Construcción de mapas 3D y extracción de primitivas geométricas del entorno

Diego Viejo
Miguel Cazorla
Dpto. Ciencia de la Computación e I.A.
Universidad de Alicante
Apdo 99 03080 Alicante
{dviejo,miguel}@dccia.ua.es

Resumen

Este trabajo se centra en la construcción de mapas 3D a partir de datos de rango obtenidos mediante una cámara estéreo. El proceso de reconstrucción lo hemos dividido en dos fases. En la primera, a partir de datos tomados a intervalos regulares por un robot dentro de un entorno, hemos solucionado parcialmente el problema del error de odometría, haciendo uso de un método novedoso de emparejamiento 3D. En la segunda y haciendo uso del resultado de la fase anterior, nos planteamos extraer primitivas geométricas (planos, cilindros, etc.) para realizar una reconstrucción del entorno mediante estas primitivas.

1. Introducción

En este trabajo pretendemos realizar la reconstrucción de un mapa 3D a partir de un conjunto de puntos 3D tomados a intervalos regulares dentro de un entorno. Los datos son tomados desde una cámara estéreo comercial, que nos proporciona tanto las coordenadas de los puntos como su valor de intensidad.

Varios enfoques se han propuesto para la reconstrucción 3D de un entorno, haciendo uso de un robot móvil. En [13] se plantea un método que reconstruye de manera eficiente el interior de una mina. En este trabajo solucionan dos problemas. El primero consiste en la resolución del problema del SLAM ([12]). Este problema consiste en construir un mapa del entorno y a la vez localizarnos dentro de él. El problema surge debido a los errores de odometría. Este problema lo solventan haciendo uso de puntos característicos en el entorno y aplicando un método basado en el algoritmo EM. El segundo problema es el de la reconstrucción. Disponen de dos sonares láser (Sick) uno dirigido en el mismo sentido de avance del robot y otro hacia el techo. Van tomando datos a mucha frecuencia y, una vez rectificados, reconstruyen utilizando los puntos devueltos por el láser. Parecido enfoque se utiliza en [11]. Aquí para reducir el error entre un par de tomas de datos, aplican un algoritmo muy parecido al ICP (comentado más adelante). Como sensor utilizan también dos láseres.

El principal problema que nos encontramos con cámaras estéreo es que en escenarios con poca textura no son capaces de extraer puntos. Esto implica una carencia de datos que nos obliga a tratar el problema de manera distinta que en otros planteamientos. Nuestro planteamiento parte de la idea de plantilla deformable [14], donde vamos a buscar las primitivas geométricas (planos, cilindros, etc.) dentro de la nube de puntos del entorno. Debido a que

trabajamos dentro de entornos estructurados parece razonable el intentar extraer las primitivas que mayor tamaño tengan. En este caso se trata de planos debido a las paredes. Una vez identificados los puntos que se corresponden con las paredes, se eliminan y entonces pasaremos a identificar otro tipo de primitivas. Este trabajo es un punto de partida, no pretende realizar toda la reconstrucción. Por ello nos hemos centrado en la extracción de planos.

Nuestro planteamiento se puede resumir en los siguientes pasos:

1. Eliminación o reducción del error de odometría. Para ellos se va a utilizar un método novedoso de extracción de pose.

2. Extracción de las normales de los puntos obtenidos.

3. Extracción de las primitivas geométricas. Empezaremos por los planos, encontrando planos y eliminando los puntos del entorno que los soporten. Continuaremos con otros tipos de primitivas.

El resto del artículo queda: en la Sección 2 describimos el experimento de la toma de datos y el equipo utilizado; el algoritmo utilizado para el emparejamiento de los conjuntos de puntos se detalla en la Sección 3; la Sección 4 introduce una aproximación para la extracción de primitivas geométricas y la Sección 5 presenta las conclusiones y trabajos futuros.

2. Descripción del experimento

El experimento se realizó en el edificio de la facultad de Ciencias Económicas y Empresariales de la Universidad de Alicante. En la Figura 2 se puede observar una planta del edificio y el recorrido aproximado que realizó nuestro robot. El robot (Frodo, ver Figura 1) es un modelo Magellan Pro de RWI con una cámara triestéreo Digiclops (para más detalles sobre la cámara ver [10]).

Figura 1: Robot utilizado en los experimentos con la cámara triestéreo utilizada.

La cámara nos proporciona un conjunto de puntos 3D y el valor de intensidad del punto. A este conjunto de puntos le llamaremos $\mathbf{S} = \{S_1, S_2, \ldots, S_n\}$ donde $S_i = \{s_1^i, s_2^i, \ldots, s_k^i\}, s_j^i = (X_j, Y_j, Z_j, G_j)$, donde X, Y, Z son las coordenadas 3D del punto (ver Figura 3 derecha) y G es su valor de gris. En la Figura 3 podemos observar un conjunto de puntos extraído en una posición del robot. Las coordenadas de los puntos son siempre referentes al sistema de referencia de la cámara. Nuestro robot realiza el recorrido indicado en la Figura 2 y va tomando imágenes a intervalos regulares. Disponemos de los datos de odometría facilitados

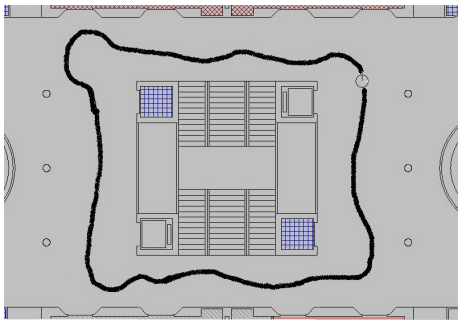

Figura 2: Planta del edificio donde se realizó el experimento y recorrido efectuado.

por los *encoders* del robot, que nos proporcionan una aproximación de las acciones realizadas por el robot $\mathbf{A} = \{a_1, a_2, \ldots, a_n\}$, donde cada acción $a_i = (x_i, y_i, \theta_i)$. Podemos realizar una reconstrucción del entorno (la creación de un mapa) haciendo uso de esta información de odometría. Para ello y dado que las coordenadas de los puntos son locales a cada posición del robot, debemos aplicar una transformación a las coordenadas de los puntos haciendo uso de la información de odometría. Empezando por la posición $(0, 0, 0)$, para calcular las coordenadas de un punto p_j obtenido después de la acción i debemos aplicar, desde 1 hasta i las transformaciones T_i correspondientes a las acciones a_i realizadas en cada paso. Para cada conjunto de puntos S_i, el nuevo conjunto de puntos S_i' se calculará con $S_i' = T(a_i, S_i)$ siendo $T(a, S)$ la transformación realizada sobre S usando los datos de a.

Figura 3: Izda: Visualización 3D de un conjunto de puntos. Dcha: Sentido de los ejes. El punto $(0, 0, 0)$ es el centro de la cámara.

La Figura 4 muestra la vista cenital de este entorno. Comentar que se han eliminado los puntos del suelo y el techo para poder observar mejor la reconstrucción. Podemos observar que existen ciertos problemas en la reconstrucción derivados de los errores de odometría. El tratamiento de estos errores se realizará en la siguiente sección.

3. Rectificación mediante emparejamientos de puntos

Como hemos visto, los errores de odometría hacen que el entorno no pueda ser reconstruido de forma exacta. Además, se producen errores entre dos posiciones consecutivas, que hacen

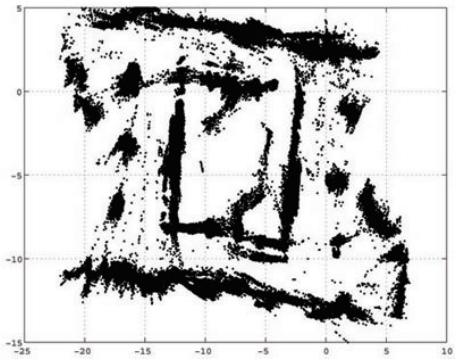

Figura 4: Vista cenital del entorno reconstruido usando odometría.

que nuestro objetivo de extraer primitivas geométricas sea complicado y fuente de posibles problemas. Uno de los algoritmos clásicos de cálculo de la transformación 3D entre dos conjuntos de puntos es el ICP (*Iterative Closest Point*) [3]. ICP calcula, de manera iterativa, la mejor transformación entre dos conjuntos de puntos en dos pasos. En el primero, basándose en una transformación inicial, calcula las correspondencias de los puntos, asignando a un punto de un conjunto el más cercano del otro, después de aplicar la transformación. En el siguiente paso, haciendo uso de las correspondencias calculadas en el paso anterior, se calcula mediante mínimos cuadrados la mejor transformación que minimiza el error de distancia de las correspondencias. Dicha transformación es utilizada como nueva transformación en el primer paso. ICP asegura que converge a un mínimo local. El problema principal de este algoritmo es la presencia de *outliers*. Si la diferencia de puntos entre los dos conjuntos es de más de un 10 ó 20 %, el algoritmo convergerá con mucha probabilidad a un mínimo local.

En nuestro problema, la presencia de *outliers* es masiva, tal como se puede observar en la Figura 5. En esta figura se muestran dos conjuntos de puntos, después de aplicar la transformación correspondiente a la información de odometría. Los puntos pertenecientes al suelo y el techo pueden provocar grandes errores en los cálculos.

Figura 5: Dos conjuntos de puntos tras la transformación de la odometría. Se puede observar, debido al campo de visión de la cámara, la presencia de gran cantidad de *outliers*.

Para intentar eliminar el mayor número de *outliers*, procedemos a eliminar los puntos del suelo y el techo. También podemos hacer uso de las restricciones geométricas de la cámara. La cámara estéreo utilizada sólo puede obtener puntos en unas ciertas posiciones, es decir, no obtiene puntos de fuera de su campo de visión. En este paso, calculamos la intersección

de los campos de visión de la cámara en dos posiciones consecutivas y eliminamos los puntos de fuera de esta intersección. En la Figura 6 (izquierda) vemos dos conjuntos de puntos y la restricción impuesta por el campo de visión de la cámara. En la parte derecha de la misma figura observamos la eliminación de los puntos del suelo y el techo y cómo podemos eliminar gran cantidad de *outliers* al aplicar dicha restricción. Sin embargo, no se eliminan todos ellos.

Figura 6: Restricción del campo de visión de la cámara y eliminación de puntos del suelo y techo. Los puntos fuera del campo de visión de las cámaras (líneas grises) no se tienen en cuenta para el cálculo.

3.1. Emparejamiento de puntos

Los algoritmos de emparejamiento encuentran la mejor transformación que empareja dos conjuntos de puntos. En nuestro problema disponemos de dos conjuntos de puntos 3D, pero debido al movimiento del robot y a que la cámara no tiene un mecanismo de giro, podemos simplificar los cálculos para restringirlos a dos dimensiones. De esta manera podemos reducir el tiempo de cálculo debido a que manejaremos puntos con dos coordenadas y las matrices de transformación se reducen a 2×2. Para este paso eliminaremos la información de altura de los puntos (la coordenada Y). Nuestro objetivo es encontrar la mejor transformación $(R^*, T^*$, rotación, traslación) que minimice la siguiente función de error sobre dos conjuntos de puntos consecutivos en la secuencia, $S_i = \{s_1^i, s_2^i, \ldots, s_k^i\}$ y $S_{i+1} = \{s_1^{i+1}, s_2^{i+1}, \ldots, s_l^{i+1}\}$:

$$(R^*, T^*) = \arg\min_{R,T} \sum_{t=1}^{N} (s_t^{i+1} - (Rs_t^i + T))^2 \tag{1}$$

El subíndice t indica correspondencia entre los puntos. Para simplificar el problema se suele descomponer en dos problemas separados: calcular la rotación primero y luego la traslación. Para calcular la rotación primero se calcula el centroide de cada conjunto de puntos. A continuación se le resta a cada uno de los puntos, es decir, los dos conjuntos se centran en el origen. Otra vez, la presencia de *outliers* hace que al centrar los dos conjuntos, queden muy separados. Por ello y debido a que disponemos de la información de odometría aplicamos la transformación con los datos de odometría (ver Figura 7 para resultados). El resultado de aplicar esta transformación hace que el error no sea muy alto debido a que no acumulamos la odometría, sino que guardamos la acción para cada par de posiciones. Así calculamos un único centroide para los dos conjuntos. Entonces, al restar el centroide a los dos conjuntos

de puntos, la Ecuación 1 queda:

$$R^* = \arg\min_R \sum_{t=1}^{N} (s_t^{i+1} - Rs_t^i)^2 \tag{2}$$

Así nos limitamos a calcular únicamente la rotación R. La obtención de esta matriz se detalla en la siguiente sección. Una vez calculada la matriz R, volvemos a los conjuntos de puntos originales y aplicamos la nueva rotación. Ahora sólo nos queda calcular la traslación. Dado que el error de traslación es muy reducido, hemos procedido a evaluar la función de error alrededor de la posición que marca la odometría. Nos quedamos con la traslación que nos hace mínima la función de error.

3.2. El algoritmo ICP-EM

Para calcular la matriz de rotación podemos utilizar el algoritmo ICP. Sin embargo, el algoritmo ICP-EM [4] proporciona menor dependencia de *outliers* y tiempo de cálculo. Este algoritmo se basa en el ICP para realizar los mismos pasos de cálculo de las correspondencias y la transformación. Sin embargo, en vez de determinar las correspondencias de forma única, utiliza variables aleatorias para calcular la esperanza de que un punto se empareje con cualquiera del otro conjunto. La función de error queda:

$$R^* = \arg\min_R \sum_{t=1}^{k} \sum_{u=1}^{l} (s_t^{i+1} - Rs_u^i)^2 z_{tu} \tag{3}$$

La variable aleatoria z_{tu} tiene valor 1 si el punto s_t^{i+1} se empareja con el s_u^i, y 0 en caso contrario. Sin embargo, no conocemos el valor de esta variable (no conocemos las correspondencias). Si las conociéramos podríamos calcular la transformación mediante mínimos cuadrados. Si conociéramos la transformación podríamos calcular las correspondencias simplemente calculando el punto más cercano. Este problema es conocido como el del "huevo y la gallina" [9]. El algoritmo EM soluciona este problema. De manera iterativa y de forma muy similar al ICP encuentra la transformación y las correspondencias. En [4] se detalla la derivación numérica del algoritmo.

3.3. Resultados

En la Figura 7 se muestra el resultado obtenido al aplicar el algoritmo ICP-EM al recorrido. Si la comparamos con la Figura 4 se puede observar que el mapa ha sido rectificado y se produce una mejora sustancial tanto en la posición como en la rotación. En la Figura 8 se puede apreciar con más detalle el resultado.

En la Figura 9 se observa con detalle el error cuando el robot llega al punto de partida. Este error (residual del algoritmo de emparejamiento) será tratado en trabajos futuros, mediante algún algoritmo de emparejamiento global.

4. Extracción de primitivas geométricas

Una vez que hemos reducido el error en la odometría del robot mediante el método descrito en la sección anterior, podemos abordar el problema de la extracción de características para la construcción de un modelo tridimensional del entorno. Los datos que disponemos para dicha extracción son una nube de puntos 3D pertenecientes a los objetos que se encuentran

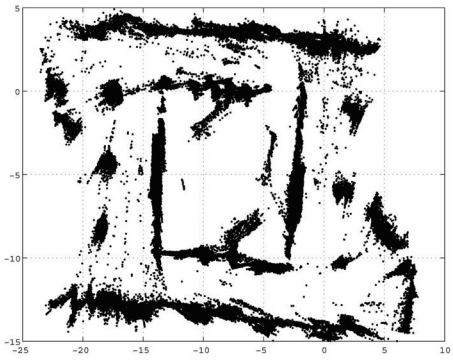

Figura 7: Vista cenital del entorno reconstruido con la rectificación.

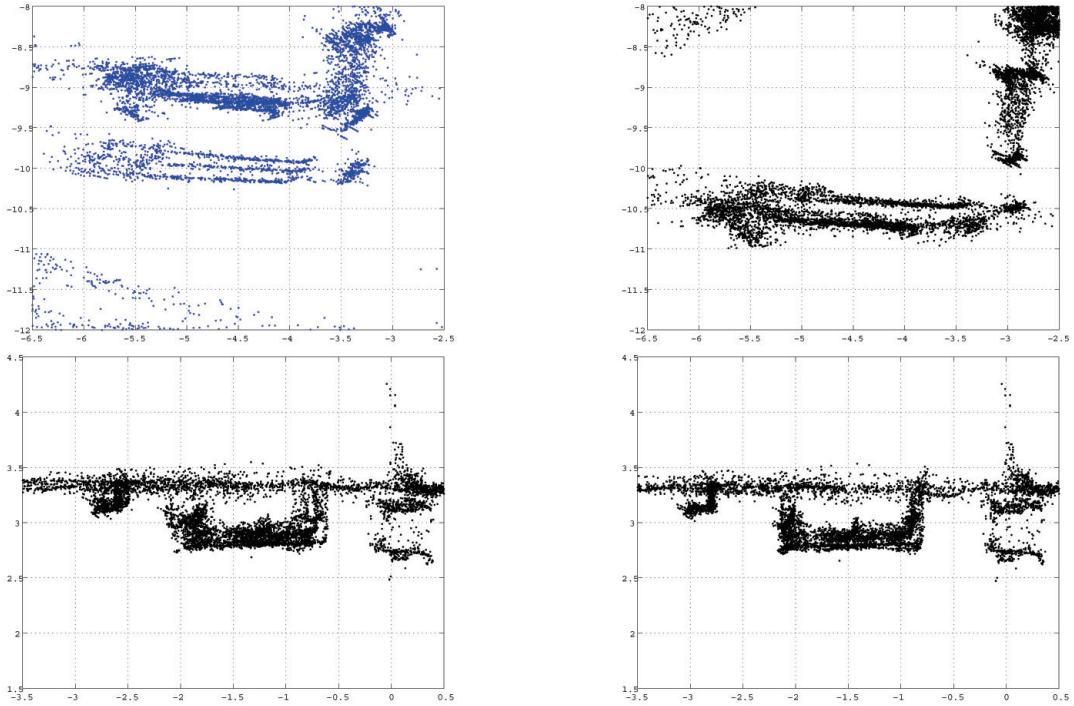

Figura 8: Detalles en la mejora de la reconstrucción. Izda: con odometría. Dcha: aplicando el ICP-EM.

en el entorno. Nuestro trabajo está encaminado a encontrar primitivas geométricas (planos, cilindros, etc.) que mejor se ajusten a estos datos.

La información de distancia entre puntos es insuficiente para extraer información de la geometría del objeto subyacente y mucho menos para diferenciar los puntos pertenecientes a distintos objetos cercanos entre sí. Es por este motivo que nos decantamos por estudiar las normales ([6], [8]) a las superficies de los objetos. Con la información de dichas normales podemos hacernos una idea de la geometría del objeto y tratar de encontrar la primitiva

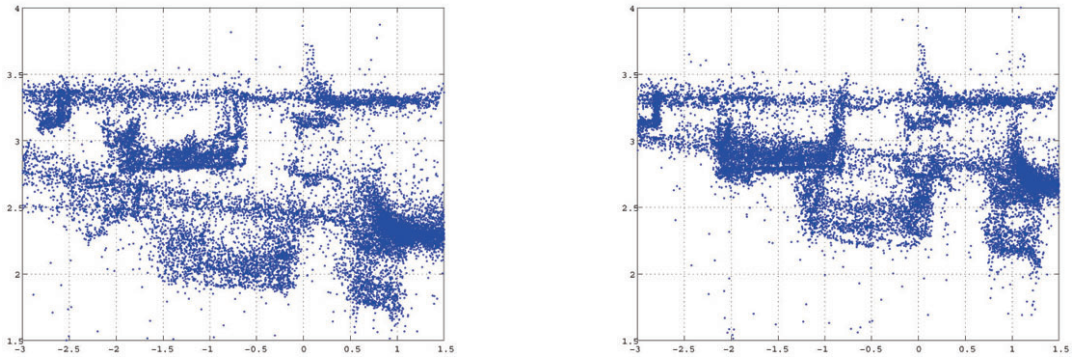

Figura 9: Resultado erróneo cuando llegamos al punto de partida.

geométrica que mejor se ajuste a dicha geometría.

4.1. El cálculo de las normales

Para extraer primitivas geométricas vamos a hacer uso de las normales de las superficies de los objetos para ajustar dichas primitivas. Para calcular normales a los puntos vamos a asumir una cierta continuidad en las superficies del entorno y, por lo tanto, asumimos que todos los puntos de una vecindad pertenecen a la misma superficie. Las discontinuidades, tales como las esquinas, nos van a permitir detectar los límites de las superficies. Realizaremos el cálculo de la normal a un punto s utilizando los puntos de su vecindad. Para mejorar la robustez de la estimación de la normal dividimos el espacio alrededor de s en g regiones R de un tamaño fijo. Obtenemos el centroide r_i de cada region R_i como el punto medio de todos los puntos que caen dentro de dicha región lo que nos permite reducir errores de la obtención de los datos. Además, no consideraremos para operaciones posteriores aquellas regiones que no tengan un número suficiente de puntos en su interior, con lo que eliminamos puntos de ruido. Una vez que tenemos los centroides $r_1, r_2, ..., r_g$ de las g regiones podemos obtener un conjunto de vectores de la forma $v_i = r_i - s$. Finalmente, la normal resultante se calcula realizando el producto vectorial de cada par de vectores v_i y tomando la media:

$$\frac{1}{n} \sum_{i=1}^{l-1} \sum_{j=i+1}^{l} v_i \times v_j \tag{4}$$

La calidad de la estimación depende del número de puntos que haya en cada región. Consideramos que la normal resultante no está definida cuando no hay evidencia suficiente para obtener una estimación robusta.

4.2. Extracción de superficies verticales planas

Si proyectamos las normales de todos los puntos de la escena en una semiesfera de radio unitario figura 10 (Izquierda) podemos apreciar que hay ciertas zonas de una densidad mayor. Esto es debido a que hay una mayoría de superficies del entorno con orientaciones similares (paredes). Aprovechamos esta característica para extraer planos verticales, con los que vamos a realizar una primera estimación del mapa 3D del entorno. En primer lugar, filtramos los puntos cuya normal no corresponda con una superficie vertical. Al resultado le aplicamos

el algoritmo *k-means* [5] para encontrar las agrupaciones de puntos correspondientes a cada una de las dos orientaciones de los muros del entorno figura 10 (Derecha).

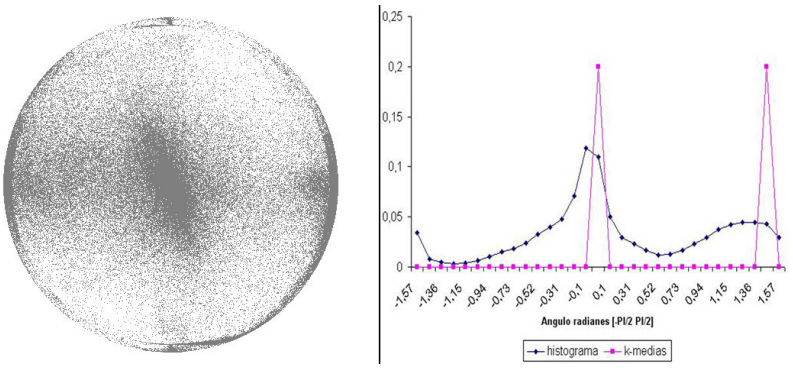

Figura 10: Izda: Visualización 3D de la proyección de la normal de los puntos. Dcha: Histograma de las orientaciones de los puntos y resultado de las k-medias.

Una vez que tenemos agrupaciones de puntos con una orientación similar procedemos a extraer planos. Consideramos que, dentro de cada agrupación, dos puntos pertenecerán al mismo plano si la distancia entre ellos es menor que un umbral λ. Al hacer esto, vamos a tener cada plano definido por un conjunto de puntos y sus normales $\Pi = (p_1, n_1), (p_2, n_2), ..., (p_n, n_1)$. Estimamos los parámetros del plano, la normal y un punto del plano, como la mediana de las normales y el centroide de todos los puntos, respectivamente. Finalmente, sólo nos queda obtener los límites de cada plano obtenido. Aprovechando que conocemos el ángulo que forma cada plano con el eje Z, rotamos el plano hasta hacerlo paralelo al plano XY. Así calculamos los límites del plano como los puntos con mayor y menor coordenadas X e Y. Después, basta con deshacer la rotación para obtener los límites reales del plano en el espacio.

4.3. Mezclando Planos

Lo que tenemos hasta este momento es un numeroso conjunto de planos (*patches*) de tamaño relativamente pequeño. Nuestro objetivo es encontrar los planos que se corresponden con las paredes del entorno. Para ello vamos a intentar fusionar aquellos *patches* que correspondan al mismo objeto del entorno. El método tiene que ser capaz de manejar los errores cometidos tanto en la obtención de los datos como en la extracción de superficies planas. Estos errores provocan que los *patches* extraídos de, por ejemplo, una pared del entorno no estén perfectamente alineados a lo largo de dicha pared, sino que presentan errores tanto en su posición como en su orientación. Uno de los principales problemas estará en discriminar cuándo dos *patches* han sido obtenidos a partir del mismo objeto del entorno.

Como criterio para la fusión de *patches* vamos a utilizar la distancia entre estos y el ángulo de su normal, es decir, su orientación. Manejamos un umbral de distancia máxima a la que pueden estar separados dos *patches* para fusionarlos $dist(\Pi_i, \Pi_j) < U_d$. Esta distancia tiene que ser mayor que el umbral que tomamos para la formación de los *patches*, y, al mismo tiempo, tendrá que ser lo suficientemente pequeña para asegurar que no se van a cerrar aperturas como inicio de pasillos, puertas, etc. Por otro lado, para hacer efectiva la fusión se ha de cumplir que la diferencia de las orientaciones sea menor que un umbral $|Orientacion(\Pi_i) - Orientacion(\Pi_j)| < U_o$. Los valores U_d y U_o se obtienen experimentalmente.

Una vez que hemos comprobado que dos *patches* forman parte del mismo objeto, pasamos a fusionarlos en uno solo. En primer lugar, hemos de obtener los parámetros del plano resul-

tante (un punto del plano y un vector normal al plano). Tomamos la media ponderada, tanto del punto como de las normales, en función del área de cada uno de los *patches*. El hacerlo de esta manera está fundamentado en el hecho de que los *patches* mayores se obtienen a partir de un número mayor de evidencia en el entorno y, por lo tanto, podemos confiar más de la información que ofrecen. En segundo lugar, hemos de encontrar los límites del nuevo plano. Para ello vamos a proyectar los límites de los *patches* de origen sobre el plano resultado y tomaremos como límites los puntos que estén más alejados del punto que define el plano.

4.4. Introduciendo información del entorno

Podemos mejorar los resultados obtenidos incorporando en nuestro modelo el conocimiento que tenemos del entorno. Sabemos que hay partes de un entorno que permanecen invariables a lo largo del tiempo, como por ejemplo las paredes de un edificio. Además, estas partes invariables presentan una serie de relaciones, como las geométricas, que podemos incorporar a nuestro modelo como restricciones que ha de cumplir el resultado obtenido. De esta manera introducimos a nuestro modelo restricciones del tipo "*Todos los planos del entorno han de ser horizontales*" o "*Dos planos del entorno tienen que ser paralelos o perpendiculares*" [2].

Con la incorporación de esta información en nuestro modelo reducimos el error cometido tanto en las medidas como durante el proceso de extracción y fusión de planos. Es importante comentar que estas restricciones no se toman como un proceso a parte que se realiza *a posteriori*, sino que está empotrado en el modelo. En nuestro caso realizamos la comprobación de las restricciones tanto durante el proceso de extracción de planos como en el de fusión.

4.5. Resultados

Partimos de un conjunto de más de 4 millones de puntos 3D obtenidos por el robot en el entorno (ver figura 11). De este conjunto de puntos eliminamos los correspondientes al techo y al suelo, en caso de que los haya, ya que estos puntos suelen incluir mucho error. Del conjunto de puntos resultante estimamos sus normales (ver figura 10) y procedemos a la extracción y agrupación de superficies verticales planas tal y como se ha explicado anteriormente (figuras 12 y 13). Finalmente, introducimos la información que tenemos del entorno para obtener una reconstrucción completa (figura 14)

Figura 11: Detalle de los puntos 3D adquiridos del entorno.

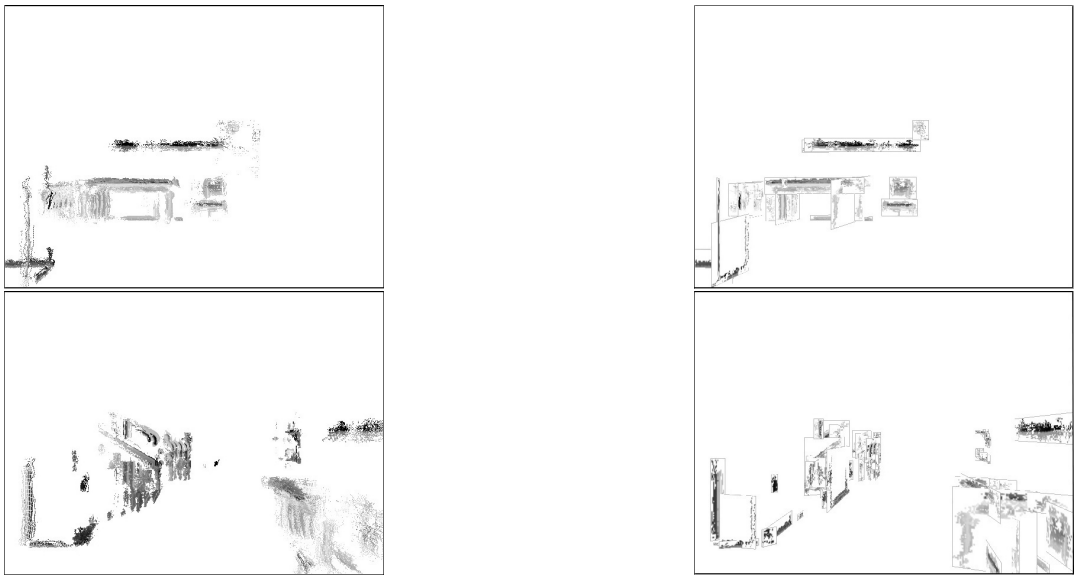

Figura 12: Detalle de la extracción de superficies verticales. Los puntos 3D de la escena se han agrupado formando *patches*. Izda: Conjunto de puntos 3D. Dcha: Planos extraídos.

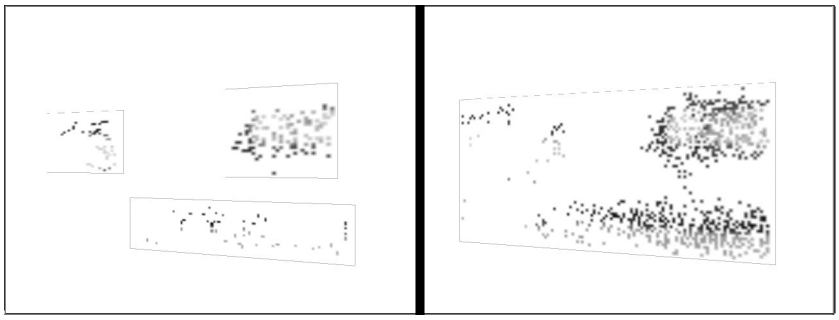

Figura 13: Detalle de la fusión de planos. Izda: planos sin fusionar. Dcha: resultado de la fusión.

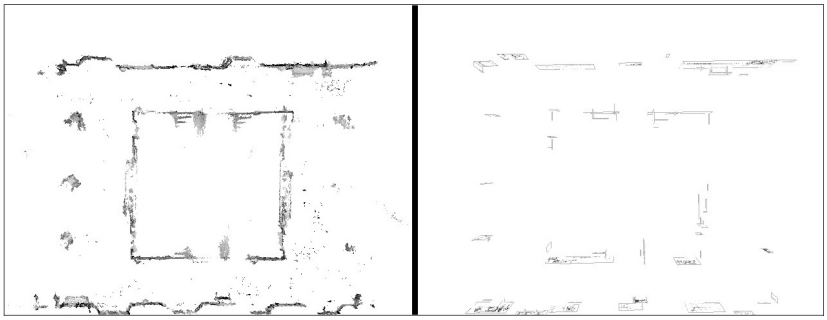

Figura 14: Vista aérea del entorno. Izda: conjunto de puntos 3D. Dcha: resultado de la extracción y fusión de planos.

5. Conclusiones y trabajos futuros

En este trabajo se ha planteado una primera aproximación a la reconstrucción 3D de un entorno. Los datos 3D son tomados a partir de una cámara estéreo, lo que implica un tratamiento especial de los datos frente a otros enfoques con mayor cantidad de ellos. Primero aplicamos un algoritmo de emparejamiento de puntos 3D adecuándolo al problema a tratar. Después realizamos la extracción de planos, como la primera primitiva geométrica a extraer.

Como continuación de este trabajo pretendemos abordar el problema del SLAM, comentado anteriormente, para resolver los errores que se producen en el cálculo del mapa 3D. También pretendemos ampliar el trabajo para extraer otras primitivas geométricas, tales como cilindros, cubos (a partir de planos), etc. Por otro lado estamos intentando reducir errores en la toma de datos, creando una textura artificial mediante emisores de infrarrojos.

Agradecimientos

Este trabajo ha sido realizado con una ayuda de la Secretaría de Estado de Educación y Universidades.

Referencias

[1] P. Allen and I. Stamos. Integration of range and image sensing for photorealistic 3d modelling. In *Proc of IEEE Inter. Conf. on Robotics and Automation*, 2000.

[2] P. Allen and I. Stamos. Solving architectural modelling problems using knowledge. In *Proc. 4th Int. Conf. on 3-D Digital Imaging and Modeling*, 2003.

[3] P. Besl and N. McKay. A method for registration of 3-d shapes. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[4] M. Cazorla and B. Fisher. Characterizing local minima in 3d registration methods. In *Proc of the British Machine Vision Conference (en proceso de revision)*, 2004.

[5] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley-Interscience, 2000.

[6] D. Hahnel, W. Burgard, and S. Thrun. Learning compact 3d models of indoor and outdoor enviroments with a mobile robot. *Robotics and Autonomous Systems*, 44:15–27, 2003.

[7] S. Hakim, P. Boulanger, and F. Blais. A mobile system for indoors 3-d mapping and positioning. In *Proc of Fourth Conference on Optical 3-D Measurement Techniques*, 1997.

[8] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun. Using em to learn 3d models of indoor environments with mobile robots. In *Proc of International Conference on Machine Learning*, 2001.

[9] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[10] J.M. Saez, A. Penyalver, and F. Escolano. Estimaciones de las acciones de un robot utilizando visión estéreo. In *Proc of 4th Workshop de Agentes Físicon*, 2003.

[11] H. Surmann, A.Ñuchter, and J. Hertzberg. An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments. *Robotics and Autonomous Systems*, 45:181–198, 2003.

[12] S. Thrun, D. Fox, and W. Burgard. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31:29–53, 1998. also appeared in Autonomous Robots 5, 253–271 (joint issue).

[13] S. Thrun, D. Hähnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, and W. Whittaker. A system for volumetric robotic mapping of abandoned mines. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[14] A.L. Yuille and P. Hallinan. Deformable templates. In A. Yuille and A. Blake, editors, *Active Vision*. MIT Press, 1992.