https://doi.org/10.17979/spudc.9788497498081.0374

ALINEAMIENTO 3D DESDE POSICIONES NO CERCANAS DE UN ROBOT PARA TRABAJOS EN INTERIORES A PARTIR DE IMÁGENES RGB-D

S. Cebollada, C. Parra, M. Juliá, M. Holloway, L.M Jiménez, O. Reinoso
Departamento de Ingeniería de Sistemas y Automática.
Universidad Miguel Hernández de Elche. Avda. de la Universidad s/n. 03202, Elche (Alicante), Spain
Q-Bot Limited, Block G, Riverside Business Centre, Bendon Valley, SW18 4UQ, London, UK
{migel.julia, mathew.holloway }@q-bot.co, {sergio.cebollada, cristobal.parra01, luis.jimenez, o.reinoso }@umh.es

Resumen

En este artículo se presenta un estudio de alineamiento 3D de diferentes posiciones de un robot en entornos de interior. El alineamiento que se desea resolver se desenvuelve en escenarios de difícil caracterización. Para resolver este problema se plantea utilizar información RGB-D, esto es, información visual y de profundidad obtenida del entorno a través de cámaras y sensores láseres respectivamente. En primer lugar se expone el algoritmo desarrollado a través de información visual. Después se expone el algoritmo desarrollado para información de profundidad. Finalmente, se presenta un algoritmo que resuelve las situaciones en las cuales el alineamiento entre dos poses no es correcto.

Palabras clave: Robótica, Reconstrucción 3D, Alineamiento, Matriz de transformación, SLAM, RGB-D.

1. INTRODUCCIÓN

Dadas las construcciones de edificios estándares, el acceso a los subsuelos de éstas es bastante difícil y en ocasiones imposible para el ser humano. A pesar de esto, se dan situaciones en las cuales hay que acceder de forma inevitable. Como consecuencia, una posible solución para acometer trabajos en este tipo de entornos consiste en el empleo de robots que estén perfectamente localizados y puedan acometer las tareas precisas.

Para generar un mapa del entorno, se requiere la obtención de datos a través de lecturas de sensores. En teoría, la transformación del sistema de coordenadas entre dos lecturas puede obtenerse a través de la odometría de las ruedas del robot, sin embargo, los modelos del movimiento son inexactos (derivas de las ruedas, fricción, parámetros cinemáticos imprecisos, etc.) y por tanto dan lugar a lecturas erróneas. Es por este motivo por el que surgen algoritmos para estimar la trayectoria del robot a partir de las lecturas obtenidas de sensores. Estos forman parte de lo que se conoce como SLAM ("*Simultaneous Localization And Mapping*") [12].

La tarea de SLAM en este entorno no resulta fácil. Los algoritmos actualmente desarrollados ([3] o [6]) no funcionan correctamente. Esto es debido a las características del entorno, ya que no se puede aproximar el movimiento a 3 grados de libertad sobre una superficie plana debido a la rugosidad del terreno, la mala iluminación y el operar en entornos confinados. Como consecuencia, el robot tiene muchas dificultades para caracterizar el entorno.

Dadas varias poses del robot en un escenario, se intentará solucionar la tarea de SLAM mediante la obtención de datos del entorno que puedan ser utilizados para conocer el movimiento que se ha producido. De modo que, para estimar la posición del robot respecto a la posición previa, se calculará la matriz de transformación, la cual, indica la rotación y traslación que se ha llevado a cabo.

Uno de los métodos empleados para resolver el problema de SLAM es a través de descriptores visuales. Este método consiste en obtener puntos característicos de imágenes obtenidas en distintas poses del robot, tras esto, se caracteriza cada punto (cálculo del descriptor), asignándole un valor que singularice el punto y permita reconocerlo frente a otros puntos. A través de los descriptores, se obtienen correspondencias entre puntos de distintas poses y a través de estas correspondencias, se obtiene la matriz de transformación. SIFT [9] y SURF [2] son dos de los algoritmos más populares para la detección de puntos característicos y el cálculo de descriptores. Estos obtienen y calculan detectores y descriptores a través de gradientes locales y direcciones específicas. Sin embargo, en prácticamente todos los planteamientos desarrollados, las características que se extraen son sensibles a variaciones de iluminación.

Otras posibilidad consiste en emplear la informa-

ción procedente de un sensor láser que barra el entorno 3D. Estos dispositivos obtienen de forma muy precisa información de profundidad (nubes de puntos). De modo que se parte de dos nubes de puntos obtenidas desde poses distintas del entorno y la tarea a realizar consistirá en conseguir un buen alineamiento entre éstas. *Iterative Closest Point* (ICP) [4] es el método comúnmente utilizado para resolver el registro entre nubes de puntos. A partir de este método, nacen variantes como [11], que utiliza correspondencias plano-plano en lugar de punto-punto. En [1] se utiliza SVD (*Singular Value Descomposition*) para calcular la matriz de transformación. Otra forma de variar el algoritmo ICP es a través de variar los métodos de búsqueda de puntos cercanos, como en [8]. Los algoritmos ICP desarrollados funcionan bastante bien. Algunos de estos pueden obtener resultados precisos en tiempo real, lo cual podría resolver el objetivo de SLAM. Sin embargo, ninguno de estos algoritmos funcionan bien en los entornos como el propuesto debido a la falta de luz, la poca variedad de colores y la geometría del entorno.

Dado que ninguno de los métodos desarrollados hasta el momento soluciona el problema en entornos como el propuesto, nace la necesidad de desarrollar un algoritmo que sea capaz de trabajar bien en estos tipos de entornos. El objetivo de este artículo es presentar dos algoritmos diseñados que posibilitan la localización y construcción de un mapa del entorno a partir de los datos adquiridos por un sistema de visión y un sistema láser que barren todo el entorno.

Este artículo se organiza como sigue. La Sección 2 presenta el sistema de adquisición de datos empleado para los experimentos llevados a cabo. A continuación, la Sección 3 describe los dos métodos propuestos para obtener alineamiento entre poses del robot. Seguidamente, en la Sección 4 se presenta un algoritmo desarrollado para solventar las situaciones en las cuales no se haya obtenido un buen alineamiento entre dos poses. La Sección 5 expone los resultados obtenidos en los experimentos. Finalmente, la Sección 6 resume las conclusiones alcanzadas.

2. SISTEMA DE ADQUISICIÓN DE DATOS

El sistema de adquisición de datos está compuesto por un sensor láser y una cámara monocular, que rotan en torno al eje perpendicular al plano de movimiento del robot, recopilando información de la escena en los 360 grados alrededor del robot, permitiendo obtener un escaneo completo del escenario y un conjunto de imágenes RGB.

Concretamente, se recopila un conjunto de 37 imágenes RGB para cada pose P_i y se define cada imagen de ese conjunto como $K_{i,j} \in \mathbb{R}^{N_x \times N_y}$, donde j=0,1,2,...,36 y $N_x \times N_y$ es la resolución de cada imagen, en este caso 1448×1928 pixeles.

Respecto al sistema de adquisición láser, existen 0.36 grados de distancia entre los puntos verticales del scan. Este scan vertical cubre 240 grados, pero solo se utilizan los 120 grados correspondientes a la parte frontal. El motor que hace girar la cámara y el láser tiene 2400 pasos, por lo tanto hay 0.15 grados de distancia entre los puntos horizontales del scan. Sin embargo, estos dos motores no están perfectamente sincronizados, por que lo puede haber un error de un paso. Por tanto, la nube de puntos creada estará formada por un total de 800.000 puntos.

3. ALINEAMIENTO

3.1. ALINEAMIENTO VISUAL

El algoritmo que se desarrolla a continuación, estima la matriz de transformación del robot $T_{i,i+1}$ entre una pose previa P_i y la actual P_{i+1}, empleando la información de profundidad facilitada por el láser y las imágenes capturadas por la cámara, que se transforman de RGB a escala de grises. Para obtener $T_{i,i+1}$, se lleva a cabo el alineamiento de dos nuevas nubes de puntos creadas a partir de la información más relevante de la escena, extrayendo características visuales entre dos poses del robot.

Dado que se desconoce la orientación en que se comenzó a capturar el conjunto de imágenes K_i en cada pose, se precisa de manera previa a la búsqueda de correspondencias entre imágenes, la búsqueda de la imagen $K_{i,j}$ en P_i más similar con la imagen $K_{i+1,j}$ en la pose P_{i+1}. Una vez conocidos los j pares de imágenes más similares entre P_i y P_{i+1}, se detectan características visuales en la imagen $K_{i,n}$ con $n = [0, 6, 12, 18, 24, 36]$ y sus correspondiente imagen más similar en P_{i+1}, definida como $K_{i+1,n+m}$. De entre estas marcas visuales, se obtienen correspondencias con el menor número de *outliers*, y para cada punto característico de ambas imágenes que obtuvo una correspondencia válida, se obtiene su información tridimensional (empleando la información de profundidad). Esta información espacial adquirida para los puntos característicos de las n imágenes se almacena en una misma nueva nube de puntos correspondiente a la pose P_i, y de igual manera, se crea otra nueva nube de puntos para la pose P_{i+1} con la información espacial de los puntos que obtuvieron correspondencias válidas en las imágenes $K_{i+1,n+m}$. Estas nuevas nubes de puntos contienen del orden de 50

a 250 puntos, lo que reduce el tiempo de ejecución para obtener la matriz de transformación considerablemente. Al alinear estas dos nubes de puntos, se obtiene una estimación de la matriz de transformación $T_{i,i+1}$ de la pose actual respecto a la anterior.

Así, el algoritmo desarrollado puede dividirse en las siguientes etapas:

- Apariencia visual para emparejar las imágenes más similares entre poses.
- Detección, descripción y correspondencia de marcas visuales entre pares de imágenes.
- Extracción de la información de profundidad para crear las nubes de puntos y su posterior alineamiento.

3.1.1. Apariencia visual

Para determinar la imagen más similar entre la imagen $K_{i,j}$ de entre todas las posibles del conjunto K_{i+1}, se emplean descriptores basados en apariencia global, de manera que se procesa la imagen en su conjunto, evitando la segmentación o extracción de puntos característicos de la escena.

Para construir el descriptor de apariencia global, se hace uso de la Firma de Fourier [10]. Esta técnica presenta ventajas como su simplicidad y su bajo coste computacional.

El método consiste en representar cada imagen $K_{i,j} \in \mathbb{R}^{N_x \times N_y}$ utilizando la Transformada discreta de Fourier de cada una de sus filas. Por tanto, es necesario expandir cada fila de la imagen $\{a_n\} = \{a_0, a_1, ..., a_{N_y-1}\}$ y obtener la secuencia de números complejos $\{A_n\} = \{A_0, A_1, ..., A_{N_y-1}\}$. De este modo, se llega a una nueva matriz $D_{i,j} \in \mathbb{C}^{N_x \times N_y}$ en la que la mayor parte de la información relevante se encuentra concentrada en las q primeras columnas, en este caso concreto $q = 32$. La nueva matriz de tamaño N_X filas y q columnas se llama *Firma de Fourier*, de la que se puede obtener su matriz de módulos $A_j = \|D_j\|$, que al concatenar resulta en un vector de una única columna con $N_x \cdot q$ filas, que es el descriptor asociado a esa imagen.

De esta forma, se calcula un descriptor para cada imagen $K_{i,j}$ y también para $K_{i+1,j}$ donde j es el conjunto de imágenes adquiridas en cada pose. Tras ello se busca la menor distancia euclídea entre descriptores de cada conjunto de imágenes, generándose una matriz de tamaño $j \times j$ con la relación de imágenes más similares entre cada pose. En la Figura 1(a) se muestra un ejemplo de esta matriz. Dado que el movimiento de adquisición de imágenes en cada posición no es aleatorio,

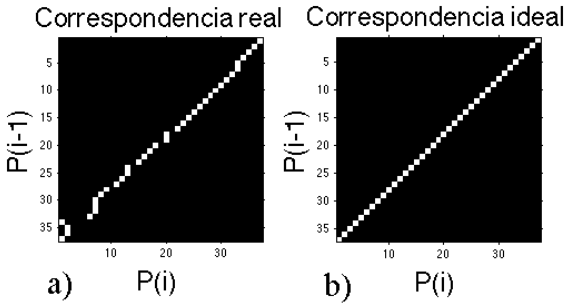

Figura 1: Relación de los conjuntos de imágenes entre dos poses: (a) es la matriz real obtenida y (b) es la matriz final tras eliminar posibles *outliers*.

sino que cada imagen del conjunto guarda relación con la anterior, para un conjunto de n imágenes se pueden encontrar únicamente $2 \cdot n$ posibles soluciones. Comparando la matriz resultante anterior con cada una de las posibles soluciones mediante una correlación empleando la distancia euclídea, se obtiene un resultado robusto ante posibles errores en el emparejamiento, con el que se obtiene el valor de m. En la Figura 1 se puede apreciar (a) una matriz con errores debido a la similitud de la escena y como (b) corrige esos errores ofreciendo la relación correcta y obteniendo así el valor m que relaciona la imagen del conjunto K_i con el conjunto K_{i+1}.

3.1.2. Detección, descripción y correspondencia de marcas visuales

Tras haber localizado los pares de imágenes entre la pose P_i y la pose P_{i+1}, se hace uso de la librería OpenCV [5] para la detección, descripción y correspondencia de marcas visuales. Un ejemplo de estas correspondencias puede verse en la Figura 2. Mientras que computacionalmente, SURF puede ser más exigente que otros como ORB (basado en el detector FAST y el descriptor BRIEF), SURF ha demostrado ofrecer buenos resultados en el escenario en el que nos encontramos.

Figura 2: Extracción de características visuales entre dos imágenes tomadas en poses consecutivas.

Para la detección de los puntos característicos, se

emplea un valor de *Hessiana* constante, y a continuación se obtiene el descriptor de cada uno de ellos. Tras la descripción, se realiza un ajuste automático que limita el número de puntos de interés en la escena por debajo de un determinado umbral, basado en la calidad del descriptor obtenido, eliminando así posibles *outliers*. Por último, se lleva a cabo la correspondencia haciendo uso de la norma $L2$, consiguiendo de este modo pares de puntos correspondientes pertenecientes a cada imagen $K_{i,n}$ y $K_{i+1,n+m}$.

3.1.3. Extracción de la información de profundidad y alineamiento

De cada punto característico que encontró una correspondencia válida, se obtiene su información tridimensional y se incluye en una nueva nube de puntos, una para la pose P_i y otra para la P_{i+1}, cada una con la información espacial de todos los puntos que obtuvieron correspondencias válidas en su conjunto de imágenes.

A raíz de esta representación tridimensional de la escena, se procede a la estimación de la matriz de transformación del sistema, haciendo uso de la librería PCL *(Point Cloud Library)* [7], indicando al algoritmo la correspondencia entre cada par de marcas visuales obtenidas en el apartado 3.1.2 y que ahora están representadas tridimensionalmente en las dos nubes de puntos. De este modo, el programa procesa diferentes matrices de transformación eliminando posibles *outliers* hasta obtener una lo suficientemente cercana a la solución correcta para que los puntos coincidan tras su aplicación. Además de la matriz de transformación, es posible obtener una medida de confianza, que indica la validez de los resultados obtenidos en función de unos rangos calculados de manera experimental en función de los *outliers* localizados a lo largo del algoritmo y el número de correspondencias encontradas.

3.2. ALINEAMIENTO A PARTIR DE DATOS DE PROFUNDIDAD

Los algoritmos ICP desarrollados funcionan bastante bien en cuanto a precisión y tiempo de computo. Sin embargo, como ya se mencionó anteriormente, ninguno de estos algoritmos parece funcionar bien en los entornos propuestos. La poca iluminación y la poca variedad de colores aporta poca información. Además, como hemos mencionado previamente, no existen objetos característicos, con lo que el registro es todavía mas difícil.

En este apartado, se plantean posibles propuestas a realizar sobre la información de profundidad que se dispone para así obtener un alineamiento satisfactorio.

El algoritmo desarrollado puede dividirse en tres partes:

- Selección de puntos
- Registro
- Validación

3.2.1. Selección de puntos

Por lo general, los sensores láser obtienen nubes de puntos que aportan más de 800.000 puntos. Como consecuencia, el tiempo de computo en cualquier algoritmo ICP es muy elevado y los resultados obtenidos no son lo suficientemente buenos. Dado que trabajar con la nube de puntos completa no funciona, parece una buena alternativa usar ciertos puntos de ésta.

La naturaleza del entorno presenta varias adversidades para obtener un registro correcto. Como se ha mencionado previamente, el entorno no tiene características suficientemente válidas para poder utilizar extracción de características. Además, la información que aporta el techo es mala debido a la presencia de vigas, las cuales confunden al algoritmo ICP, realizando alineaciones entre nubes incorrectas. Por lo que un nuevo método de extracción de puntos resultará interesante para facilitar el registro.

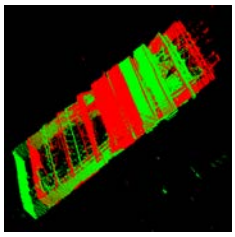

Figura 3: Un ejemplo de registro donde la alineación no ha sido correcta. Las vigas producen un importante error.

Para el método que se presenta, en un primer paso se realiza un filtro homogéneo sobre toda la nube de puntos para hacer la nube más ligera. Esto reducirá el tiempo de computo y facilitará la tarea de registro. Gracias a la librería PCL, se pueden utilizar funciones ya implementadas para obtener buenos filtros de puntos. Estas funciones son *VoxelGrid* y *RandomSample*, las cuales están implementadas en las librerías *voxel _grid* y *random_sample* respectivamente. En un primer paso, se aplica sobre la nube de puntos un filtrado de rejilla (también conocido como filtro voxelgrid), es decir, se mantiene como mucho un punto en cada cubo de 1 x 1 x 1 cm. El segundo paso será

realizar un filtrado aleatorio sobre los puntos resultantes. Este paso será más rápido cuantos más puntos eliminemos, sin embargo, eliminar demasiados puntos puede resultar contraproducente ya que el algoritmo ICP podría no funcionar correctamente.

En segundo lugar, se lleva a cabo una segmentación de la nube de puntos. Como fue mencionado anteriormente, el techo y el suelo presentan muchos inconvenientes. Esto hace casi imposible obtener un buen alineamiento entre las dos nubes de puntos. En consecuencia a esto, la idea será seleccionar únicamente planos que puedan ayudar a alcanzar un registro satisfactorio. A través de la función en la PCL, *SACSegmentation*, la nube de puntos es reducida y los puntos pertenecientes al techo o suelo son eliminados. Como resultado, se obtiene una nube más ligera en la cual las vigas del techo ya no causarán problemas de alineación.

3.2.2. Registro

Se han desarrollado muchas versiones de ICP que funcionan mejor que el estándar, como en [8], donde se acelera el proceso de búsqueda, o [11], que ha probado ser mas robusto frente correspondencias incorrectas. A pesar de esto, puede resultar interesante el uso de la original, en la cual se realiza alineación punto a punto [4], ya que los resultados que se obtienen con este método en cuanto a precisión y tiempo de computo son suficientemente buenos.

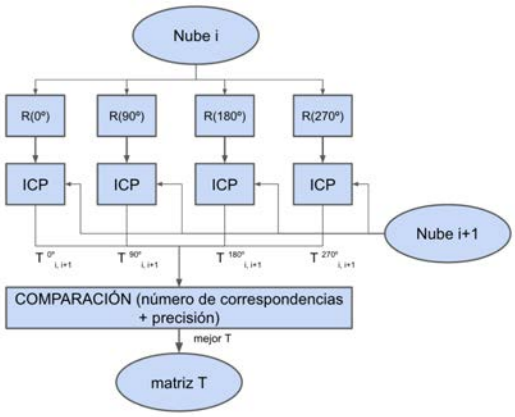

Figura 4: Esquema del algoritmo. R(x^{o}) es la rotación inicial introducida, donde x indica el ángulo de rotación. $T^{x^{o}}_{i,i+1}$ es la matriz de transformación obtenida tras realizar ICP donde x es el ángulo de rotación inicial. Inicialmente se rota la nube i en cuatro diferentes ángulos (0º, 90º, 180º y 270º). Después, se realiza ICP para cada una de las nubes rotadas con respecto a la nube *i+1*. A través de los valores de precisión y número de correspondencias, se escoge la mejor matriz de transformación.

3.2.3. Validación

Aunque este proceso selecciona el mejor registro ICP de entre los 4 realizados, esto no asegura que la alineación sea correcta. Por tanto, el algoritmo incluye un paso de validación general. Después de haber seleccionado el mejor registro, se comprueba si la precisión es inferior a un cierto valor y si el porcentaje de correspondencias con respecto al número de puntos que forman la nube (filtrada) es mayor a un determinado umbral. En resumen, si los valores obtenidos en el mejor registro ICP son mejores que los establecidos por los umbrales, el registro entre las dos nubes será aceptado como válido, en caso contrario, el registro será rechazado.

4. ALGORITMO PARA RECUPERAR POSICIONES PERDIDAS

Nada puede asegurar que la distancia entre una pose y la siguiente sea pequeña. Adicionalmente, debido a las condiciones del suelo, el robot realiza movimientos en 3 dimensiones, ya que durante el movimiento, puede haber variaciones en el suelo, las cuales harán que varíe la altura y la inclinación del robot.

Ante estas dificultades, pese haberse desarrollado algoritmos de alineamiento bastante robustos, pueden surgir casos en los que la alineación entre dos poses no sea correcta. Es por eso que nace la necesidad de implementar un algoritmo que sea capaz de localizar la pose del robot en situaciones como la descrita. Este algoritmo, además de intentar alinear una pose que no ha sido registrada correctamente con la anterior, también intentará localizar poses que fueron descartadas en pasos anteriores.

Algoritmo 1 Algoritmo para obtener una alineación correcta

si $Registro\{P_{\text{i}}, P_{\text{i-1}}\} \neq$OK **entonces**
  $P_{\text{i}} \leftarrow T * P_{\text{i-1}}$
  $Ordenar\{Poses_{Correctas}, Distancia\{P_{\text{i}}\}\}$
  para x=0:(N_{poses} - 2) **hacer**
    $Registro\{P_{\text{i}}, P_{\text{x}}\}$
    si $Registro\{P_{\text{i}}, P_{\text{x}}\}$=$OK$ **entonces**
      guardar{T}
      posicionar{P_{i}}
      Fin
    fin si
  fin para
  $Lista_{pendientes} \leftarrow P_i$
fin si

Para explicar el algoritmo, se parte del supuesto en

el que se quiere localizar la pose P_i. En un primer paso, se intenta realizar uno de los métodos de alineamiento que se han desarrollado. Para ello, se realiza el registro entre P_i y la pose detectada previamente a esta (P_{i-1}).

Si el registro no fuese válido, se comenzaría la búsqueda de otra pose con la que poder conseguir una buena alineación (Algoritmo 1). Para esto, pese a que la matriz de transformación obtenida en el registro entre P_i y P_{i-1} no es correcta, utilizamos esta matriz (T) para localizar la última pose en el mapa con respecto al resto de poses ya localizadas. La ubicación de la pose no será fiable, ya que la hemos calculado a través de una mala matriz de transformación, sin embargo, esta ubicación es suficiente para hacer una gruesa aproximación. Tras esto, calculamos la distancia que existe entre P_i y el resto de poses (todas las poses desde la 0 hasta la i-2 que estén correctamente alineadas). Una vez calculada las distancias entre poses, se comienza a intentar registrar P_i con la pose mas cercana a ésta, este paso se realizará de forma iterativa probando la poses de menor a mayor distancia hasta que se consiga una alineación satisfactoria.

Llegados a este momento, pueden suceder dos situaciones, la primera, que se haya conseguido una alineación satisfactoria con alguna de las poses. En cuyo caso, el problema de alineamiento quedaría resuelto para la pose P_i. La segunda situación que se puede dar es que no se consiga realizar un correcto registro con ninguna pose, en este caso, P_i se guardará en una lista de poses pendientes por alinear.

Llegado el momento de localizar P_i, si el registro con la pose P_{i-1} fue satisfactorio pero P_{i-1} está en la lista de pendientes, se realizará de forma recursiva el registro de P_i con las poses más cercana hasta que se alinee correctamente. Si P_i fue registrada de forma correcta con alguna de las poses que están bien localizadas, se posicionará en el mapa y tras esto, P_{i-1} pasará a estar considerada como localizada correctamente, eliminándola de la lista de poses pendientes y posicionándola en el mapa (Algoritmo 2).

Finalmente, con la finalidad de intentar localizar de forma correcta las poses que están pendientes. Tras cada nueva pose ubicada exitosamente, se intentará registrar la nueva pose con todas las que están en la lista de pendientes.

Algoritmo 2 Algoritmo para solventar situaciones cuando la pose anterior no fue alineada correctamente

si ($Registro\{P_i, P_{i-1}\}$=OK) Y ($Lista_{pendiente}\{P_{i-1}\}$=$True$) **entonces**
  $P_i \leftarrow T * P_{i-1}$
  $Ordenar\{Poses_{Correctas}, Distancia\{P_i\}\}$
  para x=0:(N_{poses} - 2) **hacer**
    $Registro\{P_i, P_x\}$
    si $Registro\{P_i, P_x\}$=OK **entonces**
      guardar{T}
      posicionar{P_i}
      posicionar{P_{i-1}}
      eliminar{$Lista_{pendiente}\{P_{i-1}\}$}
      Fin
    fin si
  fin para
  $Lista_{pendientes} \leftarrow P_i$
fin si

5. EXPERIMENTOS Y RESULTADOS

En esta sección se muestran los resultados obtenidos de los experimentos que se han llevado a cabo para comprobar los dos métodos presentados en la Sección 3. Ambos métodos son desarrollados en un entorno indoor que cumple las condiciones descritas anteriormente por las cuales estos métodos pueden resultar interesantes.

Los experimentos han sido desarrollado en un PC con un procesador 2 x 2,8 GHz Quad-Core Intel Xeon ®. Para la realización de este experimento, se obtienen las imágenes y las medidas láseres tomadas por el robot en una determinada posición del entorno. Con esta información, se obtiene una nube de puntos del entorno. Por tanto, se obtendrá una nube de puntos por cada pose del robot a lo largo del entorno. A priori, el movimiento llevado por el robot es totalmente desconocido, por lo que no se pueden comparar los resultados medidos con los reales. Por este motivo, la mejor forma de medir la calidad de la alineación obtenida es a través de unos valores de confianza que indican la calidad en la estimación de la matriz de transformación obtenida.

5.1. ALINEAMINETO VISUAL

Como se puede apreciar, en la **tabla 1** se han recogido los valores obtenidos llevando a cabo un alineamiento visual entre poses consecutivas (P_{i+1} con respecto a P_i). De modo que para cada par de nubes de puntos resultantes, se recoge la siguiente información: **Características**, es decir, el número total de puntos obtenidos en cada conjunto de

Tabla 1: Resultados del experimento de alineamiento visual a partir de imágenes y datos de profundidad.

Poses	**Corr.**	**Caract.**	**% Corr.**
P_0, P_1	102	138	73
P_1, P_2	105	160	65
P_2, P_3	84	105	80
P_3, P_4	130	162	80
P_4, P_5	91	142	64
P_5, P_6	65	97	67
P_6, P_7	87	115	75
P_7, P_8	107	164	65
P_8, P_9	79	105	75
P_9, P_{10}	126	206	80
P_{10}, P_{11}	86	112	76
P_{11}, P_{12}	141	204	69
P_{12}, P_{13}	98	132	74
P_{13}, P_{14}	33	64	51
P_{14}, P_{15}	47	90	52
P_{15}, P_{16}	102	122	83
P_{16}, P_{17}	30	71	42
P_{17}, P_{18}	50	85	58
P_{18}, P_{19}	36	131	27
P_{19}, P_{20}	27	73	36

imágenes. **Correspondencias**, esto es, el número de correspondencias llevadas a cabo entre las dos nubes de puntos para obtener la mejor estimación de T. **% Correspondencias**, el porcentaje de correspondencias encontradas entre una nube con respecto al número total de puntos que forman la nube de puntos.

Los resultados muestran que el proceso de alineamiento entre dos nubes de puntos llevado a cabo a través de este método es fiable para el entorno evaluado. Sin embargo, se puede apreciar cómo entre los dos últimos pares de poses (P_{18} con P_{19} y P_{19} con P_{20}), se obtiene un porcentaje de correspondencias por debajo del 40 %, valor experimental a partir del cual se conoce que la alineación no ha sido del todo precisa.

5.2. ALINEAMINETO A PARTIR DE DATOS DE PROFUNDIDAD

Como se puede apreciar, en la **tabla 2** se han recogido los valores obtenidos para cada registro realizado entre dos nubes de puntos. Se ha realizado el registro entre poses consecutivas. De modo que, para cada par de nubes registradas, se recoge la siguiente información: **% Correspondencias**, esto es, el número de correspondencias encontradas entre las dos nubes con respecto al número total de puntos que forman la nube de puntos (tras haber sido filtrada uniformemente). **RMSE**

Tabla 2: Mediciones tomadas en el experimento de alineamiento a partir de datos de profundidad

Poses	**% Corr. nube *i-1***	**% Corr. nube *i***	**RMSE (m)**	**Tiempo (s)**
P_0, P_1	45,08	41,66	0,0136	5,10
P_1, P_2	47,07	45,51	0,0028	6,38
P_2, P_3	43,97	40,72	0,0072	6,39
P_3, P_4	45,47	44,02	0,0063	5,58
P_4, P_5	32,69	31,42	0,0129	7,33
P_5, P_6	41,56	41,15	0,0149	8,58
P_6, P_7	36,43	37,00	0,0172	7,51
P_7, P_8	40,05	40,20	0,0126	5,07
P_8, P_9	42,03	44,09	0,0091	5,88
P_9, P_{10}	40,36	39,50	0,0137	4,89
P_{10}, P_{11}	42,93	45,71	0,0143	4,83
P_{11}, P_{12}	30,21	34,18	0,016	4,69
P_{12}, P_{13}	39,64	43,46	0,0137	5,69
P_{13}, P_{14}	38,52	31,58	0,0221	4,47
P_{14}, P_{15}	36,48	34,31	0,0168	4,62
P_{15}, P_{16}	47,51	46,47	0,0084	6,52
P_{16}, P_{17}	38,03	36,34	0,0156	4,87
P_{17}, P_{18}	37,03	36,25	0,0198	9,41
P_{18}, P_{19}	13,78	19,40	0,01829	5,26
P_{19}, P_{20}	28,95	22,36	0,011	5,09

(Root Mean Square Error), basado en la distancia euclídea entre el conjunto de puntos origen transformados y los puntos destinos (solo se realiza el cálculo para puntos emparejados como correspondencias). Por último, el **Tiempo** que se ha tardado en realizar el registro.

Los resultados muestran que el proceso de alineamiento entre dos nubes de puntos se lleva a cabo a través de este método con un tiempo medio de unos 6 segundos, con lo que este método podría ser utilizado en sistemas SLAM en tiempo real. La media total de correspondencia de puntos está entorno al 38 % , sin embargo, se puede apreciar que el registro entre los dos últimos pares de poses (P_{18} con P_{19} y P_{19} con P_{20}) tienen un porcentaje de correspondencias muy bajo, de lo que se deduce que la alineación no ha sido correcta. Era de esperar que alguna alineación no fuese correcta, esto no supone un gran problema, ya que, como se ha explicado en la Sección4, para estos casos es por los que se ha desarrollado un algoritmo de recuperación de posiciones perdidas. Se puede deducir que el porcentaje medio de correspondencias para una alineación correcta en un entorno des este tipo es del 40 %.

6. CONCLUSIONES

Este artículo ha presentado dos nuevos algoritmos para tratar de resolver la tarea de alineamiento entre las poses de un robot en escenarios de difícil caracterización. Por un lado, visualmente, se han obtenido puntos característicos y se han calculado los descriptores, los cuales se corresponden con los puntos característicos obtenidos en otra pose del robot, con lo que se calcula la matriz de transformación a través de dichas correspondencias. Por otro lado, a través de la información de profundidad obtenida a través de sensores láser, se realiza el alineamiento de dos nubes de puntos, cada una obtenida desde una pose. Así, filtrando la información de techo y suelo y suponiendo 4 rotaciones iniciales, se obtiene el mejor alineamiento calculado, obteniendo de aquí la matriz de transformación. Por último, se ha presentado un algoritmo que solventa las situaciones en las cuales el alineamiento entre dos poses consecutivas no ha sido satisfactorio, tratando de alinear la pose actual con otras poses anteriormente alineadas.

En el futuro, será interesante tratar de mejorar la robustez de los algoritmos desarrollados, tratando de disminuir al mínimo el número de casos erróneos. Puede resultar interesante combinar ambos métodos, de forma que se puedan aprovechar las ventajas que presenta el alineamiento visual y el de profundidad.

Agradecimientos

Este trabajo ha sido financiado por el gobierno español a través del proyecto DPI2013-41557-P: "Navegación de robots en entornos dinámicos mediante mapas compactos con información visual de apariencia global". Y por la Generalitat Valenciana a través de las ayudas del programa AICO de la Consellería de Educación, Investigación, Cultura y Deporte con Ref: AICO/2015/02.

Referencias

[1] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, Sept 1987.

[2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia.

[3] Dominik Belter, Michał Nowicki, and Piotr Skrzypczyński. *Robot 2015: Second Iberian Robotics Conference: Advances in Robotics, Volume 2*, chapter Accurate Map-Based RGB-D SLAM for Mobile Robots, pages 533–545. Springer International Publishing, Cham, 2016.

[4] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256, Feb 1992.

[5] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Incorporated, 2008. C interface only, not C++ from version 2.1 onwards. New version coming out in mid-2012.

[6] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the rgb-d slam system. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1691–1696, May 2012.

[7] D. Holz, A. Ichim, F. Tombari, R. Rusu, and S. Behnke. Registration with the point cloud library: A modular framework for aligning in 3-d. *Robotics Automation Magazine, IEEE*, 22(4):110–124, Dec 2015.

[8] Timothée Jost and Heinz Hügli. *Pattern Recognition: 24th DAGM Symposium Zurich, Switzerland, September 16–18, 2002 Proceedings*, chapter Fast ICP Algorithms for Shape Registration, pages 91–99. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[9] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[10] E. Menegatti, T. Maeda, and H. Ishiguro. Image-based memory for robot navigation using properties of the omnidirectional images, 2004.

[11] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. *Robotics: Science and Systems*, 2(4), 2009.

[12] Sebastian Thrun and John J. Leonard. *Springer Handbook of Robotics*, chapter Simultaneous Localization and Mapping, pages 871–889. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.