture Notes in Computer Science*, pp. 1170–1178. Springer Berlin / Heidelberg. 15

[21] Rabiner, L. R. ; Juang, B. H. (1986). *An introduction to hidden Markov models. IEEE ASSP Magazine*, pp. 4–15. 30

[22] Savage, J. (2006). *A system for the operation of virtual and real mobile robots (ViRbot).* [Consulta: 21 de abril de 2011], http://biorobotics.fip.unam.mx/index.php?option=com_content&view=article. 24

[23] Savage, J., Llarena, A., Carrera, G., Cuellar, S., Esparza, D., Minami, Y., ; Peñuelas, U. (2008). *ViRbot: A SYSTEM FOR THE OPERATION OF MOBILE ROBOTS. UNAM.*

[24] Simmons, R. (1996). *The Curvature-Velocity Method for Local Obstacle Avoidance.* In *International Conference on Robotics and Automation*, School of Computer Science. Carnegie Mellon University. 13

[25] Sniedovich, M. (2011). *Dynamic Programming. Foundations and Principles.* CRC Press.

[26] Sóbol, I. (1976). *Método de Montecarlo.* Mir, Moscú. 27

[27] Tabuse, M. (2010). *Mobile Robot Navigation using SURF features.* In *Applied Computer Science*, Japan. 15

[28] Thrun, S. (1993). *A Probabilistic Online Mapping Algorithmfor Teams of Mobile Robots.* In Mirta, S. K. ; Kaiser, J. F., editors, *Handbook for Digital Signal Processing*, New York, NY, USA. John Wiley & Sons, Inc.

[29] Thrun, S. (2001). *A Probabilistic Online Mapping Algorithm for Teams of Mobile Robots. International Journal of Robotics Research*, 20(5):335–363. 15

[30] Zhou, C., Wei, Y., ; Tan, T. (2003). *Mobile robot self-localization based on global visual appearance features.* In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference*, vol. 1, pp. 1271–1276. IEEE. 15