Universidad Nacional de Luján

Tesis de Grado
Licenciatura en Sistemas de Información

Robótica Móvil

Técnicas, Modelos y Análisis de un Robot Autónomo

Autor: José Luis Todea
Director: Javier Blanqué

Universidad Nacional de Luján
Int. Ruta 5 y 7
6700 Luján, Buenos Aires
República Argentina
2013

Resumen Ejecutivo

El presente trabajo pretende explorar la utilización diferentes técnicas de localización, mapeo, planificación y visión robótica (visión computacional) a fin de lograr a través de la integración de estas técnicas la navegación exitosa de agentes autónomos hacia un objetivo fijado.

Palabras claves: Robótica móvil, control de robots, inteligencia computacional, Localización, Mapeo, Visión Robótica.

Agradecimientos

A mis padres, mis hermanos, amigos y todas aquellas personas que me ayudaron y brindaron su apoyo durante el desarrollo de esta tesis. A mi director de Tesis, Javier Blanqué por su constante apoyo, su dedicación y rápidas respuestas a todas las consultas que se me presentaban. A todos ellos que fueron un sostén fundamental durante todos estos años de cursada y a quienes les estaré siempre agradecido.

Tabla de contenido

1. Introducción

Según la RAE un robot es “una máquina o ingenio electrónico programable, capaz de manipular objetos y realizar operaciones antes reservadas solo a las personas” [RAE01]. Un robot autónomo móvil es aquel capaz de desplazarse en su entorno sin necesidad de contar con asistencia humana permanente, evitando obstáculos y trazando el recorrido óptimo, con el fin de alcanzar un objetivo.

Al momento de entender a lo que nos referimos cuando hablamos de robots autónomos (con inteligencia de navegación) y robots basados en sistemas de tele-comando (sin inteligencia de navegación)

- **Robots autónomos:** Los robots autónomos son aquellos que pueden realizar las tareas deseadas en entornos no estructurados y sin guía humana.

  Existen muchos tipos de robots que tienen diversos grados de autonomía, por ejemplo, aquellos preparados para la exploración espacial deben contener un alto grado de autonomía debido a la ausencia o imposibilidad de obtener asistencia humana y a la agresividad del entorno en el cual deben realizar su trabajo, por otro lado aquellos utilizados para realizar tareas domesticas como ser la limpieza de pisos o cortar el césped no requieren tal grado de autonomía.

- **Robot tele-operados:** Un sistema tele-operado es aquél que permite a un humano manipular a un robot (controlar su movimiento y la fuerza ejercida) desde una ubicación distante. En esta clase de sistema la información del entorno (sensores) es interpretada por los humanos y transmitidas al robot a través de comandos.

Un sistema autónomo debe poseer las siguientes características:

- Movilidad total con respecto a su entorno (Tierra, Agua o Aire)
- Un cierto nivel de autonomía
- Habilidad de percepción, sentido y reacción ante su entorno[ECI01].

Para lograr que un robot móvil sea autónomo debemos responder a 3 preguntas básicas: ¿Dónde estoy?, ¿Hacia dónde voy? y ¿Cómo llego ahí?, estas preguntas que parecen

triviales para un ser humano, en realidad realiza innumerables operaciones de análisis a cada segundo de manera sub-consiente, en el caso de robots, hay que describir este mismo ambiente de una manera formal, debe analizarse de forma detallada, y teniendo en cuenta que un robot es una máquina que, al menos por ahora, no tiene inteligencia de la forma que los humanos tenemos, no sabe que es "el adentro y el afuera", y no poseen nuestro nivel de conexiones neuronales, entonces, desglosar estas preguntas puede significar:

¿Dónde estoy?, Una representación al menos tridimensional y preferentemente n-dimensional detallada del ambiente en el cual se mueve y en el cual reside actualmente, que no solo implique localización topológica y morfológica, sino tipo de objetos y obstáculos, factibilidad de uso o para por ejemplo usar de apoyo, dureza y solidez de los mismos, su posible movimiento, dirección y trayectoria, de acuerdo a la mejor interpretación que puedan brindar los sensores y la base de datos o base de conocimiento (memoria) y un conjunto de reglas de deducción lógica.

¿Hacia dónde voy?, Una representación al menos tridimensional y preferentemente n-dimensional detallada del ambiente al cual el robot quiere llegar o un conjunto de condiciones que debe cumplir para modificar el ambiente inicial. Esto implica de cierta forma que el robot debe "imaginar" el ambiente futuro y saber diferenciarlo de la situación actual.

¿Cómo llego ahí?, El robot tiene que tener un mecanismo basado en un conjunto de reglas y procedimientos que se apliquen a sus efectores (ruedas, motores, brazos, etc.) que le permitan diseñar un plan razonable (sino el de menor costo) para llegar de la situación actual a la deseada, "imaginarlo", es decir, probarlo en un ambiente simulado en su propio "cerebro", y si el plan es realizable, efectuarlo, evitando los problemas, restricciones y obstáculos.

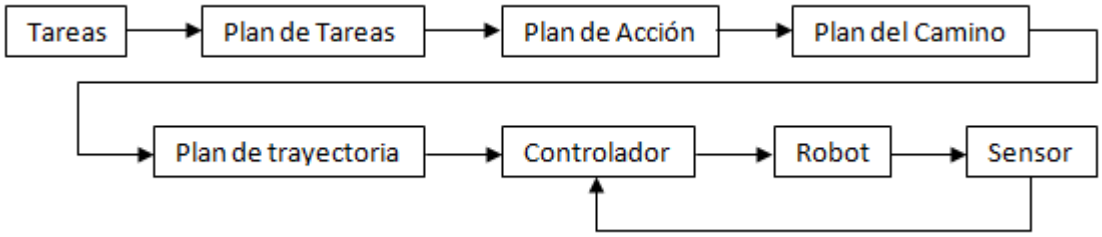
Existen diversas técnicas para lograr esto, algunas de ellas serán analizadas a lo largo de esta tesis.

El primer problema que se nos presenta es el de localizar al robot, es decir, debemos saber de la manera más exacta posible, dónde se encuentra el robot, para así poder decidir el trazado del recorrido. Para esto debemos tener en cuenta los siguientes aspectos:

- La posición relativa del robot con respecto a su punto inicial
- Las coordenadas de la posición del robot en un mapa conocido (Espacio N-dimensional)
- La velocidad, la aceleración y la dirección en la que el robot se desplaza.

El problema consiste en determinar estos aspectos, pero no es posible realizar estas mediciones con un solo sensor, debido a que estos artefactos no son exactos, sino que son afectados por el ruido externo, por lo tanto las mediciones deben realizarse con un conjunto de sensores y combinarlos a fin de obtener la posición correcta.

Además de la localización, el siguiente problema a tener en cuenta es la planificación del camino y la generación de la trayectoria que el robot tal como podemos observar en la imagen 1.



[Imagen 1]

Básicamente se debe tener en cuenta:

- Generar un camino libre de colisiones para el robot móvil teniendo en cuenta obstáculos estáticos (como una forma simplificada) y dinámicos para lograr generar el camino a seguir en un entorno real.

- Entrada:
    - Geometría del robot móvil y de los obstáculos
    - Como se mueve el robot
    - La cinemática del robot

- Salida
    - Secuencia continua de la configuración que debe seguir el robot desde el punto inicial al final.

Es importante saber que cuanto mejor es el mecanismo de uso de sensores para representar un ambiente, menor es la inteligencia necesaria para "imaginarlo" por ejemplo, si el robot tiene un GPS que puede decirle la localización, dirección, velocidad y sentido, no necesitará realizar "mediciones" de objetos externos o utilizar instrumentos menos precisos como giróscopos de efecto Doppler o inerciales. A la inversa, cuanto más complejo y cambiante es el ambiente, o menos desarrollados son los sensores, mayor capacidad de feedback e inteligencia deberá tener el robot para navegar por un ambiente determinado.

Otros de los problemas a tener en cuenta es el mapeo, el cual nos permite llevar a cabo de manera correcta, la localización, definir los caminos y realizar las tareas de planificación. El mapeo involucra simultáneamente la estimación de la posición (orientación) del robot y del mapa, es decir involucra al problema de localización y mapeo al mismo tiempo.

Con el fin de resolver este problema, debemos responder a una serie de preguntas:

- **Sensores**
    - ¿Cómo extraer información relevante de los sensores disponibles?
    - ¿Cómo integrar la información disponible?
- **Localización**
    - ¿Cómo identificar la posición?
    - ¿Cómo identificar los lugares ya visitados?

Para responder a estas cuestiones debemos representar (modelar) el entorno del robot, para lo cual debemos tener en cuenta:

- **Representación del entorno**
    - Modelos Continuos
    - Modelos Discretos
    - Topología del entorno

- **Modelado del entorno**
    - Datos de los sensores
    - Características geométricas del entorno desde la perspectiva tridimensional del Robot
    - Características del entorno
        - Bajo nivel: Características geométricas del entorno
        - Alto nivel: Puertas, autos, edificios, etcétera.

Utilizando la información que conocemos sobre la visión biológica, podemos emular este comportamiento a través de sensores, de esta forma podemos obtener información del entorno en el cual el robot de mueve.

Finalmente, las tendencias de la robótica móvil están apuntando a las nuevas tecnologías de visión robótica (computacional), que tienen como propósito programar una computadora (robot) para que "entienda" una escena o las características de una imagen / escena [WIKI01].

2. Fundamentos de la robótica móvil

2.1. Historia

Durante años el ser humano ha construido maquinas que emulan el comportamiento del cuerpo humano, desde el tiempo de los antiguos romanos, cuando se utilizaban mecanismos hidráulicos adheridos a las estatuas de sus dioses para entretenimiento de sus adoradores en los templos, siguiendo por el gran avance que se realizo en la Europa medieval (ver Tabla01).

Si bien en la actualidad la robótica se utiliza con fines más productivos (industria, Rescates en lugares extremos, Exploración del espacio) debido a la viabilidad económica de utilizar estos mecanismos en tareas repetitivas y de alta complejidad, la emulación de los mecanismos de la naturaleza sigue siendo claves en la robótica actual.

En la actualidad, la robótica se diversifica en varios modelos, cada uno con objetivos específicos, como es el caso del IT, diseñado para expresar emociones, el COG, también conocido como el robot de cuatro sentidos, el famoso SOUJOURNER o el LUNAR ROVER, vehículo de turismo con control remotos, y otros mucho más específicos como el CYPHER, un helicóptero robot de uso militar, el guardia de tráfico japonés ANZEN TARO o los robots mascotas de Sony.

[Imagen 2]

[Imagen 3]

Uno de los mayores ejemplos del actual avance de robótica en la actualidad es ASIMO (acrónimo de "Advanced Step in Innovative Mobility") diseñado por la empresa Honda.

ASIMO es un robot humanoide creado en el año 2000, y ya lleva varias actualizaciones (en 2004, 2005, 2009 y 2011), es capaz de andar, correr, subir y bajar escaleras, girarse suavemente e imitar muchos otros movimientos humanos.

A diferencia de sus predecesores, ASIMO fue el primero en incorporar predicción del movimiento, lo que permite una mayor flexibilidad articular y una más suave, además de un movimiento más humano al caminar.

En general la historia de la robótica la podemos clasificar en cinco generaciones: las dos primeras, ya alcanzadas en los ochenta, incluían la gestión de tareas repetitivas con autonomía muy limitada. La tercera generación incluiría visión artificial, en lo cual se ha avanzado mucho en los ochenta y noventa. La cuarta incluye movilidad avanzada en exteriores e interiores y la quinta entraría en el dominio de la inteligencia artificial en lo cual se está trabajando actualmente [WEB01].

Los robots han sido clasificados, de acuerdo a su generación, a su nivel de inteligencia, a su nivel de control, y a su nivel de lenguaje de programación:

a. **Robots Play-back:** los cuales regeneran una secuencia de instrucciones grabadas, como un robot utilizado en recubrimiento por spray o soldadura por arco.

b. **Robots controlados por sensores:** estos tienen un control en lazo cerrado de movimientos manipulados, y toman decisiones basadas en datos obtenidos por sensores.

c. **Robots controlados por visión:** donde los robots pueden manipular un objeto al utilizar información desde un sistema de visión.

d. **Robots controlados con retroalimentación:** donde los robots pueden automáticamente reprogramar sus acciones sobre la base de los datos obtenidos por los sensores.

e. **Robots con inteligencia artificial:** donde los robots utilizan técnicas de inteligencia artificial para tomar sus propias decisiones y resolver problemas.

f. **Los robots médicos**: son prótesis para disminuidos físicos que se adaptan al cuerpo y están dotadas de potentes sistemas de mando. Con ellos se logra igualar al cuerpo con precisión los movimientos y funciones de los órganos o extremidades que suplen.

g. **Los androides:** son robots que se parecen y actúan como seres humanos.

h. **Cyborg:** Organismos avanzados que combinan partes biológicas y cibernéticas (artificiales), semejantes a los androides.

i. **Los robots móviles:** Están provistos de patas, ruedas u orugas que los capacitan para desplazarse de acuerdo su programación. Elaboran la información que reciben a través de sus propios sistemas de sensores y se emplean en determinado tipo de instalaciones

industriales, sobre todo para el transporte de mercancías en cadenas de producción y almacenes. También se utilizan robots de este tipo para la investigación en lugares de difícil acceso o muy distantes, como es el caso de la exploración espacial y las investigaciones o rescates submarinos[WEB02].

En estos últimos centraremos el análisis de esta tesis, repasando todos los mecanismos necesarios para convertir a esta clase de robots, en mecanismos autónomos que puedan realizar las tareas para las cuales fueron diseñados, si bien esta tesis no se centra en un tipo particular de robot móvil, sino que se trata de un estudio general de las distintas metodologías involucradas en su desarrollo.

En la siguiente tabla se enumeran los principales hitos de la evolución de los robots a lo largo de la historia de la humanidad.

Tabla 1[Tabla 1]

Año	**Avances Tecnológicos**
270 AC	El físico e inventor griego Ctesibius fabrica órganos y relojes de agua con partes móviles.
1495	El anthrobot, un hombre mecánico, es diseñado por Leonardo da Vinci.
1540	Una figura capaz de tocar la mandolina es creada por el inventor Gianello Torriano.
1772	Pierre y Henri Jacquet-Droz crean un niño robótico (el escritor), el cual era capaz de escribir mensajes de hasta 40 caracteres. En la misma época, un humanoide capaz de tocar el piano era construido.
1801	Joseph Jacquard inventa una maquina textil programable, la cual utilizaba tarjetas perforadas
1830	Christopher Spencer diseña un torno accionado por levas.
1890's	Nikola Tesla desarrolla los primeros vehículos a control remoto.
1892	Seward Babbitt diseña una grúa motorizada con pinza para extraer lingotes de un horno.
1938	Willard Pollard y Harold Roselund diseñan un mecanismo programable de aplicación de pintura para la compañía de DeVilbiss.
1948	Norbert Wiener, un profesor del M.I.T. publica el libro cibernética, en el cual se describe el concepto de las comunicaciones y de control en los sistemas electrónicos, mecánicos y biológicos.
	Nacimiento del Robot industrial (1950 -1979)
1951	Un brazo articulado teleoperado es diseñado para ser utilizado por la comisión de energía atómica.
1954	El primer robot programable es diseñado por George Devol.
1959	Planet Corporation comienza a vender el primer robot comercial.
1961	El primer robot industrial se pone en funcionamiento en la plata de New Jersey de GM, realizaba soldadura por puntos y extracción de piezas fundidas.
1963	Fue diseñado el primer brazo robótico que era controlado por una computadora, fue diseñado como una herramienta para las personas con discapacidad y sus seis articulaciones le dieron la flexibilidad de un brazo humano.
1965	DENDRAL fue el primer sistema experto creado para acumular el conocimiento de los expertos en un

	determinado campo.
1968	El brazo de pulpo (tentáculo) artificial fue desarrollado por Marvin Minsky.
1969	El brazo robótico creado por la Universidad Stanford fue el primero en utilizar energía eléctrica y ser controlado por una computadora.
1970	Shakey fue presentado como el primer robot móvil controlado por inteligencia artificial. SRI International en California produjo esta pequeña caja con ruedas que utilizaba su memoria para resolver los problemas y navegar.
1970/1973	Lunojod 1 y 2 fueron dos astromóviles soviéticos no tripulados que alunizaron. Estuvieron en funcionamiento junto con la serie de misiones de sobrevuelo Zond. El objetivo principal de las misiones era explorar la superficie y enviar imágenes.
1974	Se completa el diseño de un brazo robótico (el Brazo de Plata) que realizaba ensamblado piezas pequeñas utilizando la retroalimentación de tacto y sensores de presión.
1976	Brazos robóticos se utilizan en las sondas espaciales Viking 1 y 2.
1979	El carrito de Stanford cruza una habitación llena de sillas sin ayuda humana. El robot estaba equipado con una cámara de televisión que tomaba imágenes y las retransmitía a una computadora de manera que podían ser analizadas.
Despegue de la era de la robótica (1980 - Presente)	
1983	El vehículo de reconocimiento remoto se convirtió en el primer vehículo para entrar en el sótano inundado de la isla Three Mile Island para examinar y limpiar.
1985	Rex fue la primera máquina de excavación autónoma. Detecta y planea donde excavar sin dañar las tuberías de gas enterradas.
1986	El vehículo de trabajo a distancia fue desarrollado para una amplia agenda de las operaciones de limpieza, como lavar las superficies contaminadas, la eliminación de los sedimentos, la demolición de las estructuras radiadas, la aplicación de tratamientos de superficie, y el embalaje y transporte de materiales.
1990	El Ambler fue el primer robot móvil desarrollado como un banco de pruebas para la investigación de robots que operan en terreno accidentado.
1997	PathFinder de NASA aterriza en Marte y el Rover Sojourner comienza la captura de imágenes.
2000	Los humanoides ASIMO (Honda), Dream Robots SDR (SONY), y el perro robot Aibo salen a la luz.
2004	El humanoide, Robosapien es creado por EE.UU. el físico y experto en robótica, el Dr. Mark W Tilden.
2004/2010	Spirit Rover, MER-A (Mars Exploration Rover – A), es un robot que exploró la superficie de Marte, estuvo activo desde 2004 al 2010. Fue uno de los 2 robots enviados por la nasa en la Misión de exploración de la superficie Marte.
2004/2012	Opportunity Rover, MER-B (Mars Exploration Rover – B), es un robot que exploró la superficie de Marte, llego a Marte 3 semanas después que MER-A y aun continua activo.
2012	El Mars Science Laboratory (abreviada MSL), conocido como Curiosity, es un robot programado en para realizar exploración en la superficie de Marte fue lanzado el 26 de noviembre de 2011 a las 10:02 am EST, y aterrizó en Marte exitosamente en el cráter Gale el 6 de agosto de 2012.

[Tabla 1]

En la actualidad los robots son utilizados en una gran diversidad de aplicaciones, desde robots en los salones de clases, robots soldadores en la industria automotriz, hasta brazos tele-operados en el transbordador espacial, algunas de las más notables aplicaciones son:

- Industria (Transferencia de material, Carga y descarga de maquinas, Fabricación automotriz)
- Operaciones de procesamiento
- Laboratorios
- Manipuladores cinemáticos (Industria energía nuclear)
- Agricultura
- Trabajos en el espacio
- Rescates en lugares de difícil acceso
- Educación

Tanto la utilidad como la flexibilidad de un robot están determinadas por la potencia del software, la capacidad de los sensores y limitado por el diseño mecánico utilizado.

Con la llegada de tecnologías de planificación y razonamiento automático, la investigación y diseño de robots móviles (cada uno con características muy diferentes a los demás) creció de manera exponencial.

Los robots móviles brindan la posibilidad de navegar en distintos terrenos y tienen aplicaciones como: exploración minera, exploración planetaria, misiones de búsqueda y rescate de personas, limpieza de desechos peligrosos, automatización de procesos, vigilancia, reconocimiento de terreno, y también son utilizados como plataformas móviles que incorporan un brazo manipulador.

La robótica es una tecnología del presente y con un gran futuro. Si se continúa con las tendencias actuales, y los numerosos estudios de investigación que actualmente se llevan a cabo en laboratorios alrededor del mundo, pueden convertir una tecnología factible en la actualidad en un modelo tecnológico sin límites, los robots del futuro serán unidades con la misma potencia de procesamiento de datos y de cálculo que las grandes computadoras actuales pero con un tamaño completamente reducido. Serán capaces de interactuar de manera más natural con los seres humanos. Podrán ver, oír, palpar, aplicar una fuerza media con precisión a un objeto y desplazarse por sus propios medios.

El paso del presente al futuro exigirá mucho trabajo de ingeniería mecánica, ingeniería electrónica, ingeniería informática, ingeniería industrial, tecnología de materiales, ingenieras de sistemas de fabricación y ciencias sociales. La robótica es una tecnología que no solo puede destinarse al beneficio de la humanidad, y es precisamente por eso que debemos ser cuidadosos y con los argumentos hasta aquí presentados motivar el análisis de este tipo de robots y al mismo tiempo su impacto en la humanidad.

2.2. Robótica móvil

El área de la robótica móvil ha tenido un importante avance en el desarrollo en los últimos años, ya que ha sido el sistema preferido de los investigadores en áreas tales como la inteligencia artificial, el control inteligente y la instrumentación, debido a su versatilidad y posibilidad de manipulación. Si bien se han realizados muchos progresos en el desarrollo de los robots, estos están muy lejos de estar a la par del desarrollo del ser humano a nivel intelectual, sin embargo en la actualidad ya sustituyen humanos en muchos casos en manufactura, logística y servicios industriales, y día a día se está logrado construir robots con cada vez mayor movilidad y autonomía que podrán ir reemplazando al humano en tareas más complejas y especificas.

- **Movilidad:** La movilidad de un robot se mide según los grados de libertad en el que este es capaz de movilizarse en su entorno. En general, los robots más utilizados en la industria, consisten en un brazo móvil, conectado a un punto fijo en el suelo, utilizados para ensamblar o construir diferentes elementos. El problema con esta clase de robots, es que al estar fijos, no pueden realizar tareas tales como limpiar habitaciones, distribuir correo, etc, por lo cual se debe llegar a un estado más avanzado de movilidad.
- **Autonomía:** La autonomía de un robot depende del nivel de confianza que el robot tiene con respecto a su conocimiento previo, y la información de su ambiente para lograr las tareas que se requieren de él. La autonomía de un robot se puede dividir en 3 clases:
    - **Sin autonomía:** Robots controlados completamente y de manera remota por humanos.
    - **Semi-autónomos:** Se trata de robots que pueden navegar por sí mismos pero al mismo tiempo reciben control remoto humano.

- **Autónomos:** Son robots que no requieren de ningún tipo de interacción humana para poder movilizarse en su entorno.

2.2.1. Cinemática de un robot móvil

La cinemática del robot estudia el movimiento del mismo con respecto a un sistema de referencia:

- Descripción analítica del movimiento espacial del robot como una función del tiempo.
- Relaciones entre la posición y orientación del extremo del robot (localización) y los valores de sus coordenadas articulares.

Existen 3 modelos para realizar la cinemática de un robot móvil:

- **Problema cinemático directo:** Determinar la posición y orientación del extremo del robot, con respecto a un sistema de coordenadas de referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot.
- **Problema cinemático inverso:** Determinar la configuración que debe adoptar el robot para alcanzar una posición y orientación conocidas.
- **Modelo diferencial (matriz Jacobiana):** Relaciones entre las velocidades del movimiento de las articulaciones y las del extremo del robot[WEB03].

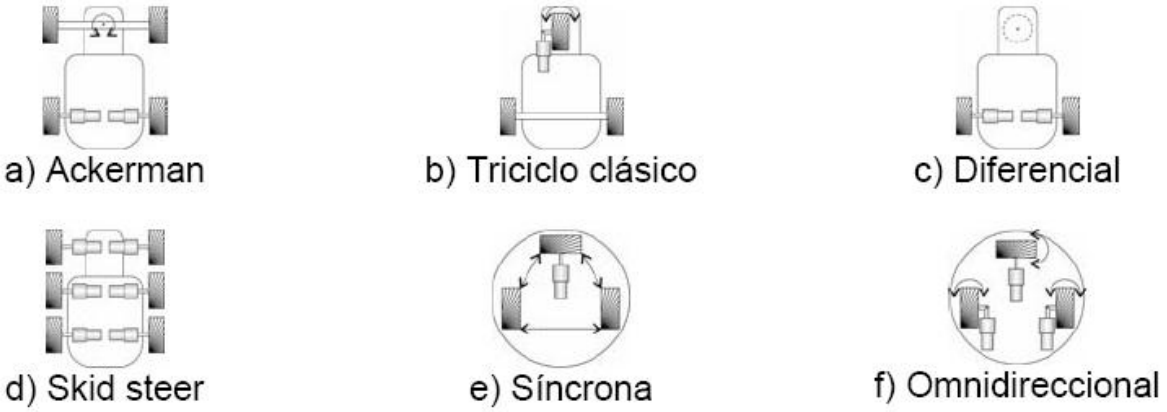
2.2.2. Configuraciones cinemáticas

Existen numerosos sistemas para proporcionar movilidad a un robot, siendo las características de maniobrabilidad y controlabilidad las que se deben tener más en cuenta, sin embargo no debemos olvidar aspectos básicos como la eficiencia o estabilidad, a continuación se verán los distintos sistemas rodados de locomoción en robótica (RMR).

Las configuraciones de los RMR son muy variadas, y dependen principalmente de la aplicación que se le va a dar al sistema, sin embargo se pueden diferenciar las siguientes configuraciones generales (Ver [Imagen 4])

a) Ackerman
b) Triciclo clásico

c) Tracción diferencial
d) Skid steer
e) Síncrona
f) Tracción omnidireccional



Configuraciones de los RMR

[Imagen 4]

2.3. Localización

El principal problema de la localización en robots, se reduce a contestar una simple pregunta: ¿Dónde estoy?, desde el punto de vista del robot. Esto significa que el robot debe ser capaz de descubrir su posición relativa con respecto al entorno. La localización en robots, es un tema muy importante, ya que de esto depende el éxito o el fracaso de un sistema autónomo.

Al momento de obtener la posición relativa de un robot podemos utilizar entre otros, dos tipos de metodologías, Odometría y Navegación inercial.

- **Odometría:** La palabra "odometría" se compone por las palabras griegas hodos ("viajar", "trayecto") y metron ("medida"), es la técnica más utilizada al momento de estimar la posición de un robot. Sin embargo, debido a los errores derivados de la integración de ruedas en los robots, la distancia recorrida y la orientación, pueden no ser exactas.

  A pesar de que esta técnica produce un incremento en el error de la estimación de la posición, es el método más fácil para acceder a la información de a

ubicación y por lo tanto es una fuente importante de información para la localización.

- **Navegación inercial:** Utilizando giroscopios y acelerómetros para medir los giros y la aceleración que el robot realizo a lo largo del tiempo. Al igual que con odometría, las estimaciones de posición de navegación inercial se adquieren mediante la integración de la información obtenida de los sensores (velocidad y distancia recorrida) del robot.

  Ambos sistemas son independientes de las fuentes externas de información.

Otro tipo de medición que es posible utilizar al momento de localizar un robot, es obtener la posición absoluta del robot, la cual es independiente de los anteriores resultados de la estimación de la posición, a diferencia de la posición relativa, la localización no deriva de la integración de las mediciones anteriores, sino que deriva directamente de una medición. La posición absoluta puede proveernos de la localización completa del robot, como también de parte de ella, por ejemplo la orientación.

Existen diferentes formas de obtener la localización absoluta de un robot, y estas técnicas pueden dividirse en **Basadas en sitios conocidos (landmarks)** o **Basadas en mapas**.

- **Basadas en sitios conocidos:** Se basa en la utilización de marcas en el medio ambiente que un robot puede detectar. Las lecturas del sensor de un robot son analizadas en busca de dichas marcas. Una vez detectadas, se las compara con el conocimiento previo que se tenga con respecto a ellas con el fin de determinar la posición. Estas marcas pueden ser catalogadas en activas y pasivas:
  - **Activas:** Son aquellas que activamente pueden enviar información sobre la su posición al robot, en general pueden obtenerla de satélites u otros dispositivos, el robot simplemente escucha en busca de estas señales y así logra determinar su posición.
  - **Pasivas:** Este tipo de marcas, no envían ningún tipo de señal, el robot activamente las busca para adquirir su posición.
- **Basadas en mapas:** Este enfoque utiliza las características geométricas del entorno para calcular la ubicación del robot. Sin embargo utilizar esta técnica para determinar la posición absoluta de un robot tiene la desventaja de que es

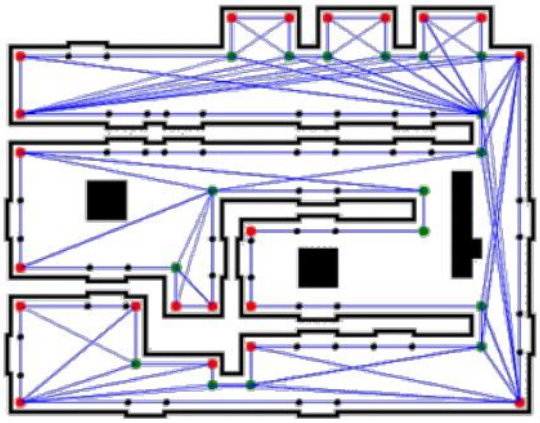
necesario que haya suficiente información de los sensores para que se genere un mapa lo suficientemente preciso para llegar a determinar la posición correctamente.

2.4. Mapeo

Uno de los objetivos de un robot autónomo es lograr ser capaz de construir o utilizar un mapa (3D o 2D) y luego localizarse a sí mismo en el.

Existen diferentes formas de generar un mapa que el robot sea capaz de utilizar, aquí se enumeran los más importantes[WEB04]:

- **Mapeo Topológico simple:** El mapeo topológico considera cada lugar y la relación con los objetos que se encuentran en el, en general se almacenan las distancias entre los objetos situados en el mapa, este tipo de mapeo es representado así por un grafo en el cual los notos corresponden a cada lugar y los arcos son los caminos que unen cada lugar en el. Los mapeos topológicos en general tienen problemas para representar "espacios abiertos", dado que para obtener este tipo de mapas, se libera un robot en un espacio cerrado y estático que el recorrerá, y a medida que lo haga, genera el mapa completo.



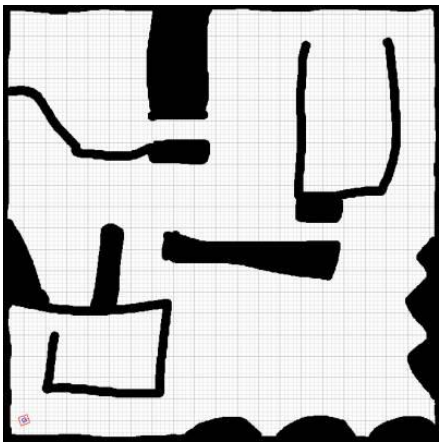
[Imagen 5]

Se trata de una representación simple, que es bueno para ser almacenado, y compartido con otros robots, esencialmente codifica solo la información necesaria para la navegación.

- **Mapeo Métrico:** El enfoque métrico es el más intuitivo para los humanos, y solo considera espacios en dos dimensiones, en los cuales los objetos se posicionan, utilizando coordenadas precisas. Si bien esta representación es muy útil, también es muy sensible a ruidos.

El tipo de mapa métrico más común, es la matriz de ocupación, básicamente se podría interpretar como un pedazo de papel cuadriculado, en el cual cada cuadricula es rellenada si el espacio está ocupado, y se deja vacio si este está libre (en general se utiliza la probabilidad para calcular la ocupación de cada espacio.

La lectura de los sensores, es fácilmente utilizable en este tipo de mapeo, de allí su predominio en las técnicas de SLAM, el principal problema lo encontramos en el uso de la memoria, si tenemos que mapear un gran espacio abierto, se ocupa demasiado espacio para almacenarlo, para solucionar esto es posible utilizar tamaños de celda dinámicos que pueden ser calculados de forma computacional muy eficientemente ahorrando así memoria.



[Imagen 6]

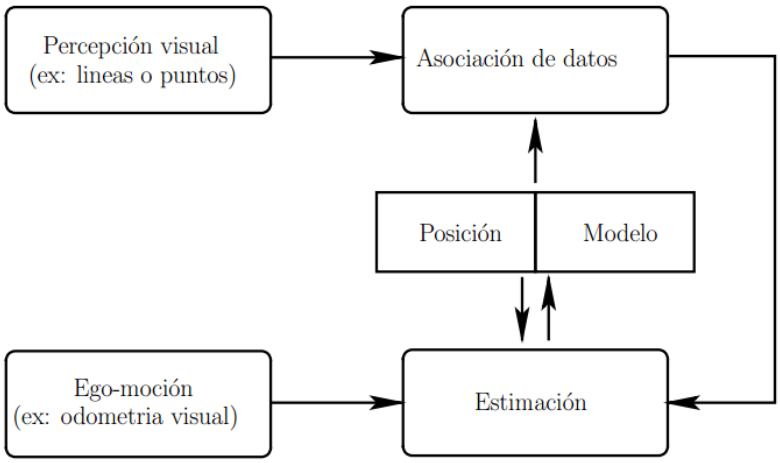
2.5. SLAM (Simultaneous Localization and Mapping)

SLAM es el proceso por el cual un robot genera un mapa del entorno en el que se encuentra y el que a su vez luego utiliza para calcular su posición, si bien este tipo de solución puede ser pensada como un problema con el cual podemos fácilmente entrar en un ciclo infinito, ya que un mapa libre sin sesgo es lo que se necesita para la calcular la localización precisa del robot, mientras que una estimación de este tipo necesita a su vez ese mapa. Sin embargo, este problema, como veremos en esta sección, muy fácil de evitar, logrando que el robot genere el mapa y calcule su posición al mismo tiempo.

- Navegación = Localización + Mapeo
    - **Localización**: Inferencia de la localización, dado un mapa del entorno.
    - **Mapeo**: Inferencia del mapa, dada la localización.

La forma general y más simple de lograr la localización simultanea y el mapeo (SLAM) es utilizando la recursión bayesiana, como se puede ver en el esquema [Imagen 7].

- **Asociación de datos**
    - Asociar correctamente los datos observados a los elementos de mapa a que corresponden
    - En visión computacional, y otros sistemas sensoriales como sonares y radares, problema de reconocimiento
- **Estimación**
    - El problema principal se resumen en determinar el tamaño (alcance) del estado que considerar a la hora de realizar la estimación.
    - Procesos generalmente no lineales[WEB05].



[Imagen 7]

Existen diferentes tipos de SLAM

1. Grid-based (Basado en grillas / matriz de ocupación)
2. Feature-based (Basado en funciones)
3. Topological (Topológica)

2.6. Planificación

El problema fundamental en la robótica es decidir cuáles son los movimientos que el robot debe realizar a fin de lograr alcanzar su meta evitando tropezar con los objetos físicos que se interponen en su camino.

Debido a la versatilidad de los robots, este resulta ser un problema extremadamente difícil, especialmente si tenemos que realizar tareas complejas que involucren una realimentación de los sensores.

La planificación en robótica puede dividirse en dos grandes grupos:

- Planificación de tareas
- Planificación de movimiento

Utilizando como base que el mapa es estático, es decir que no existen objetos móviles además del robot, podemos aproximar el modelo de estado y lograr una planificación correcta utilizando algunos de los siguientes modelos:

- Sistemas CAD, con el fin de obtener un mapa de los objetos y en el mapa y las configuraciones deseadas
- Utilizar los sensores del mismo robot para lograr localizar los objetos
- El método más común, es utilizar relaciones espaciales simbólicas entre los objetos, estas relaciones son almacenadas en formato de números a fin de disminuir el espacio necesario para el almacenamiento

Una vez obtenido el mapa y las configuraciones espaciales de los objetos, se implementan diferentes tipos de algoritmos, que completan el traslado del robot del punto a al punto b evitando todos los posibles obstáculos[WIKI02].

- Búsqueda basada en grillas
- Algoritmos geométricos
- Campos potenciales
- Algoritmos basados en muestras

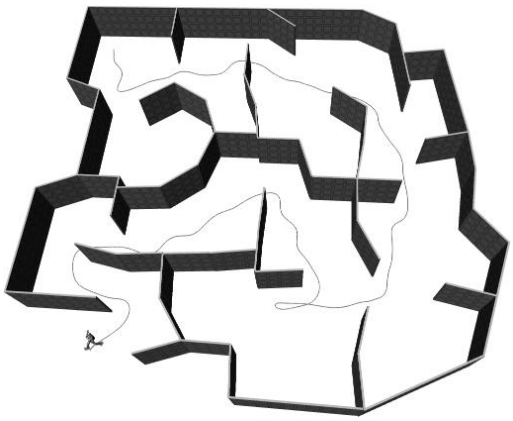


Imagen: Mapas basados en CAD

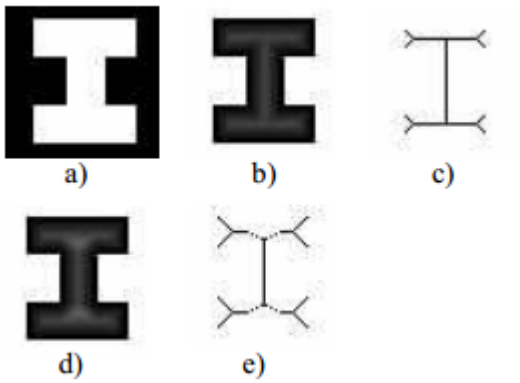
2.7. Visión Robótica

Existen diferentes definiciones del concepto de “visión”, una forma de definirlo, es la reconstrucción de las características del entorno a través de la utilización de imágenes.

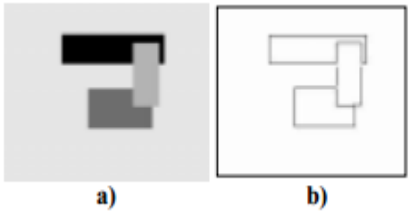
Si bien la visión biológica es efectiva, a nivel computacional tenemos varios factores a favor, en el momento de procesar imágenes de manera artificial, por ejemplo la óptica de una cámara puede ser fácilmente modificada (Zoom, Campo de visión, etc.), tenemos un mayor espectro (infrarrojos, ultravioletas) y la velocidad de obtención de las imágenes es mayor que la del ojo humano[WEB06].

Una vez que la imagen es obtenida debe ser procesada, algunas de las principales formas de procesamiento de imágenes incluyen:

- Técnicas de segmentación por escalas de grises
- Detección de bordes
- Morfología digital
- Texturas
- Algoritmos de esqueletización



Proceso de esqueletización de una imagen



Ejemplo de detección de bordes

A lo largo de los siguientes capítulos veremos más en detalle los temas vistos en esta sección de forma general, y nos centraremos en las características básicas que debe poseer un robot móvil autónomo para poder alcanzar el objetivo para el cual fue creado. Si bien a lo largo de este trabajo no se hará hincapié en ninguna implementación en particular, se presentara el marco teórico sobre el cual se podrán realizar futuras implementaciones.

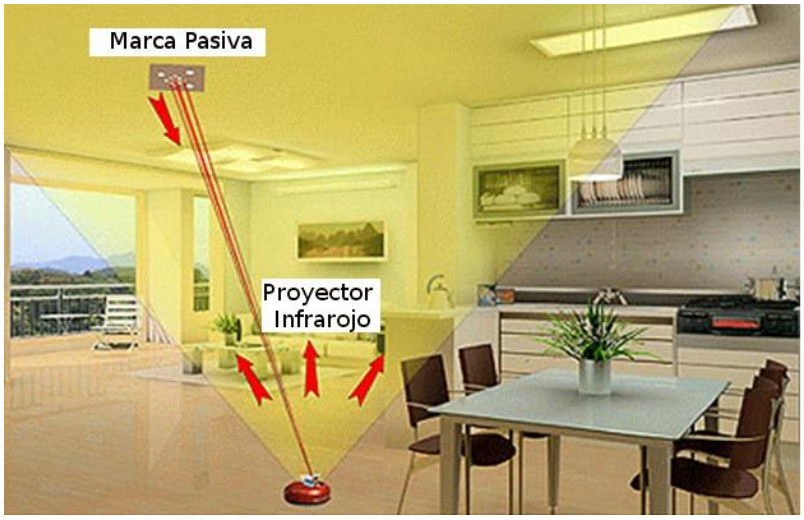
3. Localización

3.1. Introducción

Los robots que tienen la capacidad de moverse libremente en su entorno, deben enfrentar los mismo problemas que enfrentamos los humanos cuando nos movemos, si los robots solo se movieran sin mirar a su alrededor, rápidamente perderían el rumbo y se perderían. Por esto al igual que los humanos usan sus sentidos, los robots pueden valerse de sensores, que los ayuden a saber donde están.

Consideremos el siguiente escenario, un robot se encuentra en una posición desconocida en un entorno en el cual existe un mapa, el robot debe, basándose en observaciones de los sensores que posea, inferir el lugar del mapa en el cual se encuentra (su posición), esto se conoce comúnmente como el problema de la localización robótica.

El uso correcto de técnicas de localización elimina la necesidad de contar con sensores de posicionamiento complejo, por el contrario, utilizando sensores de menor complejidad y la correcta implementación podemos lograr información con un alto grado de exactitud. Una forma simple de lograr esto es, como muestra la imagen 10, utilizando marcas que el robot pueda reconocer, sin embargo esto no siempre es realizable, por lo cual debemos buscar maneras igualmente efectivas que logren el objetivo sin la necesidad de ningún tipo de agregado al entorno.



[Imagen 10]

Si bien este problema se reduce a ser un problema fundamentalmente geométrico, para lograr completar esta tarea existen diversas aplicaciones prácticas, cada una con sus pros y contras, los cuales analizaremos a lo largo de este capítulo.

3.2. Métodos de localización robótica

3.2.1. Monte Carlo (MCL)

En la robótica, la localización Monte Carlo (MCL) es un método muy poderoso, que se basa en la metodología del mismo nombre introducida en la década del 1940 por John von Neumann, Stanislaw Ulam y Nicholas Metropolis.

3.2.1.1. Metodología Monte Carlo

La metodología Monte Carlo se base una clase de algoritmos computacionales que utilizan el muestreo repetido aleatorio para calcular los resultados, este método se utiliza a menudo en simulaciones computacionales de sistemas físicos y matemáticos [WE07]. Para determinar la posición de un robot dado un mapa de su entorno basado en la localización de Markov es posible utilizar la localización Monte Carlo.

Se trata básicamente de una implementación del filtro de partículas aplicado a la localización robótica, y se ha vuelto muy popular en los últimos años. Básicamente, un gran número de hipotéticas configuraciones actuales son inicialmente dispersadas al azar en el espacio de configuración y a medida que los sensores devuelven información se va actualizando la probabilidad de cada una de las configuraciones iniciales, basándose en el modelo estadístico de los sensores y el teorema de Bayes[WEB08].

3.2.1.2. Filtros por Partículas

El objetivo principal de los filtros por partículas es rastrear a la variable de interés mientras va evolucionando a lo largo del tiempo, en general este seguimiento se realiza utilizando una función de densidad de probabilidad (fdp) no Gaussiana y multimodal. La base de este método es construir una representación muestra de la fdp, a partir de ese modelo, se realizan diferentes acciones con el fin de modificar la variable de interés de acuerdo a un modelo predefinido.

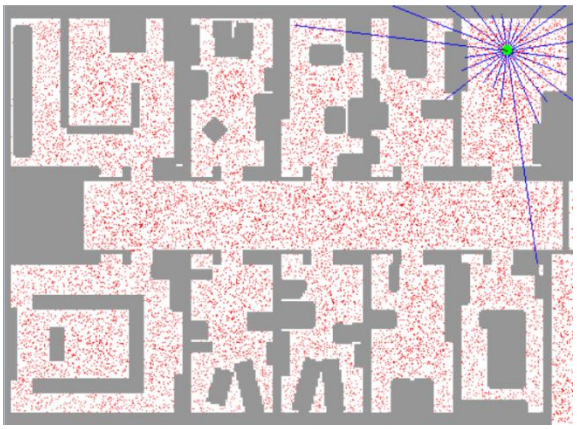


Imagen 8

Se utilizan simultáneamente múltiples copias (partículas) de la variable, cada una asociada con un peso diferente, que especifica la calidad de cada partícula, la estimación de la variable de interés es obtenida por la suma de los pesos de cada partícula. El filtro por partículas es recursivo por naturaleza y opera en dos fases: predicción y retroalimentación, con cada iteración, cada una de las partículas es modificada de acuerdo al modelo especificado (estado de predicción), al mismo tiempo que se incluye ruido aleatorio para simular los efectos reales del ruido a la variable de interés. Luego, el peso de cada partícula es reevaluado basado en la última medición disponible de los sensores (estado de |retroalimentación / actualización). A medida que las partículas se actualizan, aquellas que poseen pesos muy pequeños son eliminadas, a este proceso se lo conoce como re muestreo.

De una manera más formal, la variable de interés, en nuestro caso la posición del robot en movimiento $X^k = [x^k, y^k, \theta^k]^T$, en el tiempo $t = k$ es representada como un conjunto de muestras (partículas) $S_i^k = \left[X_j^k, w_j^k\right]: j = 1 \dots M$, donde el subíndice j denota a la partícula, no al robot, cada partícula consiste en una copia de la variable de interés y de un peso (w_j^k) que define la contribución de esta partícula a la estimación general de la variable.

Si en un determinado tiempo $t = k$ conocemos el estado previo de la fdp del sistema (tiempo $t = k - 1$), podemos modelar (predecir) el efecto que obtendremos en el tiempo $t = k$, es decir, la fase de predicción utiliza un modelo con el fin de simular el efecto que una acción va a causar sobre un grupo de partículas con el correspondiente ruido agregado y luego, la fase de retroalimentación utiliza la información de los sensores para actualizar los pesos de cada partícula y así describir de forma correcta el movimiento del robot.

Dada una distribución estándar de partículas, debemos realizar ciertas acciones basándonos en la pose del robot, existen 3 métodos para determinar la pose del robot. Primero, la media del peso de las partículas $P_{est} = \sum_{j=1}^{M} w_j X_j$, segundo, la mejor partícula (la P_j tal que $w_j = max(w_k): k = 1 \dots M$ y tercero, combinando los dos métodos anteriores, se obtiene a partir de la media de las partículas que están próximas a la mejor, este es en general el mejor método, pero también el más complicado de obtener.

3.2.1.3. Teorema de Bayes

En la teoría de la probabilidad el teorema de Bayes es un resultado enunciado por Thomas Bayes en 1763 que expresa la probabilidad condicional de un evento aleatorio A dado B en términos de la distribución de probabilidad condicional del evento B dado A y la distribución de probabilidad marginal de sólo A [WEB09].

De una manera más formal, el teorema de Bayes puede ser definido en la siguiente fórmula matemática:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)}$$

Donde,

$P(A_i)$ Representa la probabilidad a priori

$P(B|A_i)$ Representa la probabilidad condicional

$P(B)$ Representa la probabilidad total

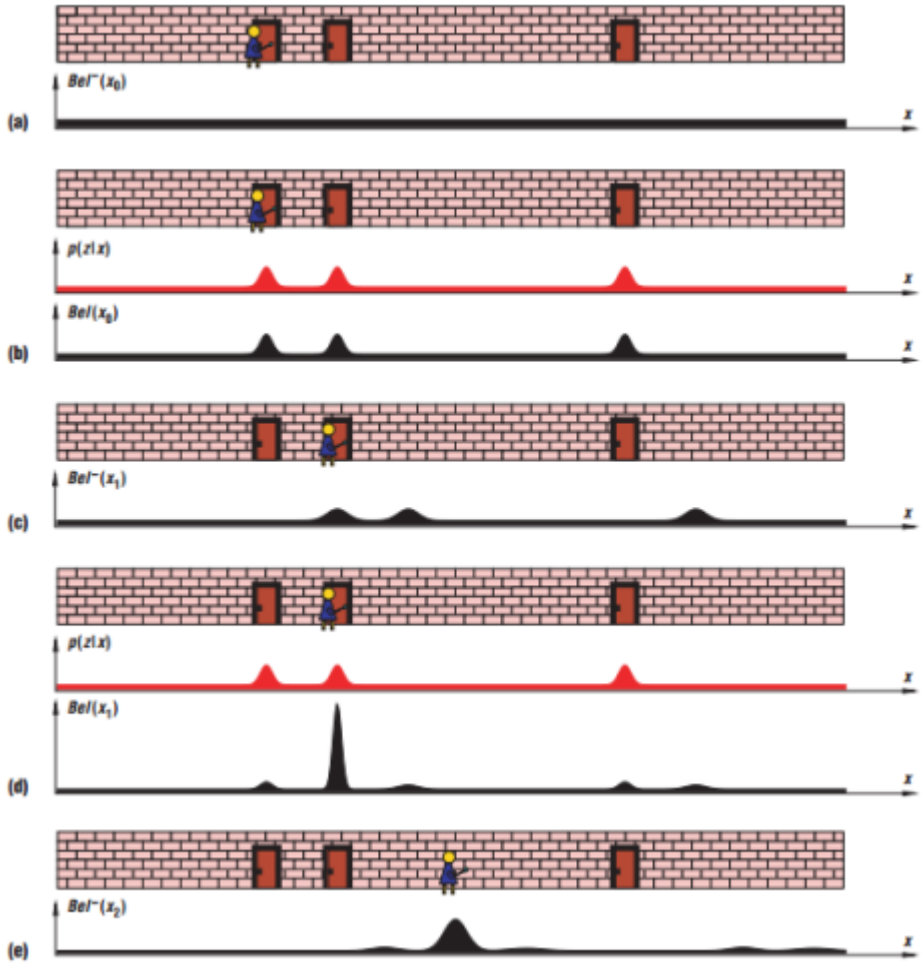
$P(A_i|B)$ Representa la probabilidad a posteriori

3.2.1.3.1. Filtro de Bayes

Los filtros que se realizan utilizando el teorema de Bayes resuelven el problema de estimar el estado de x en un sistema dinámico (cadena de Markov parcialmente observable) utilizando las mediciones recolectadas de los sensores, por ejemplo, en nuestro caso, el sistema dinámico es el robot móvil y su entorno, el estado es la posición del robot (en general representa por x, y, θ) [WEB10].

La imagen 9 muestra una ilustración de una dimensión de los filtros de Bayes. Una persona lleva una cámara que es capaz de detectar puertas, pero que no puede distinguir entre diferentes puertas. Cada cuadro representa la posición de la persona en el pasillo y la actual creencia $\mathrm{Bel}(\mathrm{xt})$: (a) la ubicación de la persona es desconocida. (b) El sensor envía la señal de "puerta encontrada". (c) La persona se mueve. (d) El sensor observa otra puerta. (e) La persona se mueve de nuevo. Adicionalmente, (b) y (d) representan el modelo de observación $\mathrm{p}(\mathrm{z} \mid \mathrm{x})$, la probabilidad de observar una puerta en diferentes lugares en el pasillo.

La teoría bayesiana es una rama de la teoría de la probabilidad matemática que permite a las personas modelar la incertidumbre sobre una variable y los resultados de esta mediante la incorporación de conocimiento previo y la evidencia observacional. El análisis Bayesiano, es la forma de interpretar la probabilidad condicional como una medida de incertidumbre, es uno de los métodos más populares para resolver los problemas inversos.



[Imagen 9]

Estadísticas Bayesianas suficientes: Dado $P(x,y)$ (densidad de la probabilidad de x dado y), una estadística $\Psi(\mathrm{x})$ se dice que es "suficiente" si la distribución de x, dado Ψ, no depende de y, en otras palabras, dado $P(x,y) = P(x,y')$, cualquiera sean y e y', $\Psi(\mathrm{y}) = \Psi(\mathrm{y}')$ [WEB11].

Existen tres tipos de problemas de difícil solución que se desprenden de la definición de la estadística bayesiana:

- **Normalización:** Dado la información a priori $P(x)$ y la probabilidad $P(y|x)$, la información a posteriori $P(x|y)$ es obtenida a partir de la multiplicación de los dos primeros, divido por un factor de normalización:

$$P(x|y) = \frac{\mathrm{P}(y|x)P(x)}{\int_x \mathrm{P}(y|x)P(x)dx}$$

- **Marginalización:** Dada la conjunción posterior (x, z), la posterior marginal se define como:

$$P(x|y) = \int_{\boldsymbol{z}} \mathrm{P}(x, z|y)dz$$

- **Expectativas:** Dado la función de densidad de probabilidad condicional, algunas estadísticas de interés se pueden calcular de la siguiente manera:

$$E_{P(x|y)}[\boldsymbol{f}(\boldsymbol{x})] = \int_{\boldsymbol{x}} \mathrm{f(x)P}(x|y)dx$$

La estadística suficiente $\Psi(\mathrm{x})$ contiene toda la información presentada por x sobre y. El teorema de Rao-Blackwell dice que cuando un estimador se evalúa en una pérdida convexa, el procedimiento óptimo sólo depende de las estadísticas suficientes. Principio de Suficiencia y Principio de Verosimilitud son dos principios axiomáticos en la inferencia bayesiana[WEB11].

3.2.1.4. Localización Monte Carlo

La localización Monte Carlo, también conocida como implementación de filtros "bootstrap", se basa en una combinación de algoritmos, el conjunto de estos es conocido como filtro por partículas. La idea principal con esta metodología es representar el conocimiento posterior (posición), $\mathrm{Bel(l)}$, a través de un conjunto N de partículas aleatorias $(S = \{S_i | i = 1..N\})$, cada una ponderada con un peso diferente. La combinación inicial de partículas se aproxima a una distribución de probabilidad predefina (FDP).

Cada partícula en la localización Monte Carlo, se define como $[\langle x, y, \theta \rangle, p]$, donde, $\langle x, y, \theta \rangle$ representan la posición del robot, y $p \geq 0$ representan el peso de esa partícula (en un sentido más práctico, representa la probabilidad de cada partícula, por lo cual para mantener la consistencia, se asume $\sum_{n=1}^{N} p_n = 1$).

El estado inicial de las partículas representa un conjunto de estados (posiciones), dispersados utilizando una distribución uniforme a lo largo del universo del robot. El algoritmo recursivo que se utiliza para obtener la posición del robot, es realizado siguiendo los siguientes pasos:

- Realizar el muestreo de $x_i^{(t-1)} \sim Bel(l^{(t-1)})$, el cual representa un estado anterior $(t-1)$ del muestreo de partículas $Bel(l^{(t-1)})$.

- Realizar el muestreo de $x_i^{(t)}$, el cual representa un estado propuesto para el tiempo t del muestreo de partículas $Bel(l^{(t)})$, también conocido como la distribución propuesta.
- Luego se debe realizar la diferencia entre la distribución propuesta y la distribución deseada, luego de generar los m estados, los pesos de cada partícula deben ser normalizados para lograr que sumen 1 [WEB12].

Imagen 11

La imagen 11, muestra 3 diferentes estados por los que el algoritmo de localización global por partículas pasa aplicado a la robótica móvil, utilizando localización Monte Carlo.

Si bien este método ha comenzado a ser el más utilizado en los últimos años, no es perfecto y presenta algunos inconvenientes bajo ciertas condiciones.

Un ejemplo muy claro, es el pobre desempeño que presenta el filtro por partículas básico, cuando la distribución propuesta para generar las partículas, genera pocas muestras en lugares donde el posterior deseado $\mathrm{Bel(l)}$ en muy grande.

3.2.2. Localización a través de Filtros Kalman

En este apartado nos centraremos en la revisión detallada de la formulación del Filtro de Kalman (KF) para localización robótica.

3.2.2.1. Filtros de Kalman (KF)

Los filtros de Kalman, introducidos en la década de 1960 (Kalman y Bucy, 1961), son un método secuencial de asimilación de datos, existen dos estados involucrados para lograr resolver el conjunto de ecuaciones que definen el estado del sistema, primero, el estado de previsión, en el cual, dado una estimación inicial, o un estado previo, pronostica el estado del sistema, realizando esta estimación a partir de un modelo lineal. Cada estado previo contiene un cierto grado de error de covarianza, el cual debe ser arrastrado e ir evolucionando a lo largo del tiempo, el método de filtros Kalman logra esto.

El segundo estado es el estado de análisis, en el cual se realiza un promedio de los errores que están incluidos en la predicción obtenida en el estado anterior, y los errores conocidos por la observación del sistema, en base a esta comparación, se realiza la asignación de pesos, los que finalmente se utilizan para calcular y analizar el error de covarianza[web13]. En resumen, un filtro Kalman es un algoritmo recursivo que estima el estado de un sistema lineal dinámico que posee ruido.

Dado una matriz conocida M_j y un modelo lineal de un sistema dinámico en un tiempo j, el estado real del sistema puede ser definido como:

$$x^t(t_{j+1}) = M_j x^t(t_j) + n(t_j)$$

-Donde:

x Es el vector de estados del sistema de dimensión n

x^t Es el estado real del sistema

$x^t(t_j)$ Es el estado real del sistema en el tiempo t_j

$n(t_j)$ Es el modelo de error aleatorio en el tiempo t_j

Las ecuaciones de estado de definen de la siguiente manera:

- Estado de previsión

$$x^f(t_j) = Mx^a(t_{j-1}) \text{ y}$$

$$P^f(t_j) = MP^a(t_{j-1})M^T + Q$$

- Estado de Análisis

$$K(t_j) = P^f(t_j)H^T\left(HP^f(t_j)H^T + R\right)^{-1}$$

Con el fin de lograr una correcta predicción de los estados, el filtro de Kalman actúa sobre la base de corregir cada predicción, es decir, luego de realizar la predicción basándose en la dinámica del sistema, lo corrige usando las mediciones del mismo.

De una manera más formal, el filtro Kalman estima la probabilidad condicional de un estado x_k dados las mediciones conocidas $z_1, \dots, z_k$ de la siguiente manera:

$$Bel(x_k) = P(x_k|z_1, \dots, z_k)$$

Así mismo, podemos dividir la ecuación anterior en la probabilidad anterior $Bel^-(x_k)$ y la probabilidad posterior $Bel^+(x_k)$:

$$Bel^-(x_k) = P(x_k|z_1, \dots, z_{k-1})$$

$$Bel^+(x_k) = P(x_k|z_1, \dots, z_k)$$

Entiéndase $Bel^-(x_k)$ como la probabilidad condicional de estar en el estado x_k dados todas las mediciones de z hasta el estado k, y $Bel^+(x_k)$ como la probabilidad condicional de estar en el estado x_k dados todas las mediciones de z incluyendo al estado k. Esta corrección es necesaria dado que por definición, el sistema sobre el cual trabajamos posee ruido [web14].

3.2.2.2. Localización utilizando filtros Kalman

La pregunta que intentaremos responder en esta sección es porque utilizar filtros Kalman para localizar un robot.

Para modelar la posición del robot deseamos conocer sus coordenadas X e Y y su orientación. Estos tres parámetros se pueden combinar en un vector llamado vector de variables de estados. El robot utiliza las mediciones de distancia, de ángulos y la información de la locomoción para saber cuánto se ha caminado y así calcular su posición.

Al igual que con cualquier sistema real, estas mediciones incluyen un componente de error (o ruido), si utilizamos solo las nociones de trigonometría para calcular la posición del robot puede tener un gran nivel de error lo cual puede generar posibles localizaciones que cambian de manera significativa con cada nueva medición de los sensores. Esto hace que el

robot parezca que está "saltando" por todo el entorno. El filtro Kalman es una manera inteligente de integrar los datos de cada medición y el ruido del sistema en una estimación más correcta del estado.

El efecto del ruido del sistema puede ser variable, y esto depende del la variable que el usuario le asigne al filtro, esto, incluyendo las mediciones, las entradas y el modelo calcula una estimación de la posición en la cual se tiene en cuenta el ruido generando así una medición con cierto grado de confianza sobre la posición actual.

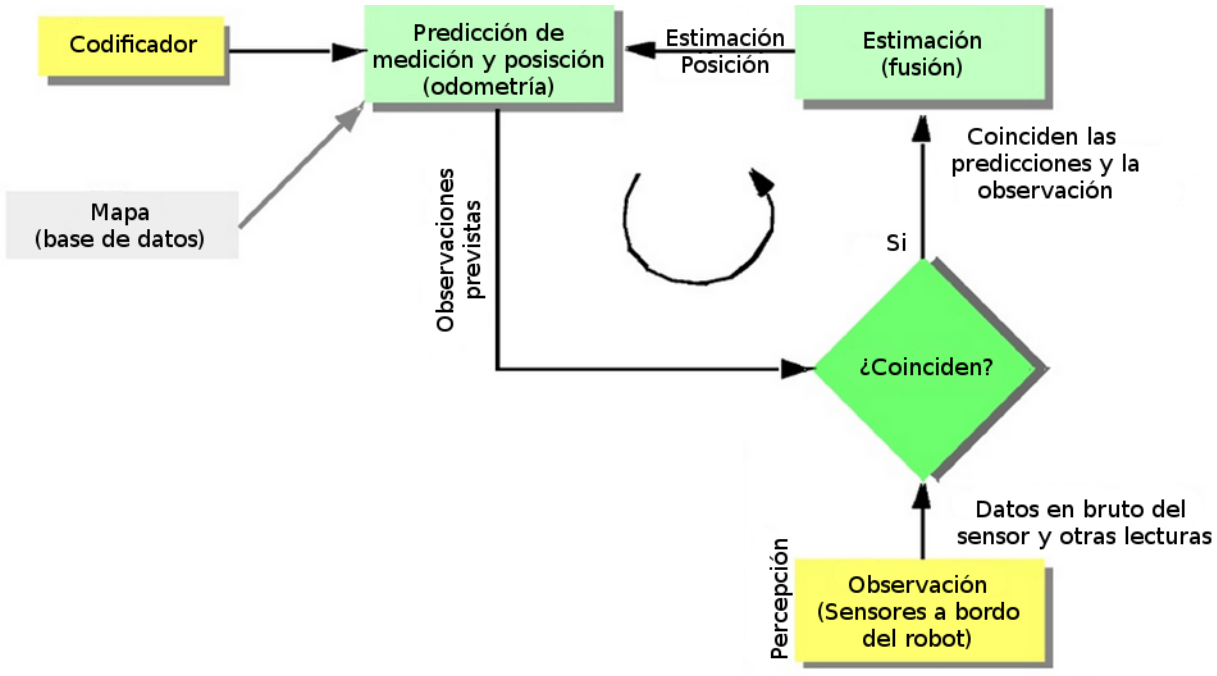


Imagen 12

Una forma de lograr la lograr localizar al robot móvil utilizando el filtro Kalman es siguiendo los pasos descriptos en el flujo de la imagen 12*:

a) **Predicción de la Posición:** Basándose en las información del estado anterior.

b) **Observación:** La información cruda de los sensores del robot.

c) **Predicción de la medición:** Basándose en las información de la predicción del paso (a) y de la información de los mapas.

d) **Cotejamiento:** Utilizando la información de la observación y los mapas.

e) **Estimación:** Actualización de la posición

Teniendo como base que tanto el modelo de movimiento, como los modelos de los sensores son Gaussianos, cada estado de la función es caracterizado unívocamente por su media μ y por su matriz de covarianza $\sum$. Si bien conocemos el comportamiento del sistema y tenemos el mismo modelizado, no conocemos el ruido real que el sistema posee, solo podemos estimarlo.

Tanto el modelo de movimiento, como el del sensor pueden ser definidos de la siguiente manera[web15]:

- **Modelo de movimiento** (Se trata de un sistema lineal, discreto y dinámico a través del tiempo) [web16]

$$X_{t+1} = F_t X_t + B_t U_t + G_t w_t$$

Donde:

- X representa el estado
- F representa la función de transición de estado
- B representa la función de control de entradas (input)
- U representa el control de entradas
- G representa la función de ingreso de ruido (con covarianza)
- w representa el proceso de ruido
- t representa el tiempo

- **Modelo del sensor** (Se trata de una ecuación de medición) [web16]

$$Z_{t+1} = H_{t+1} x_{t+1} + n_{t+1}$$

Donde:

- Z representa la lectura del sensor
- H representa la función del sensor
- x representa el estado
- n representa el ruido del sensor con covarianza R.

Un diagrama de bloques puede ayudar a entender estos modelos:

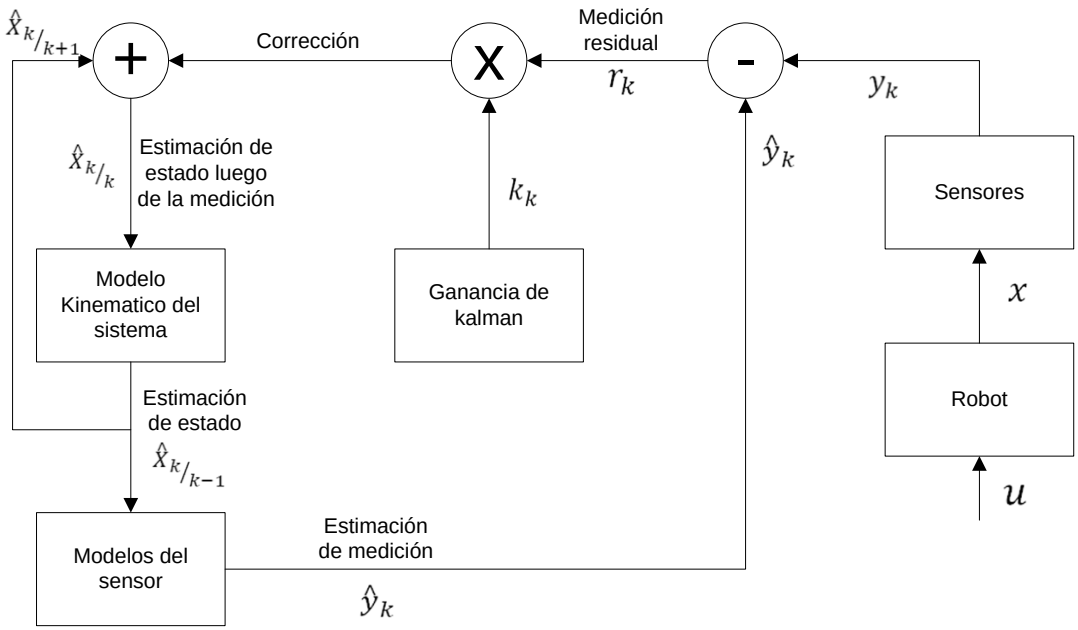


Imagen 13

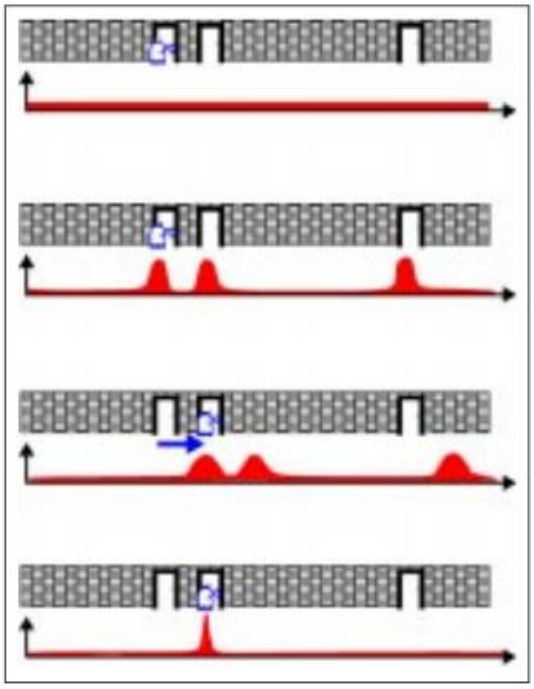
En conclusión teniendo en cuenta la gran cantidad de experimentos disponibles en diferentes papers, podemos decir que es posible obtener una estimación de una posición de un robot estacionario utilizando un filtro de Kalman con una precisión de unos pocos centímetros y con una coherencia razonable. El rendimiento de la localización puede disminuir a medida que el robot se mueve, pero es altamente dependiente de la cantidad de datos de medición que se recibe cuando está caminando.

3.2.3. Localización Markov

En la siguiente sección se analizara de manera detallada la formulación del modelo de Markov aplicado a la localización robótica.

3.2.3.1. Modelo de Markov

La localización de Markov abarca el problema de la estimación del estado del robot (posición) a partir de los datos obtenidos por los sensores.



Se trata de un algoritmo probabilístico, que en lugar de mantener solo una hipótesis de donde se encuentra el robot en el mapa, el modelo de Markov mantiene una distribución probabilística de todas las hipótesis, y luego, a partir de la aplicación de las reglas del teorema de Bayes y los conceptos de convolución para actualizar la posición correcta cada vez que el robot realiza un movimiento o cada vez que recibe una actualización de los sensores.

Se asume que tanto la información del pasado como la futura son independientes del estado actual del robot [web17].

Como se definió anteriormente, se define el teorema de Bayes de la siguiente forma:

$$P(x,y) = P(x|y)P(y) = P(y|x)P(x)$$

$$\boxed{P(x|y) = \frac{P(y|x)P(x)}{P(y)}}$$

Definiremos la ley de probabilidad total como:

$$P(x) = \int P(x,z)dz$$

$$P(x) = \int P(x|z)P(x)dz$$

$$P(x|y) = \int P(x|y,z)dz$$

La convolución es un operador matemático que transforma dos funciones f y g en una tercera función que en cierto sentido representa la magnitud en la que se superponen f y una versión trasladada e invertida de g. Podemos definir más formalmente a la convolución $f * g$ con desplazamiento n de la siguiente manera[web18]:

$$f_t * g_t = \int_{-\infty}^{\infty} f_{(n)} g_{(t-n)} dn$$

3.2.3.2. Localización utilizando el modelo Markov

De la misma forma que con otros métodos, denotaremos la localización (posición) del robot móvil de manera univoca utilizando una variable tridimensional $l = (x, y\ \theta)$, siendo x e y la posición del robot en un eje cartesiano, y θ la orientación. l_t representa la posición del robot en el tiempo t y L_t denota la correspondiente variable aleatoria.

Dado que el robot no conoce su posición real, podemos definir $Bel(L_t)$ como la posición creida por el robot en el tiempo t, se trata de una distribución probabilística sobre el espacio de todas las posiciones posibles[web19].

Cada creencia se actualiza teniendo en cuenta 2 tipos diferentes de eventos:

- La llegada de nuevas mediciones provenientes de los sensores del robot
- La llegada de lectura odométrica (como por ejemplo, la cantidad de revoluciones de las ruedas)

Este método de localización estima la distribución de L_t recursivamente de la siguiente manera:

$$P(L_t = l|d) = P(L_t = l|d_0 \dots d_t)$$

Siendo $d_0 \dots d_t$, la representación de la cadena de mediciones en el cual cada d_t es la información recibida de un sensor o de una lectura odometrica.

En otras palabras, se supone que la ubicación del robot es el único estado en el medio ambiente, y sabiendo que estos datos son todo lo que hay que saber sobre el pasado para predecir el futuro. Este supuesto es claramente inexacto si el entorno contiene movimiento de otros objetos que no sea el propio robot.

El algoritmo de localización de Markov $P(L_0 = l)$, que inicializa la creencia $Bel(L_0)$, refleja el conocimiento previo acerca de la posición inicial del robot. Esta distribución se puede inicializar arbitrariamente, pero en la práctica dos casos prevalecen: Si la posición del robot con respecto a su mapa es totalmente desconocido, $P(L_0)$ generalmente se distribuye uniformemente. Si la posición inicial del robot es aproximadamente conocida, entonces $P(L_0)$ es típicamente una estrecha distribución Gaussiana centrada en la posición del robot[web19].

Una posible implementación del algoritmo utilizado para obtener la localización del robot podría ser la siguiente:

POR CADA posición l **HACER** /*Creencia Inicial*/
  $Bel(L_0 = l) \leftarrow P(L_0 = l)$
FIN POR
PARA SIEMPRE HACER
  SI nueva_lectura_sensor s_T **HACER**
    $\alpha_T \leftarrow 0$
    POR CADA posicion l **HACER**
      $\widehat{Bel}(L_T = l) \leftarrow P(s_T|l) \cdot Bel(L_{T-1} = l)$
      $\alpha_T \leftarrow \alpha_T + \widehat{Bel}(L_T = l)$
    FIN POR
    POR CADA posicion l **HACER**
      $Bel(L_T = l) \leftarrow {\alpha_T}^{-1} + \widehat{Bel}(L_T = l)$
    FIN POR
  FIN SI
  SI nueva_lectura_odometrica α_T **HACER**
    POR CADA posicion l **HACER**
      $Bel(L_T = l) \leftarrow \int P(l|l', a_T) \cdot Bel(L_{T-1} = l')\, dl'$
    FIN POR
  FIN SI
FIN PARA SIEMPRE

[Algoritmo01]

La mayoría de los métodos de localización desarrollados hasta ahora asumen que el mundo es estático y que el estado del robot es el único aspecto cambiante del mundo. Para ser capaces de localizar un robot móvil, incluso en entornos dinámicos y densamente poblados, tenemos que desarrollar técnicas para el filtrado de las mediciones de los sensores que están afectados debido a la presencia de personas u otros objetos que no están contenidos en el modelo del medio ambiente del robot.

Dado no existe un método correcto de realizar la localización de un robot, debemos evaluar cuidadosamente el método óptimo a ser utilizado en cada implementación particular, teniendo en las características del robot, la función que llevara a cabo y el ambiente con el cual tendrá interacción.

4. Mapeo

4.1. Introducción

El problema del aprendizaje de mapas es un problema importante en la robótica móvil. Contar con modelos del medio ambiente es necesario para una serie de aplicaciones, tales como el transporte, la limpieza, rescate y otras diversas tareas. El aprendizaje de mapas requiere solucionar dos problemas, la cartografía básica (mapeo) y localización. El mapeo es el problema de la integración de la información obtenida con los sensores del robot en una representación dada. Esto puede ser intuitivamente descrito por la pregunta "¿Cómo se ve el mundo?" Los aspectos centrales del mapeo son la representación del medio ambiente y la interpretación de los datos del sensor. En contraste con esto, la localización como vimos en el capitulo anterior es el problema de la estimación de la posición del robot con respecto a un mapa. En otras palabras, el robot tiene que responder a la pregunta "¿Dónde estoy?". En general estas dos tareas no se pueden resolver de forma independiente la una de la otra. Resolver ambos problemas conjuntamente se refiere a menudo como el problema de la localización simultánea y mapeo (SLAM). Hay varias variantes del problema de SLAM, incluidos los enfoques pasivos y activos, topológicos y métricos, estas variantes son las que estudiaremos a lo largo de este capítulo.

Imagen 14

4.2. El problema del mapeo robótico

El problema del mapeo en robótica se centra obtener un modelo espacial del entorno de un robot. Los mapas son comúnmente utilizados para la navegación de robots. Para adquirir un mapa, los robots deben poseer sensores que le permiten percibir el mundo exterior. Los sensores más comúnmente utilizados para esta tarea incluyen cámaras, localizadores de rango con sonar, láser y tecnología de infrarrojos, radares, sensores táctiles, brújulas, acelerómetros y GPS. Sin embargo, todos estos sensores están sujetos a errores, a menudo referido como ruido

de medición. Más importante aún, la mayoría de los sensores del robot están sujetos a estrictas limitaciones en el alcance. Por ejemplo, la luz y el sonido no pueden penetrar paredes. Estas limitaciones en el alcance hacen que sea necesario para un robot contar con un mapa para navegar a través de su medio ambiente. Los comandos de movimiento (controles) emitidos durante la exploración del ambiente contienen información importante para la construcción de los mapas, ya que ellos transmiten información sobre los lugares en los que las diferentes mediciones del sensor se tomaron. Movimiento del robot también está sujeto a errores, y los controles solos son por lo tanto insuficientes para determinar la posición de un robot (localización y orientación) con relación a su entorno[web20].

Un desafío clave en el mapeo robótico surge de la naturaleza del ruido de la medición. El modelado de estos problemas, tales como el mapeo de robótica, suelen ser relativamente fáciles de resolver si el ruido en diferentes mediciones es estadísticamente independiente. Si este fuera el caso, un robot podría simplemente tomar más y más mediciones con el fin de anular los efectos del ruido. Desafortunadamente, en el mapeo de robótica, los errores de medición son estadísticamente dependientes. Esto es debido a que en SLAM los errores se acumulan en el tiempo, y que afectan a la forma en que el robot interpreta las futuras mediciones del sensor.

4.3. SLAM

El problema de la localización simultánea y mapeo (SLAM) pregunta si es posible que un robot móvil colocado en un lugar desconocido en un entorno desconocido pueda construir incrementalmente un mapa coherente de este entorno y al mismo tiempo la determinación de su ubicación dentro de este mapa. La solución al problema de SLAM se ha visto como un "santo grial" de la comunidad robótica móvil ya que proporcionaría los medios para hacer un robot verdaderamente autónomo.

La "solución" del problema SLAM ha sido uno de los éxitos más notables de la comunidad de la robótica en la última década. SLAM se ha formulado y resuelto como un problema teórico en un número de formas diferentes. SLAM también se ha aplicado en varios dominios diferentes, [tales como robots de interior, sistemas al aire libre, tanto sobre como bajo el agua y en el aire]. A nivel teórico y conceptual, SLAM ahora se puede considerar como un problema resuelto. Sin embargo, los problemas siguen siendo importantes en la realización de soluciones [prácticas] SLAM más generales y en particular en la construcción y el uso de mapas perceptualmente ricos como parte de un algoritmo de SLAM.

Consideremos el siguiente escenario, diagramado en la imagen 15, un robot moviéndose en un ambiente desconocido, tomando mediciones relativas de las marcas que se encuentran en dicho entorno, utilizando los sensores que posee.

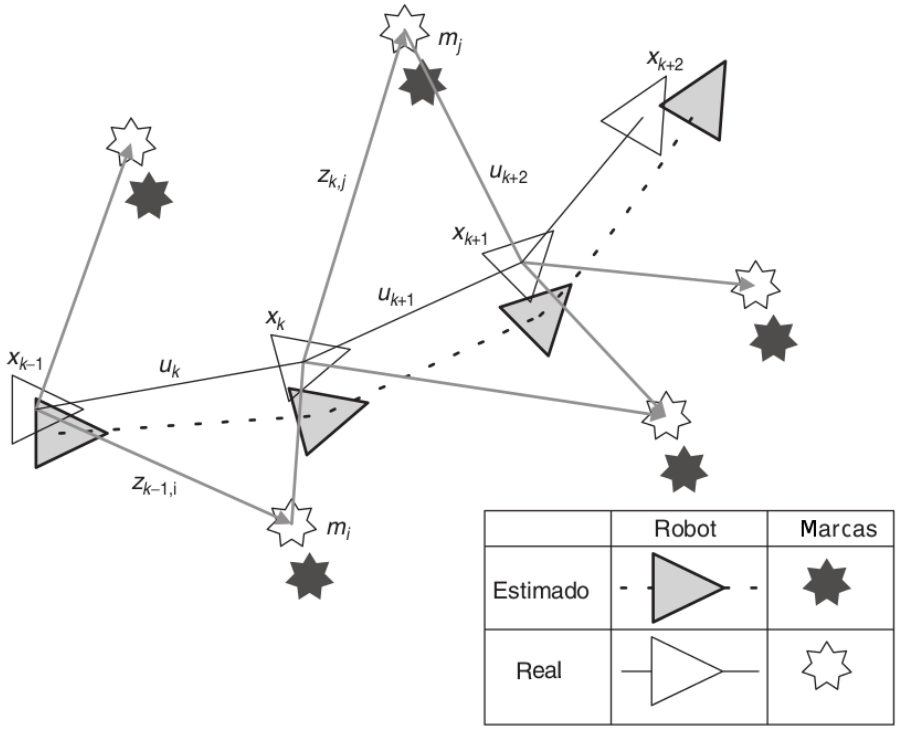


Imagen 15

En el instante k, se define cada una de las siguientes variables[web21]:

- x_k: Representa el vector de estados que describe la localización y la orientación del vehículo.
- u_k: Representa el vector de control, aplicado en el tiempo $k-1$ con el fin de trasladar al robot hasta el estado x_k en el tiempo k
- m_i: Representa el vector que describe la posición del la i-ésima marca cuya localización se considera estática.
- Z_{ik}: Representa una observación tomada desde el vehículo de la posición de la i-ésima marca en el tiempo k.
- $X_{0:k} = \langle x_0, x_1, \dots, x_k \rangle = \langle X_{0:k-1}, X_k \rangle$: Representa el histórico de la posición del vehículo.
- $U_{0:k} = \langle u_0, u_1, \dots, u_k \rangle = \langle U_{0:k-1}, U_k \rangle$: Representa el histórico de las entradas de control.

- $m = \langle m_0, m_1, \dots, m_k \rangle$: Representa el conjunto de todas las marcas.
- $Z_{0:k} = \langle z_0, z_1, \dots, z_k \rangle = \langle Z_{0:k-1}, Z_k \rangle$: Representa el histórico de las entradas de control.

En forma probabilística, el problema de la localización simultánea y mapeo (SLAM) requiere que la distribución de probabilidad $P(x_k | Z_{0:k}, U_{0:k}, x_0)$ sea calculada para todos los tiempos k.

Esta distribución de probabilidad describe la densidad posterior de la localización de las marcas y el estado del vehículo (en el momento k) dadas las observaciones registradas y las entradas de control hasta e incluyendo el tiempo k junto con el estado inicial del vehículo. En general, es deseable utilizar una solución recursiva al problema SLAM[web21].

Este cálculo requiere de un modelo de transición de estado y un modelo de observación para describir el efecto de la entrada de control y de observación, respectivamente. El modelo de observación describe la probabilidad de hacer una observación z_k cuando la localización del vehículo y de las marcas son conocidos y se describen generalmente en la forma:

$$P(z_k | x_k, m)$$

Es razonable suponer que una vez que la posición del vehículo y el mapa se definen, estas observaciones son condicionalmente independientes dado el mapa y el estado actual del vehículo.

El modelo de movimiento para el vehículo puede ser descrito en términos de una distribución de probabilidad sobre las transiciones de estado, se asume que este es un proceso de Markov, en el cual el siguiente estado x_k solo depende de su predecesor inmediato x_{k-1} y de la entrada de control aplicada u_k. Este modelo puede definirse de en la forma

$$P(x_k | x_{k-1}, u_k)$$[web21].

La solución al problema de SLAM probabilístico implica la búsqueda de una representación adecuada tanto para el modelo de observación como para el modelo de movimiento que permite calcular de manera eficiente y coherente las distribuciones anteriores y posteriores.

Por el momento, la representación más común es en forma de un modelo de espacio de estado con ruido Gaussiano adicionado, lo que lleva a la utilización de la filtro de Kalman extendido (EKF) para resolver el problema de SLAM.

4.3.1. SLAM-EKF - Filtro de Kalman extendido aplicado a SLAM

El enfoque basado en EKF de solución de SLAM se caracteriza por la existencia de un vector discreto de estados aumentados a lo largo del tiempo, compuesto de la ubicación del vehículo y la ubicación de los elementos del mapa, estimados de forma recursiva a partir de las observaciones del sensor disponibles obtenidas en el instante k, y un modelo del movimiento del vehículo, entre pasos de tiempo $k-1$ y k. Dentro de este marco, la incertidumbre se representa por las funciones de densidad de probabilidad (FDP) asociados con el vector de estado, el modelo de movimiento, así como las observaciones de los sensores. Se supone que la propagación recursiva de la media y de la covarianza de las FDPs convenientemente se aproxima a la solución óptima de este problema de estimación[web22].

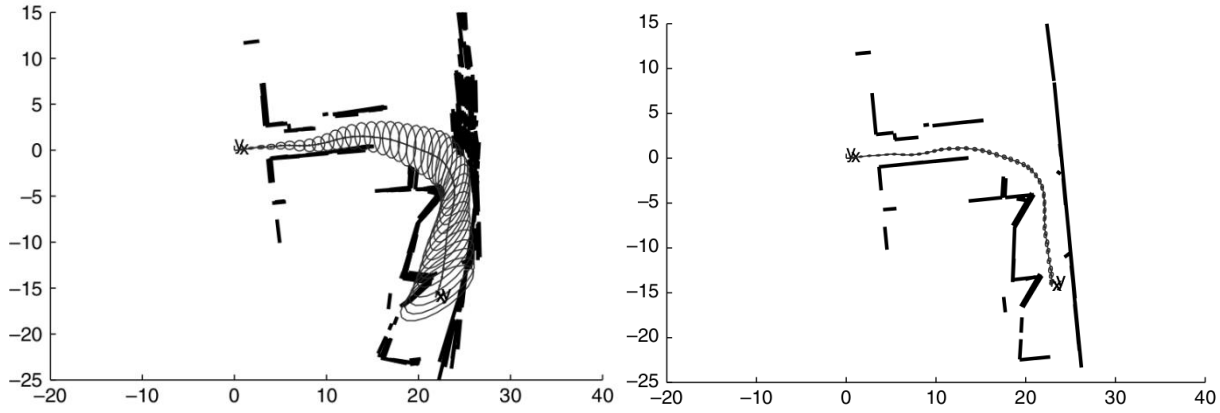


Imagen 16

En las imágenes anteriores (Imagen 16) se demuestra la necesidad de utilizar técnicas de SLAM, en la primer imagen se observan las lecturas odométrica y paredes segmentadas por láser de la trayectoria de un vehículo en el edificio y luego de aplicar los filtros se puede ver en la segunda imagen el mapa y la trayectoria resultante del algoritmo de SLAM utilizando los mismos datos (95% elipses de error se eliminan).

En la creación de un mapa estocástico, se debe seleccionar una referencia de base, es decir, se comienza a construir un mapa con respecto a una base fija de referencia diferente de la ubicación inicial del vehículo. Esto normalmente requiere la asignación de un nivel inicial de incertidumbre a la ubicación estimada del mismo.

En el caso teórico lineal, la incertidumbre de la posición del vehículo siempre debe mantenerse por encima de este nivel inicial. En la práctica, debido a linealizaciones, cuando se utiliza una incertidumbre inicial distinto de cero, la incertidumbre estimada del vehículo cae rápidamente por debajo de su valor inicial, por lo que la estimación se vuelve inconsistente después muy pocas etapas de actualización de EKF.

El siguiente algoritmo es una implementación de estas técnicas de SLAM[web22]:

$\mathbf{x}_0^B = \mathbf{0}; \mathbf{P}_0^B = \mathbf{0}$ {Inicialización del mapa}
$[\mathbf{z}_0, \mathbf{R}_0]$ = Obtener_mediciones
$[\mathbf{x}_0^B, \mathbf{P}_0^B]$ = Nuevas_características $(\mathbf{x}_0^B, \mathbf{P}_0^B, \mathbf{z}_0, \mathbf{R}_0)$
PARA $k = 1$ HASTA pasos HACER
$[\mathbf{x}_{R_k}^{R_{k-1}}, \mathbf{Q}_k]$ = Obtener_Odometría
$[\mathbf{x}_{k|k-1}^B, \mathbf{P}_{k|k-1}^B]$ = Calcular_Movimiento $(\mathbf{x}_{k-1}^B, \mathbf{P}_{k-1}^B, \mathbf{x}_{R_k}^{R_{k-1}}, \mathbf{Q}_k)$ {Predicción EKF}
$[\mathbf{z}_k, \mathbf{R}_k]$ = Obtener_Movimiento
$\mathcal{H}_k$ = Asociación_de_datos $(\mathbf{x}_{k|k-1}^B, \mathbf{P}_{k|k-1}^B, \mathbf{z}_k, \mathbf{R}_k)$
$[\mathbf{x}_k^B, \mathbf{P}_k^B]$ = Actualizar_Mapa $(\mathbf{x}_{k|k-1}^B, \mathbf{P}_{k|k-1}^B, \mathbf{z}_k, \mathbf{R}_k, \mathcal{H}_k)$ {Actualizar EKF}
$[\mathbf{x}_k^B, \mathbf{P}_k^B]$ = Nuevas_características $(\mathbf{x}_k^B, \mathbf{P}_k^B, \mathbf{z}_k, \mathbf{R}_k, \mathcal{H}_k)$
FIN PARA

[Algoritmo02]

Esta solución EKF-SLAM es muy conocida y hereda muchos de los mismos beneficios y problemas que las soluciones del EKF estándar para la navegación o para el seguimiento de problemas. Cuatro de los principales problemas se discuten brevemente a continuación.

- **Convergencia:** En el problema EKF-SLAM, la convergencia del mapa se manifiesta en la convergencia monótona del determinante de la matriz de covarianza del mapa y todas las submatrices de las marcas, hacia cero. Las variaciones individuales de las marcas convergen hacia un límite inferior determinado por las incertidumbres iníciales en la posición del robot y observaciones.
- **Esfuerzo computacional:** El paso de actualización de observación requiere que todos los puntos de referencia y la matriz de covarianza se actualicen cada vez que se realiza una observación. Pero esto significa que la complejidad de este cálculo crezca cuadráticamente con el número de puntos de referencia. Ha habido una gran cantidad de trabajo realizado en el desarrollo de variantes eficientes de las implementaciones EKF-SLAM en tiempo real y con miles de puntos de referencia.
- **Asociación de datos:** La formulación estándar de la solución EKF-SLAM es especialmente frágil a la asociación incorrecta de las observaciones de las marcas o puntos de referencia. El problema de cierre del circuito, se da cuando el robot vuelve a observar puntos de referencia después de un largo tiempo, el problema se agrava cuando la marcar (puntos de referencia) no son simples puntos y en realidad se ven diferentes desde distintos puntos de vista.

- **No linealidad:** La no linealidad puede ser un problema significativo en EKF-SLAM y conduce a la inconsistencia inevitable, ya que emplea modelos linealizados de movimiento no lineal y modelos de observación. La convergencia y la coherencia sólo se puede garantizarse en el caso lineal.

4.4. Otros métodos de Mapeo

4.4.1. Mapeo Topológico

La tarea de generar un mapa topológico a partir de datos de vídeo ha ganado popularidad en los últimos años.

Representaciones topológicas de las rutas que abarcan varios kilómetros son más robustas que las métricas y cognitivamente más sencillas de utilizar para el uso humano. Son generalmente utilizados para realizar la planificación de la ruta del robot, proporcionando puntos de interés, y la definición de la accesibilidad de los distintos lugares del mapa.

Los Mapas topológicos pueden corregir la planificación y el camino que recorre el robot en sistemas de odometría visual y pueden ser parte de representaciones híbridas, donde se representa el medio ambiente local de forma métrica pero de forma topológica a nivel global[web23].

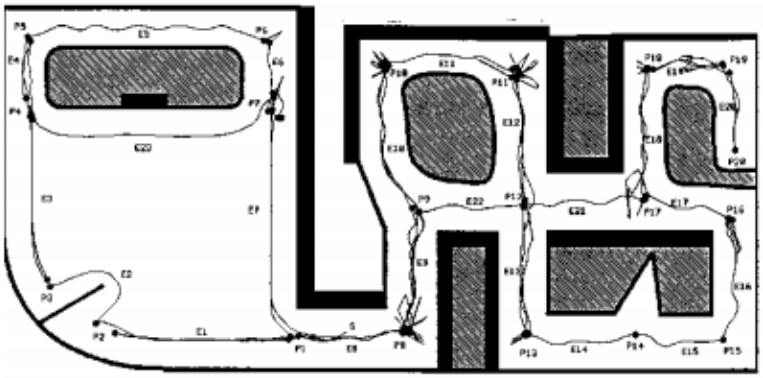


Imagen 17

Un mapa topológico T puede ser definido formalmente como un gráfico $T = (K, E_T)$, donde K es un conjunto de fotogramas clave (que son los representantes de las ubicaciones) y los bordes E_T describen conectividad entre fotogramas clave. Es deseable que T posea las siguientes propiedades:

- **Cierre de ciclos:** Para cualquier par de ubicaciones $i, j \in K$, E_T contiene el borde (i, j) si, y sólo si es posible llegar a la ubicación j desde la ubicación i sin pasar por ningún otro lugar $k \in K$.

- **Compacidad:** Dos imágenes tomadas en el "mismo lugar" deben estar representados por el mismo fotograma clave.
- **Diferenciación espacial:** Dos imágenes de "lugares diferentes" no puede ser representado por el mismo fotograma clave[web24].

La matriz de decisión D se utiliza para definir los fotogramas clave K y determinar el mapa de conectividad E_T. D puede ser vista como una matriz de adyacencia de un grafo no dirigido. Puesto que no hay garantía de que D se ha encontrado a través de la propagación de creencia (lo cual generaría una matriz simétrica), lo que inicialmente se suele hacer es tratar a D como una matriz de adyacencia de un grafo dirigido, y luego quitar la dirección de todos los bordes resultando en un grafo simétrico $D' = D \vee D^T$.

Es posible usar el gráfico definido por D' como un mapa topológico. Sin embargo, esta representación es prácticamente inútil porque varios nodos representan la misma ubicación. Para lograr un diseño compacto, es necesario simplificar D' sin que deje de ser fiel a la estructura general del medio ambiente. Al realizar esta simplificación (obteniendo el conjunto mínimo dominante de D') nos encontramos con los fotogramas clave K. Encontrar la solución óptima es NP-completo, sin embargo el algoritmo03 presenta una buena aproximación[web25].

```
Entrada: Matriz de adyacencia D'
Salida: K,{Nk:kЄK}
K←0
MIENTRAS D' no_vacio HACER
   k←Nodo con mayor grado
   K←K U {k}
   Nk←{k} U Nbk
   Remover todos los Nk de la matriz D'
FIN MIENTRAS
```

algoritmo03: Mínimo aproximado del conjunto dominante

4.4.1.1. Tipos de Mapeo Topológico

Existen varios algoritmos para realizar mapeos topológicos en ambientes interiores. Estos mapas representan la conectividad entre los espacios del medio ambiente, por lo general se representa como una estructura de grafo donde los vértices son "lugares distintivos" en el medio ambiente y las aristas representan los caminos, los cuales se utilizan para viajar entre los lugares.

- **Mapeo topológico simple de un solo robot**

Aunque no siempre, muchos mapas topológicos se pueden aumentar con alguna información métrica, tal como las longitudes de los caminos (basado en la odometría del robot). Es importante que el robot mantenga las estimaciones de longitud de ruta y utilice un modelo de error de odometría para mantener los límites de confianza de las estimaciones[web26].

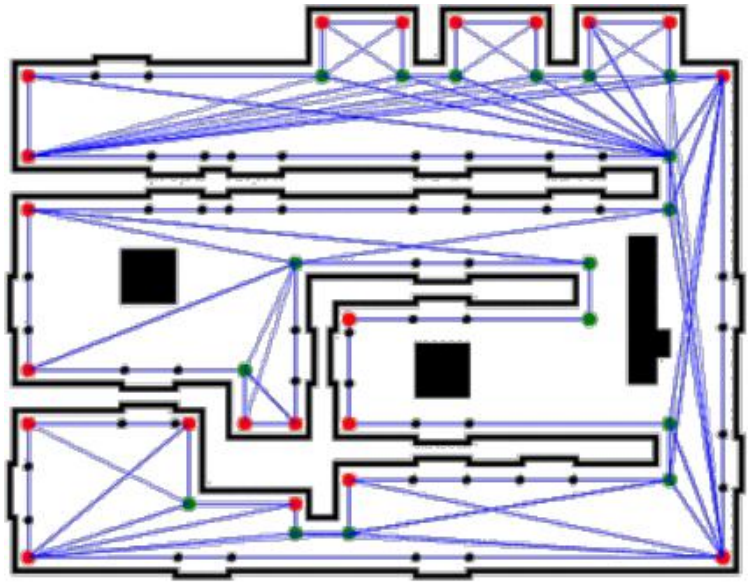


Imagen 18

Los mapas topológicos en general tienen dificultades para representar "espacios abiertos", es decir, grandes extensiones con pocos puntos de referencia. Este problema puede ser abordado de dos maneras. En primer lugar, introduciendo la idea de "portales": lugares en los que una transición entre un espacio abierto y un espacio cerrado se produce. En espacios cerrados (por lo general los corredores en los que el robot es capaz de ver ambos lados del pasillo). En segundo lugar, se desarrolla la idea de "refinamientos" a un mapa creado con la pared y la sala siguiente. En los espacios abiertos, los robots hacen "incursiones" a través de las áreas vacías, añadiendo conexiones entre puntos situados en el mapa original cuando sea posible. Esto puede mejorar significativamente la eficiencia de la navegación en zonas abiertas.

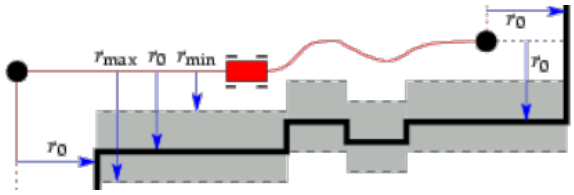


Imagen 19

- **Mapas topológicos con cierre de bucles**

Un problema difícil en la cartografía es "cerrar el círculo": reconocer cuando el robot ha regresado a un lugar que ya ha estado. Cuando la estimación del mapa indica que el robot pudo haber vuelto a un lugar previamente visitado, se "presume" que ha cerrado el ciclo. Luego continúa atravesando el medio ambiente en búsqueda de "evidencias" (por lo general mediante la adopción de medidas de longitud de pared) que apoya o refuta su hipótesis. Finalmente, cuando el robot construye evidencia suficiente para aceptar o rechazar una hipótesis particular, asume que su mapa es coherente[web26].

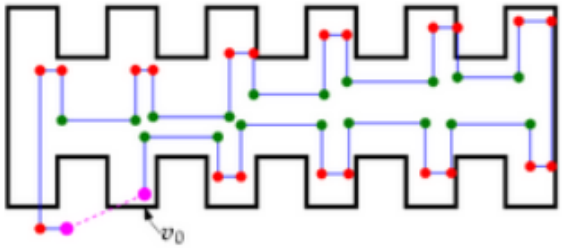


Imagen 20

- **Mapeo topológico con SGVD**

Otro método es una representación mapa topológico basado en una versión generalizada del diagrama de Voronoi (GVD, por sus siglas en Ingles) del medio ambiente.

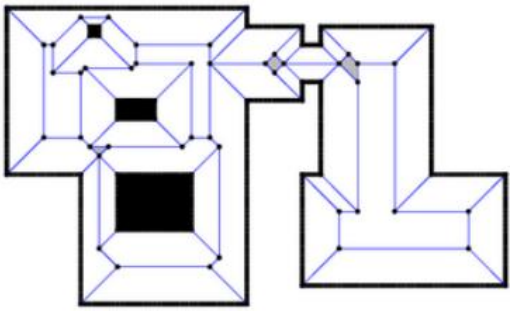


Imagen 21

La forma estándar del GVD básico como un mapa topológico, con puntos en el GVD como nodos en el mapa, y los bordes de la GVD como aristas. Con un sensor de comportamiento omnidireccional se pueden desarrollar el GVD y construir el mapa[web26].

- **Fusión de mapas topológicos**

Cuando múltiples robots exploran el entorno en paralelo, es útil para combinar sus mapas para obtener un mapa global. Este es un problema difícil cuando los mapas carecen de un marco de referencia común.

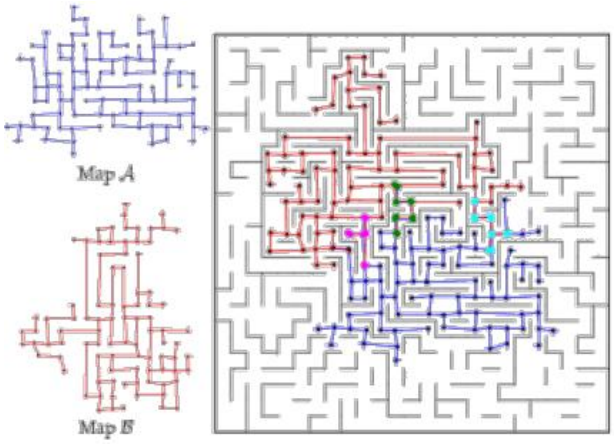


Imagen 22

Un enfoque que podemos utilizar para combinar mapas topológicos utiliza aspectos de los métodos de comparación para determinar el subgrafo "estructural" entre los grafos del mapa. Durante esta fase del algoritmo, los vértices y aristas en cada mapa se comparan de acuerdo a sus características, las características tanto absolutas, como el grado de vértices, y las características aproximadas, tales como longitudes de la trayectoria de medición. La adaptación estructural típicamente reduce las posibles coincidencias entre los mapas y las lleva a un número manejable.

Después de encontrar coincidencias estructuralmente consistentes entre los dos mapas, utilizamos técnicas de los métodos de registro de imágenes para encontrar la mejor transformación para cada coincidencia. Las coincidencias son entonces agrupadas en el espacio transformación en grupos coherentes. El mejor grupo (generalmente el que tiene la mayor cantidad de vértices y el menor error) se devuelve como la unión de las hipótesis entre los mapas[web26].

4.4.2. Mapeo Métrico

Las características utilizadas para la representación métrica del ambiente son líneas infinitas. Estos son menos informativos que los segmentos de línea, pero tienen un mejor modelo probabilístico con solución analítica y permiten una representación muy compacta de entornos estructurados geométricamente.

El mapeo métrico, que se basa en el mismo marco estadístico que el mapeo topológico, utiliza mediciones de proximidad (escaneos de rango) obtenidos con un láser para la construcción del mapa métrico. Su modelo perceptual se define a través de un método de comparaciones geométricos, el cual determina la probabilidad de exploración láser basándose en la proximidad de obstáculos percibidos en coordenadas x,y.

Este método representa todas las densidades (posición, mapas, modelos de movimiento, y el modelo perceptual) a través de una distribución Gaussiana. Las distribuciones Gaussianas tienen una doble ventaja: en primer lugar permiten determinar la posición del robot y la ubicación de obstáculos con punto flotante de resolución, produciendo mapas de alta resolución. En segundo lugar, hacen posible la aplicación de métodos lineales altamente eficientes de programación al maximizar la función de probabilidad[web27].

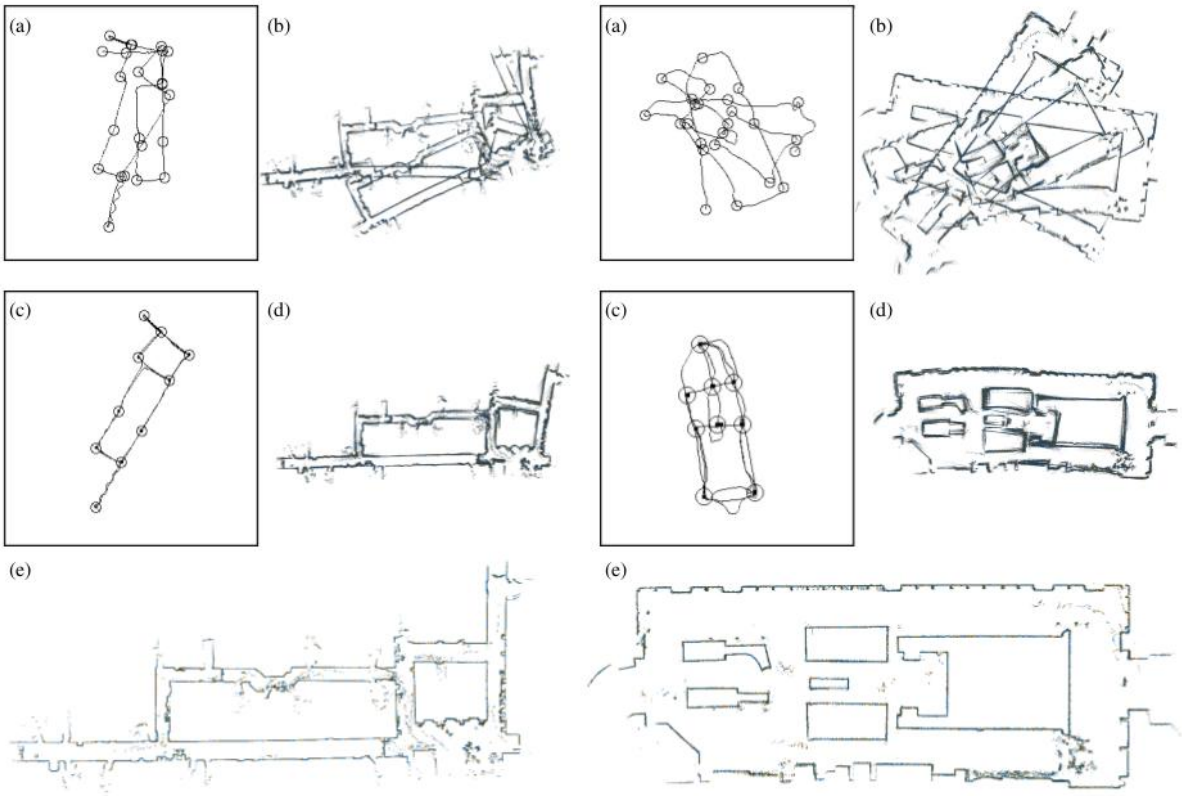


Imagen 23 Imagen 24

(a) Boceto inicial topológico, (b) los datos de sensores sin procesar, (c) mapa topológico, (d) correspondiente mapa métrico inicial, y (e) mapa métrico final.

Técnicamente, el mapa métrico se construye como una red de relaciones espaciales entre todas las posiciones que el sensor ha explorado en el área de estudio. Las limitaciones espaciales probabilísticas entre las pociones se derivan de las comparaciones entre los escaneos del laser y las mediciones de odometría.

Con el fin de evaluar el uso de estas representaciones, se debe evaluar lo bien que el robot puede identificar su submapa actual, basado en una sola lectura de un láser, es decir en

una sola exploración (esto es similar al problema del robot secuestrado, es efectivamente evaluar el error del conjunto de pruebas de los clasificadores del submapa) y también clasificar el submapa del robot.

Si bien ambos métodos presentados en esta sección (topológico y métrico) son completamente diferentes por la naturaleza a partir de la cual son construidos, pero a su vez se trata de métodos complementarios, característica por la cual es en general muy usual que en la actualidad ambos sean utilizados, aprovechando las mejores características de cada uno. Por ejemplo, en lugares donde tenemos múltiples configuraciones, como ser un ambiente con espacios abiertos y a sus vez con habitaciones más pequeñas en otra parte, la combinación de estos métodos hace que podamos lograr obtener un mapa correcto y eficaz, ya que en general en los ambientes donde el topológico falla, el métrico logra ser una buena implementación.

5. Planificación

La planificación es fundamental en la robótica, el problema de planificación de trayectoria clásica se describe de la siguiente manera: dado un cuerpo tridimensional rígido y un conjunto conocido de obstáculos, la tarea es encontrar un camino libre de colisiones desde una configuración inicial hasta el objetivo final. Además, esta tarea debe ser completada en un plazo razonable de tiempo. Esto se conoce como el problema del transportador del piano. Un escenario más general, conocido como el problema del transportador, define al robot como un poliedro flexible con obstáculos poliédricos en movimiento. La planificación de movimientos del robot abarca varias disciplinas, especialmente la robótica, la informática, teoría de control y las matemáticas.

El problema de planificación es generalmente dividido en dos, Planificación de movimiento y planificación de tareas.

5.1. Planificación de movimiento

Con el fin de lograr las tareas para las cuales fueron construidos, los robots autónomos tienen que ser inteligentes y deben decidir su propia acción. Cuando el robot autónomo decide su acción, es necesario planificar un camino libre de colisiones minimizando el costo del mismo (tiempo, energía y distancia). Cuando un robot autónomo se mueve de un punto A a un punto de destino en su entorno determinado, es necesario planificar una ruta óptima y factible evitando obstáculos en su camino y respondiendo a algún criterio de los requisitos de autonomía como: el tiempo, la energía térmica, y la seguridad, por ejemplo. Por lo tanto, la principal tarea para la planificación de ruta para el robot móvil autónomo es buscar un camino libre de colisiones[web29]. Este es uno de los ejes fundamentales de la robótica, decidir qué movimientos el robot debe realizar con el fin de lograr el objetivo propuesto. Este resulta ser un problema muy difícil. El problema básico de planificación de movimiento se puede definir de la siguiente manera:

Sea A un solo objeto rígido (el robot) que se mueve en un espacio W, llamado el espacio de trabajo, representada como R^n (donde $n = 2\ o\ 3$). Siendo los obstáculos objetos rígidos $B_1 \ldots B_q$ distribuidos en W. Asumiendo que tanto la geometría de A, como las geometrías y las ubicaciones de los B_i se conocen con precisión y suponiendo también que no hay restricciones

cinemáticas que limitan los movimientos de A (de forma que A es un objeto de movimiento libre).

Dada una posición inicial y una orientación, una posición objetivo y la orientación de A en W, el problema se reduce a generar un camino t especificando una secuencia continua de posiciones y orientaciones para lograr llegar al objetivo planteado tratando de evitar el contacto con los B_i. (Básicamente, dado un robot, un conjunto de objetos, un estado inicial y un estado final, debemos encontrar un camino para que el robot pueda llegar al estado final)[web28].

Para comenzar, debemos definir tres conceptos claves en planificación:

- **Espacio de trabajo:** Medio ambiente en el que opera robot
- **Obstáculos:** Espacios ya ocupados del mundo.
- **Espacio libre:** espacio desocupado del mundo

El Espacio de trabajo (W) puede ser definido de dos formas diferentes, dependiendo de las necesidades del entorno del robot:

- Un mundo en 2D, en el cual $W = \mathbb{R}^2$
    - Por Ejemplo: Robots rodantes en tierra, navegantes en la superficie del agua y caminantes.
- Un mundo en 3D, en el cual $W = \mathbb{R}^3$
    - Por Ejemplo: Robots voladores, en el espacio o bajo el agua.

El espacio de trabajo solo contiene dos clases de entidades:

- **Obstáculos:** Se trata de porciones del mundo (W) que se encuentran permanentemente ocupadas. El conjunto de estas regiones se denota como un conjunto de puntos (ϑ) los cuales están incluidos en W, por lo tando $\vartheta \subseteq W$ [web29].

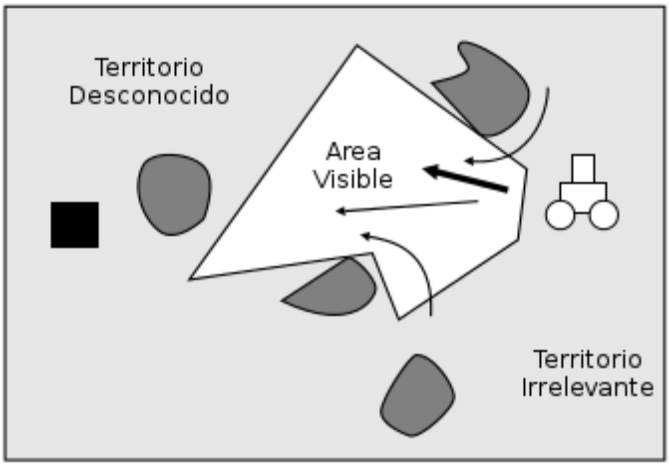


Imagen 25

- **Robots:** Son los cuerpos modelados geométricamente con cierta cantidad de grados de movimiento y los cuales son controlados a través del plan de movimiento.

Un tema central en toda la planificación de movimientos es transformar el modelo continuo en uno discreto. Debido a esta transformación, muchos algoritmos están integrados en los algoritmos de planificación de movimientos. Pero antes de definir estos métodos debemos definir los siguientes conceptos:

- **Grados de libertad:** Los grados de libertad, en un contexto de la mecánica, son modos específicos, definidos en el cual un dispositivo mecánico o sistema puede moverse.

  El número de grados de libertad es igual al número total de desplazamientos o aspectos de movimiento independientes. Una máquina puede funcionar en dos o tres dimensiones, pero debe tener más de tres grados de libertad. El término se usa ampliamente para definir las capacidades de movimiento de los robots. Considere la posibilidad de un brazo robótico construido para funcionar como un brazo humano. El movimiento del hombro puede ocurrir de arriba a abajo o de izquierda a derecha. El movimiento del codo puede ocurrir sólo de arriba a abajo. El movimiento de muñeca se puede producir de arriba a abajo o de izquierda a derecha. La rotación también puede ser posible para la muñeca y el hombro.

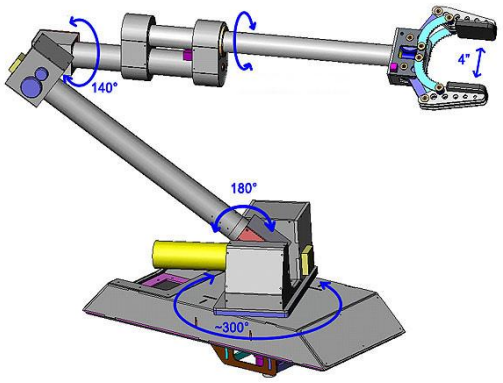


Imagen 27

  Tal un brazo de robot tiene entre cinco y siete grados de libertad. Si un robot complejo dispone de dos brazos, se duplicó el número total de grados de libertad. En un androide, grados de libertad también existen en las piernas, la cabeza, la cintura, la columna, etc.[web33].

- **Espacio de configuración (Espacio-C):** Una configuración describe la postura del robot, y el espacio de configuración C es el conjunto de todas las configuraciones posibles. En la mecánica clásica, los parámetros que definen la configuración de un sistema se denominan coordenadas generalizadas, y el espacio vectorial definido por estas coordenadas se llama el espacio de configuración del sistema físico (espacio-c) [web32]. Por ejemplo:
    - Si consideramos al robot como un punto singular, en un espacio bidimensional, el espacio-c es un plano, y es definido por dos parámetros, las coordenadas (x,y).
    - Si el robot es una forma bidimensional, que puede ser trasladado y rotado y el espacio también es bidimensional, el espacio-c es un grupo Euclediano particular, llamado grupo ortogonal, y es definido por tres parámetros, las coordenadas (x,y,θ).
    - Si el robot es una forma tridimensional solida, el espacio debe ser también tridimensional, y el espacio-c requiere seis parámetros, (x,y,z) para la traslación y los ángulos (α,β,γ) para la rotación.
- **Espacio Libre:** El conjunto de configuraciones que evita la colisión con obstáculos se llama espacio libre. El complemento de este espacio en el espacio-C se llama el obstáculo o región prohibida.

  A menudo, es prohibitivamente difícil de calcular explícitamente la forma del espacio libre. Sin embargo, probando si una determinada configuración está en este espacio es eficiente. En primer lugar, la cinemática directa determina la posición y la geometría del robot, y las pruebas de detección de colisiones determinan si choca la geometría del robot con la geometría del medio ambiente.

Hay dos alternativas para lograr esta transformación:

Planificación del movimiento basado en muestreo, se refiere a los algoritmos que utilizan métodos de detección de colisiones para muestrear el espacio de configuración y llevar a cabo búsquedas discretas que utilizan estas muestras. En este caso, la integridad se sacrifica, pero a menudo se sustituye con mejoras en otros aspectos tal como resolución completa o integridad probabilística[web30].

La filosofía de la planificación basada en muestreo utiliza la detección de colisiones como una "caja negra" que separa la planificación de movimientos de los modelos geométricos y cinemáticas particulares.

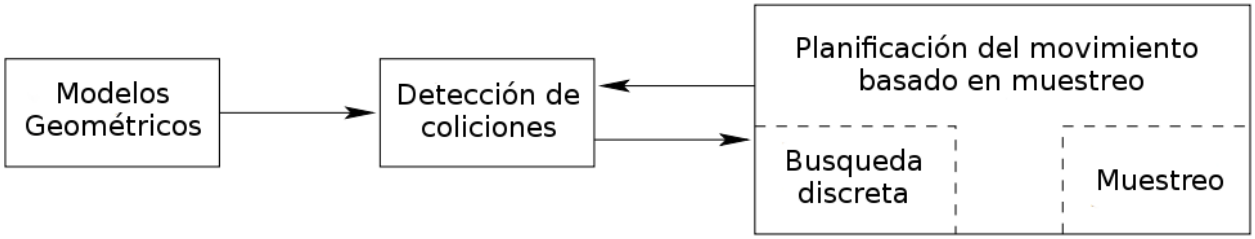


Imagen 26

Como todos los métodos para resolver problemas, existen pros y contras que son los que debemos evaluar a la hora de elegir el método más adecuado:

- Pros:
  - Probabilísticamente completo
  - No es necesario construir el espacio-C
  - Se aplica fácilmente a espacios-C con altas dimensiones
  - Admite consultas rápidas con suficientemente procesamiento.
- Contras
  - no funcionan tan bien para algunos problemas:
    - Poco probable que el método de los muestreos de los nodos sea correcto en espacios estrechos
    - Difícil de lograr un muestreo correcto / conectar los nodos en las superficies con restricciones
  - Puede no ser optimo o no contener integridad completa

El muestreo aleatorio es el más fácil de todos los métodos de toma de muestras para aplicar a espacios C. Una de las razones principales es que el espacio C se forma a partir de productos cartesianos, y las muestras aleatorias independientes se extienden fácilmente a través de estos productos. Si $X = X_1 \times X_2$, y muestras aleatorias uniformes x_1 y x_2 se toman de X_1 y X_2, respectivamente, a continuación, (x_1, x_2) es una muestra aleatoria uniforme para X.

Esto es muy conveniente en implementaciones. Por ejemplo, supongamos que el problema de la planificación de movimiento implica 15 robots que traducen para cualquier $(x_t, y_t) \in [0,1]^2$; esto da $C = [0,1]^{30}$. En este caso, 30 puntos pueden ser elegidos de manera uniforme al azar de $[0,1]$ y se combinan en un vector de 30 dimensiones. Las muestras generadas de esta manera están uniformemente distribuidas al azar en C [web34]. Por lo tanto, es importante darse cuenta de que incluso las muestras "al azar" son deterministas. Están diseñadas para optimizar el rendimiento en las pruebas estadísticas. Se utilizan muchas pruebas estadísticas sofisticadas de aleatoriedad uniforme. Uno de las más sencillos, la prueba de chi-cuadrado, mide hasta qué punto las estadísticas calculadas son de su valor esperado. Como un simple ejemplo, supongamos que $C = [0,1]^2$ y se divide en un arreglo de 10 por 10 de 100 cajas cuadradas. Si un conjunto P de k muestras se elige al azar, entonces intuitivamente cada caja debe recibir aproximadamente $K/100$ de las muestras. Una función de error puede ser definida para medir qué tan lejos de ser cierto esta intuición es:

$$e(P) = \sum_{i=1}^{100} (b_i - k/100)^2$$

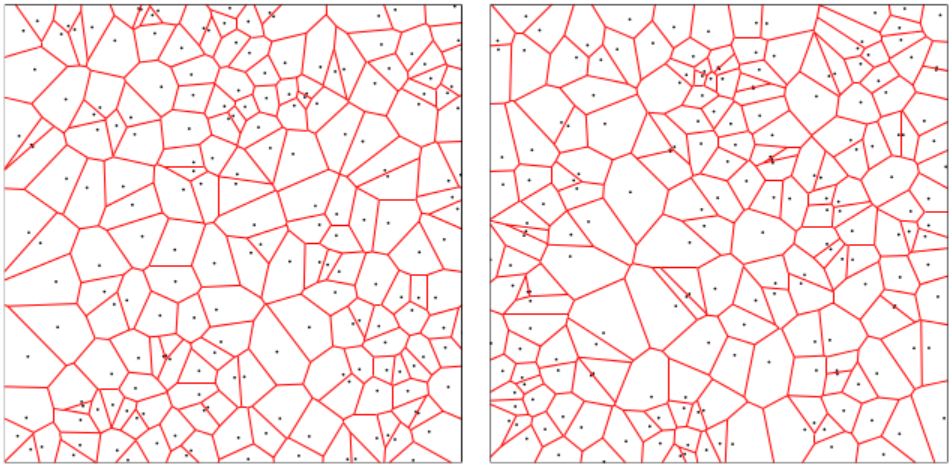


Imagen 28: 196 Muestras pseudo aleatorias

Una vez que se ha decidido el método de muestreo, el siguiente problema es determinar si la configuración elegida entra en colisión con los obstáculos. Por lo tanto, la detección de colisiones es un componente crítico de la planificación basada en muestreo. A pesar de que a menudo se trata como una caja negra, es importante estudiar su funcionamiento interno para comprender la información que proporciona y su costo computacional asociado. En la mayoría de las aplicaciones de planificación de movimiento, la mayoría de tiempo de cálculo se gasta en la comprobación de colisión.

Una variedad de algoritmos de detección de colisiones existe, que van desde algoritmos teóricos que tienen excelente complejidad computacional a, algoritmos heurísticos prácticos cuyo rendimiento está adaptado a una aplicación particular. Existen técnicas que pueden ser utilizadas para desarrollar un algoritmo de detección de colisiones mediante la definición de un predicado lógico utilizando un modelo geométrico. Para el caso de un mundo 2D con un robot convexo y un obstáculo, esto conduce a un algoritmo de detección de colisiones de tiempo lineal. En general, sin embargo, se puede determinar si una configuración es más eficiente en colisión al evitar la construcción completa.

Planificación por combinatoria de movimiento, significa que a partir del modelo de entrada los algoritmos se construyen para generar una representación discreta que representa exactamente el problema original. Esto nos lleva a completar los enfoques de planificación, que garantizan encontrar una solución cuando existe, o informan correctamente el fracaso si no existe uno.

Existen varias razones para estudiar enfoques combinatorios al momento de realizar la planificación de movimiento:

- Muchas veces, cuando empezamos a realizar la planificación de un robot, sólo estamos interesados en una clase especial de problema de planificación, como por ejemplo, el mundo podría ser en 2D, y el robot sólo puede ser capaz de trasladarse. Para muchos de estos casos especiales se pueden desarrollar algoritmos elegantes y eficientes, estos algoritmos suelen ser completos y no dependen de la aproximación, por lo cual pueden ofrecer un rendimiento mucho mejor que los métodos de planificación basados en muestreo vistos anteriormente.
- Es a la vez interesante y satisfactorio saber que existen esta clase de algoritmos completos para resolver una gran cantidad de problemas de planificación de movimientos. Por lo tanto, incluso si la clase de planificación que estamos realizando no tiene ningún algoritmo específico realizado, todavía existen herramientas y algoritmos de propósito general que puede resolverlo. Estos algoritmos también proporcionan límites teóricos en el tiempo máximo que es necesario para resolver los problemas de planificación de movimientos.

Al estudiar los algoritmos de planificación de movimientos combinatorios, es importante considerar cuidadosamente la definición de la entrada. ¿Cuál es la representación utilizada para

el robot y los obstáculos? ¿Qué conjunto de transformaciones se puede aplicar al robot? ¿Cuál es la dimensión del mundo? ¿Son los robots y los obstáculos convexos? La especificación de las posibles entradas define un conjunto de instancias de problemas en la que va a operar el algoritmo. Si las instancias tienen ciertas propiedades convenientes, por ejemplo, baja dimensionalidad, modelos convexos, un algoritmo combinatorio puede proporcionar una solución elegante y práctica. Si el conjunto de instancias es demasiado amplio, una solución que provea ambas integridad y practicidad puede no ser razonable. Sin embargo, existen algoritmos de planificación de movimientos generales y completos[web35].

Debemos tener en cuenta que centrarse en la representación es la filosofía opuesta a la planificación basada en muestreo, que oculta estos temas en el módulo de detección de colisiones.

Supongamos que $W = \mathbb{R}^2$, los obstáculos, $\mathcal{O}$, son poligonales, y el robot, A, es un cuerpo poligonal que sólo es capaz de trasladarse. Bajo estos supuestos, el espacio será poligonal. Para el caso especial en el que A es un punto en el W, $\mathcal{O}$ es asignado directamente sin distorsión alguna. Por lo tanto, los problemas considerados en esta sección también pueden ser considerados como la planificación para un robot puntual (robot teórico).

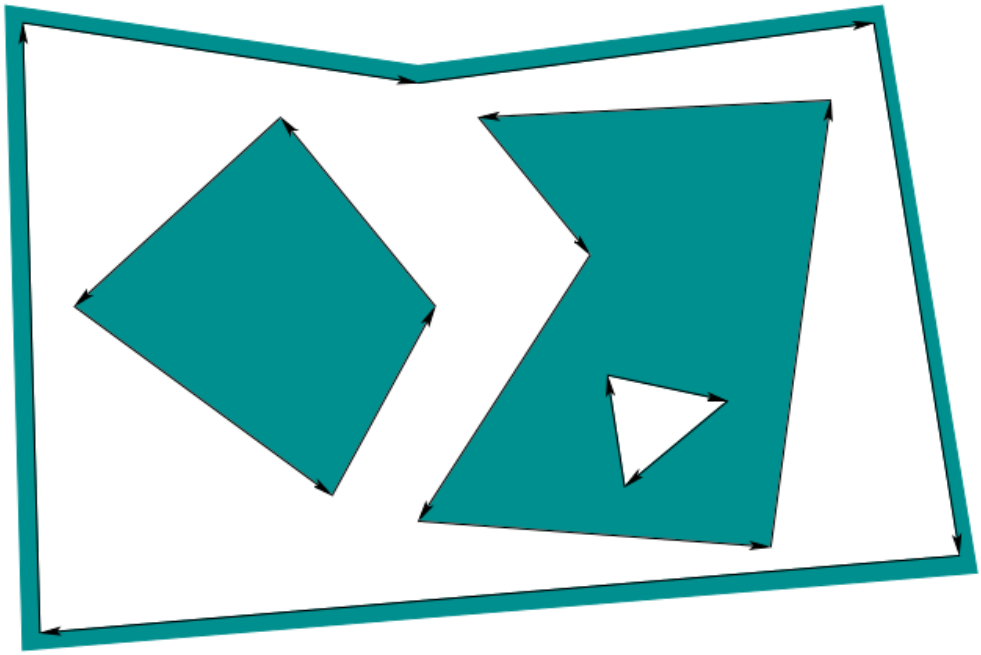


Imagen 29: Un modelo poligonal especificado por cuatro polígonos simples orientados.

Para concluir estas comparaciones, los métodos combinatorios pueden resolver prácticamente cualquier problema, sin embargo, para la mayoría de los problemas de planificación de movimiento "de tipo industrial", los tiempos de funcionamiento y dificultades de implementación de estos algoritmos los hacen poco atractivos. Algoritmos basados en muestreo han cumplido gran parte de esta necesidad en los últimos años mediante la resolución de problemas difíciles en varios ámbitos, como el montaje de automóviles, la planificación robots humanoides, y en el análisis, diseño y producción de fármacos. Aunque la integridad garantizada es más débil, la eficiencia y la facilidad de aplicación de estos métodos han reforzado interés en la utilización de estos algoritmos de planificación de movimiento para una amplia variedad de aplicaciones[web31].

5.2. Planificación de tareas

El razonamiento de alto nivel y la planificación de tareas son esenciales si queremos lograr que los robots realicen autónomamente tareas de alto nivel (por ejemplo, "Tráeme una taza de café"). Se han desarrollado una gran cantidad de algoritmos de planificación que, en principio, son lo suficientemente eficaces para resolver problemas complejos de planificación en tiempo real. Sin embargo, varios enfoques de planificación de IA se basan todavía en supuestos demasiados simplificadores. Por lo tanto, es en muchos casos difícil de aplicar estos enfoques a la robótica del mundo real. En particular, el hecho de que en escenarios de la vida real a menudo no toda la información necesaria está disponible al comienzo del proceso de planificación, hace que sea difícil generar un plan completo a priori como es necesario en la mayoría de los enfoques de planificación de IA. Desafortunadamente la mayoría de los enfoques anteriores, que son capaces de generar planes en entornos parcialmente conocidos logran generar planes condicionales para todas las contingencias posibles pero son computacionalmente difíciles, en general no son buenos en dominios dinámicos y sólo son aplicables si es posible prever todos los posibles resultados de un proceso de adquisición de conocimientos[web36].

La introducción de los robots en nuestra vida diaria plantea una cuestión clave que se añade al desafío estándar de los robots autónomos: la presencia de seres humanos en el medio ambiente y la necesidad de interactuar con ellos. Claramente, el ser humano se debe tomar en cuenta explícitamente en todos los pasos del diseño del robot. Actualmente se están llevando a cabo investigaciones sobre las capacidades de los robots al momento de tomar decisiones teniendo en cuenta la interacción el entorno humano y en la capacidad del robot para lograr sus tareas en ese contexto.

La planificación de tareas para los robots móviles por lo general se basa únicamente en la información espacial y el conocimiento de un dominio superficial, como etiquetas adheridas a los objetos y lugares. Aunque la información espacial es necesaria para la realización de las operaciones básicas del robot (navegación, localización, y evasión de obstáculos), el conocimiento profundo del dominio es fundamental para dotar a un robot con un mayor grado de autonomía e inteligencia, centrándose en un profundo conocimiento semántico, y mostrando cómo este tipo de conocimiento puede ser utilizado provechosamente para la planificación de las tareas del robot. Estos incluyen el conocimiento causal, que es el conocimiento sobre los efectos de las acciones del robot, y el conocimiento del mundo, es el conocimiento acerca de los objetos en el mundo, sus propiedades y sus relaciones. Por ejemplo, dado a la tarea de buscar una botella de leche, un planificador de tarea podría producir una secuencia de acciones como " Ir a la cocina, llegar a la heladera, abrir la heladera, perceptivamente encontrar la botella de leche, agarrar y extraer la botella, cerrar la heladera, volver con la botella". Para ello, el planificador necesita saber, por ejemplo, que la leche se almacena en una heladera y que esta está en la cocina[web37].

El conocimiento sobre la estructura y el estado actual del mundo se codifica generalmente en la forma de un mapa. El problema de cómo representar, construir y mantener un mapa para el robot ha sido una de las áreas más activas de la investigación en robótica en las últimas dos décadas, y soluciones de gran valor a este problema ya están disponibles. Sin embargo, la mayor parte de este trabajo se ha centrado en las representaciones de la estructura espacial del medio ambiente, como los mapas métricos y mapas topológicos. Este tipo de mapas son necesarios a nivel de la planificación y ejecución de la navegación, pero no contienen los tipos más cualitativos de información necesaria para llevar a cabo la planificación de tareas. Por ejemplo, un mapa métrico puede representar la forma de una habitación, pero no indica si esta habitación es una oficina, una cocina o un dormitorio - de hecho, ni siquiera indican que la forma dada es una habitación. En la mayoría de los casos prácticos, este tipo de conocimiento, que llamamos semántica, es codificado como parte de la descripción del dominio del planificador de tareas utilizando un lenguaje ad-hoc.

En la actualidad la mayoría de las investigaciones activas de inteligencia artificial se concentra en encontrar solución a este problema, intentando lograr robots que sean capaces de realizar la planificación de un camino de manera eficaz, minimizando los recursos utilizados, minimizando la interacción humana, y a su vez aprendiendo a medida que realiza esta tarea

para luego reutilizar ese conocimiento adquirido y aplicarlo en nuevos ambientes con nuevos problemas. Las variadas tecnologías de inteligencia artificial nos proveen de herramientas muy importantes para lograr esto, sin embargo aun son muy limitadas y solo aplicables para problemas particulares.

6. Visión Robótica

La visión robótica se refiere a la modelización de la visión humana utilizando software y hardware. Combina los conocimientos en ciencias de la computación, ingeniería eléctrica, las matemáticas, la fisiología, la biología y las ciencias cognitivas. Se necesitan conocimientos de todos estos campos con el fin de entender y simular el funcionamiento del sistema de visión humana. Por otro lado, la visión computacional (interpretación de imágenes) es una disciplina que estudia cómo reconstruir, interpretar y comprender una escena 3D a partir de sus imágenes 2D en términos de las propiedades de las estructuras presentes en la escena.

La visión artificial se Superpone significativamente con los siguientes campos: el procesamiento de imágenes, reconocimiento de patrones, y fotogrametría.

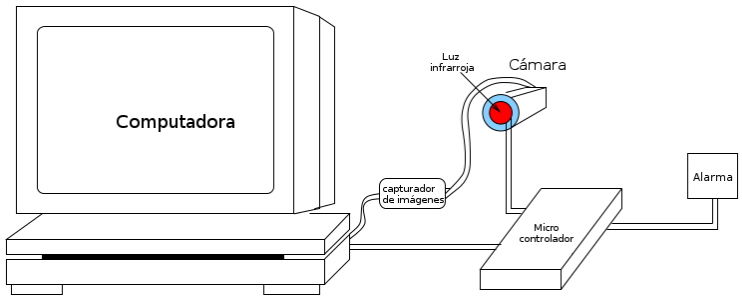


Imagen 30

El procesamiento de imágenes se centra en la manipulación de imágenes para mejorar la calidad de la imagen, para restaurar una imagen o para comprimir / descomprimir una imagen. La mayoría de los algoritmos de visión por lo general asumen una gran cantidad de procesamiento de imágenes que se llevan a cabo para mejorar la calidad de la imagen.

Estudios de reconocimiento de patrones utilizando diferentes técnicas (como las técnicas estadísticas, la red neuronal, máquina de vectores de soporte, etc.) para reconocer / clasificar los diferentes patrones. Las técnicas de reconocimiento de patrones se utilizan ampliamente en la visión por computadora. La fotogrametría se refiere a la obtención de mediciones precisas y fiables a partir de imágenes. Se centra en la medición precisa. La calibración de la cámara y la reconstrucción 3D son dos áreas de interés para ambas, la visión por computadora y los investigadores de fotogrametría.

6.1. Sistemas de visión Robótica

Actualmente hay muchos robots que se están desarrollando o han sido desarrollados. La robótica es un área de interés muy popular para muchos investigadores e ingenieros y al mismo tiempo la aplicación de los robots autónomos está pasando de los campos de fabricación e inspección hacia el sector de los servicios. En el sector de los servicios, hay muchas tareas en las que los robots autónomos se pueden aplicar, desde tareas en la medicina, la cocina, el hogar, etc. Esto se debe a que hay una gran variedad de servicios y tareas que son desagradables o peligrosas para los humanos, pero son adecuadas para los robots. Como tal, hay actualmente un gran interés y la investigación sobre los diferentes métodos de visión por computador.

Los métodos de visión por computadora están muy bien adaptados para satisfacer los objetivos de cada robot, hay varias adaptaciones para cada sistema de visión. Los métodos de procesamiento de imágenes utilizados para cada uno pueden diferir, pero los objetivos de los sistemas de visión por lo general coinciden.

6.1.1. Técnicas de procesamiento de imágenes para visión artificial

6.1.1.1. Técnicas de segmentación por niveles de gris

La segmentación por niveles de gris (o Thresholding, umbral) es un concepto esencial relacionado con el procesamiento de imágenes y visión artificial.

Para lograr la segmentación de una imagen, se debe particionar la misma en regiones distintas que contienen cada una, píxeles con atributos similares. Para ser significativo y útil para el análisis e interpretación de imágenes, las regiones deben relacionarse fuertemente a los objetos o características de interés representados [web38].

La segmentación significativa es el primer paso del proceso de transformación de bajo nivel de una imagen en escala de grises o en color en una o varias imágenes con descripciones de alto nivel en términos de características, objetos y escenas. El éxito del análisis de la imagen depende de la fiabilidad de la segmentación, pero un particionamiento correcto de imagen es vital para lograr buena información de salida, sin embargo esto es generalmente un problema muy difícil de resolver.

Técnicas de segmentación son contextuales o no contextuales. Estas últimas no tienen en cuenta las relaciones espaciales entre las características en una imagen y el grupo de píxeles juntos sobre la base de algún atributo global, por ejemplo, nivel de gris o color. Las técnicas contextuales, por otro lado, aprovechan estas relaciones, por ejemplo, agrupar píxeles con niveles de gris similares y cercanas localidades espaciales[web39].

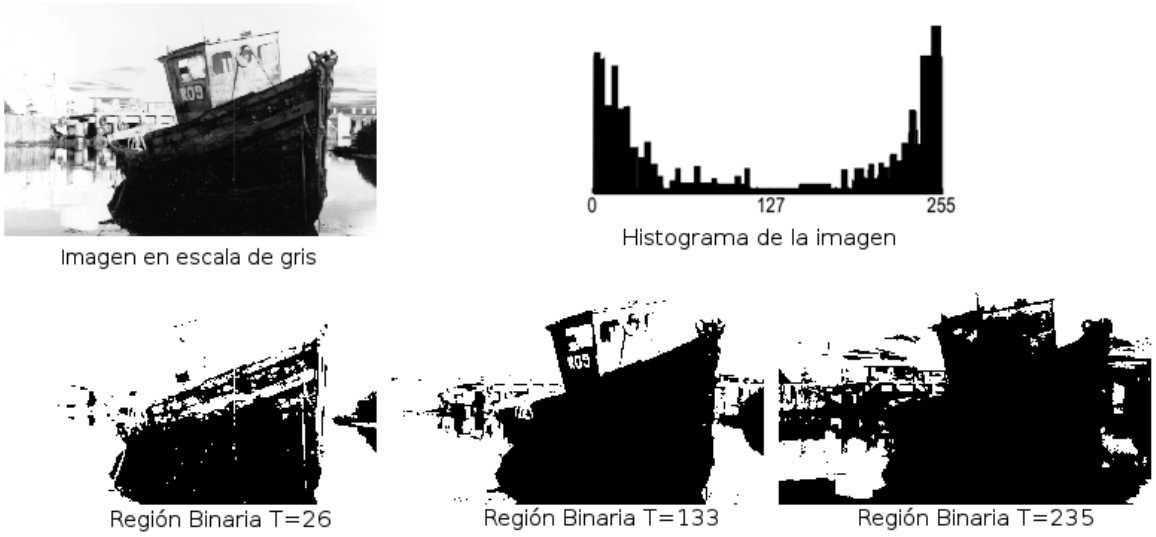


Imagen 31

Thresholding (umbral) es una conversión entre una imagen en escala de grises a una imagen de dos niveles, es decir una imagen monocromática con un nivel medio de gris definido, es la técnica más simple de segmentación no contextual. Con un único umbral, se transforma una imagen en escala de grises o en color en una imagen binaria considerada como una región de mapa binario. El mapa binario contiene dos regiones disjuntas, una de ellas contiene píxeles con valores de datos de entrada más pequeños que el umbral y la otra contiene los valores de entrada que están en o por encima del umbral[web40].

6.1.1.2. Técnicas detección de bordes

En el procesamiento de imágenes, un borde es el límite entre un objeto y su fondo. Representa la frontera de los objetos individuales. Por lo tanto, si los bordes de los objetos de la imagen se pueden identificar con precisión, todos los objetos que pueden ser localizados y sus propiedades tales como área, perímetro, forma, etc, se pueden calcular. La detección de bordes es una herramienta esencial para la visión artificial y procesamiento de imágenes.

Hay muchas formas de realizar la detección de bordes. Sin embargo, la mayoría de los diferentes métodos pueden agruparse en dos categorías:

- **Detección de bordes basado en Gradiente:** El método del gradiente detecta los bordes mediante la búsqueda del máximo y mínimo en la primera derivada de la imagen.

El gradiente de una función $f(x,y)$ en 2D, se define con el siguiente vector:

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix}$$

La magnitud de este vector se define como:

$$\nabla f = mag\left(\nabla f = \left[g_x^2 + g_y^2\right]^{\frac{1}{2}}\right) = \left[\left(\partial f / \partial x\right)^2 + \left(\partial f / \partial y\right)^2\right]^{\frac{1}{2}}$$

Algunas veces podemos aproximar esta fórmula, omitiendo la raíz cuadrada (a) o utilizando valores absolutos (b).

$$\text{(a) } \nabla f \approx g_x^2 + g_y^2 \text{ , (b) } \nabla f \approx |g_x^2| + |g_y^2|$$

- **Detección de bordes basado en Laplace:** El método Laplaciano busca cruces por cero en la segunda derivada de la imagen para encontrar los bordes. Un borde tiene forma unidimensional y mediante el cálculo de la segunda derivada de la imagen se puede hacer posible localizar su ubicación.

  En matemáticas el Laplaciano es un operador diferencial dado por la divergencia del gradiente de una función en el espacio euclidiano. El laplaciano de una imagen $f(x,y)$, es denotado por $\nabla^2 f(x,y)$, y definido como:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

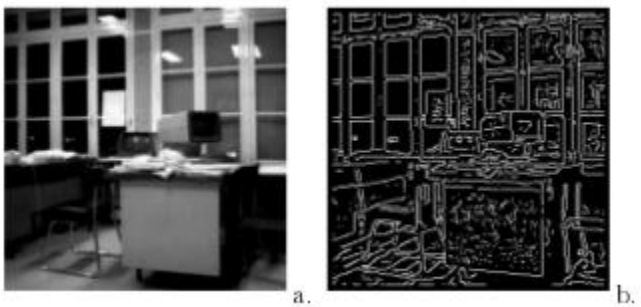


Imagen 32

a) Imagen Original – b) Imagen transformada con detección de bordes

La detección de bordes puede ser dividida en tres pasos básicos, Filtro, mejora y detección. La visión general de los pasos de detección de bordes son los siguientes:

1. **Filtros:** Las imágenes son a menudo dañadas por variaciones aleatorias en los valores de intensidad, generalmente llamado ruido. Algunos tipos comunes de ruido son el ruido sal y pimienta, ruido de impulso y el ruido Gaussiano. El ruido sal y pimienta contiene ocurrencias aleatorias de los valores de intensidad tanto en blanco y como en negro. Sin embargo, existe una gran relación entre detección de bordes y la reducción de ruido. Demasiado filtrado para reducir el ruido resulta en una pérdida en la capacidad de detectar el borde.

2. **Mejora:** Con el fin de facilitar la detección de bordes, es esencial determinar los cambios en la intensidad en la zona alrededor de un punto. La mejora hace hincapié en pixeles donde hay un cambio significativo en los valores de intensidad locales y por lo general se lleva a cabo mediante el cálculo de la magnitud del gradiente.

3. **Detección:** Muchos puntos en una imagen tienen un valor distinto de cero para el gradiente, y no todos estos puntos son bordes. Por lo tanto, algún método se debe utilizar para determinar qué puntos son puntos de borde. Con frecuencia, la umbralización proporciona el criterio utilizado para la detección.

Al momento de determinar el método para realizar la detección de bordes, tenemos muchas opciones, algunas de las cuales podemos encontrar ya definidas en Matlab, el lenguaje de programación y modelización usado en el ambiente científico y académico. El siguiente código es un ejemplo de algunas de las funciones predefinidas del lenguaje:

```
a= imread('img1.jpg');
subplot(3,3,1); imshow(a);
ag=rgb2gray(a);
subplot(3,3,2); imshow(ag);


b= edge(ag,'canny');
subplot(3,3,3); imshow(b);

c= edge(ag,'sobel');
subplot(3,3,4); imshow(c);

d= edge(ag,'prewitt');
subplot(3,3,5); imshow(d);

e= edge(ag,'roberts');
subplot(3,3,6); imshow(e);
```

algoritmo04

El resultado del algoritmo anterior se observa en la siguiente imagen:

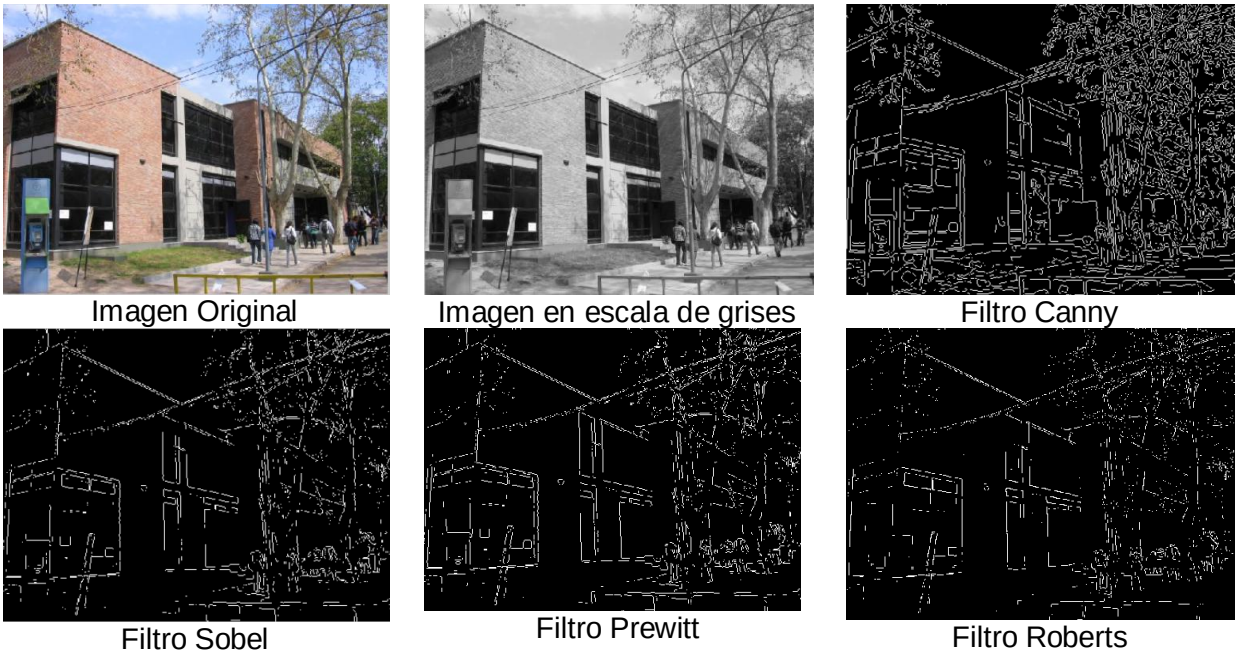


Imagen Original | Imagen en escala de grises | Filtro Canny

Filtro Sobel | Filtro Prewitt | Filtro Roberts

Imagen 33

Filtro Canny: El detector de bordes de Canny es un método popular para la detección de bordes que se inicia utilizando el suavizado de una imagen mediante la convolución con una gaussiana de un valor sigma dado. Basado en la imagen suavizada, se calculan tanto en la dirección X e Y las derivadas, las cuales a su vez se utilizan para calcular la magnitud del gradiente de la imagen.

Filtro Sobel: El operador Sobel es utilizado en procesamiento de imágenes, especialmente en algoritmos de detección de bordes. Técnicamente es un operador diferencial discreto que calcula una aproximación al gradiente de la función de intensidad de una imagen. Para cada punto de la imagen a procesar, el resultado del operador Sobel es tanto el vector gradiente correspondiente como la norma de éste vector.

Filtro Prewitt: El operador Prewitt es un operador de diferenciación discreta, calcula una aproximación del gradiente de la función de intensidad de la imagen. En cada punto de la imagen, el resultado del operador de Prewitt es o bien el vector de gradiente correspondiente o la norma de este vector. El operador de Prewitt se basa en la convolución de la imagen con un entero valorado en dirección horizontal y vertical y por lo tanto es relativamente barata en términos de cálculo.

Filtro Roberts: El detector de bordes de Roberts se basa en el uso del operador gradiente que aplica la derivada respecto a los ejes x.

6.1.1.3. Procesamiento morfológico de imágenes

La palabra morfología denota normalmente una rama de la biología que se ocupa de la forma y estructura de los animales y las plantas. En procesamiento de imágenes usamos la misma palabra en el contexto de la morfología matemática como una herramienta para la extracción de los componentes de la imagen que son útiles en la representación y la descripción de la forma de la región, tales como las fronteras, los esqueletos, etc.

El lenguaje de morfología matemática es la teoría de conjuntos. Como tal, la morfología ofrece un enfoque unificado y potente a numerosos problemas de procesamiento de imágenes. Los conjuntos en la morfología matemática representan objetos de una imagen. Por ejemplo, el conjunto de todos los píxeles negros en una imagen binaria es una descripción morfológica completa de la imagen. En las imágenes binarias, los conjuntos en cuestión son miembros del espacio entero, Z^2, donde cada elemento de un conjunto es una tupla (vector 2-D) cuyas coordenadas (x, y) son las coordenadas de los píxeles negros (o blancos, dependiendo en la convención que usemos) de la imagen. Las imágenes digitales en escala de grises se pueden representar como conjuntos de cuyos componentes se encuentran en Z^3. En este caso, dos componentes de cada elemento del conjunto se refieren a las coordenadas de un píxel, y el tercero corresponde a su valor discreto de nivel de gris. Conjuntos en espacios de dimensiones superiores pueden contener otros atributos de imagen, como el color y variabilidad en el tiempo los componentes[web41].

Las técnicas morfológicas sondean una imagen con una pequeña forma o plantilla llamada elemento de estructuración, el cual se posiciona en todas las posibles ubicaciones en la imagen y se compara con la correspondiente zona de píxeles. Con cada posible posicionamiento, se analiza si la plantilla cayó en una zona ocupada por la imagen, si la zona es mixta, o si no pertenece a la imagen. Estos tres casos posibles se pueden ver en la siguiente imagen:

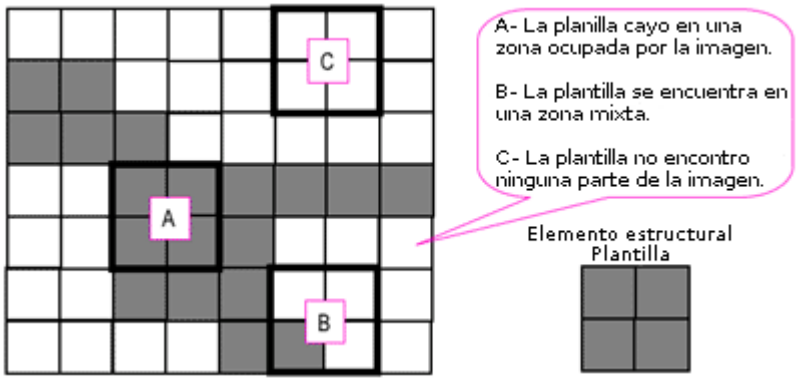


Imagen 34

La mayoría de los algoritmos morfológicos utilizados hoy de día hacen uso de 2 operaciones fundamentales en el procesamiento morfológico, Dilatación (dilation) y Erosión (erosion):

- **Dilatación:** Dados A y B conjuntos en Z^2, la dilatacion de A por B es denotada por $A \oplus B$, y se define de la siguiente forma:

$$\boldsymbol{A \oplus B = \left\{ z | \left(\hat{B} \right)_z \cap A \neq \emptyset \right\}}$$

Esta ecuación se basa en la obtención de la reflexión de B sobre su origen y trasladando esta reflexión por z. La dilatación de A por B es el conjunto de todos los desplazamientos, z, tal que $\hat{B}$ y A se solapan al menos por un elemento. En base a esta interpretación, la ecuacion anterior se puede escribir como

$$\boldsymbol{A \oplus B = \left\{ z | \left[\left(\hat{B} \right)_z \cap A \right] \subseteq A \right\}}$$

El conjunto B se define comúnmente como el elemento de estructuración de la dilatación, así como en otras operaciones morfológicas.

La ecuación presentada no es la única definición de la dilatación en la literatura actual sobre la morfología, sin embargo, la definición anterior tiene una clara ventaja sobre otras formulaciones ya que es más intuitivo ver el elemento

estructurante B como una máscara de convolución. Aunque la dilatación se basa en operaciones de conjunto, mientras que la convolución se basa en operaciones aritméticas, el proceso básico de transformación de B sobre su origen y luego, sucesivamente, desplazándolo de forma que se desliza sobre el conjunto (imagen) A es análogo al del proceso de convolución[web42].

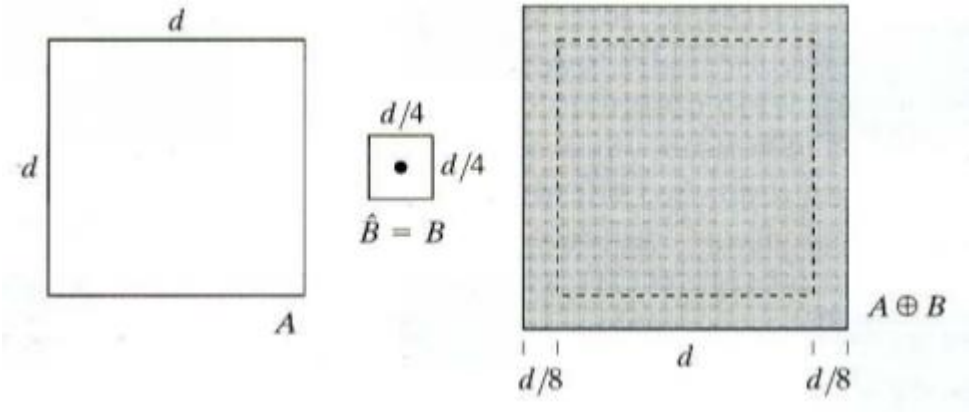


Imagen 35

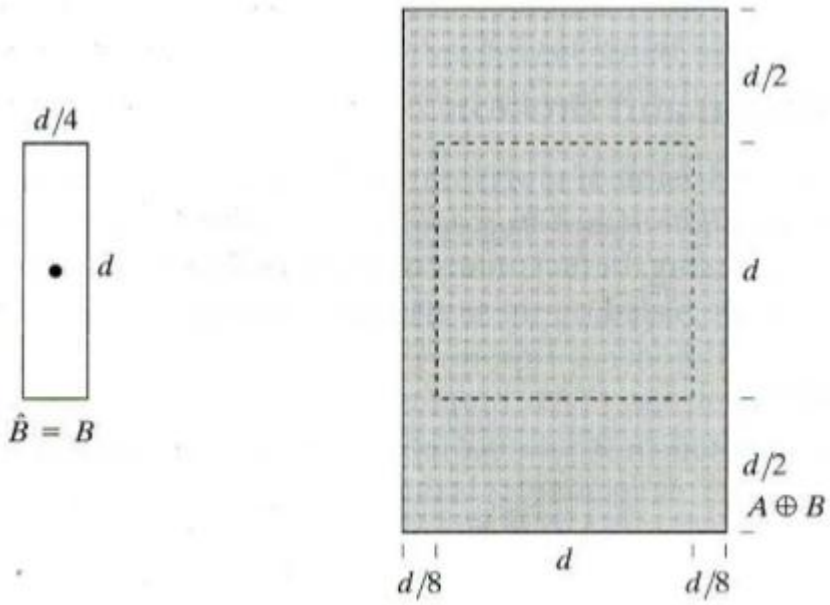


Imagen 36

Como puede verse en la Imagen 35, dado un conjunto A (imagen) y un elemento de estructuración B, cuadrado, de lado ¼ del de la imagen A, el

resultado de la dilatación es en observado en $A \oplus B$, y denotado por la línea punteada.

Otro ejemplo se puede observar en la imagen 36 en la cual se utilizó un elemento estructural rectangular del mismo alto de la imagen A y de ancho ¼ de la misma, el resultado se observa en $A \oplus B$ y al igual que la anterior es mostrado con la línea punteada, esta estructura se utiliza generalmente cuando se desea lograr una dilatación vertical mayor que la horizontal.

Se puede observar en la imagen 37 un claro ejemplo del uso práctico de esta operación, en A, observamos un texto con poca calidad, luego de aplicar la dilatación utilizando el elemento estructural B, podemos denotar que las secciones rotas de los caracteres fueron unidas.

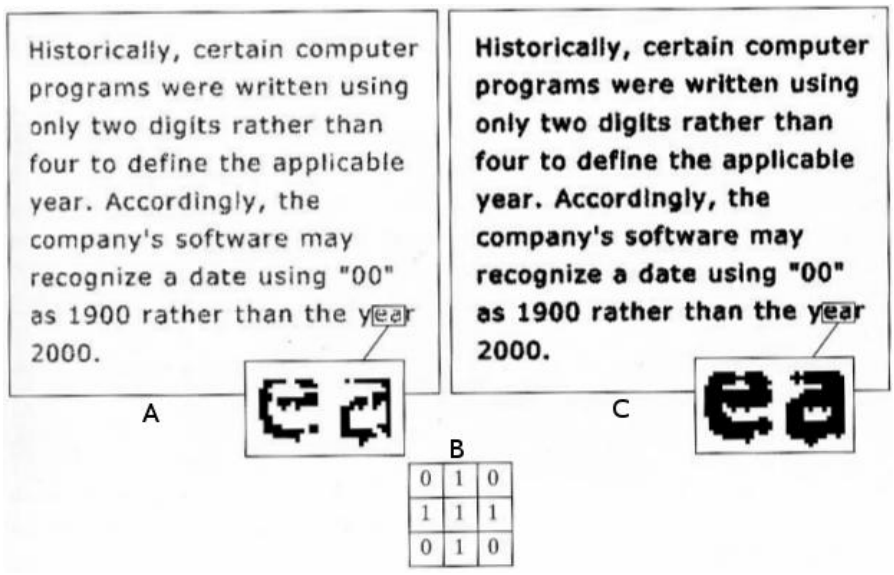


Imagen 37

- **Erosión:** Dados A y B conjuntos en Z^2, la erosión de A por B es denotada por $A \ominus B$, y se define de la siguiente forma:
    - $\boldsymbol{A \ominus B = \{z|(B)_z \subseteq A\}}$

En otras palabras, esta ecuación indica que la erosión de A por B es el conjunto de todos los puntos z tales que B, trasladada en z, están contenidos en A. Como en el caso de la dilatación, esta no es la única posible definición de erosión, pero de la misma forma, es más intuitivo mostrarla así.

Para entender mejor la diferencia entre estos operadores, utilizando la misma Imagen A y el mismo elemento estructural de las imágenes 35 y 36, observamos lo siguiente al aplicar erosión:

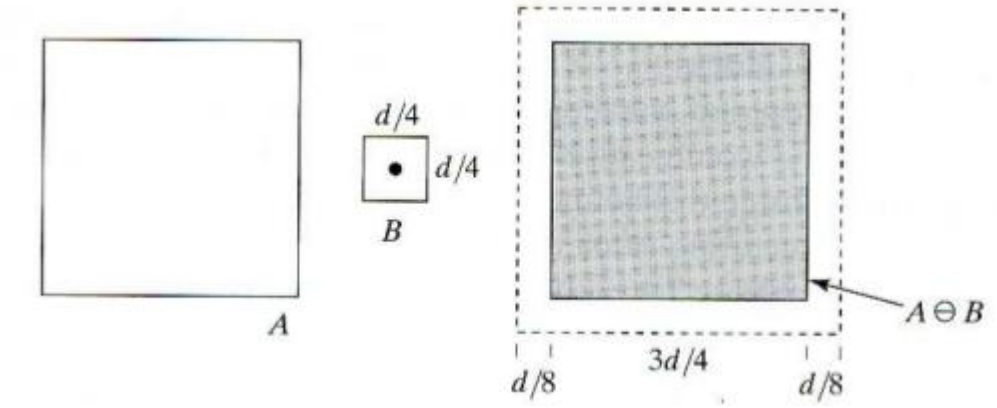


Imagen 38

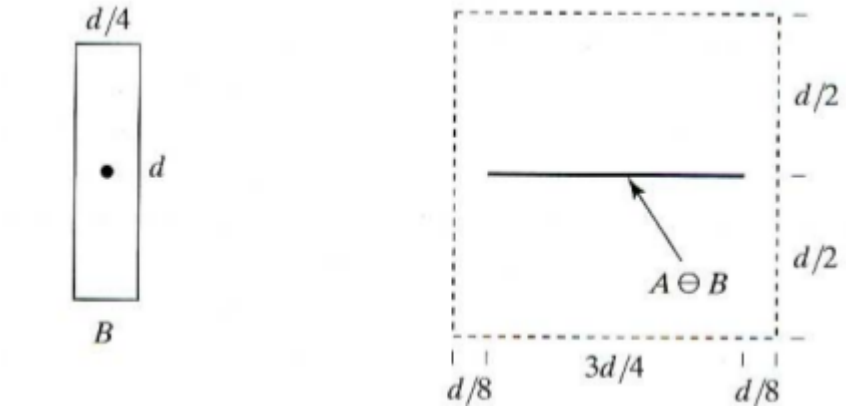


Imagen 39

La imagen 38 muestra un proceso similar al que se muestra en la imagen 35. Como antes, el conjunto A se muestra como una línea punteada de referencia en $A \ominus B$. El límite de la región sombreada muestra el límite a partir del cual si se realiza un desplazamiento adicional de B, obtendríamos un conjunto que ya no sería completamente contenido en A. Por lo tanto el lugar geométrico de los puntos dentro de este límite (es decir, la región sombreada) constituye la erosión de A por B. la Imagen 39 muestra un elemento de estructuración alargado, y el resultado muestra la erosión de A por este elemento, en este caso el conjunto original fue erosionado hasta una línea.

La dilatación y la erosión son duales entre sí con respecto a la complementación y la reflexión. Es decir,

$$(\boldsymbol{A} \ominus \boldsymbol{B})^{\boldsymbol{c}} = \boldsymbol{A}^{\boldsymbol{c}} \oplus \widehat{\boldsymbol{B}}$$

6.1.1.4. Algoritmos de Esqueletización

La esqueletización es una transformación de un componente de una imagen digital en un subconjunto de la componente original.

Hay diferentes categorías de métodos esqueletización: una categoría se basa en transformaciones de distancia, y el subconjunto especificado de la imagen transformada es un esqueleto. Como en todas las categorías, el componente original puede ser reconstruido a partir de este esqueleto.

Otra categoría se define por el adelgazamiento, y el resultado de la esqueletización utilizando algoritmos de adelgazamiento debe ser conectado a un conjunto de curvas o arcos.

La principal motivación del interés en algoritmos de esqueletización es la necesidad de optimizar el cálculo utilizando una cantidad reducida de datos o para simplificar la forma de un objeto con el fin de encontrar características que pueden ser utilizadas por algoritmos de reconocimiento y clasificación. Además, la transformación de un componente en una imagen que muestra las características esenciales puede eliminar el ruido local en la frontera del objeto[web43].

Existen Varias formas de lograr la esqueletización de una imagen[web44]:

- **Adelgazamiento Homotópico:** Históricamente, las primeras aproximaciones algorítmicas para el cálculo de los esqueletos utilizaron adelgazamiento homotópico. Su principio consiste en la eliminación de capas sucesivas de los conjuntos que se esqueletizan hasta que se alcanza el espesor de un píxel. En cada paso, se pueden eliminar sólo los píxeles exteriores al área particular. Las configuraciones se eligen de manera que la homotopía del conjunto no se vea afectada en la eliminación de cualquier pixel. En el campo de la morfología matemática, estas configuraciones son a menudo referenciadas por medio de elementos de estructuración y la operación de eliminación de capas se llama adelgazamiento morfológico. trabajan en paralelo a través de todos los píxeles de la imagen. Por lo tanto son particularmente adecuados para implementaciones de hardware y son ampliamente utilizados en la práctica.

- **Funciones de distancia Cresta-Línea:** Esta segunda clase de algoritmos se basa en el hecho de que el esqueleto se puede obtener como conjunto de los máximos locales de la función de distancia. Estos máximos locales son en realidad un subconjunto de los puntos de la cresta de la función, la idea es partir de estos píxeles, conectarlos de la mejor manera usando las líneas de la cresta de la función de distancia. Este último paso puede llevarse a cabo secuencialmente, de modo que estos algoritmos son mucho más rápidos que los basados en adelgazamiento paralelo. También están diseñados para producir un superconjunto homotópico del esqueleto y son por lo tanto más precisos que los anteriores. Sin embargo, implican estudios complejos de las configuraciones del entorno correspondientes a los puntos de la cresta y los píxeles susceptibles de ser modificados durante la corrida del algoritmo, por lo tanto no pueden transponerse fácilmente de una imagen a otra y, además, que sólo permiten el cálculo de una pequeña variedad de esqueletos.

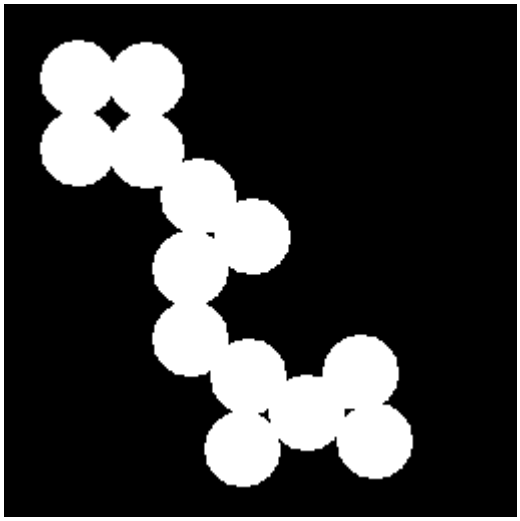
- **Algoritmos basados en contornos:** Varios autores han tratado de transcribir directamente en términos algorítmicos la descripción del esqueleto como el lugar geométrico en la que dos frentes procedentes de los bordes del objeto se encuentran. Entre los métodos más interesantes, los que utilizan técnicas de bucle o sucesivos rastreos de nivel son los más utilizados. Estas técnicas disminuyen el tiempo de cálculo requerido por escaneos continuos de los comjuntos de píxeles, ya que los píxeles esqueleto sólo se detectan a partir de los bucles, sin tener que volver a la imagen. Esto hace a estos algoritmos extremadamente rápidos, pero a cambio, nos enfrentamos a los inconvenientes comunes a la mayoría de los algoritmos de este tipo: es muy difícil para ellos transponer de una imagen a otra, y es imposible hacerlas extensivas a los espacios 3-d. Por otra parte, no permitirán el cálculo de muchos esqueletos diferentes.

Una forma muy simple de lograr la esqueletización de una imagen, es utilizando los algoritmos ya definidos en Matlab, la función bwmorph es la encargada de aplicar este filtro, el siguiente es un ejemplo de esto:

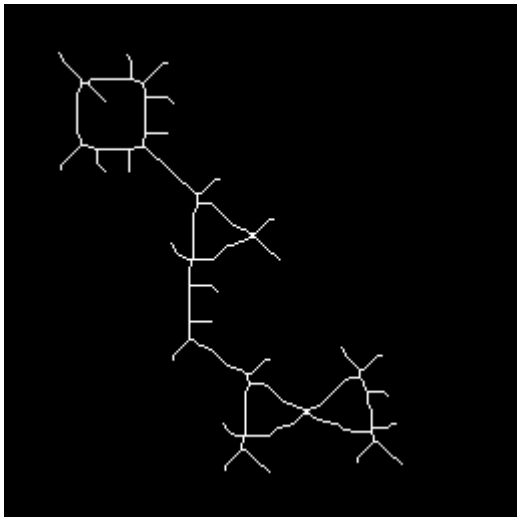
Codigo:

```
BW = imread('circles.png');
imshow(BW);
BW2 = bwmorph(BW,'skel',Inf);
figure, imshow(BW2)
```

Resultado:



A= BW, Imagen Original



B= BW2, Esqueletización

6.1.2. Visión Estereoscópica

La Visión estereoscópica artificial es un problema en la informática con muchas soluciones, cada una con sus propios problemas. Dadas dos imágenes que fueron tomadas a la misma altura y con el mismo ángulo, estos algoritmos tratan de encontrar la profundidad de cada píxel de las imágenes. Nuestros cerebros resuelven este problema 30 veces por segundo para proveernos con la percepción de profundidad. La Visión estereoscópica artificial es una rama de la informática (mas específicamente, de la Inteligencia artificial) que trata de resolver este problema y crear mapas de disparidad que dan aproximaciones de profundidad para cualquier conjunto de imágenes [web45].

Según mathworks.com (Empresa responsable de Matlab) La visión estéreo es el proceso de recuperación de profundidad a partir de imágenes mediante la comparación de dos o más vistas de la misma escena. La visión estéreo binocular utiliza sólo dos imágenes, típicamente tomadas con cámaras paralelas que se han separado por una distancia horizontal conocida como la "línea de base". La salida del algoritmo de visión estéreo es un mapa de disparidad (que es traducible a una imagen de rango) que indica la distancia que cada punto en la escena estaba de la cámara [web46].

La siguiente imagen muestra un ejemplo del mapa de disparidad que estos algoritmos intentan obtener:

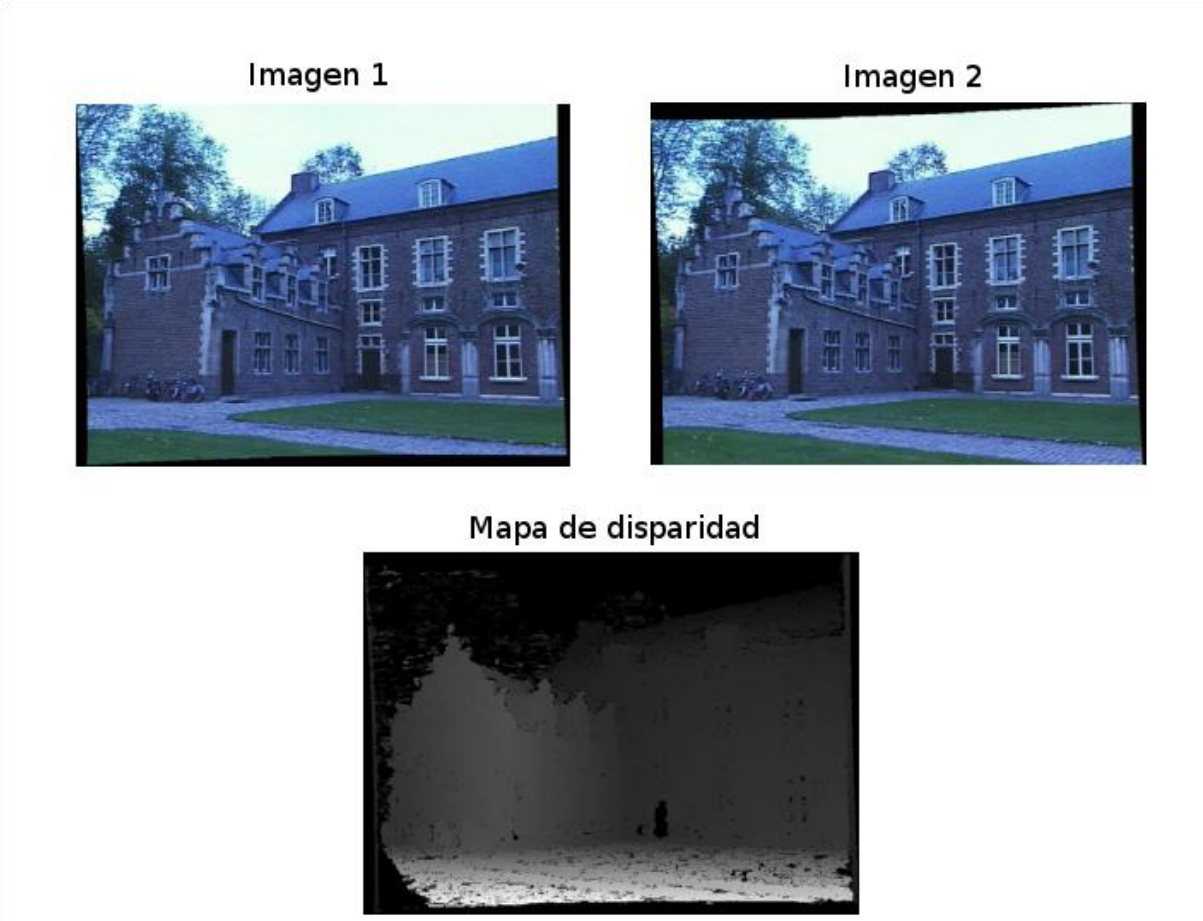


Imagen 40

El problema de hacer coincidir los mismos puntos o zonas de pares de imágenes estéreo se llama el problema de la correspondencia. La coincidencia de estas imágenes podría ser realizada por algoritmos que incluyen búsqueda y comparación entre las partes de dos imágenes.

Si se conoce la geometría de la cámara, la búsqueda en dos dimensiones de los puntos correspondientes podría simplificarse a una búsqueda unidimensional. Esto se hace mediante la rectificación de las imágenes que se basa en la geometría epipolar (rectificación epipolar) [web47].

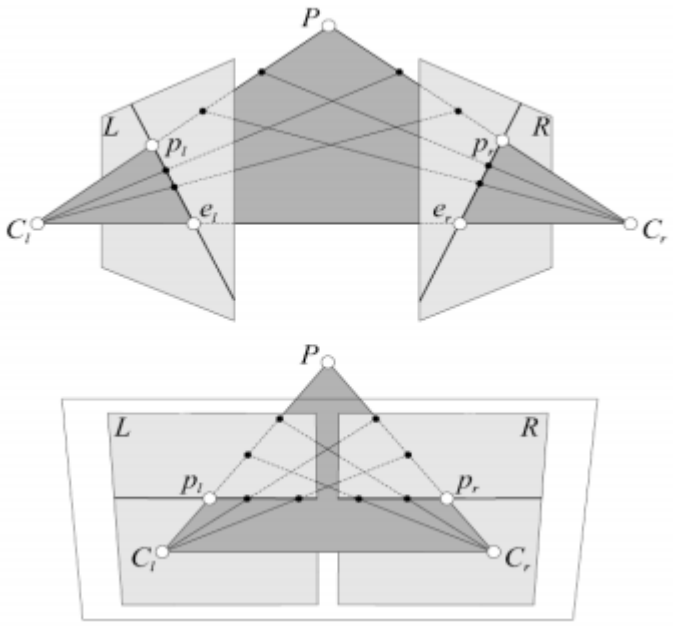


Imagen 41

Rectificación epipolar

La rectificación epipolar asume que las cámaras son paralelos entre sí y que tienen longitudes focales idénticas. Si se conoce la geometría del sistema de las dos cámaras, la longitud de línea base ($C_l - C_r$) es un valor constante (B). Comparación de los triángulos (triangulación), es posible determinar la distancia a un punto u objeto (X, Y, Z) o la profundidad del mismo (Z):

$$Z = \frac{B \cdot f}{x_l - x_r} = \frac{B \cdot f}{d}$$

Aunque la búsqueda de los puntos correspondientes de la imagen de la izquierda en la imagen de la derecha se simplifica, todavía hay algunos problemas específicos en la visión estéreo. La falta de coherencia entre las fotos (imágenes) es común debido al hecho de que la intensidad y los colores pueden variar dependiendo del punto de vista. La intensidad y los colores en una pareja de imágenes también podrían variar debido a las diferentes características de los sensores de cámaras. Además, la cámara electrónica produce ruido que

afecta a la adquisición de imágenes. En general, todas estas diferencias son muy pequeñas por lo que puede dejarse de lado y la consistencia de la foto se asume como una limitación en los algoritmos de correspondencia [web48].

Con el uso de pares de imágenes como representación de una escena, es posible extraer información estéreo de la misma, utilizar la rectificación epipolar simplifica la correspondencia de pixeles en cada imagen, utilizando búsqueda y correlación unidimensional. Encontrar píxeles correspondientes en dos imágenes, cuyos desplazamientos o disparidades pueden ser calculados y cuya profundidad puede ser extraída a partir de estas disparidades.

El proceso es posible para todos los píxeles de la imagen que se corresponden correctamente en los pares de imágenes, pero la profundidad se pierde para los píxeles no coincidentes. La coincidencia de algunos píxeles es imposible debido a los obstáculos que se ven solamente en una de las cámaras. El objetivo es desarrollar algoritmos que puedan manejar estas zonas durante el proceso de comparación.

La visión por computadoras entra a la robótica, no como un elemento fundamental de los robots móviles, pero si como una adición muy positiva para mejorar la forma en la cual los robots interactúan con su entorno, aportando datos de gran utilidad a otras de las aéreas vistas a lo largo de esta tesis (en especial, localización y mapeo).

Existe una gran cantidad de aplicaciones para estas técnicas, y aun se continúa trabajando para mejorar el procesamiento de las imágenes y lograr que el robot tenga una mayor comprensión de lo que la imagen está mostrando.

7. Conclusión y Escenarios Futuros

7.1. Alcance y Limitaciones

Si bien a lo largo de esta tesis se ha hecho un completo repaso por las áreas más importantes de la robótica en la actualidad, no se han incluido demostraciones matemáticas o físicas ni implementaciones específicas de los métodos estudiados.

Esto se realizo con la intención de mantener el foco en los métodos y no en la matemática y la física que los respaldan, a fin de no perder de vista el tema central de la tesis ni extender demasiado en contenidos que no hacen al objetivo de la misma. Para una implementación futura, atomizando cada uno de los capítulos vistos aquí en tesis individuales, sería posible profundizar en estos temas.

7.2. Escenarios futuros

Los futuros trabajos en el ámbito de la robótica, consistirán en mejorar la percepción del mundo que los rodea y mejorando la interacción de los robots con su ambiente, y en particular con los humanos, utilizando sensores más precisos y menos sensibles a ruidos externos.

La visión de un mundo futuro poblado por humanos, cyborgs, robots y androides plantea muchas cuestiones fundamentales. Una de estas preguntas plantea que es lo que esto significa para los derechos fundamentales o constitucionales, también conocidos como derechos humanos. ¿Los cyborgs pueden considerar suficientemente humano para seguir siendo sujetos de derechos "humanos"? Puede androides reclamar derechos "humanos" si se ven y funcionan de la misma manera en la sociedad como cyborgs? Otra cuestión importante es la relación entre las personas no mejoradas y las mejoradas: ¿Habrá una brecha social? Y pueden los seres humanos tener robots bajo control, ya que son cada vez más autónoma, es decir, serán robots capaces de cumplir con tres leyes de Asimov de la robótica hasta el final del día, o serán, como HAL en 2001 - Una Odisea del Espacio, una revuelta e intentaran controlar a los humanos? Estos tipos de problemas son escenarios ampliamente desarrollados en la ciencia ficción, sin embargo es muy poco probable que tengamos que lidiar con estas cuestiones en el futuro cercano.

7.3. Conclusión

Para finalizar, los métodos enumerados en esta tesis tienen como intensión crear un marco teórico para lograr realizar el análisis de un robot móvil utilizando la tecnología actual. Futuras implementaciones prácticas pueden ser realizadas a partir de este trabajo, tomando como guía cada apartado visto aquí.

Es importante remarcar que aún queda mucho camino por delante, desde la creación de nuevas tecnologías hasta el desarrollo de algoritmos más completos que sigan llevando a la práctica los principios vistos en esta tesis, haciendo hincapié en las mejoras que se pueden logar y en el uso correcto de las mismas.

La utilización de robots en nuestra vida cotidiana ya es una realidad, podemos enunciar varios ejemplos de esto, de hecho, la mayoría de los objetos que utilizamos regularmente, fueron en algún momento (o completamente) de su proceso de manufactura manipulados por robots, y las mejoras que se han introducido en la industria y en los campos de investigación han logrado que se ponga especial atención en estas tecnologías.

Pero a su vez, como se dijo al principio de este trabajo, es muy importante tener en cuenta las implicaciones socio-económicas de incorporar cada vez más a los robots en nuestras vidas, y tender a la fabricación de mecanismos que mejoren la vida humana.

8. Anexos

8.1. Anexo A: Robots Actuales

8.1.1. Nasa Curiosity (Mars Science Laboratory)

La Misión Mars Science Laboratory de la NASA logro llevar un gran laboratorio móvil - el Rover Curiosity - al cráter Gale en Marte, utilizando tecnología de aterrizaje de alta precisión que hace que muchas de las regiones más interesantes de Marte puedan ser accesibles por primera vez. Durante los 23 meses después del aterrizaje, Curiosity analizará decenas de muestras de rocas perforadas o excavado de la tierra, ya que explora con mayor alcance que cualquier Rover marciano enviado anteriormente.

El Curiosity lleva la carga más avanzada de equipo científico jamás utilizado en la superficie de Marte, una carga útil de más de 10 veces la masa de los anteriores Rovers de Marte. Su tarea: Investigar si las condiciones han sido favorables para la vida microbiana y para la preservación de las pistas en las rocas sobre una posible vida pasada.

El Curiosity fue lanzado desde Cabo Cañaveral el 26 de noviembre de 2011, a las 10:02 EST bordo de la nave espacial MSL y aterrizó con éxito en Aeolis Palus en el cráter Gale de Marte el 6 de agosto, 05:17 UTC. El sitio de aterrizaje fue menos de 2,4 kilometros (1,5 millas) del centro de la meta del rover después de 563 millones kilometros (350 000 000 mi) viaje.

Imagen 42

El Curiosity es el doble de tamaño (alrededor de 3 metros) y cinco veces más pesado que los anteriores vehículos de exploración de Marte de la NASA, Spirit y Opportunity, lanzados en 2003. Heredó muchos elementos de diseño de los mismos, incluyendo seis ruedas, un sistema de suspensión rocker-bogie y cámaras montadas en un mástil para ayudar al equipo de la misión explorar objetivos selectos de la Tierra y rutas de exploración. A diferencia de exploradores anteriores, Curiosity lleva el equipo para recoger muestras de rocas y suelo,

procesarlos y distribuirlos a bordo cámaras de prueba dentro de los instrumentos de análisis[web49].

"Cuerpo" del Rover: El cuerpo Rover se llama caja electrónica cálida, o "WEB" por sus siglas en ingles. Como el cuerpo de un auto, el cuerpo del Rover es una capa fuerte exterior que protege el equipo y la electrónica (que son básicamente el equivalente de cerebro y corazón del Rover) del Rover. Así, el cuerpo móvil mantiene los órganos vitales del explorador protegidos y con temperatura controlada.

La caja electrónica cálida se cierra en la parte superior por una pieza llamada el Equipo de la cubierta Rover (RED). El Equipo de la cubierta Rover hace que el vehículo sea como un coche convertible, lo que permite un lugar para el mástil y las cámaras utilizadas tomar fotografías y observar con claridad el terreno marciano a medida que viaja.

"Cerebro" del robot: El equipo móvil ("cerebro") se encuentra dentro de un módulo llamado "The Rover Compute Element" (RCE) en el interior del cuerpo móvil. La interfaz de comunicación que permite que el equipo principal intercambie datos con los instrumentos y sensores del explorador se llama un "bus"). Este es un bus de interfaz estándar para comunicarse y controlar todos los motores rover, instrumentos científicos, y las funciones de comunicación.

Mejor Memoria: El equipo dispone de memoria especial para tolerar el ambiente de radiación extrema desde el espacio, así como contra los ciclos de apagado para que los programas y los datos se mantengan y no se borren accidentalmente cuando el vehículo se cierra por la noche. La memoria de a bordo incluye 256 MB de DRAM y 2 GB de memoria flash ambas con detección y corrección de errores y 256 KB de memoria EEPROM. Esta memoria interna es de aproximadamente 8 veces más capaz que las a bordo de los Rovers anteriores.

Mejor equilibrio y posición: El Rover lleva una Unidad de Medida Inercial (IMU) que proporciona información de 3 ejes en su posición, lo que permite que el Rover pueda hacer movimientos precisos verticales, horizontales y de lado a lado. El dispositivo se utiliza en la navegación móvil para lograr un apoyo seguro y para estimar el grado de inclinación que rover está experimentando en la superficie de Marte.

Seguimiento de su "salud": Al igual que el cerebro humano, las computadoras del Rover llevan registro de los signos de la salud, la temperatura y otras características que mantienen el Rover "vivo". Este bucle de control principal en esencia mantiene el Rover en constante control de sí mismo para asegurarse de que es a la vez capaz de comunicarse en toda la misión con la superficie y que permanece estable térmicamente (no demasiado caliente

o demasiado frío) en todo momento. Lo hace mediante la comprobación periódica de las temperaturas, sobre todo en el cuerpo rover, y responde a las posibles condiciones de sobrecalentamiento, el registro de generación de energía y los datos de almacenamiento de energía en todo el sol de Marte (un día marciano), y la programación y la preparación para las sesiones de comunicación.

Comunicaciones: Actividades tales como la toma de fotografías, la conducción y el funcionamiento de los instrumentos se realizarán bajo órdenes transmitidas en una secuencia de comandos para el móvil del equipo de vuelo. El rover generará constante ingeniería, limpieza y análisis de la telemetría y de los informes de eventos periódicos que se almacenan para su posterior transmisión una vez que el equipo de vuelo solicita la información del rover. El vehículo tiene dos "cerebros informáticos", uno que esta normalmente apagado. En caso de problemas este otro cerebro puede ser despertado para tomar el control y continuar la misión[web50].

Cámaras de evasión de peligro (HAZCAMS): Montado en la parte inferior de la parte delantera y trasera del rover, estas cámaras blanco-negro utilizan la luz visible para capturar imágenes en tres dimensiones (3-D). Estas imágenes evitan que el rover se pierda o sin querer choque contra obstáculos inesperados, y trabaja en conjunto con un software que permite al rover tomar sus propias decisiones de seguridad y pensar por sí mismo la solución optima.

Las cámaras tienen un campo de visión amplio de aproximadamente 120 grados. El rover utiliza pares de imágenes Hazcam para trazar la forma del terreno hasta 3 metros en frente a ellas, en forma de "cuña" que tiene más de 4 metros de ancho a la máxima distancia. Las cámaras tienen que ver a uno y otro lado, porque a diferencia de los ojos humanos, las cámaras Hazcam no pueden moverse de forma independiente, sino que están montados

Imagen 43

directamente en el cuerpo móvil.

Cámaras de navegación (NAVCAMS): Montado en el mástil (el "cuello y la cabeza" rover), estas cámaras blanco y negro utilizan la luz visible para recopilar las imágenes panorámicas en tres dimensiones (3D). La unidad de la cámara de navegación es un par estéreo de cámaras, cada una con un campo de 45 grados de vista que apoyará la planificación de la navegación en tierra por los científicos e ingenieros. Trabajan en colaboración con las cámaras de evasión de peligros, proporcionando una visión complementaria del terreno.

Cámaras de ciencia (MastCam, ChemCam, MAHLI): Cámara Mástil tomará imágenes en color, imágenes estéreo tridimensionales y video en color del terreno marciano y tienen un zoom de gran alcance.

Al igual que las cámaras de los Rovers de Exploración que aterrizaron en el planeta rojo en 2004, el diseño MastCam consta de dos sistemas de cámara duplicados montados en un mástil que se extiende hacia arriba desde de la cubierta del rover. Las cámaras funcionan como los ojos humanos, producen imágenes estéreo en tres dimensiones mediante la combinación de dos imágenes lado a lado tomadas de posiciones ligeramente diferentes.

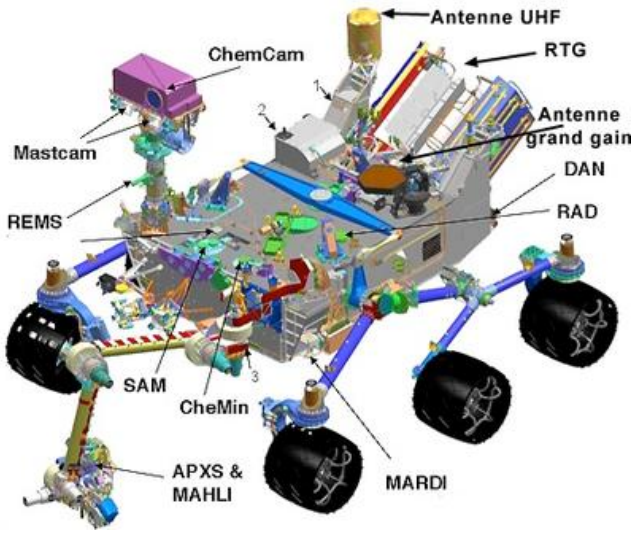


Imagen 44

Los láseres de teledetección inducida para Química y micro-imaging (ChemCam) dispararán un láser y analizan la composición elemental de materiales vaporizados desde zonas de 1 milímetro de diámetro en la superficie de las rocas y suelos de Marte. Un espectrógrafo a bordo proporciona detalles sin precedentes sobre los minerales y microestructuras en rocas mediante la medición de la composición del plasma resultante - un gas extremadamente caliente hecho de iones de libre flotación y electrones.

La lente de mano (MAHLI) es el equivalente a la mano de un geólogo y proporcionará vistas en primer plano de los minerales, texturas y estructuras de las rocas marcianas y la capa superficial de restos de rocas y polvo. Con este nuevo dispositivo, los geólogos terrestres

podrán ver las características de rocas marcianas en secciones más pequeñas que el diámetro de un cabello humano.

Camaras MARDI: Los ingenieros que trabajaron en la misión anterior fueron capaces de tener una idea de lo que el terreno marciano "parecía" a través de las cámaras DIMES del Spirit y Opportunity. Este sistema se utiliza para detectar el movimiento de la nave y ajustarlo - con cohetes retro - si es necesario. El Curiosity cuenta con un sistema visual aún más capaz. MARDI (Marte Descent Imager) proporcionó cuatro cuadros por segundo de vídeo en alta resolución durante el aterrizaje del Curiosity. Las imágenes son de "color verdadero", o como el ojo humano las vería. Además del impresionante video, los datos que la cámara recoge permitirán a los científicos e ingenieros observar los procesos geológicos en una variedad de escalas, muestra el perfil de viento horizontal, crea un mapa geológico y geomorfológico detallado.

"Brazo" del Curiosity: El brazo del robot sostiene y maniobras de los instrumentos que ayudan a los científicos ver de cerca las rocas y el suelo marciano. Al igual que un brazo humano, el brazo robótico tiene flexibilidad a través de tres articulaciones: hombro, codo y muñeca. El brazo permite tener un cinturón de herramientas de instrumentos científicos para extender, doblar, y cambiar el ángulo de precisión contra una roca para trabajar como un geólogo humano haría: moliendo capas, tomando imágenes microscópicas, y analizando la composición elemental de las rocas y el suelo. En el extremo del brazo hay una torre, con forma de cruz. Esta torre, es una estructura similar a la mano, tiene varias herramientas que pueden girar a través de un rango de giro de 350 grados.

"Mano" del Curiosity: En la punta del brazo se encuentra la estructura de torre en la que se montan 5 dispositivos. Dos de estos dispositivos son instrumentos in-situ o de contacto conocidos como la partícula alfa-espectrómetro de rayos X (APXS) y la lente de la mano (MAHLI). Los tres dispositivos restantes están asociados con la adquisición de la muestra y las funciones de preparación de las muestras[web51].

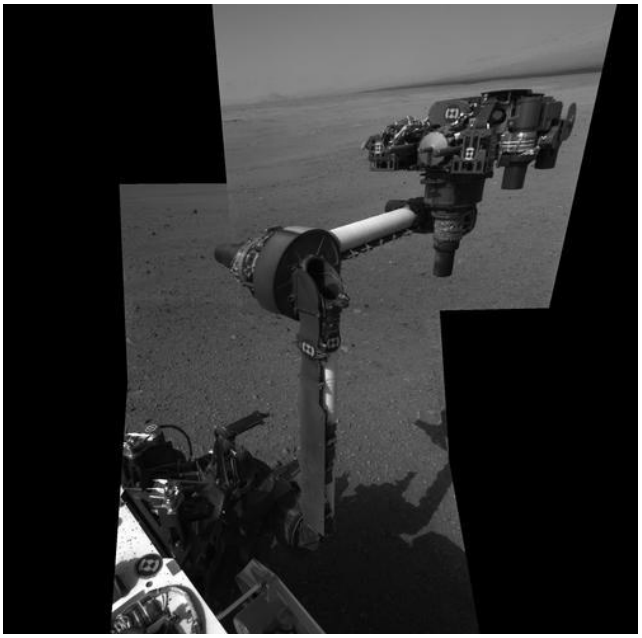


Imagen 45

Las ruedas del rover y las piernas: El Curiosity tiene seis ruedas, cada una con

su propio motor individual. Las dos ruedas delanteras y las dos traseras también tienen motores de dirección individuales (1 cada uno). Esta capacidad de dirección permite que el vehículo gire en su lugar, un total de 360 grados. La dirección en las 4 ruedas también permite al rover hacer curvas y giros arqueados.

Para contribuir a los cuatro objetivos científicos y cumplir con su objetivo específico de determinar la habitabilidad de Marte, el curiosity tiene los siguientes objetivos científicos.

- **Biológicos:**
    - Determinar la naturaleza y el inventario de compuestos de carbono orgánico.
    - Inventariar de los componentes químicos de la vida (carbono, hidrógeno, nitrógeno, oxígeno, fósforo y azufre).
    - Identificar las características que pueden representar los efectos de los procesos biológicos.
- **Geológicos y geoquímicos:**
    - Investigar la composición química, isotópica, y mineralógica de la superficie marciana y de los materiales geológicos cerca de la superficie.
    - Interpretar los procesos que han formado y modificado las rocas y los suelos marcianos.
- **Procesos planetarios:**
    - Evaluar los procesos de evolución atmosféricos a largo plazo (4 mil millones de años).
    - Determinar el estado actual, la distribución, y el ciclo de agua y dióxido de carbono.
- **Radiación superficial:**
    - Caracterizar el amplio espectro de radiación superficial, incluyendo la radiación cósmica galáctica, eventos de protones solares y neutrones secundarios.

8.1.2. ASIMO

ASIMO, un acrónimo de Advanced Step in Innovative MObility, es un robot humanoide diseñado y desarrollado por Honda. Introducido el 21 de octubre de 2000, ASIMO ha sido diseñado para ser un asistente móvil multifuncional. Con aspiraciones de ayudar a aquellos que carecen de una plena movilidad, ASIMO se utiliza con frecuencia en demostraciones en todo el mundo para fomentar el estudio de las ciencias y las matemáticas. Con 130 cm de altura y 54 kg, ASIMO ha sido diseñado para funcionar en entornos del mundo real, con la posibilidad de caminar o correr sobre dos pies a velocidades de hasta 2 kilómetros por hora. En los EE.UU., ASIMO es parte de la atracción Innoventions en Disneyland y ha sido presentado en un programa de 15 minutos llamado "Say 'Hello' a ASIMO de Honda" desde junio 2005.The robot ha hecho apariciones públicas en todo el mundo, incluido el consumidor Electronics Show (CES), el Museo Miraikan y Honda Collection Hall, en Japón, y el festival Ars Electrónica en Austria.

En 1986, los ingenieros de Honda se propusieron crear un robot que camina. Los modelos tempranos (E1, E2, E3) se centraron en el desarrollo de las piernas que pueden simular el andar de un ser humano. La siguiente serie de modelos (E4, E5, E6) se han centrado en la estabilización de pie y subir escaleras. A continuación, se añadieron a la cabeza, el cuerpo y los brazos para el robot para mejorar el equilibrio y la funcionalidad. El primer robot humanoide de Honda, P1 era bastante resistente a su altura y peso. El P2 fue mejorado con un diseño más amigable, mejorado para caminar, subir y bajar escaleras y movimientos automáticos inalámbricos. El modelo P3 fue aún más compacto, y estable que sus predecesores.

ASIMO ha sido concebido para funcionar en un entorno real de la vida humana en el futuro próximo. Es fácil de manejar, tiene un tamaño y peso conveniente y puede moverse libremente dentro de las condiciones de vida humana, todo ello con un diseño amigable para la gente.

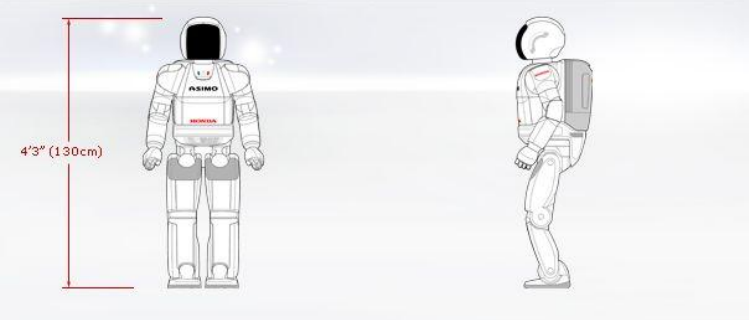


Imagen 46

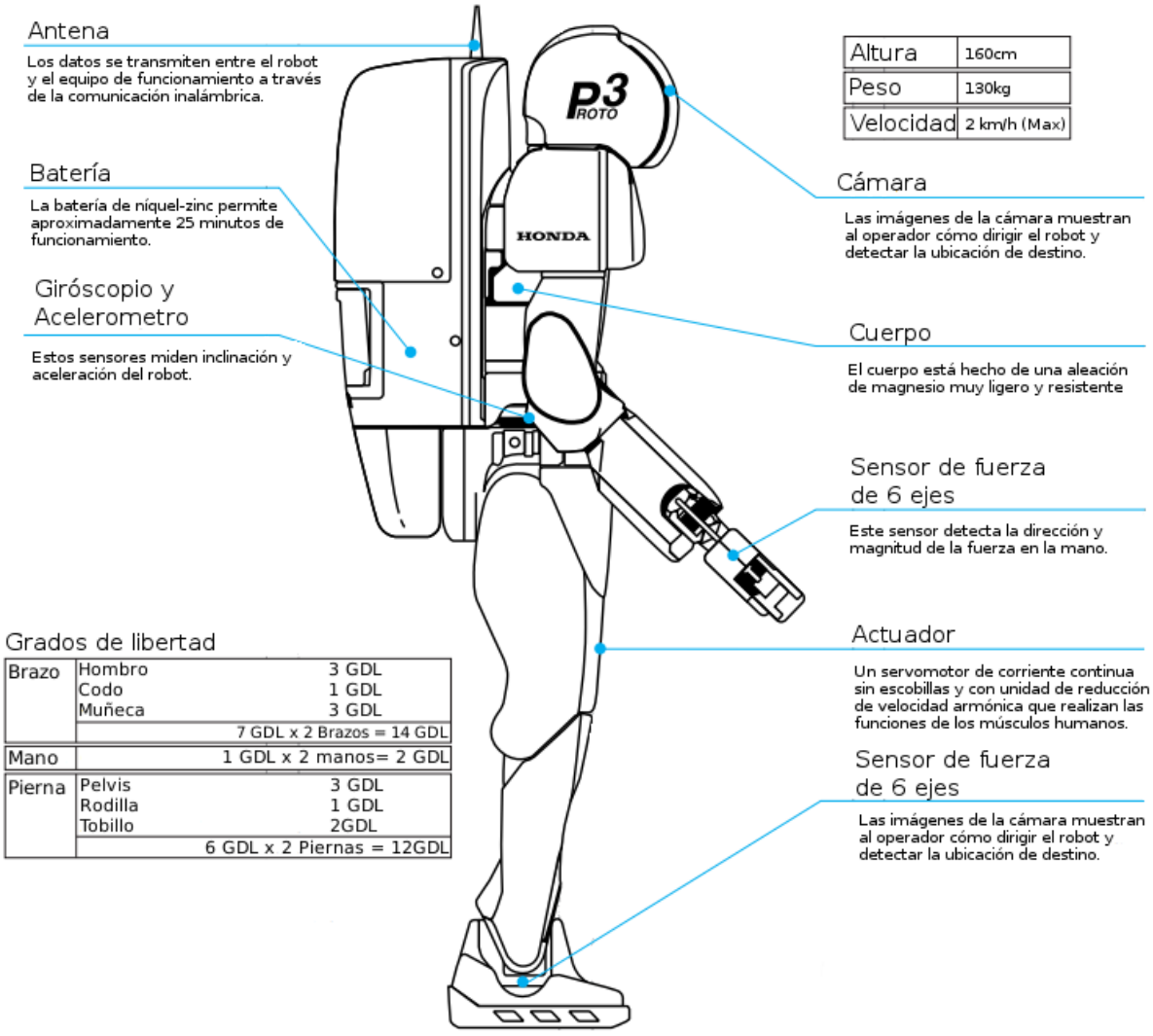


Imagen 47

Si bien al ASIMO está entre los robots androides más avanzados del mundo aun requiere de mucho desarrollo, pero aun así, uno de los principales museos de la ciencia de Japón lo ha pedido prestado como guía. Pero el bot confundido tiene muchas dificultades para distinguir entre las personas que levantan sus manos para hacer una pregunta y las otras que sólo desean tomar fotos.

En una demostración para la prensa en el Museo Miraikan, ASIMMO se congeló y repitió las preguntas que los clientes introdujeron en un dispositivo táctil. Los funcionarios del museo

admitieron que la máquina de control remoto es aun incapaz de responder a las preguntas verbales, o de diferenciar entre un niño y un adulto.

Pareció un espectáculo pobre para lo que es a menudo considerado como el robot humanoide más avanzado del planeta. "En este momento, se puede reconocer a un niño agitando la mano, pero no es capaz de comprender el significado de la ondulación" dijo el jefe de Honda del área de robótica, Satoshi Shigemi.

Desde su debut hace 13 años, ASIMO se ha convertido en robot más conocido de Japón, precedió la apertura de la Bolsa de Valores de Nueva York y ha hecho apariciones para audiencias de todo el mundo, incluso en el Museo de la Ciencia Británica. En 2011, sirvió té para Stephen Fry y bailó con Jo Brand durante un episodio de QI de la BBC. Pero frente a un trabajo de verdad, ASIMO demostró tristemente ser aun inadecuado.

En 2011, Honda considero enviar a ASIMO para ayudar a los científicos a solucionar las consecuencias de la catástrofe nuclear de Fukushima, que dejó atrás radiación mortal para los seres humanos. Pero los ingenieros decidieron que era un trabajo demasiado sensible y podría tropezar con escombros esparcidos por una serie de explosiones de hidrógeno.

Honda ha sido criticado por gastar dinero en lo que algunos llamaron un juguete caro (un portavoz de la compañía previamente se negó a especificar el costo del desarrollo de ASIMO, solo dijo que era menos de 1 millón de dólares). En respuesta a las críticas, Honda dijo este mes que ha creado un nuevo prototipo de ASIMO para inspeccionar la planta del reactor 2 que está paralizada, utilizando su capacidad de mapear un entorno.

El robot fue enviado para comprobar los niveles de radiación y las condiciones en el interior del edificio del reactor 2. Sin embargo, el prototipo, no se parecía en nada al humanoide parlante.

En condiciones cuidadosamente coreografiadas, ASIMO puede recorrer un camino, jugar al fútbol, servir bebidas, responder a unas 100 preguntas e incluso tocar el violín. Pero guías de museo de carne y hueso tienen poco que temer todavía. Honda dijo que uno de sus posibles aplicaciones futuras es ayudar a la gente a comprar boletos en las máquinas expendedoras - en otras palabras, una máquina que ayuda a un equipo. El objetivo del robot es ayudar a los seres humanos ", dicen desde Honda. ¿No se supone que debería hacer la vida más fácil? [web52].

8.2. Anexo B: Sensores y Cámaras

8.2.1. Sistema de Localizacion Hagisonic StarGazer

El Sistema de Localización Robot StarGazer Hagisonic es una solución para la localización en interiores de los robots móviles inteligentes. En él se analizan las imágenes de rayos infrarrojos reflejados desde un punto de referencia pasiva con características únicas. La salida de la posición y el ángulo de partida del robot se obtienen con una resolución muy precisa y gran velocidad. Es muy robusta en ambientes que contienen luces infrarrojas, fluorescencia o luz solar. Es uno de los mejores del mundo en el rendimiento, la comodidad y el costo.

Concepto básico:

• Como un barco navega por las estrellas, un robot móvil hace que un sistema autónomo de señales artificiales fijados en el techo.

• Cada señal tiene un número de identificación independiente y la información sobre la posición y el ángulo de un robot autónomo.

Imagen 48

Cómo funciona:

1) El proyector IR en StarGazer dispara rayos infrarrojos a los puntos de referencia en el techo.

2) A continuación, los puntos de referencia reflejan los rayos a la StarGazer en la parte superior de un robot.

3) Al mismo tiempo, los rayos infrarrojos reflejados se convierten en una imagen utilizando la cámara CMOS StarGazer.

4) A través de procesamiento de imagen digital, StarGazer calcula la posición y el ángulo de un robot mediante el análisis de la imagen adquirida.

8.2.2. CMUcam2

La cámara CMUcam2 consiste en un microcontrolador SX52 (http://www.ubicom.com/products/sx/sx.html) interconectado con una cámara Omnivisión OV6620 o OV7620 CMOS (http://www.ovt.com) en un chip que permite que los datos simples de alto nivel puedan extraerse de la transmisión de vídeo de la cámara. La placa se comunica a través de un chip RS-232 o un puerto serie TTL y tiene las siguientes funciones:

- Personalización por el usuario los colores y la cantidad fotogramas por segundo (maximo de 50).
- Reconocimiento de movimiento por diferenciación de imagenes cuadro a a cuadro a 26 fotogramas por segundo
- Encontrar el centro de gravedad de los datos de seguimiento
- Reunir datos y calcular la media de color y la variación de datos
- calcular un histograma de cada canal de color
- Transferencia de un mapa de bits binario en tiempo real de los píxeles en una imagen
- Ajuste de las características de la imagen de la cámara
- Volcado de una imagen RAW (canales simples o múltiples, de resolución hasta 160 x 255)
- Compatible con múltiples velocidades de transmisión
- Control de servo de 5 salidas
- Modo esclavo paralelo de procesamiento de imágenes de un solo bus de cámara
- seguimiento automática de los servos de ejes de color
- Salida de vídeo analógico B / W (PAL o NTSC)
- Personalización flexible de paquetes de salida

Imagen 49

Uno de los principales usos de la CMUcam2 es el seguimiento o monitoreo del color. El mejor rendimiento se puede lograr cuando hay colores contrastantes y muy intensos. Por ejemplo, puede fácilmente rastrear una bola roja sobre un fondo blanco, pero sería difícil diferenciar entre diferentes tonos de marrón en el cambio de luz. El seguimiento de objetos de colores se puede utilizar para localizar puntos de referencia, seguir líneas, o perseguir un objetivo móvil. Haciendo uso de las estadísticas del color, es posible supervisar una escena, detectar un color específico o hacer una detección de movimiento primitiva. Si la cámara detecta un cambio de color drástico, entonces es probable que algo en la escena haya cambiado. Con uso del "modo de línea", el CMUcam2 puede de manera fácil obtener imágenes binarias de baja resolución de los objetos de colores. Esto se puede utilizar para hacer un seguimiento de línea más sofisticada que incluye la detección de bordes, o reconocimiento, incluso de formas simples. Estas operaciones más avanzadas requerirían algoritmos personalizados que aumente[web53].

El siguiente diagrama muestra la disposición de los componentes del CMUcam2:

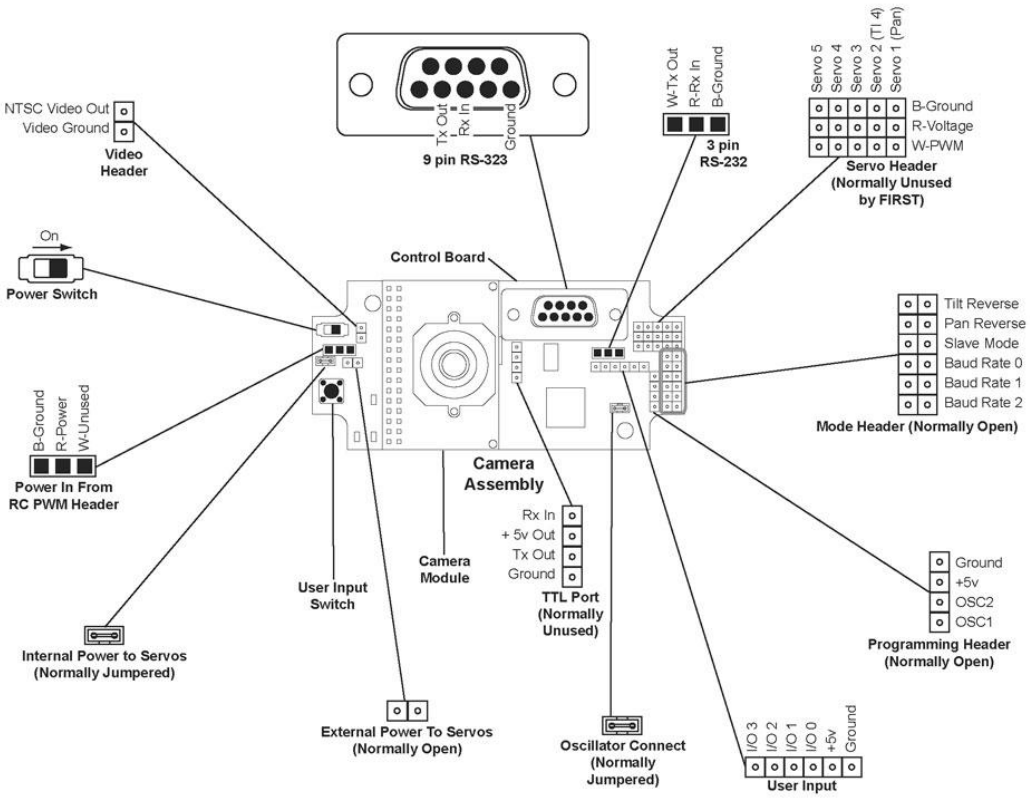


Imagen 50

Una ampliación mas especifica del uso de la CMUcam2 es la interfaz CMUcam2GUIStereo que es una interfaz de usuario creada a partir de la CMUcam2GUI y adaptada para la utilización de dos cámaras en simultáneo para permitir la simulación de la visión estereoscópica.

8.2.3. Raspberry Pi

En implementaciones pequeñas o para dividir el procesamiento de diferentes partes del robot es muy útil contar con computadoras de pequeño tamaño. Una de las implementaciones más recientes es la Raspberry Pi que se trata de una computadora del tamaño de una tarjeta de crédito y que incluye varias interfaces de usuario (USB, Video Output, puerto Ethernet, Almacenamiento SD) [web54].

Imagen 51

El Modelo más reciente Raspberry Pi modelo b tiene las siguientes características:

- Chip: Broadcom BCM2835 SoC full HD multimedia applications processor
- CPU: 700 MHz Low Power ARM1176JZ-F Applications Processor
- GPU: Dual Core VideoCore IV®, Multimedia Co-Processor
- Memory: 512MB SDRAM
- Ethernet: onboard 10/100 Ethernet RJ45 jack
- USB 2.0: Dual USB Connector
- Video Output: HDMI y Composite RCA (PAL and NTSC)
- Audio Output: 3.5mm jack, HDMI
- Onboard Storage: SD, MMC, SDIO card slot
- Operating System: Linux
- Dimensions: 8.6cm x 5.4cm x 1.7cm

8.3. Anexo C: Códigos Útiles

8.3.1. OpenCV

Para el primer ejemplo de código fuente, se presenta un simple código de reconocimiento de rostros[web55]:

```
/*
 * Copyright (c) 2011. Philipp Wagner <bytefish[at]gmx[dot]de>.
 * Released to public domain under terms of the BSD Simplified license.
 *
 *   See <http://www.opensource.org/licenses/bsd-license>
 */

#include "opencv2/core/core.hpp"
#include "opencv2/contrib/contrib.hpp"
#include "opencv2/highgui/highgui.hpp"

#include <iostream>
#include <fstream>
#include <sstream>

using namespace cv;
using namespace std;

static Mat norm_0_255(InputArray _src) {
    Mat src = _src.getMat();
    // Create and return normalized image:
    Mat dst;
    switch(src.channels()) {
    case 1:
        cv::normalize(_src, dst, 0, 255, NORM_MINMAX, CV_8UC1);
        break;
    case 3:
        cv::normalize(_src, dst, 0, 255, NORM_MINMAX, CV_8UC3);
        break;
    default:
        src.copyTo(dst);
        break;
    }
    return dst;
}

static void read_csv(const string& filename, vector<Mat>& images, vector<int>& labels,
char separator = ';') {
    std::ifstream file(filename.c_str(), ifstream::in);
    if (!file) {
        string error_message = "No valid input file was given, please check the given
filename.";
        CV_Error(CV_StsBadArg, error_message);
    }
    string line, path, classlabel;
```

```
        while (getline(file, line)) {
            stringstream liness(line);
            getline(liness, path, separator);
            getline(liness, classlabel);
            if(!path.empty() && !classlabel.empty()) {
                images.push_back(imread(path, 0));
                labels.push_back(atoi(classlabel.c_str()));
            }
        }
    }

    int main(int argc, const char *argv[]) {
        // Check for valid command line arguments, print usage
        // if no arguments were given.
        if (argc < 2) {
            cout << "usage: " << argv[0] << " <csv.ext> <output_folder> " << endl;
            exit(1);
        }
        string output_folder;
        if (argc == 3) {
            output_folder = string(argv[2]);
        }
        // Get the path to your CSV.
        string fn_csv = string(argv[1]);
        // These vectors hold the images and corresponding labels.
        vector<Mat> images;
        vector<int> labels;
        // Read in the data. This can fail if no valid
        // input filename is given.
        try {
            read_csv(fn_csv, images, labels);
        } catch (cv::Exception& e) {
            cerr << "Error opening file \"" << fn_csv << "\". Reason: " << e.msg << endl;
            // nothing more we can do
            exit(1);
        }
        // Quit if there are not enough images for this demo.
        if(images.size() <= 1) {
            string error_message = "This demo needs at least 2 images to work. Please add
more images to your data set!";
            CV_Error(CV_StsError, error_message);
        }
        // Get the height from the first image. We'll need this
        // later in code to reshape the images to their original
        // size:
        int height = images[0].rows;
        // The following lines simply get the last images from
        // your dataset and remove it from the vector. This is
        // done, so that the training data (which we learn the
        // cv::FaceRecognizer on) and the test data we test
        // the model with, do not overlap.
        Mat testSample = images[images.size() - 1];
        int testLabel = labels[labels.size() - 1];
```

```
            images.pop_back();
            labels.pop_back();
            // The following lines create an Fisherfaces model for
            // face recognition and train it with the images and
            // labels read from the given CSV file.
            // If you just want to keep 10 Fisherfaces, then call
            // the factory method like this:
            //
            //      cv::createFisherFaceRecognizer(10);
            //
            // However it is not useful to discard Fisherfaces! Please
            // always try to use _all_ available Fisherfaces for
            // classification.
            //
            // If you want to create a FaceRecognizer with a
            // confidence threshold (e.g. 123.0) and use _all_
            // Fisherfaces, then call it with:
            //
            //      cv::createFisherFaceRecognizer(0, 123.0);
            //
            Ptr<FaceRecognizer> model = createFisherFaceRecognizer();
            model->train(images, labels);
            // The following line predicts the label of a given
            // test image:
            int predictedLabel = model->predict(testSample);
            //
            // To get the confidence of a prediction call the model with:
            //
            //      int predictedLabel = -1;
            //      double confidence = 0.0;
            //      model->predict(testSample, predictedLabel, confidence);
            //
            string result_message = format("Predicted class = %d / Actual class = %d.",
predictedLabel, testLabel);
            cout << result_message << endl;
            // Here is how to get the eigenvalues of this Eigenfaces model:
            Mat eigenvalues = model->getMat("eigenvalues");
            // And we can do the same to display the Eigenvectors (read Eigenfaces):
            Mat W = model->getMat("eigenvectors");
            // Get the sample mean from the training data
            Mat mean = model->getMat("mean");
            // Display or save:
            if(argc == 2) {
               imshow("mean", norm_0_255(mean.reshape(1, images[0].rows)));
            } else {
               imwrite(format("%s/mean.png",                                output_folder.c_str()),
norm_0_255(mean.reshape(1, images[0].rows)));
            }
            // Display or save the first, at most 16 Fisherfaces:
            for (int i = 0; i < min(16, W.cols); i++) {
               string msg = format("Eigenvalue #%d = %.5f", i, eigenvalues.at<double>(i));
               cout << msg << endl;
               // get eigenvector #i
```

```
            Mat ev = W.col(i).clone();
            // Reshape to original size & normalize to [0...255] for imshow.
            Mat grayscale = norm_0_255(ev.reshape(1, height));
            // Show the image & apply a Bone colormap for better sensing.
            Mat cgrayscale;
            applyColorMap(grayscale, cgrayscale, COLORMAP_BONE);
            // Display or save:
            if(argc == 2) {
                imshow(format("fisherface_%d", i), cgrayscale);
            } else {
                imwrite(format("%s/fisherface_%d.png", output_folder.c_str(), i), norm_0_255(cgrayscale));
            }
        }
        // Display or save the image reconstruction at some predefined steps:
        for(int num_component = 0; num_component < min(16, W.cols); num_component++) {
            // Slice the Fisherface from the model:
            Mat ev = W.col(num_component);
            Mat projection = subspaceProject(ev, mean, images[0].reshape(1,1));
            Mat reconstruction = subspaceReconstruct(ev, mean, projection);
            // Normalize the result:
            reconstruction = norm_0_255(reconstruction.reshape(1, images[0].rows));
            // Display or save:
            if(argc == 2) {
                imshow(format("fisherface_reconstruction_%d", num_component), reconstruction);
            } else {
                imwrite(format("%s/fisherface_reconstruction_%d.png", output_folder.c_str(), num_component), reconstruction);
            }
        }
        // Display if we are not writing to an output folder:
        if(argc == 2) {
            waitKey(0);
        }
        return 0;
    }
```

Otra aplicación muy interesante para ser realizada con OpenCV es el reconocimiento de movimiento, a continuación se describe el código fuente[web56]:

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <time.h>
#include <dirent.h>
#include <sstream>
#include <dirent.h>
#include <sys/types.h>
#include <sys/stat.h>

using namespace std;
using namespace cv;

bool directoryExists( const char* pzPath )
{
    if ( pzPath == NULL) return false;
    DIR *pDir;
    bool bExists = false;
    pDir = opendir (pzPath);
    if (pDir != NULL)
    {
        bExists = true;
        (void) closedir (pDir);
    }
    return bExists;
}
```

bool saveImg(Mat image, const string DIRECTORY, const string EXTENSION, const char * DIR_FORMAT, const char * FILE_FORMAT){

```
    stringstream ss;
    time_t seconds;
    struct tm * timeinfo;
    char TIME[80];

    time (&seconds);
    timeinfo = localtime (&seconds);

    // convert dir...
    strftime (TIME,80,DIR_FORMAT,timeinfo);
    ss.str("");
    ss << DIRECTORY << TIME;

    if(!directoryExists(ss.str().c_str()))
        mkdir(ss.str().c_str(), 0777);

    // convert image name
    strftime (TIME,80,FILE_FORMAT,timeinfo);
    ss.str("");
    ss << DIRECTORY << TIME << EXTENSION;
```

```
    // save image
    return imwrite(ss.str().c_str(), image);
}

int main (int argc, char * const argv[]){

    // const
    const string DIR = "/Users/cedric/Desktop/OpenCVTester/pics/";
    const string EXT = ".jpg";
    const int DELAY = 700; // mseconds

    string DIR_FORMAT = "%d%h%Y";
    string FILE_FORMAT = DIR_FORMAT + "/" + "%d%h%Y_%H%M%S";

    // create all necessary instances
    CvCapture * camera = cvCaptureFromCAM(CV_CAP_ANY);
    Mat original =  cvQueryFrame(camera);
    Mat next_frame = original;
    Mat current_frame = cvQueryFrame(camera);
    Mat prev_frame = cvQueryFrame(camera);

    cvtColor(current_frame, current_frame, CV_RGB2GRAY);
    cvtColor(prev_frame, prev_frame, CV_RGB2GRAY);
    cvtColor(next_frame, next_frame, CV_RGB2GRAY);

    Mat d1, d2, result;
    int window = 200;
    bool movement;
    while (true){

        movement = false;
        absdiff(next_frame, current_frame, d1);
        absdiff(current_frame, prev_frame, d2);
        bitwise_xor(d1, d2, result);

        int middle_y = result.rows/2;
        int middle_x = result.cols/2;

        // Center window
        threshold(result, result, 140, 255, CV_THRESH_BINARY);
        for(int i = middle_x-window; i < middle_x+window; i++)
            for(int j = middle_y-window; j < middle_y+window; j++)
                if(result.at<int>(j,i)>0)
                {
                    movement = true;
                    break;
                }

        if(movement==true)
            saveImg(original,DIR,EXT,DIR_FORMAT.c_str(),FILE_FORMAT.c_str());

        imshow("Motion", result);
```

```
        prev_frame = current_frame;
        current_frame = next_frame;

        // get image from webcam
        next_frame = cvQueryFrame(camera);
        cvtColor(next_frame, next_frame, CV_RGB2GRAY);

        // semi delay and quit when press Q/q
        int key = cvWaitKey (DELAY);
        if (key == 'q' || key == 'Q')
            break;
    }

        return 0;
}
```

9. Glosario

- **Algoritmo:** Un algoritmo (del griego y latín, dixit algorithmus) es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad.

- **Ambiente:** Condiciones o circunstancias físicas, humanas, etc., que rodean a las personas, robots, etc.

- **Camino (Planificación):** Recorrido que se ha creado para que el robot pueda lograr su objetivo y evitar las colisiones.

- **Cinemática:** La cinemática (del griego kineo, movimiento) es la rama de la física que estudia las leyes del movimiento de los cuerpos sin considerar más causas que lo originan (las fuerzas) y se limita, esencialmente, al estudio de la trayectoria en función del tiempo.

- **Colisión (Robótica):** Choque entre dos cuerpos, siendo estos cuerpos el robot y el obstáculo.

- **Comandos (Informática):** Un comando (derivado del inglés command) es una instrucción u orden que el usuario proporciona a un sistema informático.

- **Conjunto (Matemática):** En matemáticas, un conjunto es una agrupación de objetos considerada como un objeto en sí.

- **Derivada (Matemática):** En matemáticas, la derivada de una función es una medida de la rapidez con la que cambia el valor de dicha función matemática, según cambie el valor de su variable independiente. La derivada de una función es un concepto local, es decir, se calcula como el límite de la rapidez de cambio media de la función en un cierto intervalo, cuando el intervalo considerado para la variable independiente se toma cada vez más pequeño. Por ello se habla del valor de la derivada de una cierta función en un punto dado.

- **Dirección:** Rumbo u orientación que un cuerpo sigue en su movimiento.

- **Distribución (probabilidad):** Una distribución de probabilidad indica toda la gama de valores que pueden representarse como resultado de un experimento si éste se llevase a cabo. Es decir, describe la probabilidad de que un evento se realice en el futuro, constituye una herramienta fundamental para la prospectiva, puesto que se puede diseñar un escenario de acontecimientos futuros considerando las tendencias actuales de diversos fenómenos naturales.

- **Emular:** Imitar las acciones de otro procurando igualarlo o superarlo.

- **Estadística:** Es una ciencia formal que estudia la recolección, análisis e interpretación de datos de una muestra representativa, ya sea para ayudar en la toma de decisiones o para explicar condiciones regulares o irregulares de algún fenómeno o estudio aplicado, de ocurrencia en forma aleatoria o condicional.
- **Estocástico:** Se denomina estocástico (del latín stochasticus, "hábil en conjeturar") al sistema cuyo comportamiento es intrínsecamente no determinístico. Un proceso estocástico es aquel cuyo comportamiento es no determinista, en la medida que el subsiguiente estado del sistema está determinado tanto por las acciones predecibles del proceso como por elementos aleatorios.

- **Fotograma:** Cada una de las imágenes que se suceden en una película cinematográfica consideradas de forma aislada

- **Giroscopio:** Es un dispositivo mecánico que sirve para medir, mantener o cambiar la orientación en el espacio de algún aparato o vehículo.

- **GPS:** Global Positioning System: sistema de posicionamiento global, es un sistema global de navegación por satélite (GNSS) que permite determinar en todo el mundo la posición de un objeto, una persona o un vehículo con una precisión hasta de centímetros (si se utiliza GPS diferencial), aunque lo habitual son unos pocos metros de precisión.

- **Gradiente (Matemática):** Intensidad de aumento o disminución de una magnitud variable, y curva que lo representa.

- **Grado de libertad:** El número de grados de libertad en ingeniería se refiere al número mínimo de parámetros que necesitamos especificar para determinar completamente la velocidad de un mecanismo o el número de reacciones de una estructura. En estadística, los grados de libertad están dados por el número de valores que pueden ser asignados de forma arbitraria, antes de que el resto de las variables tomen un valor automáticamente, producto de establecerse las que son libres.

- **Grafo:** Un grafo (del griego grafos: dibujo, imagen) es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto.

- **Hibrido:** En la programación software, sistemas híbridos denotan a los sistemas de software que emplean, en paralelo, una combinación de modelos, métodos y técnicas de distintos subcampos.

- **Humanoide:** El término humanoide se refiere a cualquier ser cuya estructura corporal se asemeja a la de un humano.

- **Linealización:** Conversión de un sistema no-lineal en uno lineal haciendo una transformación apropiada de sus variables.

- **Matriz:** Una matriz es un arreglo bidimensional de números, que se usan generalmente para describir sistemas de ecuaciones lineales, sistemas de ecuaciones diferenciales o representar una aplicación lineal (dada una base).

- **Modelo (Matemático):** En ciencias aplicadas, un modelo matemático es uno de los tipos de modelos científicos que emplea algún tipo de formulismo matemático para expresar relaciones, proposiciones sustantivas de hechos, variables, parámetros, entidades y relaciones entre variables y/o entidades u operaciones, para estudiar comportamientos de sistemas complejos ante situaciones difíciles de observar en la realidad.

- **Objeto / Obstáculo:** Cada una de las barreras físicas que presenta una pista o un recorrido:

- **Posición:** La posición de un objeto indica su localización en el espacio o en el espacio-tiempo. Se representa mediante sistemas de coordenadas.

- **Retroalimentación:** La retroalimentación, cuyo término correcto es realimentación (en inglés feedback) es un mecanismo de control de los sistemas dinámicos por el cual una cierta proporción de la señal de salida se redirige a la entrada, y así regula su comportamiento.

- **Ruido (Matemático):** Es una señal aleatoria (proceso estocástico) que se caracteriza por el hecho de que sus valores de señal en dos tiempos diferentes no guardan correlación estadística.

- **Sensor:** Un sensor es un dispositivo capaz de detectar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. Las variables de instrumentación pueden ser por ejemplo: temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, movimiento, pH, etc.

10. Bibliografía

Capitulo 1

[RAE01]	Diccionario de la Real academia española, Edición web (http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=robot)
[ECI01]	ECI 2011 (Escuela de ciencias Informáticas, UBA 2011) "Introducción a la Robótica móvil", Profesor: Miroslav Kulich
[WIKI01]	http://es.wikipedia.org/wiki/Visi%C3%B3n_artificial

Capitulo 2

[Imagen 2]	Aibo: Robot Mascota Sony (http://www.sonyaibo.net/home.htm)
[Imagen 3]	ASIMO: Humanoide Honda (http://world.honda.com/ASIMO/)
[WEB01]	http://vedcc.org/Robotics/itec424/lec3-robot_technology.pdf
[WEB02]	http://robotec11.tripod.com/id4.html
[Tabla 1]	http://www.used-robots.com/robot-education.php?page=robot+timeline http://es.wikipedia.org/wiki/Programa_Lunojod http://en.wikipedia.org/wiki/Spirit_rover http://en.wikipedia.org/wiki/Opportunity_rover http://es.wikipedia.org/wiki/Curiosity
[WEB03]	http://forums.parallax.com/attachment.php?attachmentid=40183&d=1138274073
[Imagen 4]	http://www.publicaciones.urbe.edu/index.php/telematique/article/viewArticle/833/2037
[WEB04]	Topological mapping with sensing-limited robots (Kris Beevers) - Rensselaer Polytechnic Institute Algorithmic Robotics Laboratory, Troy, NY, April 2004.
[Imagen 5]	Topological mapping with sensing-limited robots (Kris Beevers) - Rensselaer Polytechnic Institute Algorithmic Robotics Laboratory, Troy, NY, April 2004.
[Imagen 6]	http://www.arpith.com/index.php/2004/10/30/Mobile-Robotics-Simple-Mapping.html
[Imagen 7]	http://www.cimat.mx/~jbhayet/CLASES/VISIONROB/clase5.pdf
[WEB05]	http://www.cimat.mx/~jbhayet/CLASES/VISIONROB/clase5.pdf
[WIKI02]	http://en.wikipedia.org/wiki/Motion_planning
[WEB06]	http://www.info-ab.uclm.es/robotica/material/martes1.pdf

[WEB07]	http://en.wikipedia.org/wiki/Monte_Carlo_method
[WEB08]	http://en.wikipedia.org/wiki/Monte_Carlo_localization
[Imagen 8]	http://www.cs.washington.edu/robotics/mcl/animations/global-floor.gif
[WEB09]	http://es.wikipedia.org/wiki/Teorema_de_Bayes
[WEB10]	www.aaai.org/Papers/AAAI/2000/AAAI00-132.pdf Monte Carlo Localization With Mixture Proposal Distribution (Thrun, Fox, Burgard, 2000)
[Imagen 9]	http://swarmlab.unimaas.nl/wp-content/uploads/2012/07/fox2003bayesian.pdf Bayesian Filters for Location Estimation (Fox, Hightower, Liao, Schulz, Borriello - 2003)
[WEB11]	http://www.damas.ift.ulaval.ca/_seminar/filesA11/10.1.1.107.7415.pdf Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond (ZHE CHEN)
[Imagen 10]	http://www.robotshop.com/content/images/hagisonic-stargazer-localization-sensor-detail.jpg
[WEB12]	www.aaai.org/Papers/AAAI/2000/AAAI00-132.pdf Monte Carlo Localization With Mixture Proposal Distribution (Thrun, Fox, Burgard, 2000)
[Imagen 11]	www.aaai.org/Papers/AAAI/2000/AAAI00-132.pdf Monte Carlo Localization With Mixture Proposal Distribution (Thrun, Fox, Burgard, 2000)
[WEB13]	Localization in the ensemble Kalman Filter (Ruth Elizabeth Petrie, August 2008)
[WEB14]	Robot Localization and Kalman Filter (Rudy Negenborn, 2003)
[Imagen 12]	http://www.slideshare.net/artintelligence/markov-localization-and-kalman-filter-localization
[WEB15]	CMRoboBits: Creating an Intelligent AIBO Robot (Paul E. Rybski)
[WEB16]	COE 584: Robotics - KFUPM (Ahmad Salam AlRefai, Mohammad Shahab) http://staff.kfupm.edu.sa/SE/mshahab/072_COE584_prsnt2_ShahabSalam.pdf
[Imagen 13]	Robot localization and kalman Filters (Rudy Negenborn) http://www.negenborn.net/kal_loc/kal_loc.pdf
[WEB17]	Markov Localization for mobile Robots in Dynamic Environments (D. Fox, W. Burgard, S. Thrun)

[WEB18]	Convolution http://en.wikipedia.org/wiki/Convolution
[WEB19]	Hidden Markov models for radio localization (C. Morelli, M. Nicoli, V. Rampa, U. Spagnolini)
[Algoritmo01]	Hidden Markov models for radio localization (C. Morelli, M. Nicoli, V. Rampa, U. Spagnolini)
[Imagen 14]	Hidden Markov models for radio localization (C. Morelli, M. Nicoli, V. Rampa, U. Spagnolini)

Capitulo 4

[WEB20]	Robotic Mapping, Sebastian Thrun, February 2002 http://174.120.175.51/robottheory/Robotic_Mapping_A_Survey.pdf
[Imagen 15]	Simultaneous Localization and Mapping (HUGH DURRANT-WHYTE AND TIM BAILEY)
[WEB21]	Simultaneous Localization and Mapping (HUGH DURRANT-WHYTE AND TIM BAILEY)
[WEB22]	Autonomous Mobile Robots (Shuzhi Sam Ge, Frank L Lewis)
[Imagen 16]	Autonomous Mobile Robots (Shuzhi Sam Ge, Frank L Lewis)
[Algoritmo02]	Autonomous Mobile Robots (Shuzhi Sam Ge, Frank L Lewis)
[Imagen 17]	http://www.cs.utexas.edu/~kuipers/slides/L19-topological-mapping.pdf
[WEB23]	http://www.cis.upenn.edu/~kostas/mypub.dir/anati09nips.pdf
[Algoritmo03]	http://www.cis.upenn.edu/~kostas/mypub.dir/anati09nips.pdf
[WEB24]	http://www.cs.toronto.edu/~zemel/documents/unsup-fieldRobot.pdf
[WEB25]	http://www.cis.upenn.edu/~kostas/mypub.dir/anati09nips.pdf
[WEB26]	Topological mapping with sensing-limited robots (K. Beevers, Rensselaer Polytechnic Institute)
[Imagen 18]	Topological mapping with sensing-limited robots (K. Beevers, Rensselaer Polytechnic Institute)
[Imagen 19]	Topological mapping with sensing-limited robots (K. Beevers, Rensselaer Polytechnic Institute)
[Imagen 20]	Topological mapping with sensing-limited robots (K. Beevers, Rensselaer Polytechnic Institute)

[Imagen 21]	Topological mapping with sensing-limited robots (K. Beevers, Rensselaer Polytechnic Institute)
[Imagen 22]	Topological mapping with sensing-limited robots (K. Beevers, Rensselaer Polytechnic Institute)
[WEB27]	Integrating Topological and Metric Maps for Mobile Robot Navigation: A Statistical Approach, Thrun, Gutmann, Fox, Burgard, Kuipers
[Imagen 23]	Integrating Topological and Metric Maps for Mobile Robot Navigation: A Statistical Approach, Thrun, Gutmann, Fox, Burgard, Kuipers
[Imagen 24]	Integrating Topological and Metric Maps for Mobile Robot Navigation: A Statistical Approach, Thrun, Gutmann, Fox, Burgard, Kuipers

Capitulo 5

[WEB28]	Robotics - Robotics Planning http://www.electronicsteacher.com/robotics/robotics-planning.php
[Imagen 25]	Robot Planning Drew McDermott (1992)
[WEB29]	International journal of systems applications, engineering & development Issue 4, Volume 2, 2008 (http://www.naun.org/multimedia/UPress/saed/saed-45.pdf)
[WEB30]	Planning Algorithms By Steven M. LaValle, Copyright 2006
[WEB31]	Planning Algorithms By Steven M. LaValle, Copyright 2006
[Imagen 26]	Planning Algorithms By Steven M. LaValle, Copyright 2006
[WEB32]	http://www.cs.cmu.edu/afs/cs/academic/class/15381-s07/www/slides/020807motion.pdf
[Imagen 27]	http://www.cs.cmu.edu/afs/cs/academic/class/15381-s07/www/slides/020807motion.pdf
[WEB33]	http://whatis.techtarget.com/definition/degrees-of-freedom
[WEB34]	Planning Algorithms By Steven M. LaValle, Copyright 2006
[Imagen 28]	Planning Algorithms By Steven M. LaValle, Copyright 2006
[WEB35]	Combinatorial Motion Planning of Multiple Vehicle Systems, Malik, W. ; Dept. of Mech. Eng., Texas A&M Univ., College Station, TX.
[Imagen 29]	Planning Algorithms By Steven M. LaValle, Copyright 2006
[WEB36]	Continual HTN Robot Task Planning in Open-Ended Domains: A Case Study Dominik Off and Jianwei Zhang

[WEB37]	Robot Task Planning using Semantic Maps - C. Galido, J. Fernández-Madrigal, J. Gonzalez

Capitulo 6

[Imagen 30]	R. M. Haralick and L. G. Shapiro, “Glossary of Computer Vision Terms,” Pattern Recognition, 1991.
[WEB38]	IMAGE PROCESSING TECHNIQUES FOR MACHINE VISION Alberto Martin and Sabri Tosunoglu
[WEB39]	http://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic3.htm
[Imagen 31]	http://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic3.htm
[WEB40]	IMAGE PROCESSING TECHNIQUES FOR MACHINE VISION Alberto Martin and Sabri Tosunoglu
[Imagen 32]	Edge Detection Techniques - An Overview Djemel Ziou, Salvatore Tabbon
[WEB41]	Digital Image Processing, Second Edition (Rafael C. Gonzalez - Richard E. Woods)
[Imagen 34]	http://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm
[WEB42]	Digital Image Processing, Second Edition (Rafael C. Gonzalez - Richard E. Woods)
[Imagen 35]	Digital Image Processing, Second Edition (Rafael C. Gonzalez - Richard E. Woods)
[Imagen 36]	Digital Image Processing, Second Edition (Rafael C. Gonzalez - Richard E. Woods)
[Imagen 37]	Digital Image Processing, Second Edition (Rafael C. Gonzalez - Richard E. Woods)
[Imagen 38]	Digital Image Processing, Second Edition (Rafael C. Gonzalez - Richard E. Woods)
[Imagen 39]	Digital Image Processing, Second Edition (Rafael C. Gonzalez - Richard E. Woods)
[WEB43]	Skeletons in Digital Image Processing (Gisela Klette)
[WEB44]	Eficient computation of various types of skeletons (Luc Vincent)
[WEB45]	http://www.scs.ryerson.ca/mfiala/robotics_club/robot_projects/riddle_dalek/stereo_dalek.htm
[WEB46]	http://www.mathworks.com/products/computer-vision/examples.html?file=/products/demos/shipping/vision/videostereo.html
[Imagen 40]	http://www.cs.unc.edu/~lazebnik/spring09/lec14_multiview_stereo.pdf
[WEB47]	http://www.vcl.fer.hr/papers_pdf/Accomplishments%20and%20%20Challenges%20of%20Computer%20Stereo%20Vision.pdf

[Imagen 40]	http://www.vcl.fer.hr/papers_pdf/Accomplishments%20and%20%20Challenges%20of%20Computer%20Stereo%20Vision.pdf
[WEB48]	http://www.vcl.fer.hr/papers_pdf/Accomplishments%20and%20%20Challenges%20of%20Computer%20Stereo%20Vision.pdf

Anexos

[Imagen 42]	http://mars.jpl.nasa.gov/msl/mission/rover/body/
[WEB49]	http://www.jpl.nasa.gov/news/fact_sheets/mars-science-laboratory.pdf
[WEB50]	http://mars.jpl.nasa.gov/msl/mission/rover/brains/
[Imagen 43]	http://mars.jpl.nasa.gov/msl/mission/rover/body/
[WEB51]	http://mars.jpl.nasa.gov/msl/mission/rover/brains/
[Imagen 44]	http://en.wikipedia.org/wiki/Curiosity_(rover)#Specifications
[Imagen 45]	http://mars.jpl.nasa.gov/msl/mission/rover/body/
[Imagen 46]	http://asimo.honda.com/downloads/pdf/asimo-technical-information.pdf
[Imagen 47]	http://asimo.honda.com/default.aspx
[WEB52]	hhttp://www.independent.co.uk/life-style/gadgets-and-tech/news/worlds-bestloved-robot-flunks-its-first-proper-job-asimo-flustered-by-visitors-gestures-and-questions-while-working-as-museum-guide-8686101.html
[Imagen 48]	Localization system StarGazer for Intelligent Robots User's Guide
[WEB53]	http://www.cs.cmu.edu/~cmucam2/CMUcam2_manual.pdf
[Imagen 49]	http://www.cs.cmu.edu/~cmucam2/CMUcam2_manual.pdf
[Imagen 50]	http://www.kevin.org/frc/camera/first_cmucam2.jpg
[Imagen 51]	http://downloads.element14.com/raspberryPi1.html
[WEB54]	http://downloads.element14.com/raspberryPi1.html
[WEB55]	http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html
[WEB56]	http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html