CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

DEPARTAMENTO DE CONTROL AUTOMÁTICO

Autolocalización de robots utilizando VSLAM mediante formulación por grafos y relocalización a través de regresión de imágenes y odometría.

Tesis que presenta

Ricardo Carrillo Mendoza

Para obtener el grado de

Maestro en Ciencias

En la especialidad de

Control Automático

Director de la tesis: Juan Manuel Ibarra Zannatha

DF, México Diciembre 2015

Ricardo Carrillo Mendoza: *Autolocalización de robots utilizando VSLAM y Relocalización,* Mapeo y Autolocalización para robts humanoides, © Noviembre 2015

Dedico esta tésis a aquellas personas que siempre han creído en mi, esta tésis es para mi mejor amiga, confidente, compañera y amor de mi vida, Karina.
Para mi familia, pilar de mi fortaleza y mi motivo de superación constante.

ABSTRACT

The ambience of research in VSLAM and relocalization algorithms in the last few years allure with real-time localization and increased precision for RGBD or stereo cameras but with ambivalence requesting higher computational capacity or more expensive sensors.

The aim of this work is to present a gentle algorithm to locate a humanoid robot using relocalization view based algorithms and odometry information using general regression with Nadayara-Watson kernel for applications where the area is already known, such as, RoboCup competitions or service robots

In this thesis also an investigation and analysis is made using a graph SLAM approach in order to open a new research area to improve the data association hypothesis as part of a back-end SLAM process. A link between PTAM as front-end SLAM and g^2o as backend SLAM is presented in order to stablish the research platform for experimentation.

Los grandes espíritus siempre han encontrado violenta oposición de parte de los mediocres.
Estos últimos no pueden entender cuando un hombre no sucumbe impensadamente a prejuicios hereditarios sino que, honestamente y con coraje, usa su inteligencia.
— Albert Einstein 1950.

AGRADECIMIENTOS

La conclusión de este trabajo no hubiera sido posible sin todas las personas que me estiman y apoyan mi carrera, agradezco a todos los que colaboraron de manera directa o indirecta en mi desarrollo como profesionista.

Muchas gracias mi futura esposa, por tu sacrificio, esfuerzo, y cariño, por darme aliento siempre que lo necesité, por tus consejos y tu incansable felicidad, sin ti, nada de esto cobraría sentido.

Agradezco a mi familia, siempre alentandome a seguir mis sueños y apoyandome en los momentos difíciles aún cuando los de ellos son más difíciles. Gracias por todos los sacrificios a mi mamá Gabriela y mi papá Dagoberto, son los mejores pardes que alguien pudiera tener. Gracias a mis hermanos Pabel e Ivonne por su apoyo, sus risas y consejos.

Agradezco a todos mis amigos por hacer de esta una aventura, brindarme su sinceridad y ayuda incondicional.

Agradezco a mi asesor Juan Manuel por haberme brindado la oportunidad de crecer profesionalmente, explorar otras culturas y poder aprender de ellas. Por brindarme todas las facilidades que necesité para aprovechar mi estancia en la mestría y alentarme para seguir por el camino del conocimiento.

Agradezco al CONACyT por otorgarme todos los apoyos económicos necesarios para concluír mi tésis y realizar mis estancias.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

ACRÓNIMOS

EKF Extended Kalman Filter

IMU Inertial Measurement Unit

VA Variable Aleatoria

CINVESTAV Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional

SLAM Simultaneous Localisation and Mapping

FIRA Federation of International Robot-soccer Association

DRC DARPA Robotic Challenge

KAIST Korea Advanced Institute of Science and Technology

IDE Integrated Development Environment

DARPA Defense Advanced Research Projects Agency

FMP Función de Masa de Probabilidad

FDP Función de Distribución de Probabilidad

VSLAM Visual Simultaneous Localisation and Mapping

DBN Dynamic Bayesian Network

JC Joint Compatibility

PTAM Parallel Tracking and Mapping

SIFT Scale-Invariant Feature Transform

SURF Speeded-Up Robust Features

BRIEF Binary Robust Independent Elementary Features

FAST Features from Accelerated Segment Test

ORB Oriented FAST and Rotated BRIEF

PTAM Parallel Tracking and Mapping

1 INTRODUCCIÓN

1.1 GENERALIDADES

A través del tiempo, el ser humano a trabajado para obtener dispositivos autónomos que le faciliten y mejoren el trabajo físico y los servicios que necesita para entretenerse y subsistir. En la actualidad, la robótica ha cobrado una importancia relevante, debido a que las tareas en las que se desenvuelven los robots móviles son vastas, desde la mensajería y la cinematografía hasta la industria militar. Sin embargo, la autonomía de dichos robots se encuentra limitada, su incapacidad de ubicarse en el espacio con precisión, les impide interactuar con el entorno o con el ser humano.

La intensión de esta tesis es analizar los métodos de autolocalización disponibles y proponer un algoritmo que contribuya con el estado del arte actual. Además se busca implementar un algoritmo de autolocalización en un robot humanoide que pueda utilizarce en competencias de fútbol. 1.2

1.2 PLANTEAMIENTO DEL PROBLEMA.

Robots de servicio

Para poder desarrollar algoritmos de control en los que el robot se desenvuelva de manera precisa es necesario que cuente con información de su localización y del entorno cambiante que lo rodea. Dicha información permitiría a los robots interactuar con objetos que forman parte de su ambiente, un ejemplo claro de ello son los robots de servicio , los cuales se elaboran para desarrollar tareas de limpieza, camarería, entre otras. Otro ejemplo es el auge de los automóviles autónomos, para los cuales es necesario contemplar la variabilidad de los mapas en el tiempo por efecto de un edificio en construcción, una reparación o modificación en una calle o carretera, negocios nuevos, y muchos otros factores que exigen que los algoritmos sean adaptables, dicho problema es tema abierto de investigación actualmente.

Aprendizaje colaborativo

Para poder lograr que exista aprendizaje colaborativo, un robot, debe realizar una tarea en un entorno en donde necesite colaborar con otros individuos (sean seres humanos u otros robots) implica que éste debe tener la capacidad de tomar decisiones autónomas a partir de los eventos programados o inesperados, sin embargo, al carecer de un sentido espacial de su entorno no podría lograrlo, p. ej. tomar un extremo de una viga para levantarla junto con otro individuo, la tarea tan simple de servir un vaso de agua tendría que contemplar muchas variables, ubicación del vaso, posible movimiento del vaso, posible

movimiento del vaso en caso de que este sostenido por una persona o por vaciar el liquido en sí, posición del dispensador, trayectoria del dispensador al vaso, velocidad del vaciado, entre otros problemas.

Asimismo, el cálculo de trayectorias para un robot móvil en un espacio desconocido es una de las tareas con mayor nivel de dificultad en las que se han visto envueltos los investigadores e ingenieros. Sin embargo, este problema puede resolverse utilizando esquemas de Localización y Mapeo Simultáneos, SLAM (Simultaneous Localization and Mapping), que promete ser la pieza angular que permita a un robot desplazarse dentro de un entorno desconocido de manera segura.

Realidad aumentada

Otra área de enorme interés en donde resultan de mucha utilidad las técnicas de SLAM es el área de servicios en donde los algoritmos de Realidad Aumentada están buscando expandir su capacidad para poder interactuar con el usuario de manera más eficiente. Actualmente, la realidad aumentada consiste en producir la visión de un entorno físico del mundo real a través de un dispositivo tecnológico, cuyos elementos se combinan con elementos virtuales creando una realidad mixta en tiempo real.

1.2.1 *Problemática en el equipo dotMEX*

Desde el 2008, en el Departamento de Control Automático del CINVESTAV, se vienen realizando proyectos de investigación en el área de la Robótica de Humanoides, teniendo al fútbol como una de sus principales aplicaciones. El equipo dotMEX de CINVESTAV obtuvo el primer lugar en el campeonato mundial organizado por la FIRA en Bristol, Reino Unido, en agosto del 2012. En este contexto, los robots humanoides deben ser capaces de ubicarse y orientarse correctamente dentro del terreno de juego, tarea de gran dificultad debido a que este tipo de robots carece de un sistema preciso de odometría. En efecto tareas como la de ingresar al terreno de juego y ubicarse correctamente en su posición de juego, relocalizarse después de una de las frecuentes caídas que sufre el robot, decidir su jugada en función de su ubicación y la de la pelota o de los demás jugadores, tienen una cierta complejidad a pesar de desarrollarse dentro de un entorno conocido (terreno de juego conocido). La eficiente ejecución de estas tareas requiere de un buen sistema de localización eficiente.

Competencias

Localización del entorno

El cálculo de la posición de los objetos en el área respecto al robot es otra de las tareas necesarias para poder generar estrategias de juego, el simple hecho de calcular la trayectoria de caminado del robot a la pelota y la trayectoria que debe seguir la pelota para meterse a gol y así poder elegir el tipo de patada es una tarea imposible de lograr sin contar con información espacial del entorno.

Interacción entre agentes

La interacción entre los diversos integrantes del equipo para lograr un pase o establecer una estrategia de juego necesita del perfecto co-

nocimiento de la ubicación de cada uno de ellos dentro de la cancha, dicha información puede obtenerse de manera fácil si los integrantes no se mueven, sin embargo, en un partido de fútbol todos los agentes involucrados inclusive la pelota están en constante movimiento, además el entrono exterior a la cancha no se encuentra controlado, la tribuna y aficionados forman parte de un ambiente extremadamente dinámico. El robot debe ser capaz de distinguir entre el ambiente dinámico y los objetivos de juego como la pelota, la portería y los jugadores.

Autolocalización propuesta

En este marco de referencia, se ubica el trabajo de investigación reportado en esta tesis. En efecto, aquí se aborda el problema de autolocalización en un entorno conocido (terreno de fútbol). Este problema consiste en la aplicación de técnicas de Visual Simultaneous Localisation and Mapping (VSLAM) para desarrollar un algoritmo que sea capaz de localizar a un jugador del equipo con respecto a un marco de referencia fijo. Para lograr dicha tarea se analiza el uso de métodos de relocalización por medio de regresión general utilizando imágenes, mediante un proceso que tendrá como ventaja tener un costo computacional constante.

Optimización de mapas

Asimismo se explora el problema de SLAM por medio de análisis de grafos con el afán de optimizar los mapas creados por medio de algoritmos de front-end SLAM, la optimización del problema de SLAM mediante grafos permite de manera robusta eliminar las perturbaciones que impactan en una estimación pobre del mapa, obteniendo así, un sistema de SLAM que describe de la manera más acertada la trayectoria del robot en su entorno.

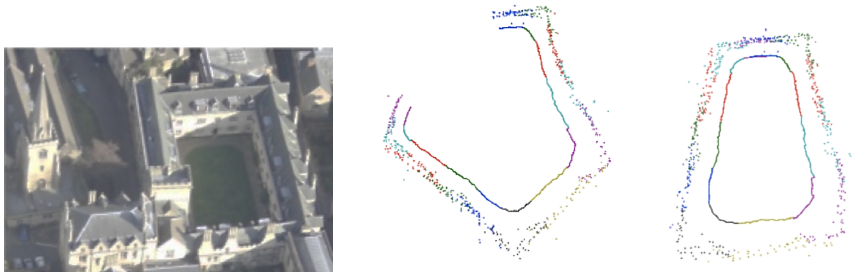
1.3 JUSTIFICACIÓN

La investigación de casi dos décadas en VSLAM comenzada en 1986 han permitido contar con algoritmos que pueden ser ejecutados en *tiempo real* (a una cadencia de al menos 30Hz) y son capaces de dotar a los robots con información acerca de su ubicación y el lugar que los rodea de manera aproximada. El uso de estas técnicas en lugares con ambientes controlados ha mostrado buenos resultados, incluso el problema se ha considerado resuelto, sin embargo, la naturaleza probabilística de estas metodologías han impedido acotar el error de la ubicación, lo que genera la necesidad del empleo de algoritmos para estimar el cerrado de bucles (en circuitos como: una pista de atletismo, un museo, entre otros) para reducir los errores, como se puede observar en la Figura 1.1 donde se muestra el mapa obtenido mediante Hierarchical SLAM (HSLAM) en la explanada de la Universidad de Oxford .

Tiempo real

Cerrado de bucles

Los algoritmos de cerrado de bucle tomaron importancia crítica dentro de los algoritmos de VSLAM, sin embargo, puede ser difícil obtener un cerrado de bucle si se piensa que el robot nunca llegará



(a) Explanada (b) Error de posición (c) Mapa corregido

Figura 1.1: Reducción del error con algoritmo de cerrado de bucle para un mapa obtenido de la Universidad de Oxford [32].

al lugar donde inició el recorrido, un robot cartero podría describir fácilmente esta dificultad puesto que la trayectoria de un punto A a otro punto B no necesariamente comenzará y terminará en la misma localización, el propósito de esta tesis no es analizar las técnicas de cerrado de bucles, pero puede encontrarse información al respecto en [8, 15].

Attractive Ellipoid Method (AEM)

En el Departamento de Control Automático del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV) se ha estudiado el problema de VSLAM Monocular por medio del método de la Elipsoide Atractiva [1, 3], el cuál es un enfoque determinístico que permite acotar el error con certeza, por lo que puede ser una alternativa a la solución de este problema que aporte precisión a la estimación.

Futbol de humanoides

Por otro lado el uso de VSLAM en torneos de fútbol de robots humanoides, ha cobrado fama debido a que es posible desarrollar tácticas de juego en equipo y algoritmos que indiquen correctamente la dirección en la que un jugador debe atacar o defender. Las reglas de torneos internacionales como RoboCup cambian con los años con la intención de que los algoritmos de control y diseños físicos de los humanoides les permitan en un futuro confrontar un equipo de fútbol de seres humanos. Es por eso que, una de las reglas del torneo con las cuales se pretende orillar a los investigadores a desarrollar mejores sistemas de control, es eliminar la orientación espacial por medio del color de las porterías, por lo que a partir del 2013 el color de estas, es el mismo, volviendo imprescindible el uso de algoritmos de SLAM.

1.4 ESTADO DEL ARTE

Origen

El problema de localización y mapeo simultaneo fue propuesto en el año de 1986 en la conferencia de Robótica y Automatización del Institute of Electrical and Electronics Engineers (IEEE) en San Fran-

cisco,los esfuerzos por estimar la localización o mapeo de manera separada no dieron resultados.

Al desarrollo de la solución han aportado sus esfuerzos diversos investigadores como Peter Cheeseman, Jim Crowley, Durrant Whyte, Raja Chatila, Oliver Faugeras, Randal Smith, Sebastian Thrun, José Neira, Andrew Davison, Walterio Mayol entre muchos otros. SLAM.

Los primeros resultados de SLAM fueron obtenidos por Randall C. Smith and Peter Cheeseman en 1987, quienes propusieron el primer algoritmo donde se empleaba el filtro extendido de Kalman (EKF, por sus siglas en inglés) como estimador principal [34]. El EKF ha sido utilizado extensamente hasta la actualidad para resolver el problema de VSLAM, sin embargo, se ha trabajado de manera extenuante en reducir los costos computacionales. SCALABLE SLAM.

Un primer acercamiento para disminuir los costos computacionales fue el uso de "Scalable SLAM" por Eade et. al. [10], en donde era necesario almacenar el video obtenido por la cámara para posteriormente analizarlo con el algoritmo de VSLAM, sin embargo, en Junio del 2007 Andrew J. Davison presentó el primer algoritmo que realizára VSLAM en tiempo real, es decir a la cadencia imagen de 30 Hz, por medio de una cámara monocular. El algoritmo de Davison trabaja MONOCULAR bien para una cantidad de datos acotada por lo cual, desafortunada- SLAM mente, no permite obtener mapas de gran escala [29]. DIVIDE AND

Un esfuerzo por obtener de manera más sencilla un algoritmo que CONQUER SLAM pueda realizar mapas de áreas grandes se realiza en el trabajo de Tardós y José Neira [35], en donde se utiliza el algoritmo "Divide and Conquer SLAM", el cual, para obtener un mapa completo, el programa debía de calcular mapas locales independientes de manera que la computadora realizara operaciones con la menor cantidad de datos posible, finalmente el mapa se conformaba de la unión de los mapas locales independientes. CF SLAM

En el año 2009 en el trabajo de Cadena, José Neria et. al.[3], se propuso un filtro combinado (Combined Filter SLAM (CF SLAM)) entre EKF y el filtro extendido de información (Extended Information Filter (EIF)) obteniendo un algoritmo que ha mostrado tener los menores costos computacionales hasta la fecha como puede apreciarse en la tabla 1.1.

En la actualidad encontrar un algoritmo que asegure un costo constante en cada ciclo de ejecución es tema de investigación abierto y se considera una de las metas que definirá la solución definitiva del problema de VSLAM.

Investigación vigente

Otros temas abiertos de investigación dentro del problema de VSLAM se encuentran:

- Resolver el problema de VSLAM para ambientes dinámicos, es decir, en donde el entorno contiene objetos móviles.

Método	Costo por paso
EKF-SLAM	$\mathcal{O}\left(n^2\right)$
D&C SLAM	$\mathcal{O}\left(n\right)$
EIF SLAM	$\mathcal{O}\left(n\right)$
CF SLAM	$\mathcal{O}\left(log(n)\right)$

Cuadro 1.1: Costo computacional en VSLAM

- VSLAM de larga vida. Los mapas deben ser flexibles a los cambios del ambiente en donde existen objetos temporales.
- VSLAM semántico. Reconocimiento de objetos.
- Asociación de datos.

Dentro de estos problemas la asociación de datos es crucial para lograr que cualquier filtro utilizado en el algoritmo mantenga la estabilidad, por lo que se ha catalogado como el principal problema a resolver, aunque los algoritmos de relocalización han ayudado a que los prototipos puedan seguir trabajando después de que el filtro diverja.

1.5 OBJETIVO DE LA TESIS

A fin de que los robots que forman parte del equipo dotMEX puedan ser competitivos en competencias de RoboCUP y torneos de fútbol en general, los problemas mencionados en la sección 1.2.1 deben ser superados. La presente tesis se concentrará en resolver estos retos otorgando al robot información acerca de su ubicación espacial respecto un marco de referencia fijo.

El progreso en la solución del problema de SLAM ha llevado a pensar que es posible utilizar una cámara de video como único sensor, lo que representa una ventaja muy grande en el abaratamiento de costos y en la simplicidad de la implementación física en un robot.

Este trabajo pretende contribuir al desarrollo de un algoritmo que sea robusto en ambientes no controlados así como desarrollar un algoritmo para robots humanoides que compitan en el torneo de fútbol en la categoría "kid size".

Se implementará un algoritmo de VSLAM que derivado de un análisis pueda resolver el problema de autolocalización para un robot humanoide en un juego de fútbol.

1.6 APORTACIÓN DEL ESTUDIO

En la presente tesis se abordará el problema de autolocalización desde dos puntos de vista principales: Optimización de grafos orientado

a la generación de mapas coherentes y relocalización. En el primer eje de esta tesis (ver capítulo 4), se propone el uso de un software de optimización como plataforma para mejorar la hipotesis de asociación de datos a partir del planteamiento del problema de autolocalización como un grafo con una configuración propuesta. Se implementa un algoritmo para la generación de un mapa de grafos en un software de autolocalización de libre distribución, dicho grafo se utiliza para mejorar el mapa y la trayectoria del programa de autolocalización a través de un algoritmo de optimización. En el segundo eje (ver capítulo 6), se implementa en un robot humanoide un algoritmo de autolocalización basado en algoritmos de relocalización e información de odometría, se ejecutaron pruebas y se discuten los resultados del método elegido.

1.7 ESTRUCTURA DE LA TESIS

En el capítulo 2, se habla del contexto para el desarrollo del modelo. Se introduce la competencia de RoboCup y el equipo dotMEX. Finalmente se explica como fueron agregados los módulos de programación para incluir los algoritmos de autolocalización en el robot Darwin y las herramientas utilizadas para medir su efectividad.

El capítulo 3 cubre algunos conceptos básicos de robótica probabilística y matemáticas que son utilizados para entender los algoritmos utilizados en esta tesis.

En el capítulo 4 se aborda el tema de autolocalización como un problema de grafos, se incluyen los conceptos básicos para el entendimiento de la optimización de grafos. El capítulo se divide en el análisis de sistemas para front.end SLAM y back-end SLAM, en cada una de estas secciones se expone el trabajo realizado en dichas áreas para la implementación de un algoritmo que permita trabajar la optimización de grafos.

El capítulo 5 otorga los conocimientos básicos acerca del problema de relocalización y el algoritmo base utilizado en la implementación del algoritmo de autolocalización.

El capitulo 6 explica como fue implementado en el robot Darwin un algoritmo de autolocalización basado en relocalización y odometría.

Finalmente en el capítulo 7 se discuten los resultados obtenidos en el capítulo 4 y el capítulo 6 y se propone como continuar el trabajo hecho en esta tesis.

2

TORNEOS DE FÚTBOL Y EQUIPO DOTMEX

2.1 LIGA "KID SIZE" DE HUMANOIDES EN ROBOCUP

Una de las principales actividades que han impulsado el desarrollo de los robots humanoides son las competiciones organizadas por federaciones como Federation of International Robot-soccer Association (FIRA) y RoboCup1, así como por entidades gubernamentales de países desarrollados como DARPA (Defense Advanced Research Projects Agency (DARPA))en los Estados Unidos de América. Las dos primeras promueven el desarrollo de los robots humanoides en el ámbito académico internacional a través de tornos de fútbol en diversas categorías y otras competiciones deportivas2, mientras que DARPA lo hace a través de competiciones de rescate que tiene como objetivo lograr que un humanoide realice todas las funciones de un bombero3, en dicha competición, participan grandes empresas transnacionales y muchas de las Universidades más importantes del mundo. El programa DARPA DARPA Robotic Challenge (DRC) tuvo una duración de tres años finalizando el verano del 2015 y teniendo como ganador del primer premio de dos millones de dólares al humanoide DRC-HUBO del Korea Advanced Institute of Science and Technology (KAIST); En dicha competición participaron 23 humanoides, doce de ellos de los EUA y once más de Corea del Sur, Japón, Alemania, Italia y Hong Kong.

2.1.1 *Competiciones de fútbol en RoboCup*

RoboCup

RoboCup es una competencia de talla internacional donde se con-
RoboCup gregan institutos de investigación y universidades para presentar y compartir los resultados de sus investigaciones en las áreas de inteligencia artificial, visión artificial y robótica. En este evento equipos de robots compiten en partidos de fútbol. La primera de estas competiciones se llevó a cabo en 1997, desde entonces los retos se han modificado constantemente, el objetivo principal de la RoboCup es ahora, ya bien conocido:

> *"Para la mitad del siglo 21, un equipo de robots humanoides completamente autónomos deben ganar un partido de fútbol contra el equipo FIFA que sea campeón mundial de la copa más reciente"*[1]

Robotics Challenge

1 *http://www.RoboCup.org/about-RoboCup/objective/*

La dificultad del concurso demanda competencias importantes de inteligencia artificial y robótica, cualidades que han cobrado mucha importancia en los robots humanoides desde el inicio del milenio.

Entorno de RoboCup

El ambiente donde se realiza la competición de RoboCup es dinámico; el movimiento del público,de los robots contrarios, la pelota y los cambios de luz son algunos de los factores que convierten al problema en un reto, la información que obtiene el robot por medio de los sensores exteroceptivos debe ser utilizada para crear un modelo del mundo que sea útil para el robot y que este, pueda realizar tareas. Además, los robots tienen que cooperar y jugar como un equipo, tomar decisiones colectivas e intercambiar información con el objetivo de anotar un gol y prevenir que les sea anotado uno. Todos los aspectos antes descritos, involucran incertidumbre, se carece de información exacta y certera como lo es, por ejemplo, para tareas de robots industriales.

Ligas en RoboCup

Desde 1997, el número de ligas de soccer en las cuales se puede competir, a incrementado. Algunas de estas ligas son:

- Small Sized League. La primer liga de RoboCup. Un equipo de robots pequeños en ruedas controlados por una computadora central y una cámara de visión ubicada en la parte superior de la cancha.
- Mid Sized League. La diferencia con la liga además del tamaño es que cada robot cuenta con su propio sistema de visión y computadora.
- Standard Platafor League. Un equipo de cinco robots humanoides (De forma estandarizada povista por Aldebaran Robotics).
- Humanoid League. La mayoría de los robots humanoides que participan en esta liga son hechos por los mismos participantes, equipados con su propio procesamiento y cámara. Las sub-ligas son: Kid Size, Teen Size, y Adult Size.

Otras competencias

Algunas otras ligas no referentes al soccer pero también incluidas en RoboCup son RoboCup@Home y RoboCup Rescue. En @Home, los robots de servicio se encargan de asistir a las personas en las tareas comunes y rutinarias como abrir la puerta de la casa, llevar una bebida, planchar la ropa, regar las plantas, entre otras. RoboCup Rescue involucra búsqueda y rescate en situaciones de desastres.

La liga de humanoides ofrece los mejores retos dentro de todas las ligas de soccer en la RoboCup. Los robots deben desarrollar diversas tareas dinámicas básicas para su movimiento, como lo son: caminado estable, caminado robusto (equilibrio ante empujones o desniveles del suelo), levantarse de caídas, patear el balón, caminar en trayectorias no rectas, entre otras. Otro de los retos al desarrollar un robot humanoide es el espacio útil disponible para dotarlo de sensores, computadora, actuadores y baterías, pues es muy escaso. Además la

Sensores

forzosa apariencia humanoide del robot obliga a los investigadores a utilizar solo sensores que puedan convivir con ella, por ejemplo, en la competencia esta prohibido utilizar algún sensor láser de rango, y es aquí donde la cámara se convierte en un sensor muy importante para la adquisición de datos, así mismo pueden utilizarce algunas centrales inerciales (Inertial Measurement Unit (IMU)) y sensores de presión usualmente en las manos o pies.

2.1.2 *Retos de la liga Kid-Size de RoboCup*

El ambiente en el cual se llevan a cabo las competiciones de RoboCup es altamente dinámico; el movimiento del público alrededor del terreno de juego, el movimiento de los robots del equipo contrario y del mismo equipo así como el de la pelota y los cambios de luz, son algunos de los factores que convierten al problema en un gran reto. La información que obtiene el robot de su entorno y del estado del partido mediante sus sensores exteroceptivos debe utilizarse para crear un modelo del mundo que sea útil para el robot, así como para planear sus tareas y monitorear la correcta ejecución de éstas. Además, como los robots tienen que cooperar y jugar como un verdadero equipo, deben tomar decisiones colectivas e intercambiar información entre ellos con el objetivo de ganar el partido anotando goles y previniendo que les sean anotados. Así, en la planeación y ejecución de todas las tareas de un humanoide futbolista se involucran incertidumbres y se carece de información exacta y certera.

Los partidos de fútbol se llevan a cabo en canchas de 10 x 5 m, las lineas de campo son similares a las utilizadas en una de fútbol regular; existe una linea media, un circulo en el centro, un área chica, puntos de penalti, y delimitaciones en los extremos de la cancha en color blanco. En los inicios de la competencia en 1997, la cancha fue planteada para que el robot pudiera utilizar los colores de la misma con facilidad, la pelota, las lineas y la cancha contaban con colores diferentes (naranja, blanco y verde respectivamente) así como entre las porterías (azul y amarillo) que ayudaban enormemente a la orientación de los robots hasta el año 2012, sin embargo, a partir del año 2013 la portería del contrario y la propia contaban con el mismo color (amarillo), esto se debió a que se pretende impulsar los algoritmos de autolocalización para la ubicación y orientación del robot, ya que es imprescindible para lograr resolver problemas como planeación de trayectorias y el juego cooperativo. La pelota inicialmente era de color naranja, pero ahora, se utiliza un balón FIFA No. 1 estándar que generalmente tiene varios colores sobre una superficie blanca.

Semantic SLAM

En el año 2015 se planea que las porterías tengan color blanco aumentando la complejidad de los algoritmos para identificar objetos o de los algoritmos de autolocalización basados en reconocimiento objetos (Semantic SLAM). Los jugadores de cada equipo también de-

ben vestir un identificador de color que los distinga como jugadores de cada equipo (equipo magenta y azul) y así, sea fácil para el robot localizar a sus compañeros.

Se comenzó jugando con un robot por equipo haciendo tandas de penales, con el tiempo, se realizaron partidos con tres robots por equipo, primero en cancha de 4m y luego en una de 6m; ahora se juega en cancha de 10m con 5 jugadores en cada equipo. Existen dos tiempos, cada uno de diez minutos y un cambio de lado en la cancha al medio tiempo. Una parte muy importante es que los robots reciben los hitos importantes del partido por medio de una computadora nombrada "Game Controller" que es a su vez, controlada por un referí humano o su asistente. Dichos hitos incluyen: inicio del juego, gol del equipo magenta o azul, medio tiempo, robot penalizado y otros.

Game controller

2.2 EQUIPO DOTMEX

En el Departamento de Control Automático del CINVESTAV Unidad Zacatenco se desarrollan actividades académicas, científicas y tecnológicas por parte de un grupo de profesores y estudiantes interesados en la Robótica del Siglo XXI, o sea la Robótica de Servicios, en particular en los robots humanoides con alto grado de autonomía tanto de movilidad como en la toma de decisiones, basados en sistemas computacionales complejos, alto nivel de percepción visual y control eficiente de su comportamiento.

Actividades

Desde sus inicios, en el 2008, este grupo adoptó al fútbol como una de las principales aplicaciones de los robots humanoides, naciendo así el equipo dotMEX. El mayor interés se centra en los Robots Humanoides y en sus aplicaciones ligadas al fútbol en el contexto de competiciones internacionales (RoboCup, FIRA, TMR). También se elaboran aplicaciones médicas en el contexto del proyecto OpenSurg[2],[3].

Logros

El equipo dotMEX ganó el torneo FIRA realizado en Bristol, RU en el 2012. Además, se han desarrollado prototipos propios como el AH1N1, el AH1N2 de 60 cm de altura y Johnny de 90cm de altura. Estos prototipos propios cuentan con visión estéreo y son capaces de caminar con estabilidad y equilibrio y de guiarse mediante visión artificial.

El equipo funciona como base de desarrollo de algoritmos para diversas ramas de investigación como lo son: VSLAM, caminado dinámico, IA, entre otras. Las plataformas en las que trabaja el equipo son: Nao de Aldebaran Robotics, DarwinOPOP y BIOLOID de la compañía ROBOTIS.

Esta tesis esta enfocada en dotar a los humanoides del equipo dotMEX la capacidad de autolocalización, motivo por el cual se abor-

2 http://ctrl.cinvestav.mx/~coordinación/jmibarraz/index.htm
3 https://sites.google.com/a/goumh.umh.es/opensurg/

darán a continuación las características del funcionamiento del robot Nao y DarwinOP, siendo este último en donde se desarrolla el algoritmo de relocalización finalmente.

2.2.1 *Robot humanoide Nao*

NAO es un robot humanoide avanzado de precio razonable con características atractivas que la convierten en una plataforma ideal para la investigación. Esta diseñado con base en una filosofía de arquitectura abierta y puede ser programado por medio de diferentes plataformas de programación. En esta sección, el propósito es presentar una visión general de las partes por las que esta compuesta el robot NAO, incluyendo sus sensores y arquitectura.

2.2.1.1 *Hadware del robot humanoide Nao*

NAO es un robot de talla media completamente programable para poder realizar tareas de manera autónoma. El robot NAO utilizado en el equipo tiene versión de hardware v.3.3, su altura es 573mm y su peso es de 5.02 Kg. Esta equipado con un CPU x86 AMD Geode 500MHz y una memoria flash de 2GB, la batería es de ion-litio. Nao tiene 25 grados de libertad distribuidos como se muestra en la Figura 2.1.

2.2.1.2 *Sensores exteroceptivos*

El robot Nao cuenta con los siguientes sensores exteroceptivos:

- Un sensor capacitivo en la cabeza: Se utiliza generalmente para indicar de manera física al NAO que inicie alguna tarea o la detenga, sin embargo, el modulo de control puede utilizarse como al programador le convenga.
- Sensores de tacto: Cuenta con dos botones enfrente de los pies que se utilizan para detectar colisiones del robot con elementos de su entorno, de la misma manera pueden programarse para realizar una actividad a convenir.
- Sensores ultrasónicos: Cuenta con dos emisores y dos receptores, la frecuencia de la señal emitida es de 40Khz, tienen un rango de detección de 0.25-2.55m en un cono efectivo de 60°.
- Cuenta con dos cámaras HD idénticas con resolución de 1280x960, las características que describen la cámara se encuentran el la tabla 2.1. La posición de las cámaras se puede observar en la Figura 2.2, las cámaras no permiten utilizar algoritmos de visión estereoscópica puesto que campos de visión son disjuntos, inicialmente en las versiones de NAO solo se incluía la cámara

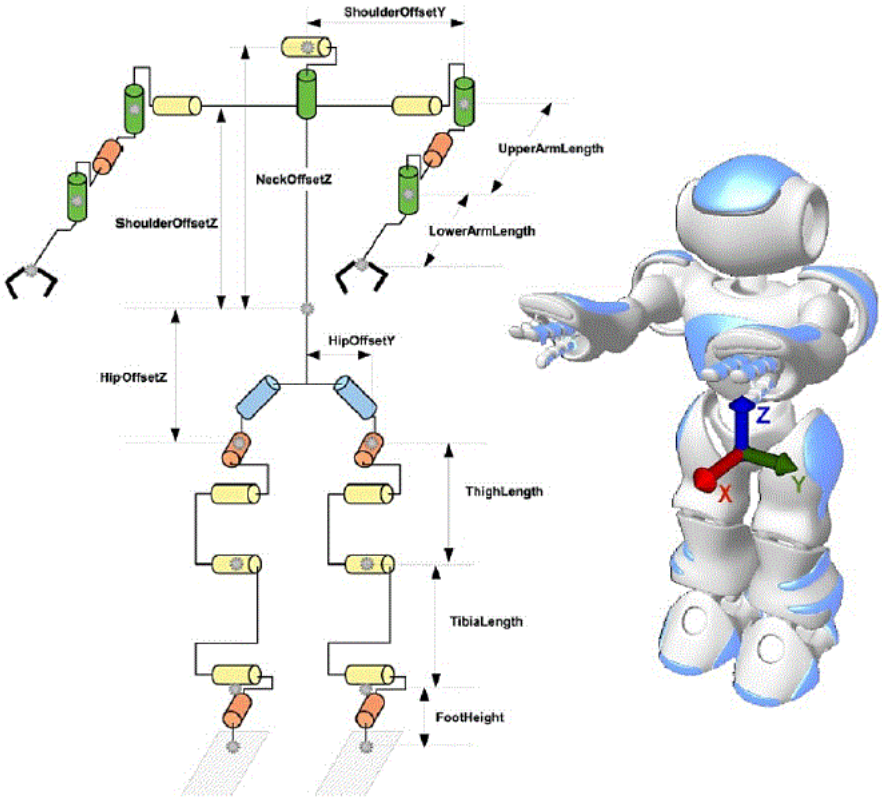


Figura 2.1: Cinemática del robot humanoide NAO.

superior, pero puesto que la pelota en un juego de fútbol se encontraba fuera del alcance de visión cuando esta se encontraba a los pies del robot, se incluyo la cámara inferior.

2.2.1.3 *Sensores propioceptivos*

Dentro de los sensores propioceptivos se puede encontrar:

Característica	Despcripción
Modelo del sensor	Despcripción Modelo del sensor OV7670
Salida de la cámara	VGA@30fps (Espacio de color YUV422)
Campo de visión	58°DFOV (47.8°HFOV, 36.8°VFOV)
Rango de enfoque	30cm ~ infinito
Tipo de focus	Foco fijo

Cuadro 2.1: Características de las cámaras en el robot NAO.

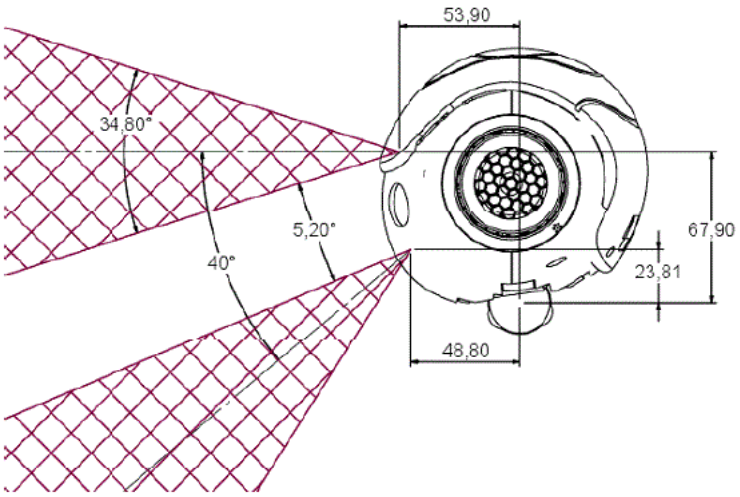


Figura 2.2: Disposición geométrica de las cámaras en el robot NAO.

- Unidad inercial: Girómetro de 2 ejes (precisión del 5 % con una velocidad angular de ~500°/s), acelerómetro de 3 ejes (precisión de 1 % con una aceleración de ~2G)
- Encoders magnéticos: El robot cuenta con 32 sensores de efecto Hall, tienen una precisión de 12 bits, es decir, 4096 valores por vuelta con una precisión de 0.1°.

2.2.1.4 *Framework del robot humanoide Nao*

El sistema operativo nativo en el NAO es Linux y es ampliamente recomendable utilizarlo como plataforma de trabajo en la PC puesto que permite flexibilidad de programación en el NAO altamente necesarias para aplicaciones de alto nivel.

Sin embargo, esta diseñado para que programadores de diferentes habilidades, desde aquellos que no estén capacitados hasta los expertos, puedan programar el robot. Para poder desarrollar funciones y control en el robot de manera adecuada, es necesario entender las partes en las que esta constituida la arquitectura y como están relacionadas entre si.

El robot Nao esta formado por tres partes principales: Software interno , software externo y herramientas de programación. El software interno juega un papel importante en la tarjeta madre del robot y organiza las tareas en módulos, el software que se encarga de ejecutar los módulos, así como establecer comunicación con el usuario es NAOqi, el cual trabaja dentro del sistema operativo del robot OpenNao. Por otro lado, se encuentra el software externo “Choregraphe” y “Monitor” , distribuido por Aldebaran, o Webots para Nao, en estos programas es posible elaborar tareas que se almacenaran en la PC

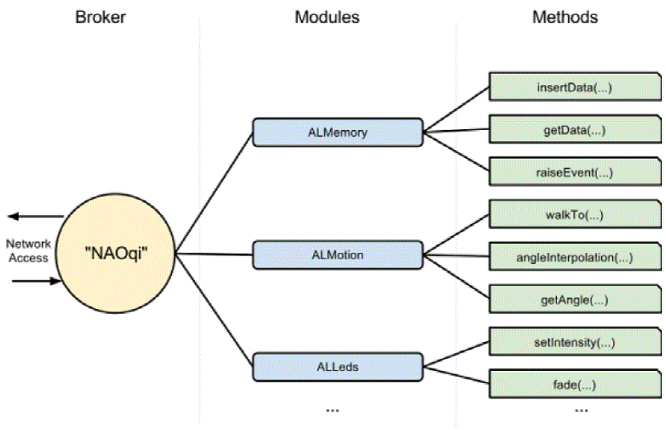


Figura 2.3: Naoqui Framework.

pero que pueden ejecutarse de forma temporal en el Nao (módulos externos) o simplemente leer datos de los sensores del NAO.

Finalmente las herramientas de programación otorgan al usuario la flexibilidad de crear su propio código y por ende módulos que pueden guardarse en el sistema operativo del NAO para poder desarrollar diversas tareas.

En la Figura 2.3 se muestra un diagrama de la arquitectura de software en el NAO.

El "Broker" es un objeto que se encarga de comunicar los módulos internos del NAO con los módulos externos creados mediante algún Integrated Development Environment (IDE) o plataforma como Choregraphe.

Se recomienda utilizar un "system development kit" (Naoqi SDK) con Qt Creator o algún otro IDE para desarrollar los módulos externos e internos en lenguaje C++ pues este conjunto de opciones permitirá explotar por completo la plataforma abierta que ofrece el robot NAO.

En la Figura 5 se puede observar como trabajar sobre un IDE permite crear módulos evitando transformaciones que retrasen la comunicación con el NAO.

2.2.1.5 *Programación del robot*

Para programar el robot es necesario contemplar, que de manera similar con otros sistemas embebidos, es necesario poder realizar una compilación cruzada para ejecutar programas fuera del sistema operativo del robot, es por esta razón que se utilizará CMake.

CMake es una herramienta multiplataforma de generación o automatización de código. El nombre es una abreviatura para "cross

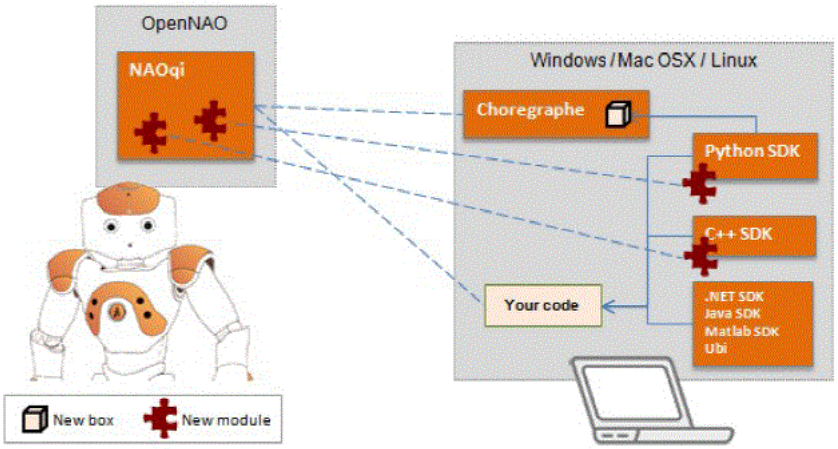


Figura 2.4: Comunicación OpenNAO con PC.

platform make" (make multiplataforma); más allá del uso de "make" en el nombre, CMake es una suite separada y de más alto nivel que el sistema make común de Unix, siendo similar a las autotools.

CMake es una familia de herramientas diseñada para construir, probar y empaquetar software. CMake se utiliza para controlar el proceso de compilación del software usando ficheros de configuración sencillos e independientes de la plataforma. Cmake genera makefiles nativos y espacios de trabajo que pueden usarse en el entorno de desarrollo deseado. Es comparable al GNU build system de Unix en que el proceso es controlado por ficheros de conFiguración, en el caso de CMake llamados CMakeLists.txt. Al contrario que el GNU build system, que está restringido a plataformas Unix, CMake soporta la generación de ficheros para varios sistemas operativos, lo que facilita el mantenimiento y elimina la necesidad de tener varios conjuntos de ficheros para cada plataforma.

CMakelist.txt

El proceso de construcción se controla creando uno o más ficheros CMakeLists.txt en cada directorio (incluyendo subdirectorios). Cada CMakeLists.txt consiste en uno o más comandos. Cada comando tiene la forma COMANDO (argumentos...) donde COMANDO es el nombre del comando, y argumentos es una lista de argumentos separados por espacios. CMake provee comandos predefinidos y definidos por el usuario. Existen generadores makefile para Unix, Borland make, Watcom make, MinGW, MSYS y Microsoft NMake. También es posible generar ficheros de proyecto para Code::Blocks, Eclipse CDT, Microsoft Visual Studio de la 6 a la 10 incluyendo versiones de 64 bits y KDevelop, por supuesto en este trabajo se utilizó junto con Qt Creator como IDE para programar los módulos del NAO.

Además de Cmake es necesario utilizar “qibuild”, es un programa que se ejecuta en la computadora del desarrollador. Es la herramienta

más importante para construir, conFigurar y compilar proyectos en el robot NAO.

Qibuild puede conFigurar proyectos para diversos robots (NAO Atom o Geode) o para ejecutar los programas en una computadora separada para lo que utilizará CMake.

Es necesario que al conFigurar el proyecto en qibuild asignar una "toolchain", es decir, un archivo de datos en donde se encuentra la conFiguración del robot que utilizaremos, esta deja saber a qibuild que datos y herramientas se necesitan para ejecutar el programa.

2.2.2 *Robot humanoide DarwinOP*

El robot Darwin comercializado por Robotis®es una plataforma desarrollada por dicha empresa en cooperación con la Universidad de Pensilvania, quien con su equipo DARwIn fue campeona en la liga Kid Size de RoboCUP en los años 2011, 2012 y 2013. La arquitectura del robot es abierta, a diferencia del robot Nao la accesibilidad del DarwinOP es mas transparente, cuenta con una fit-PC2i cargada con Unbuntu 9.10 en la cuál es posible programar remota o directamente en el lenguaje elegido (C++ o Phyton). A continuación se presenta un resumen de los elementos más importantes que conforman al humanoide.

2.2.2.1 *Hardware del robot humanoide DarwinOP*

El robot DarwinOP utilizado en el equipo es una versión ROBOTIS OP sin sensores de presión en la planta de los pies y sin manos articuladas. Cuenta con 20 grados de libertad, su disposición puede observarse en la Figura 2.5.

2.2.2.2 *Sensores exteroceptivos*

La versión utilizada en el equipo solo cuenta con una cámara entre los ojos LED del robot con las características mostradas en la tabla 2.2.La cámara es utilizada para algoritmos de visión monocular, por lo que representa una desventaja contra los robots que estén equipados con visión estéreo. Al no contar con sensores de fuerza o tacto es más difícil desarrollar algoritmos de estabilidad tales como ZMP para aumentar la robustes en el caminado en superficies rugosas, velocidad y fuerza en el tiro a gol. Sin embargo, para desarrollar algoritmos de VSLAM y relocalización, la plataforma es suficiente.

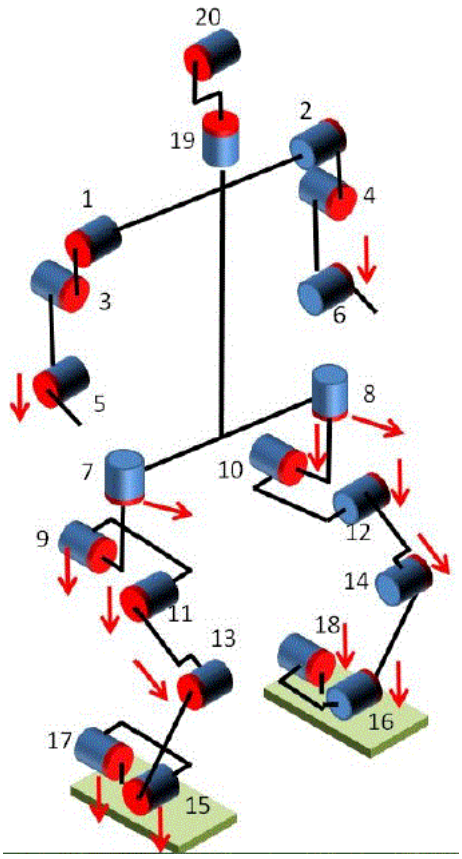


Figura 2.5: Cadena Cinemática del DarwinOP.

Característica	Despcripción
Lente	Carl Zeiss® con autofoco.
Sensor	2 MP HD
Video	HD (arriba de 1600x1200@10fps, 1280x720@30fps)
Foto	Por encima de 8 MP
Microfono	Logitech® tecnología RightSound™
Formato de salida	MJPG, YUYV

Cuadro 2.2: Características de la cámara en DarwinOP OP.

2.2.2.3 *Sensores propioceptivos*

Derivado de las restricciones de RoboCup por emular los sentidos de los seres humanos en los robots humanoides, la flexibilidad para utilizar sensores propioceptivos es mayor, aunque la versión utilizada en del equipo dotMEX cuenta con los siguientes sensores:

- Unidad inercial: Girómetro de 3 ejes (velocidad angular máxima observable de ~1600°/s), acelerómetro de 3 ejes (aceleración máxima observable de ~4G)
- Dos micrófonos (uno por oído).

2.2.2.4 *Software Framework del robot humanoide DarwinOP*

El Framework del robot DarwinOP se encuentra dividido en módulos, se trata de una arquitectura que separa las tareas que realiza el robot y se controlan mediante funciones independientes que permiten a nuevos programadores estudiantes adaptarse fácilmente a la plataforma y trabajar casi de manera inmediata. La idea principal es que los desarrolladores creen sus propios módulos sin necesidad de modificar los ya existentes y utilizando los que crea conveniente.

Módulos

Módulo Motion: este módulo se encarga de calcular el movimiento de los motores que conforman la cinemática del robot cuando ejecuta una tarea preprogramada como caminar, patear, sentarse, pararse, mover la cabeza, entre otras. Genera las trayectorias de movimiento y utiliza un control PID para lograr en medida de lo posible que el robot desarrolle un caminado correcto. Además, este módulo recibe información importante del módulo de control, tal como la posición y la velocidad de los servomotores, útil en algoritmos de odometría por mencionar un ejemplo.

Módulo Visión: la principal tarea de este módulo es obtener la imagen de la cámara y almacenarla en una matriz. Existen algunas herramientas de visión artificial preprogramadas tal como reconocimiento de pelota (BallTracker), seguimiento de pelota (BallFollower) y segmentación por color (ColorFinder), sin embargo, para el desarrollo de aplicaciones que requieran de herramientas especializadas, es recomendable utilizar librerías más completas como OpenCV que se encarguen de reemplazar este módulo.

Módulo Etc: el robot cuenta con un módulo propio de álgebra lineal, el cual se utiliza en el módulo motion para la solución de la cinemática directa del robot así como para realizar las funciones del módulo de visión.

Módulo H/W: su función es la comunicación con el exterior; este módulo de encarga de transferir información de la computadora fitPC2i a la tarjeta CM-730 encargada del control de los servomotores, asimismo se encarga de comunicarse con el usuario a través de Linux en una computadora remota.

Algoritmo 2.1 Nueva_herramienta.h en el módulo PAKAL.

```
#ifndef NUEVA_HERRAMIENTA_H
#define NUEVA_HERRAMIENTA_H
#include <librerías_útiles.h>
class nueva_herramienta
{
    private:
    //Se genera una instancia para acceder a la clase.
    static Skill* m_UniqueInstance;
    Skill();
    //Aqui se definen variables globales que pueda utilizar la herramienta, así como funciones que deseen crearse.
    int variable1;
    double vector[#];
    void nueva_función();
};
#endif
```

Módulo Pakal (Creado para esta tésis) : en este módulo se encuentran las herramientas programadas por el equipo dotMEX, necesarias para la realización de tareas como: Autolocalización y/o relocalización, Odometría, Estrategia para la toma de decisiones durante el partido, Comunicación con el Game Controller, entre otras. La forma de utilizar este módulo en el programa principal es a través de la librería Skill.h. Además de Skill.h, es posible crear otras herramientas dentro de este módulo; para ello, primero es necesario definir la librería en la dirección Darwin/Framework/include tal y como se muestra en el Algoritmo 2.1.

Enseguida debe crearse el archivo .cpp en DarwinOP/Framework/src/ pakal con el algoritmo base mostrado en 2.2, finalmente se genera el objeto (.o) dentro de la misma dirección.

La linea de comunicación de los módulos más importantes (motion y vision) puede observarse en la Figura 2.6, sin embargo, la arquitectura puede ser modificada.

2.2.2.5 *Herramientas adicionales*

El algoritmo desarrollado se enfoca a poder proporcionar a los integrantes del equipo dotMEX habilidades de localización en una cancha de fútbol para la competencia de RoboCup, dicho algoritmo debe ser efectivo aún en un ambiente dinámico, y para poder poder medir dicha efectividad se aprovecha el módulo de visión y la clase

Algoritmo 2.2 Nueva_herramienta.cpp

```
#include <librerías_útiles.h>
nueva_herramienta* nueva_herramienta::m_UniqueInstance = new
nueva_ herramienta();
nueva_herramienta::nueva_herramienta()
{
}
nueva_herramienta~nueva_herramienta()
{
}
void nueva_herramienta::nueva_función()
{
//Operaciones de nueva_función()
}
```

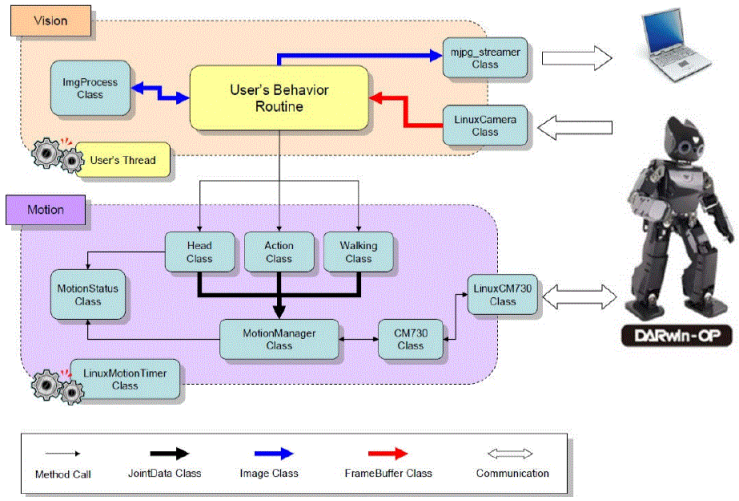


Figura 2.6: Linea de comunicación de los módulos de DarwinOP.

mjpg_streamer para enviar información a una computadora remota. La interfaz muestra datos útiles en el partido, como lo son: valor del giroscopio y el acelerómetro, posición de las articulaciones y ubicación estimada del robot. Asimismo es posible informar al robot acerca de su entorno, corregir las estimaciones de profundidad o de autolocalización, muy útil para el desarrollo de algoritmos de aprendizaje, antes del partido.

Comunicación vía TCP/IP

Además para poder elaborar el mapa del entorno se optó por utilizar un programa que envía la odometría e imagen obtenida por la cámara del robot DarwinOP utilizando socket.h una librería para envíar información via TCP/IP. Con estos datos se eligen puntos es tratégicos en donde se elabore el mapa (ver capítulo 6).

Monitoreo

De las opciones posibles para obtener información certera del área analizada se eligió colocar una cámara externa (webcam logitech) que proporcionara una vista superior del área mapeada. Un algoritmo desarrollado en OpenCV produce información del área de análisis incluyendo la posición y orientación real del robot, para lo cual se pusieron marcas de color en su cabeza. A este sistema de monitoreo visual del robot humanoide DarwinOP se le denominó Quetzal.

Esta aplicación se desarrolló con base en algoritmos para transformar las coordenadas utilizando cuatro marcas que delimitan el área de análisis de las cuales se conoce su ubicación exacta en el terreno de juego, se utilizo además un filtro para corregir la distorsión de barril con base en los parámetros obtenidos mediante calibración y, finalmente, se proyecto la posición real del robot graficándola en la imagen para obtener así la posición real de los objetos en el área. El sistema experimental implementado requiere de varias aproximaciones en cuanto a la proyección, por lo que produce datos de posición con un error, sin embargo, es útil comparar los datos obtenidos por nuestros algoritmos de relocalización con la posición otorgada por el sistema de monitoreo Quetzal.

3

BASE TEÓRICA

3.1 PRELIMINARES MATEMÁTICOS Y PROBABILIDAD

3.1.1 *Probabilidad frecuentista contra probabilidad Bayesiana.*

Eventos

El punto de vista clásico de la estadística habla de *eventos* (E). Los eventos se utilizan para describir si algo es cierto o falso, por ejemplo, un volado fue cara. Aventar una moneda al aire es un ensayo. La probabilidad esta definida como la razón entre los eventos que son verdaderos (la moneda fue cara diez veces) y el número total de ensayos (la moneda se lanzo veinte veces al aire) $P(E) = {}^{10}/_{20} = 0.2$. La aproximación clásica se llama *frecuentista*. El frecuentismo tiene sus limites cuando se requiere predecir eventos que no han sucedido.

Frecuentismo

Bayesianismo

El *Bayesianismo* (*del inglés «Bayesanism»*) define la probabilidad como el grado de creencia en la ocurrencia de un evento. Por lo tanto los eventos como el resultado de una elección o el contagio de una enfermedad pueden ser predichos. Las bases de esta teoría fue introducida por Bayes (1701 - 1761) (ver sección 3.1.3).En las decadas recientes el bayesianismo a sobrevivido en el nicho de la estadística aún cuando se ha utilizado con éxito en economía y encriptación de información. Se ha calificado como subjetiva debido a que necesita un *antecesor*, es decir, un conocimiento subjetivo del evento que puede diferir de persona a persona. Puede consultarse McGrayne [26] para conocer la historia del teorema de Bayes mas a fondo.

3.1.2 *Probabilidad básica y distribuciones de probabilidad.*

Variable Aleatoria

Una Variable Aleatoria (VA) es una variable que puede tomar diferentes valores. Los valores pueden ser discretos o continuos. Por ejemplo, el caso de lanzar una moneda al aire, la variable aleatoria puede tomar los valores cara o cruz. La probabilidad de cada resultado es $P(X = cara) = 0.5$, $P(X = cruz) = 0.5$.

Media

La *media* es el valor esperado de una VA. Por ejemplo, la media de un dado no cargado es 3.5. La suma de todas los resultados multiplicadas por su probabilidad individual da como resultado la media:

$$E[X] = 6\tfrac{1}{6} + 5\tfrac{1}{6} + 4\tfrac{1}{6} + 3\tfrac{1}{6} + 2\tfrac{1}{6} + 1\tfrac{1}{6} = 3.5$$

De forma general la formula de la media es:

$$E[X] = x_1p_1 + x_2p_2 + \ldots + x_np_n = \sum_{i \epsilon N} x_ip_i$$

donde p_i es probabilidad de obtener el resultado x_i. Generalmente la media se representa mediante la letra griega μ.

Varianza

Muy a menudo la media ofrece poca información por si sola. La *varianza* es una medida de cuanto se alejan las muestras de la media. Para ser más exactos es el cuadrado de la desviación de la media:

$$\sigma^2 = Var\,(X) = E\left[\left(X - \mu^2\right)\right] = \frac{1}{n}\sum_i (x_i - \mu)^2$$

Desviación estandar

La *desviación estándar* σ es la raíz cuadrada de la varianza. Las unidades de la desviación estándar serán las mismas que las unidades de las mediciones utilizadas en el cálculo. Esto permite evaluar de manera intuitiva el significado de la desviación estándar, mientras que es más difícil para la varianza.

Covarianza

La *covarianza* es una medida de como dos VAs se correlacionan.

$$cov\,(X, Y) = E\left[(X - E\left[X\right])\,(Y - E\left[Y\right])\right] = E\left[XY\right] - E\left[X\right]E\left[Y\right]$$

$E\left[X\right]$ es el valor esperado de x, que es comúnmente escrito como μ_x.

La matriz de covarianza es la generalización de la covarianza a un vector de VAs. Cada elemento en la matriz de covarianza indica la covarianza de una VA a otra VA en un vector de VAs:

$$\sum_{i,j} = cov\left(X_i, X_j\right) = E\left[\left(X_i - \mu_i\right)\left(X_j - \mu_j\right)\right]$$

Es decir

$$\sum = \begin{bmatrix} cov\,(X_1, X_1) & cov\,(X_1, X_2) & \cdots & cov\,(X_1, X_n) \\ cov\,(X_2, X_1) & cov\,(X_2, X_2) & \cdots & cov\,(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ cov\,(X_n, X_1) & cov\,(X_n, X_2) & \cdots & cov\,(X_n, X_n) \end{bmatrix}$$

Distribucion de probabilidad

Para poder expresar los datos de manera más completa, se pueden utilizar las *distribuciones de probabilidad* en lugar de sumatorias estadísticas tales como la media o la varianza. Las distribuciones asignan a cada valor una probabilidad. Estas existen para los casos continuos y discretos. Dados los resultados para una VA discreta, la Función de Masa de Probabilidad (FMP) asigna a cada resultado de de la variable aleatoria discreta una probabilidad. Por ejemplo, una FMP al arrojar una moneda al aire podría asignar a cada resultado de 1...6 la probabilidad de $^1/_6$.

Función másica de probabilidad

Función de distribución de probabilidad

Una Función de Distribución de Probabilidad (FDP) describe para una variable aleatoria continua lo que una FMP para una discreta. Sin embargo, para obtener la probabilidad de una variable aleatoria

continua, es necesario integrar alrededor de un intervalo. La FDP es una función que asigna a una variable aleatoria una probabilidad de tomar el valor de cierta cantidad. La sumatoria de las FMPs deben ser uno

$$\sum_{x \epsilon X} P(x) = 1$$

mientras que, la integral de las FDPs debe ser uno

$$\int P(x)\, dx = 1$$

Moda

La *moda* es el valor con el mayor número de repeticiones en un grupo de datos. Ejemplo: la moda del conjunto $4, 5, 4, 8 = 4$.

Unimodal

Las FDP y FMP pueden ser *unimodales* o *multimodales*. Existen diferentes definiciones del concepto de unimodal, la más aceptada lo define como una distribución que solo tiene un máximo y por lo tanto solo un movimiento. Por el contrario, las distribuciones con más de un máximo se llaman multimodales.

Multimodal

Distribución gausiana

La *distribución gausiana* es una distribución de probabilidad muy común, su FDP tiene forma de campana y es unimodal. Esta completamente definida por sus primeros dos momentos: la media m y la covarianza σ^2.

$$P(x) = \left(2\pi\sigma^2\right)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right\}$$

Una distribución Gausiana es generalmente llamada Gausiana o Distribución Normal y como tal, debe escribirse como sigue:

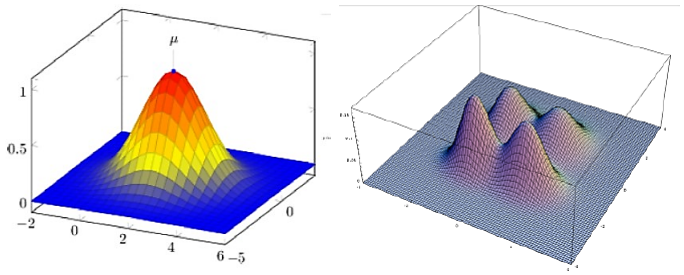
$$N\left(\mu, \sigma^2\right)$$

o al evaluar una FDP en la posición x

$$N\left(x \mid \mu, \sigma^2\right)$$

Las distribuciones Gausianas son utilizadas para aproximar distribuciones desconocidas alrededor de un punto. Algunas propiedades útiles son:

- Es fácil manipular analíticamente una distribución Gausiana.
- Una Gausiana puede ser trasladada y conservarse como Gausiana.
- Una Gausiana puede ser linealizada y conservarse como Gausiana.
- Una Gausiana multiplicada con otra Gausiana resulta en una Gausiana.



(a) Gausiana Unimodal (b) Gausiana multimodal

Figura 3.1: Distribución Gausiana.

Gausiana mutltivariante

Cuando la Gausiana es multivariada el estado x no es un escalar si no un vector, μ es la media del vector y $\sum$ es la matriz de covarianza.

$$N\left(x \mid \mu, \sum\right) = det\left(2\pi \sum\right)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\left(x-\mu\right)^{T} \sum \left(x-\mu\right)\right\}$$

En ocasiones las distribuciones unimodales como la Gausiana no describen suficientemente bien algunos problemas. La suma de Gausiana es una extensión del caso Gausiano multi-modal. Cualquier distribución puede aproximarse mediante una suma de Gausianas.

Una característica adicional de la suma de Gausianas además de la media μ_i y la matriz de covarianza $\sum_i$, es el peso ω_i que especifica la influencia de la Gausiana sobre toda la distribución.

$$\sum_{i \epsilon I} \omega_i N\left(\mu_i \sum_i\right)$$

La suma de todos los pesos de la Gausiana debe ser uno.

$$\sum_{i \epsilon I} \omega_i = 1$$

Consecuentemente la integral sobre la suma de Gausianas es uno. En la figura 3.1 puede observarse una FDP Gausiana unimodal y multimodal.

3.1.3 *Teorema de Bayes.*

Probabilidad conjunta

A la probabilidad de que dos VAs sean verdaderas al mismo tiempo se le llama probabilidad conjunta.

$$P\left(a \wedge b\right) = P\left(a\right) P\left(b \mid a\right) = P\left(b\right) P\left(a \mid b\right) \tag{3.1}$$

Si y solo si, a y b son condicionalmente independientes, la probabilidad conjunta puede ser escrita como:

$$P(a \wedge b) = P(a)\,P(b)$$

Probabilidad condicional

Tener conocimiento acerca de la VA (p. ej. llovió) generalmente otorga información acerca de otra VA (p. ej. el pasto esta mojado). Esto puede expresarse con el concepto de *probabilidad condicional.*

$$P(a \mid b) = \frac{P(a \wedge b)}{P(b)}$$

Teorema de Bayes

De la ecuación 3.1, el *teorema de Bayes* puede inferirse fácilmente resolviendo para uno de las dos probabilidades condicionales, p. ej. $P(b \mid a)$:

$$P(b \mid a) = \frac{P(b)\,P(a \mid b)}{P(a)}$$

Iterpretación diacrónica

Thomas Bayes nombro su método *probabilidad inversa* [26].

Una interpretación diferente del teorema de Bayes se conoce como interpretación diacrónica. Diacrónico proviene del griego Diahronikos que significa: algo que pasa en el tiempo. En esta interpretación, se plantea como un cuerpo de evidencia E afecta la hipotesis H en el tiempo. Nueva evidencia influye en la probabilidad de la hipotesis anterior.

$$P(H \mid E) = P(H)\,\frac{P(E \mid H)}{P(E)}$$

Esto puede ser expresado fácilmente remplazando las probabilidades con términos más apropiados:

$$posterior = hipotesis\,anterior\frac{probabilidad\,de\,la\,evidencia}{normalización}$$

Aunque el teorema de Bayes parece simple e insignificante, forma parte del núcleo de la mayoría de los algoritmos modernos en robótica y "machine learning". Ver sección3.2.

3.1.4 *Suposición de Markov y Cadenas de Markov.*

Propiedad de Markov

La *suposición de Markov* o propiedad de Markov puede resumirse de la siguiente manera: el futuro es independiente del pasado dado el presente. Por lo tanto si el estado actual describe el mundo lo suficientemente bien, para que la predicción del futuro no se vea alterada por la información del pasado, la suposición de Markov se cumple. Matemáticamente, la propiedad de Markov se escribe:

$$P(x_{t+1} \mid x_{i:t})$$

la cuál utiliza todos los estados anteriores para predecir el siguiente estado, puede reducirse a:

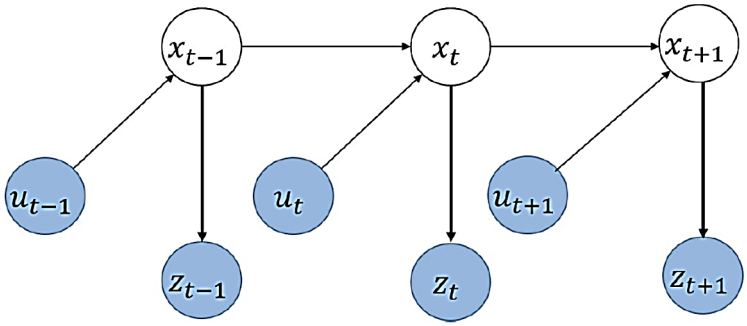


Figura 3.2: Cadena de Markov.

$$P\left(x_{t+1} \mid x_t\right)$$

la cuál solo utiliza el estado actual para predecir el siguiente estado. La propiedad de Markov se utiliza comúnmente para estimación recursiva de estado, comúnmente llamados observadores en control automático.

Estado completo

Un estado que satisface la propiedad de Markov se llama *estado completo*. Para problemas prácticos, el concepto de estado completo es una suposición teórica, por lo que es casi imposible modelar el espacio de estados como un estado completo. Sin embargo, para facilitar los cálculos, el estado se asume completo, aún si se habla de un *estado incompleto*.

Estado incompleto

Cadena de Markov

La Figura 3.2 muestra una *cadena de Markov*. Una cadena de Markov es un proceso temporal que cumple la propiedad de Markov. el próximo estado x_{t+1} depende solamente del estado previo x_t y (si existe) el el control actual u_{t+1}.

3.1.5 *Linealización de funciones.*

Algunos algoritmos se diseñan para manejar funciones lineales. Sin embargo, en muchos casos, la relación entre las VAs es no lineal. En robótica, es necesario encontrar una aproximación lineal de las funciones no lineales. La *linealización* de funciones f en el punto k esta definida como:

Linealización

$$f\left(x\right) \approx f\left(k\right) + \frac{df}{dx}\left(k\right)\left(x - k\right)$$

Jacobiano

La matriz Jacobiana o *Jacobiano* extiende la aproximación de la linealización al caso multi dimensional. Se trata de una matriz consistente en la derivada parcial de primer orden de un vector o una función con respecto otro vector.

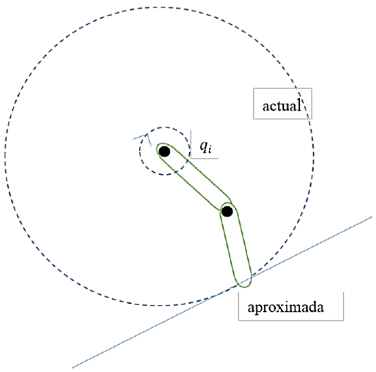


Figura 3.3: Brazo robótico dos grados de libertad.

$$J_{i,j} = \frac{\partial F_i(x)}{\partial x_j}$$

$$J = \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \cdots & \frac{\partial F_m}{\partial x_n} \end{pmatrix}$$

Una aplicación típica de un Jacobiano es la generación de movimiento de un brazo robótico (ver Figura 3.3). La cinemática inversa se utiliza para calcular los ángulos de las articulaciones. Las lineas punteadas indican el movimiento exacto del brazo cuando cambia q_i. El movimiento predecido por el Jacobiano, la aproximación lineal, se muestra con la linea negra. Cerca al punto de la aproximación la predicción es normalmente correcta. Sin embargo, entre mayor sea la distancia, menos precisa es la predicción.

Otra aplicación de los Jacobiano es el filtro extendido de Kalman (Extended Kalman Filter (EKF)).

3.2 ROBÓTICA PROBABILÍSTICA

Incertidumbre

La *incertidumbre* esta presente en muchos de los problemas de robótica. Muchos de los sensores y actuadores solo tienen un grado de exactitud limitado, además su área de trabajo es limitada. Cuando se mide la distancia a un objeto, es poco probable que esta, sea la correcta. Una característica del ambiente puede ser clasificada erróneamente o inclusive la marca en sí puede ser un error del sensor. Los actuadores no necesariamente ejecutan las tareas tal y como se ordenan. Las actividades que se realizan en general en el mundo a diario pueden ser impredecibles, los fenómenos naturales aún con las leyes físicas que los rigen pueden tener comportamiento inesperado. No importante las acciones y mediciones que realice el robot, este no puede estar seguro de lo que sucede a su alrededor.

Robótica probabilística

La idea principal de la *robótica probabilística* es incorporar la incertidumbre en los modelos. El objetivo no esta solo en crear una mejor suposición, si no, una distribución de creencia matemática sobre todo el espacio de estados. Las ambigüedades pueden representarse y dependiendo de la incertidumbre del estado el robot puede responder correspondientemente. En general, al contemplar todas las posibles situaciones, un modelo probabilístico puede utilizarce como un mejor modelo de un evento. La desventaja de este método es que el tiempo computacional se incrementa. La mayor parte de los beneficios de la robótica probabilística se logran utilizando el teorema de Bayes (ver sección 3.1.3), y la extensión temporal del teorema de Bayes, el filtro de Bayes.

En esta sección se introducirán los conceptos básicos de robótica probabilística seguida por una discusión de algunos aspectos básicos, problemas y posibles soluciones.

3.2.0.1 *Conceptos Básicos*

Agente

Un *agente* es normalmente un robot que se mueve e interactua con su ambiente. El ambiente en el cuál este robot interactua se llama mundo.

Mundo

El *mundo* puede ser desde una tablero de juego, parte de un cuarto, un piso completo o un área grande como un estadio. En el mundo, el agente utiliza sus sensores disponibles para medir las propiedades

Percepciones

del mundo. Las observaciones son representadas internamente a través de estructuras de datos llamadas *percepciones*. Las observaciones de los objetos son usualmente adquiridas en forma de planos coordenados locales. El robot es el marco coordenado origen, y la medición es relativa a la posición de la percepción en este marco. Las percepciones solo existen mientras que una propiedad del ambiente este siendo

Modelo

observada.

Un *modelo* es una abstracción de una propiedad del mundo, son validos a través del tiempo y no dejan de existir aún que la percepción ya no exista. Cualquier fenómeno o actividad que realice un robot puede modelarse, ya sea la ejecución de movimientos o la medición

Espacio de estados

de odometría. Las variables de un modelo suelen llamarse *espacio de estados*.

Para algunas tareas, ciertas propiedades del mundo son mas relevantes que otras. En un modelo de SLAM, identificar marcas en el entorno ("*landmarks*") o medir los movimientos del robot son sumamente importantes pues afectan directamente en la estimación de la

Landmarks

Marcas estáticas

posición del robot. Existen dos tipos de marcas, *marcas estáticas*; son propiedades del entorno que no cambian su ubicación o textura, y

Marcas dinámicas

marcas dinámicas; son aquellas que pueden cambiar su ubicación y/o textura, por ejemplo un automóvil en movimiento o una persona. En este contexto los entornos pueden clasificarse en: Ambientes estáticos y ambientes dinámicos. Como el nombre lo sugiere, un ambiente estático es un entorno controlado, sin movimiento y sin cambios, por

lo que el espacio de estados para describir un ambiente dinámico es mucho más grande.

3.2.0.2 *Modelos.*

Localización

En general existen tres tipos de tareas de modelado. La primera es una tarea de *localización*. El entorno con sus características y features son percibidas por el agente. El agente debe localizarse a si mismo en este entorno utilizando conocimiento previo, es decir, el mapa del entorno. La segunda es *SLAM* . El agente se encuentra en un entorno completamente desconocido y necesita crear un mapa propio y localizarse en este al mismo tiempo. La tercer tarea involucra *seguimiento de objetos* en un entorno, p. ej. predecir el movimiento de otro robot o el de una pelota en una cancha de fútbol.

SLAM

Seguimiento de objetos

3.2.0.3 *Localización.*

La tarea de localización de la sección anterior puede dividirse en diferentes categorías:

Seguimiento de la posición, en donde la posición inicial del agente es conocida. La posición se actualiza agregando datos de odometría y las percepciones se utilizan para corregir la predicción.

Seguimiento de posición

En contraste con la localización global, la posición inicial es desconocida. El robot debe estimar su pose en el entorno y después rastrearla utilizando control y mediciones de las landmarks. El problema es que la posición del robot no puede ser medida directamente. En general las distribuciones unimodales son suficientes para el seguimiento de trayectorias pero no para localización global.

Problema del robot secuestrado

El *problema del robot secuestrado* es una extensión de la localización global mas complicada. La complejidad radica en que el problema contiene todos los retos del problema de localización global pero el agente puede teletransporse, es decir, puede moverse a diferentes lugares sin el conocimiento del agente. El agente tiene entonces que identificar cuando es transportado y en dónde fue ubicado después de la reubicación. Generalmente sí es posible atacar el problema del robot secuestrado, el desempeño de la localización mejora en robustez debido a que el agente realmente nunca sabe cuál es su posición correcta.

3.2.0.4 *Aproximaciones pasivas y activas.*

El modelado en robótica probabilística puede realizarse bajo dos aproximaciones: pasivas o activas. En una *aproximación pasiva* el modelo sólo percibe el mundo, no tiene control sobre el agente y no explora activamente el entorno ni planea trayectorias, a fin de que la percepción de objetos se vea maximizada. En contraste, las *aproximaciones activas* sí exploran activamente el entorno y planean trayectorias para

Aproximación pasiva

Aproximación aciva

el robot. El modelo toma acciones para mejorar la cantidad y/o calidad de la observación. En general la aproximación activa es más útil aunque juega un papel crucial en el comportamiento del robot.

3.2.0.5 *Agente simple y multi agente.*

Agente simple

Un escenario en donde un robot modela algunos aspectos del mundo se denomina escenario de *agente simple*. El agente sólo utiliza sus propios sensores para recolectar datos del ambiente.

Multiagente

Si existen diversos agentes en un entorno el escenario se convierte en uno *multiagente*. El conocimiento de cada agente puede utilizarce para mejorar el modelo de ellos mismos. Al tener más agentes, diferentes características del entornose puede medir y los agentes pueden interactuar para compartir dicha información.

Sin embargo, más información no siempre conlleva a mejores modelos. Deben tomarse en cuenta primeramente los problemas asociados con sistemas distribuidos, además, debe tomarse en cuenta que el error no se acumula en todos los agentes por lo que la información en cada robot no debe ser muy vieja. Debe priorizarse la información, es decir, la información local es más actualizada y potencialmente menos ruidosa. Aún con los nuevos problemas, los escenarios multi agente ofrecen la posibilidad de mejorar el modelo de manera muy sobresaliente.

3.2.1 *Estimadores de Estado.*

Como se mencionó en el capítulo anterior los estimadores de estado son el principal instrumento para estimar distribuciones de estado, agregando control y mediciones. En esta sección se explicará el funcionamiento básico del filtro de Bayes (ver sección 3.3.1) y el principio del filtro de Kalman visto como un observador de estados discretos (ver sección 3.3.2), sin embargo, la implementación del filtro extendido de Kalman como una solución del problema de Visual Simultaneous Localisation and Mapping (VSLAM) se estudia en la sección 4.2.2. Se ha decidido incluir sólo estos dos estimadores con el afán de dar a conocer al lector los métodos básicos por los cuales se ha atacado el problema de SLAM, sin embargo, este trabajo está enfocado en encontrar una alternativa a los problemas de localización mencionados en la sección 3.2.3.

FILTRO DE BAYES El filtro de Bayes es un filtro que calcula la distri-

Creencia

buciones de creencia. Una *creencia* representa el conocimiento interno del robot acerca del estado del mundo. Esto no es necesariamente un estado verdadero del mundo. Matemáticamente, la creencia es lo mismo que la distribución de probabilidad sobre el estado. Sólo es el estado de conocimiento del agente. El filtro de Bayes asigna todo el estado posible dependiendo de la información con la que cuenta

el agente hasta ese momento. Calcular la *distribución de creencia* se realiza en dos pasos: el paso de predicción y el paso de corrección. Después de incorporar todas las mediciones y controles anteriores $z_{1:t}$ y $u_{1:t}$, el filtro de Bayes regresa la creencia $(belief)$ $bel(x_t)$:

Distribución de creencia

$$bel\left(x_t\right) = P\left(x_t \mid z_{1:t}, u_{1:t}\right) \tag{3.2}$$

Existen dos conceptos importantes del filtro de Bayes: la probabilidad de transición de estado y la probabilidad de la medición.

Probabilidad de transición de estado

La *probabilidad de transición de estado* describe la probabilidad de estar en el estado x_t dados todos los estados previos, controles y mediciones:

$$P\left(x_t \mid x_{0:t-1}, z_{1:t}, u_{1:t}\right)$$

Bajo la suposición que x_{t-1} es un estado completo (ver sección 3.1.4) la ecuación es igual a:

$$P\left(x_t \mid x_{t-1}, u_t\right) \tag{3.3}$$

esta fórmula es más fácil y menos costosa computacionalmente para calcular la probabilidad para cada estado dado, el estado previo y el control. La ecuación 3.2 se simplifica para representar el estado antes de incorporar la medida más nueva z_t:

$$\overline{bel}\left(x_t\right) = P\left(x_t \mid z_{1:t}, u_{1:t}\right)$$

junto con la ecuación 3.3, esta ecuación se utiliza para calcular el paso de predicción.

Probabilidad de la medición

La *probabilidad de la medición* describe la probabilidad de obtener una medición z_t dado todo el estado anterior, mediciones y controles.

$$P\left(z_t \mid x_{0:t}, z_{1:t}, u_{1:t}\right)$$

Bajo la suposición de que x_t es un estado completo, esto puede simplificarse a:

$$P\left(z_t \mid x_t\right)$$

Con estos cuatro conceptos (la probabilidad de transición de estado, la creencia antes de incorporar la medición $\overline{bel}(x_t)$, la probabilidad de medición, y el estado después de incorporar la medición $bel(x_t)$, la predicción y el paso de la corrección del filtro de Bayes puede establecerse como:

Predicción: El paso de predicción combina la creencia anterior $bel(x_{t-1})$ con el control actual u_t. Es decir, dada la creencia del estado anterior $bel(x_{t-1})$ y el control actual u_t, la predicción para la creencia actual $\overline{bel}(x_t)$ se calcula integrando sobre el producto del antecesor, el estado previo $bel\left(x_{t-1}\right)$,y la probabilidad de la transición de estado.

Algoritmo 3.1 Algoritmo del filtro de Bayes.

def Bayes_filter($bel\,(x_{t-1}, u_t, z_t)$):
for x_t **en todo** X:
//Predicción
$\overline{bel}\,(x_t) = \int \underbrace{P\,(x_t \mid x_{t-1}, u_t)}_{\text{probabilidad de estado}} bel\,(x_{t-1})\,dx_{t-1}$
//Corrección
$bel\,(x_t) = \eta \quad \underbrace{P\,(z_t \mid x_t)}_{\text{probabilidad de mediciones}} \quad \overline{bel}\,(x_t)$
return $bel\,(x_t)$

$$\overline{bel}\,(x_t) = \int P\,(x_t \mid x_{t-1}, u_t)\,bel\,(x_{t-1})\,dx_{t-1}$$

Corrección: En el paso de corrección, las mediciones z_t mejoran el estado predecido $\overline{bel}\,(x_t)$. La creencia final $bel(x_t)$ es la distribución del producto normalizado de la creencia predecida $\overline{bel}(x_t)$ y la probabilidad de la medición.

$$bel\,(x_t) = \eta P\,(z_t \mid x_t)\,\overline{bel}\,(x_t)$$

Los dos pasos se repiten para todos los posible estados. El algoritmo completo puede observarse en el algoritmo 3.1.

3.2.1.1 *Filtro de Kalman*

Modelo dinámico y de observación

El filtro de Kalman es un estimador óptimo de un sistema dinámico y lineal, basado en observaciones ruidosas y en un modelo de la incertidumbre de la dinámica del sistema. Considerese un sistema discreto, representado por su modelo dinámico y de observación siguientes:

$$\begin{aligned} x\,(k+1) &= \Phi x\,(k) + \Gamma u\,(k) + \Gamma_j w\,(k) \\ y\,(k) &= H x\,(k) + u\,(k) \end{aligned}$$

en donde $w(k)$ es el ruido de la medición de la odometría y $u\,(k)$ el ruido de la medición de la profundidad de la marca en el escenario.

Es útil recordar que el ruido tiene media cero

$$\varepsilon\,\{w\,(k)\} = \varepsilon\,\{v\,(k)\} = 0$$

no tiene correlación en el tiempo o es ruido blanco expresado como sigue:

$$\varepsilon\left\{w\,(i)\,w^T\,(j)\right\} = \varepsilon\left\{v\,(i)\,v^T\,(j)\right\} = 0$$

$$j \neq i$$

la covarianza esta caracterizada por

$$\varepsilon\left\{w\left(k\right)w^{T}\left(k\right)\right\}=R_{w},\varepsilon\left\{v\left(k\right)v^{T}\left(k\right)\right\}=R_{v}$$

Ganancia de Kalman

El objetivo del filtro de Kalman es encontrar una ganancia L óptima tal que la ecuación del estimador se acerque a los estados lo más eficientemente posible, satisfaciendo una funcional de costo basado en la covarianza de las variables. La ecuación del estimador es:

Estimación

$$\hat{x}\left(k\right)=\overline{x}\left(k\right)+L\left(k\right)\left(y\left(k\right)-H\overline{x}\left(k\right)\right)$$

en donde

$$L\left(k\right)=P\left(k\right)H^{T}R_{v}^{-1}$$

en donde $P\left(k\right)$ es la matriz de covarianza de la estimación actual en función de la covarianza de la estimación anterior $\left(M(k)\right)$

$$P\left(k\right)=\left[M^{-1}+H^{T}R_{v}^{-1}H\right]^{-1}$$

Al utilizar las dos anteriores expresiones se realiza una actualización en la recolección de datos por la cámara y la central inercial.

Actualización

Sin embargo, así como en un estimador, es útil saber cual es el estado del sistema entre mediciones, puede utilizarse el modelo para interpolar el comportamiento del mismo.

$$\bar{x}\left(k+1\right)=\Phi\hat{x}\left(k\right)+\Gamma u\left(k\right)$$

y la respectiva covarianza de la medición:

$$M\left(k+1\right)=\Phi P\left(k\right)\Phi^{T}+\Gamma_{j}R_{w}\Gamma_{j}^{T}$$

donde las condiciones iniciales se asumen cero.

Ya que la covarianza M es variante con el tiempo, la estimación de la ganancia del estimador L también lo será. La formulación del estimador es bastante similar a los estimadores no lineales. Sin embargo, cuando se habla de un modelo no lineal es necesario utilizar una aproximación del filtro de Kalman tal como el filtro de Kalman extendido (EKF) .

3.2.1.2 *Filtro de Kalman Extendido*

El filtro de Kalman extendido (FKE) puede superar el problema de la no linealidad de un modelo dinámico. En lugar de utilizar matrices de transición $\left(\Phi,\Gamma,H\right)$ se utilizan funciones no lineales para el modelo dinámico y el modelo de observación:

$$\begin{aligned}x_{t} &= g\left(u_{t},x_{t-1}\right)+\varepsilon_{t}\\ z_{t} &= h\left(x_{t}\right)+\delta_{t}\end{aligned}$$

El filtro de Kalman extendido utiliza la expansión de Taylor para linealizar las funciones no lineales. La linealización de g y h se calculan por medio de sus Jacobianos. Para ver el computo completo de dichas matrices ver la sección 4.2.0.2.

3.2.1.3 *Unscented Kalman Filter (UKF)*

El filtro UKF establece su origen en la intuición de que es más sencillo aproximar una distribución de probabilidad que una función arbitraría no lineal o transformación [19]. A diferencia del FKE el UKF no linealiza las funciones por medio de expansiones de Taylor, si no que reconstruye la nueva distribución de probabilidad escogiendo los puntos de muestreo cuidadosamente. Dichos puntos se denominan puntos sigma y el método para reconstruír la distribución se le denomina "Unscented transformation" (UT) . Debido a que el UKF no utiliza Jacobianos, también se le llama el filtro de Kalman sin derivación. El filtro UKF tiene algunas ventajas sobre el EKF, entre ellas, que el método de linealización tiene mayor exactitud.

Puntos Sigma

Unscented Transformation

Unscented Transformation. Los puntos sigma se eligen de tal manera que su media y covarianza sean exactamente $x^a_{k.1}$ y P_{k-1} . Cada punto sigma se propaga entonces a travéz de la no linealidad produciendo al final una nube de puntos transformados. La nueva media estimada y covarianza se calculan basados en su estadistica. Este proceso se llama transformación sin aroma (unscented transformation (UT)). Esta transformación es un método para calcular la estadistica de una variable aleatoria la cual sufre una transformación no lineal [38].

Considere el siguiente sistema no lineal, y el modelo de observación con ruido aditivo.

$$\begin{aligned} x_k &= f(x_{k-1}) + w_{k-1} \\ z_k &= h(x_k) + v_k \end{aligned}$$

El estado inicial x_0 es un vector aleatorio con media conocida $\bar{x} = E[x_0]$ y covarianza $P_0 = E\left[(x_0 - \mu_0)(x_0 - \mu_o)^T\right]$. Para calcular la estadística de z_k, se calcula una matriz χ de $2L+1$ vectores sigma χ_i (con pesos correspondientes W_i), con respecto a lo siguiente:

$$\begin{aligned} \chi_0 &= \bar{x} \\ \chi_i &= \bar{x} + \left(\sqrt{(L+\lambda)P_x}\right), i = 1, \ldots, L \\ \chi_i &= \bar{x} - \left(\sqrt{(L+\lambda)P_x}\right)_{i-L}, i = L+1, \ldots, 2L \\ W_0^{(m)} &= \lambda / (L+\lambda) \\ W_0^{(c)} &= \lambda / (L+\lambda) + \left(1 - \alpha^2 + \beta\right) \\ W_i^{(m)} &= W_i^{(c)} = 1 / \{2(L+\lambda)\} \; i = 1, \ldots, 2L \end{aligned}$$

donde $\lambda = \alpha^2 (L + \kappa) - L$ es un parametro de escala, α determina la propagación de los puntos sigma alrededor de $\bar{x}$ y usualmente se establece como un valor positivo y pequeño (e.g., 1e-3). κ es un parametro secundario de escala que usualmente tiene valor de cero, y β se utiliza para incorporar conocimiento previo de la distribución (para distribuciones gausianas, $\beta = 2$ es óptimo). $\left(\sqrt{(L + \lambda) P_x}\right)_i$ es la i-esima fila de raiz cuadrada de la matriz. Los vectores sigma se propagan a travéz de la función no lineal,

$$z_i = h(\chi_i) \, i = 0, \ldots, 2L$$

Entonces la media y la covarianza de reconstruyen desde la media ponderada de los puntos sigma transformados z como sigue:

$$\begin{aligned} \bar{y} &= \sum_{i=0}^{2n} W_i^m z_i \\ \overline{\sum} &= \sum_{i=0}^{2n} W_i^{(c)} (z_i - \bar{x}) (z_i - \bar{x})^T \end{aligned}$$

Sin embargo, también existen algunas desventajas. El UKF es más lento que el EKF por un factor constante y también es más difícil de implementar por lo que se deben evaluar las condiciones del ambiente (dinámica del ambiente) y el equipo computacional disponible para su correcto aprovechamiento [39].

4

LOCALIZACIÓN Y MAPEO VISUAL SIMULTÁNEOS

4.1 INTRODUCCIÓN

Segmentación del problema de SLAM

La adquisición de mapas para ambientes cerrados, donde típicamente los GPS no se encuentran disponibles, ha sido un punto de enfoque muy grande para la comunidad de investigadores en las ultimas decadas. Aprender mapas bajo incertidumbre de la posición, es referida a menudo como el problema de SLAM. En la literatura se ha propuesto una gran variedad de soluciones, dichas aproximaciones pueden clasificarse en filtraje y suavizado.

Modelo del filtro

El *modelo del filtro* estima el estado en línea donde el estado del sistema consiste en la posición del robot y el mapa. La estimación se calcula y se corrige incorporando nuevas mediciones en cuanto están disponibles. Existen técnicas populares como Kalman y filtros de información [33],[17], filtros de partículas [6],[4], [14], o filtros de información [11],[5] que caen en esta categoría. Las técnicas de filtraje, se han bautizado como métodos "*on-line*" de SLAM.

Métodos on-line

Metodo de suavizado

Por otro lado, el *método de suavizado* estima la trayectoria completa del robot a partir del conjunto completo de mediciones [23],[22],[30], estas aproximaciones se especializan en resolver el problema completo de SLAM ("*full* SLAM") y típicamente trabajan en técnicas de minimización del error por medio de mínimos cuadrados.

Investigación en SLAM

Desde que se comenzó el estudio del problema de VSLAM las investigaciones se han diversificado, obtener un buen algoritmo de VSLAM depende de encontrar una sinergia efectiva entre diversos aspectos, de entre los cuales los más importantes son:

- Modelo de predicción.
- Estimador.
- Asociación de datos.
- Cerrado de bucle.
- Relocalización.
- Identificación de objetos.
- Coherencia del mapa.
- Costo computacional.

Sin duda, incluir todos los puntos anteriores en un simple algoritmo es la parte difícil de la solución del problema. Con el afán de poder

cubrir todos estos aspectos, La investigación en SLAM se ha dividido en dos grandes grupos: Back-end SLAM y Front-end SLAM [13].

Separar el análisis de SLAM en estos grupos se logra convirtiendo al problema en un análisis de grafos. El propósito de este capítulo es introducir el problema de SLAM en su forma probabilística y describir los principios de la metodología basada en grafos para su solución.

En la siguiente sección se hablará de la construcción del grafo con técnicas de filtraje tradicionales, generalmente llamado *front-end* SLAM.

4.2 FRONT-END SLAM

Formulación por grafos

Una forma intuitiva de atacar el problema de SLAM es por medio de la *formulación por grafos*. Para poder resolver el problema mediante esta perspectiva debe construirse un grafo cuyos nodos representen posiciones del robot o marcas en el mapa y en el cual cada arista entre dos nodos represente una medición del sensor que restrinja las posiciones conectadas. Por supuesto dichas restricciones pueden ser contradictorias pues las mediciones están afectadas por ruido. Una vez que tal grafo esta construido, el problema crucial es encontrar la configuración de los nodos que es maximamente consistente con las mediciones. Esto requiere resolver un problema de minimización del error bastante largo.

La formulación por grafos del SLAM fue propuesta por Lu y Milios en 1997 [23], sin embargo, no encontró popularidad hasta más de una década después debido a que la gran complejidad del problema no permitía u resolución por medio de técnicas estandar.

4.2.0.1 *Formulación probabilística del SLAM*

Resolver el problema de SLAM consiste en estimar la trayectoria del robot y el mapa del entorno mientras éste viaja en él. debido al ruido inherente de las mediciones del sensor, un problema de SLAM es descrito usualmente mediante herramientas probabilísticas. Se asume que el robot se mueve en un ambiente desconocido a lo largo de una trayectoria se describe por una secuencia de variables aleatorias $x_{1.T} = \{x_1, \ldots, x_T\}$. Mientras que en el movimiento adquiere una secuencia de mediciones de odometría $u_{1.T} = \{u_1, \ldots, u_T\}$ y percepciones del ambiente $z_{1.T} = \{z_1, \ldots, z_T\}$. Resolver el problema "*full* SLAM" consiste en estimar la probabilidad a posteriori de la trayectoria del robot $x_{1:T}$ y el mapa m del entorno dadas todas las mediciones, más la posición inicial x_0:

Full SLAM

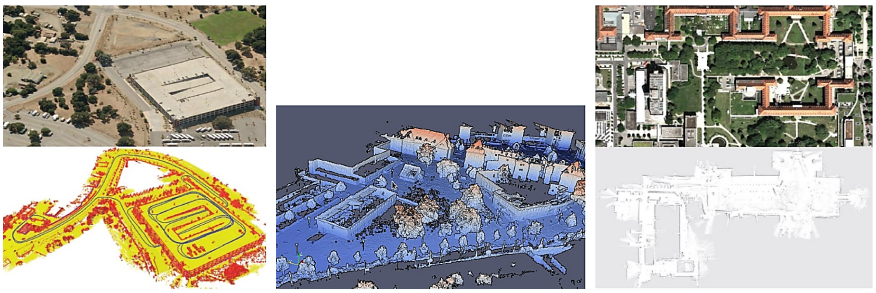
$$p\left(x_{1:T}, m \mid z_{1:T}, u_{1:T}, x_0\right) \tag{4.1}$$

La posición inicial x_0 define la posición del mapa y puede ser elegida arbitrariamente. Para simplificar la notación, en este trabajo se

omitirá x_0. Las posiciones $x_{1:T}$ y la odometría $u_{1:T}$ son usualmente representadas como transformaciones 2D o 3D en $SE(2)$ o $SE(3)$, mientras que el mapa puede representarse en diferentes maneras.

Tipos de mapas

Los mapas pueden parametrizarse como un conjunto de marcas ubicadas de manera espaciada, por representaciones densas como "occupancy grids", mapas de superficie o por mediciones aleatorias de sensores. Elegir una representación del mapa depende de los sensores utilizados, de las características del ambiente y del algoritmo de estimación. Las marcas en un mapa [23],[6] se utilizan preferentemente en ambientes en donde los "features" pueden identificarse fácilmente y especialmente cuando se utilizan cámaras. En contraste, las representaciones densas [36],[4],[14]se utilizan generalmente junto con sensores de rango. Independientemente del tipo de representación, el mapa se define como las mediciones y las ubicaciones en donde estas fueron hechas [21], [16]. La Figura 4.1 muestra tres mapas densos para 3D: mapas de superficie multinivel, nubes de puntos y "occupancy grids".



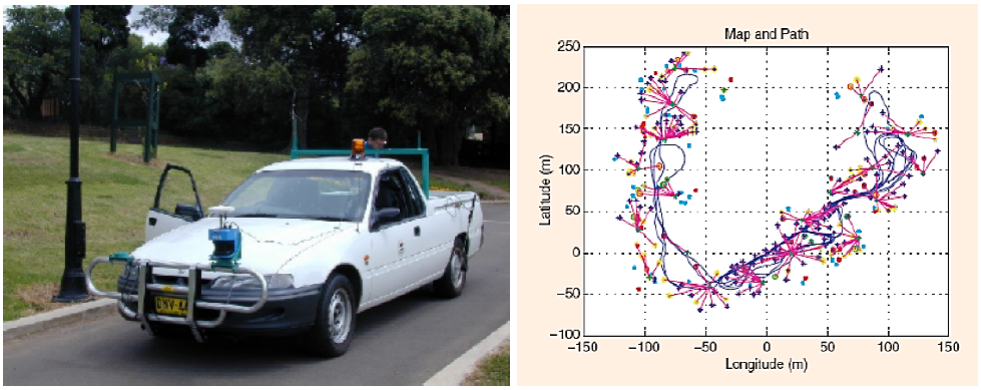
(a) (b) (c)

Figura 4.1: Tipos de mapas en SLAM. a) Un mapa 3D del estacionamiento en Stanford adquirido con un auto instrumentado y su vista satélite en la parte superior. b) Mapa de puntos de la Universidad de Freiburg. (http://www.paraview.org/lidar/). c) Mapa de celdas adquirido en el hospital de Freiburg y su vista satélite en la parte superior.

La Figura 4.2 muestra un mapa basado en marcas 2D.

Estimar el estado anterior dado en (4.1) implica realizar operaciones en espacios de estado multidimensionales. El problema de SLAM no podría manejarse si no existiera una estructura bien definida del problema. Esta estructura se plantea a partir de ciertas suposiciones comunes, como lo es suponer un espacio estático y la suposición de Markov. Un modo conveniente de describir esta estructura es por medio de una *red dinámica de Bayes* (Dynamic Bayesian Network (DBN)) representada en la Figura 4.3. Una red bayesiana es un modelo gráfico que describe un proceso estocástico como un grafo dirigido. La gráfica tiene un nodo por cada variable aleatoria en el proceso y una arista dirigido (o flecha) entre dos nodos que modela una dependencia condicional entre ellos. En la Figura 4.3, se pueden distinguir no-

Red dinámica de Bayes



(a) (b)

Figura 4.2: Mapa 2D de el parque Victoria, hecho por la Universidad de Sydney.

dos azules indicando las variables observadas ($z_{1:T}$ y $u_{1:T}$) y nodos blancos que son las variables ocultas. Las variables ocultas $x_{1:T}$ y el mapa m modelan la trayectoria del robot y el mapa del entorno respectivamente. La conectividad de la DBN sigue un patrón recurrente caracterizado por el modelo de la transición del estado y el modelo de observación. El modelo de transición $p(x_t \mid x_{t-1}, u_t)$ se representa mediante los dos aristas apuntando a x_t y representa la probabilidad que el robot al tiempo t se encuentra en x_t dado que en el tiempo $t-1$ se encontraba en x_{t-1} y obtuvo una medición de odometría u_t.

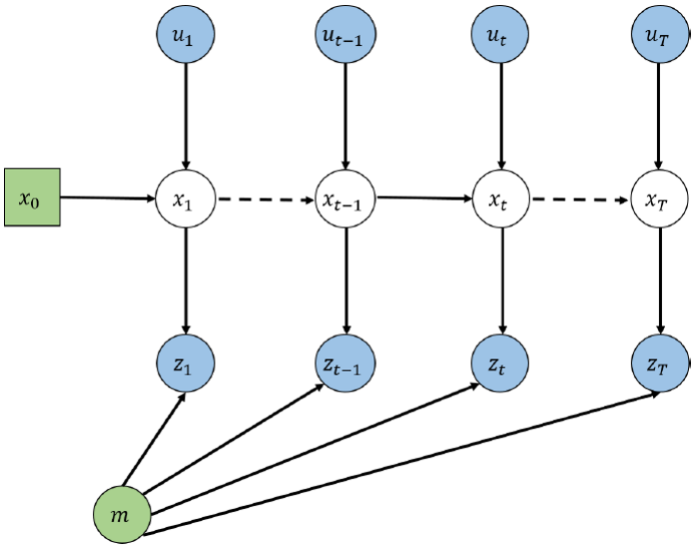


Figura 4.3: Red Dinámica Bayesiana para un proceso de SLAM.

Modelo de observación

El *modelo de observación* $p(z_t \mid x_t, m_t)$ modela la probabilidad de realizar la observación z_t dado que el robot se encuentra en la posición x_t dentro del mapa y se representa por las flechas entrando a z_t. La observación exteroceptiva z_t depende sólo de la ubicación actual x_t

del robot y del mapa estático m. Expresar el SLAM como un DBN resalta su estructura temporal y, por lo tanto, este modo de representar el problema se adapta muy bien para describir el proceso de filtraje que puede utilizarse para resolver el problema de SLAM.

Formulación basada en redes

Una representación alternativa a la DBN es la *formulación basada en grafos o en redes*, que resaltan la subyacente estructura espacial. En el SLAM basado en grafos, las posiciones del robot se modelan por nodos en una gráfica y se etiquetan con su posición en el ambiente [23], [16]. Las restricciones espaciales entre las posiciones que resultan de las observaciones z_t o de la mediciones de odometría $u_{1:T}$ son codificadas en aristas entre los nodos. Un algoritmo basado en grafos de SLAM, construye una gráfica fuera de las mediciones directas del sensor. Cada nodo en el grafo representa una posición del robot y una medición adquirida en esa posición. Una arista entre dos nodos representa una restricción espacial relacionando las dos posiciones del robot.

Una restricción consiste en una distribución de probabilidad sobre las transformaciones relativas entre dos posiciones. Estas transformaciones son mediciones de odometría entre dos posiciones consecutivas del robot o se determinan alineando las observaciones hechas por el robot entre dos posiciones consecutivas mediante asociación de datos.

4.2.0.2 *Solución del problema de SLAM con Filtro de Kalman Extendido (FKE)*

Los algoritmos basados en EKF, operan con incertidumbre de primer orden en la estimación de posición del robot y posición de los *features* del mapa. En estos tipos de métodos, el estado x del sistema se representa como un vector que puede ser dividido en el estado de la posición del robot (o de la cámara) xy la posición de los features z_i. El vector de estado es acompañado por una *matriz de covarianza* P que puede representarse como sigue:

Matriz de covarianza

$$x = \begin{pmatrix} \widetilde{x} \\ \widetilde{z_1} \\ \widetilde{z_2} \\ \vdots \end{pmatrix}, P = \begin{bmatrix} P_{xx} & P_{xy_1} & P_{xy_2} & \cdots \\ P_{y_1x} & P_{y_1y_1} & P_{y_1y_2} & \cdots \\ P_{y_2x} & P_{y_2y_1} & P_{y_2y_2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

La tarea de la matriz de covarianza es representar la incertidumbre entre todas las variables en el vector de estado.

Las estimaciones $\widetilde{z_i}$ pueden agregarse o eliminarse del vector de estado a través del tiempo debido a que algunas marcas características en el entorno pueden salir del campo de visión y algunas otras presentarse de un momento a otro, por ejemplo, al dar una vuelta por tanto, los vectores x y P pueden crecer o reducirse dinámicamente. En una operación normal $\tilde{x}$ y P cambian en dos pasos:

1. Durante el movimiento el paso de predicción utiliza un modelo de movimiento para calcular cómo se mueve el robot y cómo la incertidumbre de la posición se incrementa.

2. Cuando se obtiene posición de un *feature*, un modelo de medición describe cómo la incertidumbre de la posición del robot y el mapa pueden reducirse (actualización).

Es crítico mantener una matriz de covarianza P completa con elementos fuera de la diagonal, estos elementos representan la correlación entre las estimaciones, característica inherente en la construcción de un mapa pues cada estimación de un estado afecta directamente la estimación de los demás.

Se define un marco coordenado fijo en el mundo W y un marco coordenado respecto a la cámara R. Para eliminar problemas de singularidades y linealización se utiliza preferentemente una representación mediante cuaterniones, por lo que el vector que se utiliza para representar la posición del robot x_p es:

$$\tilde{x} = \begin{pmatrix} r^W \\ q^{WR} \end{pmatrix} = \begin{pmatrix} x & y & z & q_0 & q_x & q_y & q_z \end{pmatrix}^T$$

de manera general el algoritmo del observador puede estudiarse en 4.1.

Algoritmo 4.1 Algoritmo para SLAM basado en EKF.

$x_0^W = 0;\ P_0^W = 0$ { Inicialización del mapa }
$[z_0\,,\ \hat{x_0}]$ =obtener_mediciones
$\left[x_0^W\,,\ P_0^W\right]$ =agregar_nuevos_features $\left(x_0^W,\ P_0^W,\ z_0,\ \hat{x_0}\right)$
for $k = 1$ **to steps do**
$\left[\hat{x}_{R_k}^{R_{k-1}},\ Q_k\right]$ =obtener_odometría { Predicción de EKF }
$\left[\hat{x}_{k|k-1}^W,\ P_{k|k-1}^W\right]$ =calcular_movimiento $\left(\hat{x}_{k-1}^W,\ P_{k-1}^W,\ x_{R_k}^{R_{k-1}},\ Q_k\right)$
$[z_k,\ \hat{x}_k]$ =realizar_mediciones
H_k =asociación_de_datos $\left(x_{k|k-1}^W,\ P_{k|k-1}^W,\ z_k,\ \hat{x}_k\right)$
$\left[x_k^W,\ P_k^W\right]$ =actualizar_mapa $\left(x_{k|k-1}^W,\ P_{k|k-1}^W,\ z_k,\ \hat{x}_k, H_k\right)\ldots$ { Actualización EKF }
$\left[x_k^W,\ P_k^W\right]$ =agregar_features $\left(x_k^W,\ P_k^W,\ z_k,\ \hat{x}_k, H_k\right)$
end for

4.2.0.3 *Paso de predicción del FKE*

Cuando el vehículo se mueve de la posición $k-1$a la posición k, el modelo de su movimiento describirá la forma en la cual se genera el desplazamiento, Si se desea un modelo en el que la cámara esté manejada por un ser humano o algún robot del que no podamos saber la voluntad del movimiento, se recomienda utilizar el modelo

propuesto en Davison et. al. [29]. En este capítulo, para simplificar la explicación, se propone la odometría de un vehículo con ruedas del que es posible medir sencillamente la odometría como sigue:

$$\tilde{x}_{R_k}^{R_{k-1}} = u_{R_k}^{R_{k-1}} + v_k$$

donde v_k es el ruido de la medición debida al deslizamiento, el cua se considera aditiva, de media cero y de ruido blanco con covarianza Q_k. Esto significa que todas las v_k son mutuamente independientes [18]. después de este movimiento, la ubicación del robot será:

$$x_{R_k}^W = x_{R_{k-1}}^W \oplus x_{R_k}^{R_{k-1}}$$

donde $\oplus$ representa la composición de las transformaciones:

$$x_C^A = x_B^A \oplus x_C^B = \begin{bmatrix} x_1 + x_2 cos\phi_1 - y_2 sin\phi_1 \\ y_1 + x_2 sin\phi_1 - y_2 cos\phi_1 \\ \phi_1 + \phi_2 \end{bmatrix}$$

Por lo que, dado un mapa $m_{k-1}^K = \left(\tilde{x}_{k-1}^W, P_{k-1}^W\right)$ en el paso $k-1$, la predicción del mapa $m_{k|k-1}^W$ en el paso k se obtiene de la siguiente manera:

$$x_{k|k-1}^B = \begin{bmatrix} \tilde{x}_{R_{k-1}}^W \oplus \tilde{x}_{R_k}^{R_{k-1}} \\ \tilde{z}_{1,k-1}^W \\ \vdots \\ \tilde{z}_{i,k-1}^W \end{bmatrix}$$

$$P_{k|k-1}^W = F_k P_{k-1}^W F_k^T + G_k Q_k G_k^T$$

donde:

$$F_k = \left.\frac{\partial x_{k|k-1}^W}{\partial x_{k-1}^W}\right|_{\left(\tilde{x}_{R_{k-1}}^W, \tilde{x}_{R_k}^{R_{k-1}}\right)} = \begin{bmatrix} J_{1\oplus}\left\{\tilde{x}_{R_{k-1}}^W, \tilde{x}_{R_k}^{R_{k-1}}\right\} & 0 & \cdots & 0 \\ 0 & I & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & & I \end{bmatrix}$$

$$G_k = \left.\frac{\partial x_{k|k-1}^W}{\partial x_{k-1}^{R_{k-1}}}\right|_{\left(\tilde{x}_{R_{k-1}}^W, \tilde{x}_{R_k}^{R_{k-1}}\right)} = \begin{bmatrix} J_{2\oplus}\left\{\tilde{x}_{R_{k-1}}^W, \tilde{x}_{R_k}^{R_{k-1}}\right\} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

donde $J_{1\oplus}$ y $J_{2\oplus}$ son los jacobianos de la composición de transformación:

$$J_{1\oplus}\left\{\hat{x}_B^A, \hat{x}_C^B\right\} = \left.\frac{\partial\left(x_B^A \oplus x_C^B\right)}{\partial x_B^A}\right|_{\left(\hat{x}_B^A, \hat{x}_C^B\right)}$$

$$J_{2\oplus}\left\{\hat{x}_B^A, \hat{x}_C^B\right\} = \left.\frac{\partial\left(x_B^A \oplus x_C^B\right)}{\partial x_C^B}\right|_{\left(\hat{x}_B^A, \hat{x}_C^B\right)}$$

4.2.0.4 *Asociación de datos.*

Al paso k los sensores obtienen del entorno E_k un conjunto de mediciones $z = \{z_1, \ldots, z_m\}$. La asociación de datos consiste en determinar el origen de cada medición en terminos de las características (*features*) del mapa F_j, $j = 1, \ldots, n$. Considerese S como la covarianza de la estimación del error de mediciónz:

$$z_{k,i} = h_{k,j}\left(x_{k|k-1}^B + w_{k,i}\right)$$

donde $w_{k,i}$ es el ruido de la medición con covarianza $\Upsilon_{k,i}$ el cual se asume como aditivo, con media cero, blanco e independiente del ruido de la planta v_k. Para establecer la consistencia de la hipotesis $H = \begin{bmatrix} j_1 & j_2 & \cdots & j_m \end{bmatrix}$ asociando cada medición del entorno E_k con su *feature* correspondiente F_{j_1}($j_i = 0$ indicando que z_i no corresponde a ningún *feature* del mapa), las mediciones pueden ser juntamente predichas utilizando la función h_{H_k} :

$$h_{H_k}\left(x_{k|k-1}^W\right) = \begin{bmatrix} h_{j1}\left(x_{k|k-1}^W\right) \\ \vdots \\ h_{jm}\left(x_{k|k-1}^W\right) \end{bmatrix}$$

que puede ser linealizada alrededor de la mejor estimación:

$$h_{H_k}\left(x_{k|k-1}^W\right) \backsimeq h_{H_k}\left(\hat{x}_{k|k-1}^W\right) + S_{H_k}\left(x_k^W - \hat{x}_{k|k-1}^W\right);\ S_{H_k} = \begin{pmatrix} S_{j1} \\ \vdots \\ S_{jn} \end{pmatrix}$$

el error de la medición y su covarianza pueden calcularse como:

$$e_{H_k} = z_k - h_{H_k}\left(\hat{x}_{k|k-1}^W\right)$$

$$G_{H_k} = S_{H_k} P_k^W S_{H_k}^T + \Upsilon_{k,i}$$

El espacio de correspondencias entre las mediciones y los *features* se puede representar mediante un diagrama de árbol (ver Figura 14) en donde cada nodo en el nivel k representa un *feature* del mapa. Cada nodo tiene $n + 1$ ramas, una por cada *feature* detectado a priori,

representando una posible correspondencia o alternativa para la medición E_k, aimismo debe incluirse la posibilidad de que la asociación sea incorrecta o de que el *feature* esta repetido. Podría decirse que los algoritmos de asociación de datos deben seleccionar una de los $(n+1)^K$ interpretaciones como la hipótesis correcta. Una vez de que se obtiene una hipótesis, puede usarse para mejorar la estimación de x_k^W . Es necesario analizar la hipótesis utilizando distancia de Mahalanobis para verificar su consistencia en lo que se denomina Joint Compatibility (JC).

Compatibilidad conjunta

Se escoge la distancia de Mahalanobis debido a que su utilidad radica en que es una forma de determinar la similitud entre dos variables aleatorias multidimensionales. Se diferencia de la distancia euclídea en que tiene en cuenta la correlación entre las variables aleatorias.

Formalmente, la distancia de Mahalanobis entre dos variables aleatorias con la misma distribución de probabilidad $\vec{x}$ y $\vec{y}$ con matriz de covarianza Σ se define como:

$$d_m(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T\Sigma^{-1}(\vec{x}-\vec{y})}$$

Los algoritmos de asociación de datos pueden ser la clave de la solución del problema de SLAM por lo que a esta fecha es un tema de investigación de alto interés, la técnica más utilizada es la llamada "el vecino mas cercano" de Neira et. al. [28].

4.2.0.5 *Actualización del mapa: Paso de estimación del FKE*

Una vez decididas las correspondencias entre las mediciones z_k, se utilizan para mejorar la estimación del vector de estado estocástico utilizando las ecuaciones de actualización del EKF como sigue:

$$\hat{x}_k^W = x_{k|k-1}^W + K_{H_k} e_{H_k}$$

donde la ganancia del filtro K_{H_k} se obtiene de:

$$K_{H_k} = P_{k|k-1}^W S_{H_k}^T G_{H_k}^{-1}$$

Finalmente, el error de covarianza estimado del vector de estado es:

$$P_k^B = \left(I - K_{H_k} S_{H_k}\right) P_{k|k-1}^W$$

4.2.0.6 *Consistencia del filtro*

Debido a que SLAM es un problema no lineal verificar la consistencia del observador es de máxima importancia; una manera de hacerlo es utilizando la distancia de Mahalanobis tal que:

$$D_{H_k}^2 = e_{H_k}^T G_{H_k}^{-1} e_{H_k} < \chi_{d,1-\alpha}^2$$

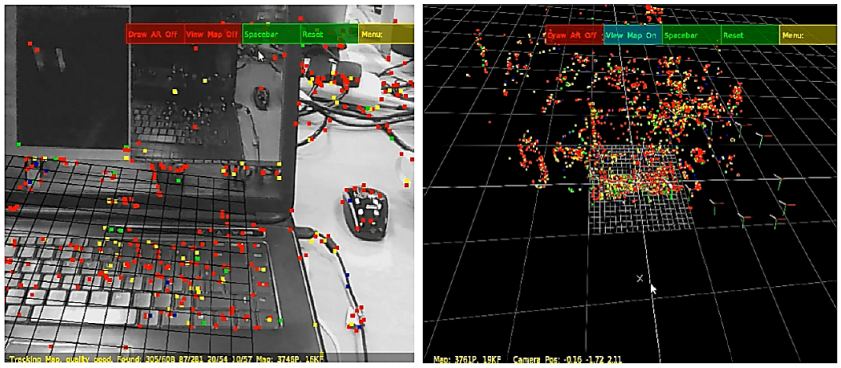
donde $d = dim\left(h_{H_k}\right)$y α es el nivel de seguridad. Este tipo de verificación se denomina *Joint Compatibility*.

4.3 PTAM COMO PLATAFORMA FRONT-END.

Para iniciar una nueva línea de investigación en el Departamento de Control Automático con el enfoque de optimización de mapas para VSLAM se planteó utilizar PTAM como plataforma front-end. PTAM es un método para estimar la posición de la cámara en un mapa desconocido, a diferencia de un sistema típico de SLAM, PTAM está diseñado específicamente para realizar el seguimiento de una cámara de mano en un espacio de trabajo pequeño de realidad aumentada.

PTAM divide el seguimiento y la generación del mapa en dos procesos diferentes, ejecutados en un procesador doble núcleo. Un hilo se ocupa de realizar el seguimiento de manera robusta del errático movimiento de la cámara, mientras que el hilo restante produce el mapa 3D de puntos *feature*s previamente observados por los fotogramas de la cámara. Esto permite aprovechar los recursos del CPU para resolver los computacionalmente costosos algoritmos de optimización. El resultado es un sistema que produce mapas detallados con miles de marcas del mapa a los que se les pude realizar un seguimiento a la velocidad de los 30 cuadros por segundo utilizados por la cámara. La Figura 4.4 muestra un mapa obtenido con este método.

Para implementar la generación del grafo utilizando este software debieron tomarse en cuenta algunos aspectos.



(a) Entorno (b) Mapa de puntos.

Figura 4.4: Mapa obtenido con PTAM.

Inicialmente PTAM no realiza un grafo a partir de las mediciones que realiza, sin embargo, puede ser utilizado como tal con las siguientes consideraciones:

- PTAM sólo utiliza los *feature*s vigentes para estimar la posición de la cámara, por lo que todas las marcas anteriores se eliminan del banco de datos, para la elaboración del grafo estas marcas deben quedar registradas por lo que es necesario crear un mapa simultáneo al de PTAM que almacene todos los *feature*s.

- Al no utilizar un algoritmo estocástico de SLAM tradicional, las matrices de correlación no existen. Puesto que crear las aristas de un grafo es la pieza más importante de la optimización, es necesario crear dichas matrices a partir de la estimación de los errores de medición y posición de la cámara que sí calcula PTAM. Sin embargo, la correlación entre los cálculos de las posiciones de los *feature*s no existe, por lo que las matrices de correlación serán diagonales.

- Como formato general el grafo se almacenó en un archivo .txt con la estructura del algoritmo 4.2.

Algoritmo 4.2 Realización del grafo con PTAM.

//declaración de vértices para las posiciones del robot
VERTEX_SE2 ID X Y q {Vértice con transformación SE2}.
VERTEX_SE3:QUAT ID x y z qx qy qz w {Vértice con transformaciones SE3 y rotación en quaternios}.
//declaración de posición de *feature*s
VERTEX_POINT_XY x y {Vértice *feature* con coordenadas (x,y)}
VERTEX_POINT_XYZ x y z {Vértice *feature* con coordenadas (x,y,z)}
//declaración de aristas para las mediciones de incertidumbre para odometría y *feature*s
EDGE_SE2 IDFrom IDTo σ_{xx} σ_{xy} $\sigma_{x\theta}$ σ_{yx} σ_{yy} $\sigma_{y\theta}$ $\sigma_{\theta x}$ $\sigma_{\theta y}$ $\sigma_{\theta\theta}$ {Arista desde un vertice a otro con matriz de incertidumbre}

4.4 BACK-END SLAM

Una vez construido el grafo, se busca la configuración de las posiciones del robot que mejor satisfacen las restricciones. Por lo tanto, en un SLAM basado en grafos el problema se descompone en dos tareas: construir el grafo de las mediciones originales, determinando la configuración más probable de las posiciones dados las aristas del grafo (optimización del grafo). La construcción del grafo es usualmente llamada "front-end" y es muy dependiente de sensores, mientras que la segunda parte se llama "back-end" y yace en una representación abstracta de los datos que es poco dependiente de sensores. La Figura 4.6 muestra un grafo de posiciones sin corregir y el grafo corregido correspondiente.

Uno de los aspectos más importantes en la construcción de un algoritmo para SLAM es determinar la actualización mas probable resultado de una observación. Esta decisión depende de la distribución de probabilidad sobre las posiciones del robot. El problema se conoce como asociación de datos y usualmente se resuelve con algoritmos de front-end SLAM. Para calcular una asociación de datos correcta, un *front-end* usualmente requiere una estimación consistente a partir de

la estimación anterior de la trayectoria del robot $p\left(x_{1:T}, m \mid z_{1:T}, u_{1:T}\right)$, por lo que alternar los métodos de *front-end* y *back-end* resulta útil mientras que el robot explora el entorno. Por lo tanto, la exactitud y la eficiencia del *back-end* es crucial para obtener un buen sistema de SLAM.

Para explicar esta sección se utilizará g^2o, un framework utilizado para Hiper optimización de grafos, que permitirá ejemplificar una aplicación de un front-end utilizando PTAM y optimizar el mapa obtenido de este por medio de la optimización de un grafo.

4.4.1 *Optimización de grafos con g^2o.*

Los programas de optimización de hiper grafos como g^2o resuelven un problema de mínimos cuadrados que puede describirse como:

$$F\left(x\right) = \sum_{k \epsilon C} \underbrace{e_k\left(x_k, z_k\right)^T \Omega_k e_k\left(x_k, z_k\right)}_{F_k}$$

$$x^* = \underset{x}{argmin} F(x) \tag{4.2}$$

donde:

- $x = \left(x_1^T, \ldots, x_n^T\right)^T$es un vector de parámetros, donde cada x_i representa un bloque de parámetros.
- $x_k = \left(x_{k_1}^T, \ldots, x_{k_q}^T\right)^T \subset \left(x_1^T, \ldots, x_n^T\right)$es el subconjunto de parámetros implicados en la $k - esima$ restricción.
- z_k y Ω_k representan la media y la matriz de información de la relación entre dos parámetros en x_k.
- $e_k\left(x_k, z_k\right)$es una función del vector de error que mide que tan bien el parámetro en x_k satisface la restricción z_k. Es cero cuando x_k y x_j coinciden perfectamente con la restricción.

Una interpretación gráfica de este grafo puede observarse en la Figura 4.5.

Si z_{ij} y Ωij son la media y matriz de información entre el nodo i y el nodo j. Esta medición virtual es una transformación que hace que las observaciones adquiridas en i se aproximen lo más posible con las observaciones adquiridas en j. Considerando $\hat{z}_{ij}\left(x_i, x_j\right)$como la predicción de una medición dada la configuración de los nodos x_i y x_j. Usualmente esta predicción es la transformación relativa entre los dos nodos. La probabilidad logarítmica I_{ij} de una medición z_{ij} es por lo tanto:

$$I_{ij} \propto \left[z_{ij} - \hat{z}_{ij}\left(x_i, x_j\right)\right]^T \Omega_{ij} \left[z_{ij} - \hat{z}_{ij}\left(x_i, x_j\right)\right]$$

y la función del error:

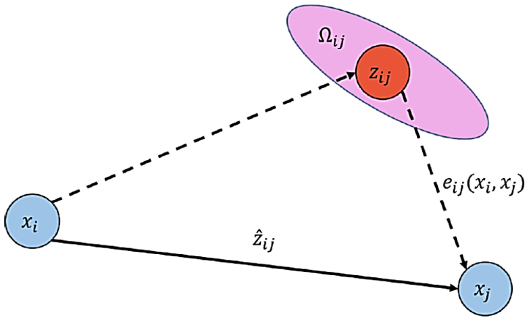


Figura 4.5: Incertidumbres en un grafo construido con front-end SLAM.

$$e_k\left(x_k, z_k\right) = z_{ij} - \hat{z}_{ij}\left(x_i, x_j\right)$$

Si se conoce una posición inicial de un robot $\widetilde{x}$ la solución numérica de (4.2) puede obtenerse utilizando el algoritmo de Gauss-Newton. La idea es aproximar la función del error por la expansión de Taylor de primer orden alrededor de la suposición actual $\widetilde{x}$.

$$\begin{aligned} e_{ij}\left(\tilde{x}_i + \Delta x_i, \tilde{x}_j + \Delta x_j\right) &= e_{ij}\left(\widetilde{x} + \Delta x\right) && (4.3) \\ &= e_{ij} + J_{ij}\Delta x && (4.4) \end{aligned}$$

donde J_{ij}es el Jacobiano de $e_{ij}\left(x\right)$ calculado en $\tilde{x}$ y $e_{ij} \overset{def}{=} e_{ij}\left(\tilde{x}\right)$. Sustituyendo en (4.4) en términos del error de la ecuación (4.2) se obtiene:

$$\begin{aligned} Fa\left(\widetilde{x} + \Delta x\right) &= \sum_{(i,j)\epsilon C} F_{ij}\left(\tilde{x} + \Delta x\right) && (4.5) \\ &= \sum c_{ij} + 2\sum b_{ij}^T \Delta x + \Delta x^T \sum H_{ij}\Delta x && (4.6) \end{aligned}$$

y puede resolverse minimizando en Dx resolviendo el sistema lineal:

$$H\Delta x^* = -b \tag{4.7}$$

La matriz H es la matriz de información del sistema, ya que es obtenida proyectando el error de la medición en el espacio de las trayectorias con los Jacobianos. Es una matriz dispersa por construcción, por lo que existen cantidades no cero entre las posiciones conectadas por una restricción, el número de bloques no cero es dos veces el número de las restricciones mas el número de nodos. Esto permite resolver la ecuación (4.7) por factorización dispersa de Cholesky. La solución linealizada entonces se obtiene agregando a la suposición inicial los incrementos calculados.

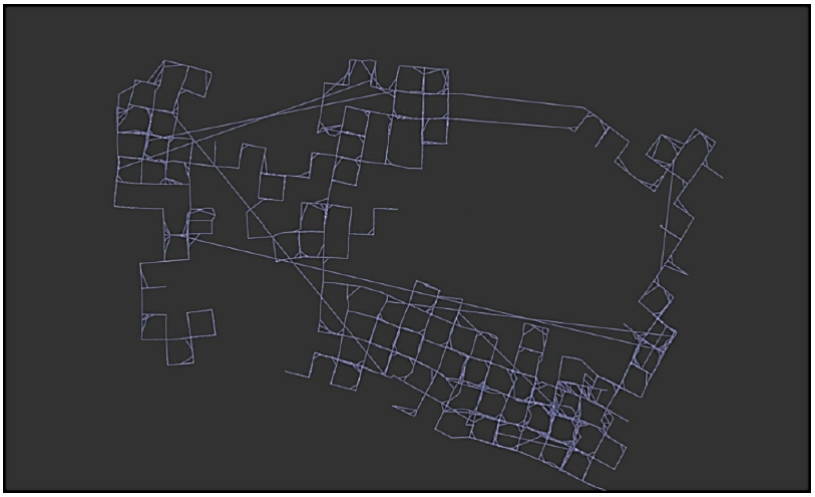
$$x^* = \tilde{x} + \Delta x^* \tag{4.8}$$

El algoritmo Gauss-Newton itera la linealización de la ecuación (4.6), la solución de la ecuación (4.7) y la actualización de (4.8). En cada iteración, la solución previa se utiliza en el punto de la linealización y la suposición inicial.

Utilizando los conceptos anteriores puede concluírse que la optimización del mapa a partir del grafo construido analiza las matrices de información (incertidumbre) y descarta aquellas que no son congruentes con las mediciones hechas en las diferentes posiciones de la cámara. Al descartarlas, la estimación de la posición $k+1$ mejora considerablemente. En la Figura 4.6 puede apreciarse el grafo obtenido por medio de un método de *Front-end* SLAM y a la derecha el mismo mapa optimizado con g^2o. El mapa fue tomado del conjunto de datos "Manhattan" disponible en el sitio oficial del algoritmo, se agregaron diez restricciones falsas de cerrado de bucle además de los errores en asociaciones de datos intrínsecos a la base de datos.

4.5 OPTIMIZACIÓN DE UN GRAFO OBTENIDO CON PTAM

Con el grafo creado con PTAM en la sección 4.3 puede utilizarse la interface gráfica que permite abrir el grafo y visualizar el problema de optimización. Además algunos parámetros de la optimización pueden controlarse tal como el número de iteraciones o el método de optimización (Gauss-Newton / Luenberger) y el optimizador. La interfaz se muestra en la Figura 4.7.



(a) Mapa con inconsistencias.

(b) Mapa optimizado.

Figura 4.6: Optimización del mapa con g^2o.

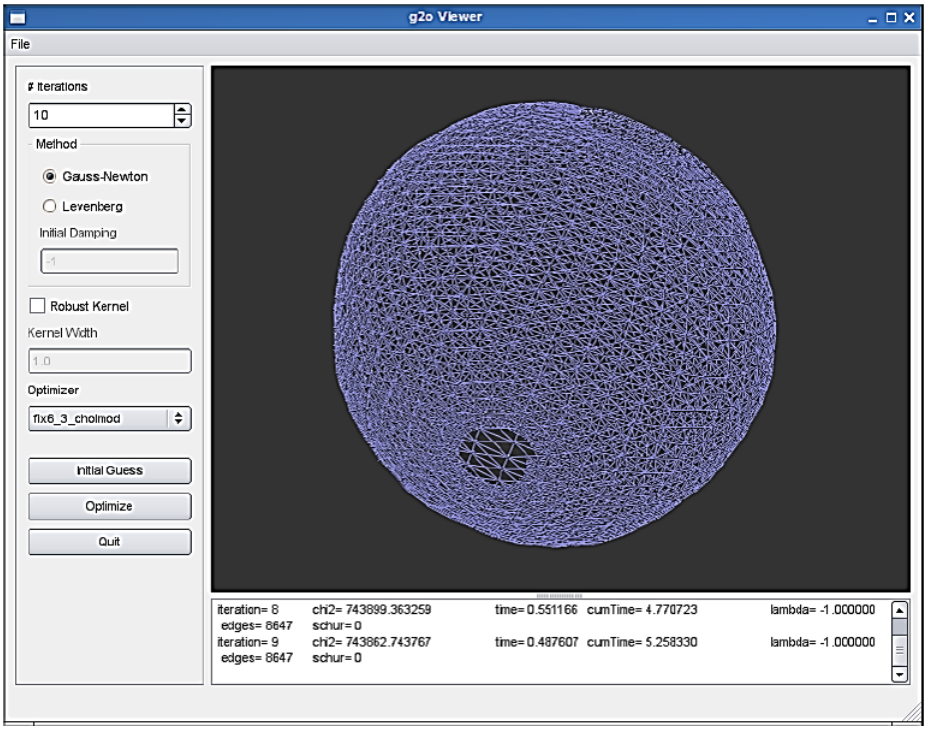
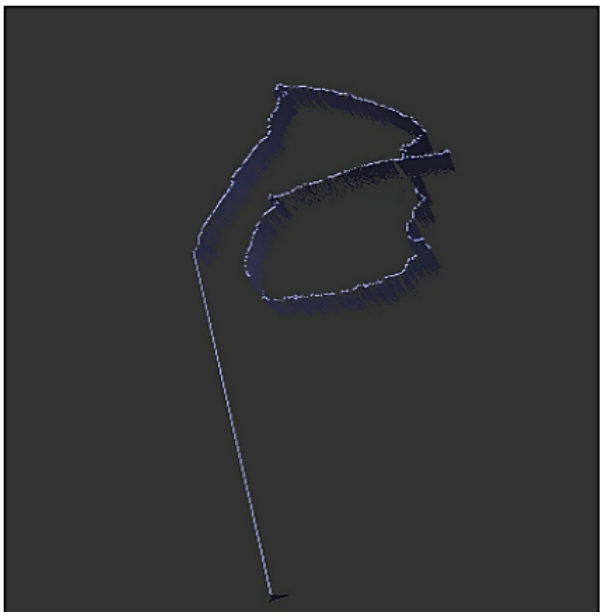


Figura 4.7: Interfaz de g^2o.

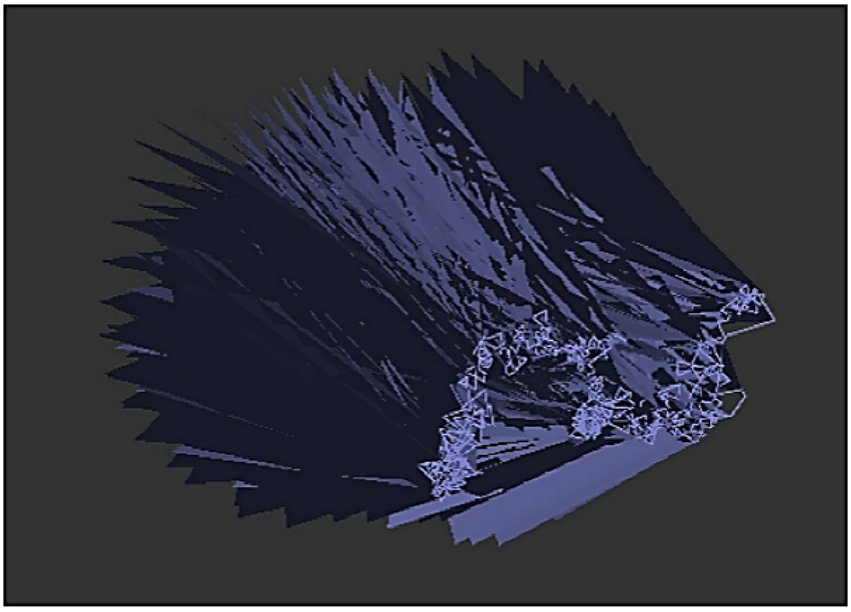
Se obtuvo un grafo con PTAM que describe el mapa mostrado en la Figura 4.8a. Este se encuentra realizado a partir de un área de trabajo muy similar al de la Figura 4.4, un área pequeña para conservar la consistencia del algoritmo y siempre observando la laptop mostrada, es decir, el mapa inicial obtenido mediante el grafo no describe ya orientación y ubicación de la cámara congruente al movimiento desarrollada por el usuario al correr PTAM.

La optimización del mapa mostrado en la Figura 4.8b, muestra una trayectoria congruente con líneas visuales acorde al experimento realizado por lo que la conexión entre PTAM y g^2o demuestra como una relación entre un algoritmo *front-end* y uno *back-end* generan mapas y estimaciones mucho más exactas que los algoritmos de SLAM en línea.

Como conclusión,para la optimización del error de funciones no lineales basadas en gráficos, la implementación actual ofrece soluciones para algunas variantes de SLAM. El objetivo general del problema es encontrar la configuración de parámetros o variables de estado que máximamente explican un conjunto de mediciones afectadas por el ruido de Gauss.



(a)



(b)

Figura 4.8: Optimización de un mapa utilizando PTAM como front-end y g^2o como back-end SLAM.

5 RELOCALIZACIÓN

5.1 INTRODUCCIÓN

En robótica, un problema muy diferente al de localización global (SLAM) es aquel en donde el entorno es previamente conocido para el robot pero este no sabe si en algún momento será o fue trasladado de un lugar a otro, por lo tanto, debe ser capaz de reconocer lugares previamente visitados.

Problema del robot secuestrado

Este problema es conocido como el *problema del robot secuestrado*, dicho problema, por su complejidad, no ha sido completamente resuelto. La dificultad de este problema radica en el hecho de que el robot puede creer saber con certeza dónde se encuentra, cuando en realidad, esa hipótesis es equivocada; mientras que en el problema de localización global el robot tiene la certeza de encontrarse en un ambiente desconocido.

Resolver el problema de relocalización o del robot secuestrado es sumamente importante cuando el objetivo es obtener robots completamente autónomos, pues, aunque a primera instancia no sea intuitivo pensar que un robot pueda ser secuestrado muy a menudo, los algoritmos desarrollados para localización hasta ahora han demostrado no ser lo suficientemente robustos. En efecto, las fallas al seguimiento, perder consistencia en el filtro y por lo tanto la localización del robot, son intrínsecas a todos ellos.

Es por esto que el proceso de relocalización es una tarea primordial, el cual, puede emprenderse una vez que la localización global ha fallado, aproximaciones recientes se han basado en comparar características del entorno previamente escogidas y almacenadas, con aquellas que el robot esta observando. Los algoritmos más utilizados pueden ser agrupados en dos categorías [7]:

Descriptores de imagen

- Basados en *descriptores de imagen*:
  - Estos métodos son utilizados mayormente cuando la cámara disponible es monocular y se encuentra difícil obtener un modelo 3D preciso. Los descriptores como Speeded-Up Robust Features (SURF), Scale-Invariant Feature Transform (SIFT), Features from Accelerated Segment Test (FAST), Binary Robust Independent Elementary Features (BRIEF) u Oriented FAST and Rotated BRIEF (ORB), pueden obtener puntos característicos de la imagen que difícilmente volverán a repetirse (features) basados en información como: textura, intensidad de color, brillo, escala y orientación. Una vez que

el robot ha perdido la localización, dichos puntos pueden obtenerse de la imagen actual y compararse con una base de datos de los features previamente guardados, dicha comparación puede realizarce utilizando el bien conocido método de Fischler y Bolles para relocalizar la posición de la cámara relativa al mapa utilizando tres correspondencias de features y su algoritmo de posición de tres puntos [41]. Una de las mayores desventajas de estos métodos es que dependen de regiones bien texturizadas lo que limita el área de operación. Otros métodos de relocalización basados en descriptores pueden consultarse en [9], [40] y [2].

Key frames

- Basados en modelos tridimensionales del entorno:
  - Estos métodos se utilizan principalmente con imágenes generadas con cámaras que obtienen la profundidad de cada pixel, como cámaras estereoscópicas o cámaras RGBD. La ventaja de utilizar dichas cámaras es la posibilidad de acceder a un modelo tridimensional (mapa denso) del entorno, del cuál, se puede obtener más información, por ejemplo, imágenes sintéticas, además es posible trabajar en toda la imagen. Una clara dificultad al utilizar toda la imagen es el consumo de tiempo de computo, pues a mayor información y operaciones, el poder de cómputo restante para las demás tareas de localización puede no ser suficiente para obtener un código que se ejecute en tiempo real. Es por esto que algunos algoritmos seleccionan fotogramas clave (*Key-frames*) que reducen la base de datos para la búsqueda, la desventaja es que el robot sólo podrá relocalizarse en dichos Key-frames.
    Algunos algoritmos de relocalización que analizan toda la imagen y pueden ser de mucho interés son [25] y [24].

5.2 RELOCALIZACIÓN POR REGRESIÓN GENERAL.

El método del cual se hablará en esta sección encuentra su origen el el articulo de Andrew P. Gee et al. [12], el cual utiliza imágenes sintéticas, cámara RGBD y un algoritmo de regresión general para comparar la imagen actual con el mapa previamente almacenado. Una de las ventajas de este método es su rapidez, por lo que es posible crear una base de datos de las imágenes obtenidas a la velocidad de muestreo de la cámara.

La metodología descrita en el artículo es la siguiente:

1. Generar l imágenes sintéticas a partir de la vista actual de la cámara, lo que en total resultará en $m = l + 1$ imágenes.

2. Reducir la resolución de las imágenes; 80x60 y 20x15 pueden utilizarse con resultados favorables.

3. Imágenes en escala de grises son suficientes para resolver el problema sin perder mucha información, es además es recomendable normalizar dicha intensidad de acuerdo a la siguiente formula:

$$c_{ij} = \frac{c_{ij} - \overline{c}_i}{\sigma_{ci}}$$

donde $\overline{c}_i$ es la media de la intensidad y σ_{ci} es la desviación estándar de la $i - esima$ imagen.

4. Construir $n * i$ vectores $I_j = [u_j, v_j, \rho_j, c_j]$, en donde: n es el número de pixeles de la imagen, (u_{ij}, v_{ij}) son las coordenadas del pixel, r_{ij} es la profundidad del pixel y c_{ij} es la intensidad del pixel. Finalmente cada imagen contará con una matriz I_i formada por $n$vectores I_j.

5. Junto con la matriz I_i se debe almacenar la posición $x_i = [t, ln\,(q)]$ de la cámara en el espacio, donde t es el vector de traslación y q es el cuaternión que describe la orientación de la posición.

6. Para estimar la posición $\widetilde{x}$ de la relocalización se utiliza un estimador Nadaraya-Watson [31] con un kernel Gausiano como se muestra a continuación:

$$\widetilde{x} = \frac{\sum_{i=1}^{m} x_i d_i}{\sum_{i=1}^{m} d_i} \tag{5.1}$$

donde:

$$d_i = exp\left(-\frac{1}{n\alpha}\sum_{j=1}^{n}\left(\frac{\left(c_{0j} - c_{ij}\right)^2}{\sigma_{cj}^2} + \frac{\left(\rho_{0j} - \rho_{ij}\right)^2}{\sigma_{\rho j}^2}\right)\right)$$

σ_{cj}y $\sigma_{\rho j}$ son las desviaciones estándar de la intensidad y de la profundidad respectivamente, de todos los pixeles$j - ésimos$ almacenados en la base de datos y α es un parámetro de ajuste.

Desde un punto de vista analítico, la ecuación (5.1) es una suma ponderada en donde la posición x_i contribuye a dicha suma en función del parámetro d_i, este parámetro se acercará a cero en caso de que las matrices I_i sean totalmente diferentes y a la unidad en caso de que sean iguales.

Una de las desventajas de este método es que la precisión de la estimación $\widetilde{x}$ no es buena y la posición de la cámara no participa dentro del cálculo del parámetro d_i.

Para poder utilizar el algoritmo anterior en un robot Darwin-Op, es necesario modificarlo considerando que se cuenta sólo con una cámara monocular y por ende, no existe un modelo tridimensional del entorno, por lo que no es posible generar vistas sintéticas de la imagen original. Una primera aproximación puede conseguirse modificando el parámetro d_i de la ecuación (5.1) de la siguiente manera:

$$d_i = exp\left(-\frac{1}{n\alpha}\sum_{j=1}^{n}\left(c_{0j}-c_{ij}\right)^2\right)$$

En imágenes muy similares la división $\left(c_{0j}-c_{ij}\right)^2/\sigma_{cj}^2$ puede generar grandes cifras, por lo que una imagen similar puede resultar matemáticamente en una imagen que no colabore con la suma ponderada al cálculo de la estimación de la posición, por lo que la diferencia entre intensidades, en la práctica resulto entregar resultados más favorables.

El algoritmo fue programado utilizando como front-end SLAM la plataforma PTAM [20], el método de relocalización de PTAM fue inhabilitado para observar el desempeño de la relocalización una vez que la localización falla.

Debido a los recursos limitados de computo, disponibles en el robot humanoide, PTAM o algún otro algoritmo de VSLAM en el procesador, comprometería el funcionamiento general del robot, costo computacional de las actividades básicas de Darwin-Op sólo permite capturar 5 fotogramas por segundo, por lo que es de esperarse que el muestreo baje con un algoritmo de localización programado. Es por esto que se ha propuesto no utilizar VSLAM dentro de la computadora del humanoide, en su lugar, y teniendo en cuenta que el método de relocalización no necesita mas que la posición del robot y una imagen para generar un mapa, se utilizará la odometría del robot y un mapa previamente almacenado "a mano" del área de interés en donde el robot realiza su tarea.

Debido a que los recursos computacionales disponibles en el robot humanoide son muy limitados, la implementación de PTAM, o algún otro algoritmo de VSLAM, comprometería el funcionamiento general del robot.

6

IMPLEMENTACIÓN DEL SISTEMA DE AUTOLOCALIZACIÓN.

6.1 INTRODUCCIÓN

La autolocalización tiene un papel importante en las tareas de un robot móvil, en los últimos años competiciones como "DARPA robot challenge" han mostrado la importancia de conocer la posición del robot y la estructura del entorno a fin de interactuar con él [27]. Los torneos de RoboCup necesitan dos aspectos de la autolocalización: local y global; en los juegos de fútbol es necesario estimar la distancia a un objeto especial respecto a un marco coordenado global y fijo. Conocer la ubicación de los objetos más importantes en el campo tales como la pelota, robots oponentes, portería y compañeros de equipo, pudiera conducir al mejoramiento de algoritmos de estrategia o toma de decisiones.

De la sección 1.4 es claro que el costo computacional es el principal problema si se utiliza un algoritmo de autolocalización en un dispositivo móvil de bajas prestaciones como un cuadrorotor o un robot humanoide para las competiciones de RoboCup que es el principal interés del equipo dotMEX del Departamento de Control del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV).

Impulsados con la idea de las competiciones de RoboCup donde el mapa es previamente conocido pero extremadamente dinámico, y el procesador del robot debe resolver la cinemática y dinámica del robot, toma de decisiones entre otras tareas, se propone un método de relocalización basado en regresión general (Ver sección 5.1) y odometría con el objetivo de mantener el costo computacional constante.

6.2 MAPA

Usualmente, para resolver el problema de Relocalización se crean mapas del entorno en forma de bases de datos en las que se guardan los mapas de dicho entorno, creados mediante algún algoritmo de SLAM y cámaras RGBD [37], que permitan la futura relocalización del robot después de, por ejemplo, una caída o de haberse ausentado de dicho entorno.

Específicamente en A. Gee y W. Mayol [12] es utilizado para proveer una gran cantidad de información al algoritmo, con el afán de obtener un buen resultado.

Figura 6.1: Vista superior para seguimiento de trayectoria.

A pesar de las restricciones para utilizar sensores activos en las competencias de RoboCup, crear un banco de datos sin un algoritmo de SLAM es posible para aplicaciones en ambientes cerrados.

Debido a las restricciones en las competiciones RoboCup, en donde se prohíbe el uso de sensores activos, no es posible utilizar mediciones de profundidad, la cual sólo se puede obtener con sensores activos. Más aún, la disponibilidad de potencia de cómputo reducida en los robots humanoides de competición no permite el uso de técnicas de VSLAM para crear los mapas SLAM para relocalización. Sin embargo, proponemos resolver el problema creando una base de datos con imágenes capturadas por el humanoide desde un conjunto pequeño de posiciones conocidas elegidas juiciosamente para cubrir la mayor cantidad posible del espacio de trabajo. Para ello es necesario conocer con precisión las posiciones desde las que se capturan las imágenes guardadas, lo cual se hace mediante una cámara suspendida del techo que proporciona una vista superior del entorno, incluido el robot, a partir de la cual se calcula la posición (x, y, θ) desde la que el robot captura tales imágenes, tal como se muestra en la figura 6.1. En cada una de las posiciones elegidas el robot captura 10 imágenes en las direcciones Este, Sur, Oeste, Norte, Noreste, Noroeste, Sureste y Suroeste.

La información de las imágenes y odometría obtenida del robot Darwin son transmitidas vía TCP/IP a una computadora central que es la responsable de recolectar la información. Se eligen pocas imágenes pero de alto contenido estratégico para crear el mapa y generar key-frames, manteniendo así un costo computacional compatible con las prestaciones del sistema.

Además, los experimentos han mostrado que no todos los pixeles son útiles debido a que la posición de relocalización del robot puede establecerse en una posición cinemática fija del humanoide, por lo tanto, se utilizan pequeñas imágenes para generar los bancos de da-

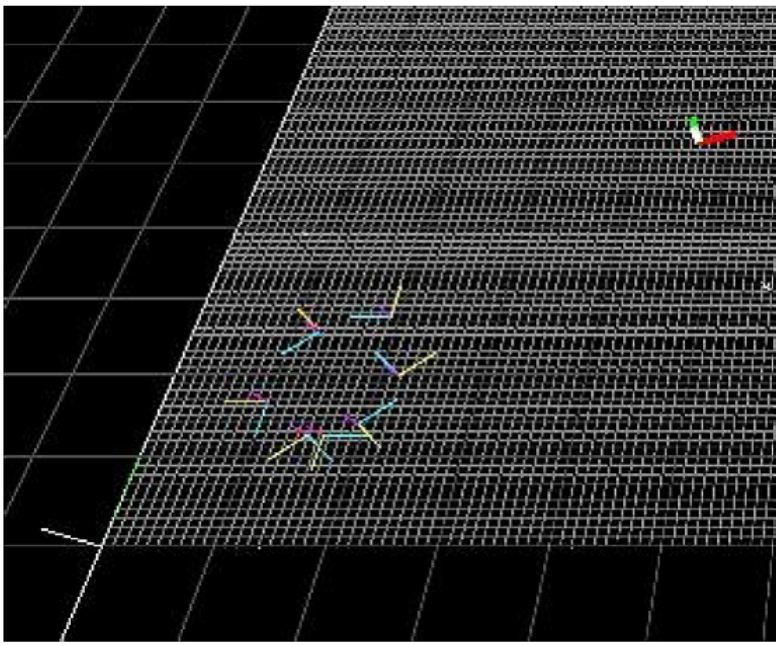


Figura 6.2: Interfaz gráfica para la generación del mapa.

tos, en este trabajo imágenes de 6x40 pixeles de resolución mostraron tener buenos resultados. La Figura 6.2 muestra posiciones de cámara para un mapa pequeño creado con la metodología descrita. La coordenada en lineas gruesas es la vista actual del robot, los demás describen las posiciones del mapa de imágenes.

Antes de guardar las imágenes del mapa en la base de datos se escalan y se normalizan.

La relocalización se realiza tan solo en ciertas posiciones establecidas mediante una solicitud especial al Módulo de Toma de Decisiones a fin de optimizar los recursos del CPU. No se consideró hacer la relocalización de manera continua para no distraer al Módulo de Toma de Decisiones quien está encargado de muchas tareas sustantivas y no es conveniente exigirle una tarea constante de análisis de imágenes correspondiente a la tarea de relocalización.

El sistema visual de obtención de la posición actual del robot también se utiliza para capturar la trayectoria real que sigue el humanoide DarwinOP durante su recorrido y así poder compararla contra la posición estimada por nuestro sistema VSLAM y calcular de manera objetiva el error correspondiente.

6.3 REGRESIÓN GENERAL

A diferencia de la sección 5.1 la regresión general utilizada en el algoritmo tiene una diferencia en el kernel Gausiano utilizado tal que:

$$\widetilde{x} = \frac{\sum_{i=1}^{m} x_i d_i O_j}{\sum_{i=1}^{m} d_i O_j}$$

donde:

$$d_i = exp\left(-k \sum_{j=1}^{n} \left(c_{0j} - c_{ij}\right)^2\right) \tag{6.1}$$

$$O_j = exp\left(-k\left(\sum_{j=1}^{n} \left(\left\|\bar{x}_j\right\| - \left\|x_i\right\|\right)^2 + \left(\bar{q}_j - q_i\right)^2\right)\right) \tag{6.2}$$

donde c_0 es la intensidad del pixel de la imagen actual, $\bar{x}$ y $\bar{q}$ son la estimación de posición y la estimación del $ln(q)$ de la odometría respectivamente. Por lo que la estimación $\widetilde{x}$ es una suma ponderada, donde la contribución de cada imagen muestra se determina por la distancia euclidiana entre la imagen muestra y la vista actual de la cámara, la posición de la odometría y la posición de cada imagen en el mapa.

6.4 ODOMETRÍA

Se eligió utilizar cinemática directa para obtener la posición estimada del robot, empleando la posición de las articulaciones en todo momento. Se utilizó la configuración de la cinemática del humanoide DarwinOP de la sección 2.2.2 para generar la matriz Denavit-Hartenberg siguiente:

$$D = \begin{bmatrix} i & \theta & d & a & \alpha \\ 1 & 0 & -e & 0 & 0 \\ 2 & \frac{\pi}{2} + q_1 & 0 & 0 & \frac{\pi}{2} \\ 3 & \frac{\pi}{2} + q_1 & d & 0 & \frac{\pi}{2} \\ 4 & q_3 & 0 & -c & 0 \\ 5 & q_4 & 0 & -b & 0 \\ 6 & q_5 & 0 & 0 & -\frac{\pi}{2} \\ 7 & q_6 & 0 & -a & 0 \end{bmatrix}$$

A partir de la solución de la cinemática, se genera un vector 6D con la información de la ubicación del robot. El desplazamiento se calcula continuamente leyendo el vector de posición. La distancia en la dirección de x se calcula a través de la amplitud del paso, la Figura 22 muestra la posición en la dirección x del pie izquierdo y derecho relativo al plano coordenado del robot en un caminado recto.

La distancia recorrida se calcula de la siguiente manera:

C:/Users/Gateway/Google Drive/Tesis/Imagenes_tesis/Capitulo 6/22.tif

Figura 6.3: Posición en x del pie izquierdo y derecho.

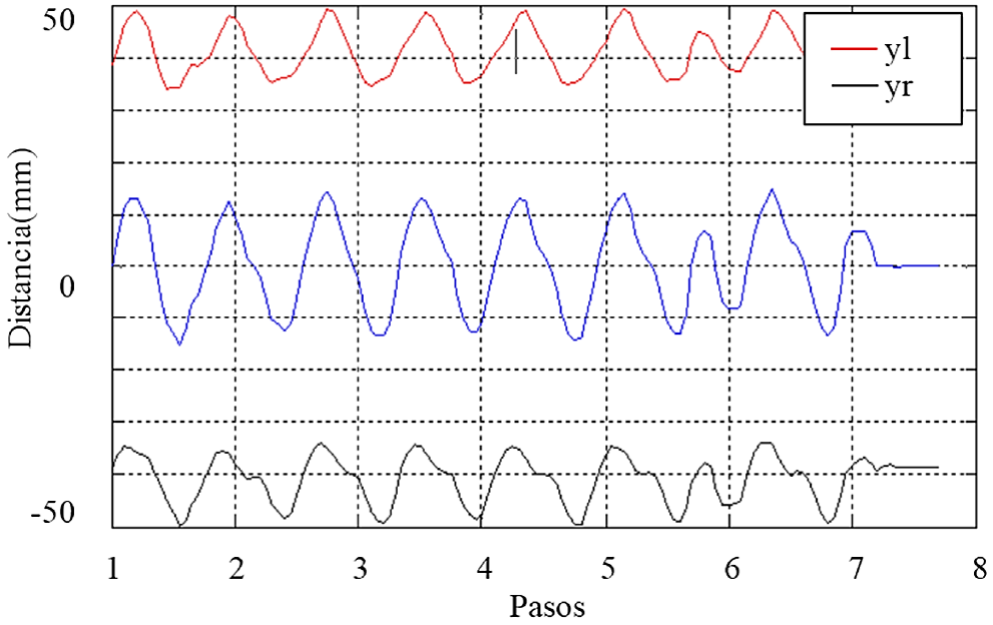


Figura 6.4: Posición en y del pie izquierdo y derecho.

$$X_R = abs\,(x_l) - abs\,(x_r)$$

De la misma manera el desplazamiento de y se obtiene con un análisis similar de la Figura 6.4:

$$y_R = y_1 + y_r$$

Finalmente, el ángulo de *yaw* se obtuvo sumando los giros de las articulaciones de las caderas debido a que la cadena cinemática es simétrica al girar. La Figura 6.5 muestra una prueba en un circuito cuadrado de 90 x 90 cm donde pueden apreciarse la trayectoria real y estimada.

Debido al deslizamiento que existe entre los pies del robot y el piso, las trayectorias de odometría nunca se repiten en el mismo experimento.

6.5 EXPERIMENTOS Y RESULTADOS

El desempeño del método de autolocalización propuesto se probó con diferente número de imágenes en la base de datos a fin de mostrar y

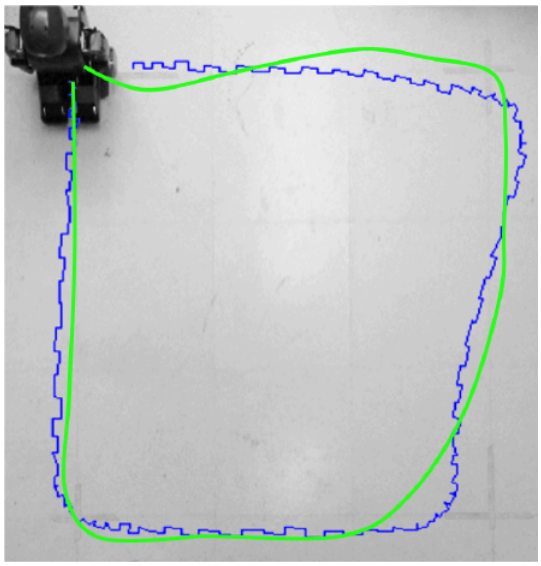


Figura 6.5: Odometría obtenida del robot humanoide.

analizar los resultados en dos trayectorias diferentes. El robot Darwin se utilizó para comparar los resultados con la trayectoria real.

En el primer experimento el objetivo fue probar la efectividad del algoritmo sin la contribución de la odometría. Se utilizó un mapa con 20 imágenes obtenidas de la grabación en trayectoria diagonal del robot en posiciones fijas.

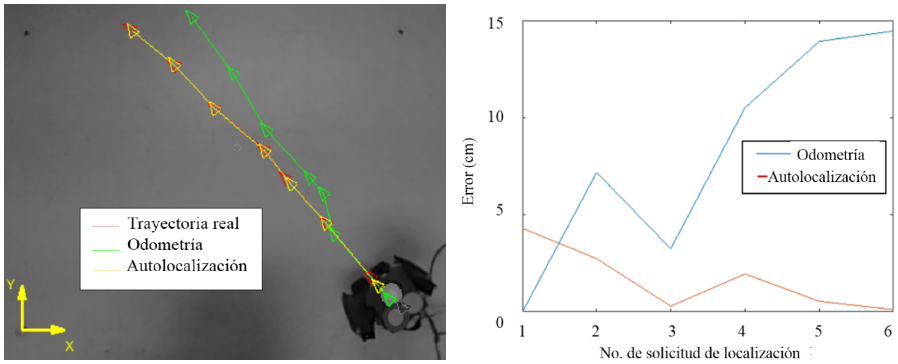
Primer experimento

Primeramente el algoritmo de autolocalización se utilizó para estimar la posición de seis imágenes que ya conformaban el mapa, con el objetivo de verificar la efectividad del método pero sin utilizar la información de la posición de la imagen. La Figura 6.6a muestra el resultado, la trayectoria en amarillo (trayectoria de la autolocalización) es muy cercana a la trayectoria real (trayectoria roja) y es claro que no es conveniente agregar la información de la odometría debido a que podría perjudicar la localización. El error entre la trayectoria real y la estimada puede observarse en la Figura 6.6b. El error fue calculado como la distancia euclidiana entre la trayectoria autolocalizada y la trayectoria real.

Segundo experimento

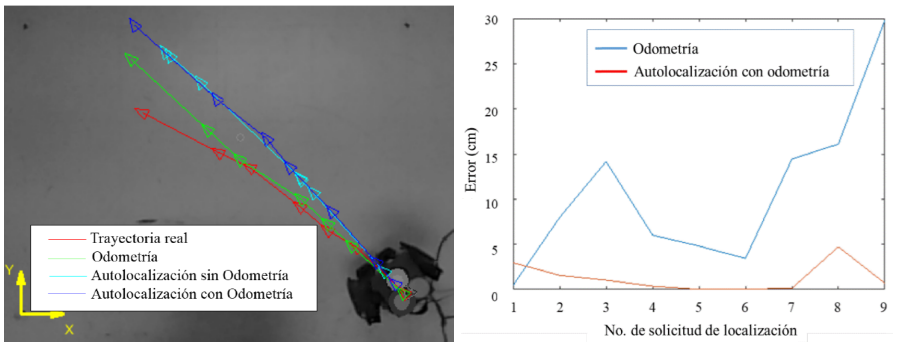
El segundo experimento consiste en probar una trayectoria diferente y estimar la posición del robot utilizando el mapa almacenado en el experimento uno pero con imágenes obtenidas por el robot en la nueva trayectoria. El robot caminó en la misma dirección del experimento uno pero con diferente trayectoria.

Agregar información de la odometría en este caso (con un experimento más real), mejora el resultado agregando consistencia en la posición en la que se solicitó una localización. Sin embargo, la orientación continúa con un incremento en la orientación debido a la pobre cantidad de imágenes que conforman el mapa, el robot debe asociar nuevas imágenes entrantes con el mundo conocido por el robot (que es una trayectoria en línea recta). Los errores respecto a la nueva tra-



(a) Trayectoria. (b) Error.

Figura 6.6: Resultados experimento No. 1.



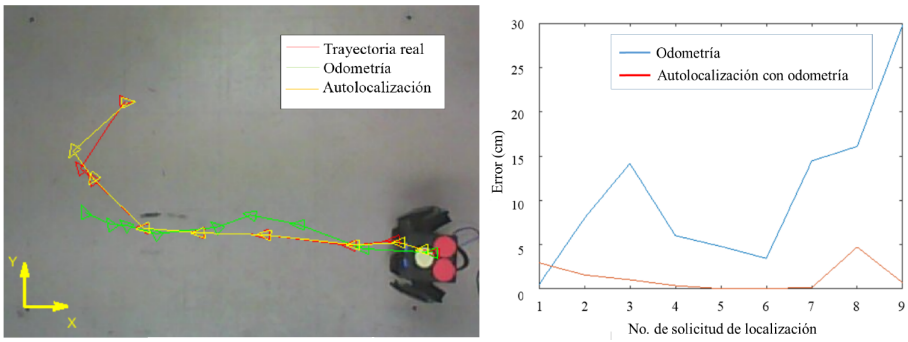
(a) Trayectoria (b) Error.

Figura 6.7: Experimento No. 2.

yectoria se observan en la Figura 6.7, puede observarse que la relocalización con odometría mejora las localizaciones y la posición.

Tercer experimento

Finalmente se grabó un mapa con 50 imágenes tomando en cuenta diferentes trayectorias y posiciones a fin de probar el método con el robot recorriendo una trayectoria con un giro de 180°. Debido a que la odometría pierde precisión con las vueltas, el factor k en (6.1) y (6.2) se utiliza para incrementar o decrementar la preponderancia de las imágenes con respecto a la información de la odometría en ese momento, también se utilizan para eliminar por completo la aportación de la odometría cuando el robot sufre una caída o pérdida de orientación. La Figura 6.8 muestra la trayectoria y el error a lo largo de las nueve solicitudes de autolocalización que hizo el robot.



(a) Trayectoria (b) Error.

Figura 6.8: Resultados experimento No. 3.

7

CONCLUSIÓN Y TRABAJO FUTURO.

7.1 OPTIMIZACIÓN DE GRAFOS.

En el capítulo 4 se presentó la plataforma que se utilizará en el Departamento de Control Automático para el análisis del problema de SLAM utilizando el panorama de optimización de grafos. Aunque PTAM no es un programa convencional de SLAM se eligió como plataforma *front-end* debido a que el objetivo de esta línea de investigación es experimentar con algoritmos de optimización robustos a ambientes dinámicos proponiendo un método de optimización como nuevo *back-end* SLAM. El propósito de esta tesis fue enlazar PTAM y g^2o y corroborar que el primero puede utilizarse como plataforma de experimentación orientada a un análisis de grafos. Los resultados fueron satisfactorios. La trayectoria obtenida por PTAM se optimizó satisfactoriamente en g^2o obteniendo un mapa de posiciones congruente a pesar de que se experimentó con oclusiones incluidas en el trayecto.

Los algoritmos *back-end* son una buena área de oportunidad para los algoritmos de optimización y control robusto desarrollados en el departamento del CINVESTAV puesto que el análisis puede desarrollarse fuera de línea tanto para mapas en 2D como para mapas en 3D utilizando seis grados para la ubicación del robot.

7.1.1 *Trabajo futuro*

La posibilidad de establecer un problema de SLAM como un problema de grafos puede utilizarse para mejorar la asociación de datos. Partiendo del hecho de que la asociación de datos se realiza en base a un análisis de incertidumbre, se propone utilizar el programa de optimización para elegir cuál será la mejor asociación a partir de las mediciones en cada paso del robot y las incertidumbres que existen entre la medición de cada feature. El grafo que deberá buscarse entonces será el que se muestra en la Figura 7.1.

A partir de aquí, la nueva asociación de datos podrá utilizarse para mejorar la trayectoria del robot y la elaboración del mapa. El principal objetivo de esta propuesta es poder establecer una hipotesis H robusta a ambientes dinámicos y perturbaciones que puedan acercar a la solución del problema de SLAM, por lo que utilizar un método robusto como AEM podría ser una alternativa viable para un *back-end* al poder realizar la optimización fuera de línea.

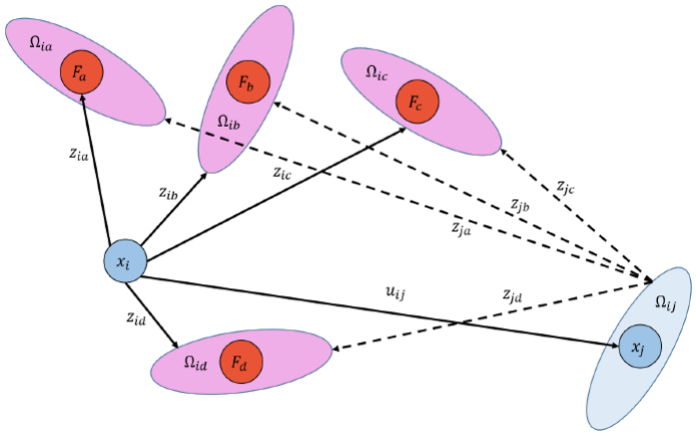


Figura 7.1: Grafo propuesto para la nueva asociación de datos.

7.2 AUTOLOCALIZACIÓN PARA ROBOT HUMANOIDE DARWINOP.

En el Capítulo 6 se presentó la implementación de un método de autolocalización basado en relocalización e información de odometría fue presentado. Se demostró que un algoritmo de costo computacional constante puede utilizarse en sistemas embarcados para entornos controlados. El error del método de regresión general fue reducido fortaleciendo el método con información de odometría.

Los experimentos mostraron que entre mayor sea la base de datos de imágenes del mapa del robot, mayor será la precisión de la localización obtenida mediante el algoritmo en el área.

La autolocalización en el robot humanoide no interfirió en la tareas sustantivas que el robot necesita desarrollar, permitiendo así incluir otros algoritmos que mejoren el desempeño de los jugadores en el partido.

Una de las desventajas del método presentado es que las ubicaciones en donde el robot solicita su ubicación son muy delicadas, al utilizar solo un fragmento de la imagen y debido a que no es posible generar imágenes sintéticas que formen parte del banco de datos del mapa, las condiciones del terreno deben ser ideales para la autolocalización, es decir, no debe haber desniveles en el piso que puedan provocar la solicitud de localización con una imagen desorientada.

La sintonización de las ganancias para el algoritmo de relocalización debe hacerse adaptable al sitio en donde el algoritmo se utiliza, el deslizamiento sigue jugando un papel importante en la exactitud de la autolocalización por lo que la aportación de la odometría debe manejarse con mesura y análisis previo al ajuste para su uso.

7.2.1 *Trabajo futuro*

Es necesario un análisis del costo computacional contra el tamaño del mapa, uno más de la precisión de la posición estimada y el área de cobertura de la autolocalización, a fin de que se establezca una cuota de los recursos computacionales que este algoritmo requiere para una aplicación en especifico.

Debido a que no es posible utilizar sensores activos para competencias ,es recomendable sustituir la cámara del robot Darwin por una cámara estéreo y poder medir la profundidad de algunos puntos para extrapolarlos y lograr una aproximación a imágenes RGBD, generar vistas sintéticas a partir de estas y mejorar la estimación de la ubicación.

Asimismo, el método presentado, puede ser utilizado para comenzar a estudiar un algoritmo de relocalización que sea robusto a perturbaciones, las oclusiones en las imágenes que adquiere el robot durante el juego serán muy probablemente diferentes a las vistas utilizadas para realizar el mapa, el robot debe ser capaz de saber si la información obtenida en el intento actual de localizarse es congruente con su historial aunque el robot haya caído o se haya desubicado.

BIBLIOGRAFÍA

[1] Hussein Alazky A. Pozniak J.M. Ibarra Zannatha E. Hernández. Attractive ellipsoid method controller under noised measurements for slam. *Submited to: International Journal of Systems Science.*

[2] Majdik Andras. New approach in solving the kidnapped robot problem. *Robotics (ISR), 41st International Symposium on y 2010 6th German Conference on Robotics (ROBOTIK)*, 2010.

[3] Eric Hernández Castillo. Slam visual no lineal para el humanoide robonova. Master's thesis, Control Automático CINVESTAV, 2013.

[4] W . Burgard D. Fox D. Hähnel and S. Thrun. An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. *In in Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS) Las Vegas, NV pp. 206211*, 2003.

[5] Y. Liu D. Koller-A. Y. Ng Z. Ghahramani S. Thrun and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *Int. J. Robot. Res., vol. 23, no. 7/8, pp. 693716*, 2004.

[6] S. Thrun D. Koller M. Montemerlo and B.Wegbreit. Fast slam: A factored solution to simultaneous localization and mapping. *In in Proc. Nat. Conf. Artificial Intelligence (AAAI) Edmonton, Canada, pp. 593598*, 2002.

[7] Burgard W. y Fox D. Thrun S. Probabilistic robotics. intelligent robotics and autonomous agents. *MIT Press*, 2005.

[8] Andrew J. Davison. Real-time simultaneous localisation and mapping with a single camera. *ICCV '03 Proceedings of the Ninth IEEE International Conference on Computer*, 2003.

[9] Georg Klein e Ian Reid Brian Williams. Real-time slam relocalisation. *IEEE International Conference on Computer Vision*, 2007.

[10] Ethan Eade and Tom Drummond. Scalable monocular slam. 2006.

[11] H. Singh R. Eustice and J. J. Leonard. Exactly sparse delayedstate filters. *In Proc. IEEE Int. Conf. Robotics and Automation (ICRA), Barcelona, Spain, pp. 24282435*, 2005.

[12] Andrew Gee and Walterio Mayol-cuevas. 6d relocalisation for rgbd cameras using synthetic view regression. *Proceedings of the British Machine Vision Conference (BMVC).*, 2012.

[13] Cyrill Stachniss Wolfram Burgard. Giorgio Grisetti, Rainer Kummerle. A tutorial on graph-based slam. *Department of Computer Science, University of Freiburg*.

[14] C. Stachniss G. Grisetti and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Trans. Robot vol., 23, no. 1, pp. 3446*, 2007.

[15] Kin Leong Ho and Kin Leong Ho Paul Newman. Loop closure detection in slam by combining visual and spatial appearance. *Robotics and Autonomous Systems*, 2006.

[16] J. Bowman J. D. Chen P. Mihelich M. Calonder V. Lepetit K. Konolige and P. Fua. View-based maps. *Int. J. Robot. Res., vol. 29, no. 10.*, 2010.

[17] J. M. M. Montiel J. Neira J. A. Castellanos and J. D. Tardós. The spmap: A probabilistic framework for simultaneous localization and map building. *IEEE Trans. Robot. Automat., vol. 15, no. 5, pp. 948953.*, 1999.

[18] A. H. Jazwinski. Stochastic processes and filtering theory. *Academic Press, New York*, 1970.

[19] S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE, 92(3):401422*, 2004.

[20] David Murray Georg Klein. Parallel tracking and mapping for small ar workspaces. *International Symposium on Mixed and Augmented Reality (ISMAR, Nara)*, 2007.

[21] K. Konolige. A gradient method for realtime robot control. *In Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS).*, 2000.

[22] F. Del laert and M. Kaess. Square root sam: Simultaneous location and mapping via square root information smoothing. *Int. J. Robot. Res.*, 2006.

[23] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonom. Robots, vol. 4, pp. 333349*, 1997.

[24] Calway A. Mayol-Cuevas W. Martinez-Carranza J. Enhancing 6d visual relocalisation with depth cameras. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[25] Calway A Mayol-Cuevas W. Martinez-Carranza J. Slam++: Simultaneous localisation and mapping at the level of objects. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[26] Sharon B. McGrayne. The theory that would not die: How bayes rule cracked the enigma code, hunted down russian submarines, and emerged triumphant from two centuries of controversy. *Yale University Press*, 2011.

[27] Jean-Philippe Tardif Michael George and Alonzo Kelly. Visual and inertial odometry for a disaster recovery humanoid. *Field and Service Robotics, volume 105 of Springer Tracts in Advanced Robotics*.

[28] Juan D. Tardós. José Neira. Data association in stochastic mapping using the joint compatibility test. *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, 2001.

[29] L. Clemente A. Davison I. Reid J. Neira and J. Tardós. Mapping large loops with a single handheld camera. 2007.

[30] J. Leonard E. Olson and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. *in Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2006.

[31] G.S. Watson. Smooth regression analysis. Smooth regression analysis. *Indian Journal of Statistics Series A, 26(4):359372*, 1964.

[32] B. Williams M. Cummins J. Neira P. Newman I. Reid and J. Tardós. A comparison of loop closing techniques in monocular slam. *Robotics and Autonomous Systems.*, 2009.

[33] M. Self R. Smith and P. Cheeseman. Estimating uncertain spatial realtion ships in robotics. *Autonomous Robot Vehicles, I. Cox and G. Wilfong, Eds. Berlin: Springer-Verlag, pp. 167193*, 1990.

[34] Randall C. Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainly. *Int. J. Rob. Res.,*, 1986.

[35] Lina M. Paz J.D. Tardos and J. Neira. Divide and conquer ekf slam in o(n). *Robotics IEEE Transactions on 24(5): 1107-1120*, 2008.

[36] P. Pfaff R. Triebel and W. Burgard. Multilevel surface maps for outdoor terrain mapping and loop closing. *In in Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2006.

[37] A.P. Gee W. Mayol-Cuevas D. Damen and A. Calway. Egocentric real-time workspace monitoring using an rgb-d camera. *In Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[38] E. Wan and R. van der Merwe. The unscented kalman filter. *Wiley Publishing*, 2001.

[39] Eric A. Wan and Rudolph van der Merw. The unscented kalman filter for nonlinear estimation. *Oregon Graduate Institute of Science & Technology*.

[40] W. Mayol-Cuevas y A.Calway D.Chekhlov. Appareance based indexing for relocalisation in real-time visual slam. *In Proc. British MAchine Vision Conf. (BMVC)*, 2008.

[41] M.A. Fischler y R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comunications of the ACM, 24(6): 381395*, 1981.