# CONSTRUCCIÓN DE MAPAS GEOMÉTRICOS BIDIMENSIONALES DE AMBIENTES INTERIORES CON UN TELÉMETRO LÁSER

**TESIS**

Que para obtener el grado de

**MAESTRÍA EN CIENCIAS EN INGENIERÍA ELÉCTRICA**

presenta

**Carlos Alberto Lara Álvarez**

**Leonardo Romero Muñoz**

**Director de Tesis**

Universidad Michoacana de San Nicolás de Hidalgo

Junio 2006

A *Arlet e Inés*, por su amor que me ha hecho superar los momentos difíciles.

A *Raquel y Carlos*, por el espíritu de lucha que me heredaron. Gracias Carlos, estés donde estés, por los momentos felices que compartimos.

A *Hugo, Ramses y Luis Gerardo*, gracias hermanos, por su ejemplar entusiasmo para seguir adelante.

A *Leonardo, Félix, Edgar Leonel, Fernando A. y José Antonio*, gracias maestros, por sus invaluables consejos.

A *Himer*, por su amistad incondicional.

A todos, que Dios los bendiga.

# Resumen

En los últimos años, gracias a los avances de la ciencia y la tecnología, se han diseñado dispositivos eléctricos y electrónicos pequeños cada día más eficientes. Ésto ha permitido la realización de investigación en otros rubros de la ciencia, la robótica no ha sido la excepción. Y tal vez, ésta área sea una de las más favorecidas gracias a la bondad que guarda con la electrónica. Los robots se definen como dispositivos mecánicos cuyo objetivo es facilitar el trabajo del hombre desde cualquier punto de vista.

Un área de la robótica conocida como robótica móvil tiene como objetivo la construcción de robots que sean capaces de navegar en entornos desconocidos sin necesidad de supervisión. Los robots móviles son un elemento primordial y esencial en la toma de decisiones donde la practicidad humana corre riesgos, son un respaldo importante para personas incapacitadas y son además de gran ayuda como guías de turismo.

Para navegar en su entorno, los robots móviles constan de sensores, procesadores y actuadores. El telémetro láser es un sensor muy popular en robótica móvil debido a que es preciso, rápido y direccional. Para actividades de alto nivel el robot requiere una descripción del ambiente (un mapa). El conjunto de mediciones de un telémetro láser se puede usar para generar mapas geométricos bidimensionales. En ambientes interiores, donde existe una gran cantidad de objetos con contornos lineales, es muy común el uso de mapas geométricos de líneas (MGL).

Existen varios métodos para encontrar un MGL a partir de las mediciones de un telémetro láser. Esta tesis analiza los métodos reportados en la literatura para extraer múltiples líneas a partir de un conjunto de puntos. Un MGL debe de ser muy preciso pues de él depende el buen desempeño del robot. En muchas ocasiones, obtener un MGL de un ambiente no controlado en un tiempo reducido constituye un reto. Por ejemplo, en muchos ambientes interiores además de los objetos con contornos lineales, existen objetos pequeños (patas de silla, mesas, etc.) que dificultan la extracción de MGL.

Esta tesis propone dos métodos para la obtención de MGL locales a partir de un

conjunto de mediciones del telémetro láser tomadas desde una posición estática: *el algoritmo de consenso de muestra en una ventana* (WSAC) y el *algoritmo de consenso de muestra en una ventana con evaluación global* (WSAC–GE). La idea básica de estos algoritmos es buscar líneas dentro de un subconjunto de mediciones consecutivas usando un método de selección aleatoria. Como lo demuestran las pruebas experimentales presentadas, los MGL extraídos por estos algoritmos son más precisos y son más asertivos que los obtenidos con otros métodos comúnmente usados.

Por último, se aborda un algoritmo simple de empate entre dos conjuntos de lecturas tomadas de posiciones adyacentes apoyado en los MGL. El objetivo de este algoritmo es generar un mapa global y conocer la posición del robot en cualquier momento.

# Abstract

In the last years, due to the advances in science and technology, electronic devices have been designed more efficiently on a day to day basis. It has allowed the realization of research in other science fields, Robotics has not been an exception. Perhaps, Robotics is one of the most favored fields due to its relationship with electronics. Robots are defined as mechanic devices in order to facilitate man's work at any point.

To navigate in its environment, the mobile robots have sensors, processors and actuators. The laser is a very used sensor due to its precision, fastness and directionality. To perform high level activities a robot requires a description of the environment (a map). The set of laser measurements can be used to generate bidimensional geometric maps. In indoor environments, where there are many planar objects, the use of Geometric Line Maps (GLMs) is very common.

There are many methods to find GLMs using laser scans. This thesis analyzes methods reported in science literature to extract multiple lines from a set of points. The GLMs must be exact since robot's performance depends on it. Sometimes, to find GLMs in an uncontrolled environment in a brief period of time is a challenging task. For instance, many indoor environments have small objects (chairs, tables, etc.) besides the planar objects.

This thesis proposes two methods to find local GLMs using laser scans obtained from a static position: the *Window Sample Consensus* algorithm (WSAC) and the *Window Sample Consensus – Global Evaluation* algorithm (WSAC–GE). The basic idea behind these methods is to search a line within a subset of consecutive measurements using a Random Sample Algorithm. GLMs extracted by these methods are more assertive and more precise than GLMs obtained by other algorithms commonly used, as is shown in the experimental results.

Finally, this thesis also deals with a simple algorithm which applies GLM to match two consecutive laser scans. The goal is to get a global map and to know the robot's position in every moment.

# Contenido

# Lista de Figuras

# Lista de Tablas

# Lista de Símbolos

| | |
|---|---|
| $x$ | el valor medido de la variable $x$ |
| $\underline{x}$ | el valor real de la variable aleatoria $x$ |
| $\widehat{x}$ | el estimado de la variable $x$ |
| $\overline{x}$ | el valor promedio de la variable $x$ |
| $\rho(\cdot)$ | función objetivo |
| $\psi(\cdot)$ | función de influencia |
| $w(\cdot)$ | función de ponderación |
| $P$ | una imagen de rango en coordenadas cartesianas |
| $S$ | una imagen de rango en coordenadas polares |
| $p_i$ | i–ésima lectura de la imagen de rango, $p_i \in P$ |
| $q_i$ | i–ésima posición del robot |
| $\Theta$ | parámetros de una línea en 2D |
| $[m, b]$ | los parámetros de la línea en forma pendiente, ordenada en el origen |
| $[\alpha, r]$ | los parámetros de la línea en forma polar |
| $e_i$ | error del punto $p_i$ a una línea |
| $e_{\perp i}$ | error perpendicular del punto $p_i$ a una línea |
| $e_{i,\Theta}$ | error del punto $p_i$ a la línea con parámetros $\Theta$ |
| $\mathcal{M}^{(i)}$ | el mapa obtenido en la i–ésima iteración |
| $g_i$ | la i–ésima posición del robot |
| $\mathcal{M}_i$ | el mapa obtenido desde la posición $g_i$ |
| $\dot{\mathcal{M}}$ | el mejor mapa de lineas estimado |
| $H(\mathcal{M}, P)$ | la evaluación global del mapa $\mathcal{M}$ respecto de la imagen de rango $P$ |

# Lista de Publicaciones

C. Lara, L. Romero, *A Robust Approach to Build 2D Geometric Maps From Laser Scans*. Research in Computing Science, 7th Conference on Computing, CORE 2006. CIC IPN. México, D.F. ISSN: 1665-9899.

L. Romero, C. Lara *A New Approach to Learn 2D Line Maps From Laser Scans*. In proceedings of the FIRA RoboWorld Congress 2006. University of Dortmund. Dortmund, Germany. (Accepted)

# Capítulo 1

# Introducción

La robótica móvil tiene como objetivo principal la construcción de robots que sean capaces de navegar en entornos desconocidos y tal vez cambiantes sin necesidad de supervisión. El éxito en la navegación del robot está ligado al buen desempeño de los siguientes sistemas: percepción, localización, cognición y control de movimiento [Siegwart04]. En el bloque de *percepción* el robot interpreta los datos sensoriales obtenidos del ambiente para recuperar información útil; en el de *localización*, deduce su posición en el ambiente; en el de *cognición*, toma decisiones sobre como actuar; y finalmente en el de *control de movimiento*, modula sus actuaciones motoras para lograr la trayectoria deseada.

En la actualidad existe una amplia gama de sensores: sonares, radares, telémetros láser y cámaras entre otros que pueden ser usados en el sistema de percepción. El sistema de percepción puede usar uno o varios sensores. Un tipo específico de sensores, denominado *sensores de rango*, mide la distancia del sensor al objeto más próximo en una determinada dirección. A una serie ordenada de mediciones tomadas por un sensor de rango se le conoce comúnmente como *imagen de rango*.

Esta tesis se enfoca al estudio de un sistema de percepción basado en un telémetro láser. El telémetro láser o simplemente láser es un sensor de rango que ha ganado popularidad gracias a su alta resolución, precisión y velocidad de adquisición de datos.

Cualquier tipo de sensor muestra esencialmente dos tipos de limitaciones: el *ruido* y la *no singularidad de las lecturas*[Siegwart04]. El ruido es una diferencia entre el valor real y el valor reportado por el sensor, la naturaleza del ruido depende del tipo de sensor. La *no singularidad* implica dificultades para diferenciar entre dos lecturas (v.g. en el caso del láser se desconoce en primera instancia si el objeto medido es una pared o una persona). Para reducir las limitaciones de los sensores es común combinar varias lecturas, ya sea del mismo sensor o de varios sensores. El procesamiento de las mediciones de los sensores para encontrar las *características del entorno* es una forma de combinación. [Siegwart04] define una característica como una estructura de elementos reconocibles en el entorno.

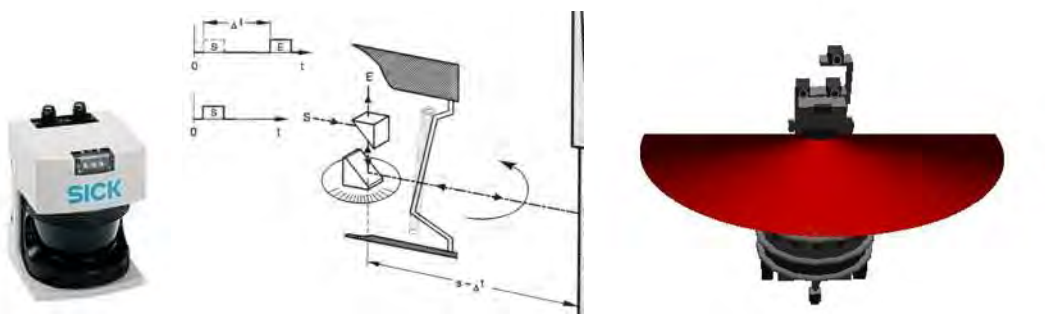
Las líneas son características muy simples y es común obtener varias líneas a partir de una imagen de rango. Al conjunto de líneas que representan un entorno le llamaremos *Mapa Geométrico de líneas* (MGL) o simplemente *mapa de líneas*. Los mapas de líneas son una forma natural de representar un ambiente interior donde abundan los objetos con contornos lineales. Por último, los MGLs tienen la ventaja de representar el ambiente de forma compacta.

El presente capítulo incluye los temas siguientes: en la sección 1.1 se explica la naturaleza de las mediciones realizadas con un telémetro láser. En la sección 1.2 se describen los problemas de la generación de mapas, mientras que en la sección 1.3 se describe la relación de los mapas con el importante problema de conocer la posición del robot. En la sección 1.4 se muestran los trabajos previos realizados en el Posgrado de Ingeniería Eléctrica de la Universidad Michoacana de San Nicolás de Hidalgo que contribuyen de manera directa en el presente trabajo.

Los objetivos generales y específicos de la tesis se presentan en la sección 1.5. Los alcances, retos y contribuciones se muestran en la sección 1.6 y finalmente en la 1.7 se describe la organización del resto de esta tesis.

## 1.1. Percepción del entorno mediante un láser

El sistema de medición basado en láser (LMS, del inglés *Láser Measurement System*) también llamado telémetro láser o simplemente láser es un sensor comúnmente usado en las áreas de robótica móvil y visión. Su uso se ha difundido gracias a que es preciso, rápido y direccional; sus aplicaciones van desde la generación de mapas bidimensionales (2D) y tridimensionales (3D), localización y reconstrucción 3D. En este trabajo se procesa la información obtenida mediante un láser, por lo que dedicamos el resto de la sección al estudio de este sensor.




(a)Láser (b)Principio de operación (c)Barrido semicircular del láser

Figura 1.1: El láser usa el principio de tiempo de vuelo para medir la distancia de los objetos más próximos.

El láser usado en este trabajo [SIC], véase la figura 1.1(a), opera sobre el principio del tiempo de reflexión de la luz (en inglés *Time Of Flight*). El láser emite pulsos de luz muy cortos, cuando los pulsos encuentran un objeto, la señal se refleja y regresa al sensor. Dentro del sensor utilizado en ésta investigación existe un espejo que rota uniformemente (figura 1.1(b)). El espejo desvía los pulsos de luz, formando un barrido de un área semicircular, como el que se muestra en la figura 1.1(c). La $i$-ésima lectura del láser está dada por $(r_i, \alpha_i)$. Donde $r_i$ es la distancia y $\alpha_i$ es la dirección en la que se detectó el objeto. Para calcular $r_i$ el láser mide el tiempo que la señal tardó en

regresar mientras que para calcular $\alpha_i$ se usa el ángulo del espejo. El láser utilizado se configuró para realizar 361 mediciones desde $\alpha_1 = -90°$ hasta $\alpha_{361} = 90°$

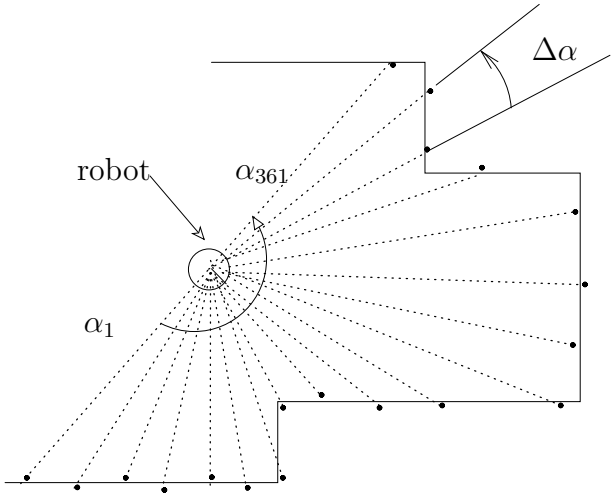



Figura 1.2: El láser mide la distancia haciendo un barrido semicircular, partiendo de un ángulo inicial de $\alpha_1 = -90°$ hasta un ángulo de $\alpha_{361} = 90°$.

En la figura 1.2 se muestra una imagen de rango tomada por un láser en un ambiente interior. En cada imagen de rango de láser se toman $n$ lecturas, desde un ángulo inicial $\alpha_1$ hasta un ángulo final $\alpha_n$. Cada lectura difiere de la anterior en un ángulo de $\Delta\alpha$, es decir, $\alpha_{i+1} = \alpha_i + \Delta\alpha$. A la secuencia de $n$ mediciones del láser $((r_1, \alpha_1), (r_2, \alpha_2), \ldots, (r_n, \alpha_n))$, se le llamará *imagen de rango bidimensional* o simplemente *imagen de rango*. Las lecturas tomadas por el láser se realizan en el mismo plano, llamado *plano de medición*. En muchas aplicaciones de robótica móvil se suele montar un solo láser en una plataforma móvil de giro–inclinación (*pan–tilt*) logrando la posibilidad de modificar el *plano de medición*. Si en una posición determinada el robot toma un conjunto de imágenes de rango con diferentes planos de medición entonces se obtiene una *imagen de rango tridimensional*.

Ningún sensor está libre de errores, por el contrario, los errores son parte de la naturaleza en cualquier proceso de medición. Aún más, cada tipo de sensor tiene sus propias limitantes. Cuando se utiliza un láser, existen factores internos y externos

que influyen en la precisión de las mediciones [Boehler03]: los factores internos tienen que ver con las características específicas del láser como ángulo mínimo, resolución angular y el error promedio de las mediciones; entre los factores externos están la reflectividad de las superficies sensadas, las condiciones ambientales de temperatura, niebla e interferencia por alguna fuente de radiación. Casos críticos de problemas de reflectividad de las superficies ocurren cuando se trata de detectar superficies transparentes (como vidrios), superficies brillantes (que actúan como espejos) o superficies negras (que absorben la luz).

## 1.2. El problema de la generación de mapas bidimensionales

Para actividades de alto nivel como planificación de trayectorias, el robot requiere un mapa. Un mapa es una descripción que representa la estructura del ambiente [Matsumoto99]. En los primeros robots móviles era común tratar de generar un mapa a mano del ambiente para que el robot pudiera navegar. Sin embargo, la generación manual del mapa como paso preliminar no siempre es posible y normalmente el mapa se construye mientras el robot recorre su entorno. .

Existen esencialmente dos tipos de mapas: los mapas métricos y los mapas topológicos. Los mapas métricos usan la geometría del ambiente para representarlo, mientras que los mapas topológicos representan el entorno con grafos no dirigidos donde cada nodo corresponde a distintos lugares del ambiente. Los mapas métricos son más exitosos cuando se requiere más precisión y los mapas topológicos han demostrado su eficiencia en la navegación en entornos donde la información métrica existente puede ser ambigua, por ejemplo en pasillos largos. Otros métodos usan ambos tipos de mapas para aprovechar lo mejor de cada enfoque [Filliat03].

Entre los mapas métricos se distinguen los *mapas de celdas* y los *mapas geométricos*. Los *mapas de celdas*, también llamados mapas basados en pixel, representan el

entorno con una matriz de celdas. Cada celda o pixel representa una cierta área o volumen en el ambiente y tiene asociada una probabilidad de que esté ocupado por un objeto. Los mapas de celdas son valiosos cuando se trata de planificar rutas para la exploración del ambiente. El principal inconveniente de los mapas métricos de celdas es la cantidad de espacio que requieren. Otra dificultad, si se requiere un mapa muy preciso, es la discretización del espacio.

Los *mapas geométricos* son aquellos construidos con primitivas geométricas: puntos, líneas, arcos de círculo, etc. y cada primitiva representa la totalidad o parte de un objeto del ambiente. Aunque los mapas geométricos son más finos y ocupan menos espacio en memoria que los mapas de celdas, tienen la desventaja de que se pueden obtener primitivas poco confiables; por lo que el mayor reto al usar mapas geométricos es usar algoritmos eficientes para extraer las características adecuadas del entorno.

Otra característica importante de un mapa es su dimensionalidad. Este trabajo se enfoca al estudio de los mapas en dos dimensiones (2D), aunque también se puede usar mapas en tres dimensiones (3D) [Thrun00, Liu01, Hähnel02, Miles04]. A lo largo de este documento se emplea el término *Mapa Geométrico de Puntos y Líneas* (MGPL), para referirse a aquellos mapas geométricos bidimensionales que usan un conjunto de líneas y puntos para representar una parte del ambiente. De forma análoga se usa el término *Mapa Geométrico de Líneas* MGL a aquel que usa un conjunto de líneas para representarlo. El MGPL puede ser visto como un MGL enriquecido con aquellos puntos dentro de la imagen de rango que no pertenecen a ninguna de las líneas del MGL.

Muchos algoritmos para navegación en ambientes interiores han preferido los MGPL, debido a que tanto los puntos como las líneas son primitivas simples y en conjunto pueden representar casi cualquier contorno. Muchos objetos tales como: muros, puertas, ventanas, sillas, cajas, etc. que abundan en los ambientes interiores se pueden representar con segmentos de líneas.

Para extraer un mapa bidimensional de una imagen de rango tomada en un plano

y desde una posición estática del robot se han propuesto varios métodos, para un panorama general consulte [Borges04, Sack03, Nguyen05]. El ruido y las oclusiones son dos factores que dificultan la extracción de las líneas. En [Forsyth03] se consideran tres los grandes problemas del proceso de extracción de líneas: calcular la cantidad de líneas, agrupar los puntos que pertenecen a cada línea, y estimar los parámetros de cada línea. Las fallas frecuentes de los algoritmos son: obtención de líneas inexistentes, falta de detección de líneas, detección de líneas imprecisas y generación de múltiples segmentos de la misma línea.

## 1.3. El problema de la localización

Para generar un mapa es necesario que el robot conozca su localización y para conocer la localización se requiere un mapa, éste es un problema similar al de averiguar quién fue primero, el *huevo o la gallina*. En la práctica, el problema se soluciona simultáneamente, se construye el mapa mientras que se realiza la localización. En la literatura a éste proceso se le conoce como SLAM (del inglés *Simultaneous Localization And Mapping*) [Leonard90]. El problema de SLAM tiene gran importancia en el campo de la robótica móvil y en la actualidad se considera como el problema de percepción más difícil en la robótica móvil [Thrun02].

Existen diferentes enfoques para solucionar el problema del SLAM los cuales se diferencian principalmente [Siegwart04] en:

1. la forma en que representan el ambiente, y

2. la forma en que representan la posición del robot dentro del ambiente.

En la sección anterior se ha discutido la manera de representar el ambiente, ahora describiremos brevemente cómo se representa la posición del robot. La posición del robot se puede hacer de dos maneras: aquellas donde se usa una única hipótesis de la posición del robot y aquellas que usan múltiples hipótesis de la posición del

robot [Thrun98, Arras02]. El primer enfoque calcula lo mejor posible la posición del robot en el entorno, mientras que el segundo enfoque calcula las probabilidades de varias posiciones posibles del robot dentro de su entorno. La principal ventaja de la representación de hipótesis única es que no existe ambigüedad de la posición por lo que se facilita la toma de decisiones. Sin embargo, esto constituye también su principal reto pues se requiere manejar de manera adecuada la incertidumbre ocasionada por los errores de sensores y actuadores [Siegwart04]. En esta sección se estudia un método sencillo que estima la posición única del robot al resolver el problema de registro entre dos imágenes de rango consecutivas.

Si se tuviera un mapa establecido entonces la localización se pudiera obtener al alinear un conjunto de observaciones con el mapa. En [Borenstein96] se describen diferentes aproximaciones para alinear observaciones. Dudek y Jenkin [Dudek00] clasifican los métodos de alineación en las siguientes categorías:

**Alineación datos–datos.** Este tipo de métodos alinean la última imagen de rango con los datos de imágenes de rango previas.

**Alineación datos–modelo.** Asocia los puntos de una imagen de rango con un mapa que almacena modelos (v.g. líneas, polígonos, curvas).

**Alineación modelo–modelo.** Alinea los modelos abstractos obtenidos de la imagen de rango actual con un mapa que almacena modelos.

En robótica móvil, la alineación datos–datos ha sido empleada con éxito gracias a que entre dos imágenes consecutivas se supone un movimiento muy pequeño, o en su defecto, existe una buena aproximación (v.g. lectura odométrica) de la nueva posición [Zhang94].

En [Arellano05] se expone una primera aproximación para resolver el problema del SLAM. La forma en que representa el ambiente es usando mapas de puntos (MP) y para representar la posición del robot se calcula una posición única. El método general

usado en [Arellano05] se muestra en el algoritmo 1. La posición del robot $q_i$ está definida por sus coordenadas cartesianas del ambiente y por la orientación del mismo. El robot efectúa un recorrido $[q_1, q_2 \dots q_n]$, donde cada posición $q_i$ se desconoce. Desde cada posición $q_i$ se toma una imagen de rango $P_i$, es decir después de efectuar el recorrido completo se tiene una secuencia de imágenes de rango, $[P_1, P_2 \dots P_n]$.

Se puede considerar la posición inicial $q_1$ como punto de referencia global y para estimar la posición $q_{i+1}$ se usa la información de las imágenes $P_i$ y $P_{i+1}$. Si se usa una alineación datos–datos entre $P_i$ y $P_{i+1}$ para encontrar la transformación rígida $\Phi$ que minimice el error entre ellos, entonces se puede calcular la nueva posición del robot $\widehat{q_{i+1}}$.

Para alinear dos imágenes de rango de láser y obtener la transformación $\Phi$, se pueden usar varias técnicas. Un procedimiento común usando el enfoque datos–datos es conocido como *Alineado Iterativo de Puntos más Cercanos* ICP (*del inglés Iterative Closest Point* ) [Besl92]. La idea de ICP es la siguiente: Dado que existe un movimiento muy pequeño entre dos imágenes entonces una curva en la primera imagen se encuentra muy cerca de la curva correspondiente en la otra imagen. Por lo tanto, si se alinean los puntos que pertenecen a una curva con los puntos más cercanos de la curva correspondiente en el segundo mapa, se puede encontrar una transformación donde las curvas estén más cercanas. El algoritmo se aplica iterativamente, en cada paso se encuentra una mejor estimación de la transformación rígida $\Phi$ [Zhang94].

Para la minimización del error es común usar un estimador de mínimos cuadrados [Gutmann96, Latecki04, Lu97], pero se puede lograr mejores resultados si se emplean estimadores robustos, por ejemplo en [Arellano05] se usa el estimador Lorenztiano.

Una de las dificultades que el algoritmo ICP tiene al aplicarse al problema de registro en imágenes de rango se ilustra en la figura 1.3. Los puntos correspondientes $p_k^1$ y $p_k^2$ tomados desde las posiciones $q_1$ y $q_2$ respectivamente en realidad representan dos posiciones distintas separadas una distancia $d$. En otras palabras, es poco probable

**Algoritmo 1** Solución al SLAM por medio de imágenes de rango consecutivos

**entradas.** Una secuencia de $n$ mapas locales tomados de posiciones adyacentes $[P_1, P_2 \dots P_n]$

**salidas.** Un MP global, la posición final da del robot $\widehat{q}$

1. Hacer $i = 1$
2. La primera imagen de rango,$P_1$, se integra al MP global. Se considera la referencia de $P_1$ como como sistema de referencia global.
3. Incrementar $i$.
4. Si $i > n$, termina.
5. Se calcula la transformación rígida $\Phi_{i+1}$ que alinee el rastreo $P_{i+1}$ con el rastreo $P_i$.
6. Usando la transformación $\Phi$, se transforma $P_{i+1}$ en $P'_{i+1}$ y se calcula la nueva posición del robot $\widehat{q}$.
7. Agregar $P'_{i+1}$ al mapa global, hacer $P_i \leftarrow P'_{i+1}$ y regresar al paso 3.

que los puntos asociados representen el mismo punto del espacio.

Para solucionar el problema causado por los errores de correspondencia se han propuesto algoritmos similares a ICP. Una buena opción consiste en minimizar las distancias entre los puntos de una imagen respecto de la línea tangente o secante del punto correspondiente en la otra imagen, como lo muestra la figura 1.3. La bondad de minimizar la distancia del punto a una tangente es que se puede aplicar a cualquier tipo de curva suave. Ejemplos algoritmos que usan este enfoque se pueden encontrar en [Chen91] y en [Arellano05]. En [Chen91], se calcula la línea tangente usando número reducido de puntos próximos al punto asociado mientras que en [Arellano05] se usa

**Algoritmo 2** Algoritmo general de alineación usando ICP

**entradas.** Las imágenes $P_i$ y $P_{i+1}$

**salidas.** La transformación rígida estimada $\widehat{\Phi}$

Iterar hasta que converja:

1. Asociar cada punto de $P_i$ al punto más cercano de $P_{i+1}$
2. Calcular la transformación rígida $\widehat{\Phi}$ que minimice el error entre los puntos apareados.
3. Aplicar la transformación a $\widehat{\Phi}$.

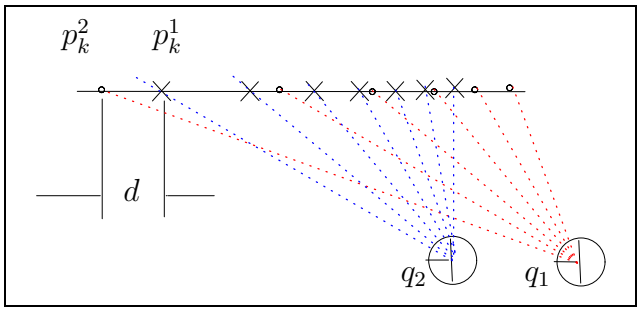
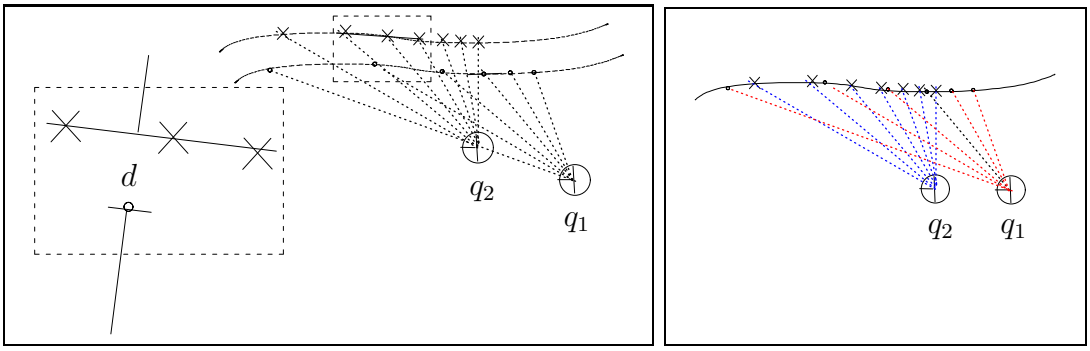



Figura 1.3: Error de correspondencia entre puntos de imágenes de rango consecutivas

el método de *componentes principales* PC lo que obtiene una línea aproximada a la tangente. Aunque tiene buenos resultados, la técnica usada en [Arellano05] consume más tiempo que el método que calcula la tangente. Por último, cabe mencionar que la alineación modelo–modelo se ha realizado con éxito en ambientes interiores al usar MGLs [Pfister02, Zezhong03]. La representación compacta de los MGLs reduce el tiempo de procesamiento y el espacio de memoria requerido. Este factor se vuelve muy importante cuando el tamaño del ambiente es demasiado grande.




(a)Aproximación inicial, el detalle muestra la minimización del punto a la línea tangente

(b)Posición final

Figura 1.4: Minimización de la distancia a la línea tangente

## 1.4. Antecedentes

En la actualidad la División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica de la UMSNH cuenta con un robot móvil. Este robot fue construido localmente con la idea de reducir costos por un lado e impulsar la investigación en el área de robótica por el otro [Romero01]. El robot del posgrado al igual que el robot humanoide PINO [Williams02] es de plataforma abierta. El concepto de plataforma abierta está regido por la *Licencia General Pública* GPL (en inglés *General Public License*) [GPL91] lo cual permite compartir y hacer modificaciones al código sin ningún costo. Nuestro robot, que se muestra en la figura 1.5, tiene una configuración de locomoción diferencial [L. Jones98], con dos ruedas de tracción y dos ruedas de apoyo, un anillo de 16 sonares, 3 cámaras y un láser montado sobre un sistema *pan-tilt*.

El robot está fabricado de aluminio con una estructura de acero inoxidable. Para el procesamiento de alto nivel el robot tiene una computadora con un procesador Pentium III @ 1 GHz, 256 MB de RAM, y 1 slot de PCMCIA. La computadora de abordo es la encargada de procesar la información tomada de los sensores. La figura 1.6 muestra el microcontrolador 9S12DP256C de Motorola [Arts04] que es usado para el procesamiento de bajo nivel del robot. El microcontrolador cuenta con dos módulos

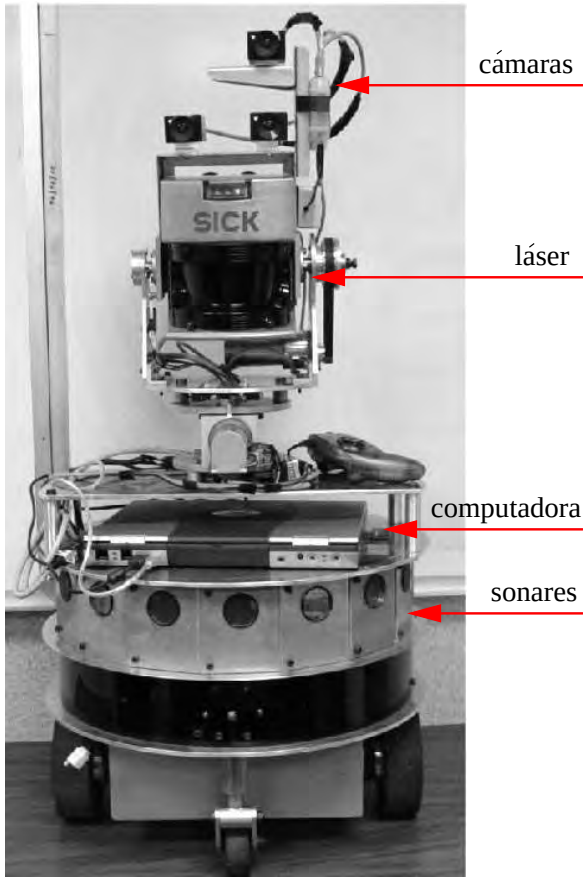



Figura 1.5: El robot

Adapt9S12DP256, un módulo controla los motores de las ruedas y el anillo de sonares, y el otro controla los motores de giro e inclinación del *pan-tilt*.

Utilizando este robot se han realizado dos tesis de maestría en la Facultad de Ingeniería Eléctrica. Una aborda la integración del telémetro láser y el sistema de visión estéreo para la detección de obstáculos [Núñez04]. En otra se propone un método para la solución al SLAM y además se trata el problema de la construcción de un mapa 3D a partir del telémetro láser montado sobre el sistema de giro – inclinación [Arellano05].

Figura 1.6: Imagen del módulo Adapt9S12DP256 del microcontrolador 68hcs12 [Arts04].

## 1.5. Objetivo de la tesis

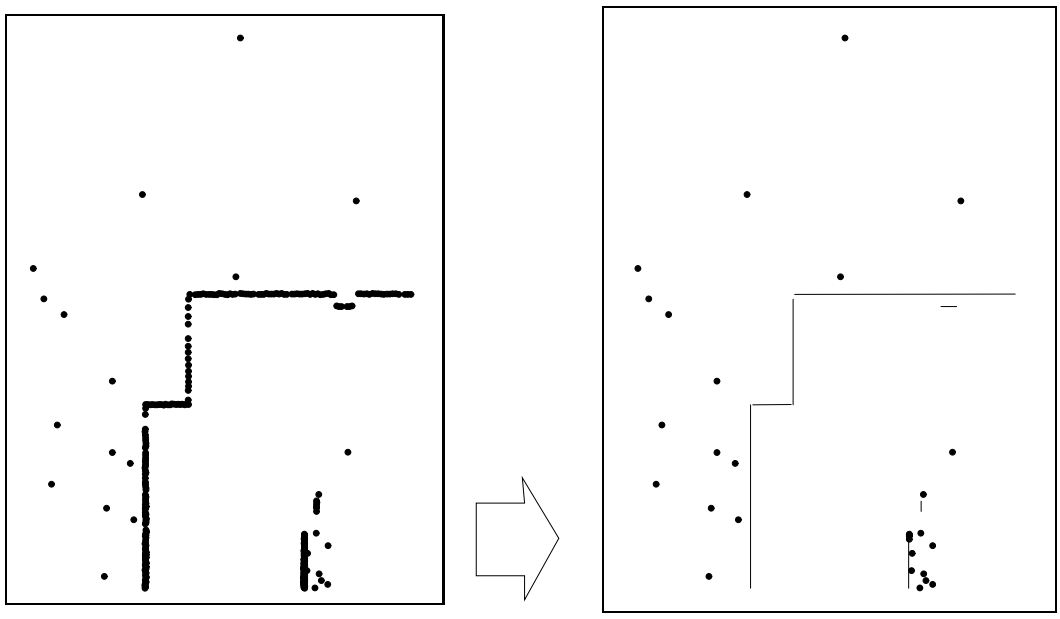
El objetivo general de esta tesis es la construcción de un mapa geométrico de puntos y líneas (MGPL) a partir de una imagen de rango tomada con el láser. La figura 1.7 ilustra el problema que se desea resolver a través de un ejemplo: a partir de las mediciones mostradas en 1.7(a) obtener las siete líneas mostradas en 1.7(b). Estas líneas más los puntos que se muestran representan el ambiente.

Los objetivos particulares son:

1. Analizar el estado del arte de los métodos de generación de un MGPL a partir de una imagen de rango. El estudio bibliográfico permitirá conocer las diferentes propuestas surgidas en los últimos años. Se identifican ideas comunes, aportes originales y se agrupan en paradigmas.

2. Proponer algoritmos robustos que generen un MGPL de una imagen de rango. El MGPL resultante deberá ser lo suficientemente preciso para que pueda servir como instrumento en la solución de otros problemas del área de robótica móvil.

3. Evaluar el desempeño de los algoritmos propuestos, señalar sus virtudes y sus debilidades. Comparar el desempeño de los algoritmos propuestos con otros que



(a)Imagen de rango (b)MGPL

Figura 1.7: Se desea obtener un MGPL a partir de una imagen de rango

comúnmente se usan en el área de robótica móvil.

4. Desarrollar bibliotecas de software que integren los trabajos previos realizados en el área de Robótica del Posgrado de Ingeniería Eléctrica de la UMSNH.

## 1.6. Motivación

Los MGL tienen aplicación directa a la solución de problemas de localización [Arras97, Castellanos96, Roumeliotis00, Alempijevic04, Einsele01, Zezhong03]. Se desea encontrar un nuevo algoritmo para obtener un MGL de buena calidad y lo suficientemente rápido. Nuestra motivación se basa en la relación que existe entre MGL y la solución al problema de localización. Por otro lado el mapa es en sí, un objetivo en algunas aplicaciones.

Aunque, por una parte, proponer nuevos métodos para generar MGL pudiera pa-

recer trivial ya que se han propuesto varios métodos para su solución en la literatura, el problema del ruido en las mediciones y datos atípicos no está resuelto del todo. Por otra, nos entusiasma la idea de encontrar un método robusto y flexible que pueda ser extendido al espacio tridimensional para la generación de mapas 3D o para la reconstrucción tridimensional [Gätcher05].

### 1.6.1. Alcance

Para la generación de MGL se tienen las siguientes limitantes:

**El Ambiente.** El ambiente en donde el robot se desenvuelve es interior, estructurado, estático y sin ninguna marca artificial.

Los ambientes para los que está pensado el robot son de tipo oficina, con objetos de uso común como muros, ventanas, sillas, mesas, libros, computadoras, etc. El único requerimiento es que el piso sea plano para que el robot pueda navegar. Es importante hacer notar que no se agrega ninguna marca artificial para facilitar la navegación del robot.

**Robot móvil cilíndrico con capacidad de giro en su lugar.** La forma cilíndrica del robot permite simplificar la representación del robot e incrementar la movilidad del mismo. Por ejemplo, si el robot llega al final de un pasillo angosto en forma de 'L' y desea continuar, simplemente puede dar un giro de 90 grados y avanzar. La navegación para un robot con otra forma resulta compleja o inclusive puede no ser factible para el caso del pasillo en forma de 'L' [Romero01].

**Telémetro Láser 2D.** El láser usado es un LMS209-S02 de SICK [SIC] que puede tomar lecturas en un rango de $[0, 180°]$. La resolución angular está configurada a $0.5°$, por lo que se tienen 361 lecturas en cada imagen de rango del láser. Su precisión es de $\pm 2cm$ con un alcance de hasta 32 $m$. El rango de transmisión es

de un máximo de 500 Kbaudios, sin embargo la velocidad usada es hasta 38400 bps por usar un dispositivo estándar de comunicación serial.

### 1.6.2. Retos

El algoritmo de generación de MGPL a partir de una imagen de rango deberá cumplir las siguientes características:

**Rápido.** El tiempo de ejecución del algoritmo debe ser el menor posible ya que el robot móvil debe de ser capaz de responder en tiempo real.

**Robusto.** Debe ser tolerante al ruido en las mediciones.

**Preciso.** Debe ser capaz de obtener un MGPL con la precisión suficiente para que sirva para diversas tareas dentro del área de robótica móvil.

**Extensible.** Se desea que el algoritmo pueda extenderse fácilmente a mapas 3D.

### 1.6.3. Contribuciones

Para encontrar un mapa de líneas a partir de imágenes de rango se han propuesto varios algoritmos, pero aun no es un problema completamente resuelto ya que cada algoritmo muestra algunas dificultades. A lo largo de esta tesis se estudian los algoritmos propuestos en la literatura para obtener un MGL a partir de una imagen de rango.

La principal contribución de este trabajo es la propuesta de dos algoritmos para la generación de un MPL a partir de una imagen de rango: el *algoritmo de Consenso aleatorio en una ventana* (WSAC, del inglés Window SAmple Consensus) y el *algoritmo de Consenso aleatorio en una ventana con evaluación global*(WSAC–GE, del inglés Window SAmple Consensus–General Evaluation). Estos algoritmos tratan de aprovechar la secuencialidad de las imágenes de rango obtenidas por un láser al hacer

una búsqueda de selección aleatoria dentro de una "ventana"(un conjunto de puntos consecutivos) y posteriormente revisan el resto de la imagen de rango. Los algoritmos son suficientemente rápidos para ser usados en aplicaciones de robótica móvil y además son robustos.

## 1.7. Organización del documento

En este primer capítulo se ha mostrado el contexto en el que se desarrolla la tesis, se parte de un panorama general sobre los mecanismos que usualmente el robot usa para percibir el entorno con un láser y se ha delimitado el problema. En el resto del documento se explicará con detalle los distintos aspectos de la solución propuesta. El capítulo 2 analiza las distintas aproximaciones reportadas en la literatura para la estimación robusta de una línea. El desarrollo matemático de uno de estos métodos, conocido como *Mínimo de cuadrados totales ponderados*, se muestra en el apéndice A. El capítulo 3 analiza los métodos para generación de un MGL partiendo de una imagen de rango. Para algunos de los algoritmos descritos en el capítulo 3 es indispensable conocer si dos lineas son iguales, es por ello que el apéndice B incluye un método de evaluación de la similitud entre dos líneas basado en sus parámetros geométricos.

El capítulo 4 introduce los algoritmos WSAC y WSAC–GE que generan un MGL a partir de una imagen de rango. Para probar estos algoritmos, el capítulo 5 compara sus resultados con los obtenidos por el popular algoritmo SMSM. Además de las pruebas que usan mediciones tomadas por telémetro láser montado en el robot se realizaron pruebas que usan imágenes sintéticas. Al usar imágenes de rango sintéticas entonces se conoce el MGL ideal equivalente y pueden calcular los errores de los MGLs obtenidos por diferentes métodos. El simulador implementado para generar las imágenes de rango sintéticas se describe en el apéndice C.

Como se ha descrito anteriormente, los MGL se pueden usar como herramienta para la solución al problema del SLAM. El capítulo 5 también describe un método

de solución de este problema y muestra los resultados experimentales obtenidos.

El último capítulo, el de conclusiones, resume las aportaciones de este trabajo, las soluciones que plantea, las limitaciones que exhibe y las líneas de trabajo propuestas para su exploración en el futuro inmediato.

# Capítulo 2

# Métodos de estimación de parámetros de una línea

Un problema común en diversas áreas de ingeniería, conocido como regresión, consiste en encontrar un modelo matemático a partir de datos experimentales. Este capítulo se enfoca en encontrar una línea que mejor ajusta a un conjunto de puntos en $\mathbb{R}^2$.

## 2.1. Introducción

Existen muchos métodos de ajuste de modelos matemáticos a un conjunto de puntos. Un enfoque clásico es el método de *Mínimos Cuadrados* (LS, del inglés *Least Squares*) que se abordará en la sección 2.3. La solución por el método LS es, en muchos casos, suficiente. Sin embargo existe un tipo de datos, llamados puntos atípicos, que ocasionan dificultades en los problemas de estimación. En [Beckman83] se define un punto atípico (u *outlier*) como un dato que se desvía marcadamente de los otros datos de la muestra donde éste ocurre. Los puntos atípicos son en realidad errores de medición introducidos por el sensor al medir el ambiente. En algunos casos un solo punto atípico es suficiente para que LS encuentre una recta completamente diferente

a la esperada [Rousseeuw87].

Debido a la dificultad de detectar los puntos atípicos y el efecto indeseado que tienen sobre la estimación se han buscado diversas técnicas de *estimación robusta*. En principio podemos asumir que una estimación robusta encontraría una línea que pase por la mayoría de los datos. De manera formal, un estimador robusto se refiere a un estimador que se comporta de forma estable frente a puntos atípicos. Los estimadores robustos que se abordan en este capítulo, por ser ampliamente usados en la literatura, son: *Least Median of Squares* (LMS), los estimadores de *máxima verosimilitud* (M–estimators), la *Transformada de Hough* (HT) y los estimadores de *selección aleatoria*.

## 2.2. Definición formal del problema

Sea $\underline{P} = \left\{\underline{p_i} | i = 1 \dots n\right\}$ el conjunto de $n$ puntos $p_i \in \mathbb{R}^2$ que cumplen exactamente la ecuación de una línea recta con parámetros $\Theta$; si $\underline{P}$ es afectado por errores que siguen alguna distribución de probabilidad $f$, el conjunto de puntos medidos $P = \{p_i | i = 1 \dots n\}$ se puede modelar como $p_i = \underline{p_i} + e_i$, donde $e_i$ denota el error de medición. El problema consiste en recuperar los parámetros $\Theta$ de la línea a partir de $P$.

## 2.3. Métodos de Mínimos Cuadrados

En esta sección se usa el enfoque de *mínimos cuadrados* así como las variantes *mínimos cuadrados totales* y *mínimos cuadrados totales ponderados* para estimar una línea. Para simplificar la discusión se limita el análisis al espacio bidimensional $\mathbb{R}^2$. Sin embargo, las ideas se pueden extender a más dimensiones.

### 2.3.1. LS

El método de *Mínimos Cuadrados* (LS, del inglés *Least Squares*), desarrollado por Legendre y Gauss [Stigler86], representa el enfoque clásico de resolver el problema. En esta sección se usa la ecuación $y = mx + b$ para representar la línea, por lo que se tienen que encontrar los parámetros $\Theta = [m, b]$ de la ecuación. De aquí en adelante se denotará una línea simplemente por $\Theta$.

Si se denota el error vertical de un punto $p_i$ a la línea $\Theta$ por $e_i$ como se muestra en la figura 2.1(a), el criterio de mínimos cuadrados nos indica que la mejor línea será aquella resultante de minimizar la suma de los cuadrados de los errores $e_i = y_i - \widehat{y_i}$, con $\widehat{y_i} = mx_i + b$ para los puntos $p_i$ definidos por $(x_i, y_i)$. Es decir, si calculamos el error total $E(m, b)$:

$$E(m, b) = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} [y_i - (mx_i + b)]^2 \tag{2.1}$$

entonces LS estima la línea al buscar aquellos coeficientes $\widehat{m}$ y $\widehat{b}$ para los que $E$ sea mínima, o sea:

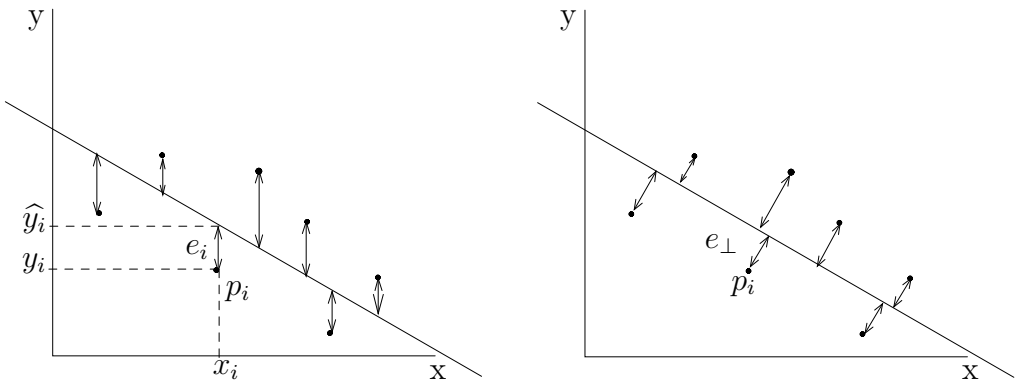
$$\left\langle \widehat{m}, \widehat{b} \right\rangle = \arg \min_{m,b} E(m, b) \tag{2.2}$$

La mejor recta se encuentra al derivar $E(m, b)$ con respecto de $m$ y $b$ e igualar a cero ambas derivadas. La solución del sistema de ecuaciones se muestra en muchos libros de texto (ver por ejemplo [Miller99]) y está dada por:

$$\widehat{b} = \frac{S_{xy}}{S_{xx}} \tag{2.3}$$

$$\widehat{m} = \widetilde{y} - \widehat{b}\widetilde{x} \tag{2.4}$$

donde $\widehat{m}$ y $\widehat{b}$ son los valores estimados de $m$ y $b$ respectivamente, y los parámetros $S_{xx}$ y $S_{xy}$ están dados por:

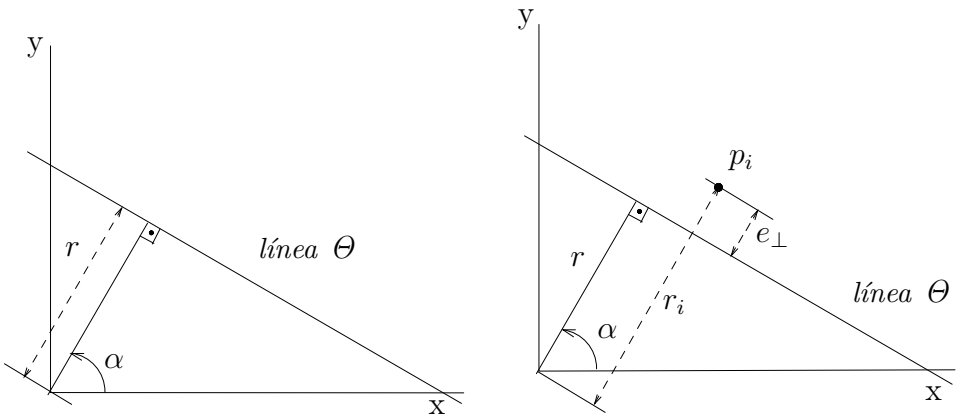



(a)Errores verticales a la línea (b)Error ortogonal a la línea

Figura 2.1: Diferentes formas de medir el error

$$
\begin{aligned}
S_{xx} &= \sum_{i=1}^{n} (x_i - \widetilde{x})^2 \qquad\qquad \widetilde{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \\
&= \sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2 \qquad\qquad \widetilde{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \\
&= \sum_{i=1}^{n} x_i^2 - n\,(\widetilde{x})^2 \\
S_{xy} &= \sum_{i=1}^{n} (x_i - \widetilde{x})\,(y_i - \widetilde{y}) \\
&= \sum_{i=1}^{n} x_i y_i - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right) \\
&= \sum_{i=1}^{n} x_i y_i - n\widetilde{x}\widetilde{y}
\end{aligned}
$$

Conviene hacer notar que este método falla cuando se requiere estimar una recta cercana a la vertical, es decir cuando $m$ crece indefinidamente; el método TLS, que se expone a continuación, no sufre de esta desventaja y proporciona una mejor estimación de una línea recta.




(a)Línea $\Theta = [r, \alpha]$ (b)Cálculo del error ortogonal

Figura 2.2: Cálculo del error ortogonal de un punto $p_i : (x_i, y_i)$ a la línea $\Theta = [r, \alpha]$

### 2.3.2. TLS

Una variante de LS, conocida como *mínimos cuadrados totales* (TLS, del inglés *Total Least Squares*), minimiza la suma de los cuadrados del error ortogonal $e_\perp$ de cada punto $p_i$ a la línea $\Theta$ como se muestra la figura 2.1(b). En forma matemática los parámetros estimados $\widehat{\Theta}$ se obtienen al resolver

$$\widehat{\Theta} = \arg\min_{\Theta} \sum_{i=1}^{n} \left[e_\perp(i, \Theta)\right]^2 \tag{2.5}$$

donde, $e_\perp(i, \Theta)$ es la distancia ortogonal del punto $p_i$ a la línea $\Theta$. Como se puede ver en la figura 2.2(a) la línea $\Theta$ se puede definir en su forma polar $\Theta = [\alpha, r]$, donde $\alpha$ es el ángulo de la normal a la línea con respecto al eje positivo, y $r$ es su distancia al origen. Para cualquier punto dado en su forma cartesiana $p : (x, y)$ que pertenezca a línea $\Theta$ se cumple

$$r = x \cos\alpha + y \sin\alpha \tag{2.6}$$

si se hace pasar por el punto $p_i : (x_i, y_i)$ una recta paralela a $[r, \alpha]$, la definida por $[r_i, \alpha]$, como se puede ver en la figura 2.2(b), se tiene

$$r_i = x_i \cos\alpha + y_i \sin\alpha \tag{2.7}$$

de tal manera que $e_\perp$ se puede expresar como

$$e_\perp(p_i, [r, \alpha]) = r_i - r = x_i \cos\alpha + y_i \sin\alpha - r \tag{2.8}$$

al sustituir esta expresión en la ecuación 2.5 resulta

$$\widehat{\Theta} = \arg\min_{r,\alpha} \sum_{i=1}^{n} \left[x_i \cos\alpha + y_i \sin\alpha - r\right]^2 \tag{2.9}$$

En la siguiente sección se encuentra la solución de la ecuación 2.9 al resolver un esquema más general llamado *Mínimos Cuadrados Totales Ponderados* (WTLS, del inglés *Weighted Total Least Squares*).

### 2.3.3. WTLS

Si el error cuadrado de cada punto $p_i$ tiene un peso específico $k_i$, es decir $e_{\perp k_i} = k_i e_\perp(p_i, [r, \alpha])$, la ec. 2.5 se transforma en la siguiente

$$\widehat{\Theta} = \arg\min_{\alpha,r} \sum_{i=1}^{n} a_i \left[r - x_i \cos(\alpha) - y_i \sin(\alpha)\right]^2 \tag{2.10}$$

donde $a_i = k_i^2$. Si se resuelven las derivadas parciales respecto de $\alpha$ y $r$ y se igualan a cero como se muestra en el apéndice A, la solución de 2.10 es

$$\alpha = \frac{1}{2} \arctan\left(\frac{-2\widetilde{S}_{xy}}{\widetilde{S}_{yy} - \widetilde{S}_{xx}}\right) \tag{2.11}$$

$$r = \widetilde{x}\cos(\alpha) + \widetilde{y}\sin(\alpha) \tag{2.12}$$

donde

$$\begin{aligned}
\widetilde{S}_{xx} &= \sum_{i=1}^{n} a_i \left(x_i - \widetilde{x}\right)^2 & \widetilde{x} &= \frac{\sum_{i=1}^{n} a_i x_i}{\sum_{i=1}^{n} a_i} \\
\widetilde{S}_{yy} &= \sum_{i=1}^{n} a_i \left(y_i - \widetilde{y}\right)^2 & \widetilde{y} &= \frac{\sum_{i=1}^{n} a_i y_i}{\sum_{i=1}^{n} a_i} \\
\widetilde{S}_{xy} &= \sum_{i=1}^{n} a_i \left(x_i - \widetilde{x}\right)\left(y_i - \widetilde{y}\right)
\end{aligned}$$

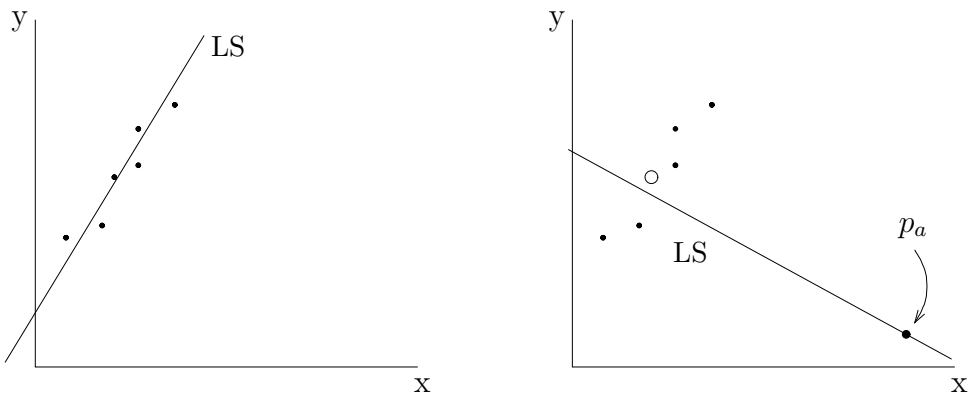
así, el cálculo de WTLS se puede resolver de forma directa. Si $a_i = 1$ $i = 1 \ldots n$, entonces 2.11 y 2.12 son la solución del método TLS visto en la sección anterior.

### 2.3.4. Problemas de los estimadores de mínimos cuadrados

El método LS es aplicado en muchas situaciones pero no siempre es posible obtener buenos resultados, sobre todo cuando se trata de extraer una línea a partir de datos con puntos atípicos. Dado el $i$–ésimo punto $p_i = \underline{p_i} + e_i$, donde $e_i$ es el error, los métodos de mínimos cuadrados se basan en el hecho de que los errores $e_i$ son independientes entre sí y siguen una distribución normal con media cero [Yohai79]. El fundamento teórico de ésta suposición es el teorema de límite central [Miller99].

En los siguientes ejemplos de estimación, tomados de [Rousseeuw87], se usará el método LS para ajustar una línea con el objetivo de mostrar el efecto que produce un punto atípico. En el primer ejemplo sin datos atípicos mostrado en la figura 2.3(a), se ajusta una línea a los datos experimentales y el modelo resultante describe de manera satisfactoria a los puntos.

En el siguiente caso, como lo muestra la figura 2.3(b), se introduce un punto atípico $p_a$. La línea que se obtiene de este conjunto de datos es muy diferente a la anterior y no pasa por la mayoría de los puntos. Si el punto $p_a$ no existiera, el ajuste sería similar a la línea obtenida en el caso de la figura 2.3(a). Cuando se realiza un ajuste en presencia de puntos atípicos se desea que la estimación resultante pase o se ajuste a la mayoría de los datos.




(a)Ajuste LS sin puntos atípicos (b)Ajuste LS con un punto atípico

Figura 2.3: Efecto de un outlier en el ajuste con LS.

## 2.4. Medidas de robustez y estimadores robustos

Los puntos atípicos son inherentes a muchos problemas reales, y en muchas ocasiones no se puede garantizar la ausencia de ellos. A través del tiempo se han introducido diversas técnicas, conocidas como estimadores robustos, que permiten el manejo adecuado de situaciones en las que existen puntos atípicos.

Para evaluar la robustez de los diferentes estimadores se han buscado diferentes formas de medirla. Un indicador común para el grado de robustez es el *punto de ruptura* definido como la máxima proporción de puntos atípicos que un método puede soportar sin que el error de la estimación resultante diverja arbitrariamente de la estimación real [Rousseeuw87]. El punto de ruptura de LS es de cero, porque un solo outlier puede ocasionar una estimación inadecuada.

Los paradigmas de estimación robusta que se describen en las siguientes secciones son LMS, HT, los estimadores–M, y los métodos de selección aleatoria.

## 2.5. LMS

El método de *Mínima Mediana de los Cuadrados* (LMS, del inglés *Least Median of Squares*) es un estimador que se propone en [Rousseeuw87] y consiste en minimizar la mediana (**med**) de las distancias absolutas de los puntos a la línea. Es decir

$$\arg\min_{\Theta}(\mathbf{med}\ \left(e_1^2, e_2^2, \ldots, e_n^2\right)) \tag{2.13}$$

El resultado de la función **med**, se obtiene al ordenar el conjunto de datos y tomar el valor que se encuentre a la mitad, o en su defecto, el promedio de los dos valores centrales. Usando el método LMS se puede tener hasta la mitad de puntos atípicos en el conjunto de datos sin que el valor de la función objetivo se modifique. Este rechazo se debe a que la función **med** rechaza la mitad de los errores $e_i$ más grandes.

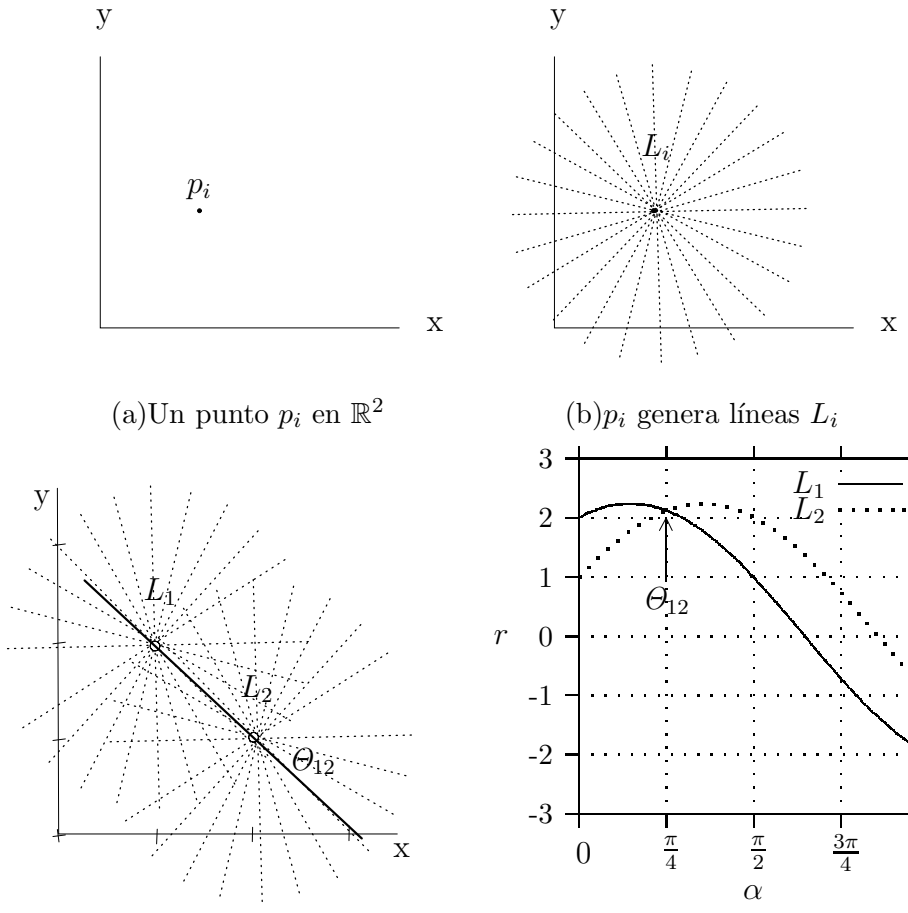
Debido a que la función **med** no es diferenciable se han utilizado técnicas alternativas para resolver 2.13. Para el caso de la regresión lineal en $\mathbb{R}^2$ se conocen algoritmos para resolver 2.13 de forma eficiente [Bernholt05].

Una técnica para resolver LMS muy usada es tomar una muestra aleatoria $S$ de $\|S\|$ puntos de $P$, a la muestra $S$ se ajusta una línea $\Theta$. Usando la línea $\Theta$ se encuentran los errores $e_i$ y la media de los errores cuadrados. Para resolver la ecuación 2.13 el proceso se repite y la línea $\Theta$ con menor error residual medio se escoge como mejor ajuste.

Una variante común a LMS consiste en usar los cuantiles en lugar de la mediana y a este método se le conoce como *Mínimo Cuantil Cuadrado* (LQS, del inglés *Least Quantile of Squares*). Si se disminuye la proporción a menos del 50 % puede ser que se encuentren resultados imprecisos y además exista más de una solución, por lo que en la práctica se prefiere LMS.

## 2.6. La transformada de Hough (HT)

La transformada de Hough (HT) [Hough59] es un método estándar comúnmente usado en el área de visión artificial, es muy útil cuando la información es ruidosa y su simplicidad la hace adaptable a muchas situaciones. La esencia de HT es trasladar la información a un espacio paramétrico adecuado de tal manera que en el nuevo espacio paramétrico se facilite la localización de curvas tales como rectas, polinomios, círculos, etc. A continuación se describe la HT aplicada a la extracción de una línea.




(a)Un punto $p_i$ en $\mathbb{R}^2$ (b)$p_i$ genera líneas $L_i$

(c)La línea extraída con 2 puntos (d)$L_1$ y $L_2$ en el espacio paramétrico

Figura 2.4: La transformada de Hough

El punto $p_i$ ilustrado en la figura 2.4(a) se puede ver como un generador de

líneas. La familia de líneas $L_i$ mostrada en 2.4(b) obtenida a partir de $p_i$ incluye a todas las líneas que pasen sobre el punto $p_i$. Dado el conjunto de puntos $P = \{p_i | i = 1 \dots n\}$, el problema consiste en seleccionar aquella línea que más se repite en $L = \{L_i | i = 1 \dots n\}$. Para el caso de los puntos $p_1$ y $p_2$ de la figura 2.4(c) la línea $\Theta_{12}$ es la única que se repite tanto en $L_1$ como en $L_2$, y por tanto, $\Theta_{12}$ es la línea buscada. Matemáticamente, la familia de líneas generadas por un punto $p_i$ es

$$L_i = \{[r_j, \alpha_j] \mid \ r_j = x_i \cos \alpha_j + y_i \sin \alpha_j\} \tag{2.14}$$

por lo que cada punto en $\mathbb{R}^2$ corresponde a una curva senoide en el espacio paramétrico $[r, \alpha]$ como lo muestra la figura 2.4(d). El problema se soluciona al encontrar el punto donde se interceptan las curvas. El mismo principio se aplica para extraer una línea de un conjunto de $n$ puntos. Para hacer el problema manejable, la HT hace una discretización de $r$ y $\alpha$ y el modelo final se obtiene al encontrar la celda con mayor votación [Goldenshluger04].

La HT tiene una gran aplicación en el espacio bidimensional aunque también se ha usado en dimensiones mayores. Su principal ventaja es la de ser muy tolerante a errores, sin embargo, tiene dificultades por su complejidad temporal y espacial, y también la pérdida de precisión inherente a la discretización. Debido a su flexibilidad y su simplicidad para abordar una amplia gama de aplicaciones, se han realizado una gran cantidad de estudios para disminuir la complejidad temporal y espacial [Weiman94, Giesler98].

## 2.7. Estimadores–M

La sección 2.3.4 ha tratado sobre los efectos indeseados que los datos atípicos ocasionan en los métodos de mínimos cuadrados. Esto se debe a que los estimadores de mínimos cuadrados suponen que los errores siguen una distribución normal y los puntos atípicos violan esta suposición. En [Huber64] se propone sustituir la función

$e^2$ que usa LS por otra que sea menos sensible a puntos atípicos. La nueva función disminuye el efecto de los errores mayores al ponderarlos con pesos pequeños. El análisis que se presenta en esta sección se basa en [Yohai79].

Si se conoce la distribución de probabilidad que siguen los errores entonces se puede utilizar la técnica de máxima verosimilitud para resolver el problema de regresión. La función de verosimilitud de los errores $L(e_1, .., e_n; \Theta)$, asumiendo independencia estadística, está dada por

$$L(e_1, .., e_n; \Theta) = \prod_{i=1}^{n} f(e_{i,\Theta}) \tag{2.15}$$

donde $e_{i,\Theta}$ es el error del punto $i$ a la línea con parámetros $\Theta$ y $f$ es la probabilidad de errores. Si se toman logaritmos naturales se tendrá

$$\ln L(e_1, .., e_n; \Theta) = \sum_{i=1}^{n} \ln f(e_{i,\Theta}) \tag{2.16}$$

al llamar $\rho_f(\cdot) = \ln f(\cdot)$

$$\ln L(e_1, .., e_n; \Theta) = \sum_{i=1}^{n} \rho_f(e_{i,\Theta}) \tag{2.17}$$

para encontrar la línea de máxima verosimilitud se requiere minimizar 2.17

$$\Theta = \arg\min_{\Theta} \sum_{i=1}^{n} \rho_f(e_{i,\Theta}) \tag{2.18}$$

En el caso particular de una distribución normal con media cero y varianza $\sigma^2$, $f = n(0, \sigma^2)$; entonces $\rho_f(e) = e^2$ y al resolver (2.17) se obtienen las expresiones 2.11 y 2.12.

La ecuación 2.18 en realidad no es muy útil ya que se desconoce la distribución $f$ y por ende tampoco se conoce $\rho_f$. No obstante, el problema puede ser resuelto al encontrar una función $\rho(\cdot)$ independiente de $f(\cdot)$ de tal manera que $\rho$ sea robusta ante la distribución $f$. Es decir se tiene que resolver

$$\Theta = \arg \min_{\Theta} \sum_{i=1}^{n} \rho(e_{i,\Theta}) \tag{2.19}$$

la función $\rho(e)$ será aceptable si cumple las siguientes condiciones [Ivan99]:

$$\begin{aligned} \rho(e) &\geq 0 \\ \rho(0) &= 0 \\ \rho(e) &= \rho(-e) \\ \rho(e_1) &\geq \rho(e_2) \qquad para\, |e_1| \geq |e_2| \end{aligned} \tag{2.20}$$

A los estimadores que siguen la ec. 2.19 con una $\rho(\cdot)$ que satisface las condiciones anteriores se les conoce como estimadores–M [Huber64]. En la literatura se han propuesto una gran variedad de funciones $\rho$.

En la tabla 2.1 se muestran cuatro estimadores–M: LS, Huber, Beaton–Tukey (BT) y Cauchy [Stewart99]. Las gráficas correspondientes se muestran en la figura 2.5. Obsérvese que en las funciones Huber, BT y Cauchy los errores grandes tienen un efecto poco significativo, mientras que en LS, los errores grandes tienen un gran efecto en la función.

| Nombre | función objetivo |
|---|---|
| LS | $\rho_{\mathrm{LS}}(e) = e^2$ |
| Huber | $\rho_{\mathrm{H}}(e) = \begin{cases} \frac{1}{2}e^2 & \text{para } \lvert e\rvert \leq k \\ k/\lvert e\rvert - \frac{1}{2}k^2 & \text{para } \lvert e\rvert > k \end{cases}$ |
| Beaton– Tukey | $\rho_{\mathrm{BT}}(e) = \begin{cases} \frac{k^2}{6}\left\{1-[1-(e/k)^2]^3\right\} & \text{para } \lvert e\rvert \leq k \\ \frac{k^2}{6} & \text{para } \lvert e\rvert > k \end{cases}$ |
| Cauchy | $\rho_{\mathrm{C}}(e) = \frac{k^2}{2}\log\left(1+(\frac{e}{k})^2\right)$ |

Tabla 2.1: Funciones objetivo

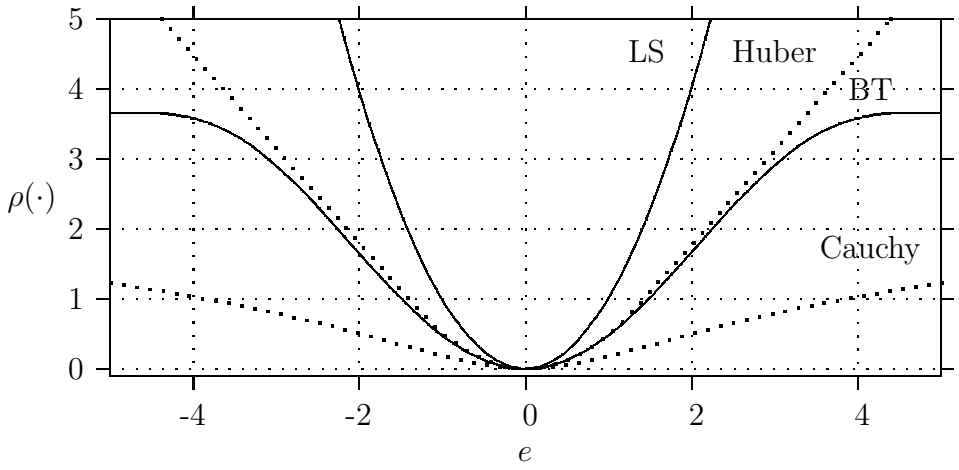



Figura 2.5: Gráfica de las funciones objetivo de la tabla 2.1

Después de seleccionar una función $\rho$ se tiene que resolver la ecuación 2.19. Para ello se deriva con respecto de cada uno de los parámetros de $\Theta$ y se iguala a cero cada derivada.

$$\sum_{i=1}^{n} \psi(e_{i,\Theta}) \frac{de_{i,\Theta}}{d\Theta} = 0 \tag{2.21}$$

donde $\psi(\cdot) = \rho'(\cdot)$ es la derivada de $\rho$ respecto de $e_i$ y se le conoce como *función de influencia* [Hampel86].

Un paso común es normalizar la función $\psi(e)$ al hacer

$$\frac{\psi(e)}{e} = w(e) \tag{2.22}$$

donde a $w(e)$ se le conoce como función de peso o función de ponderación del error. Al despejar $\psi(e)$ de la ec. 2.22:

$$\psi(e) = e \cdot w(e) \tag{2.23}$$

Sustituyendo esta última ecuación en 2.21 tenemos:

$$\sum_{i=1}^{n} w(e_{i,\Theta}) \left( e_{i,\Theta} \frac{de_{i,\Theta}}{d\Theta} \right) = 0 \tag{2.24}$$

La tabla 2.2 y la gráfica 2.6 comparan diferentes funciones de peso. De la gráfica resulta claro que Huber y BT son funciones que rechazan los errores grandes (que

se pueden considerar como puntos atípicos) debido a que mientras el error $e$ tiende a infinito la función se hace cero. Cabe mencionar que la función Cauchy (también conocida como Lorentzian) puede tener más de una solución [Zhang95].

| Nombre | función de ponderación |
|---|---|
| LS | $w_{\mathrm{LS}}(e) = 1$ |
| Huber | $w_{\mathrm{H}}(e) = \begin{cases} 1 & \text{para } \lvert e\rvert \le k \\ k/\lvert e\rvert & \text{para } \lvert e\rvert > k \end{cases}$ |
| Beaton–Tukey | $w_{\mathrm{BT}}(e) = \begin{cases} \left[1-(e/k)^2\right]^2 & \text{para } \lvert e\rvert \le k \\ 0 & \text{para } \lvert e\rvert > k \end{cases}$ |
| Cauchy | $w_{\mathrm{C}}(e) = \frac{1}{1+(e/k)^2}$ |

Tabla 2.2: Funciones de ponderación

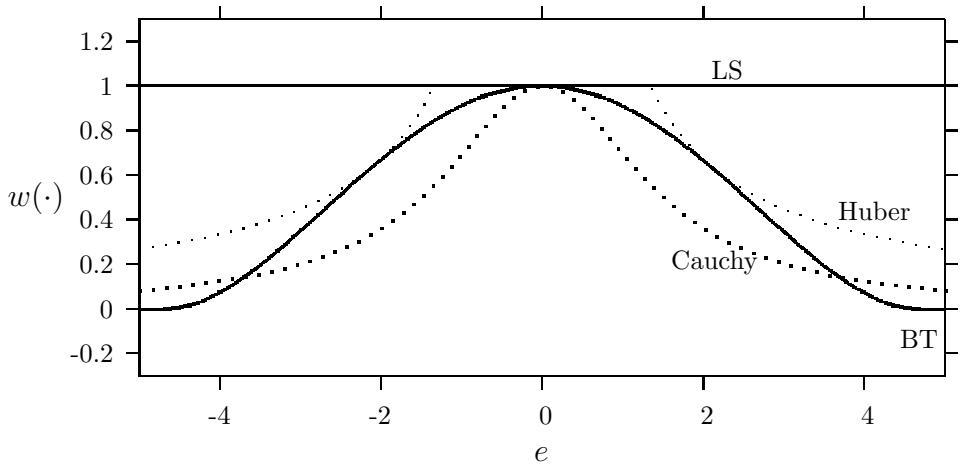



Figura 2.6: Gráfica de las funciones de ponderación de la tabla 2.2

La constante de sintonización $k$ depende del nivel de eficiencia deseado. El nivel de eficiencia normalmente se selecciona para cubrir el 95 % de los datos. La tabla 2.3 resume las constantes de sintonización.

La ecuación 2.24 se puede resolver de varias formas. Un mecanismo simple y popular, llamado *Ponderación Iterativa de Mínimos Cuadrados* (IRLS, del inglés *Iteratively Reweigthed Least Squares*) [Zhang95] consiste en repetir dos pasos. En el primer paso se considera cierta recta $\Theta$ inicial de tal forma que $w_i = w(e_{i,\Theta})$ tiene un valor fijo,

| Función | $k$ |
|---|---|
| Huber | 1.345 |
| Beaton–Tukey | 4.685 |
| Cauchy | 2.385 |

Tabla 2.3: Constantes de sintonización de las funciones de ponderación (eficiencia del 95 %) [Zhang95]

y el problema se reduce a obtener la mejor recta al resolver el sistema de ecuaciones dado por:

$$\sum_{i=1}^{n} w_i \left( e_{i,\Theta} \frac{\partial e_{i,\Theta}}{\partial \Theta} \right) = 0. \tag{2.25}$$

En el segundo paso se recalcula $w_i$ usando el nuevo modelo $\Theta$ recién obtenido. La secuencia de los dos pasos se repite hasta que converjan los parámetros $\Theta$. El método IRLS se muestra en el algoritmo 3 [Zhang95].

**Algoritmo 3** Algoritmo IRLS

**Entradas:** Dados los datos $P = \{p_i | i = 1..n\}$

**Salidas:** Los parámetros de la línea $\widehat{\Theta}$ que mejor se ajusta a $P$

1. $t = 0$
2. Hacer una estimación inicial de los parámetros $\Theta^{(t)}$ (v.g. con LS)
3. Calcular los errores $e_i^{(t)}$ y los pesos asociados $w_i^{(t)} = w(e_i^{(t)})$
4. Resolver el sistema de ecuaciones dado por 2.25 para calcular $\Theta^{(t+1)}$
5. Repetir los pasos 3 y 4 hasta que $\Theta^{(t)} \simeq \Theta^{(t+1)}$, incrementando $t$.

Es importante destacar que el primer paso del método IRLS consiste en resolver 2.25 y esta expresión tiene la misma solución que los métodos de mínimos cuadrados ponderados. Esto se demuestra a continuación para WTLS que minimiza errores ortogonales $e_{i,\Theta} = e_{\perp i,\Theta}$. Como se ha visto anteriormente WTLS minimiza la expresión (2.10), que es

$$\arg\min_{\Theta} \sum_{i=1}^{n} a_i \left(e_{\perp i,\Theta}\right)^2 = \arg\min_{\alpha,r} \sum_{i=1}^{n} a_i \left[r - x_i \cos(\alpha) - y_i \sin(\alpha)\right]^2 \tag{2.26}$$

Para encontrar la solución de 2.26, se debe derivar respecto de los parámetros $\Theta$ e igualar a cero, es decir

$$\frac{\partial}{\partial \Theta} \sum_{i=1}^{n} a_i \left(e_{\perp i,\Theta}\right)^2 = 2 \sum_{i=1}^{n} a_i \left(e_{\perp i,\Theta} \frac{\partial e_{\perp i,\Theta}}{\partial \Theta}\right) = 0 \tag{2.27}$$

esta expresión es similar a 2.25 y su solución se muestra en el apéndice A.

Los estimadores–M robustos tienen un buen desempeño ya que reducen sustancialmente la influencia de los puntos atípicos, sin embargo, al igual que LS tienen un punto de ruptura de cero. Esto se debe a que el método depende de la estimación inicial por una parte y por otra se considera que no existe error en la variable independiente. Para mejorar el punto de ruptura se han desarrollado los estimadores–M generalizados (estimadores–GM) [Yohai79] los cuales logran puntos de ruptura elevados al considerar los posibles errores en la variable independiente. Sin embargo, este tipo de métodos consumen mayor tiempo de cálculo. Otra forma de resolver esta situación de manera favorable es mejorar la estimación inicial, esto se hace comúnmente al amalgamar los estimadores–M con otras técnicas de estimación robusta, por ejemplo los estimadores de selección aleatoria cuya idea general se muestra en la siguiente sección.

## 2.8. Métodos de Selección Aleatoria

Las técnicas que se han revisado hasta el momento estiman un modelo al usar la totalidad de datos del conjunto $P$. En los métodos de *Selección Aleatoria*, por el contrario, en cada iteración usan la mínima cantidad de datos para obtener un modelo. Solamente después de un cierto número de iteraciones se selecciona el mejor

modelo que se ha presentado. Con el fin de hacer la explicación más sencilla se describirá el primer método de selección aleatoria conocido llamado RANSAC. Esto nos servirá como base para describir los métodos de selección aleatoria de forma genérica.

### 2.8.1. RANSAC

El método *Consenso de Selección Aleatoria* (RANSAC, del inglés RAndom SAmple Consensus) [Fischler81] es un algoritmo iterativo simple que se ha utilizado en muchas áreas. En el algoritmo 4 [Hartley03] se muestra la versión para extraer una línea a partir de un conjunto de puntos. En cada iteración el algoritmo: selecciona dos puntos de forma aleatoria y con ellos define una línea, evalúa el *consenso* o *soporte* de la nueva línea, después de probar varias veces, el algoritmo selecciona la línea que mayor consenso obtuvo.

Como se puede ver en las figuras 2.7(a) y 2.7(b), dada la línea $\Theta$ algunos puntos $p_i$ cumplen con

$$d(p_i, \Theta) < t$$

a estos puntos se le conoce como puntos típicos. El consenso de la línea $\Theta$ se define entonces como la cantidad de puntos típicos que ésta obtiene.

Intuitivamente, si en la selección inicial existe un outlier entonces la línea no tendrá mucho soporte. En la figura 2.7(a) se muestra un ejemplo de este caso y el consenso es de 5. En cambio, si en la selección de puntos sólo existen inliers, entonces el consenso es mucho mayor. En la figura. 2.7(b) se muestra un ejemplo de esta situación y el consenso es de 10.

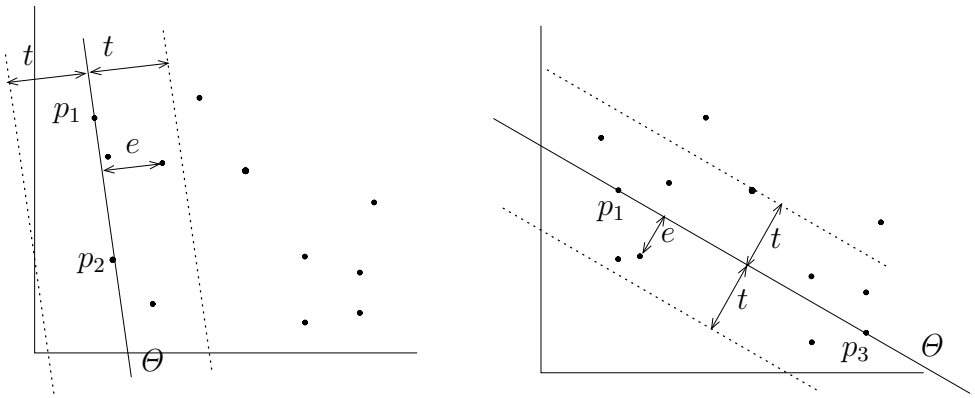
A continuación se describe la manera en que se calculan el umbral $t$ y el número de intentos $m$.

**Algoritmo 4** Algoritmo RANSAC para extraer una línea

**Entradas:** La imagen de rango $P = \{p_i | i = 1..n\}$ y un número de intentos $n$

**Salidas:** La línea $\widehat{\Theta}$ que representa a la mayor cantidad de puntos posibles de $P$

1. Hacer $j = 1$.
2. Seleccionar de forma aleatoria dos puntos de $P$. Estos dos puntos definen la $j$–ésima línea $\Theta^{(j)}$.
3. El consenso de $\Theta^{(j)}$ se mide por el número de puntos cuya distancia ortogonal a la línea es menor que el umbral previamente fijado.
4. Incrementar j.
5. Si $j < n$ ir al punto 2.
6. Volver a estimar el modelo de la línea con todos los inliers.




(a)línea con poco soporte (b)línea con mucho soporte

Figura 2.7: Consenso de RANSAC

### Cálculo del umbral

El umbral $t$ es un factor importante en el algoritmo RANSAC. Una buena selección de $t$ nos dará un resultado preciso. En la práctica se obtienen buenos resultados si se selecciona un umbral $t$ de forma empírica. Esto puede representar un problema por

ejemplo si se toma un $t$ demasiado grande se puede dar el caso que, sin importar el modelo de la línea todos los puntos de la muestra ajusten con la línea. El otro escenario es tomar un $t$ muy pequeño, con lo que la mayoría de los puntos no ajustarán con la línea. En cualquiera de los dos casos no hay forma de tomar una decisión con base al consenso.

Si se conoce la distribución de probabilidad que siguen los errores en las mediciones se puede estimar un umbral apropiado como se muestra en [Fischler81].

### Cálculo del número de intentos

Es muy importante poder calcular el número de intentos $m$ para que por una parte no se consuma tiempo en operaciones innecesarias y por otra parte se garantice que se ha obtenido la mejor línea a partir de los puntos. Para el cálculo del número de intentos $m$ se debe considerar que el número de muestras sea lo suficientemente alto para garantizar con una probabilidad $p$ (v.g. $p = 0.99$), que al menos uno de los conjuntos aleatorios seleccionado no tiene puntos atípicos.

Si se conoce la proporción de puntos atípicos $\epsilon$ del conjunto $P$, entonces la proporción de puntos típicos (puntos que no son atípicos) es $\omega = 1 - \epsilon$. Sea $n_S$ el número mínimo de puntos necesarios para evaluar un modelo, para el caso de una línea $n_S = 2$. La probabilidad de seleccionar $n_S$ inliers está dada por $\omega^{n_S}$ y la probabilidad de que exista al menos un outlier en $n_S$ es $1 - \omega^{n_S}$. Para alcanzar la probabilidad deseada $p$ serán necesarios $m$ intentos, por lo que se cumple

$$(1 - \omega^{n_S})^m = 1 - p \tag{2.28}$$

así que al despejar $m$, obtenemos el número de intentos buscado

$$\begin{aligned} m &= \frac{\log(1-p)}{\log\left(1-\omega^{n_S}\right)} \\ &= \frac{\log(1-p)}{\log\left(1-(1-\epsilon)^{n_S}\right)} \end{aligned} \tag{2.29}$$

Aunque es buena opción calcular el número de intentos $m$ con la ecuación 2.29 en muchas ocasiones se desconoce la proporción $\omega$, y en tal caso se puede calcular $m$ de forma adaptiva como lo muestra el algoritmo 5 [Hartley03].

Una forma de reducir el número de intentos es terminar una vez que el $n$–ésimo modelo $\Theta^{(n)}$ represente a la proporción de inliers $\omega$.

**Algoritmo 5** Algoritmo que usa el cálculo adaptivo de $m$

**Entradas:** La imagen de rango $P = \{p_i | i = 1..n\}$
**Salidas:** La mejor línea $\widehat{\Theta}$

1. Hacer $m = \infty$, $j = 0$
2. Seleccionar de forma aleatoria dos puntos de $P$. Estos dos puntos definen la $j$–ésima línea $\Theta^{(j)}$
3. El consenso $c$ para esta recta se mide por el número de puntos cuya distancia ortogonal a la línea es menor que el umbral previamente fijado.
4. Incrementar j
5. Calcular $\epsilon = 1 - (c/n)$
6. Calcular $m$ con la ecuación 2.29
7. Si $m > j$ ir al paso 2
8. Reestimar el modelo de la línea usando los inliers

### 2.8.2. Método de Selección Aleatoria Genérico

El método RANSAC forma parte de los llamados *métodos de Selección Aleatoria*. El algoritmo 6 [Myatt02b] describe de forma genérica el funcionamiento de estos métodos. Las diferentes versiones del algoritmo modifican esencialmente cuatro puntos clave:

1. *El mecanismo de selección de la muestra* $S \subset P$, decide la manera de elegir los elementos de S necesarios para generar un modelo. Con la muestra $S$ se obtiene un modelo. Aunque normalmente la selección de $S$ es aleatoria, se han propuesto mejorar el mecanismo de selección;

2. *La función de costo* $C(\cdot)$ que sirve para evaluar la bondad del modelo;

3. *La cantidad de intentos* $m$ que se repite la selección de la muestra $S$ y la obtención del modelo (Punto I), $m$ debe ser lo suficientemente grande para garantizar que el modelo final resultante sea óptimo; y

4. *El refinamiento*, cuyo objetivo es mejorar la estimación final alcanzada por el modelo de menor costo y usa todos los datos inliers.

**Algoritmo 6** Algoritmo Genérico de los estimadores de selección aleatoria

**Entradas:** Un conjunto de datos $P = \{p_i | i = 1..n\}$
**Salidas:** Un modelo $\widehat{\Theta}$ que represente los datos $P$

1. Tomar una muestra de datos $S \subset P$, donde $\|S\|$ es el número mínimo de datos para generar un modelo. Sea $\Theta^{(i)}$ el modelo obtenido de $S$,
2. Evaluar el desempeño de $\Theta^{(i)}$ en $P$ con una función de costo $C\left(\Theta^{(i)}\right)$,
3. Repetir $m$ veces los pasos 1 y 2,
4. Seleccionar el modelo $\dot{\Theta}$ que mejor fue evaluado por la función de costo $C$,
5. Refinar $\dot{\Theta}$ para obtener la estimación final $\widehat{\Theta}$

El algoritmo RANSAC trata de maximizar el número de inliers, esto puede ser expresado de forma equivalente mediante el algoritmo genérico 6 que minimiza la siguiente función de costo:

$$C = \sum_{i=1}^{N} \rho(e_i^2) \tag{2.30}$$

donde

$$\rho(e_i) = \begin{cases} 0 & |e| < t \\ 1 & |e| \geq t \end{cases} \tag{2.31}$$

y la mejor recta será la que tenga un costo mínimo. Es importante hacer notar que minimizar la función de costo expresada por 2.30 es equivalente a maximizar el número de inliers.

Algunas variantes del algoritmo de selección aleatoria se muestran en las siguientes secciones. Se hace énfasis en la función de costo que usa cada variante.

### 2.8.3. MSAC

Como se ha expuesto anteriormente una de las dificultades de RANSAC es encontrar un umbral $t$ adecuado. Para mejorar la situación y favorecer ajustes con puntos cercanos al modelo se propone en [Torr00] el uso de la siguiente función de ponderación

$$C = \sum_{i=1}^{N} \rho_2(e_i^2) \tag{2.32}$$

donde

$$\rho_2(e_i^2) = \begin{cases} e^2 & e^2 < t^2 \\ \text{constante} & e^2 \geq t^2 \end{cases} \tag{2.33}$$

Éste es un simple estimador–M robusto llamado *M-Estimator SAmple Consensus* (MSAC) [Torr98] que penaliza los puntos inliers de acuerdo con el error involucrado, es decir, inliers con un error grande tienen mayor penalización que aquellos con un error pequeño. Los estimadores expuestos en las siguientes secciones hacen énfasis en las distribuciones que siguen los inliers pero sobre todo en las distribuciones que siguen los puntos atípicos.

### 2.8.4. MLESAC

El método *Maximum Likelihood SAmple Consensus* (MLESAC) es una adaptación de RANSAC desarrollado en [Torr00]. Se sustituye la hipótesis de que los errores provienen de una sola distribución por otra en que los errores provienen de una mezcla. Los datos pueden ser típicos o atípicos, los puntos típicos siguen una distribución normal y los puntos atípicos siguen una distribución uniforme (en una ventana de tamaño v). La distribución resultante se puede expresar como

$$f = \gamma \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{e^2}{2\sigma^2}) + (1+\gamma)\frac{1}{v} \tag{2.34}$$

donde $\gamma$ es la proporción de inliers. La técnica maximiza la probabilidad de que los puntos sean inliers. MLESAC ha mostrado ser superior a RANSAC en ciertas tareas de estimación [Torr00].

### 2.8.5. MINIPRAN

El método MINIPRAN, propuesto en [Stewart93] y [Stewart95], usa un enfoque contrario a MLESAC. En lugar de maximizar la probabilidad de que un conjunto de puntos sean inliers como MLESAC, MINIPRAN intenta minimizar la probabilidad de que los puntos sean puntos atípicos. En el nivel básico se usa el mismo mecanismo de RANSAC para agrupar los puntos en típicos–atípicos, el conjunto de inliers son aquellos que estén a menos de un umbral $t$ del modelo hipotético. Se usa la distribución binomial para los puntos atípicos en lugar de una distribución uniforme como en MLSAC. La ventaja de usar este esquema es que se pueden tener muy buenos resultados a pesar que la distribución de inliers se desconozca.

| *Método* | *Ventajas* | *Desventajas* |
|---|---|---|
| LS, TLS, WTLS | · Rápidos<br>· Cálculo Directo | · Sensibles a puntos atípicos |
| LMS, LQS | · Tolerantes a puntos atípicos | · Complejidad en tiempo<br>· Fallan si la proporción de puntos atípicos excede la proporción preestablecida |
| Estimadores– M | · Tolerantes a puntos atípicos | · Dependen de la estimación inicial<br>· Pueden ser imprecisos |
| HT | · Tolerante a puntos atípicos<br>· No se requiere conocer la proporción de puntos atípicos | · Complejidad en tiempo<br>· Discretización |
| RANSAC | · Tolerante a puntos atípicos<br>· No se requiere conocer la proporción de puntos atípicos · obtención de curvas más precisas | · Complejidad en tiempo<br>· Se debe intentar lo suficiente para garantizar que no se seleccionaron puntos atípicos |

Tabla 2.4: Resumen de los métodos de ajuste de líneas

## 2.9. Discusión

En este capítulo se revisaron los métodos robustos para la extracción de una línea más usados en la literatura. En la tabla 2.4 se resumen las ventajas y desventajas más relevantes de los métodos de ajuste discutidos. Como se puede observar, se puede aprovechar el cálculo directo de los estimadores de mínimos cuadrados siempre y cuando se garantice que en el conjunto de puntos no existen puntos atípicos. Si no se puede garantizar que en la muestra no existen puntos atípicos entonces se debe usar un método robusto.

Los métodos robustos como RANSAC, HT y LQS permiten obtener buenos resultados con muy pocos inliers (incluso menor al 50 %). Esto representa en general una ventaja aunque es obligada una fase de postprocesamiento para seleccionar la mejor línea en el caso de que se hayan encontrado más de una.

Los estimadores–M robustos pueden ser vistos como generalizaciones de LS

[Stewart99] y tienen la característica de ser más resistentes a puntos atípicos que LS; a pesar de esto, en ciertas situaciones aún es posible que el método falle con un sólo outlier, por lo que se mantiene el punto de ruptura de cero.

Son destacables los beneficios de los métodos de selección aleatoria: son tolerantes a puntos atípicos, no requieren conocer la proporción de puntos atípicos, arrojan resultados precisos, y además se pueden adaptar a diferentes problemas más fácilmente que otros métodos (v.g. LMS) [Meer00]. Al elegir adecuadamente tanto la función de costo como la hipótesis de selección [Myatt02b] se puede mejorar el desempeño de los métodos de selección aleatoria.

# Capítulo 3

# Extracción de un conjunto de líneas

Para los robots móviles que se desempeñan en ambientes interiores es útil extraer líneas del ambiente ya que muchos objetos tienen contornos lineales (muros, puertas, ventanas, cajas, mesas, etc). Esta es la razón principal por la que este capítulo se dedica a la extracción de un conjunto de líneas a partir de una imagen de rango tomada por un láser.

La organización del capítulo es la siguiente: en la sección 3.2 se define el problema formalmente, en la sección 3.3 introducen los algoritmos de agrupamiento y su relación con la solución del problema, en la sección 3.4 se discute una forma de simplificar el problema a través del uso de detectores de puntos de ruptura, en la sección 3.5 se describen varios algoritmos de extracción que se reportan frecuentemente en la literatura y por último en se hace una comparación entre los métodos descritos.

## 3.1. Introducción

Dada la naturaleza de los sensores, las lecturas directas que se obtienen de ellos tienen cierto grado de confiabilidad. Para aumentar el grado de confiabilidad se pueden

combinar diferentes lecturas del mismo sensor o posiblemente de un grupo de sensores. Eso es precisamente lo que ocurre cuando se extraen *características* o *marcas* del ambiente. El uso de marcas en robótica móvil es una parte medular para muchos algoritmos que resuelven problemas como el de *resolver simultáneamente Localización y generación de mapas* (SLAM, del inglés *Simultaneous Localization And Mapping*) o el de *Detección y seguimiento de objetos en movimiento* (DATMO, del inglés *Detection And Tracking of Moving Objects*). En este capítulo estamos interesado en las líneas ya que son marcas que abundan en los ambientes interiores del tipo oficina.

El proceso de extraer un conjunto de líneas del entorno a partir de las imágenes de rango que se obtienen del láser puede parecer muy fácil a simple vista. Esto puede resultar engañoso, especialmente si se considera que para definir cada línea sólo se requieren dos puntos. En la realidad existen algunas dificultades para encontrar el mejor conjunto de líneas debido a que las mediciones del láser tienen una cierta cantidad de ruido, se desconoce la cantidad de líneas de un entorno y qué mediciones pertenecen a qué línea.

Algunos enfoques consideran la forma secuencial de operación el láser: en la $i$-ésima lectura el sensor mide la distancia a la que se encuentra el objeto más cercano con un ángulo determinado $\alpha_i$, lo que produce una medición en formato $(r_i, \alpha_i)$. Cada imagen de rango es una secuencia de $n$ lecturas, se parte de un ángulo inicial $\alpha_1$ hasta un ángulo final $\alpha_n$. Cada medición se realiza al incrementar el ángulo en $\Delta\alpha$ respecto de la medición anterior, es decir, $\alpha_{i+1} = \alpha_i + \Delta\alpha$. Esta forma de obtener las imágenes por el láser facilita la extracción de las características del entorno al considerar que *dos lecturas adyacentes $p_i$, $p_{i+1}$ tienen un alta probabilidad de pertenecer a la misma característica.*

El problema se puede abordar con algoritmos de agrupamiento basados en la vecindad entre puntos [Duda76]. Una vez agrupados los puntos se puede aplicar algún método de regresión, como los estudiados en el capítulo anterior. En la práctica este enfoque consume una gran cantidad de tiempo ya que se usan esquemas de agrupa-

miento muy genéricos que desaprovechan la **secuencialidad** de las lecturas que las imágenes de rango ofrecen [Siegwart04].

## 3.2. Definición del Problema

El problema que se estudia en este capítulo se define de la siguiente manera: Dada una imagen de rango $S$ se debe recuperar el Mapa Geométrico de Líneas (MGL).

Recordando que una *Imagen de Rango en 2D* es un conjunto de puntos en formato polar $S = \{(r_i, \alpha_i)|i = 1 \dots n\}$, donde $\alpha_{i+1} = \alpha_i + \Delta\alpha$; o su equivalente en formato cartesiano $P = \{(x_i, y_i)|i = 1 \dots n\}$, donde $(x_i, y_i)$ es un punto en $\mathbb{R}^2$ y además se cumple que:

$$y_i = r_i \sin(\alpha_i) \tag{3.1}$$

$$x_i = r_i \cos(\alpha_i) \tag{3.2}$$

.

El mapa geométrico de líneas (MGL) es el conjunto de líneas que mejor ajustan a $S$. El MGL se representará matemáticamente por

$$\mathcal{M} = \{\Theta_i|i = 1 \dots n_{\mathcal{M}}\} \tag{3.3}$$

donde $\Theta_i$ son los parámetros de la $i$–ésima línea y $n_{\mathcal{M}}$ es número de líneas extraído.

## 3.3. Algoritmos de agrupamiento

Como se anticipó en la introducción del capítulo, una primera aproximación para resolver el problema planteado en 3.2 es usar algún método de agrupamiento. Los algoritmos de Agrupamiento (*Clustering*) pueden ser definidos como aquellos algoritmos que organizan un conjunto de objetos en grupos o clases que tengan propiedades

similares. Los métodos de clustering han sido propuestos en tres diferentes áreas: estadística, minería de datos y reconocimiento de patrones. Aquí se hace una breve introducción de algunas técnicas de agrupamiento.

Los algoritmos de agrupamiento, al igual que los estimadores robustos, intentan minimizar una función de costo dada. Sin embargo las hipótesis que usan son más sofisticadas ya que se basan en la estructura de los datos [Myatt02b].

### 3.3.1. Algoritmos de k grupos

Los algoritmos de k grupos (*k–clustering*) tratan de organizar el conjunto inicial de datos $S$ en $k$ grupos, donde el valor de $k$ se conoce. El objetivo de estos algoritmos es lograr una partición de $S = \cup_{l=1}^{k} S_l$ que minimice alguna función de costo de $\sum_{l=1}^{k} C(S_l)$. El costo de un grupo $S_l$ se puede evaluar con la función muy conocida que usa la distancia euclidiana $d$

$$C(S_l) = \sum_{i=1}^{|S_l|} \sum_{j=1}^{|S_l|} \left(d\left(p_i, p_j\right)\right)^2 \tag{3.4}$$

Es decir, el cuadrado de las distancias entre cada punto dentro del agrupamiento. El algoritmo más conocido para *k–clustering* es el algoritmo *k–means* [MacQueen67] la cual optimiza la función

$$C(S_l) = \sum_{i=1}^{|S_l|} d\left(\overline{p}, p_i\right) \tag{3.5}$$

donde $\overline{p}$ representa la media del grupo. Los algoritmos de $k$–clustering producen agrupamientos esféricos los cuales son inapropiados para la extracción de la mayoría de características (v.g. líneas) de una imagen de rango.

### 3.3.2. Algoritmos de agrupamiento jerárquico

Los algoritmos de agrupamiento jerárquico son usados cuando la entrada no puede ser particionada en grupos discretos y en lugar de producir una partición se busca un árbol de grupos [Myatt02b]. En la raíz del árbol se encuentra el conjunto $S$ y los hijos de cualquier nodo representan diferentes formas de dividir el conjunto en subconjuntos. A diferencia de k–clustering, se puede decidir al final el número de grupos y a que grupo pertenece cada punto.

Existen dos grandes clases de técnicas de agrupamiento jerárquico, *acumulativo* y *divisivo*, que son diametralmente opuestas. En las técnicas acumulativas se comienza con grupos lo mas pequeños posibles, se mezclan grupos similares entre sí y se forman grupos más grandes cada vez. Por el contrario las técnicas divisivas comienzan con el conjunto $S$ y lo separan según sus diferencias hasta llegar a conjuntos mínimos.

Al igual que los algoritmos de k–clustering, la principal desventaja de los algoritmos de agrupamiento jerárquico es que no se aprovecha el orden en el que se obtiene la imágenes de rango. Para extraer marcas, los algoritmos de clustering se usan en conjunto con otras técnicas como en [Baltzakis02, Myatt02a].

## 3.4. Puntos de ruptura

En esta sección, a diferencia de las técnicas de agrupamiento de la sección anterior, se aprovecha la secuencialidad de las lecturas del láser. Una técnica para simplificar el problema de generación de MGL es usar un algoritmo de agrupamiento cuyo objetivo es que puntos adyacentes de la imagen de Rango $S$ que están muy cercanos entre si queden agrupados. A esta técnica se le llama *detección de puntos de ruptura* ó *algoritmos de segmentación*. Un *punto de ruptura* $p_r \in S$, es aquel punto donde se pierde sustancialmente la continuidad de $S$, como ejemplo en la figura 3.1 los puntos de ruptura de esta secuencia son representados por las cruces.

Para encontrar los puntos de ruptura es común el uso de distancias euclidianas

entre dos puntos sucesivos. Sin embargo, se puede utilizar cualquier distancia $D_{i,j}$ del punto $i$ al punto $j$ siempre que cumpla:

$$D_{1,2} = D_{2,1} \tag{3.6}$$

$$D_{1,3} + D_{3_2} \geq D_{1,2} \tag{3.7}$$

$$D_{1,1} = 0 \tag{3.8}$$

Los puntos de ruptura son de gran importancia para diferentes métodos de generación de MGL. [Premebida05] los divide en *métodos basados en la distancia entre puntos* (PDB, del inglés *Point Distance – based methods*) y *métodos basados en el filtro de Kalman* (KFB, del inglés *Kalman Filter – based methods*).

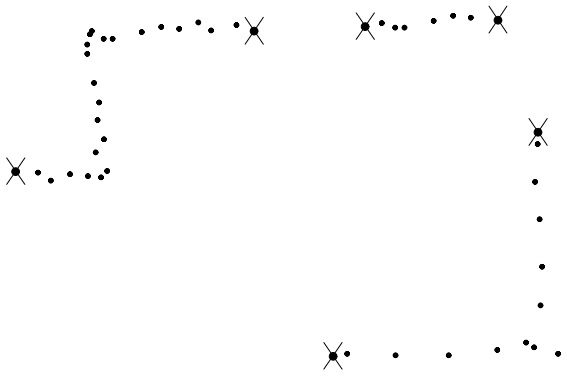


Figura 3.1: Detector de Puntos de Ruptura

### 3.4.1. Detector de Puntos de Ruptura basado en distancias

Existen varios *métodos basados en la distancia entre puntos*, en esta sección se describe el *Detector Simple de Puntos de Ruptura* (SBD) y el *Detector Adaptivo de Puntos de Ruptura* (ABD).

### SBD

El *Detector Simple de Puntos de Ruptura* (*Simple Breakpoint Detector*), calcula de forma secuencial la distancia euclidiana entre los puntos adyacentes $p_i$ y $p_{i+1}$, comenzando desde $i = 0$ hasta $i = n - 1$. Si la distancia es mayor a determinado umbral $t_b$, es decir:

$$d(p_i, p_{i+1}) > t_b \tag{3.9}$$

entonces $p_i$ se considera un punto de ruptura. Para calcular la distancia entre dos puntos se puede usar coordenadas cartesianas o polares. En el primer caso la distancia se calcula de la siguiente forma:

$$d(p_i, p_{i+1}) = \|p_{i+1} - p_i\| = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{3.10}$$

mientras que con coordenadas polares se puede usar:

$$d(p_i, p_{i+1}) = \|p_{i+1} - p_i\| = \sqrt{r_{i+1}^2 + r_i^2 - 2r_{i+1}r_i \cos \Delta\alpha} \tag{3.11}$$

En este método el valor de $t_b$ se mantiene constante aunque, como se verá en la siguiente sección también se puede usar un valor de $t_b$ que cambie para lograr mejores resultados.

### ABD

El *detector Adaptivo de Puntos de Ruptura* (ABD, del inglés *Adaptive BreakPoint Detector*) ajusta el umbral $t_b$ en la ecuación 3.9. Ésto es necesario ya que, por un lado la densidad de los puntos leídos no es uniforme ya que depende del ángulo de incidencia del rayo respecto del segmento; y por otro, el error de la $i$–ésima lectura de algunos sensores está en función de $r_i$.

En [Castro04] se usa la siguiente fórmula para calcular $t_b$

$$t_b = k_0 + k_1 \,\text{mín}\{r_i, r_{i+1}\} \tag{3.12}$$

donde

$$k_1 = \sqrt{2(1 - \cos \Delta\alpha)} \tag{3.13}$$

y la constante $k_0$ permite un ajuste del algoritmo según el ruido existente.

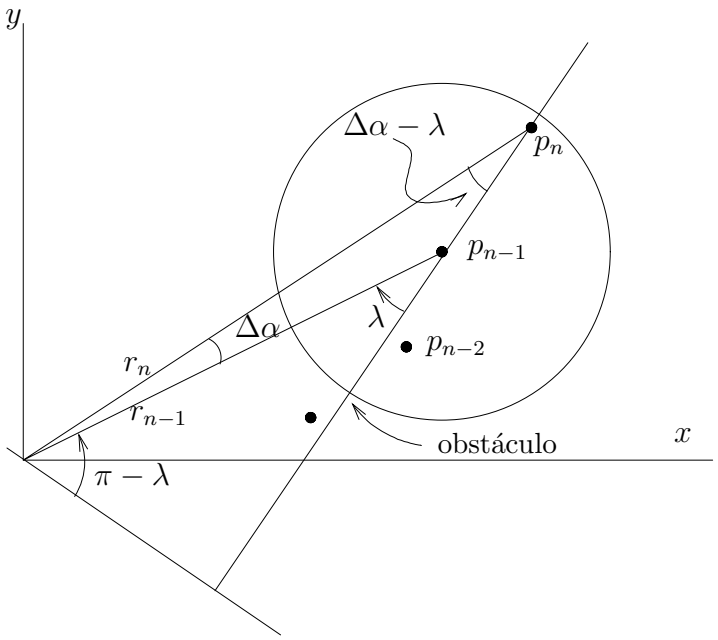



Figura 3.2: Detector de puntos de Ruptura Adaptivo. Los puntos $p_n$ y $p_{n-1}$ se suponen que pertenecen al mismo obstáculo. El radio del circulo indica la máxima distancia permitida entre los puntos

En [Borges04] se propone un método que considera el ángulo de incidencia del rayo sobre el objeto. Si se usa la ley de senos con la información que se muestra en la figura 3.2, tenemos

$$\|p_n - p_{n-1}\| = r_{n-1} \frac{\sin \Delta\alpha}{\sin (\Delta\alpha - \lambda)} \tag{3.14}$$

donde $\Delta\alpha$ es la resolución angular del láser y $\lambda$ es el ángulo de incidencia del rayo $r_{n-1}$ sobre el obstáculo; sin embargo, el ángulo de incidencia $\lambda$ en éste caso se desconoce.

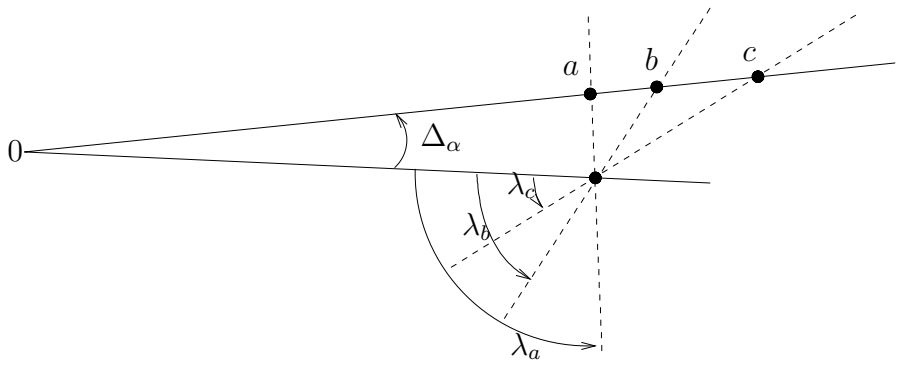



Figura 3.3: Diferentes ángulos de incidencia $\lambda$

.

En la figura 3.3 se muestran diferentes casos del ángulo de incidencia $\lambda \leq \pi$, donde se observa que la posible distancia entre puntos se incrementa mientras disminuye el ángulo $\lambda$. Por este motivo nos interesa el ángulo más pequeño que sea posible leer, para el caso de la figura 3.3 se esquematiza con $\lambda_c$. Para $\lambda \geq \pi$ ocurre una situación semejante; de tal manera que $\lambda$ en la ecuación 3.14 debe tomar el valor del peor ángulo de incidencia que pueda leer el láser, el cual se puede determinar empíricamente. Para considerar el ruido en las lecturas se toma la distancia mínima $t_b$ permitida como

$$t_b = \|p_n - p_{n-1}\| + 3\sigma_r \tag{3.15}$$

donde $\sigma_r$ es la desviación estándar del error de la distancia reportada por el sensor.

### 3.4.2. Detector de Puntos de Ruptura basado en el Filtro Kalman

Un detector de punto de ruptura basado en el Filtro Kalman (KF) se discute en [Castellanos96] donde el problema de extracción de líneas se aborda como un problema de seguimiento. Es decir, se trata de predecir la posición siguiente al considerar los puntos leídos con anterioridad. Si tal predicción falla, entonces se considera que se ha encontrado un nuevo punto de ruptura.

En este detector se usa un modelo discretizado del sistema para describir el comportamiento dinámico de las mediciones de rango $r_n$. Si consideramos el estado del sistema como $\boldsymbol{x}_n = \left(r_n \ \frac{\mathrm{d}r_n}{\mathrm{d}\alpha}\right)^t$ entonces el modelo es:

$$\boldsymbol{x}_n = \boldsymbol{A}\boldsymbol{x}_{n-1} + \boldsymbol{w}_n \tag{3.16}$$

$$\boldsymbol{z_n} = \boldsymbol{C}\boldsymbol{x}_n + \boldsymbol{v}_n \tag{3.17}$$

donde $z_n$ es la variable de medición, y $\boldsymbol{w}_n$ y $\boldsymbol{v}_n$ es ruido Gaussiano con media 0 y covarianza $\boldsymbol{Q}_n$ y $\boldsymbol{R}_n$ respectivamente. Las matrices $\boldsymbol{A}$ y $\boldsymbol{C}$ corresponden a

$$\boldsymbol{A} = \begin{pmatrix} 1 & \Delta\alpha \\ 0 & 1 \end{pmatrix} \qquad \boldsymbol{C} = (1 \ \ 0) \tag{3.18}$$

Como se discute en [Borges04] el modelo 3.16 es el modelo más simple que puede ser usado sin conocimiento previo respecto de la trayectoria.

## 3.5. Métodos de búsqueda de líneas

En esta sección se analiza la parte esencial de los métodos de extracción de líneas a partir de una imagen de rango. Dichos métodos son independientes del uso previo de un detector de puntos de ruptura. En la siguiente exposición se considera que se cuenta con la totalidad de la imagen de rango y que dicha imagen se obtuvo desde un punto estático. Existen, sin embargo, algunos mecanismos (p. ej. los basados en el filtro de Kalman) donde se puede realizar la detección de líneas en tiempo real, es decir, al mismo tiempo que se obtienen los datos de la imagen. La premisa de tener toda la imagen tomada desde un punto estático no constituye un problema dada la velocidad con la que se leen los datos.

### 3.5.1. Line Tracking

El algoritmo de *Seguimiento de Líneas* (LT, del inglés *Line Tracking*) [Duda76] también es conocido como *Algoritmo Incremental* [Nguyen05]. El algoritmo es muy sencillo y rápido, se describe en el algoritmo 7 y en la figura 3.4 se ilustra el procedimiento.

**Algoritmo 7** Algoritmo Line Tracking

**Entradas:** Una imagen de rango $S = \{p_i | i = 1 \dots n\}$, un umbral $t$, un conjunto de líneas $\mathcal{M} = \emptyset$ y $j = 1$

**Salidas:** Un mapa de líneas $\mathcal{M}$

1. Si $j + 1 > n$ terminar; si no, crear un conjunto de puntos $Q$ y agregar los puntos $p_j$ y $p_{j+1}$ e incrementar $j$.
2. Calcular los parámetros $\Theta$ de la línea usando el conjunto de puntos $Q$.
3. Si la distancia del siguiente punto a la línea es menor al umbral, $d(p_{j+1}, \Theta < t$, agregar $p_j$ a $Q$, incrementar $j$ y regresar al paso 1
4. De otra manera, agregar la línea $\Theta$ al conjunto $\mathcal{M}$ y regresar al paso 2.

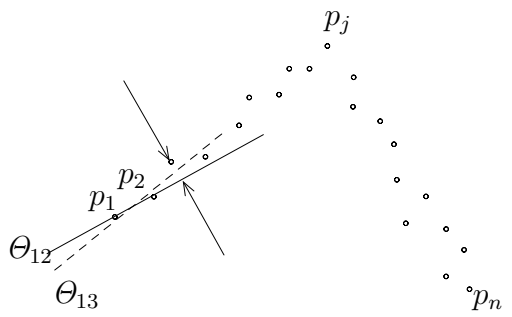



Figura 3.4: Line Tracking

### 3.5.2. Iterative End Point Fit

El algoritmo de *Ajuste Iterativo de Puntos Finales* (IEPF, del inglés *Iterative End Point Fit*) [Duda76] se ilustra en la figura 3.5. El algoritmo divide recursivamente el conjunto de puntos $S$ en grupos más pequeños como lo describe el algoritmo 8. El punto de división del conjunto $S_j$ es aquel que está más alejado a una línea de referencia $\Theta_j$. Como se puede observar, este algoritmo obtiene la línea $\Theta_j$ al unir el punto inicial con el punto final. Si en lugar de hacer ésto, la línea $\Theta_j$ se obtiene al ajustar una línea al conjunto de puntos $S_j$ entonces al método se le conoce como *Ajuste de Línea Recursivo* (RLF, del inglés *Recursive Line Fitting*) [Gätcher05].

**Algoritmo 8** Algoritmo IEPF

**Entradas:** Una imagen de rango $S = \{p_i | i = 1 \dots n\}$ y un umbral $t$

**Salidas:** Un mapa de líneas $\mathcal{M}$

1. Crear una lista $\mathcal{L}$ e introducir $S_1 = S$
2. Repetir mientras exista un conjunto $S_j = \{p_j | j = 1 \dots n_j\}$ en la lista $\mathcal{L}$ que no haya sido revisado:
   - *a*) Encontrar la línea $\Theta_j$ que pasa por el punto inicial y el punto final del conjunto $S_j$.
   - *b*) Encontrar el punto $p_k$ que tenga el mayor error ortogonal $e_{\perp max}$ a la línea $\Theta_j$.
   - *c*) Si $e_{\perp max} > t$ el conjunto de puntos $S_j$ se divide en dos conjuntos: $S_{j0} = \{p_j | j = 1 \dots k-1\}$ y $S_{j1} = \{p_j | j = k \dots n_j\}$. En la lista $\mathcal{L}$ se reemplaza $S_j$ por los conjuntos no revisados $S_{j0}$ y $S_{j1}$.
   - *d*) Si no, marcar $S_j$ como revisado

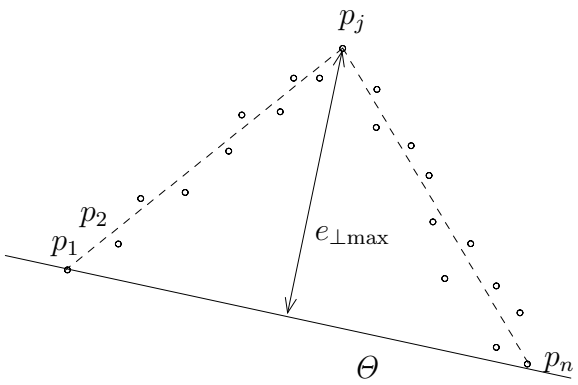



Figura 3.5: Iterative End Point Fit(IEPF)

### 3.5.3. Split–Merge Split–Merge

El método *Split–Merge Split–Merge* (SMSM) [Zezhong03] es una adaptación del método IEPF, y está está dividido en cuatro pasos como su nombre lo indica. En la primera fase de división (*Split*) se usa un detector de puntos de ruptura. En la primera fase de mezcla (*Merge*) se calcula la distancia entre el último punto de un grupo al primer punto del grupo siguiente, si la distancia es menor que un determinado umbral entonces se combinan los grupos. En la figura 3.6 se muestra un ejemplo de la primer fase de mezcla, los grupos $g_1$ y $g_2$ se mezclarán porque $d_{12} < t$ donde $t$ es un umbral preestablecido. En cambio el grupo $g_3$ no se mezcla ya que $d_{23} > t$.

En la segunda fase de división se usa el método IEPF estudiado en la sección anterior. Por último, en la segunda fase de mezcla, se combinan aquellos segmentos de línea que son similares entre sí como se describe el apéndice B.

### 3.5.4. Búsqueda de líneas con HT

El uso de la transformada de Hough descrita en la sección 2.6 es muy popular para la extracción de MGL [Giesler98, Alempijevic04, Nguyen05, Sack03] a partir de una imagen de rango $S$. La HT es una técnica que usa la información global que

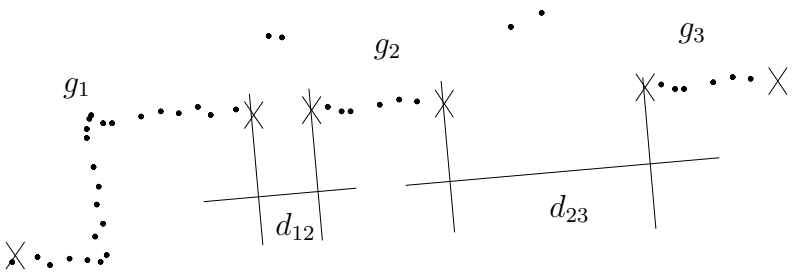



Figura 3.6: Primer fase de mezcla de SMSM

.

encuentra en una imagen de rango $S$. Esto obliga a la realización de una fase de post–procesamiento para determinar las líneas con mayor votación y los segmentos de línea involucrados.

Además de las variantes genéricas de la HT que mejoran la velocidad y uso de memoria, se han reportado en la literatura técnicas aplicables específicamente a las imágenes de rango. Por ejemplo en [Alempijevic04] se aprovecha un procesamiento directo de los puntos de la imagen de rango $S$ para detectar los parámetros de las líneas más probables y dichos parámetros sirven para focalizar la búsqueda de líneas en la HT con lo que se encuentra el MGL final. El esquema de votación de la HT se puede adaptar según la aplicación, por ejemplo, en [Gillies92] se incorpora una fase de post–procesamiento donde se incluyen criterios de votación de acuerdo a criterios perceptuales. Los criterios perceptuales derivan de los trabajos de la corriente psicológica del Gestalt que caracteriza el sistema de visión humana. Estos criterios incluyen similaridad de color e intensidad, continuidad, etc.

### 3.5.5. Búsqueda de líneas con EM

El algoritmo *Expectation–Maximization* (EM) [Dempster77] es una técnica iterativa de uso general que se usa para encontrar la máxima verosimilitud en problemas con datos incompletos. La idea fundamental es asociar al problema de datos incom-

pletos uno de datos completos para el cual la estimación de máxima verosimilitud es más fácil de realizar [McLachlan97]. En cada iteración el algoritmo EM consiste de dos pasos, el paso–E y el paso–M. En el paso–E los datos faltantes son estimados a partir de los datos observados y la estimación actual de los parámetros del modelo faltantes. En el paso–M, la función de verosimilitud se maximiza.

El algoritmo EM aplicado al ajuste de líneas es conocido en estadística como el procedimiento Healy–Westmacott, y surge 7 años antes que el algoritmo EM genérico [Latecki06]. Los siguientes dos pasos se ejecutan hasta convergencia, el algoritmo garantiza que converge a algún mínimo local. La entrada es un conjunto de puntos en el plano y un conjunto de líneas inicial [Latecki06]:

**paso–E (Expectación)** Dado el conjunto de líneas actual, estimar las probabilidades de pertenencia de cada punto a las líneas. Las probabilidades se calculan en base a la distancia de cada punto a las líneas.

**paso–M (Maximización)** Dadas las probabilidades calculadas en el paso–E, se calculan los nuevos parámetros de las líneas usando una regresión ponderada por las probabilidades.

Una variante conocida como *fuzzy–K means clustering* [Duda76] [Sack03] asume que el error de las mediciones sigue una distribución gaussiana. Mientras que en [Liu01] se usa una mezcla de distribuciones: una distribución constante para outliers y otra normal para inliers.

Un problema crucial del método EM es el número de líneas $m$ del MGL. Algunas técnicas varían el número de líneas, comienzan con una cantidad grande y desechan aquellas líneas que se queden sin puntos.

## 3.6. Resumen

En este capítulo se han descrito varios métodos comúnmente reportados en la literatura para encontrar un MGL a partir de una imagen de rango. Se inició con una breve descripción de los métodos de agrupamiento de puntos; de los cuales se estudiaron más a fondo los llamados *detectores de puntos de ruptura* que son algoritmos que realizan la agrupación en función tanto del orden en que los puntos se encuentran dentro de la imagen de rango como la distancia que existe entre puntos adyacentes. Los detectores de puntos de ruptura son usados en muchos métodos como un mecanismo para simplificar la extracción de líneas.

Los métodos para la extracción de líneas que se estudiaron aquí son LT, IEPF, HT y EM. De ellos, LT e IEPF se usan frecuentemente debido a su simplicidad y rapidez, su principal inconveniente es que solo usan información local, con lo que obtienen resultados poco precisos. Por otra parte los métodos basados en HT o EM obtienen sus resultados usando toda la información dentro de la imagen de rango, con lo que se obtienen mapas de líneas más precisos. Su principal inconveniente es la cantidad de tiempo que invierten para realizar la extracción. Ésto dificulta su aplicación en el área de robótica móvil.

| Algoritmo | velocidad [ms] | número de líneas | precisión | | Asertividad % |
|---|---|---|---|---|---|
| | | | $\sigma_{\Delta_r}$ [cm] | $\sigma_{\Delta_\alpha}$ [°] | |
| SMSM | 0.7 | 641 | 1.95 | 0.74 | 86.0 |
| LT | 3.0 | 561 | 2.04 | 0.72 | 77.8 |
| LT + Clus. | 1.6 | 567 | 2.04 | 0.72 | 79.2 |
| RANSAC | 34.5 | 749 | 1.68 | 0.77 | 75.6 |
| RANSAC + Clus. | 10.8 | 547 | 1.37 | 0.70 | 70.7 |
| HT | 125 | 825 | 1.63 | 0.76 | 82.0 |
| HT + Clus. | 111.1 | 600 | 1.51 | 0.67 | 79.5 |
| EM | 1666.7 | 1153 | 2.09 | 0.97 | 78.6 |
| EM + Clus. | 1428.6 | 709 | 1.58 | 0.73 | 80.3 |

Tabla 3.1: Comparación de los algoritmos estudiados para la extracción de MGL (adaptado de [Nguyen05])

Como se puede observar en la tabla 3.1 [Nguyen05] el algoritmo *SMSM* es el más rápido, mientras que HT y EM son los que más tiempo invierten para la extracción. Por otra parte, el algoritmo *SMSM* es el que tiene mayor índice de asertividad (proporción de líneas correctamente obtenidas).

En el siguiente capítulo se propone un esquema para la extracción de un MGL que busca mejorar el desempeño al combinar los enfoques local y global.

# Capítulo 4

# Métodos Propuestos

En el capítulo 2 se abordaron algunos métodos para resolver el problema de extraer una línea a partir de un conjunto de puntos, en específico los *métodos de selección aleatoria* formulan hipótesis de líneas usando la mínima cantidad de puntos y después eligen el modelo que mejor fue evaluado por una función determinada. Los algoritmos que se proponen en este capítulo están inspirados en los algoritmos de Selección Aleatoria y además tratan de aprovechar la secuencialidad de los datos obtenidos con el telémetro láser.

De forma resumida, el problema que tratamos de resolver es el de obtener el modelo $\dot{\mathcal{M}} = \{\Theta_j | j = 1 \dots n_{\mathcal{M}}\}$ de $n_{\mathcal{M}}$ líneas que mejor represente la imagen de rango $P = \{(x_i, y_i) \, | i = 1 \dots n\}$. El objetivo de este capítulo es exponer dos nuevos algoritmos que resuelven el problema de manera satisfactoria a pesar de la existencia de una gran cantidad de ruido en la imagen de rango.

En la sección 4.1 se propone el *algoritmo de Consenso aleatorio en una ventana* (WSAC, del inglés Window SAmple Consensus). Este algoritmo busca iterativamente líneas de forma local, si la búsqueda local tuvo éxito entonces la línea encontrada se integra al ML incorporando todos aquellos puntos de la imagen que cumplan las condiciones para pertenecer a esa línea.

El *algoritmo de Consenso aleatorio en una ventana con evaluación global*(WSAC–

GE, del inglés Window SAmple Consensus–General Evaluation) se presenta en la sección 4.2. Este algoritmo también extrae líneas de forma local pero a diferencia del método anterior se realiza un procedimiento más elaborado para decidir qué líneas deben incluirse en el ML resultante y cuáles puntos están asociados a cada línea.

Un mecanismo importante que usan los métodos WSAC y WSAC–GE es el de obtener un conjunto de segmentos a partir de un conjunto de puntos asociado a una línea. En la siguiente sección se describe este mecanismo en la siguiente sección.

**Detector de segmentos de una línea SLD**

En ocasiones tenemos una línea $\Theta$ que tiene asociado un conjunto de puntos $S_\Theta$, y se desea determinar los segmentos de la línea. El problema de encontrar el conjunto de segmentos es equivalente a encontrar los puntos de ruptura del conjunto de puntos $S_\Theta \subset S$ asociados a una línea $\Theta$, de tal manera que:

$$S_\Theta = \{p_a | d(p_a, \Theta) < t\}$$

como lo muestra la figura 4.1. El objetivo es encontrar aquellos puntos elementos de $S_\Theta$ en donde se pierde la continuidad. En la figura 4.1 los puntos de ruptura están indicados por las cruces.

Para resolver este problema se pueden aplicar técnicas similares a las usadas en los detectores descritos en la sección 3.4 de la página 51. El método aquí propuesto considera que si dos puntos están muy cercanos entre sí según el orden del rastreo entonces no se puede asegurar la discontinuidad. Es decir, si $p_i, p_j \in S_\Theta$ y $j > i$ entonces habrá $s = j - i - 1$ lecturas entre los puntos $p_i$ y $p_j$. Por lo que si

$$s \leq k_s$$

donde $k_s$ es el máximo de lecturas permitido entre los puntos, se puede considerar que no existe un punto de ruptura. Por último, si entre los puntos $p_i$ y $p_j$ existen más de $k_s$ lecturas entonces se considera a $p_i$ como un punto de ruptura si

$$d(p_i, p_j) > t_b$$

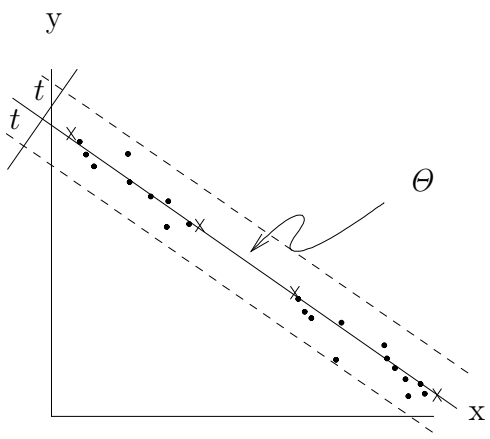



Figura 4.1: Detector de segmentos de la línea

.

donde $t_b$ es un umbral predefinido y $d(p_i, p_j)$ es la distancia entre los puntos.

## 4.1. Método WSAC

Para encontrar el mejor mapa de líneas $\mathcal{M}$, el método WSAC busca iterativamente nuevas líneas aprovechando la secuencialidad de la imagen de rango. El algoritmo de búsqueda de una línea se describe a continuación, en las secciones 4.1.2 y 4.1.3 se describen aspectos específicos de la búsqueda y en qué momento el mapa $\mathcal{M}$ se ha completado, por último en la sección 4.1.4 se muestra el algoritmo completo y se muestra un ejemplo.

### 4.1.1. Búsqueda de una línea

La figura 4.2 ilustra el mecanismo que el algoritmo usa para encontrar una línea. En el primer paso se busca la mejor línea $\Theta_l$ dentro de una ventana local $S_l$ de tamaño $t_l$ como lo muestra la figura 4.2(a). Para encontrar la mejor línea $\Theta_l$ dentro de la ventana $S_l$ se puede usar cualquier método de selección aleatoria. Si la línea

extraída $\Theta_l$ tiene suficiente consenso entonces se puede realizar e segundo paso, de lo contrario buscaremos en otra ventana. En el segundo paso, el de refinación global, consiste de varias acciones. Primero se encuentran el conjunto de puntos $S_L$ como lo muestra la figura 4.2(b). Los elementos del conjunto $S_L$ son todos aquellos puntos $p_i$ de la imagen de rango que aún no estén asociados a otras líneas y que cumplan $d(p_i, \Theta_l) < t$. En seguida, quitamos de $S_L$ aquellos puntos que pertenecen a segmentos pequeños. Por ejemplo, el grupo $g_3$ de la figura 4.2(b) se descartaría. Para detectar los segmentos de la línea se puede usar el algoritmo SLD descrito en la sección 4 de la página 66. Por último, como se ilustra la figura 4.2(d), se ajusta la línea $\Theta'_l$ al conjunto de puntos $S_L$.

### 4.1.2. Selección de la ventana $S_l$ y condición de terminación

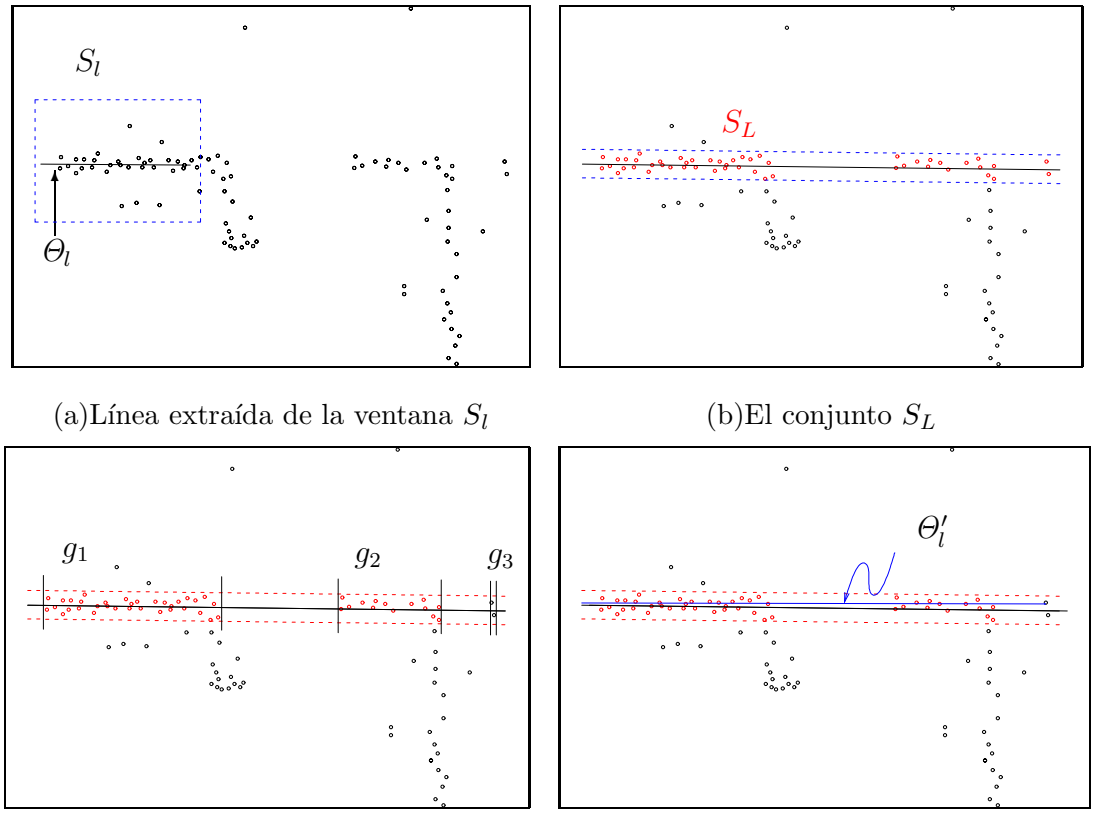
En la sección anterior se introdujo el mecanismo de búsqueda de una línea. En esta sección se introduce cómo se determina la ventana $S_l$ en cada búsqueda y cómo se determina cuando ya no es necesaria otra búsqueda.

La búsqueda de nuevas líneas en WSAC se hace de manera ordenada. La primer ventana $S_l$ se define simplemente seleccionando los primeros $t_l$ puntos de la imagen, esto es, el punto de inicio de la ventana es $l = 1$. Con esta ventana $S_l$ se realiza la búsqueda de una línea como se describió en la sección anterior. Para las nuevas busquedas, la definición de la ventana $S_l$ dependerá de el éxito de la búsqueda anterior:

- Si la búsqueda anterior con ventana $S_l^i$ y con punto inicial $l^i$ se encontró una línea, entonces la nueva ventana $S_l^{i+1}$ comienza en el siguiente punto de la imagen que aún no esté asociado a una línea. Por lo que cumple que $l^{i+1} \geq l^i$.

- Si la búsqueda anterior no se encontró una línea, entonces la siguiente ventana $S_l^{i+1}$ comienza en el $(t_d + 2)$–ésimo punto de la ventana anterior. Donde $t_d \leq t_l$ es el desplazamiento definido de la ventana.

.




(a)Línea extraída de la ventana $S_l$

(b)El conjunto $S_L$

(c)Se eliminan de $S_L$ los puntos de grupos pequeños ($g_3$)

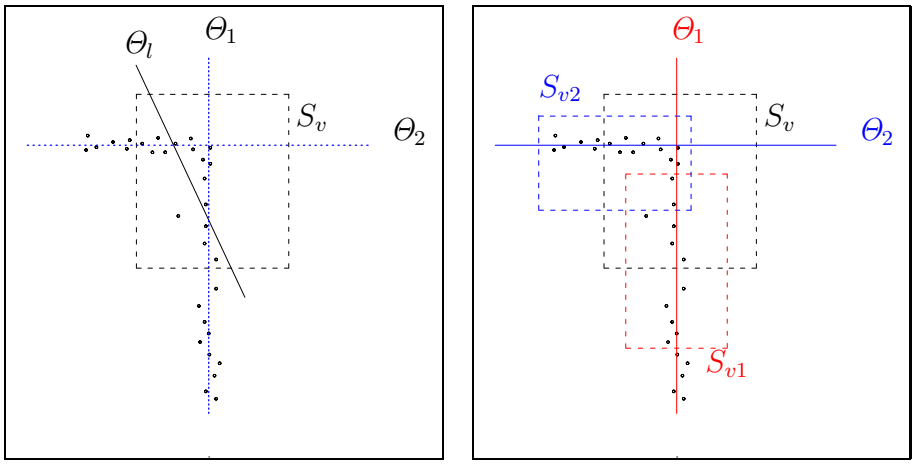
(d)Se obtiene la línea $\Theta'_l$

Figura 4.2: El método WSAC

Usando el procedimiento anterior se garantiza que se busca a lo largo de toda la imagen de rango y que no se repite dos veces la misma búsqueda. La búsqueda de líneas concluye en el momento que no se puede seleccionar una ventana cuyo punto inicial cumpla $l < n$, donde $n$ es el número de puntos de la imagen de rango.

### 4.1.3. La ventana deslizante

En las secciones anteriores se describió cómo se encuentra una línea y cuándo concluye la búsqueda de nuevas líneas. En esta sección se estudia la naturaleza de la




(a)Dificultad de $S_l$ fija (b)$S_l$ deslizante

Figura 4.3: Ventanas fija y deslizante

ventana local $S_l$.

La primer aproximación consiste en seleccionar una ventana fija durante la búsqueda local en la existan $t_l$ puntos que aún no hayan sido asociados a una línea. La principal dificultad de esta aproximación se ilustra en la figura 4.3(a). La línea $\Theta_l$ se obtiene si se elige la ventana $S_l$. A partir de esa ventana no es posible obtener las líneas reales, que en la figura están marcadas por $\Theta_1$ y $\Theta_2$. Este problema se debe a una elección inadecuada de $S_l$. La idea inmediata que surge para solucionar el problema es aumentar el ancho de la ventana local $t_l = \|S_l\|$, pero esto no representa una buena solución. Ya que, por un lado se requerirá mayor cantidad de iteraciones para encontrar la línea local y por otro se pierde la ventaja de la búsqueda en un área local reducida: encontrar una primera aproximación de la línea a partir de puntos muy cercanos entre sí.

La solución que se propone consiste en tener una ventana deslizante en lugar de mantener la ventana fija durante el proceso de extracción local. El algoritmo 9 muestra el comportamiento de la ventana deslizante usada en la busqueda local. Al usar la ventana deslizante, se pueden extraer las líneas $\Theta_1$ y $\Theta_2$, como lo muestra la

figura 4.3(b).

El requerimiento de que la cardinalidad del conjunto $S_l$ sea menor o igual que $t_l$ es necesario para poder comparar el consenso obtenido durante la fase de extracción local.

En la siguiente sección, se describe más a fondo el comportamiento de la ventana deslizante cuando se ejemplifica el comportamiento del algoritmo WSAC.

**Algoritmo 9** WSAC: Búsqueda local

**Entradas:** La imagen de rango $\mathcal{P} = \{(x_i, y_i)|i = 1 \dots n\}$, el punto de referencia $l$, el número de puntos de deslizamiento $k_d$, un tamaño de ventana $t_l$, y un número de intentos $m$

**Salidas:** Una línea $\Theta^\bullet$ con consenso $C^\bullet$

1. $j \leftarrow 0$, $C^\bullet \leftarrow 0$.
2. Calcular $k_s$, $0 \leq k_s \leq k_d$ usando una distribución de probabilidad uniforme
3. Calcular el punto inicial $l_i$ de la ventana al saltar $k_s$ puntos desde $l$
4. Calcular la ventana $S_l$, seleccionando $t_l$ puntos a partir de $l_i$
5. Seleccionar aleatoriamente dos puntos de $S_l$ y con ellos calcular los parámetros de la línea $\Theta_j$
6. Calcular el costo de la línea $C_{\Theta_j}$ usando una función de peso
7. Si $C_\Theta > C^\bullet$ entonces $C^\bullet \leftarrow C_\Theta$, $\Theta^\bullet \leftarrow \Theta_j$
8. Incrementar $j$
9. Si $j < m$ ir al paso 2

### 4.1.4. Algoritmo

WSAC se muestra en el algoritmo 10, donde se incluyen los mecanismos de selección de ventana deslizante (sección 4.1.3) y avance de ventana (sección 4.1.2). El

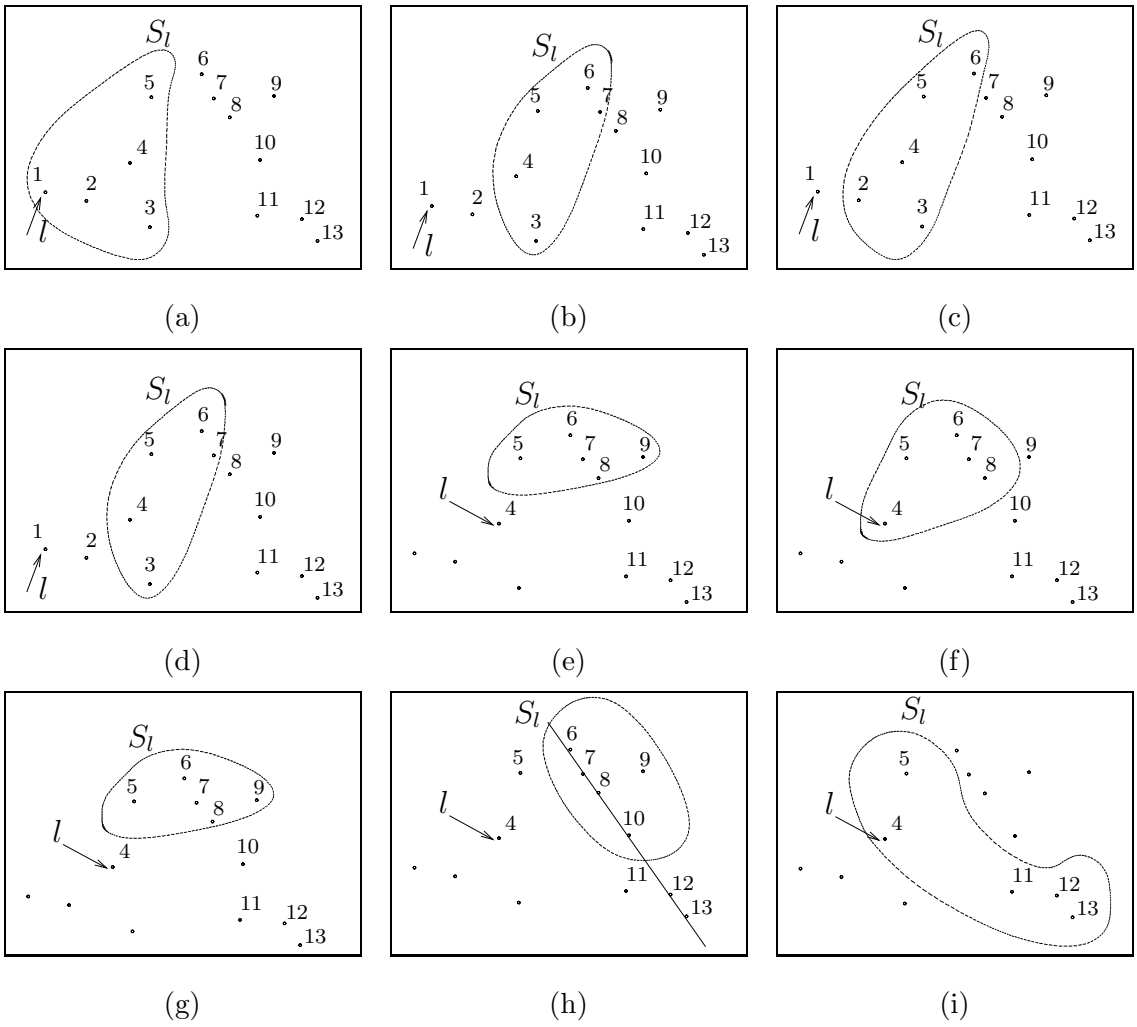



Figura 4.4: Ejemplo del método WSAC

comportamiento del algoritmo se ejemplifica en la figura 4.4, los puntos que aún no se ha realizado una búsqueda y que no están asociados a una línea están marcados por los números. Si definimos el ancho de la ventana local por $t_l = 5$, el desplazamiento de ventana $k_d = 2$ y el consenso mínimo $c_{\min} = 4$. La primera búsqueda usará una ventana con punto de referencia $l = 1$. En las figuras 4.4(a), 4.4(b), 4.4(c) y 4.4(d) no es posible encontrar una línea con suficiente consenso. El nuevo punto de referencia es el cuarto ($k_d + 2 = 4$) de la ventana anterior, o sea el número 4. En las figuras

4.4(e), 4.4(f), 4.4(g) y 4.4(h) se realiza la nueva búsqueda local, la mejor línea de esta búsqueda se muestra en la figura 4.4(i). Después del refinamiento global, los puntos $6 \ldots 10$, 12 y 13 son asociados a la línea. La siguiente búsqueda nuevamente comienza desde $l = 4$, pero ahora abarca todos los puntos restantes. De esta nueva búsqueda no se podrá obtener una nueva línea y el algoritmo termina con un mapa global con una única línea.

Una característica importante del algoritmo es que la pertenencia de un punto a una línea es irrevocable. Aunque esta característica hace que el algoritmo sea simple y rápido, pueden ocurrir ciertos errores en la asociación punto–línea. El algoritmo de la siguiente sección hace un procesamiento más elaborado para mejorar el resultado final.

**Algoritmo 10** Algoritmo WSAC

**Entradas:** La imagen de rango $\mathcal{P} = \{(x_i, y_i) | i = 1 \ldots n\}$

**Salidas:** Un mapa de líneas $\mathcal{M}$

1. $l \leftarrow 1$
2. Realizar la búsqueda local (algoritmo 9) para obtener $\Theta$, $S_\Theta$ y $C_\Theta$
3. Si $C_\Theta < C_{min}$ entonces
   - *a*) Agregar a $S_\Theta$ los puntos de $\mathcal{P}$ que están representados por $\Theta$
   - *b*) Eliminar de $S_\Theta$ aquellos puntos que pertenecen a segmentos de longitud pequeña
   - *c*) Si $S_\Theta$ tiene más de un elemento, reestimar los parametros de la línea usando $S_\Theta$, y eliminar los elementos de $S_\Theta$ de $\mathcal{P}$ y agregar $\Theta$ a $\mathcal{M}$
4. si no
   - *a*) Incrementar $l$ en un valor constante $l \leftarrow l + k$
5. Si no se ha alcanzado el final de $P$, ir al paso 2

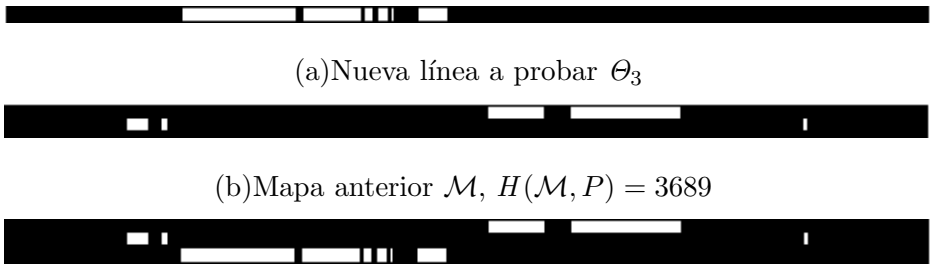
## 4.2. WSAC-GE

Dado que se desconoce el número total de líneas $n_{\dot{\mathcal{M}}}$ del mejor mapa de líneas $\mathcal{M}$ ($n_{\mathcal{M}} = \left\|\dot{\mathcal{M}}\right\|$), este nuevo algoritmo parte de un mapa sin líneas $\mathcal{M} = \emptyset$ y de acuerdo a la situación, se trata de aumentar o disminuir el número de líneas. Así, en cada iteración el método propuesto realiza dos tareas: *la prueba de agregación de líneas* y *prueba de remoción de líneas*. La intención es mejorar en todo momento el modelo maximizando una función $H(\mathcal{M}, P)$ que evalúa la bondad del modelo $\mathcal{M}$ respecto de la imagen de rango $P$.

### 4.2.1. Prueba de Agregación de líneas

En esta fase se genera una nueva línea $\Theta$ como se describe más adelante, la nueva línea puede enriquecer al modelo anterior $\mathcal{M}$. Para saberlo, se evalúa $H$ con el nuevo modelo propuesto que incorpora la nueva línea $\mathcal{M}' = \mathcal{M} \cup \{\Theta\}$ y se acepta en caso de que la evaluación de $H$ mejore respecto de la evaluación anterior.

La figura 4.5 muestra el proceso de agregación de una nueva línea. En la figura 4.5(a) se representa la cercanía de los puntos $p_i$ a una línea en forma gráfica. En el extremo izquierdo se presenta la cercanía de $p_1$, seguida de $p_2$, y así sucesivamente hasta llegar a $p_{361}$. Es decir una imagen de $1fila \times 361$ pixeles. Un tono de gris claro denota que el punto en cuestión está cercano a la línea, mientras que tonos oscuros denotan puntos más alejados. El nivel de gris es función de la proximidad del punto a la línea. En 4.5(b) se presenta una imagen de $2filas \times 361$ pixeles que representa un mapa $\mathcal{M}$. Al unir las figuras 4.5(a) y 4.5(b) se obtiene el nuevo mapa propuesto $\mathcal{M}' = \mathcal{M} \cup \{\Theta_3\}$ lo que se muestra en la figura 4.5(c). Se puede observar en el mapa $\mathcal{M}'$ se aceptó debido a que las líneas tienen diferentes puntos de la imagen que las soportan y entonces $\mathcal{M} \leftarrow \mathcal{M}'$.

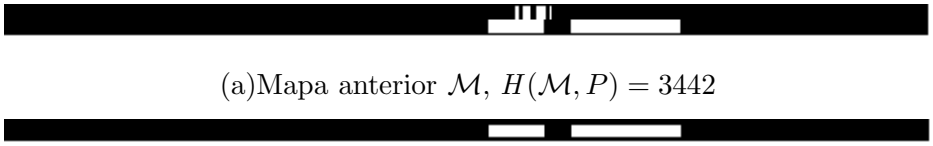


(a)Nueva línea a probar $\Theta_3$

(b)Mapa anterior $\mathcal{M}$, $H(\mathcal{M}, P) = 3689$

(c)Nuevo mapa $\mathcal{M}'$, $H(\mathcal{M}', P) = 9857$

Figura 4.5: Agregación de una línea



(a)Mapa anterior $\mathcal{M}$, $H(\mathcal{M}, P) = 3442$

(b)Nuevo mapa $\mathcal{M}'$, $H(\mathcal{M}, P) = 3609$

Figura 4.6: Remoción de una línea

### 4.2.2. Prueba de Remoción de Líneas

Existe la posibilidad de que un mapa parcial $\mathcal{M}$, tenga una línea con poco consenso y que los puntos que la soporten además realmente pertenezcan a otra línea. Para quitar ese tipo de líneas en la *prueba de remoción* se selecciona aleatoriamente una línea del mapa actual. La línea se desecha si se mejora la evaluación de $H$ al suprimirla del mapa actual. En la figura 4.6 se muestra un ejemplo de una línea removida. En la figura 4.6(a) se muestra un modelo $\mathcal{M}$ cuya primer línea tiene poco consenso. En 4.6(b) se muestra el modelo $\mathcal{M}' = \mathcal{M} - \{\Theta_1\}$ resultante de desechar la primera línea. El modelo $\mathcal{M}'$ se acepta ya que $H(\mathcal{M}', P) > H(\mathcal{M}, P)$ y entonces $\mathcal{M} \leftarrow \mathcal{M}'$.

## 4.3. Algoritmo

Las fases de prueba de agregación y prueba de remoción se realizan en cada iteración y al final se obtiene el ML cuando ya no se pueda mejorar $H$.

El método se describe en el algoritmo 11, la función de evaluación $H$ y los detalles del algoritmo se describen más adelante.

**Algoritmo 11** Algoritmo WSAC–GE

**Entradas:** Una imagen de rango $P = \{(x_i, y_i) \,|i = 1 \dots n\}$ y un número dado de intentos $n_t$.
**Salidas:** un mapa de líneas $\mathcal{M}$

1. Se inicializa $\mathcal{M} \leftarrow \emptyset$, $i = 1$, $l = 0$
2. Iterar mientras que $i < n_t$
   - *a*) Agregación de líneas
     Se genera una nueva línea $\Theta$ a partir de dos puntos (4.5),
     Se propone un nuevo mapa $\mathcal{M}' = \mathcal{M} \cup \{\Theta\}$,
     Si $H(\mathcal{M}', P) > H(\mathcal{M}, P)$ entonces $\mathcal{M} \leftarrow \mathcal{M}'$ e $i = 0$.
   - *b*) Eliminación de líneas
     Se selecciona aleatoriamente la línea $\Theta_j \in \mathcal{M}$,
     Se propone el nuevo modelo $\mathcal{M}' \leftarrow \mathcal{M} - \{\Theta_j\}$,
     Si $H(\mathcal{M}', P) > H(\mathcal{M}, P)$ e $i = 0$.
   - *c*) se incrementa $i$
3. Asociar puntos con líneas (sección 4.5.3)
4. Refinación de las líneas (sección 4.5.4)

## 4.4. Función de evaluación del mapa

Dada una imagen de rango $P = \{(x_i, y_i)\,|i = 1 \ldots n\}$ y el mapa de líneas $\mathcal{M} = \{\Theta_j | j = 1 \ldots l\}$ donde $l$ es el número de líneas. Se calculan los errores $e_{\perp ij}$ de cada punto $p_i \in P$ a cada línea $\Theta_j \in \mathcal{M}$. Aplicando la función de ponderación Beaton–Tukey $w(e_{\perp ij})$ (sección. 2.7) dada por:

$$w_{\mathrm{BT}}(e) = \begin{cases} \left[1 - (e/k)^2\right]^2 & \text{para } |e| \leq k \\ 0 & \text{para } |e| > k \end{cases}$$

En la tabla 4.1 se muestra el resultado en forma de matriz obtenido de realizar este paso. Los renglones denotan las líneas y las columnas los puntos.

| | $p_1$ | $p_2$ | $\cdots$ | $p_n$ |
|---|---|---|---|---|
| $\Theta_1$ | $w\left(e_{\perp 11}\right)$ | $w\left(e_{\perp 21}\right)$ | $\cdots$ | $w\left(e_{\perp n1}\right)$ |
| $\Theta_2$ | $w\left(e_{\perp 12}\right)$ | $w\left(e_{\perp 22}\right)$ | $\cdots$ | $w\left(e_{\perp n2}\right)$ |
| | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| $\Theta_l$ | $w\left(e_{\perp 1l}\right)$ | $w\left(e_{\perp 2l}\right)$ | $\cdots$ | $w\left(e_{\perp nl}\right)$ |
| | $s_1$ | $s_2$ | $\cdots$ | $s_n$ |

Tabla 4.1: Errores del punto $p_i$ a la línea $\Theta_j$

El último renglón es el acumulado de la columna correspondiente. Esto es

$$s_i = \sum_{j=1}^{l} w(e_{\perp ij})$$

Así se puede definir la matriz de correspondencias normalizada $C$ de los puntos $p_i \in P$ a las líneas $\Theta_j \in \mathcal{M}$:

$$C = [c_{ij} \mid i = 1 \ldots n, j = 1 \ldots l] \tag{4.1}$$

donde $c_{ij}$ se calcula de la siguiente manera

$$c_{ij} = \begin{cases} \frac{w\left(e_{\perp ij}\right)}{s_i} & \text{si } s_i < t \\ 0 & \text{si } s_i = 0 \end{cases} \tag{4.2}$$

donde $t$ es un umbral predefinido que sirve para evitar que ponderaciones por debajo de $t$ sean consideradas como 1 después de la normalización. Con la matriz de correspondencias normalizadas $C$ se puede calcular entonces el consenso normalizado $h_j$ de la línea $\Theta_j$ usando

$$h_j = \sum_{i=1}^{n} c_{ij} \tag{4.3}$$

como se ilustra la tabla 4.2.

| | $p_1$ | $p_2$ | $\cdots$ | $p_n$ | $h_j$ |
|---|---|---|---|---|---|
| $\Theta_1$ | $c_{11}$ | $c_{21}$ | $\cdots$ | $c_{n1}$ | $h_1$ |
| $\Theta_2$ | $c_{12}$ | $c_{22}$ | $\cdots$ | $c_{n2}$ | $h_2$ |
| | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ | $\vdots$ |
| $\Theta_l$ | $c_{1l}$ | $c_{2l}$ | $\cdots$ | $c_{nl}$ | $h_n$ |

Tabla 4.2: Cálculo del consenso normalizado $h_j$ de la línea $\Theta_j$

Finalmente el mapa $\mathcal{M}$ se puede evaluar respecto de la imagen $P$ usando la función

$$H(\mathcal{M}, P) \;=\; \sum_{j=1}^{l} h_j^2 \tag{4.4}$$

La tabla 4.2 muestra gráficamente en las figuras 4.5 y 4.6.

Si el entorno de donde se obtuvo la imagen de rango $P$ consistiera de un único objeto lineal, todos los puntos $p_i \in P$ fueran lecturas de ese objeto y además no existiera error en las mediciones del láser, entonces la función $H(\dot{\mathcal{M}}, P)$ tendría un máximo cuando se obtiene el mejor modelo $\dot{\mathcal{M}}$ con una sola línea $\Theta$ que pase por todos los puntos y $H(\dot{\mathcal{M}}, P) = n^2$, siendo $n = |P|$. En el caso de varias líneas, la función tendrá un máximo cuando el modelo $\dot{\mathcal{M}}$ tenga la mayor cantidad de líneas cada una de ellas con la mayor cantidad de puntos que solo pertenezcan a una sola línea a la vez.

## 4.5. Generación de hipótesis

Como se describió anteriormente, en cada fase de prueba de agregación se busca integrar una nueva línea $\Theta$ al mapa. Al mecanismo de generar una nueva línea le llamaremos generación de hipótesis. Si en lugar de generar hipótesis al encontrar la línea que pasa por dos puntos seleccionandos aleatoriamente, se aprovecha la secuencialidad de $P$ entonces se puede mejorar el desempeño del algoritmo 11. A continuación se describen dos métodos sugeridos para la generación de hipótesis.

### 4.5.1. Generación de hipótesis a partir de puntos

Para aprovechar la organización secuencial de $P$, se puede generar la nueva línea $\Theta$ al unir dos puntos estén próximos entre sí. Con este objetivo se selecciona un punto $p_{k_1}$ dentro del rastreo de forma aleatoria, pero el siguiente punto $p_{k_2}$ se obtendrá usando una distribución gaussiana centrada en $p_{k_1}$ y con una desviación estándar $\sigma_L$. Este mecanismo se describe en el algoritmo 12. Un factor importante es el valor que se seleccione para $\sigma_L$. Un valor pequeño de $\sigma_L$ indica una fuerte tendencia a seleccionar puntos muy próximos entre sí, mientras que valores grandes de $\sigma_L$ tenderán a una distribución uniforme. La elección de $\sigma_L$ puede hacerse de manera empírica considerando la longitud promedio de las líneas del entorno, la resolución angular del láser, etc.

### 4.5.2. Generación de hipótesis usando una ventana estática

Otra manera de lograr buenas hipótesis iniciales es usar el mecanismo de ventana estática $S_l$. Para ello, seleccionamos de forma aleatoria un punto inicial $k_1$ y a partir de ahí definimos la ventana estática $S_l$ tomando $t_l$ puntos consecutivos de la imagen de rango. La nueva hipótesis se encuentra usando algún método de selección aleatoria (v.g. MSAC) para encontrar la mejor línea en esa ventana.

**Algoritmo 12** WSAC–GE: Generación de una línea local $\Theta_{l+1}$

**Entradas:** Una imagen de rango $P = \{(x_i, y_i) \,|i = 1 \dots n\}$ y la desviación estándar $\sigma_L$

**Salidas:** La línea $\Theta_{l+1}$

1. Seleccionar aleatoriamente un primer punto $p_{k_1} \in P$, para $1 \leq k_1 \leq |P|$
2. Seleccionar un segundo punto $p_{k_2} \in P$ usando una distribución gaussiana con media $k_1$ y desviación estándar $\sigma_L$, verificando que $1 \leq k_2 \leq n$
3. Obtener la nueva línea $\Theta$ al unir los puntos $p_{k_1}$ y $p_{k_2}$

### 4.5.3. Asociación final de puntos y líneas

Si el mapa $\mathcal{M}$ no se puede mejorar después de $n_t$ intentos entonces se ha encontrado el mapa $\dot{\mathcal{M}}$ y sólo resta encontrar la asociación entre puntos y líneas. La asociación entre puntos y líneas estará dada al asociar a cada punto $p_i$ a su línea más próxima. Esto se hace usando la tabla 4.2, para cada punto (columna) se selecciona el máximo valor de la línea (fila).

### 4.5.4. Refinación de las líneas

Aunque una línea se obtenga al unir dos puntos típicos (respecto de un grupo), es común que dicha línea no represente de forma adecuada al conjunto de puntos de donde se extrajo. Como ejemplo de este hecho, en la figura 4.7 se observa que la línea $\Theta_a$ que une a los puntos $p_1$ y $p_2$ no es la más representativa. Después de un ajuste, la línea $\Theta'_a$ mejora sustancialmente la representación de los puntos. Una vez que se ha encontrado $\dot{\mathcal{M}}$ se propone una fase de post–procesamiento que refine las líneas usando para ello un ajuste con IRLS aprovechando que tal método converge a un mínimo local.

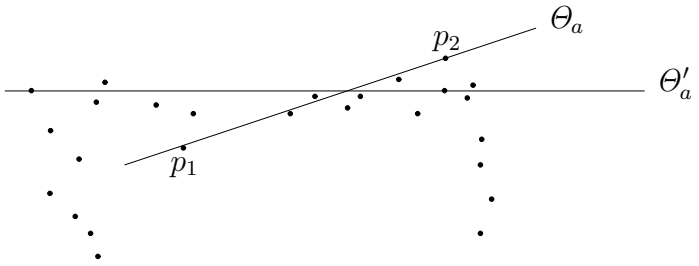
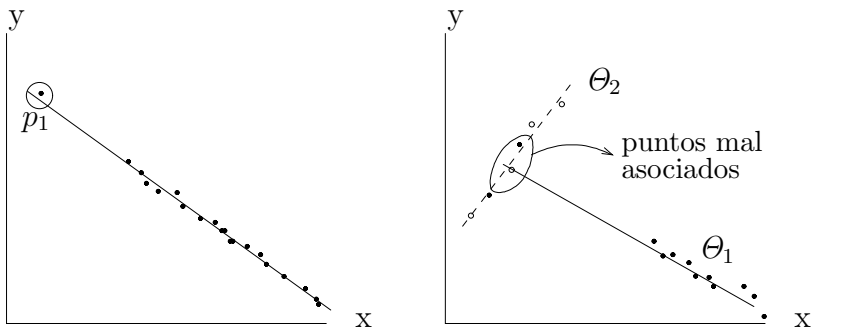



Figura 4.7: Refinación de la línea




(a)Problema de alcance (b)Problema en una intersección

Figura 4.8: Problemas de agrupación

## 4.6. Variantes del método

El método expuesto localiza líneas generadas a partir de puntos cercanos, favorece a la selección de líneas con soporte alto y busca que cada punto se asocie a una sola línea. Pero se pueden lograr mejoras tanto en el tiempo que consume el algoritmo como en la calidad del ML al emplear algunas técnicas que en esta sección se abordan.

### 4.6.1. Penalización por segmentos

En el método WSAC–GE no se usan detectores de puntos de ruptura que simplifiquen el procesamiento de la imagen de rango $P$. Al procesar todos los puntos de $P$ a la vez se encuentran dos tipos de dificultades para asociar correctamente los puntos

$p_i \in P$ a las líneas $\Theta_j \in \dot{\mathcal{M}}$:

1. Agrupación por alcance,

2. Agrupación en una intersección

.

**Algoritmo 13** Penalización de puntos asociados a segmentos pequeños

**Entradas:** El vector de asociación $c_{ij}$, $i = 1, \ldots, |P|$ de la línea $\Theta_j \in \mathcal{M}$

**Salidas:** El vector de asociación $c_{ij}$, $i = 1, \ldots, |P|$ con penalización de puntos asociados a segmentos pequeños

1. Encontrar el conjunto de segmentos poblados de la línea $\Theta_{l+1}$ usando un SLD (secc. 4 pág. 66), $\{s_i | i = 1 \ldots n_{\text{seg}}\}$.

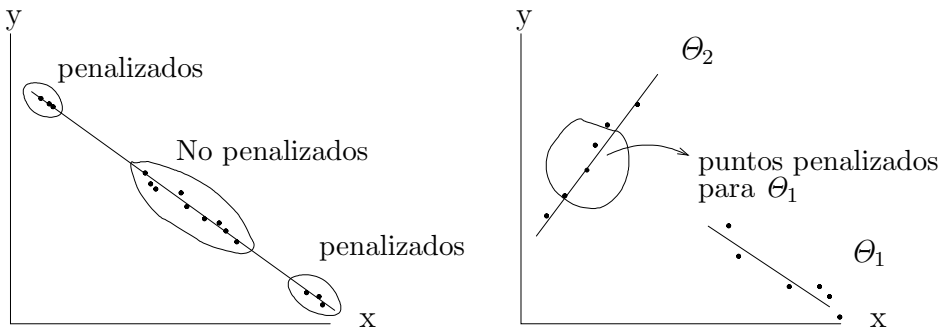
2. Para cada segmento $s_i$ hacer:

   *a*) Si la cantidad de puntos asociados a cada segmento es menor a un cierto umbral $n_{\min}$ (v.g. $n_{\min} = 1$ para puntos aislados) entonces los puntos que pertenecen a ese segmento se les asocia un $c_{ij} = 0$.

   *b*) Si la distancia euclidiana entre el primer y último punto del segmento es menor a un cierto umbral $n_{\text{lmin}}$ (v.g. $n_{\text{lmin}} = 5\text{cm}$) entonces el valor de $c_{ij}$ se multiplica por una constante de penalización, $k_p < 1$.

El problema de *agrupación por alcance* se ilustra en la figura 4.8(a). En éste caso no existe información suficiente para decidir si el punto $p_1$, que es el más alejado del grupo, pertenece a la línea o no. Aunque este tipo de problema pudiera parecer irrelevante ocasiona errores de precisión pues puntos como el $p_1$ suelen funcionar como "puntos palanca" ya que afectan considerablemente el ajuste final.

El problema de *agrupación en una intersección* ilustrado en la figura 4.8(b) consiste en encontrar la asociación correcta cuando los puntos parecen pertenecer a dos líneas a la vez (v.g. en esquinas o intersecciones de segmentos). En este caso, es posi-




(a)Corrección de alcance (b)Corrección de agrupación

Figura 4.9: Corrección con penalización

ble que la línea $\Theta_2$ no se encuentre debido a que algunos puntos están mal agrupados y están considerados como parte del grupo que soporta a $\Theta_1$.

Para resolver ambas situaciones se puede realizar una *penalización por segmentos* como se describe a continuación. En el paso de agregación de una nueva línea, después de haber generado la nueva línea $\Theta$ y antes de introducir la nueva fila en la matriz de correspondencias $C$ se sugiere el uso de la penalización por segmentos como lo describe el algoritmo 13. En la figura 4.9(a) se muestra que, con la penalización realizada al grupo de puntos se logra que los puntos dentro de los dos segmentos pequeños reduzcan su influencia tanto para el cálculo del consenso $h$ de la línea como en el momento del ajuste final de la línea descrita más adelante. Con ello se consigue que se le de más importancia a aquellos puntos que se encuentran en segmentos más grandes. En la figura 4.9(b) se ilustra que con la penalización realizada al segmento de longitud pequeña se facilita la extracción tanto de $\Theta_1$ como de $\Theta_2$.

### 4.6.2. Lineas Similares

Si en el paso de agregación del algoritmo 11 se intenta introducir al mapa $\mathcal{M}$ una línea $\Theta$, y existe una línea $\Theta_j \in \mathcal{M}$ tal que $\Theta_j \simeq \Theta^o$, entonces después de evaluar $\mathcal{M}$

se rechazará. Para evitar un procesamiento innecesario se puede evaluar la similitud entre la nueva línea $\Theta$ con cada elemento del modelo $\mathcal{M}$, si existe una línea similar entonces se omite la evaluación de $\mathcal{M}'$ y se rechaza directamente el nuevo modelo $\mathcal{M}'$. En el apéndice B se muestra la forma de evaluar la similitud entre líneas.

La figura 4.10 muestra dos líneas similares $\Theta_1$ y $\Theta_2$, como se puede observar las dos líneas tienen los mismos puntos de soporte, y son similares entre sí. Por tal motivo, si ya se ha encontrado $\Theta_1$ no es necesario evaluar el modelo agregando $\Theta_2$, en cualquier caso, después de la refinación de la línea se obtendrá la línea buscada $\Theta$.

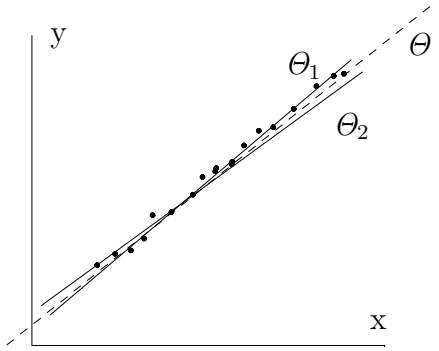



Figura 4.10: Líneas similares.

## 4.7. Resumen

Los algoritmos propuestos en este capítulo muestran que por un lado consideran toda la información dentro la imagen del rastreo $P$, y por otro, aprovechan la secuencialidad de los puntos. Dichos algoritmos usan un mecanismo de selección aleatoria, y una función redescendente para orientar la búsqueda a aquellas líneas que estén soportadas por puntos que se encuentren lo más cercano posible a la línea.

Los métodos propuestos parten de una búsqueda local y luego hacen consideraciones globales. Sin embargo, mientras que para el algoritmo WSAC, es suficiente que una línea cumpla ciertas condiciones locales (número de puntos representados

| Algoritmo | WSAC | WSAC–GE |
|---|---|---|
| Aprovecha secuencialidad | SI | SI |
| Uso de un estimador – M | SI | SI |
| Evaluación Global | NO | SI |
| Penalización por segmentos | SI | Opcional |
| Corrección en asociación | NO | SI |
| Uso de detectores de Punto de Ruptura | NO | NO |
| Generación de nuevas líneas con una ventana estática | NO | Opcional |
| Generación de nuevas líneas con una ventana deslizante | SI | NO |
| Generación de nuevas líneas a partir de dos puntos (sin uso de ventana) | NO | SI |
| Comprobación de líneas similares | NO | Opcional |
| Refinación final de las líneas | SI | SI |

Tabla 4.3: Resumen de características de los algoritmos propuestos

por la línea, longitud del segmento, etc.) para integrarla al mapa; para el algoritmo WSAC–GE, es necesaria además una evaluación global del sistema con el objetivo de mejorar la asociación entre puntos y líneas y por ende la calidad del ML. La tabla 4.3 muestra un resumen de los métodos empleados por los algoritmos El algoritmo WSAC, se puede emplear en situaciones donde el tiempo es un factor importante. Para aplicaciones fuera de línea, donde la precisión y exactitud son factores valiosos se puede usar el algoritmo WSAC-GE. Por último, ambos métodos son capaces de encontrar MGPL de buena calidad en un ambientes demasiado ruidosos. En el siguiente capítulo se muestran algunas ideas del uso de MGL para la solución del problema del SLAM, y en capítulo 5 se presentan los resultados experimentales.

# Capítulo 5

# Pruebas y resultados

Este capítulo presenta en primer lugar, los resultados de las pruebas realizadas a los métodos propuestos para la generación de MGL a partir de imágenes de rango; y en segundo lugar, los resultados de integrar los MGLs en el algoritmo ICP.

Como se abordó en la sección 1.3 (página 7), el objetivo del algoritmo ICP es empatar dos imágenes de rango consecutivas usando puntos y representa una primera aproximación a la solución del problema de localización del robot. El algoritmo ICP trata de disminuir la distancia entre los puntos de una imagen con sus correspondientes en la otra imagen. Sin embargo, es poco probable que un mismo punto se haya leído en las dos imágenes consecutivas, y esto representa un problema para el algoritmo ICP. Una solución consiste en proyectar cada punto de una imagen en la línea tangente del punto más cercano en la otra imagen. La propuesta que se probará en este capítulo consiste en proyectar cada punto de una imagen en la línea a la que pertenezca el punto más cercano de la otra imagen.

En general, los programas usados en las pruebas se codificaron con lenguaje C (en particular usando el compilador libre GCC) sobre una plataforma Linux. Para las pruebas se usó una computadora portátil HP Pavilion con un procesador Celeron a 1.1 GHz y con 256 Mb de memoria.

## 5.1. Pruebas a los algoritmos de generación de MGL

Debido a las ventajas que presenta el algoritmo SMSM (tabla 3.1, página 62): fácil de implementar, rápido y muy asertivo; se decidió tomar sus resultados como referencia para evaluar los resultados de los algoritmos WSAC y WSAC–GE.

Para la evaluación de los algoritmos de generación de MGL se realizaron dos tipos de pruebas: en la primera se usaron datos sintéticos, mientras que en la segunda, se usaron datos reales generados en un robot móvil equipado con un láser.

Para los métodos WSAC y WSAC-GE se usó la función Beaton–Tukey, además de los mecanismos de penalización de segmentos y refinación de líneas usando IRLS.

### 5.1.1. Evaluación con datos sintéticos

La primera prueba se realizó usando datos sintéticos obtenidos por el simulador de robot móvil. El simulador se diseñó específicamente para esta prueba, el uso de este programa se describe en el apéndice C. El propósito de la prueba es observar tanto el grado de robustez como el tiempo consumido por cada algoritmo. La principal ventaja de usar datos sintéticos es la de tener mayor control sobre las variables del problema (v.g. posición y orientación del robot en el ambiente). De tal manera que se pueden contrastar los parámetros estimados por los algoritmos contra los parámetros de las mismas líneas que fueron fijados en el programa de simulación. El entorno sintético usado para esta prueba se muestra en la figura 5.1. Para mayor simplicidad de la comparación se etiquetan las líneas con un número al lado.

La proporción de errores del láser (apéndice C.2) usada, se muestra en la tabla 5.2. En cada imagen se toman 361 lecturas cubriendo una área semicircular. El recorrido que el robot siguió en la prueba se muestra en la figura 5.2, obteniendo un total de 350 imágenes de rango.

Cada imagen de rango $R_i | i = 1, \ldots, 350$ es procesada por el algoritmo que se esté probando, con lo que se obtiene el mapa $\mathcal{M}_i$. Con el mapa obtenido $\mathcal{M}_i$, se evalúa

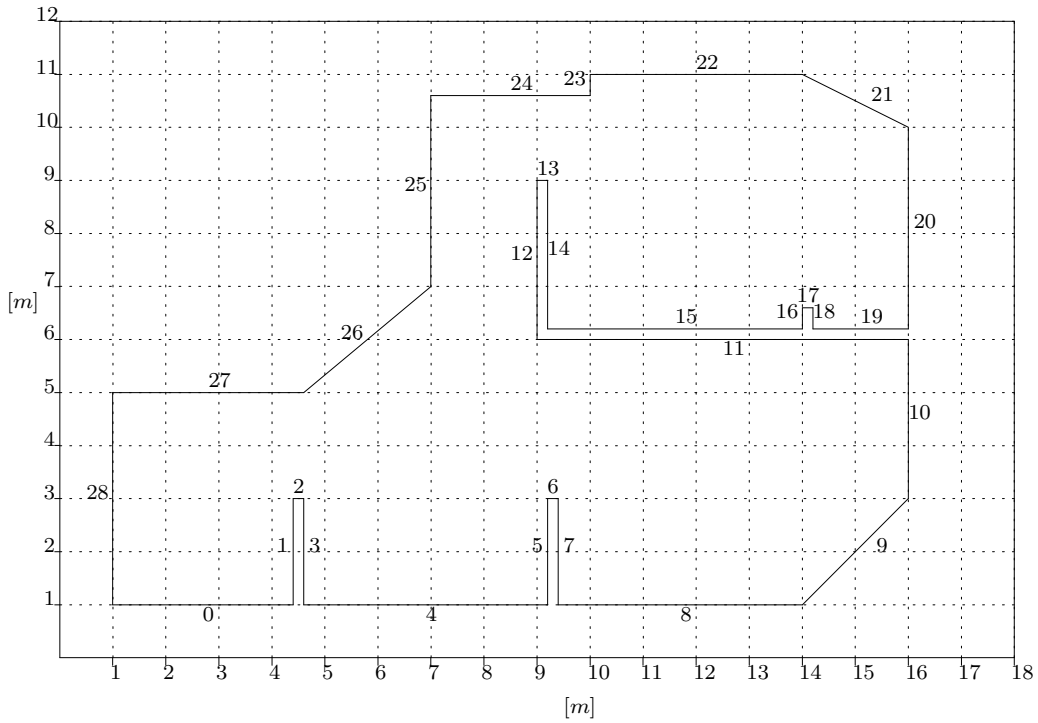



Figura 5.1: Entorno sintético

| Caso | p |
|---|---|
| A | 0.1 |
| B | 0.25 |
| C | 0.65 |

Tabla 5.1: Proporción de errores generados por el simulador

| | |
|---|---|
| $\sigma_r$ | 0.02 m |
| $\Delta\alpha$ | 0.5° |
| $\alpha_i$ | 0° |
| $\alpha_f$ | 180° |

Tabla 5.2: Configuración de los parámetros del láser en el simulador

la similitud de las líneas $l \in \mathcal{M}_i$ con las líneas del mapa original $l \in \mathcal{M}_{\text{orig}}$ usando el mismo sistema de referencia. Una vez encontrada la línea original equivalente de

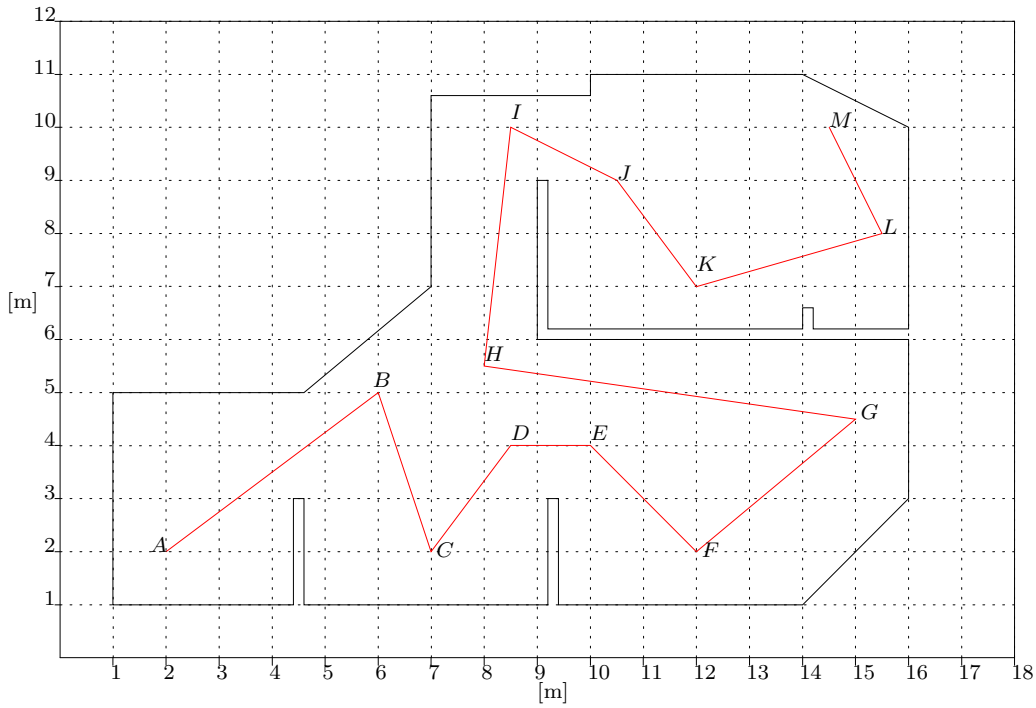



Figura 5.2: Recorrido del robot en el entorno sintético

cada linea $l \in \mathcal{M}_i$ se obtiene la diferencia entre sus parámetros, es decir $\Delta r$ y $\Delta\alpha$.

La tabla 5.3 muestra las estadísticas de los algoritmos SMSM, WSAC y WSAC–GE. Esta tabla muestra el error de distancia y el error angular promedio de cada línea. La numeración de las líneas sigue las etiquetas de la figura 5.1. Los valores marcados en negritas indican el algoritmo que obtuvo mejor resultado en cada caso. Como se puede apreciar, la tabla 5.3 no reporta las líneas con etiquetas: 14, 18 y 23, ya que en el recorrido (fig. 5.2) no se tomó nunca mediciones referentes a dichas líneas.

La tabla 5.3 muestra un resultado interesante, el algoritmo WSAC–GE no encuentra las líneas 2, 6, 13 y 17 (todas ellas de 20cm). Esto se puede explicar de la siguiente manera: si en el proceso de búsqueda de líneas WSAC–GE encuentra un segmento pequeño (como es el caso), sus puntos asociados serán penalizados. El segmento con puntos penalizados se removerá del mapa en un paso de remoción. Si ese no fuera el caso, en la asociación final de puntos a líneas, el segmento se quedará sin soporte y

| | SMSM | | WSAC | | WSAC–GE | |
|---|---|---|---|---|---|---|
| linea | $\lvert\Delta_r\rvert$ [mm] | $\lvert\Delta_\alpha\rvert$ [°] | $\lvert\Delta_r\rvert$ [mm] | $\lvert\Delta_\alpha\rvert$ [°] | $\lvert\Delta_r\rvert$ [mm] | $\lvert\Delta_\alpha\rvert$ [°] |
| 0 | **0.88** | 0.03 | 2.27 | 0.10 | 1.05 | **0.03** |
| 1 | 2.06 | 0.29 | 1.33 | 0.14 | **0.35** | **0.04** |
| 2 | 23.04 | 1.28 | **1.12** | **0.14** | – | – |
| 3 | 5.36 | 0.16 | 3.10 | 0.07 | **2.38** | **0.06** |
| 4 | 9.28 | 1.04 | 1.51 | **0.06** | **1.16** | 0.06 |
| 5 | 38.31 | 1.83 | 32.04 | 0.11 | **0.51** | **0.03** |
| 6 | 16.24 | 1.80 | **2.25** | **0.69** | – | – |
| 7 | 2.12 | 0.08 | 2.15 | 0.06 | **1.27** | **0.03** |
| 8 | 7.67 | 0.81 | 2.16 | 0.06 | **1.09** | **0.03** |
| 9 | 7.35 | 0.16 | **2.04** | 0.09 | 2.98 | **0.05** |
| 10 | 3.30 | 0.40 | 1.39 | 0.10 | **0.63** | **0.03** |
| 11 | 0.92 | 0.06 | 1.28 | 0.06 | **0.60** | **0.02** |
| 12 | 15.05 | 0.43 | 12.96 | 0.09 | **1.35** | **0.03** |
| 13 | 6.53 | 1.31 | **6.12** | **0.63** | – | – |
| 15 | 0.48 | 0.01 | 1.57 | 0.04 | **0.43** | **0.01** |
| 16 | 42.02 | 0.67 | **16.46** | **0.27** | 23.09 | 0.36 |
| 17 | **46.34** | **0.81** | – | – | – | – |
| 19 | **1.05** | **0.02** | 1.46 | 0.05 | 1.73 | 0.02 |
| 20 | 5.63 | 0.54 | 1.36 | 0.04 | **0.67** | **0.02** |
| 21 | 4.91 | 0.06 | **2.48** | 0.03 | 2.50 | **0.03** |
| 22 | 6.85 | 0.12 | 18.31 | 0.18 | **1.55** | **0.04** |
| 24 | 1.66 | 0.10 | 24.41 | 0.24 | **0.59** | **0.05** |
| 25 | 3.88 | 0.22 | 1.63 | 0.14 | **1.15** | **0.05** |
| 26 | 3.12 | 0.13 | 1.91 | 0.11 | **0.96** | **0.03** |
| 27 | 4.08 | 0.17 | 1.53 | 0.13 | **0.54** | **0.04** |
| 28 | 1.96 | 0.08 | 1.49 | 0.06 | **0.94** | **0.04** |

Tabla 5.3: Comparación de los algoritmos SMSM, WSAC y WSAC–GE

por ese motivo no se reportará como encontrado.

Ese efecto no ocurre con el algoritmo WSAC, que encuentra las líneas 2, 6 y 13, la línea 17 no se reporta pero eso es ocasionado porque WSAC nunca pudo garantizar la existencia del segmento en la búsqueda local.

En la tabla 5.4 se muestra el desempeño general obtenido. La última columna muestra la asertividad a, este indicador se encuentra calculando:

$$\mathrm{a} = \frac{n_\mathrm{s}}{n_\mathrm{n}}$$

donde $n_\mathrm{s}$ es el número de líneas que se extrajeron por el algoritmo y que tienen correspondiente en el entorno simulado, y $n_\mathrm{n}$ es el número total de líneas extraídas.

La tabla 5.3 muestra que el algoritmo SMSM es muy rápido, la primera columna muestra el máximo tiempo consumido. Por otra parte es menos asertivo y menos preciso que los algoritmos propuestos. El algoritmo WSAC–GE encontró más líneas, pese a estar rechazando a los segmentos muy pequeños. La alta asertividad del algoritmo WSAC representa una ventaja cuando se trata de resolver problemas de más alto nivel, como la localización. Un patrón que ocurre frecuentemente, es el siguiente: el MGL ideal de la figura 5.3(a) cuya aproximación por el método SMSM se ve en la figura 5.3(b). Como se puede ver SMSM encuentra segmentos más cortos en comparación con los encontrados por los algoritmos WSAC y WSAC–GE (figuras 5.3(c) y 5.3(d)).

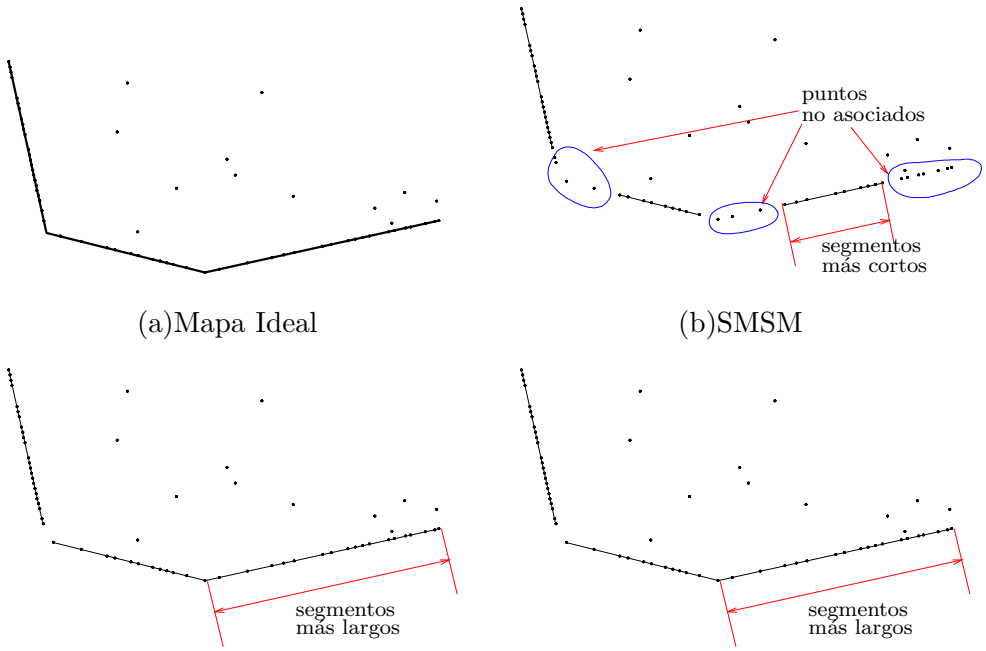
| Algoritmo | Tiempo [ms] | lineas asociadas | $\lvert\Delta\sigma_r\rvert$ [mm] | $\lvert\Delta\alpha\rvert$ [°] | Asertividad % |
|---|---|---|---|---|---|
| SMSM | 29.5 | 1557 | 6.37 | 0.2670 | 87.4 |
| WSAC | 50.1 | 1319 | 5.37 | 0.1000 | 99.2 |
| WSAC–GE | 94.5 | 1620 | 4.12 | 0.1323 | 94.8 |

Tabla 5.4: Resumen de los resultados para el ambiente simulado

### 5.1.2. Evaluación con datos reales

Para esta prueba se tomaron imágenes de rango del laboratorio de sistemas del Posgrado de Ingeniería de la Facultad Eléctrica de la UMSNH. El láser usado es un LMS209-S02 de SICK [SIC] que puede tomar lecturas en un rango de $[0, 180°]$.

La figura 5.4(a) muestra un mapa ideal hecho *a mano* a partir de una imagen de rango original tomada en el laboratorio, las figuras 5.4(b), 5.4(c) y 5.4(d) muestran




(a)Mapa Ideal (b)SMSM

(c)WSAC (d)WSAC–GE

Figura 5.3: Mapa parcial de una imagen de rango sintética

una porción de el mapa obtenido por los algoritmos SMSM, WSAC, y WSAC–GE, respectivamente. De la misma manera, las figuras 5.5(a), 5.5(b), 5.5(c) y 5.5(d) son acercamientos de las figuras 5.4(a), 5.4(b), 5.4(c) y 5.4(d), respectivamente. Como se puede apreciar, los algoritmos WSAC y WSAC–GE se aproximan más al ideal, también se puede concluir que es muy difícil lograr al MGL ideal en situaciones como ésta. En el ejemplo mostrado, las lecturas del láser corresponden al umbral de una puerta. La superficie de la puerta es obscura y además tiene relieve como se puede ver en la fotografía 5.6.

## 5.2. Prueba de generación de mapas globales

Usando los mismos datos que en la sección anterior se integraron 100 mapas parciales obtenidos de las imágenes de rango reales. En la figura 5.7 se muestra el mapa

obtenido usando el algoritmo ICP por el método propuesto en [Arellano05]. En la figura 5.8 se puede observar un mapa muy semejante obtenido al minimizar la distancia de los puntos proyectados a la línea correspondiente en el MGL. Como se puede observar los mapas de las figuras 5.7 y 5.8 no tienen una diferencia sustancial. Sin embargo, mientras que el método usado en [Arellano05] consume un promedio de 4 segundos para encontrar la mejor estimación de la nueva posición del robot, al usar MGL se consume un promedio de 0.3. Esto supone una mejora sustancial muy importante, sobre todo cuando se trata de aplicaciones en tiempo real.

.

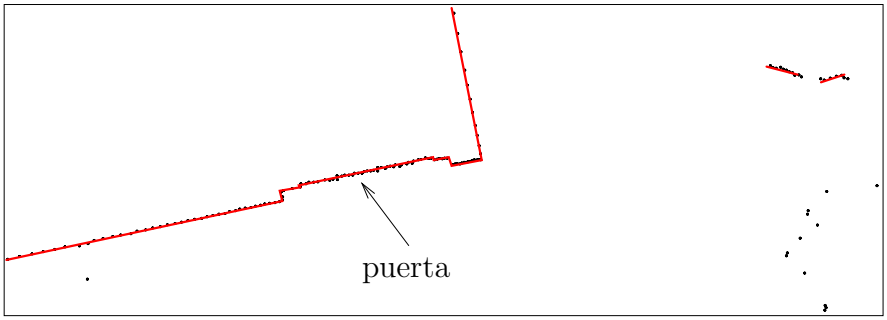
## 5.3. Resumen

Los resultados mostrados en este capítulo reflejan que los algoritmos WSAC y WSAC–GE logran buenos resultados en ambientes ruidosos.

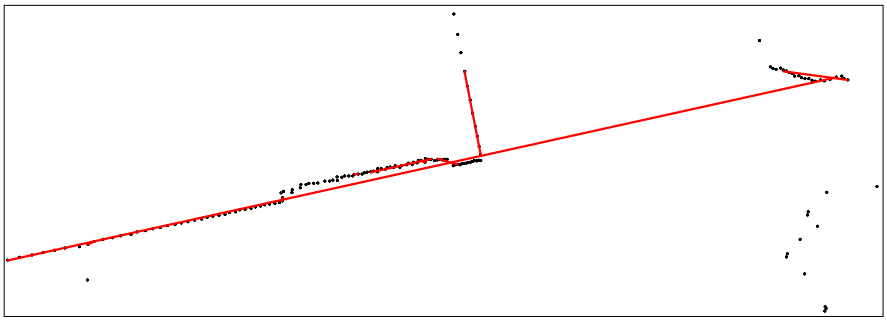
La pruebas realizadas usando el ambiente sintético mostradas en la sección 5.1.1 se muestra que los algoritmos WSAC y WSAC–GE encuentran líneas más precisas y con una mejor asociación entre puntos y líneas en un tiempo razonable en comparación con el método SMSM.

En la sección 5.1.2 se hizo un análisis cualitativo de los mapas obtenidos en el ambiente interior real usando el robot móvil. Se mostró que las líneas de los mapas obtenidos por los algoritmos WSAC y WSAC–GE representan de mejor forma el ambiente interior que los logrados con el método SMSM.

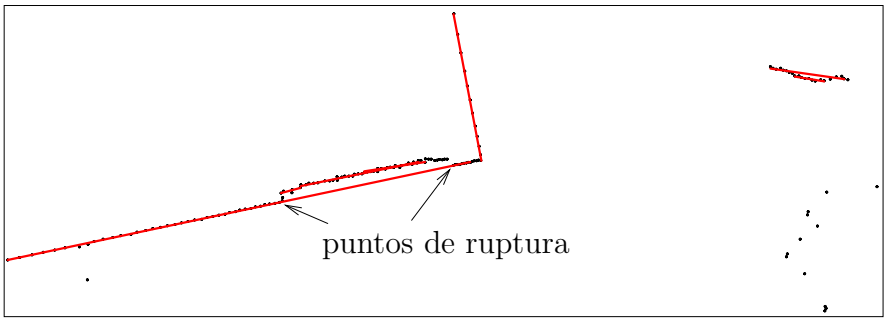
Por último, en la sección 5.2 se mostró que al usar MGL para resolver el problema de generación de mapas globales con ICP se disminuye el tiempo empleado.



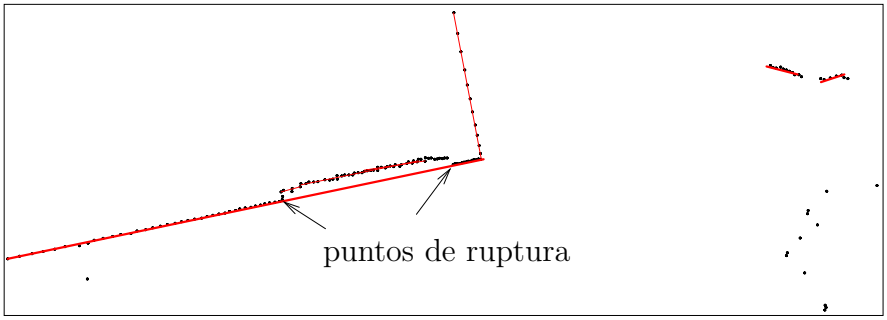

(a)Mapa a mano



(b)SMSM

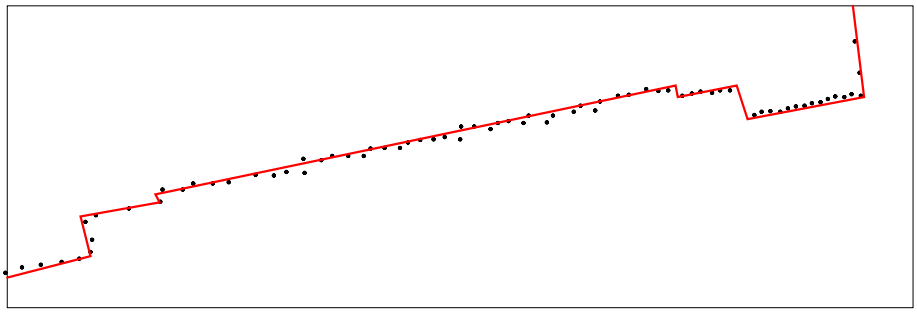



(c)WSAC

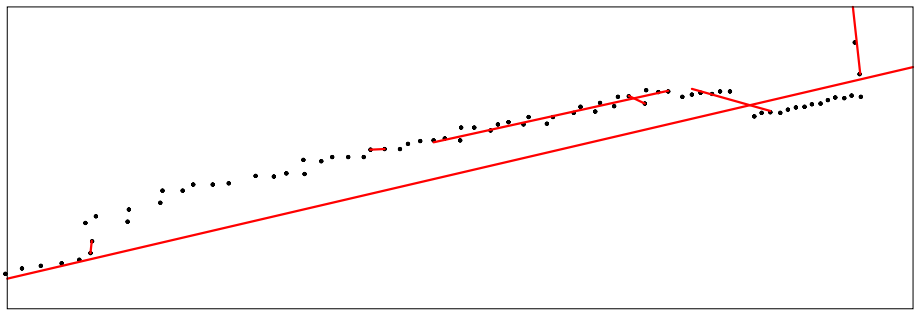



(d)WSAC–GE

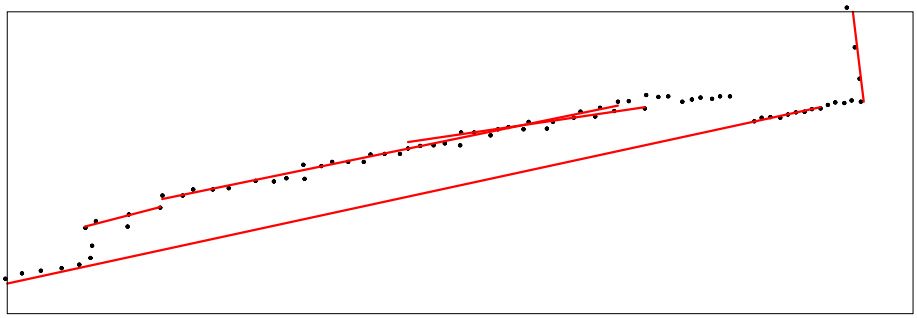
Figura 5.4: Prueba en el entorno real



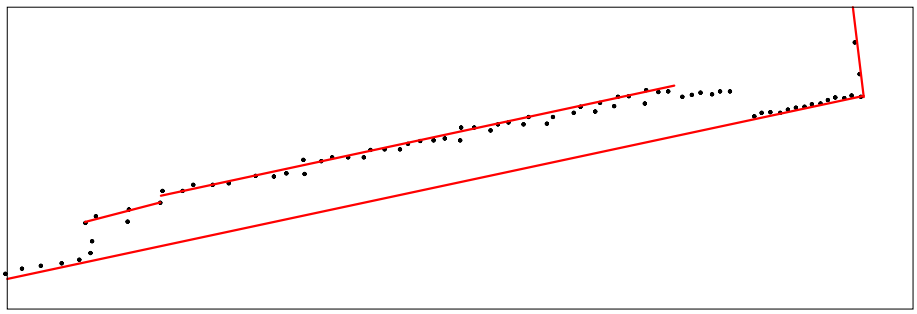
(a)Mapa a mano



(b)SMSM



(c)WSAC



(d)WSAC–GE

Figura 5.5: Prueba en el entorno real

(a)La puerta y la pared forman dos líneas paralelas muy próximas

(b)Detalle de la puerta

Figura 5.6: Entorno con líneas paralelas muy próximas

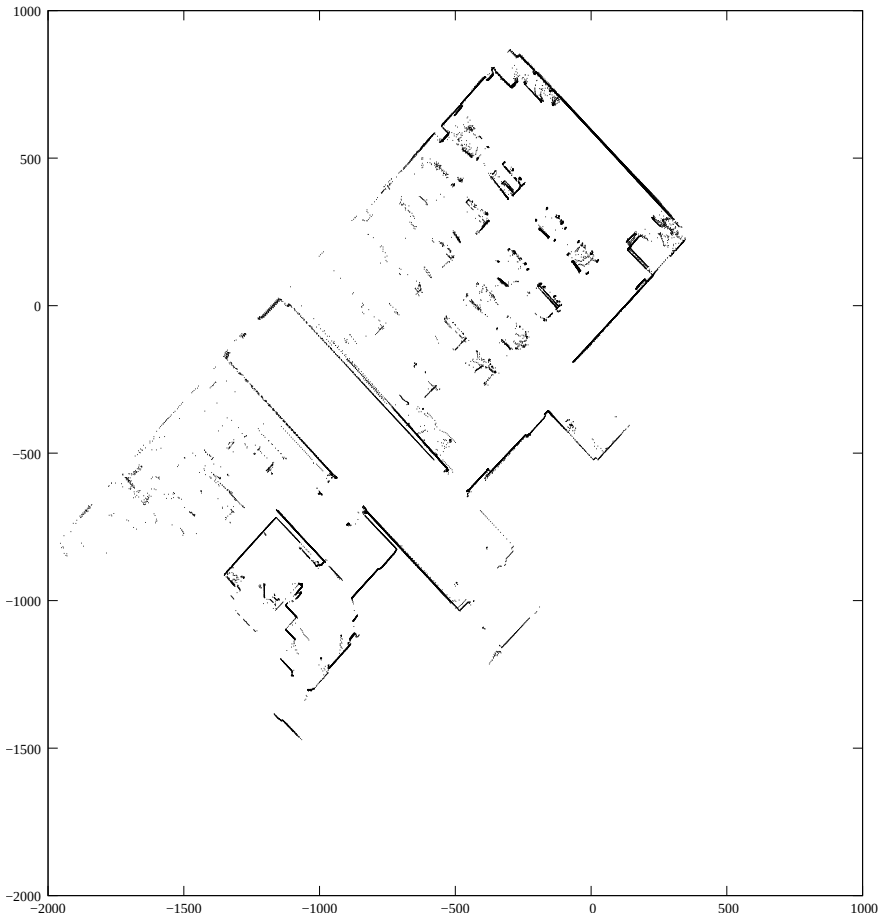



Figura 5.7: Mapa parcial generado en el entorno Real usando ICP [Arellano05]

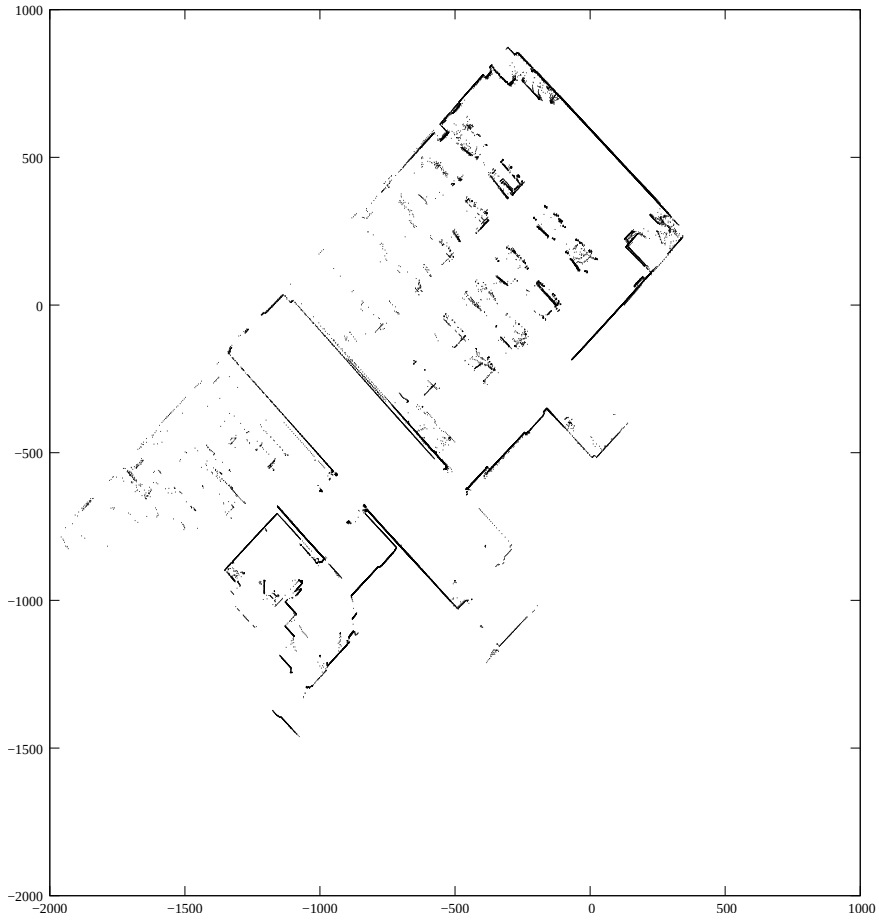



Figura 5.8: Mapa parcial generado en el entorno Real usando MGL

# Capítulo 6

# Conclusiones y trabajos futuros

Esta tesis propone dos nuevos métodos para la extracción de líneas a partir de imágenes de rango obtenidas con un láser. Ambos métodos en esencia adaptan el método de selección aleatoria en los siguientes aspectos: (1) mejoran la hipótesis de selección inicial para aprovechar la secuencialidad de las imágenes de rango; (2) usan una función de ponderación del consenso para rechazar puntos atípicos y lograr líneas soportadas por puntos muy cercanos y (3) utilizan umbrales suaves en lugar de rígidos para evaluar el soporte de una línea.

## 6.1. Discusión de los métodos propuestos

El método WSAC parte de una línea obtenida localmente y después considera todos los puntos dentro de la imagen de rango para mejorar la estimación. El método WSAC–GE usa además una función de evaluación general del modelo de líneas.

Aunque existen otros métodos para la extracción de líneas tales como LT e IEPF, o algunos que usan un enfoque global tal como EM ó HT, los métodos propuestos están pensados para funcionar en entornos reales donde existe una gran cantidad de objetos y se dificulta la extracción de las líneas. Dada la gran cantidad de ruido en un ambiente, pueden existir casos donde no es posible obtener el 100 % de las líneas sin

errores. Los resultados experimentales han mostrado también las debilidades y fortalezas de los métodos y, en contraste con los métodos convencionales, los propuestos logran mayor cantidad de éxitos. En cuanto al tiempo que los algoritmos consumen, podemos decir que el método WSAC es un poco más rápido que WSAC–GE, pero ambos son aceptables para aplicaciones de tiempo real en robótica móvil. Por último, usando el método WSAC–GE se incrementa la detección de líneas reales.

Los mapas geométricos de líneas (MGLs) son mucho más compactos que los mapas de puntos. Sin embargo, los MGLs más que una manera de representar entornos, son una herramienta útil cuando se trata de encontrar la localización del robot.

## 6.2. Trabajos futuros

Del trabajo desarrollado en esta tesis surge la idea de encontrar un algoritmo que modele la incertidumbre de cada línea del MGL. La medida de incertidumbre de cada línea puede permitir el uso de algoritmos probabilísticos, como el filtro de Kalman, para mejorar la calidad del mapa gobal de 2D y éste a su vez la localización del robot.

Se pueden usar los MGLs para facilitar la localización del robot a pesar de que el ambiente sufra constantes cambios como el de personas que interactúen con el robot. Se sugiere estudiar algoritmos para la *alineación entre mapas geométricos de líneas* como alternativa al algoritmo de alineación entre mapas locales de puntos. Otra idea que se puede estudiar es la combinación de mapas geométricos con mapas topológicos para obtener mejores resultados. Para generar mapas topológicos se requiere un procesamiento de alto nivel para obtener marcas como puertas, ventanas, libreros, etc.

Los algoritmos desarrollados para la extracción de líneas en una imagen de rango en 2D pueden adaptarse para proponer un algoritmo que extraiga planos y otras superficies a partir de una imagen de rango tridimensional (imagen de rango 3D). Para obtener una imagen de rango 3D es necesario que el láser tome lecturas en

diferentes planos de medición, esto se logra al modificar la inclinación del *pant–tilt*. Si se reconocen las superficies de un entorno entonces se pueden generar mapas en tercera dimensión y se puede realizar la reconstrucción 3D. La reconstrucción 3D se puede enriquecer con la información de imágenes de las cámaras para agregar textura a las superficies.

# Apéndice A

# Ajuste de una línea a puntos usando WTLS

Después de calcular los errores ortogonales del punto a la línea (como se abordó en el 2). El ajuste por WTLS se formula como sigue:

$$E\left(\left(x_1, y_1\right), \ldots, \left(x_n, y_n\right)\right) = \sum_{i=1}^{n} a_i \left(x_i \cos\alpha + y_i \sin\alpha\right)^2 - r \tag{A.1}$$

Por lo que se debe minimizar

$$\min_{\alpha, r} E\left(\left(x_1, y_1\right), \ldots, \left(x_n, y_n\right)\right) \tag{A.2}$$

Donde $r, \alpha$ son los parámetros de la línea, $x_i, y_i$ son las coordenadas del $i$–ésimo punto; y $a_i$ representa la ponderación del $i$–ésimo punto. En este apéndice se aborda la solución de ecuación. Derivando la ecuación A.1 respecto de $\alpha$

$$\begin{aligned} \frac{\partial E}{\partial r} = 2 \sum_{i=1}^{n} a_i \left(x_i \cos\alpha + y_i \sin\alpha - r_i\right) &= 0 \\ \cos\alpha \sum_{i=1}^{n} a_i x_i + \sin\alpha \sum_{i=1}^{n} a_i y_i - r \sum_{i=1}^{n} a_i &= 0 \end{aligned}$$

despejando $r$ y haciendo $\widetilde{x} = \frac{\sum a_i x_i}{\sum a_i}$ y $\widetilde{y} = \frac{\sum a_i y_i}{\sum a_i}$ tenemos.

$$\begin{aligned} r\sum_{i=1}^{n} a_i &= \cos\alpha \sum_{i=1}^{n} a_i x_i + \sin\alpha \sum_{i=1}^{n} a_i y_i \\ r &= \cos\alpha \frac{\sum_{i=1}^{n} a_i x_i}{\sum_{i=1}^{n} a_i} + \sin\alpha \frac{\sum_{i=1}^{n} a_i y_i}{\sum_{i=1}^{n} a_i} \\ r &= \widetilde{x}\cos\alpha + \widetilde{y}\sin\alpha \end{aligned} \tag{A.3}$$

sustituyendo A.3 en A.2

$$\begin{aligned} E &= \sum_{i=0}^{n} a_i \left(x_i\cos\alpha + y_i\sin\alpha - \widetilde{x}\cos\alpha - \widetilde{y}\sin\alpha\right)^2 \\ &= \sum_{i=0}^{n} a_i \left[(x_i - \widetilde{x})\cos\alpha + (y_i - \widetilde{y})\sin\alpha\right]^2 \end{aligned} \tag{A.4}$$

Por otro lado, $\frac{\partial E}{\partial \alpha}$ e igualando a 0, tenemos

$$\begin{aligned} 2\sum_{i=0}^{n} a_i \left[(x_i - \widetilde{x})\cos\alpha + (y_i - \widetilde{y})\sin\alpha\right]\left[-(x_i - \widetilde{x})\sin\alpha + (y_i - \widetilde{y})\cos\alpha\right] &= 0 \\ -\sin\alpha\cos\alpha \sum_{i=0}^{n} a_i (x_i - \widetilde{x})^2 - \sin^2\alpha \sum_{i=0}^{n} a_i (x_i - \widetilde{x})(y_i - \widetilde{y}) + & \\ + \cos^2\alpha \sum_{i=0}^{n} a_i (x_i - \widetilde{x})(y_i - \widetilde{y}) + & \qquad \text{(A.5)} \\ \sin\alpha\cos\alpha \sum_{i=0}^{n} a_i (y_i - \widetilde{y})^2 &= 0 \qquad \text{(A.6)} \end{aligned}$$

haciendo $\widetilde{S}_{xx} = \sum_{i=1}^{n} a_i \left(\widetilde{x} - x_i\right)^2$, $\widetilde{S}_{yy} = \sum_{i=1}^{n} a_i \left(\widetilde{y} - y_i\right)^2$ y
$\widetilde{S}_{xy} = \sum_{i=1}^{n} a_i \left(\widetilde{x} - x_i\right)\left(\widetilde{y} - y_i\right)$

$$\begin{aligned} -\sin\alpha\cos\alpha\widetilde{S}_{xx} - \sin^2\alpha\widetilde{S}_{xy} + \cos^2\alpha\widetilde{S}_{xy} + \sin\alpha\cos\alpha\widetilde{S}_{yy} &= 0 \\ -\sin\alpha\cos\alpha\widetilde{S}_{xx} + \left(\cos^2\alpha - \sin^2\alpha\right)\widetilde{S}_{xy} + \sin\alpha\cos\alpha\widetilde{S}_{yy} &= 0 \end{aligned}$$

$$
\begin{aligned}
-\frac{1}{2}\sin 2\alpha \widetilde{S}_{xx} + \cos 2\alpha \widetilde{S}_{xy} + \frac{1}{2}\sin 2\alpha \widetilde{S}_{yy} &= 0 \\
-\frac{1}{2}\sin 2\alpha (\widetilde{S}_{xx} - \widetilde{S}_{yy}) + \cos 2\alpha \widetilde{S}_{xy} &= 0
\end{aligned}
$$

Por último agrupando términos obtenemos la solución para $\alpha$

$$
\begin{aligned}
\frac{1}{2}\sin 2\alpha (\widetilde{S}_{xx} - \widetilde{S}_{yy}) &= \cos 2\alpha \widetilde{S}_{xy} \\
\tan 2\alpha &= \frac{2\widetilde{S}_{xy}}{\widetilde{S}_{xx} - \widetilde{S}_{yy}}
\end{aligned}
$$

de esta manera, la solución es

$$
\alpha = \frac{1}{2}\arctan\left(\frac{-2\widetilde{S}_{xy}}{\widetilde{S}_{yy} - \widetilde{S}_{xx}}\right) \tag{A.7}
$$

$$
r = \widetilde{x}\cos(\alpha) + \widetilde{y}\sin(\alpha) \tag{A.8}
$$

donde

$$
\begin{aligned}
\widetilde{S}_{xx} &= \sum_{i=1}^{n} a_i \left(x_i - \widetilde{x}\right)^2 & \widetilde{x} &= \frac{\sum_{i=1}^{n} a_i x_i}{\sum_{i=1}^{n} a_i} \\
\widetilde{S}_{yy} &= \sum_{i=1}^{n} a_i \left(y_i - \widetilde{y}\right)^2 & \widetilde{y} &= \frac{\sum_{i=1}^{n} a_i y_i}{\sum_{i=1}^{n} a_i} \\
\widetilde{S}_{xy} &= \sum_{i=1}^{n} a_i \left(x_i - \widetilde{x}\right)\left(y_i - \widetilde{y}\right)
\end{aligned}
$$

.

# Apéndice B

# Evaluación de similitud entre líneas basados en sus parámetros geométricos

En éste apéndice explica la forma de medir la similaridad entre dos líneas usando sus parámetros en coordenadas polares $\Theta = [\alpha, r]$. Supongamos que tenemos dos líneas $l_1 : \Theta_1$ y $l_2 : \Theta_2$ en el mismo sistema coordenado. Para comenzar, si $\Theta_1 = \Theta_2$ entonces dichas líneas son iguales. Sin embargo, como lo muestra la figura, si $\Theta_1 \neq \Theta_2$ no necesariamente implica que $l_1 \neq l_2$. Las líneas $l_1$ y $l_2$ son iguales, pero sus parámetros no lo son. Por lo que, en el sistema de representación polar, se vuelve necesaria la normalización de los parámetros antes de poder comparar su similitud.

Una forma práctica de evaluar la similitud entre líneas es obtener las diferencias de los parámetros entre las líneas.

$$\Delta r = r_1 - r_2 \tag{B.1}$$

$$\Delta \alpha = \alpha_1 - \alpha_2 \tag{B.2}$$

Las ecuaciones B.1 y B.2 son respectivamente la diferencia de las distancias y la diferencia angular entre las líneas $l_1$ y $l_2$. Una forma más robusta de evaluar la similitud introducida en [Einsele01] esta dada por la ecuación

$$q(l_1, l_2) = \frac{1}{1 + (c_r \Delta r)^2 + (c_\alpha \Delta \alpha)^2} \tag{B.3}$$

Donde $c_r, c_\alpha \in \mathbf{R}^2$ son dos constantes que deciden cuál es la diferencia angular y en distancia, respectivamente.

# Apéndice C

# Simulador 2D de un Robot equipado con un láser

En el presente apéndice muestra detalles acerca del diseño, características y uso del simulador empleado en las pruebas de ambiente sintético.

## C.1. Diseño

El simulador está implementado en Java y esencialmente está basado en las siguientes cuatro clases.

**RobotD.** Esta clase implementa la funcionalidad del robot. El robot es una composición de uno o varios láser y un anillo de sonares.

**Entorno.** Esta clase almacena los objetos que existen en el entorno. En esta versión la clase entorno es simplemente un conjunto de objetos de la clase Línea.

**Recorrido.** Esta clase sirve para guardar una secuencia de acciones que deben de ser ejecutadas por el robot en una simulación.

**Simulador.** Este es el programa principal, usa la tecnología swing. En realidad es una subclase de la clase JFrame y sirve de interfaz al usuario.

Las figuras C.1 y la C.2 muestran las clases RobotD y Ambiente respectivamente. El recorrido es simplemente una lista de acciones.

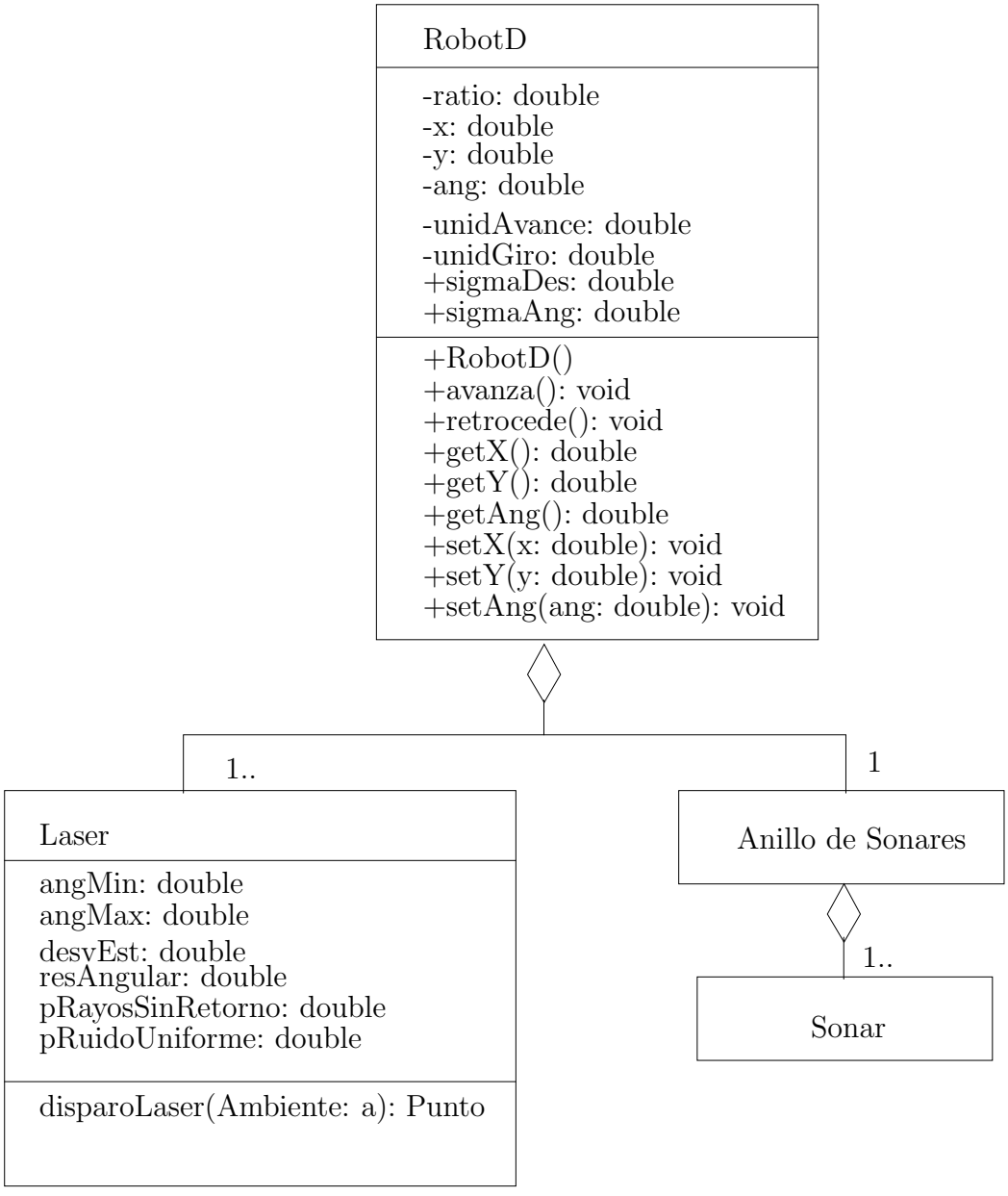



Figura C.1: La clase RobotD

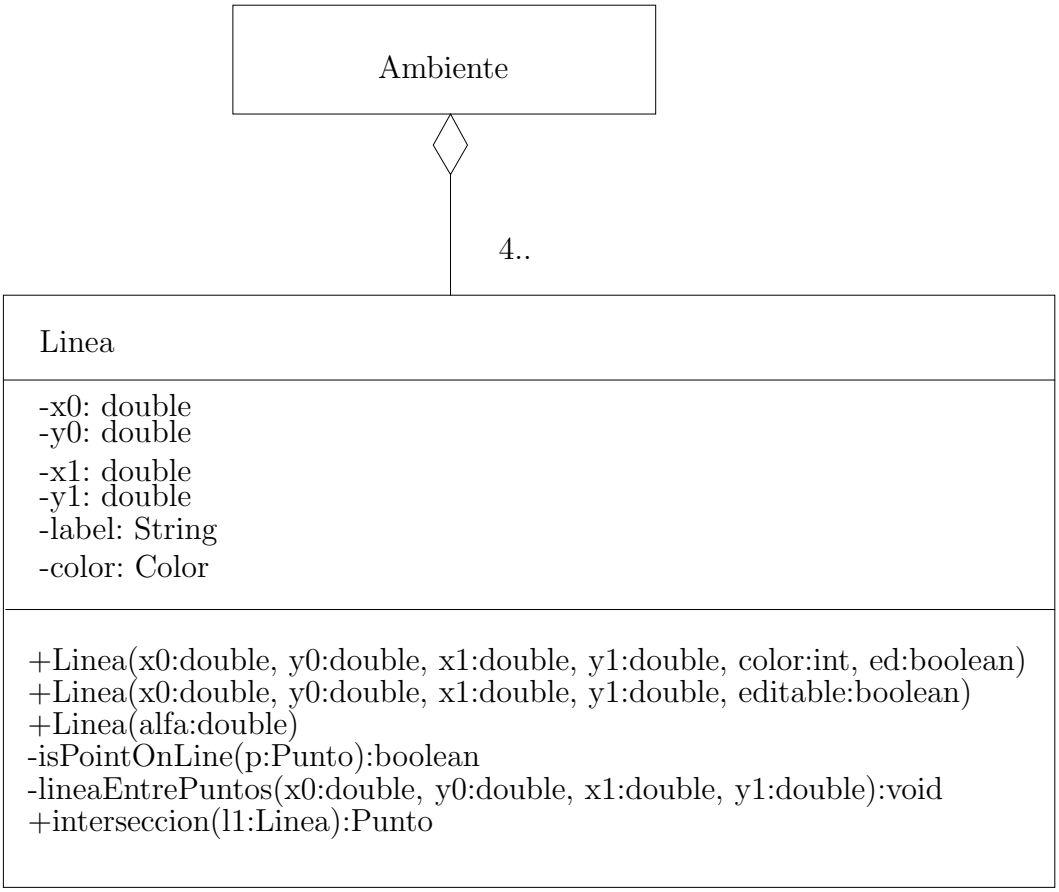



Figura C.2: La clase Ambiente

## C.2. Tipos de errores de medición reportados

En cada lectura, el láser simulado de nuestro programa reporta una medición según una de las tres situaciones siguientes:

**Caso A. Lectura errónea con valor máximo.** En este caso el rayo emitido por el láser no regresa en el tiempo máximo esperado. Este fenómeno ocurre en superficies traslúcidas u opacas y depende del ángulo de incidencia del rayo y la medición reportada por el láser es el valor máximo (v.g. 32 m).

**Caso B. Lectura corta real.** Este caso ocurre cuando el rayo láser es interceptado por algún obstáculo pequeño (v.g. patas de mesas o sillas) antes de que llegue a

cualquiera de los obstáculos lineales del entorno. Este tipo de lecturas se simulan usando una distribución uniforme evaluada entre cero y la distancia a la que se encuentre el obstáculo, y

**Caso C. Lectura normal.** Cuando la medición del obstáculo es exitosa, en este caso el simulador simplemente agrega un error gaussiano con media cero y varianza $\sigma_r^2$ a la lectura real.

En la realidad, los casos A, B, y C ocurren de acuerdo a factores como: el ángulo de incidencia del rayo sobre la superficie del objeto, la distancia a la que se realizó la medición, la opacidad y la rugosidad de la superficie, entre otros.

## C.3. Uso del programa

El requisito principal para ejecutar el programa es tener instalada la *Máquina Virtual de Java*. El primer paso consiste en descomprimir el archivo **simulador.zip** que se incluye en el CD. El simulador se ejecuta introduciendo lo siguiente en la línea de comandos desde el directorio donde se descomprimeron los archivos: `java Simulador`.

En las siguientes secciones se describe brevemente el uso del programa.

### C.3.1. El entorno gráfico

El entorno gráfico se muestra en la figura C.3 y consta de las siguientes partes:

- Barra de menús
- Barra de acciones
- Lista de líneas
- Área del entorno simulado

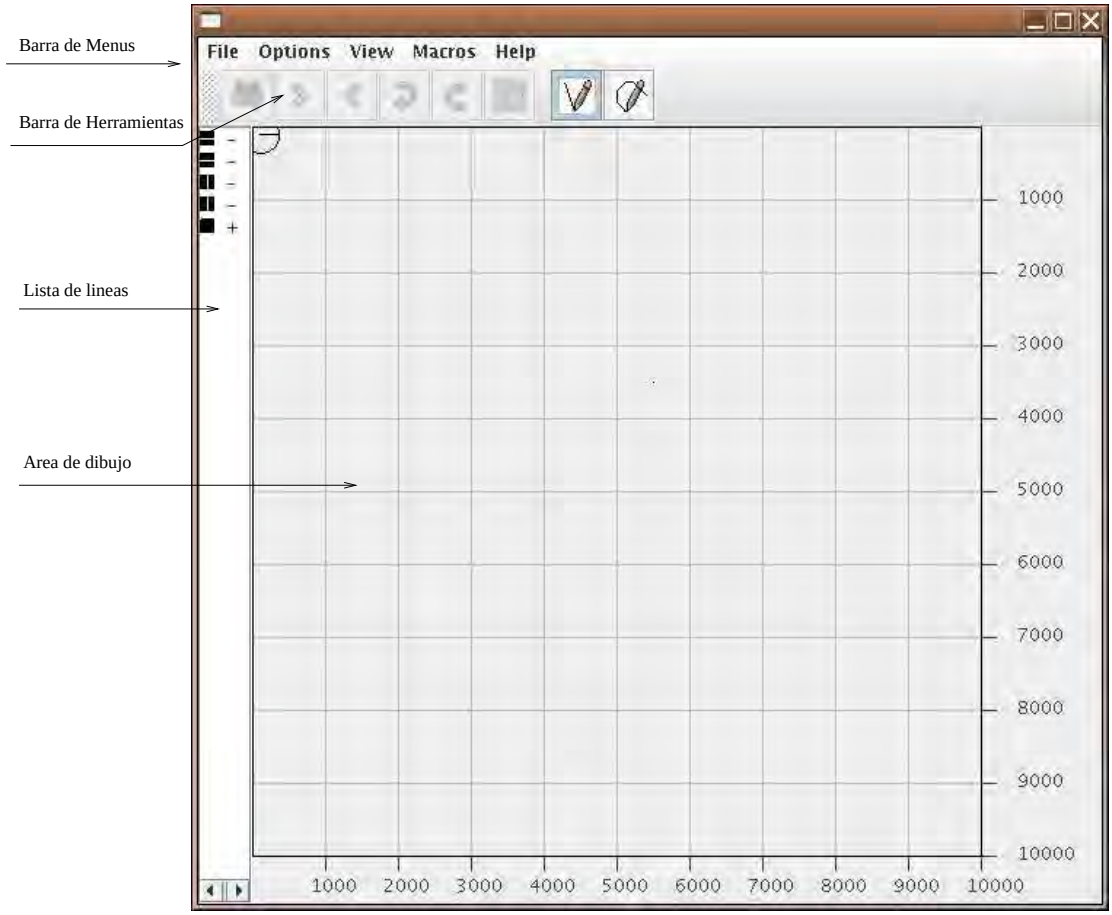



Figura C.3: La pantalla del Simulador

## C.3.2. Generación de simulaciones

Generar una simulación consiste de tres pasos: configurar el entorno, guardar las acciones del robot y ejecutar la simulación. Estos pasos se describen a continuación.

**Configurar el entorno** Esta fase consiste en crear los objetos que forman parte del entorno. Para configurar el tamaño del entorno use el menú *Options* ▷ *Environment* y para modificar la resolución del mismo use el menú *Options* ▷ *Grid*.

En esta versión solo se pueden crear objetos con contornos lineales. Las líneas se pueden crear en dos modos: líneas o polígonos. Para activar cualquiera de

| | |
|---|---|
|  | modo de líneas |
|  | modo de polígonos |

estos modos pulse el botón de la barra de acciones de acuerdo a la acción que desee como lo muestra la tabla C.3.2

**En el modo líneas** Este modo crea una línea a la vez. Para introducir una línea se pulsa el botón primario en la posición inicial y se arrastra hasta la posición final.

**En el modo polígonos** Este modo crea un polígono. Para introducir un nuevo polígono pulse el botón primario en los vértices del polígono. Para introducir el último punto pulse el botón secundario del ratón. Después de introducir el último punto del polígono se agregarán las líneas que lo forman a lista de líneas. Para cancelar la captura de un polígono se pulsa el botón central del ratón.

Para eliminar una línea seleccione la línea que desea eliminar de la lista de líneas y pulse la tecla *backspace*.

**Acciones del robot** Una vez creado el entorno se debe indicar las acciones que ejecutará el robot en la simulación. Existen dos modos de guardar las acciones:

**Por acción**. Este modo guarda una serie de acciones ordenadas que el robot debe ejecutar en el momento de la simulación, el usuario debe indicar cada acción que el robot debe realizar. Para iniciar este modo use el menú *Macro¿Record Macro*. Las acciones del robot para esta versión incluyen: avance, retroceso, disparo de láser, giro en sentido a las manecillas del reloj, giro en contra de las manecillas del reloj. Para indicar la posición inicial del robot use el menú *Options* ▷ *Robot* y para grabar una acción del robot se debe pulsar el botón de la acción correspondiente según lo indica la tabla C.1.

| Botón | Acción |
|---|---|
|  | Avance |
|  | Retroceso |
|  | Disparo de láser |
|  | Giro en sentido a las manecillas del reloj |
|  | Giro en contra de las manecillas del reloj |

Tabla C.1: Botones de la barra de herramientas

**Por trayectoria**. Este modo sirve para introducir una trayectoria que el robot seguirá el robot. Cuando se simule una trayectoria, el robot calcula automáticamente su orientación e incremento de avance después de cada paso de avance el robot disparará el láser. Para iniciar este modo use el menú *Macro* ▷ *Record Macro using Path*. Para capturar los vértices del polígono que indica el recorrido se pulsa el botón primario del botón y para introducir el último punto del recorrido se pulsa el botón secundario del ratón.

**Ejecución de las acciones** Una vez cargados el entorno y las acciones que se desean realizar se puede proceder a la simulación. Para ejecutar las acciones se usa el menú *Menu* ▷ *Run Macro*, con lo que el programa pedirá el nombre del archivo de salida (.dat). Después de haber introducido el nombre se iniciará la simulación.

El archivo de salida almacena en formato texto la simulación. Cuando se realiza un disparo del láser se almacena: los parámetros de posición del robot (ángulo, posición en X, posición en Y), seguido por la imagen de rango $S$ en formato polar. La figura C.4 muestra un fragmento del archivo de salida, el vector de posición del robot desde donde se tomó la primera imagen

de rango es $\begin{bmatrix} x & y & \theta \end{bmatrix}^T = \begin{bmatrix} 1000 & 5000 & 71.6 \end{bmatrix}^T$ y la imagen de rango es $R = \{(\alpha_i, \rho_i) | i = 1 \ldots 361\} = \{(-90.0, 3161.5), (-89.5, 4288.7) \ldots\}$, donde $\alpha_i$ es el ángulo de la $i$–ésima lectura en grados y $\rho_i$ es la distancia de la lectura en mm. De la misma manera, la siguiente imagen de rango se tomó en la posición del robot $\begin{bmatrix} x & y & \theta \end{bmatrix}^T = \begin{bmatrix} 1031.3 & 5093.8 & 71.6 \end{bmatrix}^T$ y la imagen de rango es $R = \{(\alpha_i, \rho_i) | i = 1 \ldots 361\} = \{(-90.0, 3935.6), (-89.5, 3553.0) \ldots\}$.

```
1000.0 5000.0 71.6
-90.0 3161.5 -89.5 4288.7 -89.0 3336.7 -88.5 6348.1 -88.0 3315.7
-87.5 1214.7 -87.0 3768.5 -86.5 32000.0 -86.0 5667.9 -85.5 32000.0
-85.0 4304.4 -84.5 4466.2 -84.0 32000.0 -83.5 32000.0 -83.0 341.6
-82.5 3792.6 -82.0 5525.0 -81.5 5792.5  -81.0 11659.0 -80.5 32000.0
-80.0 11629.5 -79.5 11608.7 -79.0 11596.5 -78.5 11585.6

⋮

1031.3 5093.8 71.6
-90.0 3935.6 -89.5 3553.0 -89.0 3649.5 -88.5 3752.9 -88.0 3864.9
-87.5 3983.4 -87.0 4108.6 -86.5 4245.6 -86.0 5645.3 -85.5 6504.2
-85.0 4705.5 -84.5 4884.1 -84.0 7057.9 -83.5 5289.3 -83.0 32000.0
-82.5 32000.0 -82.0 6037.5 -81.5 11645.0 -81.0 11628.8 -80.5 11612.4
-80.0 11595.0 -79.5 2837.3 -79.0 11564.4 -78.5 11759.4

⋮
```

Figura C.4: El archivo de salida del simulador

# Referencias

[Alempijevic04] Alempijevic, A. High speed feature extraction in sensor coordinates for laser rangefinders. *En Australasian Conference on Robotics and Automation 2004*. 2004.

[Arellano05] Arellano, J. *Reconstrucción tridimensional usando lecturas de un telémetro láser*. Tesis de Maestría, Universidad Michoacana de San Nicolás de Hidalgo, 2005.

[Arras97] Arras, K. y Siegwart, R. Feature–extraction and scene interpretation for map–based navigation and map–building. *En SPIE, Mobile Robotics XII*. 1997.

[Arras02] Arras, K., Castellanos, y Siegwart, R. Feature–based multi–hypothesis localization and tracking form mobile robots using geometric constraints. *En Proceedings of the IEEE International Conference on Robotics and Automation, 2002*. 2002.

[Arts04] Arts, T. Technological Arts, 2004. http://technologicalarts.com/. Updated: 2004/09/22.

[Baltzakis02] Baltzakis, H. y Trahanias, P. An iterative approach for building feature maps in cyclic environments. *En Proc. IEEE/RSJ Intlernational Conference on Intelligent Robotics and Systems (IROS)*, págs. 576–581. Lausanne, Switzerland, sep. 2002.

[Beckman83] Beckman, R. J. y Cook, R. D. Outlier . . . . . s. *Technometrics*, 25(2):120 – 179, May 1983.

[Bernholt05] Bernholt, T. Computing the least median of squares estimator in time $O(n^d)$. *En Proceedings of ICCSA 2005, LNCS*. 2005.

[Besl92] Besl, P. J. y McKay, N. D. A method for registration of 3D shapes. *IEEE Trans. PAMI*, 14(2):239–256, 1992.

[Boehler03] Boehler, W., Vincent, M. B., y Margs, A. Investigating laser scanner accuracy. *XIX CIPA Symposium at Antalya Turkey*, October 2003.

[Borenstein96] Borenstein, J., Everett, B., y Feng, L. *Navigating Mobile Robots: Systems and Techniques*. A.K. Peter, Ltd., Wellesley, MA, 1996.

[Borges04] Borges, G. A. y Aldon, M.-J. Line extraction in 2d range images for mobile robotics. *J. Intell. Robotics Syst.*, 40(3):267–297, 2004. ISSN 0921-0296. doi:http://dx.doi.org/10.1023/B:JINT.0000038945.55712.65.

[Castellanos96] Castellanos, J. y Tardós, J. Laser-based segmentation and localization for a mobile robot. *Robotics and Manufacturing*, 6:101–108, 1996. ISSN 0-7918-0047-4.

[Castro04] Castro, D., Nunes, U., y Ruano, A. Feature extraction for moving objects tracking system in indoor environments. *En Proc. 5th IFAC/euron Symposium on Intelligent Autonomous Vehicles*. 2004.

[Chen91] Chen, Y. y Medioni, G. Object modeling by registration of a multiple range images. *En Proceedings of the 1991 IEEE International Conference on robotics and Automation*. 1991.

[Dempster77] Dempster, A. P., Laird, N. M., y Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm (with discussion). *Journal of the Royal Statistical Society*, 1977.

[Duda76] Duda, R. O. y Hart, P. E. *Pattern Classification and Scene Analysis*. Wiley-Interscience, 1976.

[Dudek00] Dudek, G. y Jenkin, M. *Computational Principles of Mobile Robotics*. Cambridge University Press, 2000.

[Einsele01] Einsele, T. *Localization in Indoor Environments using a Panoramic laser range finder*. Tesis Doctoral, Technical University of München, 2001.

[Filliat03] Filliat, D. y Meyer, J. Map–based navigation in mobile robots–i. a review of localization strategies. *Journal of Cognitive Systems Research*, 2003.

[Fischler81] Fischler, M. A. y Bolles, R. C. Random Sample Consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/358669.358692.

[Forsyth03] Forsyth, D. A. y Ponce, J. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[Gätcher05] Gätcher, S., Nguyen, V., y Siegwart, R. Results on range image segmentation for service robot. Inf. téc., Laboratoire de Systèmes Autonomes, Ecole Polytechnique Fèdèrale de Laussane, 2005.

[Giesler98] Giesler, B., Graf, R., Dillmann, R., y Weiman, C. Fast mapping using the Log-Hough transformation. *En IEEE/RSJ International Conference on Intelligent Robots and Systems*. 1998.

[Gillies92] Gillies, D. F. y Khan, G.Ñ. Perceptual grouping and the hough transform. tomo 1607, págs. 188–196. SPIE, 1992.
URL `http://link.aip.org/link/?PSI/1607/188/1`

[Goldenshluger04] Goldenshluger, A. y Zeevi, A. The Hough transform estimator. *Annals of Statistics*, 2004.

[GPL91] GNU general public license, 1991. http://www.gnu.org/licenses/gpl.html.

[Gutmann96] Gutmann, J. y Schlegel, C. Amos: Comparison of scan matching approaches for self-localization in indoor environments. *En Proc. of the Ist Euromicro Workshop on Advance Mobile Robots*. IEEE Computer Society Press, 1996.

[Hähnel02] Hähnel, D., Burgard, W., y Thrun, S. Learning compact 3D models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, 2002. http://www-2.cs.cmu.edu/ thrun/papers/haehnel.indoor-outdoor.html.

[Hampel86] Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., y Stahel, W. A. *Robust Statistics: The Approach Based on Influence Functions*. John Wiley and Sons, New York, N.Y, 1986.

[Hartley03] Hartley, R. y Andrew, Z. *Multiple View Geometry in computer vision*. Cambridge University Press, 2003.

[Hough59] Hough, P. Machine analysis of bubble chamber pictures. *En International Conference on High Energy Accelerators and Instrumentation*. 1959.

[Huber64] Huber, P. Robust location of a location parameter. *Annals of Mathematical Statistical*, 35, 1964.

[Ivan99] Ivan, M. y H., M. C. Breakdown points and variation exponents of robust m–estimators in linear models. *The Annals of Statistics*, 1999.

[L. Jones98] L. Jones, J., M. Flynn, A., y A. Seiger, B. *Mobile Robots*. Cambridge University Press, 1998.

[Latecki04] Latecki, L. J., Lakaemper, R., Sun, X., y Wolter, D. Building polygonal maps from laser range data. *ECAI Int. Cognitive Robotics Workshop*, Aug 2004.

[Latecki06] Latecki, L. J. y Lakaemper, R. Polygonal approximation of laser range data based on perceptual grouping and EM. *En IEEE Conf. on Robotics and Automation*. 2006.

[Leonard90] Leonard, J., Durrant-Whyte, H., y Cox, I. J. Dynamic map building for an autonomous mobile robot. *IEEE International Workshop on Intelligent Robots and Systems*, 1:89–96, 1990.

[Liu01] Liu, Y., Emery, R., Chakrabartl, O., Burgard, W., y Thrun, S. Using EM to learn 3D models with mobile robots. *En Proceedings of the International Conference on Machine Learning (ICML)*. 2001.

[Lu97] Lu, F. y Milios, E. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 18:249–275, 1997.

[MacQueen67] MacQueen, J. Some methods for classification and analysis of multivariate observations. *En Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. 1967.

[Matsumoto99] Matsumoto, Y., Ikeda, K., Inaba, M., y Inoue, H. Exploration and map acquisition for view-based navigation in corridor environment. *En Proceedings of the International Conference on Field and Service Robotics*, págs. 341–346. 1999.

[McLachlan97] McLachlan, G. J. y Krishnan, T. *The EM Algorithm and Extensions*. Wiley–Interscience, 1997.

[Meer00] Meer, P., Stewart, C. V., y Tyler, D. E. Introduction, robust computer vision: An interdisciplinary challenge. *Computer Vision and Image Understanding*, 2000.

[Miles04] Miles, A. *Automatic 3D registration using range and colour data.* Tesis de Maestría, Carleton University, School of Computer Science, Ottawa-Carleton Institure for Computer Science, mayo 2004.

[Miller99] Miller, I. *John E. Freund's mathematical statistics.* Prentice Hall, 6ª ed., 1999.

[Myatt02a] Myatt, D. et al. NAPSAC: High noise, high dimensional robust estimation. *En British MAchine Vision Conference.* 2002.

[Myatt02b] Myatt, D. R. Robust estimation in high noise and highly dimensional data sets with application to machine vision. Inf. téc., University of Reading, October 2002.

[Nguyen05] Nguyen, V., Martinelli, A., Tomatis, N., y Siegwart, R. A comparison of line extraction algorithms using 2d. laser rangefinder for indoor mobile robotics. *En International Conference on Intelligent Robots and Systems (IROS 2005).* 2005.

[Núñez04] Núñez, A. *Fusión de lecturas de un telémetro láser con un sistema de visión estéreo trinocular para detectar obstáculos en tres dimensiones.* Tesis de Maestría, Universidad Michoacana de San Nicloás de Hidalgo, Facultad de Ingeniería Eléctrica, División de Estudios de Posgrado, Abril 2004.

[Pfister02] Pfister, S., Kriechbaum., K., Roumeliotis, S., y Burdick, J. Weighted range sensor matching algorthms for mobile robot displacement estimation. *En IEEE Int. Conf. on Robotics and Automation.* 2002.

[Premebida05] Premebida, C. Segmentation and geoemetric primitives extraction from 2d laser range data for mobile robot applications. *En Scientific meeting of the 5th National robotics festival.* 2005.

[Romero01] Romero, L. *Construcción de mapas y localización de robots móviles: un enfoque probabilísta*. Tesis Doctoral, Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Cuernavaca, nov. 2001.

[Roumeliotis00] Roumeliotis, S. y Bekey, G. Segments: A layered, dual-kalman filter algorithm for indoor feature extraction. *En 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2000.

[Rousseeuw87] Rousseeuw, P. J. y Leroy, A. M. *Robust regression & outlier detection*. John Wiley & Sons, Inc., New York, NY, USA, 1987. ISBN 0-471-85233-3.

[Sack03] Sack, D. y Burgard, W. A comparison of methods for line extraction from range data. *En Proc. of the 5th IFAC Symposium on Intelligent Autonomous Vehicle (IAV)*. 2003.

[SIC] SICK, INC., 6900 West 110th street, Bloomington mn 55438 USA. *Proximity Laser Scanner PLS, Technical Description*. Http://www.sickoptic.com.

[Siegwart04] Siegwart, R. *Introduction to Autonomus Mobile Robots*. The MIT Press, 2004.

[Stewart93] Stewart, C. V. A new robust operator for computer vision: Theoretical analysis. Inf. téc., Rensselaer Polytechnic Institute, March 1993.

[Stewart95] Stewart, C. V. Minpran: A new robust operator for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995.

[Stewart99] Stewart, C. V. Robust parameter estimation in computer vision. *SIAM Review*, 41(3):513–537, 1999.
URL `citeseer.ist.psu.edu/stewart99robust.html`

[Stigler86] Stigler, S. M. *The History of Statistics: The Measurement of Uncertainty Before 1900*. Harvard University Press, Cambridge, Massachusetts, 1986.

[Thrun98] Thrun, S., Burgard, W., y Fox, D. A probabilistic approach to concurrent mapping and localization for mobile robots. *En Auntonomous Robots*. 1998.

[Thrun00] Thrun, S., Burgard, W., y Fox, D. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. *En Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, San Francisco, CA, 2000.

[Thrun02] Thrun, S. Robotic mapping: A survey. *En* G. Lakemeyer y B.Ñebel, eds., *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaulmann, 2002.

[Torr98] Torr, P. H. S. y Zisserman, A. Robust computation and parameterization of multiple view relations. *En ICCV6*, págs. 727 – 732. 1998.

[Torr00] Torr, P. y Zisserman, A. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78:138–156, 2000.
URL `citeseer.ist.psu.edu/torr96mlesac.html`

[Weiman94] Weiman, C. F. Application of log-hough transform to lidar navigation, phase i sbir final report. Inf. téc., HelpMate Robotics Inc., 1994.

[Williams02] Williams, M. Building the linux of the robot world. *PCWorld*, mar. 2002.

[Yohai79] Yohai, V. J. Regresión robusta. CEMA Working Papers 9, Universidad del CEMA, dic. 1979. Available at http://ideas.repec.org/p/cem/doctra/9.html.

[Zezhong03] Zezhong, X., Jilin, L., y Zhiyu, X. Map building and localization using 2d range scanner. *En Proceedings 2003 IEEE International Symposium*

*on Computational Intelligence in Mobile Robotics and Automation*, págs. 848–853. 2003.

[Zhang94] Zhang, Z. Iterative point matching for registration of free-form curves and surfaces. *Int. J. Comput. Vision*, 13(2):119–152, 1994. ISSN 0920-5691. doi:http://dx.doi.org/10.1007/BF01427149.

[Zhang95] Zhang, Z. Parameter estimation techniques: A tutorial with application to conic fitting. Inf. Téc. RR-2676, INRIA Sophia Antipolis, 1995. URL `citeseer.ist.psu.edu/article/zhang97parameter.html`