Comparativa de detectores de características visuales y su aplicación al SLAM

A. M. Romero and M. Cazorla

***Abstract*—El problema del SLAM (Simultaneus Localization And Mapping) es un problema clave en el campo de la robótica. Las soluciones a dicho problema tienen problemas computacionales cuando el número de características aumenta. En este trabajo, pretendemos realizar una comparativa de dos detectores de características visuales, SIFT y SURF, así como proponer un nuevo método de emparejamiento que nos permita determinar si la imagen actual es parecida a una vista con anterioridad.**

***Index Terms*—Mobile Robotics, SLAM, características visuales, Matching de grafos**

I. INTRODUCCIÓN

El problema de robótica de Localización y Mapeado Simultáneo, conocido como SLAM (Simultaneous Localization and Mapping) trata de estimar la posición del robot y el mapa del entorno al mismo tiempo. La solución al problema no es trivial, puesto que el ruido de la posición afecta a la estimación del mapa y a su vez, el ruido que aparece en el mapa afecta a la estimación de la localización del robot. No obstante, a pesar de su dificultad, la solución a dicho problema se considera parte esencial en el campo de la robótica, puesto que permite al robot ser realmente autónomo en entornos desconocidos.

Una de las primeras soluciones aparecidas en la literatura fue EKF-SLAM (Extended Kalman Filter - SLAM) [1], [2]. Esta solución se basa en el uso de filtros de Kalman extendidos para estimar la posición de los landmarks (observaciones) y la posición del robot. EKF-SLAM utiliza modelos lineales para definir movimientos no-lineales, lo que comporta ciertos inconvenientes. La no-linealidad puede ser un problema importante para los algoritmos que utilizan EKF-SLAM, cuyas soluciones pueden llegar a ser inconsistentes [3]. La convergencia y la consistencia solamente están garantizadas cuando el problema es de tipo lineal.

El filtro de partículas Rao-Blackwellized (FastSLAM) [4] intenta solventar el problema de la no-linealidad de los filtros de Kalman. FastSLAM se basa en el método de Monte Carlo, también conocido como filtro de partículas debido a su uso de un conjunto de partículas donde cada una guarda una posible solución al problema. Esta solución representa el modelo de proceso de forma no-lineal y la distribución de la posición como no-Gaussiana (al contrario que EKF-SLAM). Además, el filtro de partículas es capaz de tratar con una gran cantidad de datos de entrada con un coste temporal aceptable.

Cuando la representación del entorno es un conjunto discreto de localizaciones (distintos lugares) del mapa global, hablamos de SLAM Topológico. En este caso SLAM no busca una localización exacta del robot, sino que lo que intenta es discernir en qué región del entorno se encuentra.

El SLAM topológico necesita un tratamiento particular de las imágenes de modo que el algoritmo sepa determinar cuando está viendo un nuevo lugar y cuando uno que se encuentra ya en su base de datos con la suficiente consistencia. Para ello se propone la utilización de imágenes omnidireccionales de las que se extraerán las distintas regiones que la forman (segmentación de imágenes en regiones) y los puntos descriptivos dentro de cada región (puntos invariantes de interés SIFT y SURF). Los algoritmos SIFT y SURF proporcionan puntos de interés invariantes a escala, rotación y en cierta parte a la iluminación. Dichas soluciones aportan gran cantidad de información por cada una de las imágenes procesadas.

El emparejamiento de los puntos invariantes suele producir numerosos "outliers", por lo que en el artículo se propone la utilización de grafos como estructura a comparar, en lugar de la comparación punto a punto. Con ello se pretende dotar de mayor fiabilidad al SLAM topológico para conseguir una localización más confiable y exacta.

En el artículo mostraremos cómo tratar las imágenes capturadas desde una cámara omnidireccional de forma que se extraigan los datos necesarios para el cálculo de la posición y el mapa mediante SLAM topológico. En la sección II describiremos los algoritmos SIFT y SURF y mostraremos una comparativa entre ambos algoritmos. A continuación (sección III) se explicará el método utilizado para la división de la imagen en distintas regiones. En IV se explicará el proceso de emparejamiento mediante grafos utilizando el algoritmo GTM. En la sección V podremos ver los resultados de los experimentos realizados. Finalmente tendremos un apartado con las conclusiones.

II. SIFT VS. SURF

A. SIFT: Scale Invariant Features Transforms

Uno de los métodos de extracción de puntos invariantes ampliamente utilizado en la literatura es el algoritmo SIFT (Scale Invariant Feature Transforms). Su nombre es debido a que transforma los datos de la imagen en coordenadas invariantes a la escala, rotación y en cierta medida al cambio de iluminación y al punto de vista 3D [5]. Cada uno de los datos extraídos se considera una característica de la imagen y se describe mediante su posición, escala, orientación y su vector descriptivo (habitualmente con un tamaño de 128).

El algoritmo se basa en cuatro etapas diferenciadas para extraer las características de la imagen:

Anna Maria Romero and Miguel Cazorla are with University of Alicante. E-mail: amrc4@alu.ua.es miguel@dccia.ua.es

1) Detección de máximos y mínimos de la escala.
2) Localización de puntos de interés.
3) Asignación de la orientación.
4) Descriptor del punto de interés.

En la primera etapa del algoritmo se identifica la posición y la escala de los puntos candidatos a ser características invariantes. El espacio de la escala, definido por $L(x, y, \sigma)$, se obtiene a partir de la convolución de la Gaussiana $G(x, y, \sigma)$ con la imagen $I(x, y)$. Para que la detección de los puntos de interés sea rápida y eficaz, el algoritmo utiliza la diferencia de Gaussianas convolucionada con la imagen. Dicha diferencia puede ser calculada a partir de las diferencias entre dos escalas vecinas separadas por un factor constante k. La función de detección queda, por tanto, de la siguiente forma:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma), \end{aligned}$$

donde L es una imagen suavizada con un filtro de Gauss.

A cada uno de los candidatos extraídos en la etapa anterior se le aplica un modelo para determinar su localización y escala. Para detectar los máximos y mínimos locales de $D(x, y, \sigma)$ cada punto se compara con sus 8-vecinos en la imagen actual y con los 9-vecinos de las imágenes superior e inferior. De este modo se seleccionan únicamente los puntos más estables. Cabe señalar que en esta etapa es posible determinar varias escalas válidas para una misma posición (x, y) de la imagen, por lo que en el resultado final, podremos tener varios descriptores para un mismo punto de la imagen.

Una vez obtenida la escala, se calcula la orientación (u orientaciones) de cada punto de interés a partir de la imagen suavizada L más cercana a la escala obtenida en la etapa anterior. En este paso, para cada imagen $L(x, y)$ con determinada escala se calcula la magnitud del gradiente, $m(x, y)$ y la orientación $\Theta(x, y)$ mediante la diferencia entre píxeles:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

$$\Theta(x, y) = \tan^{-1} \frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}$$

Para conseguir el descriptor del punto, primeramente se obtienen las magnitudes y orientaciones de los vecinos del punto de interés, utilizando la imagen suavizada más cercana a la escala calculada en el segundo paso. Una vez obtenidos, para asegurar la invarianza a la orientación, las coordenadas del descriptor y las orientaciones del gradiente se rotan de forma relativa a la orientación extraída durante la tercera etapa.

Siguiendo estos pasos, el algoritmo logra identificar un gran número de características, lo cual es particularmente importante para el reconocimiento de regiones. Además los descriptores son altamente distintivos, lo que permite que una única característica sea correctamente emparejada con una probabilidad alta en una base de datos de características grande. SIFT obtiene, por tanto, gran cantidad de características robustas y distinguibles entre sí.

B. SURF: Speeded Up Robust Features

SURF es otro de los algoritmos más utilizado en la literatura para la extracción de puntos de interés invariantes. La extracción de los puntos la realiza detectando en primer lugar los posibles puntos de interés y su localización dentro de la imagen. Posteriormente se representa la vecindad del punto de interés como un vector de características (por defecto con un tamaño de 64, aunque es posible aumentarlo).

El algoritmo SURF utiliza una aproximación básica de la matriz Hessiana para reducir el tiempo de computación [6]. La matriz Hessiana se utiliza debido a su buena relación entre la precisión y el coste temporal. El determinante de la Hessiana se utiliza para la localización de los puntos y para la determinación de la escala. Dado un punto $\mathbf{x} = (x, y)$ en una imagen I, la matriz Hessiana $\mathcal{H}(\mathbf{x}, \sigma)$ en $\mathbf{x}$ con escala σ se define como:

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix}$$

donde $L_{xx}(\mathbf{x}, \sigma)$ es la convolución de la derivada de segundo orden de la Gaussiana, $\frac{\delta^2}{\delta x^2} g(\sigma)$ con la imagen I en el punto $\mathbf{x}$, y similarmente para $L_{xy}(\mathbf{x}, \sigma)$ y $L_{yy}(\mathbf{x}, \sigma)$. Para el cálculo del determinante, se realizan aproximaciones a las derivadas de segundo orden de la Gaussiana de modo que se obtienen tres aproximaciones: D_{xx}, D_{xy} y D_{yy}. De esta forma, el determinante de la Hessiana que nos indica la escala del punto se calcula con la siguiente fórmula:

$$det(\mathcal{H}_aprox) = D_{xx}D_{yy} - (0.9D_{xy})^2.$$

El descriptor SURF es relativamente parecido al propuesto en SIFT. El primer paso para obtener el descriptor una vez calculada la escala es el cálculo de la orientación del punto de interés para, posteriormente, calcular el descriptor SURF. Para obtener un punto invariante a la orientación se calcula el "Haar-wavelet" para las direcciones x e y en una región circular de radio $6s$, donde s es la escala del punto de interés. Una vez calculados para todos los vecinos, se estima la orientación dominante calculando la suma de todos los resultados dentro de una ventana deslizante que cubre un ángulo de $\frac{\pi}{3}$. El cálculo del descriptor se realiza construyendo, primeramente, una región cuadrada centrada en el punto de interés y con un tamaño de $20s$. La región se divide regularmente en 4 sub-regiones, y para cada sub-región se calculan unas pocas características simples. Seguidamente se calculan la "Haar-wavelet" para x e y y se suavizan los resultados mediante una Gaussiana, obteniendo d_x y d_y. Para cada sub-región se suman los resultados d_x y d_y, además de calcularse su valor absoluto $|d_x|$ y $|d_y|$. De este modo, cada sub-región proporciona un vector $\mathbf{v}$ compuesto por: $\mathbf{v} = (\Sigma d_x, \Sigma d_y, \Sigma|d_x|, \Sigma|d_y|)$. El descriptor SURF se obtiene mediante la unión de los vectores de las sub-regiones.

La característica principal de los puntos de interés SURF es la repetibilidad, si el punto es considerado fiable el detector encontrará el mismo punto bajo distintos puntos de vista (diferente escala, orientación, etc) [6]. El vector obtenido por el algoritmo deberá ser distintivo y al mismo tiempo robusto al ruido, errores, y deformaciones geométricas y fotométricas.

C. Comparativa entre SIFT y SURF

Tanto SIFT como SURF obtienen puntos de interés de la imagen invariantes a escala y orientación, así como a cambios en iluminación. Ambos obtienen un vector descriptor por cada punto de interés, aunque calculados de distinta manera.

Las principales diferencias en los datos extraídos es que en SIFT se guarda la posición, la escala y la orientación, puesto que es posible que en una misma posición (x, y) encontremos varios puntos de interés con distinta escala s y/u orientación σ. En SURF, en cambio, en una misma posición (x, y) solamente aparece un único punto de interés, por lo que no guarda la escala y la orientación, aunque sí registra la matriz de segundo momento y el signo de la laplaciana.

Durante las pruebas realizadas, se le ha indicado a SURF que obtenga un descriptor con un tamaño de 128, el mismo tamaño que obtiene SIFT, puesto que el tamaño por defecto para este vector es de 64 (tamaño donde se obtiene una buena relación coste-precisión). Además, ambos algoritmos trabajan con imágenes de tipo *PGM*, es decir, trabajan con imágenes en escala de grises. De este modo, es posible comparar los dos algoritmos partiendo de la misma base.

Para la comparación entre estos dos algoritmos sobre su capacidad de obtención de características, se han tenido en cuenta tanto el tiempo que tardan en ejecutarse como el número de características invariantes obtenidas. Para SIFT se ha utilizado la versión 4 obtenida de la página web del autor, la versión de SURF es la 1.0.9, también obtenida de la página del autor. Ambas versiones están escritas en el lenguaje de programación C. Las pruebas se han realizado sobre un ordenador *Intel Core 2 Duo a 2.0 GHz*. Las imágenes se capturaron mediante una cámara omnidireccional.

Durante las pruebas se ha podido constatar que la implementación del algoritmo SURF es mucho más rápido que SIFT. En cuánto al número de puntos invariantes detectados SIFT supera a SURF con más del doble de puntos detectados. Sin embargo, debemos tener en cuenta las características especiales de SIFT a la hora de extraer conclusiones. SURF no permite que haya varios puntos invariantes en una misma posición con distinta escala y/u orientación, mientras que en SIFT sí que es posible (y habitual). Este hecho nos hace pensar que la distancia entre el número de características detectada disminuiría si SIFT no "duplicase" puntos.

En la tabla I tenemos una muestra de los datos obtenidos durante la experimentación. Puesto que las capturas se realizaron en el exterior de un edificio rectangular, los ejemplos tomados han sido las cinco primeras imágenes de cada una de las cuatro fachadas.

Imagen	Puntos SIFT	Puntos SURF	Tiempo SIFT (ms)	Tiempo SURF (ms)
Fachada 1				
imagen0001	958	397	1528	420
imagen0002	1046	405	1540	430
imagen0003	1251	451	1656	466
imagen0004	1395	499	1712	543
imagen0005	1408	468	1808	463
...				
Fachada 2				
imagen0029	1262	491	1632	453
imagen0030	1292	431	1668	450
imagen0031	1285	433	1596	405
imagen0032	1528	529	1736	476
imagen0033	1399	493	1708	476
...				
Fachada 3				
imagen0056	1388	598	1680	605
imagen0057	1027	420	1532	388
imagen0058	873	389	1408	364
imagen0059	921	410	1488	440
imagen0060	1616	470	1868	455
...				
Fachada 4				
imagen0082	1250	459	1600	424
imagen0083	1149	449	1556	424
imagen0084	1154	460	1624	409
imagen0085	1196	478	1560	478
imagen0086	1134	456	1604	618
...				

TABLE I
MUESTRA DE ALGUNOS DE LOS RESULTADOS OBTENIDOS CON SIFT Y SURF (NÚMERO DE PUNTOS OBTENIDOS Y TIEMPO NECESITADO EN MILISEGUNDOS).

De una muestra de 109 imágenes en total, hemos obtenido una media de puntos invariantes detectados por SIFT de $MediaPuntos_{SIFT} = 1292.01$ y por SURF $MediaPuntos_{SURF} = 482.69$. Por lo tanto tenemos que la relación entre los dos algoritmos es que SIFT obtiene aproximadamente 2.68 puntos más que SURF.

Por otro lado, hemos visto que SURF es más rápido que SIFT. La media de tiempo utilizado por SIFT para una imagen es de $MediaTiempo_{SIFT} = 1646.53ms$ mientras que SURF utiliza un promedio de $MediaTiempo_{SURF} = 485.77ms$. Si calculamos la relación entre el tiempo utilizado por los dos algoritmos tenemos que: $\frac{Tiempo_{SIFT}}{Tiempo_{SURF}} \approx 3.39$. Como se puede observar, aunque SIFT obtiene alrededor de 2.68 puntos más que SURF, tarda en hacerlo 3.39 tiempo más que SURF. Este dato es muy importante a la hora de decidir si lo que queremos para nuestro algoritmo es una mayor cantidad de datos sin importar tanto el tiempo utilizado para el cálculo (en ese caso escogeríamos SIFT) o si por el contrario preferimos un algoritmo más veloz a costa de una reducción del número de características detectadas (SURF).

Los datos extraídos anteriormente son de ayuda para efectuar una primera aproximación a los dos algoritmos. No obstante, no se puede medir la bondad de los algoritmos solamente por la cantidad de puntos detectados o por el tiempo empleado en ello. Es importante determinar la "calidad" de los puntos invariantes detectados, es decir, necesitamos saber si esas características podrán ser detectadas (emparejadas) en otra imagen distinta a la inicial. De ahí surge la necesidad de comparar los emparejamientos de los puntos invariantes extraídos por SIFT y SURF.

El emparejamiento se ha calculado mediante el "match" que venía en el detector (realizado por los propios autores). A continuación vamos a estudiar cómo se comportan los dos algoritmos de extracción de características durante una secuencia determinada. En este caso, se comparará la primera imagen (imagen0001) con las imágenes extraídas inmediatamente después y con las extraídas justo al final del experimento para

comprobar el cierre del bucle [1].

En primer lugar, cabe señalar que los dos algoritmos de emparejamiento obtienen resultados diferentes dependiendo del orden de las dos imágenes a comparar, es decir, no se obtiene el mismo resultado al comparar el par 007-001 que el 001-007. En la tabla II encontramos la comparación entre los emparejamientos realizados por SIFT y SURF para las imágenes del cierre del bucle y para las iniciales. Como podemos observar para las imágenes más cercanas a la comparada (imagen001) SIFT localiza más puntos en común, mientras que SURF encuentra más puntos cuando las imágenes comparadas se alejan de la inicial.

Par de imágenes	Emparejamientos SIFT	Emparejamientos SURF
103-001	5	6
104-001	9	8
105-001	12	14
106-001	23	23
107-001	42	51
108-001	130	110
109-001	213	159
002-001	416	245
003-001	139	135
004-001	55	54
005-001	22	25
006-001	18	17
007-001	15	10
008-001	11	6

TABLE II
EMPAREJAMIENTOS REALIZADOS CON LA IMAGEN0001 COMO BASE. LAS IMÁGENES DESDE LA 103 A LA 109 SE CORRESPONDEN CON LAS DEL CIERRE DEL BUCLE. LAS IMÁGENES DESDE LA 002 A LA 008 SON LAS IMÁGENES INICIALES.

Fig. 1. Emparejamiento SIFT para el par de imágenes 103 con 001.

Fig. 2. Emparejamiento SURF para el par de imágenes 103 y 001.

[1]Se considera que las imágenes con una menor numeración ya han sido procesadas y, por tanto, pertenecen a la base de datos.

Si nos fijamos en los datos aportados por el par de imágenes 103-001, veremos que SIFT ha localizado 5 puntos mientras que SURF ha encontrado 6 puntos en común, pero si observamos los emparejamientos en la imagen, los datos cambian notablemente. Como podemos ver en la imagen 1 los puntos emparejados son correctos, es decir, no hay ningún emparejamiento de puntos que no son el mismo punto en el entorno real. Sin embargo, el emparejamiento realizado por SURF (figura 2), los puntos emparejados son falsos positivos, es decir, en el entorno los seis puntos de la imagen 103 no son los mismos que los seis de la imagen inicial. A lo largo de la experimentación se ha comprobado que estos resultados se repiten con otras imágenes diferentes. Por todo ello, podemos afirmar que el algoritmo SIFT tiene mayor perdurabilidad en el tiempo que SURF, obteniendo emparejamientos más fiables.

Como se ha podido comprobar, SURF obtiene falsos positivos cuando la imagen comparada se aleja de la imagen base. Además, tanto en SURF como en SIFT podemos encontrar algunos emparejamientos que, si se comparasen con el resto de emparejamientos se podrían eliminar, puesto que son los denominados emparejamientos cruzados. Un ejemplo de estos falsos positivos se pueden ver en la imagen 3 pertenecientes al par de imágenes 109-001 y el algoritmo SIFT.

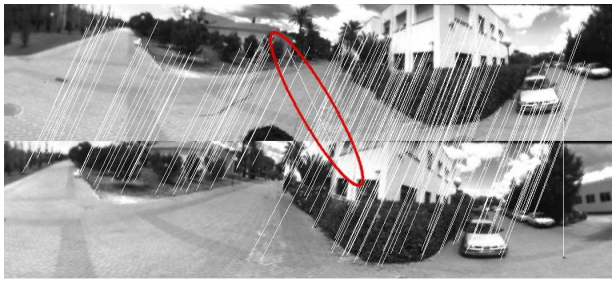

Fig. 3. Emparejamiento SIFT del par 109-001. Señalado: Falso positivo.

III. TRATAMIENTO DE LAS IMÁGENES: SEGMENTACIÓN Y EXTRACCIÓN

A pesar de la robustez de los puntos invariantes, es posible encontrarse con que algunos puntos claramente diferentes han sido emparejados como el mismo punto debido a que la distancia euclídea entre ambos descriptores se encontraba por debajo del umbral establecido. No obstante, cuando se da este caso el resto de puntos invariantes "vecinos" puede que no compartan dicha similitud. Es para estos casos para los que surge la necesidad de la utilización de los grafos como la estructura a comparar.

Cada uno de estos grafos puede representar la imagen completa o una parte de ella. En nuestro caso se compararán los distintos subgrafos de las imágenes para obtener cuál de las imágenes pertenecientes a la base de datos es la más cercana a la imagen actual. Para obtener estos subgrafos, es necesario realizar algún tipo de segmentación de modo que obtengamos cuáles son las regiones a tener en cuenta. El proceso que se sigue para obtener las distintas regiones es el siguiente:

1) Aplicar a cada imagen el algoritmo K-Means.

2) Para cada color distinto detectado en el punto anterior obtener su correspondiente binarización.
3) Calcular la lista de Blobs para todas las imágenes binarizadas.
4) Determinar el conjunto de puntos invariantes que pertenecen a cada Blob.

El primer paso consiste en aplicar a la imagen el algoritmo K-Means. Este algoritmo calcula los k colores medios de toda la imagen y asigna a cada píxel su media más cercana. Así es posible obtener una imagen donde solamente hay k colores distintos. En la imagen 4 podemos ver un ejemplo de imagen a la que se le ha aplicado una $k = 6$.

Fig. 4. K-Means con $k = 6$ aplicado a la imagen 1.

Una vez obtenidas estas medias, se binariza la imagen k veces, cada una de ellas con una media de colores distinta. Una vez binarizadas tendremos k imágenes binarias. Cada una de estas imágenes representa a uno de los colores obtenidos por el algoritmo "K-Means".

Una vez llegados a este punto podríamos considerar que cada una de las imágenes binarizadas representa una región distinta, pero observando algunas de las imágenes podemos ver que en este paso todavía podemos encontrar regiones diferenciadas (regiones que tienen el mismo color pero que se encuentran lejos entre si). Para evitar estas situaciones y obtener todas las regiones de la imagen, se aplica el algoritmo Blob a cada una de las imágenes binarizadas, obteniendo una lista de blobs para cada imagen. Cada blob contiene una lista de puntos de la imagen (x, y) que pertenecen a determinada región. Utilizando estos datos, será posible determinar a qué región pertenece cada uno de los puntos invariantes extraídos utilizando cualquiera de los dos algoritmos anteriormente descritos.

IV. Creación de grafos: GTM

Después de aplicar todas las etapas anteriores obtenemos una lista donde cada elemento contiene un conjunto de pares de puntos emparejados. Dichos elementos de la lista se corresponden con una región en común entre las dos imágenes comparadas. Este emparejamiento puede dar lugar a numerosos falsos positivos que deberán ser eliminados del resultado final. Para ello compararemos la estructura de la región en las dos imágenes eliminando aquellos puntos que "deformen" el emparejamiento utilizando el algoritmo GTM (Graph Tranformation Matching).

GTM es un algoritmo de emparejamiento de puntos basado en grafos atribuidos [7] que utiliza la información de la estructura local (grafo) para el tratamiento de falsos positivos (o "outliers").

El grafo que utiliza el algoritmo se construye añadiendo una arista en la matriz de adyacencia para el par (i, j) si el nodo j es uno de los k (siendo k un número natural estimado empíricamente) vecinos más cercanos del nodo i y si además la distancia entre ambos puntos (distancia euclídea calculada a partir de sus posiciones en la imagen) es menor a la media de todas las distancias entre todos los nodos del grafo:

$$\eta = \texttt{median}_{(l,m)\in V_p x V_p} ||p_l - p_m||$$

Si un nodo no tiene k aristas entonces se desconecta completamente hasta finalizar la construcción del grafo.

Una vez construidos los dos grafos (uno por cada imagen) el algoritmo elimina iterativamente las correspondencias que distorsionan las relaciones de vecindad. Para ello, se selecciona un "outlier", se eliminan los dos nodos que forman el emparejamiento (falso positivo) de sus grafos respectivos así como las referencias en las matrices de adyacencia a dichos nodos y se vuelven a recalcular los dos grafos. El proceso sigue hasta que la matriz residual (la diferencia entre las matrices de los dos grafos) es nula. En este momento se considera que el algoritmo ha encontrado un grafo de consenso.

Algorithm GTM {
  $dist_p \leftarrow$ `computeDistanceMatrix(`P`)`
  $dist_{p'} \leftarrow$ `computeDistanceMatrix(`P'`)`
  $median_p \leftarrow$ `computeMedian(`P`)`
  $median_{p'} \leftarrow$ `computeMedian(`P'`)`
  $Q = P$
  $Q' = P'$

  $A_p \leftarrow$ `buildMedianKNNGraph(`$dist_p, K, median_p$`)`
  $A_{p'} \leftarrow$ `buildMedianKNNGraph(`$dist_{p'}, K, median_{p'}$`)`
  `While(`$A_p != A_{p'}$`)`
    $out \leftarrow$ `findOutlier(`$A_p, A_{p'}$`)`
    `Remove` out `from` $dist_p$
    `Remove` out `from` $dist_{p'}$
    `Remove` out `from` Q
    `Remove` out `from` Q'
    $A_p \leftarrow$ `buildMedianKNNGraph(`$dist_p, K, median_p$`)`
    $A_{p'} \leftarrow$ `buildMedianKNNGraph(`$dist_{p'}, K, median_{p'}$`)`
  `End While`
  $Q =$ `removeDisconnetedVertices(`A_p`)`
  $Q' =$ `removeDisconnetedVertices(`$A_{p'}$`)`
}

outlier findOutlier($A_p, A_{p'}$**)** {
  $R \leftarrow$ `abs(`$A_p, A_{p'}$`)`
  $j^{outlier} \leftarrow argmax_{j=1..N} \sum_{i=j}^{N} R(i,j)$
}

Fig. 5. Pseudo-código del algoritmo GTM

En la figura 5 podemos ver el pseudo-código del algoritmo. La segunda función calcula la disparidad estructural mediante la matriz residual de adyacencia $R = |A_p - A_{p'}|$. Una vez calculada selecciona la columna j^{out} que tiene la mayor diferencia de aristas.

$$j^{out} = argmax_{j=1..N} \sum_{i=1}^{N} R(i,j)$$

El "outlier" seleccionado es el par $(nodo_{j_{out}}, nodo'_{j_{out}})$

Una vez obtenido el grafo de consenso, se eliminan del emparejamiento inicial todos los nodos desconectados, obteniendo un emparejamiento donde se han eliminado los falsos positivos.

V. RESULTADOS

Durante esta sección mostraremos los resultados obtenidos cuando aplicamos todos los pasos anteriores a una serie de imágenes del mundo real. Estas imágenes representan un recorrido (con cierre de ciclo, es decir finaliza en el punto de inicio) por el exterior del edificio de "Servicio de Informática" que se encuentra dentro del campus de la Universidad de Alicante (imagen 6).

Fig. 6. Recorrido realizado durante la captura de imágenes.

Debido a que alguno de los algoritmos necesita definir el valor de determinados parámetros, se han realizado algunos experimentos previos para encontrar aquéllos que obtenían mejores resultados. Los valores seleccionados para cada uno de los algoritmos son: el umbral por debajo del cuál se considera que dos puntos SIFT son iguales es de 150, el de SURF de 0.5, el número de colores a identificar para K-Means es de 6 y el número mínimo de vecinos para la construcción del grafo del GTM es de 4.

Al igual que la comparación entre los dos algoritmos de extracción de características invariantes, en esta sección vamos a tener en cuenta varios factores para poder determinar qué algoritmo es el más adecuado para cada situación.

En primer lugar, comprobaremos el número de emparejamientos realizados por el algoritmo GTM cuando se utilizan los puntos SIFT y los puntos SURF. Los umbrales seleccionados para estos dos algoritmos son valores compromiso, es decir, son valores que obtienen un gran número de emparejamientos a costa de algunos falsos positivos. Este hecho será importante, sobre todo, en la segunda parte de la experimentación.

En el gráfico 7 se muestra el resultado de comparar la imagen 1 con todas las imágenes de la base de datos. La imagen final (109) se considera como cierre de ciclo, es decir, se tomó en un punto cercano al de la imagen 1, por lo que se puede considerar que representa la misma escena. El porcentaje de emparejamientos nos indica la relación entre todos los puntos emparejados de todos los objetos con el número de puntos que tiene la imagen en total. Como podemos observar, utilizando los puntos SURF en la imagen 1 se obtiene un mayor porcentaje de emparejamientos, es decir, el número de puntos emparejados con respecto al total es mayor utilizando los puntos SURF cuando las imágenes son cercanas. No obstante, para imágenes alejadas (imágenes que no comparten la misma escena) SURF también detecta un mayor porcentaje de puntos invariantes, cuando debería emparejar menos. El algoritmo ideal sería aquél que nos permitiese tener un alto porcentaje en las imágenes cercanas a la comparada y un bajo porcentaje (o nulo) en las que están más alejadas. En nuestro caso, SURF cumple el primer requisito y SIFT cumple el segundo.

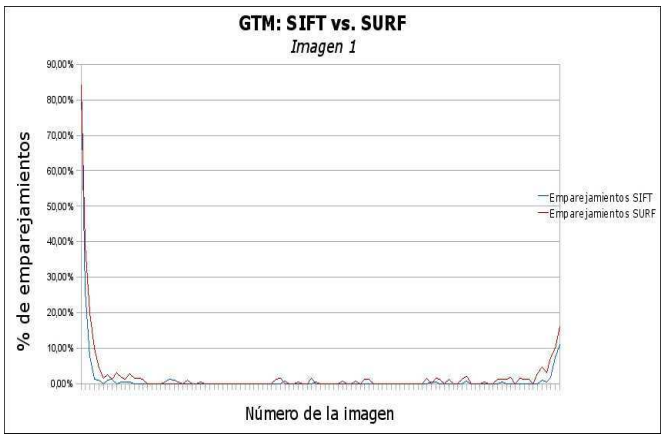

Fig. 7. Porcentaje de emparejamientos de la imagen 1 con el resto de imágenes de la base de datos. En azul los emparejamientos obtenidos mediante SIFT-GTM, en rojo SURF-GTM.

Si nos fijamos en el valor inicial del gráfico 7 (el valor correspondiente a comparar la imagen 1 consigo misma) podemos ver que el porcentaje de emparejamientos está entre el 80 y el 85% tanto para SIFT-GTM como para SURF-GTM. Este hecho es debido a varios motivos: el primero es que las regiones pequeñas (con pocos píxeles) no son tenidas en cuenta, por lo que si algún punto característico estuviese en esa región no se comparará; el segundo motivo es que cada región se representa mediante un grafo GTM, pero se compara con el grafo GTM completo, por lo que algunos nodos vecinos en el grafo de toda la imagen pueden afectar a la estructura del mismo y algunos de los emparejamientos se eliminan durante el proceso de obtención del grafo de consenso. También es posible observar que el porcentaje de emparejamientos decrece cuánto más alejada esté la imagen. Este es un punto significativo, puesto que lo que se pretende conseguir es la identificación de una zona del mapa, y este descenso nos indica que la comparación entre imagen y zona es posible, puesto que los resultados son los que cabe esperar en esta situación.

Al contrario que en el gráfico 7 dónde las imágenes comparadas tenían una gran cantidad de puntos invariantes emparejados, en el gráfico 8 nos encontramos con el resultado de la comparación de la imagen 32, una imagen donde hay grandes superficies sin textura por lo que tanto SIFT como SURF no producen buenos emparejamientos.

La zona que representa la imagen 32 es una zona sin textura, por lo que tanto esa imagen, como las más cercanas a ella son difíciles de emparejar, ya que un mismo punto en una imagen podría corresponderse a muchos en la otra (todos los puntos vecinos son prácticamente iguales). En el gráfico correspondiente a dicha imagen (figura 8) podemos ver que

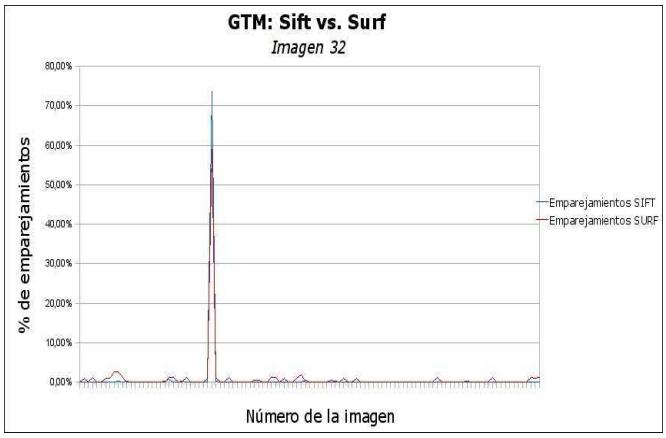

Fig. 8. Porcentaje de emparejamientos de la imagen 32 con el resto. En azul podemos ver los emparejamientos obtenidos cuando GTM utiliza los puntos SIFT, en rojo los SURF.

para la comparación de la imagen 32 consigo misma, con SIFT se emparejan aproximadamente un 74% de los puntos mientras que SURF alcanza únicamente el 59%. Además, ninguno de los dos algoritmos combinado con GTM considera que las imágenes anterior y posterior (31 y 33) pertenecen a la misma escena. En el caso de SIFT el porcentaje de puntos emparejados entre estas imágenes y la 32 es muy bajo y aunque el porcentaje obtenido por SURF es mayor, no lo es lo suficiente como para considerarlas parte de la misma escena. Asimismo, en el caso de SURF, se considera que las imágenes 9 y 10 son más parecidas (2.65%) a la 32 que las vecinas reales. Como podemos ver, para una imagen con pocas características invariantes que además son muy parecidas no se consigue un buen número de emparejamientos aún aplicando GTM.

Utilizando únicamente los datos estadísticos podríamos concluir que la combinación de SURF-GTM es mejor que la de SIFT-GTM. No obstante, es necesario comprobar si los emparejamientos obtenidos son o no buenos. Para ello vamos a observar pares de imágenes emparejadas y ver cuántos puntos se han detectado correctamente y cuántos son falsos positivos.

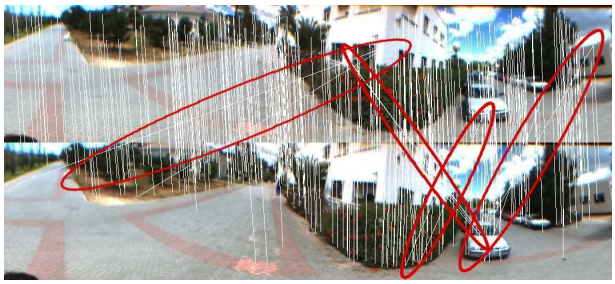

Fig. 9. Emparejamiento entre la imagen 2 (arriba) y la 1 (abajo). El emparejamiento utiliza únicamente KD-Tree con el detector SIFT, cada punto invariante de la imagen 2 está emparejado con su vecino más cercano (si está por debajo del umbral) de la imagen 1. Los "outliers" o falsos positivos están marcados por la señal roja.

La imagen 9 representa los emparejamientos de la imagen 2 con la imagen 1 sin tener en cuenta la estructura del grafo ni tampoco la segmentación de regiones. En este caso se trata del emparejamiento realizado mediante la estructura KD-Tree de los puntos invariantes SIFT de una imagen con los de otra. Para cada punto de la imagen 2, se seleccionaba el punto más cercano de la imagen 1 y si la diferencia entre sus vectores descriptivos no superaba el umbral se mantenía el emparejamiento. Señalados en rojo podemos ver todos los falsos positivos ("outliers") que se generan al realizar el emparejamiento de este modo. En la imagen 10 tenemos el mismo tipo de emparejamiento pero utilizando los descriptores obtenidos mediante el algoritmo de SURF. Si comparamos ambos resultados observamos que con SURF se han detectado más "outliers" que con SIFT. Este hecho se repite en la gran mayoría de las imágenes y tendrá mucho peso durante el emparejamiento utilizando GTM.

Fig. 10. Emparejamiento entre la imagen 2 (arriba) y la 1 (abajo). El emparejamiento se ha realizado mediante la estructura KD-Tree y los descriptores SURF. En rojo están señalados los "outliers" producidos.

Los "outliers" o falsos positivos aparecidos en las imágenes 9 y 10 son precisamente los emparejamientos que queremos evitar. Para ello vamos a tratar de eliminarlos mediante la segmentación de imágenes en distintas regiones y el uso de una estructura de grafo durante la comparación (algoritmo GTM).

En la imagen 11 tenemos el emparejamiento de la imagen 2 con la 1 utilizando los descriptores proporcionados por SIFT junto con GTM. Para obtener este resultado, se han comparado los grafos de cada una de las regiones detectadas en la imagen 2 con el grafo que describe a la imagen 1 al completo. Si comparamos esta imagen con la obtenida en 9 veremos que los falsos positivos han desaparecido por completo. Además, no se han eliminado demasiados emparejamientos válidos, por lo que el resultado nos proporciona una buena base para el reconocimiento de la escena. Si ahora nos fijamos en la imagen 12 observamos que también es este caso se han eliminado los "outliers", pero, a cambio hemos perdido más emparejamientos válidos. Si recordamos los datos del gráfico 7 SURF tenía un porcentaje de emparejamiento mayor que SIFT. Analizando todos los resultados en conjunto tenemos que aunque SURF tiene un mayor porcentaje de emparejamientos, SIFT continúa teniendo un mayor número de emparejamientos en total, además de no producir tantos falsos positivos durante la primera fase. Aunque en un primer momento (contemplando únicamente el gráfico) podríamos pensar que los resultados se SURF para las imágenes pertenecientes a una misma escena eran mejores, al examinar las imágenes con los resultados podemos ver que SIFT obtiene más emparejamientos con mayor precisión.

Fig. 11. Imagen 2 (arriba) emparejada con la 1 (abajo) utilizando el algoritmo GTM y los descriptores SIFT.

Fig. 12. Emparejamiento de la imagen 2 (arriba) con la 1 (abajo) mediante el uso del descriptor SURF y el algoritmo GTM.

VI. Conclusiones

Como hemos podido observar durante la experimentación, el algoritmo SURF es ligeramente más rápido que SIFT. No obstante, si nos fijamos en el número de características detectadas, claramente SIFT supera a SURF (extrayendo más del doble de puntos). Sobre este aspecto, cabe señalar que mientras SURF únicamente calcula un vector descriptor para una determinada posición (x, y), SIFT puede calcular varios descriptores (con distinta escala y orientación) para una misma posición de la imagen. Respecto a los algoritmos "match" SIFT emparejaba más nodos que SURF pero no el doble, como era de esperar teniendo en cuenta que SIFT detecta casi el doble de puntos. No obstante SURF detecta un mayor número de falsos positivos, mientras que los emparejamientos SIFT eran más confiables.

La elección de uno u otro algoritmo dependerá de las características buscadas. Seleccionaremos SURF para detectar los puntos con una mayor velocidad y SIFT para detectar puntos más robustos con un coste temporal más alto.

Respecto a la utilización de grafos como estructura a comparar, hemos observado que se obtienen mejores resultados si se comparan las regiones de una imagen con la imagen completa y no las imágenes completas o las regiones entre si. En estos casos los "outliers" han desaparecido aunque también los emparejamientos han sufrido un descenso. Esto es en parte debido a que durante la etapa de segmentación aparecían regiones con pocos puntos (de 1 a 5 puntos) y las regiones con menos de 6 puntos no se comparan ya que GTM ni siquiera podría construir el grafo.

Los resultados obtenidos por GTM han sido distintos dependiendo si el emparejamiento se realizaba usando puntos SIFT o SURF. Como se ha visto en los gráficos, SURF obtiene un mayor porcentaje de emparejamientos tanto en las imágenes cercanas como en las que estaban alejadas en cambio SIFT produce un menor porcentaje en ambos casos. El resultado ideal de estos gráficos sería aquel que tuviese una curva en forma de campana de Gauss, donde el valor más alto perteneciese a la imagen comparada consigo misma y los valores fuesen decreciendo de forma constante. En nuestro caso, se ha obtenido una curva donde el punto más alto es el perteneciente a la comparación de la imagen con si misma. A partir de este punto los emparejamientos decaen hasta llegar prácticamente a 0, aunque no lo hacen tan suavemente como en una campana de Gauss. No obstante, que la curva no esté tan suavizada no significa que no sea un buen resultado, puesto que cumple su función, es decir, observando el gráfico es posible determinar a qué escena pertenece la imagen en cuestión buscando entre aquéllas imágenes donde el porcentaje de los emparejamientos empieza a aumentar.

A pesar de que SURF conseguía un mayor porcentaje de emparejamientos, hemos visto que SIFT obtenía unos resultados más fiables y con menos "outliers". Además, como el algoritmo SIFT extrae una mayor cantidad de puntos, el número de emparejamientos aumenta considerablemente en las imágenes cercanas. En cuanto a las imágenes alejadas SURF obtiene muchos falsos positivos que GTM no puede eliminar del todo. Como en la construcción de los dos grafos (el de la región de la imagen a comparar y el de la imagen de la base de datos) se añaden todos los puntos emparejados, si existe un gran número de emparejamientos malos es posible encontrar un grafo de consenso (mucho más pequeño que el de la imagen inicial) que cumpla con todas las restricciones pero que contenga mayormente esos falsos emparejamientos puesto que la gran cantidad que había inicialmente no permite a GTM eliminarlos por completo.

Por último cabe destacar los buenos resultados obtenidos en comparación con los algoritmos "match" de SIFT y SURF. En nuestro caso, conseguimos eliminar muchos de los falsos positivos que dichos algoritmos no son capaces de descartar.

Acknowledgment

Este trabajo ha sido soportado con la beca JC08-00077 del Ministerio de Ciencia e Innovación.

References

[1] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *Int. J. of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[2] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous Robot Vehicles* (I. J. Cox and G. T. Wilfong, eds.), pp. 167–193, Springer-Verlag, 1990.

[3] S. Julier and J. K. Uhlmann, "A counter example to the theory of simulataneous localization and map building," in *ICRA*, pp. 4238–4243, IEEE, 2001.

[4] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *AAAI/IAAI*, pp. 593–598, 2002.

[5] D. G. Lowe, "Object recognition from local scale-invariant features," in *ICCV*, pp. 1150–1157, 1999.

[6] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *In ECCV*, pp. 404–417, 2006.

[7] W. Aguilar, Y. Frauel, F. Escolano, M. E. Martínez-Pérez, A. Espinosa-Romero, and M. Ángel Lozano, "A robust Graph Transformation Matching for non-rigid registration," in *Image Vision Computing*, 2008.