DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Mapeo y localización simultáneos en forma robusta basado en visión estéreo

Tesis de Licenciatura

Sergio Juan Cazzolato

Directora de tesis: Marta Mejail
Buenos Aires, 2007

RESUMEN

Para manejar el "problema del robot secuestrado", en el cual un robot es transportado a alguna ubicación desconocida sin ninguna información acerca de su posición y se requiere que éste genere un mapa del ambiente que lo rodea, se utiliza un algoritmo de generación de mapas y localización.

Mapeo y localización simultáneos (SLAM) es una técnica usada por robots y vehículos de navegación para construir un mapa de un ambiente desconocido mientras al mismo tiempo se actualiza la posición del mismo. Como fuente de información para realizar SLAM se pueden utilizar varios tipos de sensores: láser, sonar, cámaras, etc.

Como mecanismo para obtener información del ambiente puede usarse un sistema de visión estéreo, a partir del cual se capturan simultáneamente pares de imágenes que son utilizadas para obtener una representación tridimensional de los objetos del ambiente.

La técnica SLAM presenta grandes dificultades a vencer utilizando sistemas de visión como únicos sensores. El principal problema se encuentra en las distintas fuentes de error que afectan al método. Estas fuentes de error generan inexactitudes en los mapas, que en muchos casos pueden ser muy groseras, ya que suelen ser acumulativas.

Las dos fuentes de error más importantes provienen de la medición incorrecta de los puntos en las imágenes y de la detección incorrecta de correspondencias. Para tratar dichos problemas se utilizan algoritmos de minimización del error en el cálculo de la posición y comparación robusta respectivamente.

En este trabajo se propone un algoritmo de SLAM robusto, basado en un sistema de visión estéreo como única fuente de información, siendo el principal objetivo minimizar el efecto negativo producido por las principales fuentes de error.

ABSTRACT

In order to handle the problem of the kidnapped robot, in which a robot is transported to some location unknown without no information about its position and it is required that this one generates a map of the atmosphere that surrounds it, is used an algorithm of generation of maps and location.

Simultaneous Location And Mapping (SLAM) is a technique used by robots and vehicles of navigation to construct a map of a unknown environment while at the same time the position of he himself is updated. As source of intelligence to make SLAM several types of sensors can be used: laser, sonar, cameras, etc.

As mechanism to obtain data of the environment can be used a stereo vision system, from which they are captured simultaneously even of images that are used to obtain a three-dimensional representation of the objects of the environment.

Technique SLAM presents great difficulties using stereo vision system like only sensors. The main problem is in the different sources from error that affect the method. These sources of error generate inaccuracies in the maps, that in many cases can be very crude, since usually they are cumulative.

The two more important sources of error come from the incorrect measurement of the points in the images and the incorrect detection of correspondences. In order to treat these problems algorithms of minimización of the error in the calculation of the position and robust comparison are used respectively.

In this work we propose a robust algorithm of SLAM, based on a stereo vision system like only source of information, being the main objective for diminishing the negative effect produced by the main sources of error.

ÍNDICE GENERAL

1. INTRODUCCIÓN

Es necesario resolver el problema de mapeo y localización simultáneos conocido como SLAM (Simultaneous Localization And Mapping) para lograr la autonomía del movimiento de los robots.

La localización exacta del robot es un prerrequisito para la construcción de un buen mapa y para tener un mapa exacto del ambiente es esencial tener una buena localización del robot. Por lo tanto, se requiere localizar el robot y construir un mapa en forma simultánea. Este proceso es un factor crítico para la navegación de un robot en grandes ambientes.

Existen dos tipos de localización: local y global. Las técnicas locales sirven para compensar los errores de odometría durante la navegación del robot. Estas requieren que la ubicación inicial del robot sea conocida aproximadamente. Las técnicas globales pueden localizar un robot sin ningún conocimiento a priori de su posición, es decir, ellas pueden manejar el "problema del robot secuestrado", en el cual el robot es transportado a alguna ubicación desconocida sin ninguna información acerca de su posición. Las técnicas de localización global son más potentes que las locales y permiten al robot recuperarse de errores en el posicionamiento.

Para lograr la construcción del mapa y localización simultánea, hay diferentes tipos de modalidades. Muchos sistemas usan marcas artificiales tales como reflectores de código de barras, patrones visuales, etc, y por lo tanto no funcionan bien cuando no hay marcas. Los sistemas que utilizan marcas naturales estables, son altamente recomendados en un amplio rango de aplicaciones. La construcción de mapas a partir de estas marcas naturales sirve como base para realizar tareas de alto nivel tales como navegación de un robot móvil.

La base de la generación de los mapas 3D se centra en las fuentes de información que se utilizan, es de esperar que dichas fuentes otorguen información

en forma confiable y eficiente para su posterior procesamiento.

La extracción de puntos claves o características de las imágenes utilizando marcas naturales es el primer paso con miras a obtener la información necesaria para poder estimar el movimiento del robot. En [7] se presenta un método para la extracción de características distintivas e invariantes de imágenes llamado SIFT (Scale-Invariant Feature Transform), estas características son invariantes a la escala y rotación de la imagen, y proveen gran robustez con respecto a otras problemas como cambios de iluminación, punto de vista, ruido, etc.

Es necesario obtener correspondencias entre las características extraídas de las distintas imágenes mediante el algoritmo SIFT para poder modelar el mundo 3D, para ello se utilizan algoritmos de búsqueda que son muy eficientes, como los presentados por J. Freidman, J. Bentley y R. Finkel en [2]. El resultado del algoritmo de búsqueda suele generar malas correspondencias, por lo que en muchos casos es necesario robustecer el mecanismo.

Existen distintos algoritmos de estimación robusta para realizar filtrado de correspondencias, como por ejemplo RANdom SAmple Concensus (RANSAC) presentado por Fischler y Bolles en [1].

Trabajos previamente realizados, como se muestra en [8], dan certeza de que resolver el problema de mapeo y localización simultáneos es posible si nos enfocamos en los sistemas de visión estéreo como únicos sensores para modelar el mundo en 3D.

En los trabajos [9] y [10], S. Se, D. Lowe y J. Little proponen utilizar marcas 3D basadas en el concepto de espacio escala para computar el mapa. En este caso se utiliza la odometría como método para lograr predecir el movimiento del robot en base a las marcas detectadas. Este mecanismo es extendido en [11], donde se aplican mecanismos para realizar correcciones posteriores que mejoran el resultado obtenido, y se propone la utilización de RANSAC para robustecer el matcheo de características. Posteriormente en [12] se propone un mecanismo basado en una partición en submapas para luego generar un mapa global.

En este trabajo se propone un algoritmo de SLAM robusto utilizando un sistema de visión estéreo como única fuente de información. Se utiliza el algoritmo SIFT presentado en el la sección 2.1 para realizar la detección

de características, y un nuevo algoritmo presentado en la sección 3.2 para robustecer la detección de correspondencias. A lo largo del trabajo se intenta minimizar el impacto negativo producido por las distintas fuentes de error.

Esta tesis se organiza de la siguiente manera:

En el capítulo 2, presentamos mecanismos para extracción de características, armado de descriptores y recuperación de descriptores mediante algoritmos eficientes de búsqueda. En la sección 2.1 se presenta el algoritmo SIFT, utilizado para la extracción de características basado en el concepto de Espacio Escala, y un mecanismo utilizado para generar y serializar descriptores. En la sección 2.2 se presentan algoritmos de almacenamiento y recuperación de descriptores.

En el capítulo 3 se presentan dos estimadores robustos, donde ambos se basan en el establecimiento de grupos de consenso a partir de los cuales poder realizar las estimaciones. En la sección 3.1 se describe el algoritmo llamado RANSAC, y en la sección 3.2 se presenta un nuevo algoritmo desarrollado en el presente trabajo llamado "Eliminación por pendiente y longitud".

En el capítulo 4 se describe cómo obtener la posición de un punto en el espacio, dada su imagen en dos vistas y los parámetros de la cámaras utilizadas para cada una de esas vistas. Este capítulo está basado en conceptos presentados por R. Hartley y A. Zisserman en [3].

Habiendo presentado una forma de obtener la posición de un punto en el espacio, se expone el algoritmo SLAM en el capítulo 5.

Finalmente, en los capítulos 6 y 7 presentamos y discutimos los resultados obtenidos, aportando conclusiones y posibles trabajos futuros.

2. EXTRACCIÓN DE CARACTERÍSTICAS Y GENERACIÓN DE CORRESPONDENCIAS

Para realizar la extracción de características y generación de descriptores utilizaremos el método SIFT (Scale-Invariant Feature Transform) presentado por David Lowe en [7]. Llamamos características a puntos en la imagen que presentan propiedades que son utilizadas para establecer correspondencias con otras caracerísticas. A partir de las características extraídas utilizaremos algoritmos eficientes para el almacenamiento y recuperación de sus respectivos descriptores. Finalmente presentaremos un algoritmo para generar un conjunto de correspondencias a partir de dos fuentes de descriptores.

2.1. Extracción de características

En esta sección se presenta el algoritmo SIFT, que utilizamos para realizar la extracción de características de una imagen y la generación de sus descriptores.

2.1.1. Espacio escala

La escala de las características en una imagen puede verse como el acercamiento que se utiliza para su detección. A priori no se conoce la escala ideal para una imagen, debido a ello se necesita contemplar un rango de escalas que puede ir desde la escala interior (determinada por el tamaño de los píxeles) a la escala exterior (determinada por el tamaño de la imagen).

Con el término espacio-escala nos referimos a una representación multi-escala de una imagen que posee un parámetro de escala contínuo y preserva el mismo nivel de muestreo para todas las escalas.

El concepto de espacio-escala lineal aporta una forma sistemática para

relacionar el concepto de escala con el de suavizado.

Para ir construyendo el espacio-escala nos basamos en la idea de introducir la señal original en una familia monoparamétrica de señales que derivan de ella, las cuales son construidas mediante la convolución con otra familia de kernels gaussianos monoparamétricos que poseen varianza creciente.

El espacio-escala de una función (o señal) unidimensional $f : \mathbb{R}^N \to \mathbb{R}$ es otra función $L : \mathbb{R}^N \times \mathbb{R}_+ \to \mathbb{R}$ tal que

$$L(\bullet, t) = \begin{cases} f & \text{si } t = 0 \\ g(\bullet, t) * f & \text{si } t \neq 0 \end{cases} \tag{2.1}$$

donde $t \in \mathbb{R}_+$ es el parámetro de escala y $g : \mathbb{R}^N \times \mathbb{R}_+ \setminus \{0\} \to \mathbb{R}$ es el kernel gaussiano:

$$g(x, t) = \frac{1}{(\pi t)^{N/2}} e^{\frac{-x^T x}{2t}} \qquad \text{con } x \in \mathbb{R}^N \tag{2.2}$$

A medida que crece la escala, el espacio-escala reduce la información fina. De modo intuitivo si se quisiera eliminar las estructuras de tamaño menor a σ se debería convolucionar la imagen con un kernel gaussiano con varianza σ^2.

2.1.2. Detección de marcas naturales

La detección de marcas naturales se basa en la propuesta por Lowe [5, 6, 7] donde se buscan máximos y mínimos locales de la función Diferencia de Gaussianas (DG) sobre una representación multiescala de pirámide.

La función DG de una imagen Im se define:

$$\begin{aligned} DG(x, t) &= (G(x, t + \Delta t) - G(x, t - \Delta t)) * Im(x) \\ &= L(x, t + \Delta t) - L(x, t - \Delta t) \end{aligned} \tag{2.3}$$

El orden de cómputo es de $O(kMN \log_2(MN))$ para una imagen de $M \times N$ y k niveles de escala. Esta eficiencia se debe a la representación del espacio-escala.

Para realizar la detección de marcas se realiza la búsqueda de extremos locales en una aproximación de la derivada de DG, que llamaremos dDG.

2.1.3. Descripción del filtrado

Dado que el número de extremos locales detectados en la etapa anterior puede ser significativamente alto, y a causa de esto las etapas posteriores de descripción y apareamiento se volverían más costosas de lo deseado, se necesita determinar un subconjunto representativo de las marcas encontradas en cada nivel.

Definimos la función $P : \mathbb{Z}^2 \to \mathbb{R}$ como la potencia de un punto en dDG tal que

$$P(x,t) = \left| dDG(x,t) - \frac{1}{(\#E) - 1} \sum_{\substack{y \in E \\ y \neq x}} dDG(y,t) \right| \tag{2.4}$$

donde E es un entorno de x.

Utilizamos esta función porque es más robusta frente a cambios de luminosidad:

- *globales:* al utilizar una diferencia de niveles de gris, es insensible a sumar una constante al nivel de gris de cada pixel.
- *locales:* el valor que toma es local, lo que reduce su sensibilidad a variaciones de luz no-lineales (por ejemplo, aplicar un foco en una zona de la imagen).

Para determinar el subconjunto buscado se realiza el siguiente mecanismo:

sea $\mathcal{C}_i$ el conjunto de extremos locales del i-ésimo nivel del espacio-escala, definimos la función $P^* : \mathbb{Z}^2 \to \mathbb{R}$ como

$$P^*(\mathcal{C}_i) = \frac{1}{\#E} \sum_{x \in \mathcal{C}_i} P(x,i) \tag{2.5}$$

donde P^* es la potencia media de los extremos del conjunto $\mathcal{C}_i$.

Seleccionamos finalmente el conjunto de $\mathcal{C}_i^F$ de extremos:

$$\mathcal{C}_i^F = \{x | x \in \mathcal{C}_i \wedge P(x,i) \geq kP^*(\mathcal{C}_i)\} \tag{2.6}$$

Es decir, seleccionamos los extremos de un nivel cuya potencia es k veces más grande que la potencia media de lo extremos del mismo nivel.

2.1.4. Descriptores neurológicos

A partir de la información obtenida es necesario describir la región de la imagen correspondiente a las marcas seleccionadas, obteniendo invarianza a traslaciones y cambio de escala. Para una punto p, se define la región R de tamaño NxN como el conjunto de puntos que se encuentran a una distancia menor a $\lceil N/2 \rceil$ de p.

El descriptor de un punto clave debe describir la región correspondiente a éste, mientras mantiene invarianza con respecto a: rotaciones, cambios de iluminación, puntos de vista 3D, etc.

Con el objeto de obtener invarianza a rotaciones, a cada descriptor se le asigna una orientación, para esto utilizaremos su gradiente. Para aproximar el gradiente mencionado, utilizamos diferencias finitas sobre la imagen a la que más cerca esté el punto en términos de escala.

Asignación de orientación

A cada punto clave se le asigna una orientación basada en propiedades locales de la imagen alrededor del punto. Esta orientación se utiliza para establecer invarianza a rotaciones de la imagen. Para ello se calcula el gradiente de la magnitud y de la orientación mediante las siguientes fórmulas:

$$\begin{aligned} \theta_{x,y} &= \tan^{-1}\left(\frac{Im_{x,y+1} - Im_{x,y-1}}{Im_{x+1,y} - Im_{x-1,y}}\right) \\ mg_{x,y} &= \sqrt{(Im_{x+1,y} - Im_{x-1,y})^2 + (Im_{x,y+1} - Im_{x,y-1})^2} \end{aligned}$$

Siendo $Im_{x,y}$ la imagen, $\theta_{x,y}$ la orientación y $mg_{x,y}$ la magnitud del gradiente con coordenadas (x,y).

Teniendo en cuenta la escala de un punto clave se calcula la orientación y magnitud del gradiente, también se efectúa el cálculo para todos los puntos pertenecientes a la región del mismo.

Luego de realizar el cálculo se aplica una función Gaussiana con el objeto de otorgarle mayor peso a las magnitudes de los puntos más cercanos al centro del descriptor. Esto se efectúa para darle énfasis a los puntos centrales del descriptor.

Con la información obtenida se genera un histograma con los gradientes de las orientaciones de la región alrededor de un punto clave. Cada punto de la región puede tomar un valor que cubre un rango dentro de los 360 grados. El peso de cada orientación en el histograma está dado por la magnitud de los gradientes de la región.

En la fig. 2.1 se muestra una cuadrícula donde cada flecha representa una orientación escalada teniendo en cuenta la magnitud del punto.

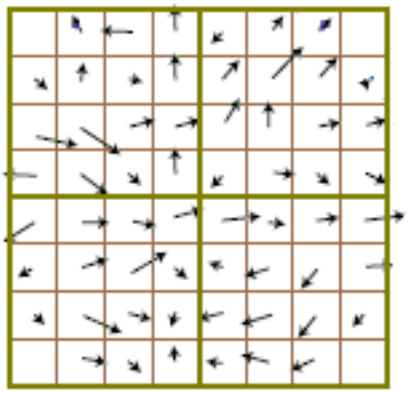

Fig. 2.1: Cuadrícula con los gradientes de orientaciones y magnitudes para cada punto de la región.

Los máximos en el histograma se corresponden con las direcciones dominantes de los gradientes. Inicialmente se detecta el máximo total, y los máximos con un valor superior al 80 % de éste. Con cada uno de ellos se genera un nuevo punto clave con su orientación. Entonces tendremos uno o más puntos claves situados en la misma posición pero con distintas orientaciones.

Para todos los puntos claves detectados se realizará una rotación según la orientación asignada al mismo, esto es fundamental para lograr la invarianza frente a rotaciones.

Representación del descriptor

La región de un punto clave se divide en $n \times n$ subregiones, para cada una de éstas se calcula el histograma de gradientes. En la fig. 2.2 se muestra

un descriptor de una región de 2×2 subregiones donde en cada una de ellas se visualiza una representación gráfica del histograma.

El descriptor del punto clave se representa por un vector que contiene los valores de todas las entradas de los histogramas de gradientes. La dimensionalidad (tamaño) del descriptor es $n \times n \times l$, donde $n \times n$ es la cantidad de subregiones y l es la cantidad de orientaciones posibles.

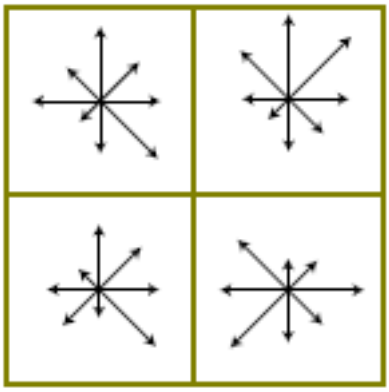

Fig. 2.2: Representación gráfica de una región, donde cada flecha se corresponde con un valor del histograma de gradientes. En este caso n = 2 y l = 8, obteniéndose en consecuencia un descriptor de dimensión 32.

2.1.5. Serialización del descriptor

Pensemos el descriptor como una representación vectorial de los valores del histograma. El objetivo de realizar la serialización del descriptor es facilitar la comparación entre estos.

Como la orientación del gradiente en el punto clave es invariante a rotaciones, podemos valernos de la misma para serializar el descriptor.

Se define el método de serialización a partir de la idea de procesar el descriptor por anillos, o sea, inicialmente procesar los histogramas correspondientes a las subregiones que se encuentran a mayor distancia del centro de la región (distancia m), siguiendo por los que se encuentran a distancia $m-1$, y así sucesivamente. Este mecanismo se continúa hasta procesar todos los histogramas de la región.

2.2. Almacenamiento y recuperación de descriptores

Con el objeto de realizar búsquedas con gran eficiencia se implementa una estructura llamada *árbol k-d*. A continuación se presenta su estructura y se detalla el mecanismo de búsqueda implementado.

2.2.1. Árboles k-d

Sea $\mathcal{C}$ un conjunto de características, éstas deben organizarse para poder ser encontradas y accedidas eficientemente.

En particular, dada un característica c que puede pertenecer o no a $\mathcal{C}$, queremos obtener la característica $c' \in \mathcal{C}$, que cumpla con la siguiente restricción:

$$c' = \arg\min_{c_i \in \mathcal{C}} d(c, c_i).$$

siendo d alguna distancia apropiada.

La condición anterior nos lleva a realizar la búsqueda de la característica más cercana a c.

La técnica utilizada para realizar eficientemente la búsqueda es la conocida como *divide & conquer*. Con ella se busca dividir un conjunto en subconjuntos de búsqueda, resolver el problema en estos subconjuntos para luego combinar los resultados. Dicha idea se puede realizar iterativamente hasta que la resolución de cada subconjunto sea trivial.

Llamamos d_c al descriptor de la característica c, como ya se ha mencionado, la dimensión de d_c es $dim = n \times n \times l$, donde $n \times n$ es la cantidad de subregiones y l es la cantidad de orientaciones posibles.

Definimos el conjunto C_j, $j \in [0, \ldots, dim - 1]$ de la siguiente manera:

$$\mathcal{C}_j = \bigcup_{c \in C} d_c(j). \tag{2.7}$$

siendo $d_c(j)$ la coordenada j de d_c.

La estrategia para armar un árbol k-d es la siguiente:

- dividir el conjunto $\mathcal{C}$ en dos partes, $\mathcal{C}^1$ y $\mathcal{C}^2$, partiéndolo en la mediana

m de la dimensión i con mayor varianza:

$$\begin{aligned} i &= \arg \max_{1 \leq j \leq k} Var(\mathcal{C}_j) \\ m_i &= mediana(\mathcal{C}_i) \\ \mathcal{C}^1 &= \{c \mid c \in \mathcal{C} \wedge c_i \leq m_i\} \\ \mathcal{C}^2 &= \{c \mid c \in \mathcal{C} \wedge c_i > m_i\} \end{aligned} \tag{2.8}$$

Notar que $|\#\mathcal{C}^1 - \#\mathcal{C}^2| \leq 1$.

- Se crea un árbol binario A en cuya raíz se almacenan m_i e i. El subárbol izquierdo de A es el resultado de aplicar recursivamente este procedimiento sobre $\mathcal{C}^1$ y el derecho, sobre $\mathcal{C}^2$.

Se termina creando un árbol binario balanceado, dado que los subárboles pertenecientes a un mismo nodo poseen la misma cantidad de elementos. La altura del árbol es

$$d = \lceil \log_2 \#C \rceil. \tag{2.9}$$

2.2.2. Ejemplo

A continuación se presenta un ejemplo en el cual se muestra el mecanismo mediante el cual se crea un árbol k-d.

Sea $\mathcal{C}$ un conjunto con 9 puntos característicos p_i, con descriptores d_i respectivamente, cuya dimensionalidad es 4. Para este ejemplo la dimensionalidad elegida es muy distinta de la utilizada en la implementación (128, con n = 4 y l = 8) debido a que el principal objetivo aquí es la comprensión del mecanismo de armado del árbol k-d.

Inicialmente se cuenta con un conjunto $\mathcal{C}$ de puntos en $\mathbb{R}^2$ para los cuales ya fue computado su descriptor.

Como primer paso se deben determinar los límites de la región en la que están contenidos los descriptores de $\mathcal{C}$. Los bordes que se usarán son el mínimo y el máximo en cada dimensión, como se muestra a continuación:

$$\begin{aligned} \text{mín}\,(\mathcal{C}_1) &= 12 \\ \text{máx}\,(\mathcal{C}_1) &= 200 \end{aligned}$$

$$\begin{aligned} \text{mín}\,(\mathcal{C}_2) &= 9 \\ \text{máx}\,(\mathcal{C}_2) &= 201 \end{aligned}$$

$$\begin{aligned} \text{mín}\,(\mathcal{C}_3) &= 47 \\ \text{máx}\,(\mathcal{C}_3) &= 222 \end{aligned}$$

$$\begin{aligned} \text{mín}\,(\mathcal{C}_4) &= 4 \\ \text{máx}\,(\mathcal{C}_4) &= 204 \end{aligned}$$

Para partir los datos se calcula la varianza en las dos dimensiones, y se parte el conjunto $\mathcal{C}$ en la mediana de la dimensión con mayor varianza.

$$\begin{aligned} \text{Var}(\mathcal{C}_1) &= 3906,53 \\ \text{Var}(\mathcal{C}_2) &= 4616,75 \\ \text{Var}(\mathcal{C}_3) &= 4414,25 \\ \text{Var}(\mathcal{C}_4) &= 6327,75 \end{aligned}$$

como $\text{Var}(\mathcal{C}_4) > \text{Var}(\mathcal{C}_2) > \text{Var}(\mathcal{C}_3) > \text{Var}(\mathcal{C}_1)$, entonces $m_4 = 76$.

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
$\mathcal{C}_1$	12	34	44	65	101	104	144	155	200
$\mathcal{C}_2$	55	13	99	201	101	144	132	11	9
$\mathcal{C}_3$	71	142	87	77	95	204	222	47	201
$\mathcal{C}_4$	24	4	9	197	174	63	76	204	92

Tab. 2.1: Conjunto $\mathcal{C}$ de puntos a insertar en el árbol k-d.

El punto $(144, 132, 222, 76)$ divide $\mathcal{C}$, como lo muestra la fig. 2.2. Dicha partición determina dos subconjuntos $\mathcal{C}^1$ y $\mathcal{C}^2$ correspondientes a los subárboles izquierdo y derecho respectivamente. Siguiendo este mecanismo se seguirá iterando hasta el caso en el que $\#\mathcal{C}^i \leq 3$, en el cual los puntos comenzarán a quedar como hojas del árbol.

	d_1	d_2	d_3	d_6	d_7
$\mathcal{C}_1^1$	12	34	44	104	144
$\mathcal{C}_2^1$	55	13	99	144	132
$\mathcal{C}_3^1$	71	142	87	204	222
$\mathcal{C}_4^1$	24	4	9	63	76

	d_4	d_5	d_8	d_9
$\mathcal{C}_1^2$	65	101	155	200
$\mathcal{C}_2^2$	201	101	11	9
$\mathcal{C}_3^2$	77	95	47	201
$\mathcal{C}_4^2$	197	174	204	92

Tab. 2.2: Conjuntos $\mathcal{C}^1$ y $\mathcal{C}^2$ de puntos a insertar en los subárboles k-d izquierdo y derecho respectivamente.

El árbol k-d resultante puede verse en la fig. 2.3.

El la fig. 2.4 se ve el árbol con las regiones que son asignadas a cada nodo. Se intuye facilmente que la región de un nodo es la unión de las regiones de los nodos hijos.

2.2.3. Búsqueda

El algoritmo utilizado para encontrar el descriptor homólogo a un descriptor dado q en un conjunto k-dimensional se basa en construir un árbol k-d A y utilizar la técnica de backtracking para la búsqueda.

Inicialmente se atravieza el árbol A para encontrar la región R que contiene al primer descriptor candidato que llamamos q'. Esto requiere tantas comparaciones como la altura del árbol en cuestión. El punto $q' \in R$ en general es una buena aproximación, aunque no siempre puede serlo, para salvaguardar dicho caso se utiliza backtracking, donde una rama entera puede

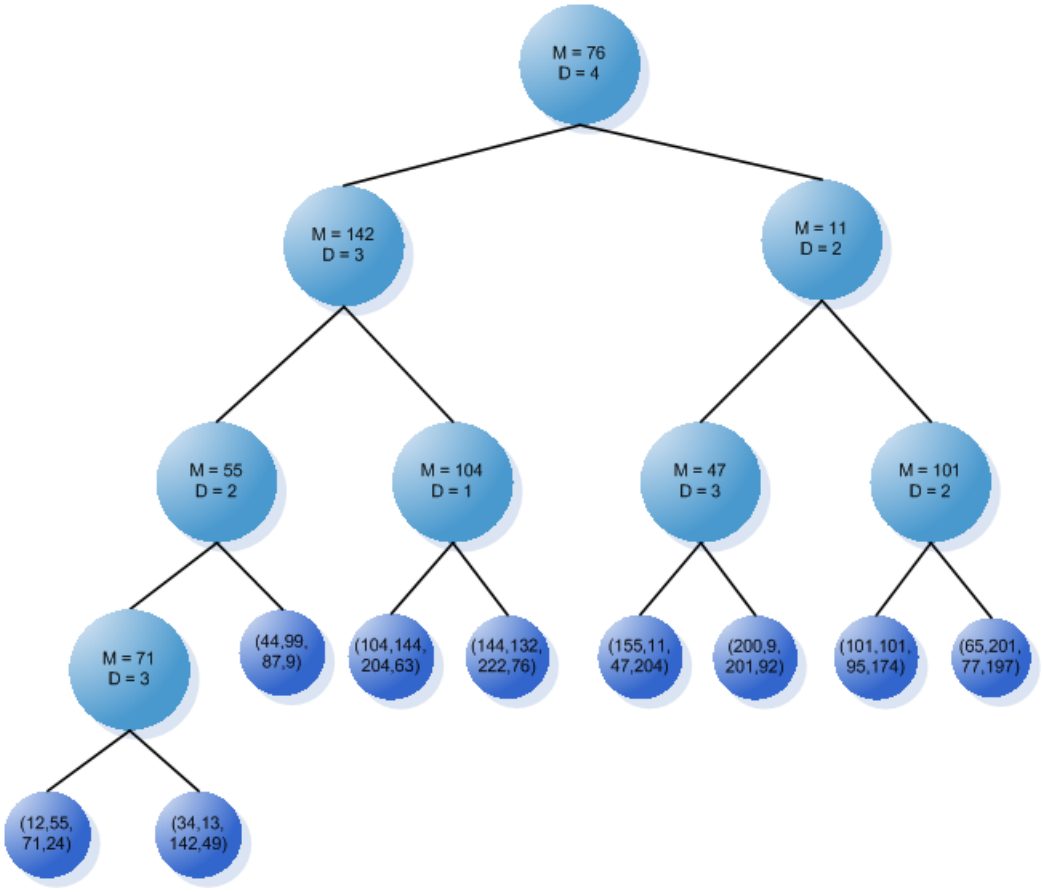

Fig. 2.3: Vista esquemática del árbol resultante.

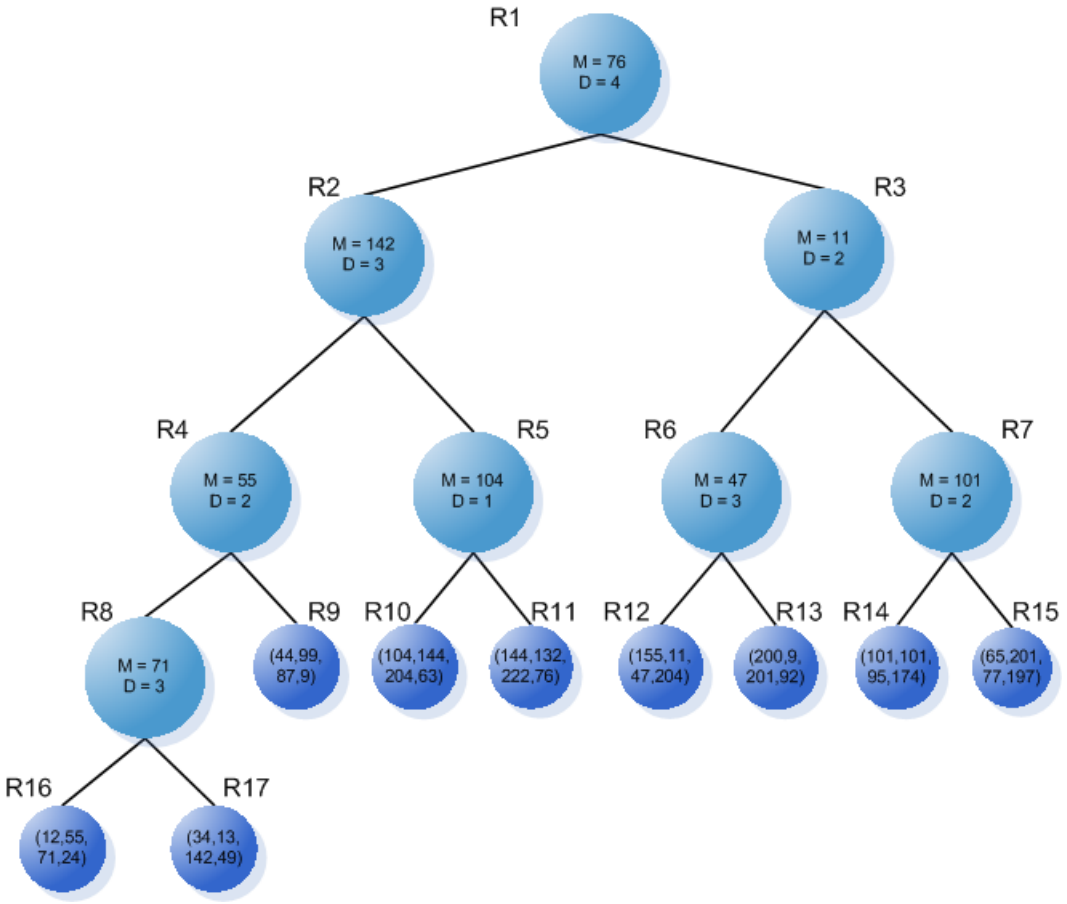

Fig. 2.4: Árbol con las regiones determinadas por los puntos.

ser podada si la región del espacio que representa tienen una distancia a q mayor que la distancia entre q' y q. En caso de que la distancia sea menor, se procede a buscar dentro de la región en busca de otros descriptores más cercanos a q. La búsqueda termina en el caso de que se han explorado todas las ramas que inicialmente no fueron exploradas.

Cuando se baja por un subárbol, se agrega el otro a una cola de prioridad de ramas no exploradas. Esta cola ordena las ramas en orden creciente de distancia a q.

2.2.4. Ejemplo

Suponiendo que se cuenta con el árbol mostrado en el ejemplo anterior, haremos la búsqueda del punto $q = (155, 124, 88, 92)$.

La búsqueda se inicia en la raíz del árbol (ver fig. 2.3). La dimensión Z posee mayor varianza en el nodo raiz, por este motivo comparamos la mediana 76 con $q_X = 92$. Como $92 > 76$ entonces se desplaza por el subárbol derecho y se encola el izquierdo (región R2) en la cola Q de subárboles no explorados.

Siguiendo el mecanismo mencionado se desciende por los subárboles hasta encontrar el punto de la región a la que pertence q. El recorrido realizado se muestra en la fig. 2.5.

A medida que se desciende por el árbol, en la cola de ramas se van encolando los subárboles no visitados, en el ejemplo visto se encola primero el subárbol correspondiente a la región R2, luego R6 y por último R14.

Luego de obtener el punto q', en este caso $q' = (155, 124, 88, 92)$, se buscará en la cola de subárboles no visitados si alguna región mejora la distancia entre ella y q, o sea, si $d(q, q') > d(q, R_i)$ con $R_i \in Q$. En caso de que alguna región cumpla con el requisito de distancia propuesto, entonces se descenderá por el subárbol correspondiente realizando el mismo mecanismo descrito anteriormente. El resultado final será el punto q' que tenga menor distancia a q. En el caso $\{\exists(q')\exists(q'')/d(q, q') = d(q, q'')\}$ entonces se optará por el que se encuentre a menor distacia, y si ocurriera que dicha distancia es la misma entonces se optará por el primero encontrado.

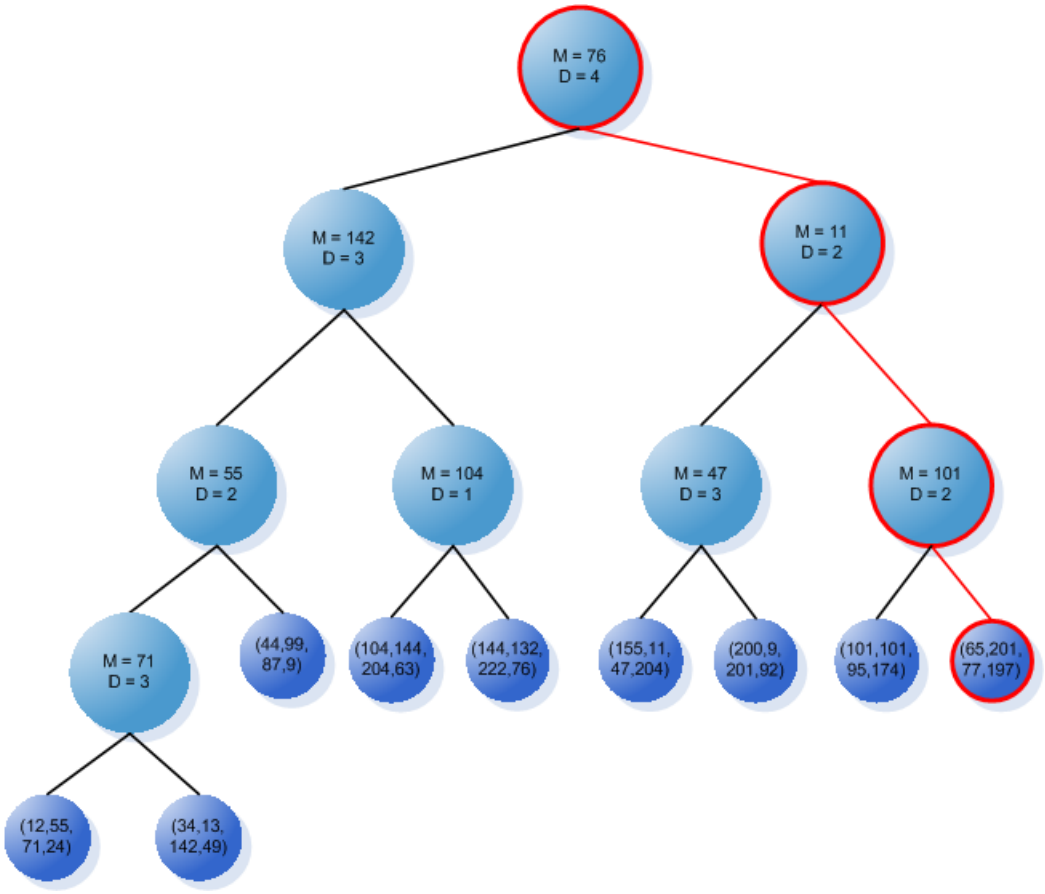

Fig. 2.5: Vista del recorrido realizado al descender por los subárboles.

2.3. Conjunto de correspondencias

A partir de dos fuentes de descriptores se puede establecer un conjunto de correspondencias. Para ello se debe construir un árbol k-d A con una de la fuentes de descriptores serializados y luego para cada uno de los descriptores de la otra fuente se debe realizar la búsqueda en A. Cada par de descriptores detectados se agrega en un conjunto de correspondencias que llamaremos CCO, el cual es utilizado para etapas posteriores.

El mecanismo completo se muestra en el algoritmo 2.1.

Algoritmo 2.1 GENERACIÓN DEL CONJUNTO DE CORRESPONDENCIAS: Se genera un conjunto de correspondencias en base a dos fuentes de descriptores.

1: $Fuente1 \leftarrow$ Primer fuente de descriptores
2: $Fuente2 \leftarrow$ Segunda fuente de descriptores
3: $Arbol \leftarrow$ CrearArbolk-d()
4: $CCO \leftarrow \emptyset 102$.
5: **for all** $d \in Fuente1$ **do**
6: agregarCaracteristica(Arbol,d)
7: **end for**
8: **for all** $d' \in Fuente2$ **do**
9: **if** estaCaracteristica(Arbol,d') **then**
10: $CCO \leftarrow CCO \cup \{$dameCaracteristica(Arbol,d')$\}$
11: **end if**
12: **end for**
13: RETURN CCO

3. ESTIMADORES ROBUSTOS

En el capítulo anterior se mostró cómo es posible armar un conjunto de correspondencias entre puntos de dos fuentes. La intención es estimar una transformación T que haga que cada punto de la primer fuente se corresponda con otro de la segunda.

Existen distintas fuentes de error que hacen que lo mencionado no ocurra, por ejemplo, una fuente de error proviene de la medición de la posición de los puntos, otra del armado del conjunto de correspondencias, así notamos que dentro de un conjunto de correspondencias habrá elementos que no deberían pertenecer, llamaremos a estos elementos *outliers*, y elementos que si deben pertenecer, a los cuales llamaremos *inliers*.

Los *outliers* pueden afectar severamente la transformación estimada y, por ende, deben ser encontrados y descartados. Luego de la eliminación de los *outliers* podremos estimar T utilizando únicamente los *inliers*. Ésta estimación es llamada *robusta* ya que no es susceptible al efecto de los *outliers*.

A continuación se presentan dos métodos que incorporan técnicas para manejar errores groseros. Uno, llamado Random Sample Consensus (RANSAC) [1] y otro incorporado en este trabajo llamado Eliminación por Pendiente y Longitud (EPL). Ambos se basan en el establecimiento de grupos de consenso a partir de los cuales poder realizar las estimaciones. En §6 se mostrarán algunos ejemplos del comportamiento de ambos métodos.

3.1. RANSAC

Fischler y Bolles, en [1] introdujeron un nuevo método para ajustar un modelo a un conjunto experimental de datos que contenga, potencialmente, un porcentaje significativo de errores groseros. El procedimiento RANSAC, es el opuesto al de las técnicas convencionales: en lugar de utilizar tantos datos

como es posible para obtener una solución inicial y luego eliminar los datos inválidos, se parte de un conjunto inicial mínimo el cual se va agrandando con datos consistentes.

Como primer paso se toma un subconjunto minimal sm del conjunto de correspondencias C. La cantidad de elementos de sm se determina dependiendo del tipo de transformación que se desee estimar. La selección de los elementos de sm se hace en forma aleatoria.

El algoritmo es iterativo, estimándose en la iteración i el modelo M_i a partir del subconjunto sm. A partir del modelo obtenido se verifica qué correspondencias de C son *inliers* al modelo M_i y a partir de ellas se forma un suboncojunto de consenso S_i dejando fuera las consideradas *outliers* según el modelo.

Se realiza el proceso mencionado iterativamente seleccionando distintos subconjuntos minimales en cada iteración. Finalmente se selecciona el modelo M_k tal que el correspondiente S_k sea el de máximo cardinal.

A continuación se muestra el algoritmo 3.1.

Algoritmo 3.1 RANSAC: Se determina en forma robusta un conjunto de correspondencias S que contiene *inliers*

1: **for all** $0 \leq i < N$ **do**
2: Seleccionar aleatoriamente una muestra minimal sm de correspondencias de C
3: Instanciar el modelo M_i a partir de sm
4: Determinar el conjunto de consenso S_i de las correspondencias que soportan el modelo M_i
5: reestimar el modelo M_i usando S_i
6: **if** $\#S_i > u$ **then**
7: **return** M_i
8: **end if**
9: **end for**
10: **return** M_k tal que $k = \arg\max_{0 \leq i \leq N}(\#S_i)$

Existen tres parámetros sin especificar:

1. Número de iteraciones
2. Tolerancia usada para determinar si un punto es compatible con el modelo.
3. Umbral u, que es el número de puntos necesarios para considerar que el modelo correcto fue encontrado.

Número de iteraciones

La cantidad de intentos necesarios para encontrar un subconjunto s de n puntos inliers del algoritmo (3.1) puede estimarse el modelo que se muestra a continuación. La probabilidad de que un punto se encuentre dentro de la tolerancia al error del modelo la llamaremos w y la cantidad de intentos que queremos estimar N. Entonces, la probabilidad de que todos los puntos de s estén en el modelo es $p = w^n$ (puede haber repeticiones en los puntos de s). Se deriba que la probabilidad de encontrar un subconjunto en N intentos está dada por:

$$P(N) = (q)^{N-1}p \quad \text{con } q = 1 - p.$$

Luego,

$$\begin{aligned} E_P(N) &= \sum_{i=1}^{\infty} i(q)^{i-1}p \\ &= p + 2qp + 3q^2p + \cdots + iq^{i-1}p + \dots \\ &= p(1 + 2q + 3q^2 + \cdots + iq^{i-1} + \dots) \end{aligned} \tag{3.1}$$

$E_P(N)$ es la cantidad de intentos necesarios que se espera realizar para obtener un subconjunto s.

Se sabe que, para una suma geométrica,

$$\frac{x}{1-x} = x + x^2 + x^3 + \cdots + x^i + \dots$$

derivando ambos términos con respecto a x, obtenemos

$$\frac{1}{(1-x)^2} = 1 + 2x + 3x^2 + \cdots + ix^{i-1} + \dots \tag{3.2}$$

Combinando 3.1 y 3.2,

$$E_P(N) = \frac{p}{(1-p)^2} = \frac{1}{p} = w^{-n} \tag{3.3}$$

En general queremos, con el fin de tener mayores certezas, realizar un exceso de una o dos desviaciones estándar suplementarias, además de los $E_p(N)$ intentos.

Tenemos que:

$$V_P(N) = \sqrt{E_P(N^2) - E(N)^2} \tag{3.4}$$

y

$$\begin{aligned} E_P(N^2) &= \sum_{i=0}^{\infty} bi^2 a^{i-1} \\ &= \sum_{i=0}^{\infty} bi(i-1)a^{i-1} + \sum_{i=0}^{\infty} bia^{i-1} \end{aligned} \tag{3.5}$$

Derivando nuevamente los dos términos de la identidad 3.2,

$$\frac{2x}{(1-x)^3} = \sum_{i=0}^{\infty} \left(i(i-1)x^{i-1}\right) \tag{3.6}$$

Finalmente,

$$E_P(N^2) = \frac{2-b}{b^2}$$

y

$$V_P(N) = \frac{\sqrt{1-w^n}}{w^n} \tag{3.7}$$

Para facilitar el análisis es útil notar que $V(k) \approx E(k)$. Por ejemplo si $w = 0{,}5$ y $n = 4$ entonces $E(k) = 16$ y $V(k) = 15,5$. Esto significa que el número de intentos total sería dos o tres veces el número esperado implicado por $E(k)$.

Tolerancia para establecer la compatibilidad entre los datos y el modelo

El error asociado a un modelo puede describirse, en parte, como una función de los errores de los datos utilizados para instanciar el modelo. Este

error, junto al de un dato particular, componen la desviación de un dato a un modelo.

Si los datos forman un modelo con una función simple, es posible establecer analíticamente cotas para el error. Sin embargo, este acercamiento es generalmente inutilizable; para estos casos es necesario utilizar cotas experimentales.

La desviación depende del error particular del dato que se está analizando, por lo que la tolerancia utilizada también debería depender de este error. Sin embargo, las variaciones en la tolerancia de dato en dato varían muy sutilmente, comparándolas con el resto de los errores presentes; entonces podemos despreciarlas y utilizar una tolerancia global.

Tamaño mínimo del conjunto de consenso

Para determinar si un subconjunto S_i de S tiene suficiente consenso para ser aceptado por el algoritmo se utiliza el umbral u (variable libre).

Una buena estimación es terminar el algoritmo si el tamaño de S_i es similar a la cantidad de *inliers* que se estima que hay en S, ya que no se dispone de un método analítico para determinar el valor de u.

3.2. Eliminación por pendiente y longitud

Supongamos que un joven se mezcla en un grupo de ancianos, ¿cómo podríamos detectar quién es el joven realizando la misma pregunta a cada uno de ellos?. Si no es posible realizar preguntas relativas como ¿es usted joven? o ¿es usted anciano?, una buena aproximación sería preguntar la edad a cada uno de ellos y luego verificar cual de las edades se diferencia de la mayoría. Seguramente la mayoría de las edades se encontrarán entre los 70 y 80, pero la edad del joven estará lejos de dicho rango. Así es posible identificar quién es el joven en el grupo de ancianos ya que éste no cumple con una característica que si cumple la gran mayoría.

El mecanismo que se presenta a continuación se basa en este mismo concepto. Se intenta establecer qué correspondencias son outliers a partir del incumplimiento de características generales.

Mecanismo general de detección

El mecanismo recibe como entrada un conjunto de elementos y determina cuales de dichos elementos son considerados inliers y cuales outliers. Para realizar dicha tarea, previamente se debe definir qué es un elemento y características comunes a estos. Existe una etapa inicial llamada **Definiciones**, la cual se describe a continuación:

- Definiciones

  Inicialmente se debe determinar qué objetos se desea filtrar, a estos objetos los llamaremos **elementos**. Un elemento puede ser una persona como en el ejemplo visto en 3.2, una correspondencia, un punto, etc.
  Habiendo determinado qué es un elemento, debemos encontrar funciones que permitan distinguir un elemento como inlier u outlier. Por ejemplo la función edad utilizada en el ejemplo dado en 3.2 permite distinguir qué personas son ancianos (inliers), y quienes no lo son (outliers). Llamaremos **características** a dichas funciones.
  Cada característica presenta la siguiente aridad $f : elemento \rightarrow \Re$, a partir de las características seleccionadas se define un conjunto de características llamado $CC = \{caracterstica\ \ c_k : 0 \leq k < m\}$.

Habiendo realizado las definiciones necesarias, se procede a describir el resto del mecanismo propiamente dicho, el cual se divide en 3 fases que se detallan a continuación:

1. Cálculo de valores asociados

   Dado un conjunto de elementos de entrada que llamaremos $CE = \{elemento\ e_i : 0 \leq i < n\}$, y el conjunto de características CC determinado previamente, definimos ${v_i}^k = c_k(e_i)$.

2. Determinación de niveles acumulados

   En segundo lugar se calcula cuánto se diferencia cada ${v_i}^k$ con respecto a ${v_j}^k(\forall j \neq i)$. Para ello se define la función llamada *da* (diferencia acumulada) que se expresa analíticamente de la siguiente manera:

$$da_i{}^k = da(v_i{}^k) = \sum_{j=0}^{n} (v_i{}^k - v_j{}^k)^2$$

La función $da_i{}^k$ retorna un valor numérico que indica cuanto difiere $v_i{}^k$ de $v_j{}^k (\forall j \neq i)$, dicha función aplicada a los elementos de un conjunto donde la mayoría de ellos dan resultados similares implica que los que no lo hagan obtendrán un resultado sustancialmente superior con respecto al resto.

Para cada característica k se define un conjunto $CDA^k = \{da_i{}^k : 0 \leq i < n\}$.

3. Propuesta de inliers

   A partir de los $da_i{}^k$ calculados se determinan los mínimos por cada característica. El mínimo de las diferencias acumuladas para la característica k se define de la siguiente forma $min^k = min(CDA^k)$.

   Habiendo calculado los mínimos se define por cada caraterística un umbral que determina si un elemento e_i se concidera inlier. El umbral para la característica k se define como $u^k = min^k * cl^k$, siendo cl^k el **coeficiente límite** de la característica de la característica k.

   Un elemento e_i es propuesto **inlier** si vale que $(\forall k)\ da_i{}^k < u^k$.

El mecanismo completo se muestra en el algoritmo 3.2.

Coeficiente límite para determinar outliers

Inicialmente se calcula un coeficiente llamado cd como se muestra en la siguiente fórmula $cd = 0{,}1n$, siendo n la cantidad total de elementos. Este coeficiente es común a todas las características.

Para cada característica k se tiene un coeficiente que se determina empíricamente al que llamaremos ce^k.

Teniendo para una característica k los coeficientes ce^k y cd definimos cl^k (coeficiente límite) en función de ellos de la siguiente manera $cl^k = ce^k + cd$.

En los casos en que la disparidad en los valores obtenidos en la etapa *Cálculo de valores asociados* sea alta (debido a propiedades de la caracerística

Algoritmo 3.2 ELIMXCAR: Se determina en forma robusta un conjunto de correspondencias que contiene únicamente *inliers*

1: $CE \leftarrow$ Conjunto de Elementos de entrada
2: $CC \leftarrow$ Conjunto de Características definidas
3: **for all** $c_k \in CC$ **do**
4: **for all** $e_i \in CE$ **do**
5: ${v_i}^k \leftarrow c_k(e_i)$
6: **end for**
7: **end for**
8: **for all** $c_k \in CC$ **do**
9: $CDA^k = \emptyset$
10: **for all** $e_i \in CE$ **do**
11: ${da_i}^k \leftarrow da({v_i}^k)$
12: $Agregar(CDA^k, {da_i}^k)$
13: **end for**
14: **end for**
15: **for all** $c_k \in CC$ **do**
16: $min^k \leftarrow Minimo(CDA^k)$
17: **end for**
18: **for all** $c_k \in CC$ **do**
19: **for all** $e_i \in CE$ **do**
20: **if** ${da_i}^k < Min^k * u^k$ **then**
21: $Agregar(INLIERS, e_i)$
22: **end if**
23: **end for**
24: **end for**
25: **return** $INLIERS$

elegida), el valor que se le atribuya al coeficiente *ce* deberá ser mayor a casos con disparidad baja, esto trae como consecuencia que no se consideren como outliers casos que no lo son.

Ejemplo

Habiendo definido un mecanismo genérico que propone elementos **inliers**, se procede a dar un ejemplo concreto, el cual se basa en la implementación realizada en el presente trabajo.

El ejemplo se compone de dos imágenes I_{izq} y I_{der}, ambas de dimensión $N \times M$, una obtenida por la cámara izquierda y otra por la derecha, dichas imágenes se muestran en 3.1 y 3.2 respectivamente. Para dichas imágenes se detectaron 70 correspondencias mediante conceptos vistos en el capítulo 2, de las cuales sólo se utilizarán 15 para el ejemplo. Estas 15 correspondencias se detallan en la tabla 3.1.

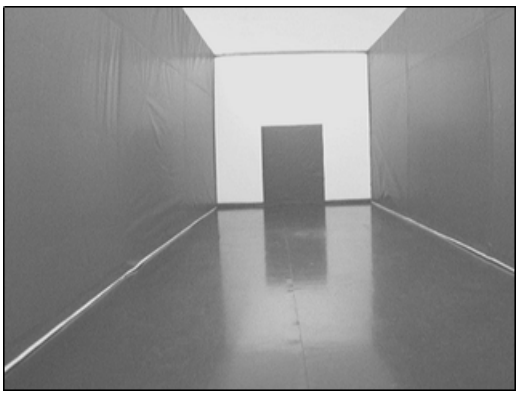

Fig. 3.1: Imagen correspondiente a la cámara izquierda.

Llamaremos UH (unión horizontal) a la imagen de dimensión $2N \times M$ que se define se la siguiente manera:

$$UH(i,j) = \begin{cases} I_{izq}(i,j) & (1 \leq i \leq N) \wedge (1 \leq j \leq M) \\ I_{der}(i-N,j) & (N+1 \leq i \leq 2N) \wedge (1 \leq j \leq M) \end{cases}$$

En la tabla 3.2 se muestran las correspondencias de las imágenes 3.1 y

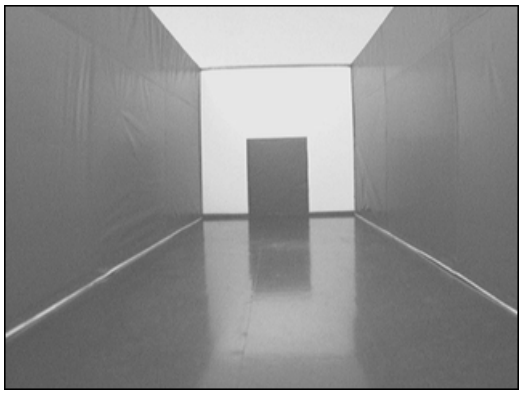

Fig. 3.2: Imagen correspondiente a la cámara derecha.

	Puntos Izq		**Puntos Der**	
	X	**Y**	**X**	**Y**
1	91	159	72	165
2	101	20	409	122
3	127	127	114	135
4	140	129	130	136
5	147	106	136	113
6	150	148	139	156
7	161	128	149	136
8	162	76	185	85
9	180	95	169	102
10	215	103	203	111
11	215	146	203	154
12	228	126	216	134
13	228	118	216	126
14	239	123	227	130
15	272	115	250	122

Tab. 3.1: Correspondencias de las imágenes.

3.2 en UH. Para cada correspondencia se traza una línea que une las características que la componen, como se muestra en la imagen 3.3.

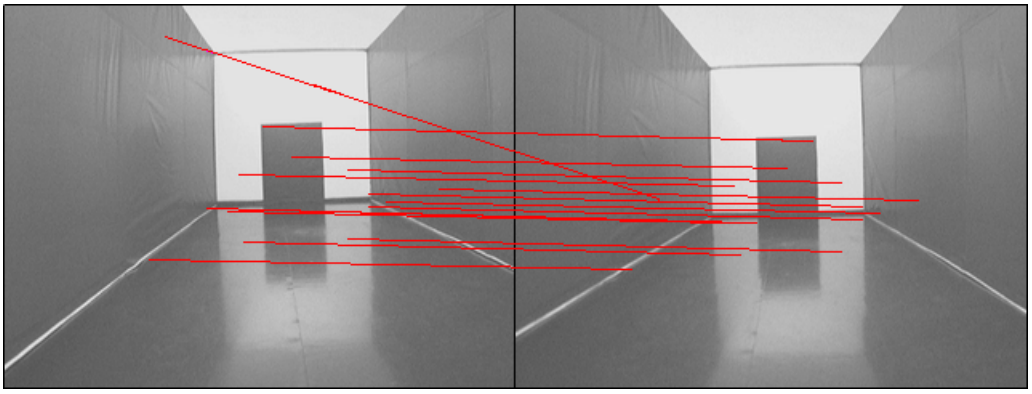

Fig. 3.3: Correspondencias en UH unidas por líneas rojas.

Los elementos $\{e_i, 0 \leq i < 15\}$ seleccionados para el ejemplo son **las correspondencias de UH**, y las características utilizadas son la pendiente c_1 y longitud c_2 de las líneas que unen las características de dichas correspondencias. Las características seleccionadas son funciones que retornan un valor numérico a partir de cada correspondencia.

Disponiendo del conjunto de características CA y de elementos CE, resta aplicar los tres pasos del mecanismo presentado en 3.2.

El primer paso consiste en la creación de los conjuntos de elementos, para ello cada correspondencia debe ser evaluada para las características seleccionadas. Si pensamos una correspondencia como un par de puntos (P1,P2), la pendiente y longitud de una correspondencia se calculan aplicando las fórmulas 3.8 y 3.9 respectivamente.

$$pendiente(P1, P2) = \frac{P1.Y - P2.Y}{P1.X - P2.X} \tag{3.8}$$

$$longitud(P1, P2) = \sqrt{(P1.X - P2.X)^2 + (P1.Y - P2.Y)^2} \tag{3.9}$$

En la tabla 3.3 se muestran los valores ${v_i}^k = c_k(e_i)$.

	Puntos Izq		**Puntos Der**	
	X	**Y**	**X**	**Y**
e_1	72	165	**392**	165
e_2	89	122	**409**	122
e_3	114	135	**434**	135
e_4	130	136	**450**	136
e_5	136	113	**456**	113
e_6	139	156	**459**	156
e_7	149	136	**469**	136
e_8	185	85	**505**	85
e_9	169	102	**489**	102
e_10	203	111	**523**	111
e_11	203	154	**523**	154
e_12	216	134	**536**	134
e_13	216	126	**536**	126
e_14	227	130	**547**	130
e_15	250	122	**570**	122

Tab. 3.2: Correspondencias en UH.

	Pendiente		**Longitud**
${v_1}^1$	0.019934	${v_1}^2$	301.0597
${v_2}^1$	0.331169	${v_2}^2$	324.4503
${v_3}^1$	0.026059	${v_3}^2$	307.1042
${v_4}^1$	0.022581	${v_4}^2$	310.0790
${v_5}^1$	0.022654	${v_5}^2$	309.0792
${v_6}^1$	0.025890	${v_6}^2$	309.1035
${v_7}^1$	0.025974	${v_7}^2$	308.1038
${v_8}^1$	0.026239	${v_8}^2$	343.1180
${v_9}^1$	0.022654	${v_9}^2$	309.0792
${v_{10}}^1$	0.025974	${v_{10}}^2$	308.1038
${v_{11}}^1$	0.025974	${v_{11}}^2$	308.1038
${v_{12}}^1$	0.025974	${v_{12}}^2$	308.1038
${v_{13}}^1$	0.025974	${v_{13}}^2$	308.1038
${v_{14}}^1$	0.022727	${v_{14}}^2$	308.0795
${v_{15}}^1$	0.023490	${v_{15}}^2$	298.0822

Tab. 3.3: Pendientes y longitudes calculadas.

Habiendo calculado los conjuntos de elementos se procede a determinar los niveles acumulados.

Para cada ${v_i}^k$ se debe calcular la **diferencia acumulada** (aplicar la función 2). Los resultados del procedimiento pueden verse en la tabla 3.4.

	Pendiente acumulada		**Longitud acumulada**
${da_1}^1$	0.097204	${da_1}^2$	2933.459
${da_2}^1$	1.317248	${da_2}^2$	4410.388
${da_3}^1$	0.093182	${da_3}^2$	1742.411
${da_4}^1$	0.095328	${da_4}^2$	1558.685
${da_5}^1$	0.095279	${da_5}^2$	1590.812
${da_6}^1$	0.093278	${da_6}^2$	1589.677
${da_7}^1$	0.093230	${da_7}^2$	1651.054
${da_8}^1$	0.093081	${da_8}^2$	17366.109
${da_9}^1$	0.095279	${da_9}^2$	1590.812
${da_{10}}^1$	0.093230	${da_{10}}^2$	1651.054
${da_{11}}^1$	0.093230	${da_{11}}^2$	1651.054
${da_{12}}^1$	0.093230	${da_{12}}^2$	1651.054
${da_{13}}^1$	0.093230	${da_{13}}^2$	1651.054
${da_{14}}^1$	0.095230	${da_{14}}^2$	1652.923
${da_{15}}^1$	0.094732	${da_{15}}^2$	3923.148

Tab. 3.4: Resultado de aplicar la función $da({v_i}^k)$.

En el último paso se deben proponer elementos inliers, o en su defecto outliers. Para esto, se debe encontrar para cada característica el valor mínimo. Los valores mínimos calculados son $min^1 = \mathbf{0.093081}$ (pendiente) y $min^2 = \mathbf{1558.685}$ (longitud).

El **coeficiente límite** *cl* se expresa como la suma de dos coeficientes, el primero de ellos *ce* se determina empíricamente para cada característica y el segundo *cd* depende de la cantidad de correspondencias, y es común a todas las características, este se calcula de la siguiente forma $cd = 0{,}1\#CE$. Los coeficientes utilizados en el ejemplo se exponen en la tabla 3.5.

	Coeficiente ce	**Coeficiente** cd	**Coeficiente** cl
Pendiente - c_1	3	1.5	**4.5**
Longitud - c_2	3.5	1.5	**5**

Tab. 3.5: Coeficientes utilizados para determinar las cotas.

Por cada característica se calcula el coeficiente límite que se define $cl^k = ce^k + cd$, siendo ce^k el coeficiente que se determina empiricamente para cada característica y $cd = 0{,}1\#CE$.

En la tabla 3.5 se muestran los coeficientes utilizados para cada característica.

Habiendo calculado el valor mínimo y el coefciente límite para cada característica, resta determinar que correspondencias (elementos) son propuestas inliers.

La correspondencia e_i es propuesta inlier si $da_i^1 < min^1cl^1 \wedge da_i^2 < min^2cl^2$. En la tabla 3.8 se muestran los resultados finales, estos se deducen de los resultados parciales presentados en 3.6 y 3.7.

Como resultado se proponen 13 correspondencias inliers y dos outliers. En la imagen 3.4 se muestran en rojo las inliers y en violeta las outliers.

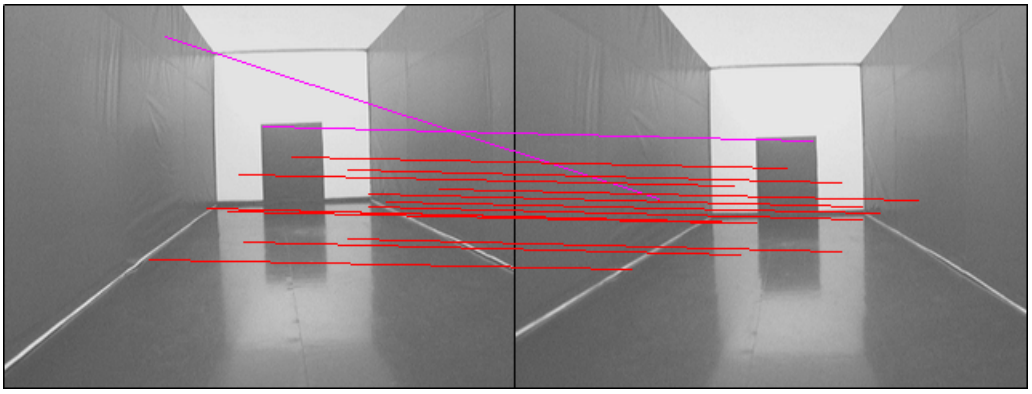

Fig. 3.4: Correspondencias propuestas como inliers y outliers.

	Condición	**¿Inlier?**
$da_1{}^1$	0,097204 < 0,418866	SI
$da_2{}^1$	1,317248 < 0,418866	NO
$da_3{}^1$	0,093182 < 0,418866	SI
$da_4{}^1$	0,095328 < 0,418866	SI
$da_5{}^1$	0,095279 < 0,418866	SI
$da_6{}^1$	0,093278 < 0,418866	SI
$da_7{}^1$	0,093230 < 0,418866	SI
$da_8{}^1$	0,093081 < 0,418866	SI
$da_9{}^1$	0,095279 < 0,418866	SI
$da_{10}{}^1$	0,093230 < 0,418866	SI
$da_{11}{}^1$	0,093230 < 0,418866	SI
$da_{12}{}^1$	0,093230 < 0,418866	SI
$da_{13}{}^1$	0,093230 < 0,418866	SI
$da_{14}{}^1$	0,095230 < 0,418866	SI
$da_{15}{}^1$	0,094732 < 0,418866	SI

Tab. 3.6: Propuesta de correspondencias inliers para la característica pendiente.

	Condición	**¿Inlier?**
${da_1}^2$	2933,459 < 7793,427	SI
${da_2}^2$	4410,388 < 7793,427	SI
${da_3}^2$	1742,411 < 7793,427	SI
${da_4}^2$	1558,685 < 7793,427	SI
${da_5}^2$	1590,812 < 7793,427	SI
${da_6}^2$	1589,677 < 7793,427	SI
${da_7}^2$	1651,054 < 7793,427	SI
${da_8}^2$	17366,109 < 7793,427	NO
${da_9}^2$	1590,812 < 7793,427	SI
${da_{10}}^2$	1651,054 < 7793,427	SI
${da_{11}}^2$	1651,054 < 7793,427	SI
${da_{12}}^2$	1651,054 < 7793,427	SI
${da_{13}}^2$	1651,054 < 7793,427	SI
${da_{14}}^2$	1652,923 < 7793,427	SI
${da_{15}}^2$	3923,148 < 7793,427	SI

Tab. 3.7: Propuesta de correspondencias inliers para la característica longitud.

Corr.	Res. pendiente	Res. longitud	¿Inlier?
1	SI	SI	SI
2	NO	SI	NO
3	SI	SI	SI
4	SI	SI	SI
5	SI	SI	SI
6	SI	SI	SI
7	SI	SI	SI
8	SI	NO	NO
9	SI	SI	SI
10	SI	SI	SI
11	SI	SI	SI
12	SI	SI	SI
13	SI	SI	SI
14	SI	SI	SI
15	SI	SI	SI

Tab. 3.8: Propuesta de correspondencias inliers.

4. GEOMETRÍA EPIPOLAR

Este capítulo describe cómo obtener la posición de un punto en el espacio, dada su imagen en dos vistas y los parámetros de la cámaras utilizadas para cada una de esas vistas. Se asume que existen errores sólo en la ubicación de las imágenes del punto en las vistas.

Antes de adentrarnos en la descripción del problema y sus soluciones, vamos a presentar ciertos conceptos introductorios necesarios.

4.1. Conceptos preliminares

Una cámara es un mapeo entre el mundo 3D y una imagen 2D. Su anatomía puede ser modelada gracias a las herramientas provistas por la geometría descriptiva.

Utilizaremos el modelo de proyección central [1], siendo centro de proyección del plano el origen de un sistema de coordenadas euclídeo. El plano $z = f$ es llamado *plano imagen* o *plano focal*. El centro de proyección es llamado centro de la cámara o centro óptico. Bajo el modelo *pinhole* un punto en el espacio $\mathrm{x} = (x, y, z)^T$ es proyectado en el punto del plano imagen en que lo corta la línea que une el punto x y el centro de proyección. Por triángulos similares, x es mapeado al punto $(fx/z, fy/z, f)^T$ en el plano imagen. Ignorando la última coordenada, vemos que

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} fx/z \\ fy/z \end{pmatrix} \tag{4.1}$$

[1] La proyección central es una correspondencia de puntos (p) en un plano en puntos(p') en otro plano, que puede ser escrita como $p' = Hp$, donde H es una matriz no singular de 3×3

describe la proyección central a coordenadas en la imagen. En otras palabras es una función desde un espacio euclídeo $\mathbb{R}^3$ a un espacio euclídeo $\mathbb{R}^2$. Todos los puntos $(x_1, x_2, x_3)^T$ tales que $x_3 \neq 0$ corresponden a puntos finitos en $\mathbb{R}^2$. Aumentando $\mathbb{R}^2$ con el conjunto de puntos tales que $x_3 = 0$, formamos el espacio proyectivo $\mathbb{P}^2$.

La ecuación 4.1 puede ser expresada entonces en términos matriciales:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} fx/z \\ fy/z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{4.2}$$

Introducimos la notación $\mathrm{X} = (x, y, z, 1)^T$, x para el correspondiente punto en la imagen y P para la matriz definida en 4.2, lo que resulta en

$$\mathrm{x} = P\mathrm{X}$$

Dado que el centro del sistema de coordenadas del plano focal está, en las fórmulas anteriores, en la proyección del centro de la cámara sobre el plano focal (lo llamamos el punto principal, p), y que normalmente el sistema de coordenadas de las imágenes no asume esta misma convención, es necesario realizar una traslación para adaptar el último al primero. Luego,

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} fx/z + p_x \\ fy/z - p_y \end{pmatrix}$$

donde $p = (p_x, p_y)^T$. Podemos entonces rescribir P como

$$P_{\text{imagen}} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & -p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

La formulación anterior presupone que las coordenadas de la imagen son euclídeas y tienen escalas iguales en ambos ejes. Ésto no es necesariamente cierto en las cámaras CCD, que pueden tener píxeles rectangulares (no cuadrados). Como las coordenadas están medidas en píxeles, se introduce una factor que escala las cordenadas en cada dirección. En particular si la

cantidad de píxeles por unidad de distancia es m_x y m_y en los ejes x e y respectivamente, se vuelve imperativo reescalar los vectores de entrada en la siguiente forma

$$P_{\text{CCD}} = \begin{bmatrix} fm_x & 0 & p_x & 0 \\ 0 & fm_y & -p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Ahora que conocemos como es modelada una cámara gracias al algebra de la geometría descriptiva, podemos continuar con el problema que nos involucra.

4.2. Geometría epipolar

La geometría epipolar es la geometría proyectiva intrínseca entre dos vistas. Es independiente de la estructura de la escena, y sólo depende de los parámetros internos de las cámaras y sus poses relativas.

La matriz fundamental F encapsula esta geometría intrínseca. Es una matriz de 3×3 y rango 2. Si un punto X en $\mathbb{R}^3$ tiene una imagen x en la primera vista, y x′ en la segunda, entonces x, x′ y los centros de las cámaras son coplanares, como lo muestra la figura 4.1. Este plano es llamado π. Claramente los rayos retroproyectados desde x y x′ se intersecan en X; los rayos también son coplanares y están en π.

Suponiendo que sólo conocemos x, buscamos establaecer como el punto x′ está restringido. El plano π está determinado por la línea de base (que pasa por los centros de las cámaras) y el rayo definido por x. Sabemos que el rayo correspondiente al punto desconocido x′ están en π, por lo que x′ está en la línea l' resultante de la intereseccion de π con el segundo plano imagen. La línea l' es la proyección en la segunda vista del rayo retroproyectado desde x. En términos de un algoritmo para encontrar correspondencias estereoscópicas, es suficiente con restringir la búsqueda a l' en lugar de buscar en todo el plano imagen.

Las entidades involucradas en la gemoetría epipolar están ilustradas en la figura 4.2. La terminología es

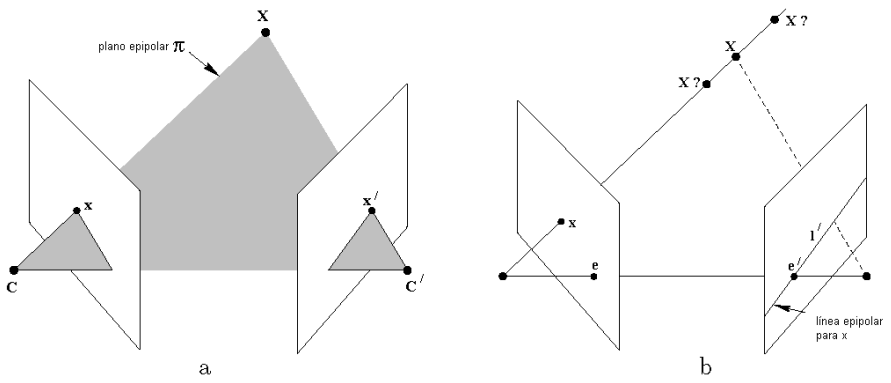

Fig. 4.1: (a) Las dos cámaras están indicadas por sus centros C y C' y sus planos focales. Estos centros, el punto tridimensional X y sus imágenes x y x$'$ son coplanares. Un punto x en la imagen es retroproyectado en un rayo tridimensional definido por C y x. El rayo está proyectado en la segunda vista como una línea l'. El punto tridimensional que se proyecta en x debe estar en el rayo, por lo que su imagen en la segunda vista está en l'.

- El epipolo es el punto de intersección de la línea que une los centros de las cámaras (la línea de base) con el plano imagen. Equivalentemente, el epipolo es la imagen en una vista del centro de la cámara de la otra vista.
- Un plano epipolar es el plano que contiene la línea de base.
- Una línea epipolar es la intersección de un plano epipolar con el plano imagen. Todas las líneas epipolares se intersecan en el epipolo. Un plano epipolar intereseca los planos imagen izquierdo y derecho en líneas epipolares, y define una correspondencia entre las líneas.

4.3. Matriz fundamental

La matriz fundamental es la representación algebráica de la geometría epipolar. A continuación derivaremos la matriz fundamental, para dos cámaras generales, de la correspondencia entre un punto y su línea epipolar y luego especificando las propiedades de la matriz.

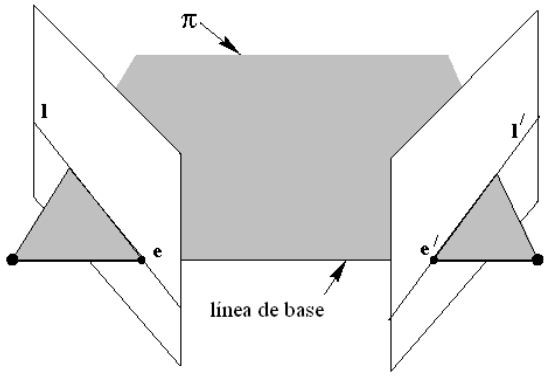

Fig. 4.2: La línea de base interseca los planos imágenes en los epipolos e y e'. Todo plano π que contenga a la línea de base es una plano epipolar e intereseca los planos imagen en las líneas l y l'.

Dado un par de imágenes, fue visto en la figura 4.1 que para cada punto x en una imagen existe una línea epipolar que le corresponde en la otra imagen. Cualquier punto x′ en la segunda imagen que se corresponda con el punto x, debe pertenecer a la línea epipolar l'. Existe por lo tanto una correspondencia

$$\mathrm{x} \mapsto l'$$

desde un punto en una imagen hacia la línea epipolar correspondiente en la otra imagen. Esta correspondencia es una correlación singular, es decir, una correspondencia proyectiva de puntos a líneas, que está representada por una matriz F, a la matriz fundamental.

La forma de la matriz fundamental en términos de las dos matrices de proyección de las cámaras puede ser derivada algebráicamente. La siguiente formulación pertenece a Xu y Zhang [13]. El rayo retroproyectado desde x por P es obtenido resolviendo $P\mathrm{X} = \mathrm{x}$. La familia de soluciones está dada por

$$\mathrm{X}(\lambda) = P^{+}\mathrm{x} + \lambda C$$

donde P^{+} es la pseudo-inversa de P y C su vector nulo, el centro de la cámara, definido por $PC = 0$. El rayo está parametrizado por el escalar λ. En particular, dos puntos en el rayo son $P^{+}\mathrm{x}$ (en $\lambda = 0$) y el centro de la primera cámara C (en $\lambda = \infty$). Estos dos puntos tienen sus imágenes, en

el plano focal de la segunda cámara, parametrizada por P', en $(P'P^{+}\mathrm{x})$ y en $(P'C)$ respectivamente. La línea epipolar es la línea que une estos dos puntos, $l' = (P'C) \times (P'P^{+}C)$. El punto $P'C$ es el epipolo en la segunda imagen llamado e'. Entonces, $l' = [e']_{\times}(P'P^{+})\mathrm{x} = F\mathrm{x}$, donde F es la matriz

$$F = [e']_{\times}P'P^{+} \tag{4.3}$$

donde,

$$\left[\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right]_{\times} = \begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix} \tag{4.4}$$

4.3.1. Condición de correspondencia

Hasta aquí hemos considerado la correspondencia $\mathrm{x} \mapsto l'$ definida por F. Podemos ahora presentar las propiedades principales de la matriz fundamental.

Resultado 4.1. *La matriz fundamental, dados dos puntos correspondientes* $\mathrm{x} \leftrightarrow \mathrm{x}'$*, satisface la condición*

$$\mathrm{x}'^{T}F\mathrm{x} = 0$$

Demostración. Si $\mathrm{x} \leftrightarrow \mathrm{x}'$, entonces x' pertenece a la línea epipolar $l' = F\mathrm{x}$ correspondiente al punto x. En otras palabras $0 = \mathrm{x}'^{T}l' = \mathrm{x}'^{T}F\mathrm{x}$. □

Del resultado anterior, se desprende la siguiente definición:

Definicion 4.1. Sean dos imágenes adquiridas por dos cámaras, con centros no-coincidentes, la matriz fundamental F es la única homogénea de dimensión 3×3 y rango 2 que satisface

$$\mathrm{x}'^{T}F\mathrm{x} = 0$$

A continuación enumeramos algunas de las propiedades de la matriz fundamental:

Traspuesta: si F es la matriz fundamental del par (P, P'), entonces F^{T} lo es del par (P', P).

Línea epipolar: para cualquier punto x en la primera imagen, la línea epipolar correspondiente es $l' = F\mathrm{x}$. En forma similar, $l = F^T\mathrm{x}'$, es la línea epipolar corespondiente a cualquier punto x' de la segunda imagen.

Epipolo: para cualquier punto x, diferente de e, la línea epipolar $l' = F\mathrm{x}$ contiene al epipolo e'. En consecuencia, se cumple $e'^T(F\mathrm{x}) = (e'^T F)\mathrm{x} = 0$ para todo x. Luego $e'^T F = 0$, e' es el espacio nulo izquierdo de F. El mismo razonamiento se aplica a $Fe = 0$, por lo que e es el espacio nulo derecho de F.

Grados de libertad: F tiene siete grados de libertad; una matriz homogénea de 3×3 tiene 8 (9 elementos y la escala común no es significante), sin embargo también se cumple $F = 0$, lo que elimina un grado de libertad.

4.4. Reconstrucción tridimensional

Un primer acercamiento consiste en proyectar las imágenes del punto y buscar su intersección (ver figura 4.3). En cada imagen tenemos $\mathrm{x} = P\mathrm{X}$, $\mathrm{x}' = P'\mathrm{X}$. Estas ecuaciones pueden ser combinadas formando $A\mathrm{X} = 0$, una ecuación lineal en X.

Planteando el producto $\mathrm{x} \times (P\mathrm{X}) = 0$ tenemos que

$$\begin{aligned} \mathrm{x}_1(p_1^T\mathrm{X}) - (p_1^T\mathrm{X}) &= 0 \\ \mathrm{x}_2(p_3^T\mathrm{X}) - (p_2^T\mathrm{X}) &= 0 \\ \mathrm{x}_1(p_2^T\mathrm{X}) - y(p_1^T\mathrm{X}) &= 0 \end{aligned}$$

donde p_i y x_i son las filas de P y x, respectivamente. Estas ecuaciones, de las cuales dos son linealmente independientes, son claramente lineales en las componentes de X. Ecuaciones similares pueden obtenerse para x' y P'.

Una ecuación de la forma $A\mathrm{X} = 0$ puede ser compuesta de la siguiente manera:

$$A = \begin{pmatrix} \mathrm{x}_1 {p_3}^T - {p_1}^T \\ \mathrm{x}_2 {p_3}^T - {p_2}^T \\ \mathrm{x}'_1 {p'_3}^T - {p'_1}^T \\ \mathrm{x}_2 {p'_3}^T - {p'_2}^T \end{pmatrix} \tag{4.5}$$

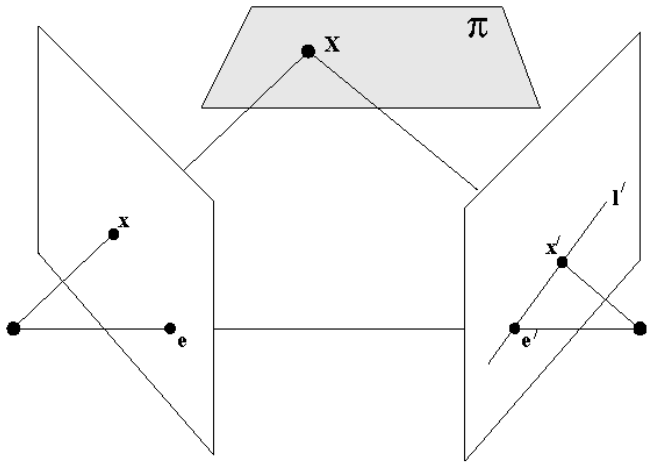

Fig. 4.3: Proyección de los rayos de las dos cámaras.

donde dos ecuaciones fueron incluídas de cada imagen, dando un total de cuatro ecuaciones con cuatro incógnitas (recordemos que X $\in \mathbb{P}^3$ [2], por lo que es un vector en $\mathbb{R}^4$).

El método DLT puede ser utilizado para encontrar una solución al sistema 4.5. Éste encuentra la solución como el vector singular unitario correspondiente al menor valor singular de A. Si la descomposición en valores singulares (*Singular Value Decomposotion*, SVD) es $A = UDVT$ con D diagonal y valores positivos ordenados descendientemente en la diagonal, entonces X es la última columna de V.

Este acercamiento es correcto desde el punto de vista teórico, pero falla en la práctica dado que, existiendo errores, las rectas obtenidas generalmente no se intersecan (ver figura 4.4). Es necesario obtener la *mejor solución* posible. Este concepto acarrea implícitamente la definición y, luego, la minimización de una función de costo acorde al problema.

4.4.1. El problema

Como hay errores de ubicación en los puntos x y x′, los rayos proyectados desde éstos no se intersecan. Esto quiere decir que no existe un punto que

[2] la contrucción de $\mathbb{P}^3$ es similar a la de $\mathbb{P}^2$, explicada en 4.1

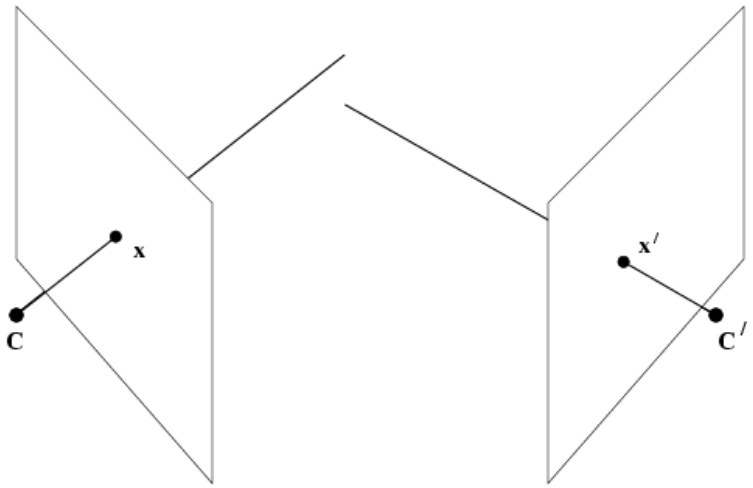

Fig. 4.4: Proyección de los rayos de las dos cámaras y cómo se recupera la posición tridimensional del punto.

satisfaga exactamente $\mathrm{x} = P\mathrm{X}$ y $\mathrm{x}' = P'\mathrm{X}$; los puntos de la imagen tampoco, por ende, satisfacen la restricción epipolar $\mathrm{x}'^T F\mathrm{x} = 0$.

Frente a una correspondencia ruidosa $\mathrm{x} \leftrightarrow \mathrm{x}'$, que no satisface la restricción epipolar se buscan los puntos $\bar{\mathrm{x}} \leftrightarrow \bar{\mathrm{x}}'$, cercanos a los originales y que si satisfagan la restricción epipolar $\bar{\mathrm{x}}'^T F\bar{\mathrm{x}} = 0$ exactamente.

En otras palabras, buscamos $\hat{\mathrm{x}}$ y $\hat{\mathrm{x}}'$ que minimizan la siguiente función

$$\mathcal{C}(\mathrm{x}, \mathrm{x}') = d(\mathrm{x}, \hat{\mathrm{x}})^2 + d(\mathrm{x}', \hat{\mathrm{x}}')^2 \quad \text{sujeto a} \quad \hat{\mathrm{x}}'^T F\hat{\mathrm{x}} = 0 \tag{4.6}$$

donde $d(*,*)$ es la distancia euclídea entre los puntos. Esto es equivalente a minimizar el error de reproyectar los puntos para un punto $\hat{\mathrm{X}}$ con puntos en las imágenes $\hat{\mathrm{x}}$ y $\hat{\mathrm{x}}'$ obtenidos mediante las matrices de proyección P y P' (consistentes con F).

Asumiendo que el error sigue una distribución gaussiana, los puntos $\hat{\mathrm{x}}$ y $\hat{\mathrm{x}}'$ son Estimadores de Máxima Verosimilitud (EMV) de la verdadera correspondencia. Una vez obtenidos $\hat{\mathrm{x}}$ y $\hat{\mathrm{x}}'$, el punto $\hat{\mathrm{X}}$ puede ser obtenido con cualquier método de triangulación, dado que los rayos correspondientes se cruzan en el espacio.

Cualquier par de puntos que satisfagan la restricción epipolar, debe pertenecer a un par de líneas epipolares en las dos imágenes. En particular, el punto óp-

timo $\hat{\mathrm{x}}$ pertenece a l y $\hat{\mathrm{x}}'$ a l'. Por otro lado, cualquier par de puntos en l y l' satisface la restricción epipolar, entre ellos el punto $\mathrm{x}_\perp$ en l más cercano a x y el punto $\mathrm{x}'_\perp$ en l' más cercano a x′. Los puntos $\mathrm{x}_\perp$ y $\mathrm{x}'_\perp$ minimizan la ecuación 4.6. Por lo tanto, $\hat{\mathrm{x}} = \mathrm{x}_\perp$ o, en forma equivalente $d(\mathrm{x}, \hat{\mathrm{x}}) = d(\mathrm{x}, \mathrm{x}_\perp) = d(\mathrm{x}, l)$. El mismo razonamiento se aplica a x′, l' y $\hat{\mathrm{x}}'$.

En resumen, buscamos minimizar

$$\mathcal{C}(l, l') = d(\mathrm{x}, l)^2 + d(\mathrm{x}', l')^2 \tag{4.7}$$

donde l y l' pueden ser cualquier par de líneas epipolares correspondientes.

La estrategia a seguir es la siguiente:

1. parametrizar el espacio de las líneas epipolares de la primera imagen con un parámetro t. Una línea epipolar en la primer imagen puede escribirse como $l(t)$;

2. usando la matriz fundamental F, computar la línea epipolar correspondiente $l'(t)$ en la segunda imagen.

3. expresar $\mathcal{C}(l(t), l'(t))$ como $\mathcal{C}(t)$ explícitamente en función de t.

De esta forma, el problema es reducido a encontrar el mínimo de una función de una sola variable:

$$\min_{\hat{\mathrm{X}}} \mathcal{C} = d(\mathrm{x}, \hat{\mathrm{x}})^2 + d(\mathrm{x}', \hat{\mathrm{x}}')^2 = \min_t \mathcal{C} = d(\mathrm{x}, l(t))^2 + d(\mathrm{x}', l'(t))^2$$

4.4.2. Detalles de la solución

Para simplificar el análisis, aplicamos una transformación para ubicar x y x′ en el origen, $(0, 0, 1)^T$ en coordenadas homogéneas [3]. Los epipolos pueden ser ubicados en $(1, 0, f)^T$ y $(1, 0, f')^T$ respectivamente. La aplicación de estas dos transformaciones no tiene efecto en la suma de los cuadrados de las distancias, y por lo tanto no afecta la naturaleza del problema.

[3] Dado que las líneas $kax + kby + kc = 0$ son en realidad la misma para todo k, los vectores $(ka, kb, kc)^T$ que las representan, forman una clase de equivalencia y se llaman homogéneos

Como $F(1,0,f)^T = (1,0,f')F = 0$, la matriz fundamental tiene una forma especial:

$$F = \begin{pmatrix} ff'd & -f'c & -f'd \\ -fb & a & b \\ -fd & c & d \end{pmatrix} \tag{4.8}$$

Denotamos $l(t)$ a la línea epipolar en la primera imagen que pasa por $(0,t,1)^T$ y el epipolo $(1,0,f)^T$. El vector que representa esta línea está dado por $(0,t,1)\times(1,0,f) = (tf,1,-t)$, por lo que la distancia cuadrada al origen x es

$$d(\mathrm{x}, l(t))^2 = \frac{t^2}{1+(tf)^2}. \tag{4.9}$$

Usando la matriz fundamental para encontrar la línea epipolar en la segunda imagen, tenemos que

$$l'(t) = F(0,t,1)^T = (-f'(ct+d), at+b, ct+d)T.$$

La distancia cuadrada de esta línea al origen x′ es

$$d(\mathrm{x}', l'(t))^2 = \frac{(ct+d)^2}{(at+b)^2 + f'^2(ct+d)^2}. \tag{4.10}$$

Sumando 4.9 y 4.10, obtenemos la distancia total

$$s(t) = \frac{t^2}{1+(tf)^2} + \frac{(ct+d)^2}{(at+b)^2 + f'^2(ct+d)^2}. \tag{4.11}$$

Queremos encontrar el mínimo de esta función. Para eso, derivaremos $s(t)$ y resolveremos $s'(t) = 0$ para buscar los extremos.

$$s'(t) = \frac{2t}{\left(1+t^2f^2\right)^2} + \frac{2(ad-bc)(at+b)(ct+d)}{\left((at+b)^2 + f'^2(ct+d)^2\right)^2}. \tag{4.12}$$

Juntando los dos términos de $s'(t)$ con un denominador e igualando el numerador a 0 obtenemos

$$\begin{aligned} s(t) &= t\left((at+b)^2 + f'^2(ct+d)^2\right)^2 \\ &\quad -(ad-bc)\left(1+f^2t^2\right)^2(at+b)(ct+d) \\ &= 0. \end{aligned} \tag{4.13}$$

Éste es un polinomio de grado 6, y los máximos y mínimos de $s(t)$ ocurrirán en las 6 raíces de este polinomio (3 mínimos y 3 máximos). En realidad sólo estamos interesados en las raíces reales, pero trabajaremos con las seis para evitar determinar si cada una es compleja o no. Simplemente, se chequea el valor de $s(\tilde{t}_i)$ en cada raíz $\tilde{t}_i$, buscando el mínimo. También es necesario chequear el valor asintótico de $s(t)$ cuando $t \to \infty$, correspondiente a una línea epipolar vertical que pasa por $(1, 0, f)$.

El método está resumido en el algoritmo que se muestra a continuación. Este tiene como objetivo, dada una correspondencia $x \leftrightarrow x'$ y la matriz fundamental F, se computa la correspondencia $\widehat{x} \leftrightarrow \widehat{x}'$ la cual minimiza el error geométrico visto en (4.6) sujeto a la constante epipolar $\widehat{x}'^T F \widehat{x} = 0$

Algoritmo 4.1 El método de triangulación óptimo

1: Definir las matrices de Transformación

$$T = \begin{pmatrix} 1 & 0 & -x \\ 0 & 1 & -y \\ 0 & 0 & 1 \end{pmatrix} y \; T' = \begin{pmatrix} 1 & 0 & -x \\ 0 & 1 & -y \\ 0 & 0 & 1 \end{pmatrix}$$

2: Remplazar la matriz fundamental F por $T'^{-T} F T^{-1}$

3: Computar los epipolos izquierdos y derechos $e = (e_1, e_2, e_3)^T$ y $e' = (e'_1, e'_2, e'_3)^T$ tal que $e'^T F = 0$ y $Fe = 0$. Normalizar e (multiplicar por una escala) tal que $e_1^2 + e_2^2 = 1$ y $e_1'^2 + e_2'^2 = 1$.

4: Crear las matrices

$$R = \begin{pmatrix} e_1 & e_2 & 0 \\ -e_2 & e_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} y \; R' = \begin{pmatrix} e_1' & e_2' & 0 \\ -e_2' & e_1' & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Notar que R y R' son matrices de rotación tal que $Re = (1, 0, e_3)^T$ y $R'e' = (1, 0, e'_3)^T$.

5: Remplazar F por $R'^F R^T$

6: Definir $f = e_3$, $f' = e'_3$, $a = F_{22}$, $b = F_{23}$, $c = F_{32}$ y $d = F_{33}$

7: Obtener las 6 raices del polinomio que se muestra en la ecuación (4.13).

8: Evaluar la función $g(t)$ (4.11) con los valores correspondientes a la parte real de cada raiz obtenida en el paso anterior. Encontrar el valor de la ecuación (4.6) para $t \to \infty$. Seleccionar el t_{min} entre las 7 evaluaciones hechas.

9: Evaluar las dos líneas $l = (tf, l, -t)$ y l' dada por la ecuación (4.10) con el t_{min} determinado y encontrar $\widehat{x}$ y $\widehat{x}'$ que son los puntos pertenecientes a las lineas más cercanas al origen. Para una línea (λ, μ, ν) la fórmula para detectar el punto más cercano al origen es $(-\lambda\nu, -\mu\nu, \lambda^2 + \mu^2)$.

10: Trasladar las coordenadas originales reemplazando $\widehat{x}$ por $T^{-1} R^T \widehat{x}$ y $\widehat{x}'$ por $T'^{-1} R'^T \widehat{x}'$.

11: El punto $\widehat{X}$ en el espacio 3-D se obtiene resolviendo el método homogéneo mencionado en la ecuación (4.7).

5. SLAM

Mapeo y localización simultáneos (SLAM), es una técnica utilizada por robots para generar mapas de ambientes desconocidos mientras al mismo tiempo se actualiza la posición absoluta del robot en dicho ambiente. La generación del mapa se basa en la información obtenida por distintos sensores, la cual suele ser inexacta causando pequeños errores a medida que el robot se desplaza, que al acumularse pueden transformarse en errores más groseros.

El robot recolecta información del ambiente para poder ir generando el mapa 3D y al mismo tiempo conocer su posición dentro de este. Dicha información proviene de una única fuente, un par de cámaras que capturan imágenes del ambiente simultáneamente. El proceso que se inicia a partir de la captura de dos imágenes y finaliza inmediatamente antes de la captura del siguiente par es llamado **iteración**.

En la iteración I_i se toman dos imágenes que llamaremos ${Im_i}^{izq}$ a la imagen capturada por la cámara izquierda en la iteración i, y ${Im_i}^{der}$ a la imagen de la cámara derecha en la misma iteración.

El presente capítulo se divide en cuatro secciones, en la primera se definen una serie de suposiciones en las que se basa el algoritmo, luego se realiza una introducción vinculando los distintos mecanismos mencionados en capítulos anteriores, en tercer lugar se detalla el mecanismo por el cual se determina la posición del robot en cada iteración y por último se muestra cómo se efectúa la actualización del mapa 3D.

5.1. Suposiciones

El algoritmo de SLAM desarrollado en el presente trabajo presenta 4 suposiciones a ser consideradas:

1 La posición y orientación de una cámara no varía con respecto de la

otra.

2 Las imágenes capturadas por la cámara izquierda y derecha son tomadas simultáneamente.

3 No hay objetos en movimiento en el ambiente, o sea, el robot se desplaza por un ambiente estático.

4 El movimiento del robot se efectúa sobre el plano del piso, éste no presenta variaciones en la altura o inclinaciones.

5.2. Introducción

En el comienzo de la iteración I_i, se cuenta con las imágenes ${Im_i}^{izq}$ y ${Im_i}^{der}$ a partir de las cuales se generan dos conjuntos de características ${CC_i}^{izq}$ y ${CC_i}^{der}$ respectivamente aplicando los conceptos vistos en el capítulo 2.1.

A partir de los conjuntos de características mencionados se genera un conjunto de correspondencias que llamamos CCO_i mediante el algoritmo de armado de correspondencias que se detalla en la sección 2.2. En forma robusta se filtran las correspondencias ouliers de CCO_i mediante el algoritmo presentado en 3.2. Por último CCO_i es utilizado para generar un conjunto de puntos 3D que llamamos $CP3_i$ aplicando los conceptos vistos en el capítulo 4.

En la figura 5.1 se muestra el proceso mencionado para la iteración I_i.

La posición del robot en la iteración I_i se expresa de la siguiente manera $P_i = (X_i, Z_i, \theta_i)$, siendo X_i y Z_i las coordenadas del robot en el plano del piso y θ_i su orientación. La posición inicial del robot (iteración I_0) es $P_0 = (0, 0, 0)$, para cada par de imagenes capturadas, se generará una nueva iteración que dará lugar a una nueva posición del robot.

Los puntos 3D calculados en cada iteración se encuentran expresados en forma relativa a la posición $(0, 0, 0)$, sólo se calculará su posición absoluta para agregar información al mapa 3D.

A continuación se detalla el proceso por el cual se calcula la nueva posición del robot en cada iteración.

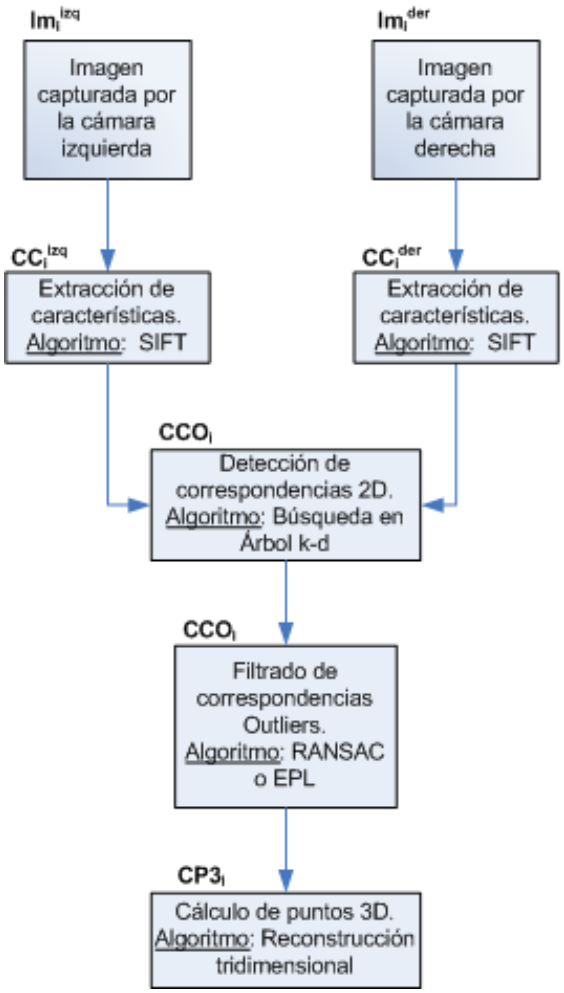

Fig. 5.1: Proceso realizado desde la captura de imágenes hasta el cálculo de los puntos 3D.

5.3. Cálculo de la posición

Para poder calcular P_{i+1} se debe contar con información de la iteración I_i ($CP3_i$ y P_i) y de la iteración I_{i+1} ($CP3_{i+1}$). El procedimiento se divide en 3 fases principales las cuales se mencionan a continuación y serán detalladas más adelante en el capítulo:

- La primer fase llamada **Detección de correspondencias 3D** se basa en determinar las correspondencias entre los puntos 3D de dos iteraciones consecutivas.
- En la segunda fase se determina la rotación y traslación realizadas por el robot para llegar de la posición P_i a la P_{i+1}, basándose en la información obtenida en el paso anterior. Llamaremos a esta fase **Cálculo del movimiento**.
- Por último, en base a la rotación y traslación calculadas, se determina la posición P_{i+1} y se trasladan los puntos de $CP3_{i+1}$. Llamamos a esta fase **Actualización**.

Habiendo presentado cada fase del procedimiento nos disponemos a entrar en detalle en cada una de ellas.

5.3.1. Detección de correspondencias 3D

El objetivo de esta fase es determinar la correspondencia entre puntos 3D detectados en iteraciones consecutivas. Notamos a la correspondencia c de la siguiente manera $c =< a, b >$, donde a y b son las características que la conforman.

Se tiene un punto 3D correspondiente a la iteración I_i compuesto por las siguientes características ${c_i}^{izq}$ y ${c_i}^{der}$, y otro correspondiente a la iteración I_{i+1} compuesto por las siguientes características ${c_{i+1}}^{izq}$ y ${c_{i+1}}^{der}$. Diremos que estos dos puntos 3D se corresponden si se cumplen las siguientes condiciones:

- $< {c_i}^{izq}, {c_{i+1}}^{izq} > \in {CCO_i}^{izq}$
- $< {c_i}^{der}, {c_{i+1}}^{der} > \in {CCO_i}^{der}$

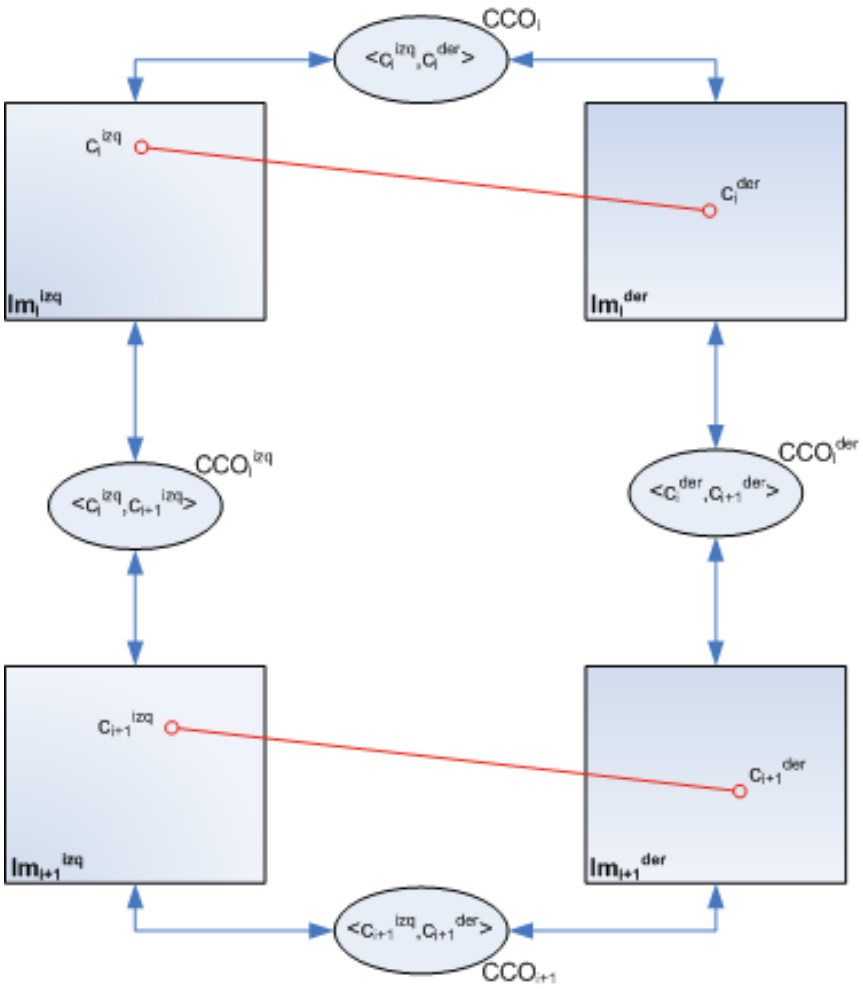

Fig. 5.2: Correspondencia entre las características de distintas iteraciones.

siendo $CCO_i{}^{izq}$ el conjunto de correspondencias determinado por las características de las imágenes $Im_i{}^{izq}$ e $Im_{i+1}{}^{izq}$, y de la misma forma $CCO_i{}^{der}$ se determina a partir de las características de las imágenes $Im_i{}^{der}$ e $Im_{i+1}{}^{der}$. Para determinar las correspondencias que componen $CCO_i{}^{izq}$ y $CCO_i{}^{der}$ se emplea el mecanismo visto en 2.2.

En la figura 5.2 se muestra gráficamente cómo se establece la correspondencia de características en distintas iteraciones, donde cada correspondencia aparece expresada como un par de características.

Con todas las correspondencias de puntos 3D se genera un conjunto que

llamaremos $CCO3$, de esta manera en la iteracion I_{i+1} calcularemos el conjunto $CCO3_{i+1}$ con puntos 3D detectados en las iteraciones I_i e I_{i+1}. Este conjunto de correspondencias se utilizará para poder estimar el movimiento del robot realizado entre la iteración I_i y la I_{i+1}.

5.3.2. Cálculo del movimiento

El cálculo del movimiento realizado por el robot entre la iteración I_i y la I_{i+1} se basa en el conjunto de correspondencias 3D calculado en el paso anterior. El movimiento del robot entre la iteración I_i y la I_{i+1} se expresa de la siguiente forma $M_i = (dx_i, dz_i, \beta_i)$, luego dicho movimiento M_i se utilizará para actualizar la posición del robot el mapa.

Llamamos $\widehat{CP3}_i$ al conjunto de todos los puntos 3D detectados en la iteración I_i que pertenecen a alguna de las correspondencias que forman $CCO3_{i+1}$, de la misma manera llamaremos $\widetilde{CP3}_{i+1}$ a los puntos 3D detectados en I_{i+1} que pertenecen a alguna de las correspondencias que forman $CCO3_{i+1}$.

Dado que el movimiento se efectua realizando un desplazamiento y rotación sobre el plano del piso, los cálculos de la nueva posición se realizan basándose en las cordenadas X y Z de los puntos 3D detectados.

El movimiento que realiza un robot durante una iteración puede descomponerse en una rotación (cambio en la orientación) y una traslación (cambio en la posición). La idea para calcular el movimiento realizado se basa en rotar y transladar los puntos de $\widehat{CP3}_i$ de forma tal que se minimice la diferencia con los puntos 3D correspondientes del estado siguiente ($\widetilde{CP3}_{i+1}$). En la ecuación 5.1 se muestra la fórmula utilizada, donde $g : \Re^2 \rightarrow \Re^2$ es la función de rotación (rota un ángulo β) y traslación (traslada dx en el eje x y dz en el eje z).

$$\min_{\beta,dx,dz} \sum_{j=1}^{\#\widehat{CP3}_i} \|{p_{i+1}}^j - g({p_i}^j, \beta, dx, dz)\|^2 \tag{5.1}$$

La rotación de los puntos 3D se calcula a partir de las ecuaciones de rotación que se muestran en la ecuación 5.2. A partir de ellas se obtienen las coordenadas resultantes de rotar un punto (x, z) un ángulo β.

$$\begin{aligned} x' &= x\cos\beta - z\sin\beta \\ z' &= x\sin\beta + z\cos\beta \end{aligned} \tag{5.2}$$

En la ecuación 5.3 se muestra la función de rotación y traslación de un punto que llamamos $g : \Re^4 \to \Re^2$, donde a cada punto se lo rota un ángulo β y se lo desplaza (dx, dz).

$$g(p_i, \beta, dx, dz) = ({p_i}^x \cos\beta - {p_i}^z \sin\beta + dx, {p_i}^x \sin\beta + {p_i}^z \cos\beta + dz) \tag{5.3}$$

Combinando las ecuaciones 5.1 y 5.3 obtenemos la ecuación 5.4, mediante ésta se calcula el movimiento buscado. Desarrollando la ecuación 5.4 se obtiene la ecuación 5.5 y luego sustituyendo la ecuación 5.6.

$$\min_{\beta,dx,dz} \sum_{j}^{\#\widehat{CP3}_i} \|({p_{i+1}}^{jx}, {p_{i+1}}^{jz}) - ({p_i}^{jx}\cos\beta - {p_i}^{jz}\sin\beta + dx, {p_i}^{jx}\sin\beta + {p_i}^{jz}\cos\beta + dz)\|^2 \tag{5.4}$$

$$\min_{\beta,dx,dz} \sum_{j}^{\#\widehat{CP3}_i} \|(\underbrace{{p_{i+1}}^{jx} - {p_i}^{jx}\cos\beta + {p_i}^{jz}\sin\beta - dx}_{A}, \underbrace{{p_{i+1}}^{jz} - {p_i}^{jx}\sin\beta - {p_i}^{jz}\cos\beta - dz}_{B})\|^2 \tag{5.5}$$

$$\min_{\beta,dx,dz} \sum_{j}^{\#\widehat{CP3}_i} \left(\sqrt{A^2 + B^2}\right)^2 \tag{5.6}$$

En la ecuación 5.7 se muestra un problema típico de cuadrados mínimos, donde $f_i : \Re^n \to \Re, i = 1, \ldots, m$.

$$\min_{x} \sum_{i}^{m} f_i(x)^2 \tag{5.7}$$

Si tomamos $x = (\beta, dx, dz)$ y $f_i(x) = \sqrt{A^2 + B^2}$, podemos resolver la ecuación 5.6 utilizando el método de cuadrados mínimos. Como la función f es una función no lineal, se debe utilizar una variante a la solución clásica de cuadrados mínimos llamada **cuadrados mínimos no lineales**.

Utilizando la resolución propuesta no se obtienen los resultados esperados, para mejorarlos se utiliza un mecanismo similar al anterior aunque en este caso la función g se remplaza por dos funciones, una de rotación (g_1) y otra de traslación (g_2), la cuales se muestran a continuación:

$$\begin{aligned} g_1(p_i, \beta) &= ({p_i}^x \cos\beta - {p_i}^z \sin\beta, {p_i}^x \sin\beta + {p_i}^z \cos\beta) \\ g_2(p_i, dx, dz) &= ({p_i}^x + dx, {p_i}^z + dz). \end{aligned} \tag{5.8}$$

De la misma forma que con g, a partir de g_1 y g_2 llegamos a las ecuaciones 5.9 y 5.10 respectivamente.

$$\min_{\beta} \sum_{j}^{\#\widehat{CP3}_i} \|(\underbrace{{p_{i+1}}^{jx} - {p_i}^{jx}\cos\beta + {p_i}^{jz}\sin\beta}_{A}, \underbrace{{p_i}^{jz} - {p_i}^{jx}\sin\beta - {p_i}^{jz}\cos\beta}_{B})\|^2 \tag{5.9}$$

$$\min_{dx,dz} \sum_{j}^{\#\widehat{CP3}_i} \|(\underbrace{{p_{i+1}}^{jx} - {p_i}^{jx} - dx}_{A}, \underbrace{{p_{i+1}}^{jz} - {p_i}^{jz} - dz}_{B})\|^2 \tag{5.10}$$

El nuevo procedimiento entonces se basa en determinar primero la rotación β resolviendo la ecuación 5.9 mediante cuadrados mínimos no lineales, luego aplicar dicha rotación β a los puntos de $\widehat{CP3}_i$ utilizando la ecuación 5.2, y por último determinar el desplazamiento (dx, dz) resolviendo la ecuación 5.10, pero tomando los puntos pertenecientes a $\widehat{CP3}_i$ ya rotados.

Habiendo calculado la rotación y translación realizada por el robot, sólo resta determinar la posición actual del robot y transladar los puntos 3D detectados en la última iteración para agregarlos al mapa.

5.3.3. Actualización

La posición del robot en la iteración I_{i+1} que llamamos P_{i+1} se construye a partir de la posición P_i y el movimiento $M_i = (dx_i, dz_i, \beta_i)$ calculado en la sección anterior.

La actualización de la posición del robot se realiza aplicando la ecuación 5.11.

$$P_{i+1} = P_i + M_i \tag{5.11}$$

Habiendo actualizado la posición, sólo resta actualizar los puntos 3D pertenecientes al conjunto $\widetilde{CP3}_{i+1}$, dicho procedimiento se realiza siguiendo la ecuación 5.12, donde $p' = (x', y', z')$ es el punto rotado y transladado, y $p = (x, y, z)$ es el punto con posición relativa $(0, 0, 0)$ inicialmente detectado.

$$p' = (dx + x * \cos\beta + z * \sin\beta, y, dz + x * \sin\beta + z * \cos\beta) \tag{5.12}$$

Habiendo determinado la nueva posición del robot y calculado la posición absoluta de los puntos 3D en el ambiente, describiremos cómo se realiza la actualización del mapa en función de dichos puntos.

5.4. Actualización del mapa 3D

Un mapa 3D es un conjunto de puntos 3D que representa el ambiente que rodea al robot. Dicho mapa también contiene información sobre el recorrido realizado por el robot.

Luego de haber actualizado la posición de los puntos, algunos de ellos se agregan al mapa 3D y se actualiza la posición del robot en éste. Los puntos que se agregan al mapa en la cada iteración son los puntos 3D para los cuales no se determinó una correspondencia 3D. Esto se hace con el objetivo de repetir en menor cantidad los puntos 3D que se agregan al mapa.

Teniendo la posición y el mapa actualizados se repite el proceso completo para la siguiente iteración.

5.5. El algoritmo

El mecanismo completo se muestra en el algoritmo 5.1.

Algoritmo 5.1 SLAM: Se genera un mapa 3D y el recorrido realizado por el robot en función de la información de las cámaras. El algoritmo describe una iteración cualquiera del algoritmo, donde la información de la iteración anterior se detalla con el sufijo "Old"

1: $Im^{izq} \leftarrow$ Imagen de entrada capturada por la cámara izquierda
2: $Im^{der} \leftarrow$ Imagen de entrada capturada por la cámara derecha
3: $CCO \leftarrow$ ExtracciónDeCorrespondenciasInliers(Im^{izq},Im^{der}) (Capítulos 2 y 3)
4: $CP3 \leftarrow$ Triangulación3D(CCO) (Capítulo 4)
5: **if** EsPrimeraIteracion() **then**
6: GuardarPuntos($CP3$) (Sección 5.4)
7: $POld \leftarrow (0, 0, 0)$ (Sección 5.2)
8: GuardarPosicion($POld$) (Sección 5.4)
9: $CP3Old \leftarrow CP3$
10: RETURN
11: **end if**
12: $CCO3 \leftarrow$ DetectarCorrespondencias3D($CP3$, $CP3Old$) (Sección 5.3.1)
13: $M \leftarrow$ CalcularMovimiento($CCO3$) (Sección 5.3.2)
14: $P \leftarrow$ ActualizarPosicion($POld$,M) (Sección 5.3.3)
15: $CP3Trans \leftarrow$ ActualizarPuntos(P,$CP3$) (Sección 5.3.3)
16: GuardarPuntos($CP3Trans$) (Sección 5.4)
17: GuardarPosicion(P) (Sección 5.4)
18: $POld \leftarrow P$
19: $CP3Old \leftarrow CP3$
20: RETURN

6. RESULTADOS Y DISCUSIÓN

En este capítulo mostraremos los resultados obtenidos en la implementación de los métodos desarrollados que fueron expuestos en los capítulos anteriores.

Para probar los algoritmos implementados se realizaron 4 casos de prueba. En cada uno de los casos se hizo recorrer al robot una trayectoria distinta, y a medida que este se desplazaba, se fue capturando imágenes estéreo las cuales se utilizaron para generar un mapa 3D e ir actualizando la posición del robot. A lo largo de este capítulo se citarán los casos mencionados y en el final se detallarán los recorridos realizados por el robot en cada uno y los mapas generados. Llamaremos a los casos de prueba de la siguiente forma: caso 1, caso 2, caso 3 y caso 4.

El robot utilizado para ejecutar las pruebas se llama fenbot, y posee la capacidad de realizar 4 tipos de movimientos: moverse hacia adelante, moverse hacia atrás, girar hacia la izquierda y girar hacia la derecha. A partir de combinaciones de estos movimientos se generan las trayectorias para los casos de prueba.

Habiendo presentado información básica para comprender el origen de los resultados, nos disponemos a presentar los resultados obtenidos.

En la sección 2.1 del capítulo 2 se presentó el algoritmo SIFT utilizado para la extracción de características. Estas características son detectadas basándose en la noción de espacio-escala a través de la relación con los máximos y mínimos locales de la función Diferencia de Gaussianas. Para describir cada característica se genera un descriptor que contiene información de la región que la rodea, con el objeto de obtener invarianza a rotaciones, a cada descriptor se le asigna un gradiente basado en las propiedades locales de la imagen alrededor del punto. Dicho gradiente se compone de una orientación

y una magnitud.
Con la información obtenida se genera un histograma formado por los gradientes de las orientaciones de la región alrededor de un punto clave. Inicialmente se detecta el máximo absoluto y algunos máximos locales. Con cada uno de ellos se genera un punto clave con su orientación. Entonces tendremos uno o más puntos claves situados en la misma posición pero con distintas orientaciones.
En la figura 6.1 se muestra el resultado de aplicar el algoritmo SIFT a una imagen tomada por la cámara izquierda del robot. Cada correspondencia se representa por una o más flechas, donde la orientación vinculada al gradiente está representada por el ángulo de la flecha y la magnitud por su longitud. Del mismo modo, en la figura 6.2 se muestra el resultado de aplicar el algoritmo SIFT a una imagen tomada por la cámara derecha del robot correspondiente al mismo plano que la imagen anterior.

En la sección 2.2 del capítulo 2 se presentó un algoritmo utilizado para determinar correspondencias entre características utilizando árboles k-d. A partir de dos fuentes de descriptores se puede establecer un conjunto de correspondencias. Para ello se debe construir un árbol k-d con una de la fuentes de descriptores y luego para cada uno de los descriptores de la otra fuente se debe realizar la búsqueda sobre el árbol. Utilizando este mecanismo el conjunto de correspondencias resultante puede contener correspondencias incorrectas que llamamos outliers.
La figura 6.3 muestra el resultado de aplicar el algoritmo de detección de correspondencias, donde para cada correspondencia detectada se traza una línea que une las características que la componen. Las líneas rosas indican que la correspondencia detectada es correcta, y las negras indican lo contrario. Del mismo modo en la figura 6.4 se muestran los resultados de la detección de correspondencias en un ambiente distinto. En la figura 6.5 se ve el resultado de la detección de correspondencias para dos imágenes de alta calidad y del mismo tamaño que las anteriores, como se nota en la figura la cantidad de correspondencias detectadas es considerablemente superior a las figuras anteriores.

En el capítulo 3 se presentan dos algoritmos para eliminar correspondencias outliers, el primero llamado RANSAC y el segundo EPL. La implementación de ambos algoritmos otorgó resultados muy similares en cantidad de correspondencias outliers detectadas y tiempo consumido, dada dicha similitud, sólamente se mostrarán los resultados del algoritmo EPL. Para las distintas pruebas el coeficiente *ce* utilizado para las características pendiente y longitud es $ce = 2,5$.

En la figura 6.6 se muestra la relación entre la cantidad de correspondencias outliers detectadas y el coeficiente *ce* utilizado para la pendiente y longitud (se utiliza para ambas características el mismo coeficiente). La región marcada en gris que comprende los valores $ce \in [2,6]$ contiene los valores que presentan resultados muy similares a los obtenidos mediante el algoritmo RANSAC. En el gráfico se nota que a medida que *ce* decrece, la cantidad de outliers aumenta.

La figura 6.7 muestra el resultado de aplicar el algoritmo EPL al resultado de la figura 6.3. Se nota que las correspondencias outliers han sido eliminadas, resultando un conjunto compuesto únicamente por correspondencias inliers. En la figura 6.8 se muestra el resultado de aplicar el algoritmo EPL al resultado de la figura 6.4, y en la figura 6.9 el resultado de aplicar el algoritmo al resultado de la figura 6.5.

En el capítulo 5 se presenta el algoritmo SLAM, encargado de mantener actualizada la posición del robot al mismo tiempo que genera un mapa del ambiente por el cual éste se desplaza. Llamamos recorrido a la trayectoria realizada por el robot que se genera uniendo con líneas rectas las posiciones consecutivas. Llamamos iteración al proceso que se inicia a partir de la captura de dos imágenes y finaliza inmediatamente antes de la captura del siguiente par, entonces tendremos un movimiento (o más dependiendo del caso) o paso realizado por el robot asociado a cada iteración.

En las tablas 6.1, 6.2, 6.3 y 6.4 se muestran los pasos que realizó el robot durante cada iteración en los distintos casos de prueba. Cada paso está representado por un símbolo que identifica el movimiento realizado. El simbolo $\circ$ indica punto de partida desde la posición $P_0 = (0,0,0)$, $\uparrow$ un movimiento de 10 cm hacia adelante, $\downarrow$ un movimiento de 10 cm en retroceso, $\leftarrow$ un giro

de 10^o hacia la izquierda y $\rightarrow$ un giro de 10^o hacia la derecha.
Los errores en el movimiento del robot pueden atribuirse principalmente a cuatro fuentes: la fuerza con que se hace girar cada rueda, la adherencia del piso en distintos sectores del ambiente, la adherencia de cada rueda al piso y la inclinación del piso. Así, en cada movimiento pueden ocurrir distintos errores como combinación de las fuente de error mencionadas. Como conclusión entendemos que las tablas presentadas sirven únicamente como guías para comprender los resultados, pero no reflejan el recorrido real efectuado por el robot.

Las figuras 6.10, 6.13, 6.15, 6.17 y 6.18 muestra el ambiente por el cual se desplaza el robot en cada caso de prueba, siendo de ayuda para comprender los mapas que se muestran en las figuras 6.12, 6.14, 6.16 y 6.21 respectivamente. Los mapas se generaron en dos dimensiones (X, Z) para mejor comprensión de los resultados.

Para los casos 1 y 4 se muestra un detalle del recorrido realizado. El recorrido del caso 1 se muestra en la figura 6.11, y el recorrido del caso 4 se muestra en la figura 6.19, y es ampliado en la figura 6.20.

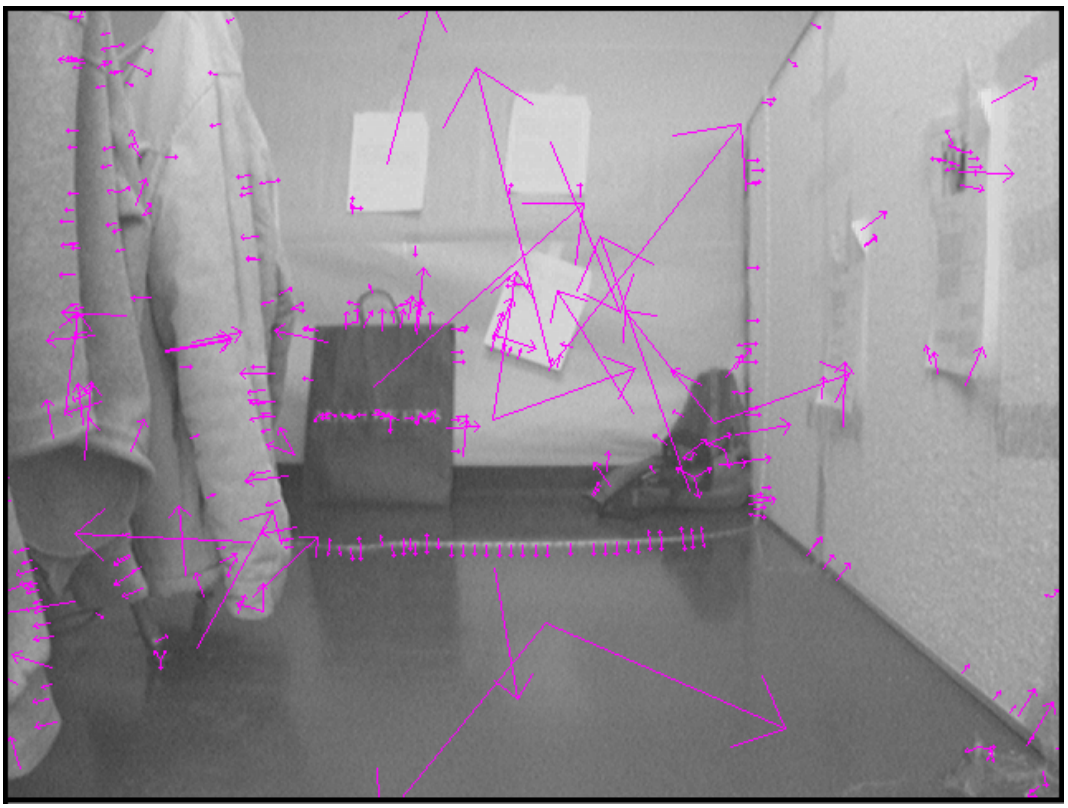

Fig. 6.1: Imagen correspondiente a la cámara izquierda de la primer iteración del caso 4, donde se muestran las características detectadas mediante el algoritmo SIFT. Por cada característica se ven flechas que representan sus respectivas orientaciones y magnitudes.

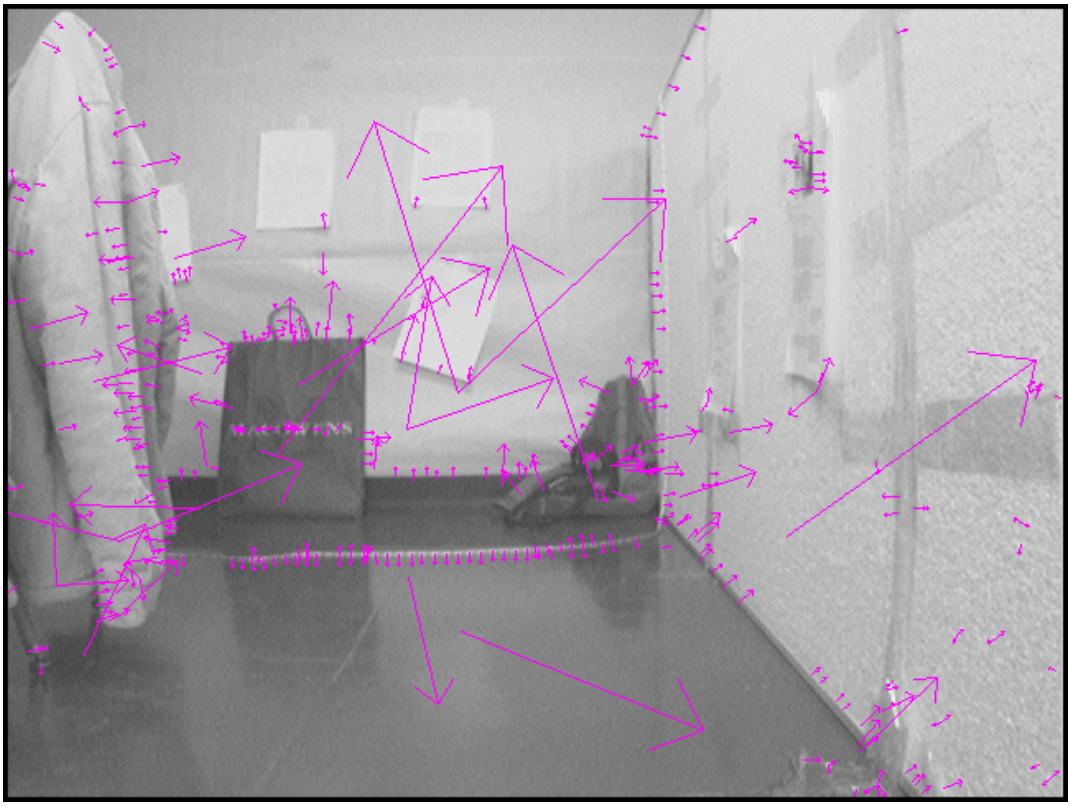

Fig. 6.2: Idem a la figura 6.1, la imagen corresponde a la cámara derecha de la primer iteración del caso 4.

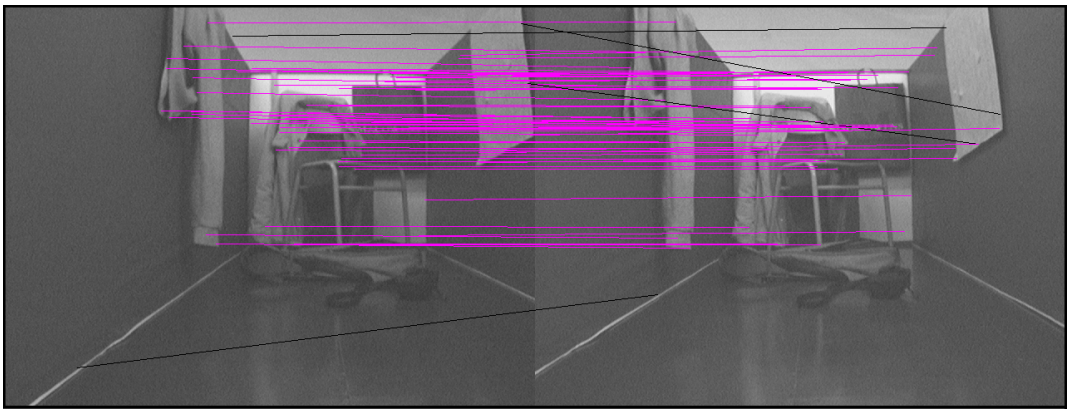

Fig. 6.3: Correspondencias detectadas utilizando un árbol k-d para dos imágenes correspondientes a la primer iteración del caso 1. En rosa se notan las líneas que unen las características de las correspondencias detectadas correctamente (71 correspondencias), y en negro las líneas que unen las características de las correspondencias detectadas incorrectamente (4 correspondencias).

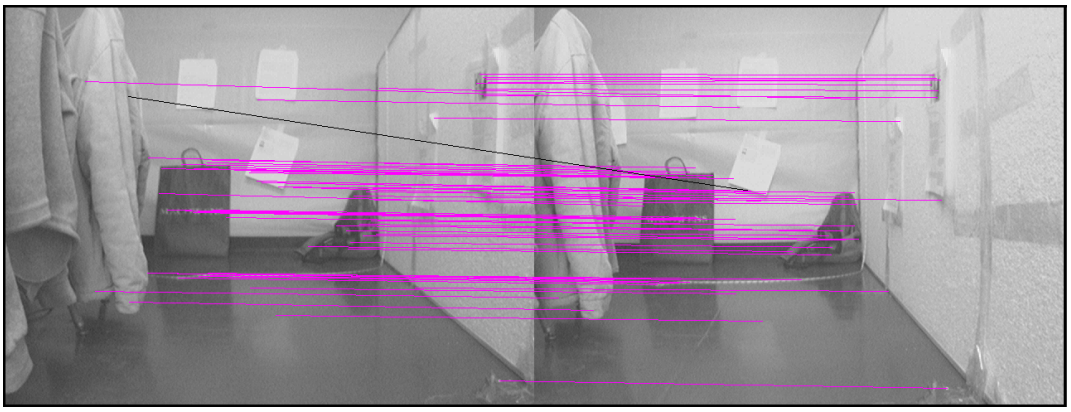

Fig. 6.4: Correspondencias detectadas utilizando un árbol k-d para las dos imágenes correspondientes a la primer iteración del caso 4. Se muestran 66 correspondencias detectadas correctamente y 1 incorrectamente.

Fig. 6.5: Correspondencias detectadas utilizando un árbol k-d para dos imágenes de alta calidad. Se muestran 222 correspondencias correctas y 13 incorrectas.

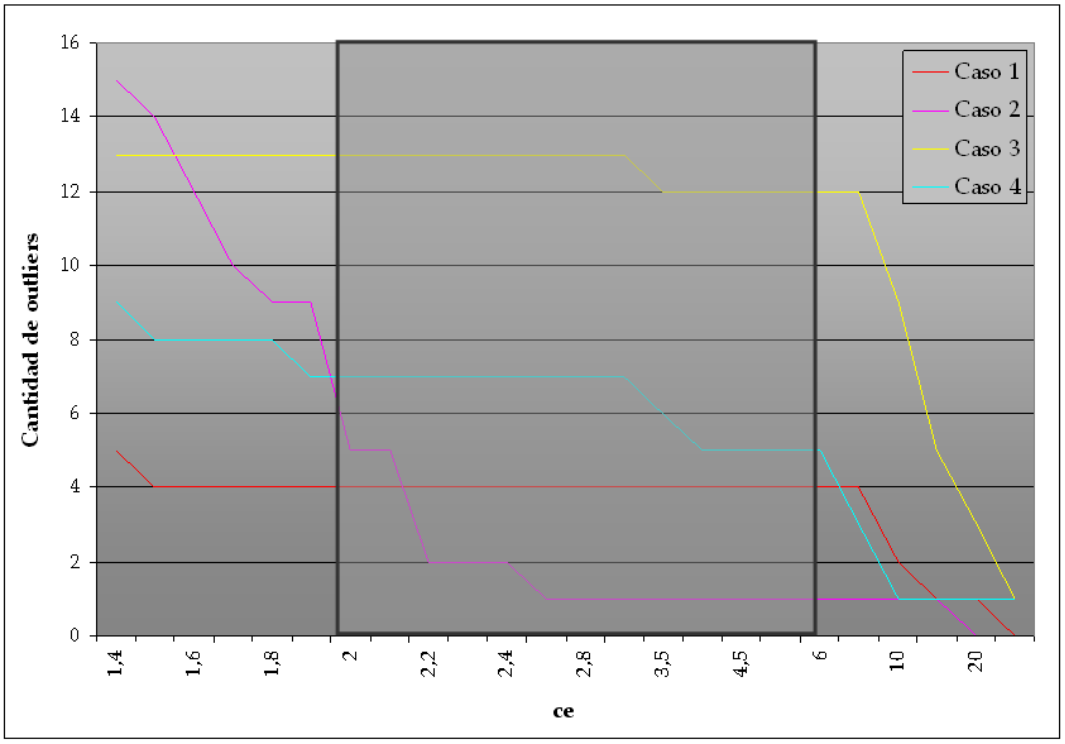

Fig. 6.6: Se muestra como varía la cantidad de correspondencias outliers detectadas por al algoritmo de Eliminación por pendiente y longitud en relación a los valores del coeficiente *ce* de las características pendiente y longitud. El recuadro gris señala el rango de valores para los cuales los resultados obtenidos son similares a los obtenidos utilizando el algoritmo RANSAC.

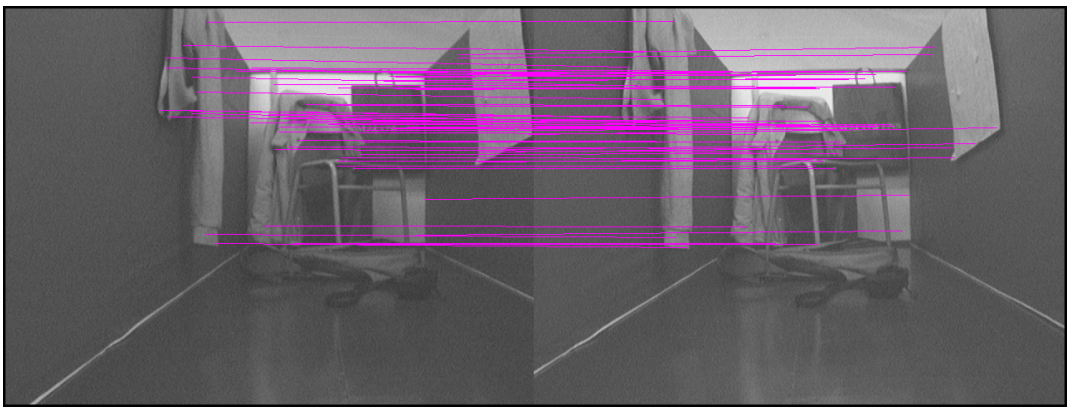

Fig. 6.7: Correspondencias detectadas utilizando un árbol k-d para dos imágenes correspondientes a la primer iteración del caso 1. En rosa se notan las líneas que unen las características de las carrespondencias que fueron consideradas inliers por el algoritmo EPL.

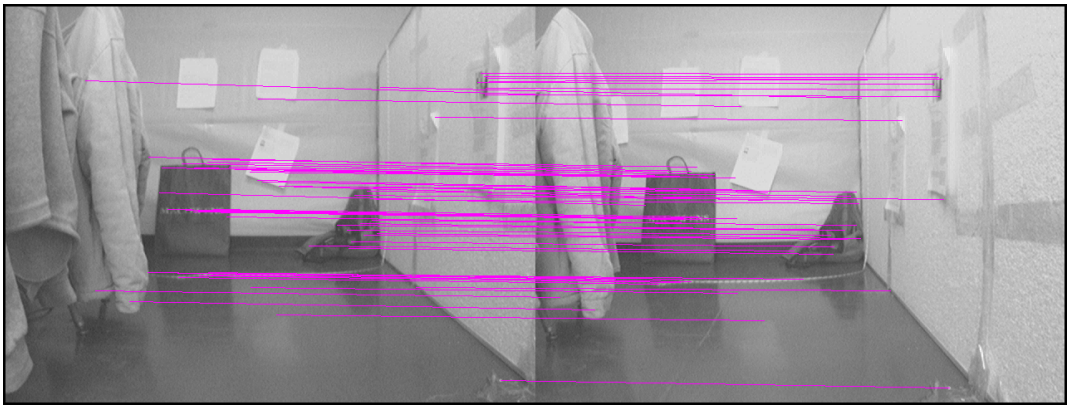

Fig. 6.8: Correspondencias detectadas utilizando un árbol k-d para dos imágenes correspondientes a la primer iteración del caso 4. En rosa se notan las líneas que unen las características de las carrespondencias que fueron consideradas inliers por el algoritmo EPL.

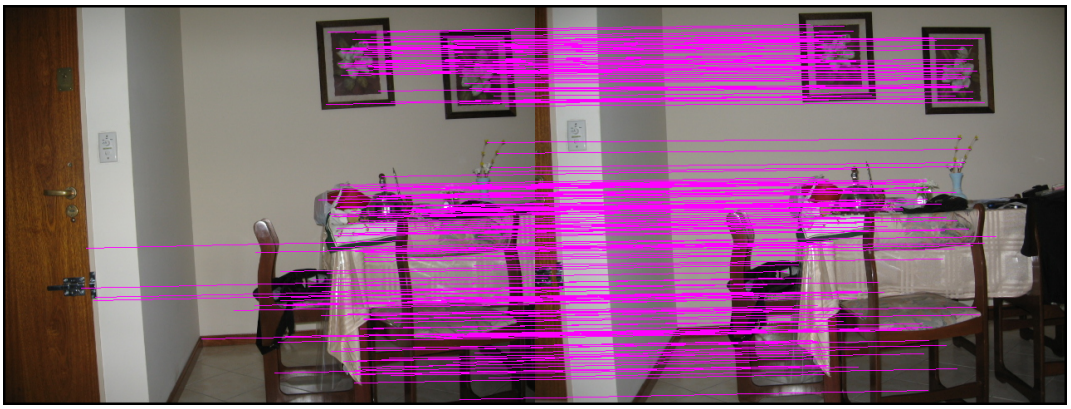

Fig. 6.9: Correspondencias detectadas utilizando un árbol k-d para dos imágenes de alta calidad. En rosa se notan las líneas que unen las características de las carrespondencias que fueron consideradas inliers por el algoritmo EPL.

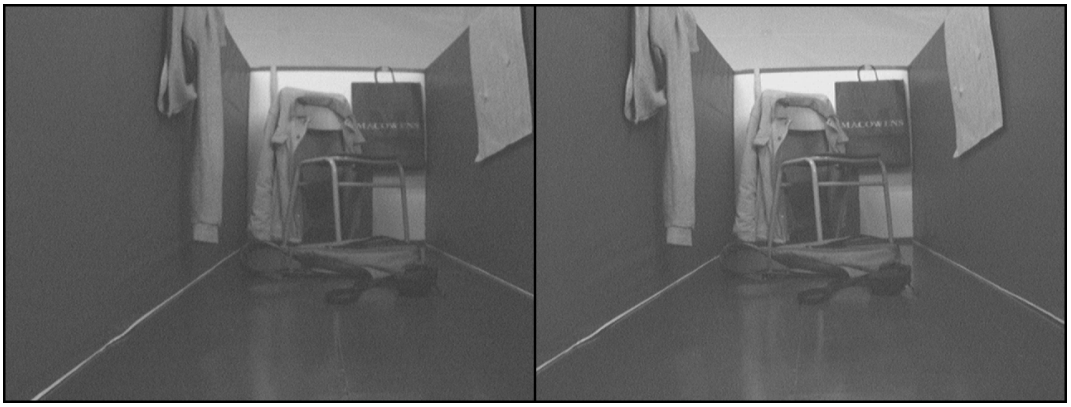

Fig. 6.10: Imágenes correspondientes a la primer iteración del caso 1. A partir de ellas se puede comprender el ambiente que rodea al robot.

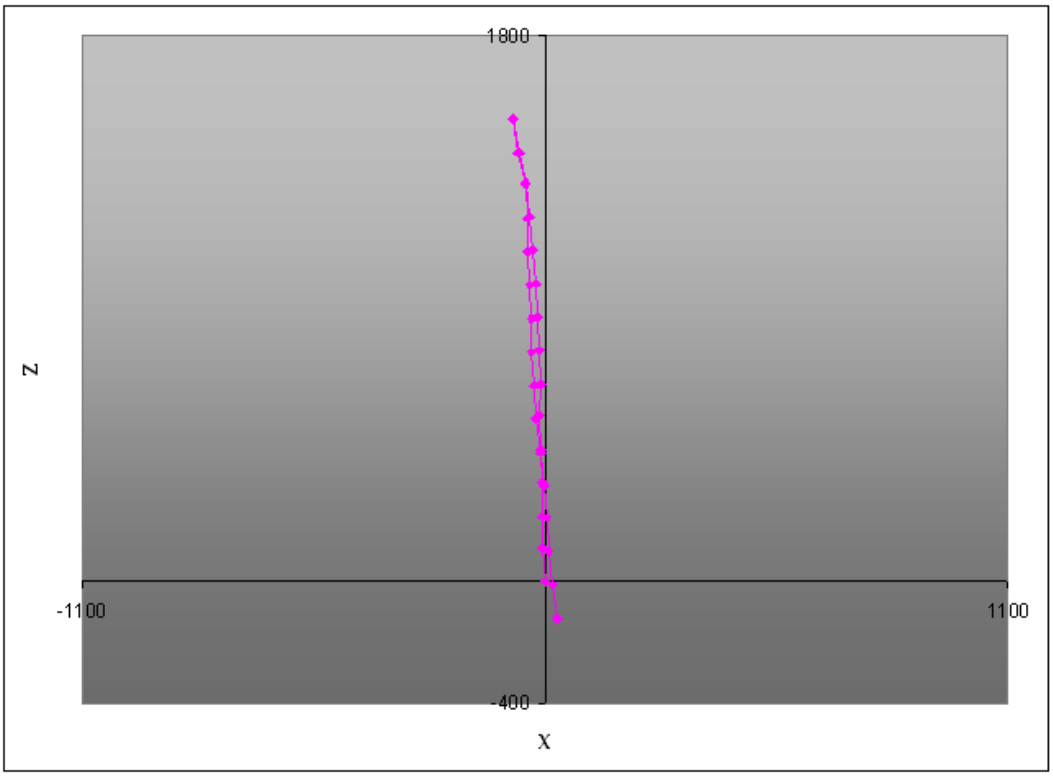

Fig. 6.11: Recorrido generado uniendo las posiciones del robot calculadas en cada iteración correspondientes al caso 1, este recorrido se corresponde con la secuencia de pasos que se muestra en la tabla 6.1.

Paso	Movimiento
0	∘
1	↑
2	↑
3	↑
4	↑
5	↑
6	↑
7	↑
8	↑
9	↑
10	↑
11	↑
12	↑
13	↑
14	↑
15	↓
16	↓
17	↓
18	↓
19	↓
20	↓
21	↓
22	↓
23	↓
24	↓
25	↓
26	↓
27	↓
28	↓
29	↓
30	↓

Tab. 6.1: Secuencia de pasos efectuados por el robot durante el recorrido del caso 1, donde ∘ indica el punto de partida, ↑ un movimiento de 10 cm hacia adelante y ↓ un movimiento de 10 cm en retroceso.

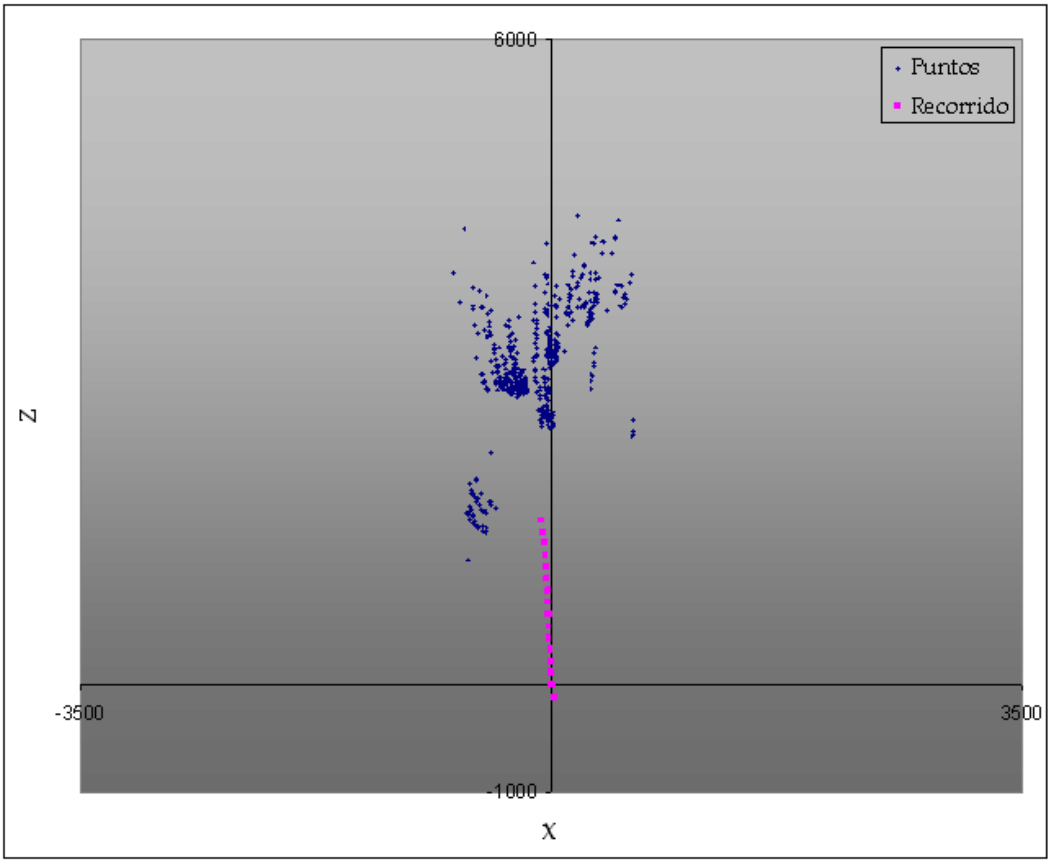

Fig. 6.12: Mapa en dos dimensiones correspondiente al caso 1. En azul se muestran los puntos detectados durante el recorrido, y en rojo las posiciones del robot calculadas en cada iteración.

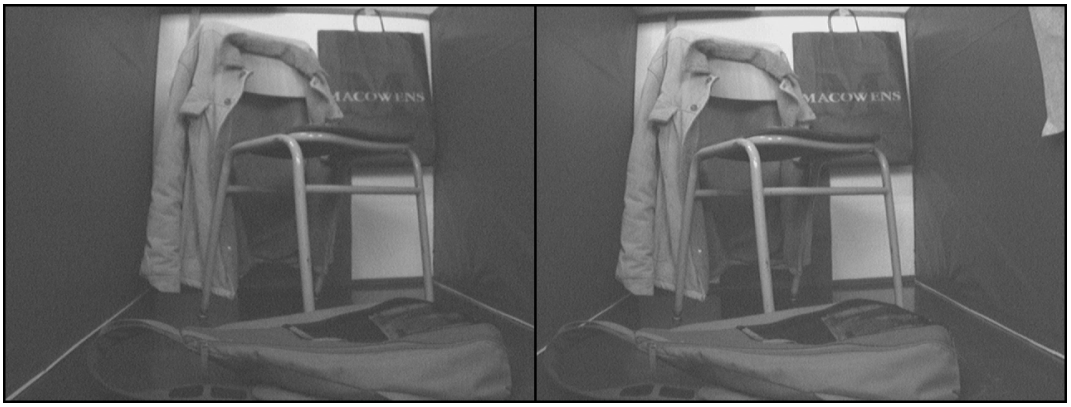

Fig. 6.13: Imágenes correspondientes a la cuarta iteración del caso 2. A partir de ellas se puede comprender el ambiente que rodea al robot.

Paso	Movimiento
0	∘
1	→
2	→
3	→
4	→
5	→
6	→
7	→
8	←
9	←
10	←
11	←
12	←
13	←

Tab. 6.2: Secuencia de pasos efectuados por el robot durante el recorrido del caso 2, donde ∘ indica el punto de partida, ← un giro de 10^o hacia la izquierda y → un giro de 10^o hacia la derecha.

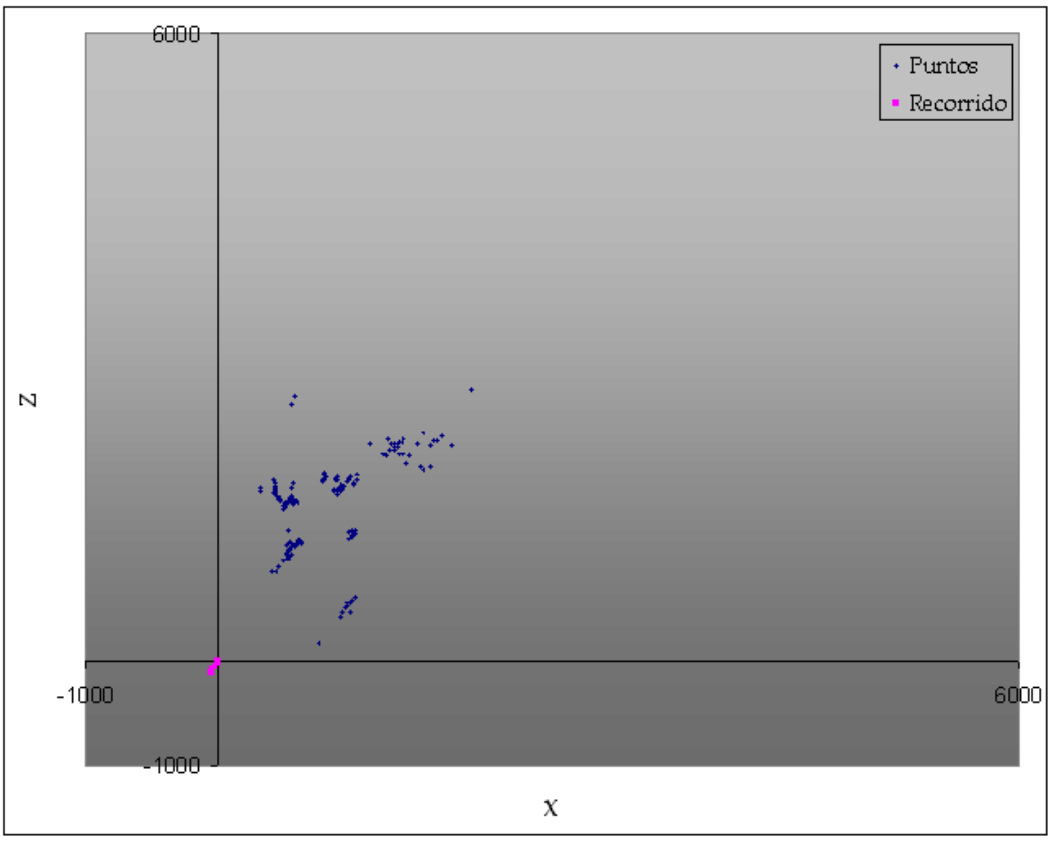

Fig. 6.14: Mapa en dos dimensiones correspondiente al caso 2. En azul se muestran los puntos detectados durante el recorrido, y en rojo las posiciones del robot calculadas en cada iteración.

Fig. 6.15: Imágenes correspondientes a la primer iteración del caso 3. A partir de ellas se puede comprender el ambiente que rodea al robot.

Paso	Movimiento
0	∘
1	↑←
2	↑
3	→↑
4	↑
5	→→
6	→

Tab. 6.3: Secuencia de pasos efectuados por el robot durante el recorrido del caso 3, donde ∘ indica el punto de partida, ↑ un movimiento de 10 cm hacia adelante, ← un giro de 10^o hacia la izquierda y → un giro de 10^o hacia la derecha. En los pasos donde se registra más de un movimiento, éstos se efectuaron uno seguido de otro siguiendo el orden de izquierda a derecha.

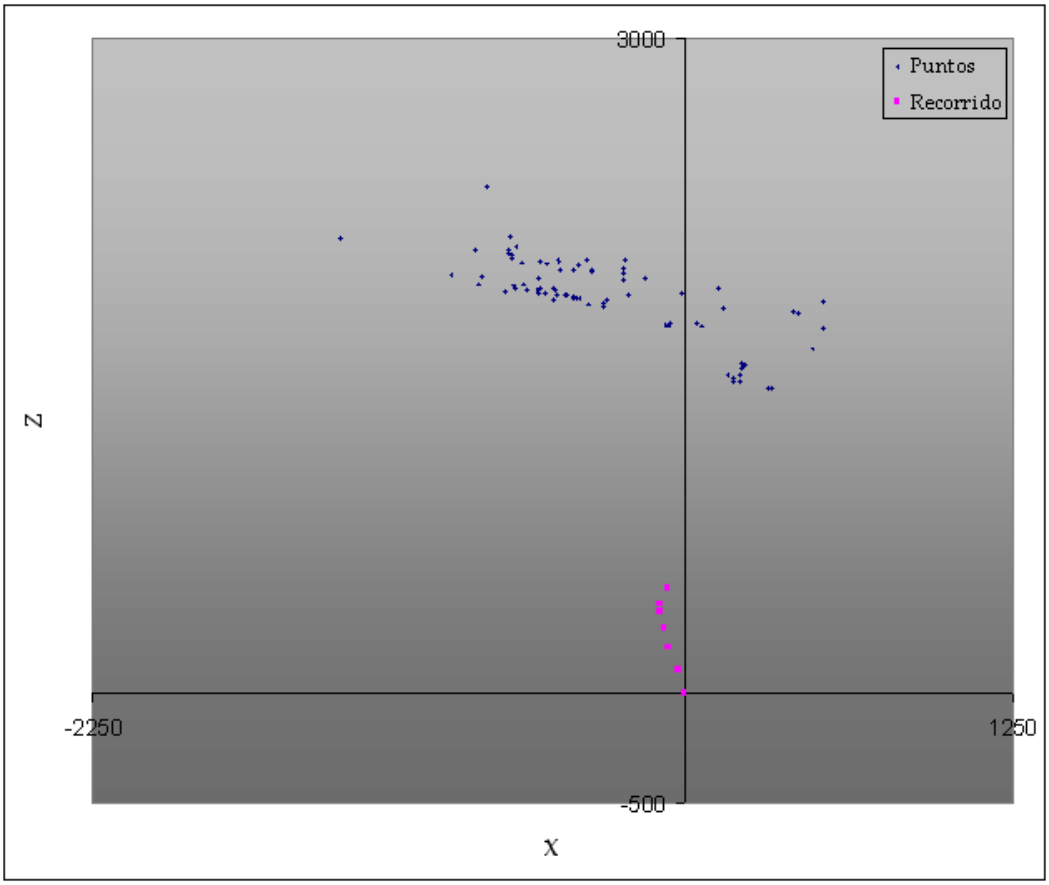

Fig. 6.16: Mapa en dos dimensiones correspondiente al caso 3. En azul se muestran los puntos detectados durante el recorrido, y en rojo las posiciones del robot calculadas en cada iteración.

Fig. 6.17: Imágenes correspondientes a la primer iteración del caso 4. A partir de ellas se puede comprender parte del ambiente que inicialmente rodea al robot.

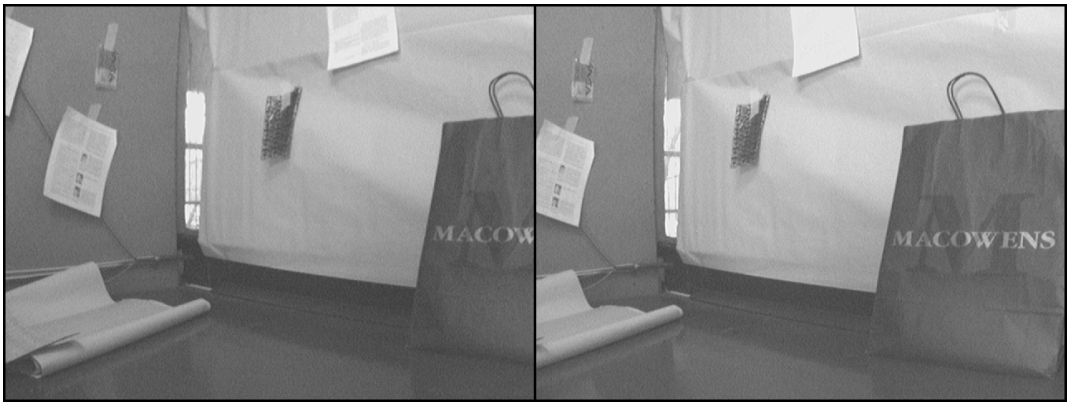

Fig. 6.18: Imágenes correspondientes a la iteración número 23 del caso 4. A partir de ellas se puede comprender parte del ambiente que inicialmente no era visible. El ambiente completo tiene forma de L invertida.

Paso	Movimiento	Paso	Movimiento
0	∘	**25**	←
1	↑	**26**	↑
2	↑	**27**	←
3	↑	**28**	←
4	↑	**29**	↓
5	↑	**30**	←
6	↑	**31**	↓
7	↑	**32**	←
8	↑	**33**	←
9	↑	**34**	←
10	↑	**35**	←
11	↑	**36**	←
12	↑	**37**	←
13	↑	**38**	←
14	↑	**39**	←
15	↑	**40**	←
16	↑	**41**	↓
17	↑	**42**	←
18	↑	**43**	↑
19	↑	**44**	↑
20	←	**45**	→
21	←	**46**	↑
22	←	**47**	↑
23	←	**48**	↑
24	↑	**49**	↑

Tab. 6.4: Pasos efectuados por el robot durante el recorrido del caso 4, donde el simbolo ∘ indica punto de partida, ↑ un movimiento de 10 cm hacia adelante, ↓ un movimiento de 10 cm en retroceso, ← un giro de 10° hacia la izquierda y → un giro de 10° hacia la derecha.

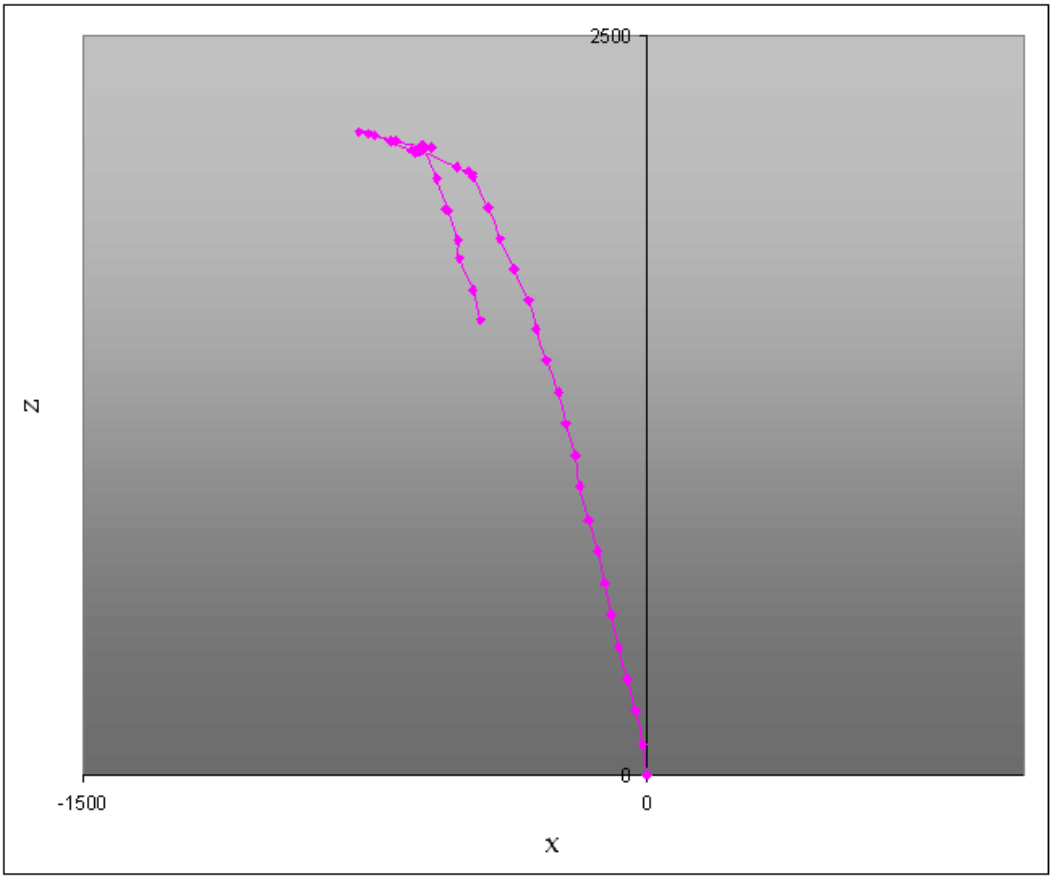

Fig. 6.19: Recorrido generado uniendo las posiciones del robot calculadas en cada iteración correspondientes al caso 4, este recorrido se corresponde con la secuencia de pasos que se muestra en la tabla 6.4.

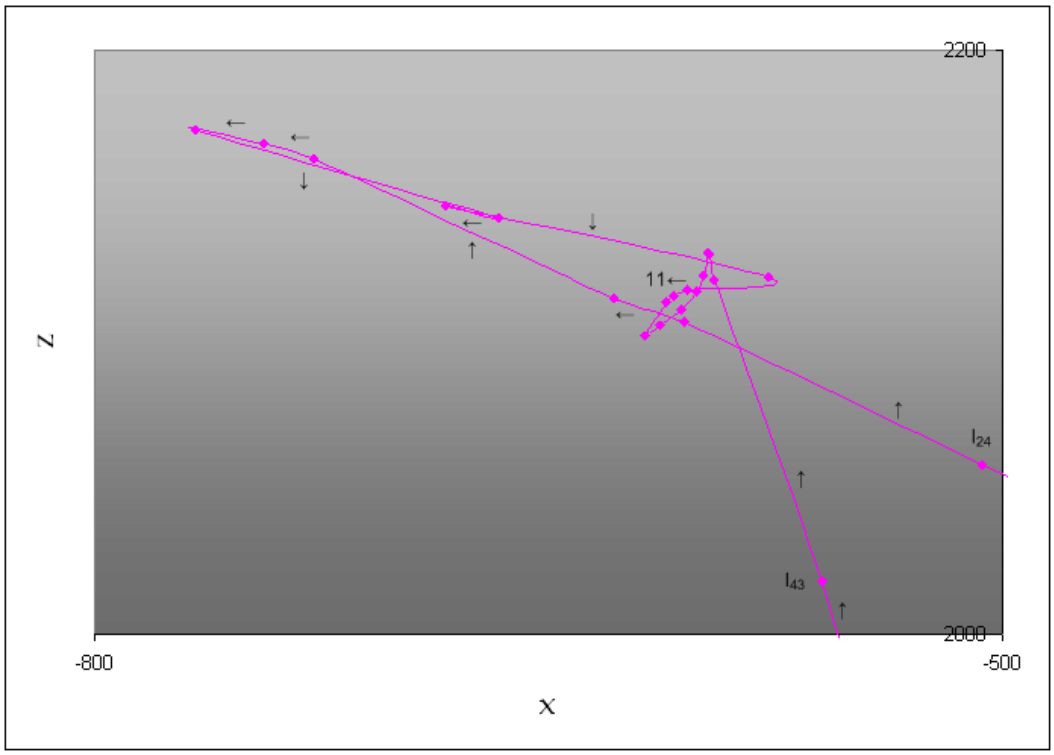

Fig. 6.20: Acercamiento del recorrido correspondiente al caso 4, el cual fue generado uniendo las posiciones del robot calculadas entre las iteraciones 24 y 43 mostradas en la tabla 6.4.

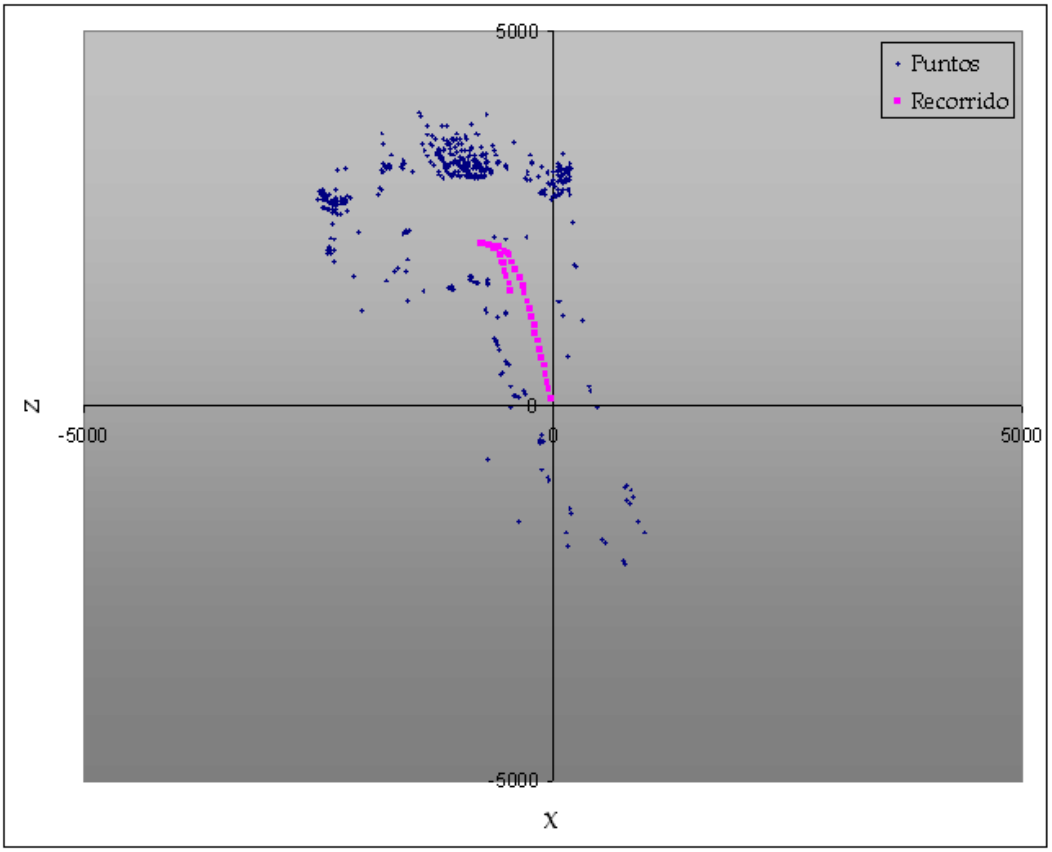

Fig. 6.21: Mapa en dos dimensiones correspondiente al caso 4. En azul se muestran los puntos detectados durante el recorrido, y en rojo las posiciones del robot calculadas en cada iteración.

7. CONCLUSIONES

En este trabajo hemos presentado un método para generar mapas de ambientes desconocidos mientras al mismo tiempo se actualiza la posición absoluta del robot en dicho ambiente. Estos mapas están formados por puntos en el espacio calculados utilizando conceptos de geometría epipolar en base a información obtenida de pares de imágenes capturadas por un sistema de visión estéreo.

Adicionalmente se presentó un algoritmo para detección de correspondencias outliers llamado "Eliminación por pendiente y longitud" (EPL). Este algoritmo se presenta como una alternativa a RANSAC. Aunque los dos algoritmos presentan resultados similares, se notan dos ventajas intrínsecas de EPL sobre RANSAC:

1 El algotitmo EPL es un caso particular del mecanismo genérico presentado en 3.2, utilizado para realizar filtrado de correspondencias. Este mecanismo genérico se basa en una idea simple e intuitiva, y por este motivo, el método puede ser parametrizado para filtrar distintos tipos de elementos en forma simple, sólo basta con definir las características y un parámetro de configuración para cada una de ellas.

2 Como se mostró en el capítulo 6, el rango de valores óptimos para los parámetros de EPL es amplio, y a causa de esto, es posible configurar el algoritmo en forma sencilla y rápida, inclusive sin tener amplio conocimiento del algoritmo en sí.

7.1. Contribuciones

Los aportes de este trabajo consiten en los siguientes puntos:

1 Se desarrolló un mecanismo para realizar filtrado de elementos basado en la idea de características comunes. Dicho mecanismo presenta la ventaja de ser fácilmente parametrizable, con el fin de poder resolver distintos problemas de filtrado.

2 Se parametrizó el mecanismo genérico para realizar filtrado de correspondencias outliers, utilizando ls características: pendiente y longitud.

3 El cálculo del movimiento se efectuó aplicando cuadrados mínimos en forma independiente para calcular el cambio de orientación y de posición en el plano.

4 Se robusteció el algoritmo de SLAM teniendo como objetivo principal minimizar el efecto negativo que producen las principales fuentes de error.

7.2. Problemas abiertos

Las superficies por las que se desplaza el robot suelen tener desniveles y cambios en la elevación que no son capturadas por el algoritmo de SLAM desarrollado. Es importante poder determinar cambios en la inclinación del robot dado que esto influye en los mapas generados.

En casos en los cuales la información obtenida para una iteración no es suficiente como para determinar el movimiento realizado por el robot, nos encontramos con un problema no trivial. Para solucionar este tipo de inconvenientes es necesario generar un sistema de recuperación. Una solución puede generarse basándose en la idea de mapas parciales, que pueden ser combinados en un proceso posterior.

Hay distintas situaciones en las cuales el robot se mueve en un ambiente en el cual hay objetos en movimiento. Esto presenta una gran dificultad en el caso de un algoritmo de SLAM basado en visión estéreo, ya que el algoritmo se basa en la idea de un ambiente estático. En muchos casos es necesario identificar elementos en movimiento con el objeto de ignorar los puntos claves pertenecientes a ellos.

APÉNDICE

A. DESCOMPOSICIÓN EN VALORES SINGULARES

La descomposición en valores singulares (*Singular Value Decomposition*, SVD) es ampliamente usada en computación numérica, especialmente para resolver sistemas de ecuaciones sobredeterminados.

Dada una matriz cuadrada A, la SVD, es una factorización $A = UDV^T$ donde U y V son matrices ortogonales y D es una matriz diagonal con todos sus valores positivos. Como convención se escribe V^T en lugar de V y D tiene sus valores ordenados en forma descendente (es posible encontrar la SVD que cumpla esta restricción).

La SVD también existe para matrices no cuadradas de $m \times n$ y resulta particularmente útil cuando $m \leq n$. En este caso, A es factorizada como UDV^T donde U es una matriz de $m \times n$ con columnas ortogonales [1], D es una matriz diagonal de $n \times n$ y V^T es una matriz ortogonal de $n \times n$.

Valores singulares y autovalores Los valores en la diagonal de D son los valores singulares de A. Para ver la conexión entre autovalores y autovectores, examinemos las siguientes ecuaciones

$$\begin{aligned} A &= UDV^T \\ A^TA &= UDV^TUDV^T = VDU^TUDV^T = VD^2V^T \\ A^TA &= VD^2V^{-1} \quad \text{ya que } V \text{ es ortogonal} \end{aligned}$$

Esta es la ecuación que define los autovalores de A^TA, ubicados en la diagonal de D^2. Los autovectores son las columnas de V. En definitiva, los valores singulares de A son las raíces cuadradas de los autovalores de A^TA. (Notar que A^TA es simétrica semidefinida positiva, por lo que sus autovalores son reales positivos, los valores singulares son siempre reales y positivos.)

[1] una matriz ortogonal M es tal que $M^TM = I$. También se cumple como corolario que $\|Mx\| = \|x\|$ para cualquier vector x

B. CALIBRACIÓN

En esta sección se muestra el procedimiento realizado para obtener los parámetros intrínsecos y extrínsecos de las cámaras utilizadas.

Inicialmente se cuenta con un tablero similar a uno de ajedrez como se muestra en la figura B.1. Este tablero tiene un total de 12 x 12 casilleros donde todos los casilleros tienen la misma dimensión (3cm x 3cm).

Con las cámaras se capturan pares de imágenes cambiando la inclinación del tablero, en las figuras B.2 y B.3 se muestran las imágenes capturadas por las cámaras.

Para cada una de las imágenes capturadas por cada cámara se deben detectar los bordes determinados por los casilleros del tablero. Este proceso puede hacerse manualmente o automáticamente utilizando un algoritmo de detección de esquinas. En la figura B.4 se muestra un tablero en el cual se encuentran marcados los bordes formados por casilleros.

A partir de los bordes detectados para las imágenes tomadas por una cámara es posible calcular los parámetros intrínsecos de dicha cámara.

Los parámetros intrínsecos calculados para cada cámara son:

- Distancia Focal: distancia focal medida en pixels.
- Punto Principal: coordenadas del punto principal.
- Coeficiente oblicuo: ángulo definido entre los ejes X e Y.
- Distorción: distorición radial y tangencial en las imágenes.

Habiendo calculado los bordes para las imágenes tomadas por ambas cámaras y teniendo los parámetros intrínsecos de cada una, entonces es posible calcular los parámetros extríncecos respecto de las dos cámaras.

Los parámetros extrínsecos calculados para las cámaras son:

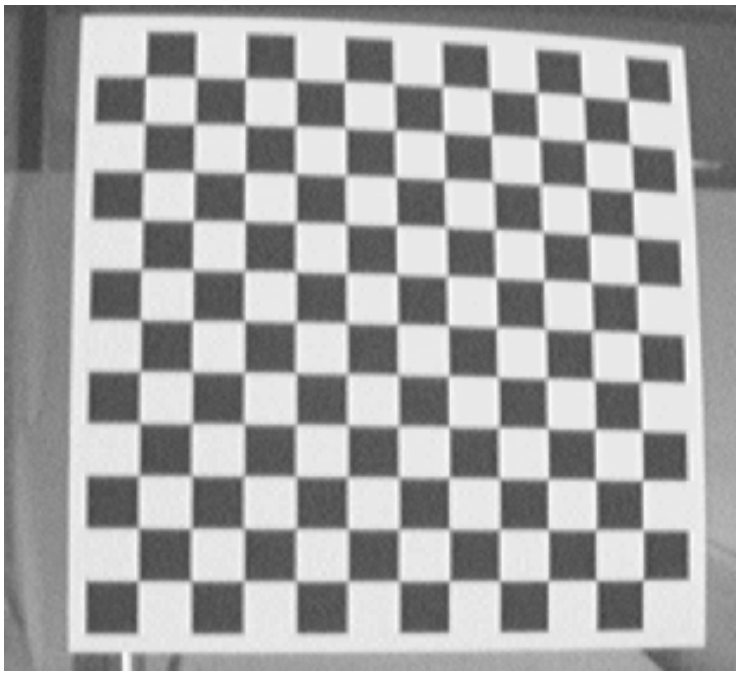

Fig. B.1: Tablero de 12 x 12 casilleros.

- Rotación: matriz de 3x3 que indica la rotación de una cámara con respecto a la otra.

- Traslación: vector de 3x1 que indica la translación de una cámara con respecto a la otra.

Existen distintas herramientas capaces de automatizar el mecanismo descrito en esta sección. Para el presente trabajo se utilizó una herramienta desarrollada por Klaus Strobl, Wolfgang Sepp, Stefan Fuchs, Cristian Paredes y Klaus Arbter en [4].

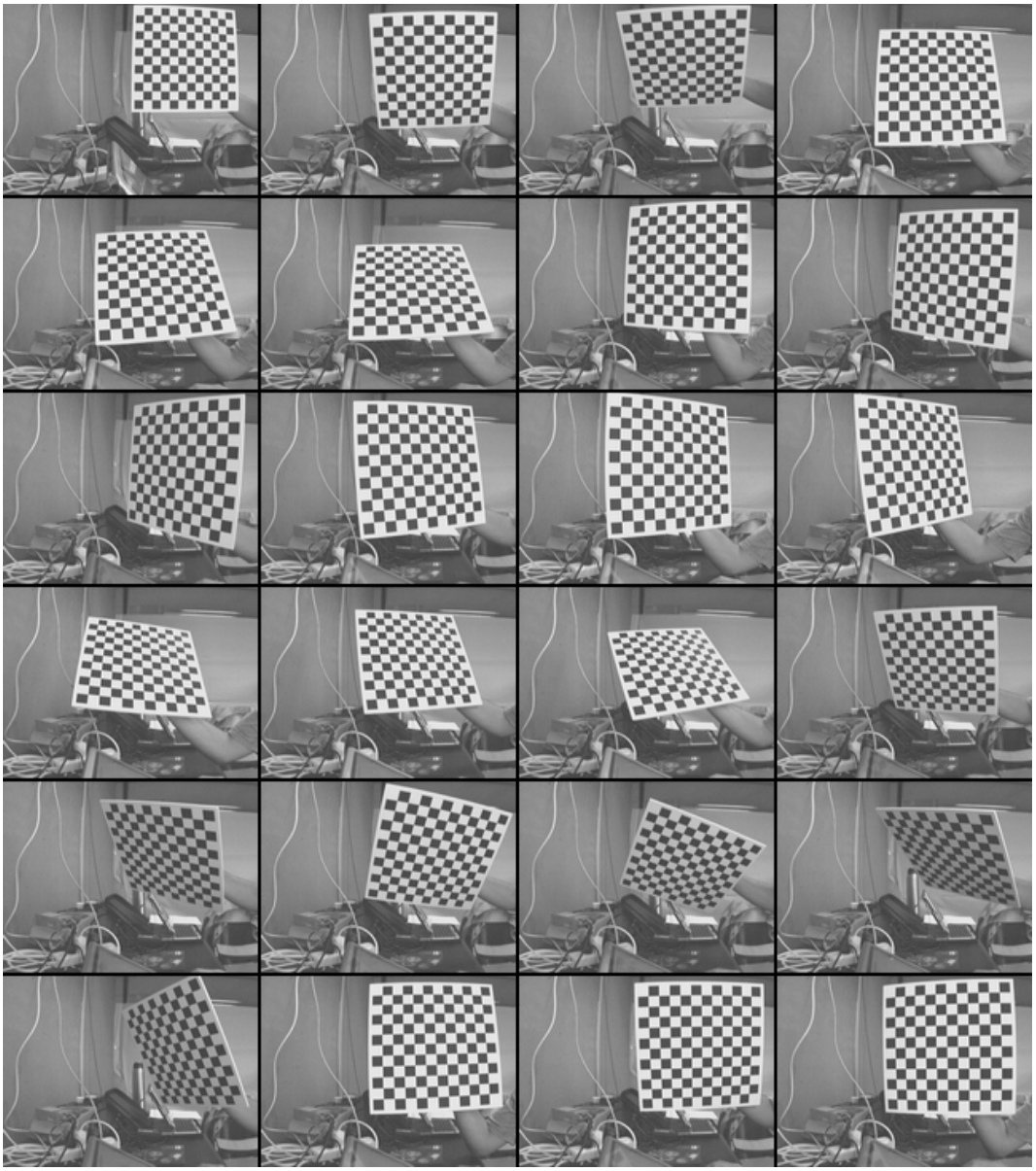

Fig. B.2: Imágenes del tablero capturadas por la cámara izquierda.

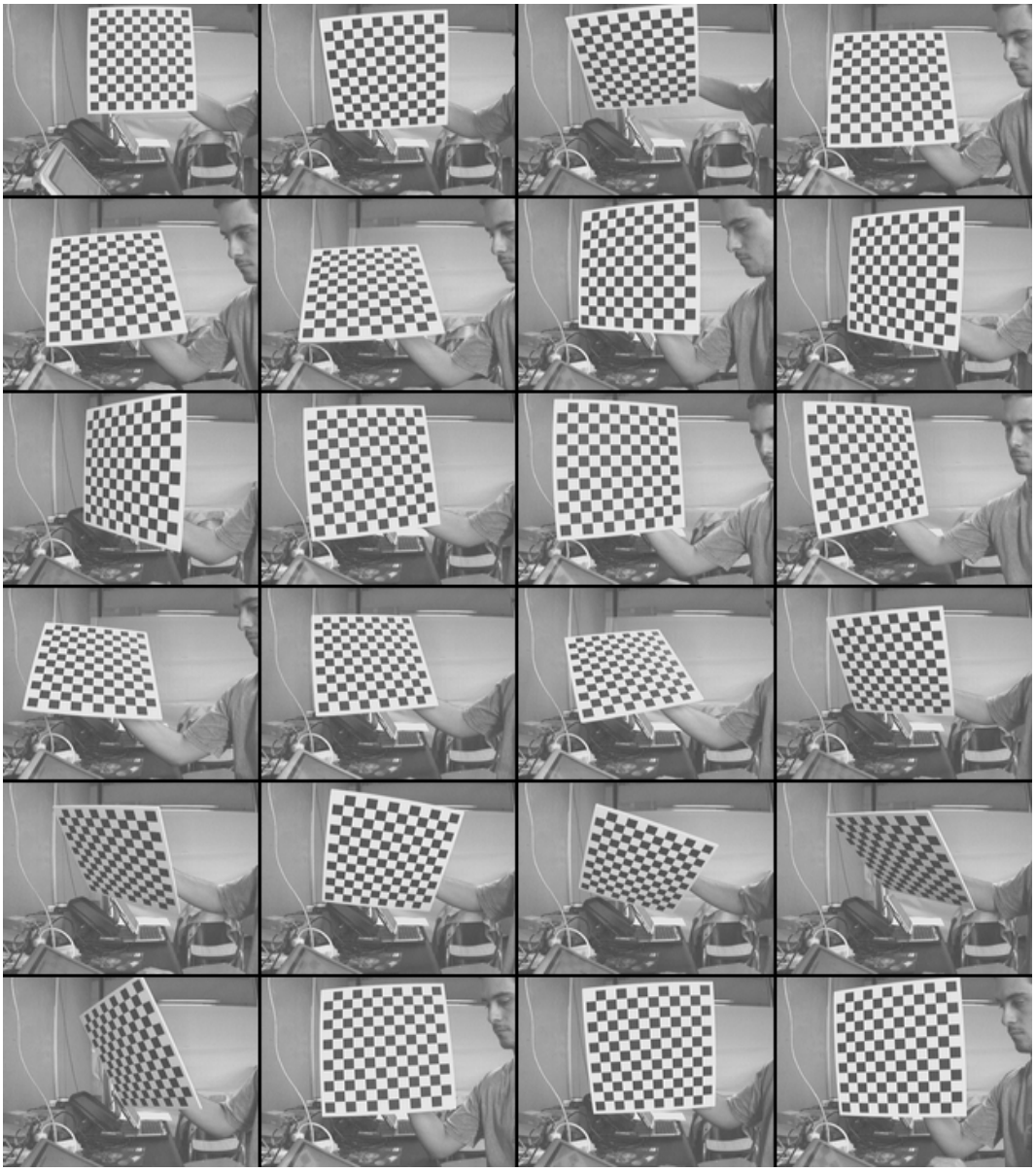

Fig. B.3: Imágenes del tablero capturadas por la cámara derecha.

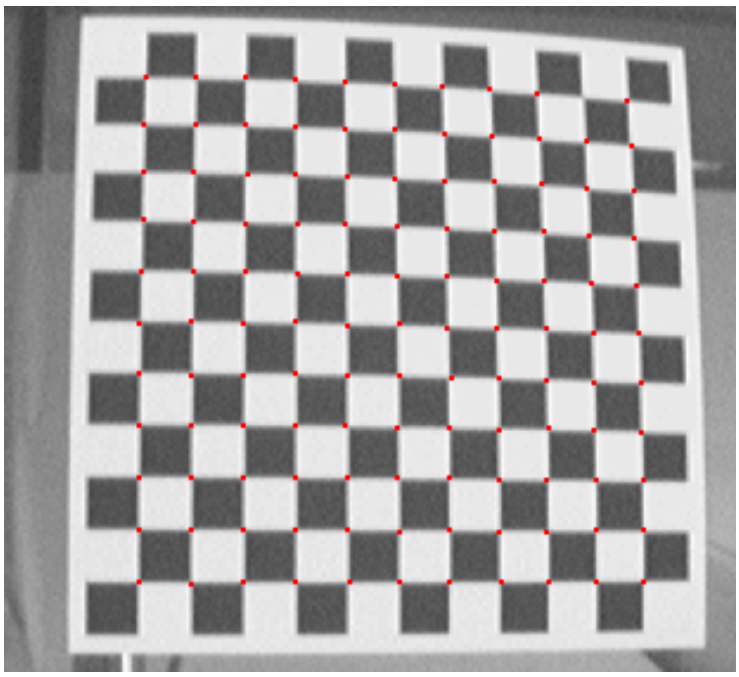

Fig. B.4: Tablero de 12 x 12 casilleros, en rojo se muestran los bordes formados por casilleros.

BIBLIOGRAFíA

[1] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[2] Jerome H. Freidman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977.

[3] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[4] Stefan Fuchs Cristian Paredes Klaus Strobl, Wolfgang Sepp and Klaus Arbter. Camera calibration toolbox for matlab.

[5] David G. Lowe. Object recognition from local scale-invariant features. *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 1150, 1999.

[6] David G. Lowe. Local feature view clustering for 3d object recognition. *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01)*, I:682–688, 2001.

[7] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[8] J.M. Saez and F. Escolano. Entropy minimization slam using stereo vision. 2005.

[9] James J. Little Stephen Se, David G. Lowe. Vision-based mobile robot localization and mapping using scale-invariant features.

[10] James J. Little Stephen Se, David G. Lowe. Mobile robot localization and mapping with uncertainty using scale-invariant landmarks. 21:735–758, 2002.

[11] James J. Little Stephen Se, David G. Lowe. Vision-based mapping with backward correction. pages 153–158, October 2002.

[12] James J. Little Stephen Se, David G. Lowe. Vision-based global localization and mapping for mobile robots. 21(3), 2005.

[13] Zhengyou Zhang and Gang Xu. A general expression of the fundamental matrix for both perspective and affine cameras. pages 1502–1510, 1997.