Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Maestría en Ciencias
en ciencias de la computación

Índices de espigas de grafos geométricos planos

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de

Maestro en Ciencias

Presenta:

José Luis Galaviz Ortiz

Ensenada, Baja California, México

2018

Tesis defendida por

José Luis Galaviz Ortiz

y aprobada por el siguiente Comité

Dr. Edgar Leonel Chávez González

Director de tesis

Dr. J. Apolinar Reynoso Hernández

Dra. Mónica Elizabeth Tentori Espinosa

Dr. Jesús Favela Vara

Coordinador del Posgrado en ciencias de la computación

Dra. Rufina Hernández Martínez

Directora de Estudios de Posgrado

Resumen de la tesis que presenta José Luis Galaviz Ortiz como requisito parcial para la obtención del grado de Maestro en Ciencias en ciencias de la computación.

Índices de espigas de grafos geométricos planos

Resumen aprobado por:

Dr. Edgar Leonel Chávez González
Director de tesis

Identificar personas por sus huellas dactilares o su iris, identificar estrellas en una fotografía o identificar canciones son problemas que se pueden resolver utilizando una representación de nubes de puntos. Construir índices de nubes de puntos permite resolver estos problemas de identificación eficientemente. Hay dos problemas principales en la identificación de nubes de puntos, la oclusión y el ruido. En la literatura se han construido índices que o toleran ruido o toleran oclusión. En este trabajo se construye el primer índice que simultáneamente tolera ruido y oclusiones, permitiendo al mismo tiempo consultas rápidas. De acuerdo a los resultados experimentales el índice construido presenta una tolerancia de hasta un 20 % de oclusión y hasta 5 pixels de ruido, las consultas se pueden realizar en 280 milisegundos para una base de datos de 1000 nubes de 300 puntos.

Palabras clave: espigas, nubes de puntos, similitud, métrica, índice, cadenas binarias

Abstract of the thesis presented by José Luis Galaviz Ortiz as a partial requirement to obtain the Master of Science degree in Computer Science.

Spike Indexes for geometric planar graphs

Abstract approved by:

Dr. Edgar Leonel Chávez González
Thesis Director

Identifying a person by her fingerprints or her iris, stars in a sensor or songs in a recording are problems that can be solved using a point cloud representation. Building a point cloud index allows solving the identification problem more efficiently. There are two main problems on point cloud identification, occlusion, and noise. In the literature, we can found indices with tolerance to occlusion or noise. In this work, the first index that simultaneously tolerates noise and occlusions is built, allowing simultaneously fast query times. According to the experimental results, the index we propose can stand a tolerance up to 20 % for occlusion and up to 5 pixels of noise; query time is 280 milliseconds for a database of 1000 point clouds with 300 points each.

Keywords: spikes, point clouds, similarity, metric, index, binary strings

Dedicatoria

A mi madre, por todo el amor y apoyo a sus hijos.

Agradecimientos

Al Centro de Investigación Científica y de Educación Superior de Ensenada por darme la oportunidad para realizar los estudios y el excelente ambiente con el que se trabaja.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme el apoyo económico para realizar mis estudios de maestría. No. de becario: 795047

A mi director de tesis, el Dr. Edgar Chávez , gracias por todo el apoyo, tiempo y paciencia para guiarme todas las veces que perdí el camino durante el desarrollo de este trabajo de tesis.

A los miembros de mi comité de tesis, la Dra. Mónica Elizabeth Tentori Espinosa y el Dr. J. Apolinar Reynoso Hernández, por sus consejos y comentarios realizados durante el desarrollo de esta tesis.

A todo el departamento de ciencias de computación, por la calidad de sus cursos y por el gran ambiente que permite despejarse cuando se necesita.

A mis compañeros de ciencias de datos, por las largas charlas y pláticas amenas.

A mis compañeros de generación, por todo el apoyo durante estos dos años y todos esos días de convivencia que alegran el alma.

A mi familia, por todo el apoyo y consejos brindados sin importar la distancia.

Tabla de contenido

Página

Tabla de contenido (continuación)

Lista de figuras

Figura Página

Lista de figuras (continuación)

Lista de figuras (continuación)

Figura Página

Lista de tablas

Tabla Página

Capítulo 1. Introducción

Una gran cantidad de problemas computacionales se pueden representar por conjuntos de puntos. Algunos ejemplos son el reconocimiento facial, reconocimiento dactilar, reconocimiento de retina o la recuperación de imágenes. En todos estos problemas los objetos presentan características únicas que permiten compararlos o diferenciarlos. Utilizando la información espacial cada característica estará representada por la ubicación donde se encuentra.

Por dar un ejemplo, en el reconocimiento dactilar la huella de los dedos esta formada por pliegues en la piel conocidos como elevaciones y valles. Las huellas presentan características globales y características locales. Una característica global es la forma que presenta la huella. Está forma es llamada singularidad o región singular, y se puede clasificar en *delta*, *loop* o *whorl*(Zaeri, 2011), en la *figura 1* se muestran ejemplos de formas presentes en las huellas dactilares. Las características globales se suelen utilizar cuando se buscan huellas de manera aproximada, huellas con la misma forma se considerarán similares. Las características locales son llamadas *minutiae* y son utilizadas cuando se buscan huellas de manera exacta. Minutiae significa "*pequeño detalle*" en el contexto de huellas dactilares (Zaeri, 2011). Las minutiae se definen por las discontinuidades presentes en las elevaciones. En la *figura 2* se muestran las minutiae que se utilizan. Una vez que se localizan las minutiae y utilizando su información espacial se puede representar la huella por las coordenadas (x, y) donde se encuentran, de esta manera la huella pasa a ser representada por un conjunto de puntos.

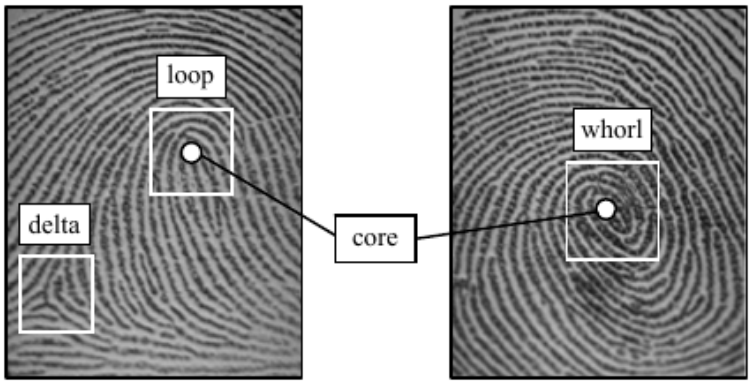

Figura 1. Regiones singulares(cuadros blancos) y puntos centrales(círculos pequeños) en imágenes de huellas dactilares (Maltoni *et al.*, 2009a), Fig. 3.2

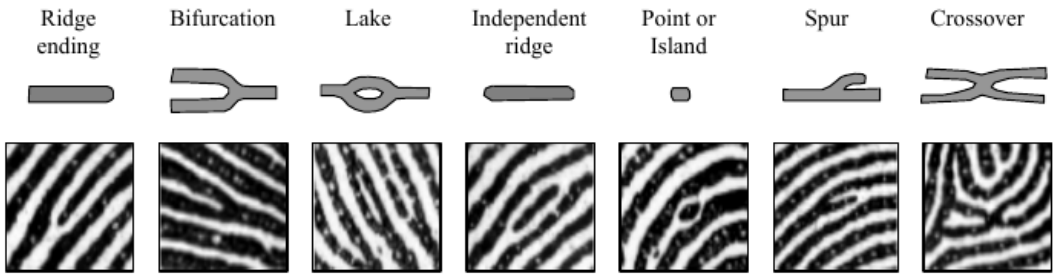

Figura 2. Tipos de minutiae más comunes (Maltoni *et al.*, 2009a), Fig. 3.4

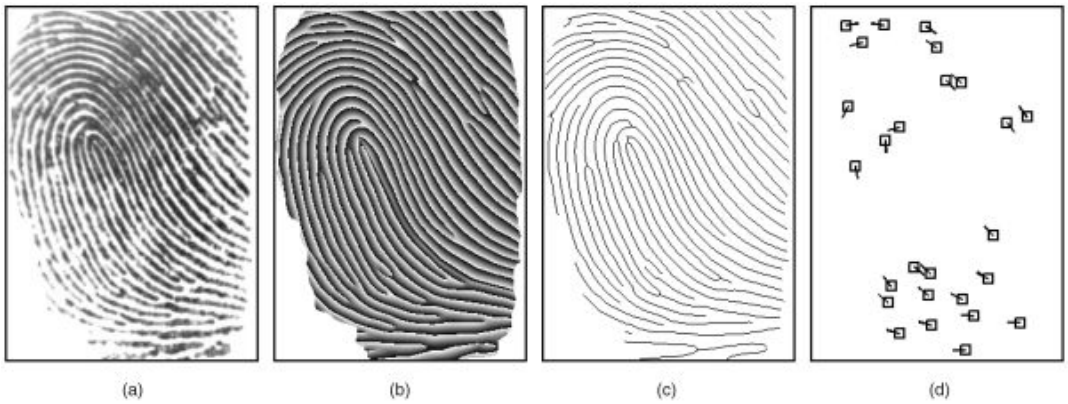

Figura 3. Esquema de representación de huella dactilar (Zaeri, 2011), Fig. 1

Los conjuntos de puntos se utilizan con dos enfoques, identificar o verificar. Al identificar, un conjunto de puntos se compara contra una colección de conjuntos de puntos con el objetivo de determinar si se encuentra dentro de la colección. Al verificar, dos conjuntos de puntos son comparados con el objetivo de determinar si son el mismo. Para ambos casos es necesario determinar la similitud que existe entre dos conjuntos de puntos. Para determinar la similitud se emparejan los puntos que son iguales en ambos conjuntos, a mayor cantidad de puntos emparejados mayor será la similitud que presenten los objetos. Los conjuntos de puntos pueden estar etiquetados, es decir cada punto tiene una etiqueta con un identificador. Si los puntos están etiquetados el problema se reduce a encontrar las etiquetas que son iguales; sin embargo, cuando los puntos no están etiquetados se necesita encontrar la correspondencia entre los puntos de dos conjuntos, la complejidad del problema es de orden polinomial en el tamaño del conjunto de puntos, llegando a ser intratable para casos de conjuntos de puntos de gran tamaño.

En este trabajo nos enfocaremos en el diseño de representaciones para conjuntos de puntos que permitirán la búsqueda de objetos por similitud. Se busca que las representaciones a desarrollar sean robustas ante transformaciones lineales, inserciones, eliminaciones y ruido.

1.1. Justificación

La búsqueda en medios ricos en datos tales como audio, imágenes y videos, se mantiene como uno de los grandes retos computacionales debido a que existe una brecha entre las soluciones disponibles y las necesidades prácticas tanto en precisión como en el costo computacional (Wang *et al.*, 2016). En la literatura se encuentran trabajos en los cuales se presentan técnicas para comparar conjuntos de puntos; sin embargo, no presentan robustez ante todas las operaciones de transformaciones lineales, inserciones y eliminaciones y ruido. En este trabajo se presenta una nueva representación junto con una métrica de distancia. Está nueva representación deberá presentar una buena robustez ante las operaciones antes mencionadas.

1.2. Objetivos

A continuación se muestran los objetivos a lograr con el desarrollo de este trabajo de tesis.

1.2.1. Objetivo general

Diseñar una estructura de datos que permita la búsqueda de objetos utilizando una representación de nube de puntos.

1.2.2. Objetivos específicos

- Implementar una métrica de espigas e índices métricos para búsqueda de espigas
- Evaluar los tiempos de consulta del índice propuesto
- Evaluar la robustez del método de emparejamiento de nube de puntos

1.2.3. Preguntas de investigación

- ¿Cuál es la robustez de la representación propuesta?
- ¿Es posible disminuir la complejidad computacional de la representación por espigas?
- ¿Qué grafo genera espigas que presentan mejores resultados?

1.2.4. Organización de la tesis

Este trabajo de tesis está dividido en 7 capítulos organizados de la siguiente manera, en el capítulo 2 se presentan conceptos importantes que permitirán una mejor compresión del desarrollo del trabajo realizado. En el capítulo 3 se plantea el problema que se busca resolver, se describe parte de la propuesta debido a que de ella se genera otro de los problemas que se buscan resolver. En el capítulo 4 se presentan algoritmos y estructuras de datos presentes en la literatura que permiten resolver una gran cantidad de problemas relacionados. En el capítulo 5 se presentan trabajos encontrados en el estado del arte, además de algunos trabajos relacionados que justifican la importancia de este trabajo. En el capítulo 6 se presenta la estrategia utilizada en la experimentación, se divide en dos partes, la primera describe las características de los experimentos realizados y la segunda presenta los resultados obtenidos. Finalmente en el capítulo 7 se presentan las conclusiones obtenidas y propuestas de trabajos futuros utilizando como base este trabajo.

Capítulo 2. Marco Téorico

En este capítulo se presentan conceptos básicos y definiciones que serán fundamentales para el desarrollo de capítulos posteriores.

2.1. Consultas en espacios métricos

Un espacio métrico es un par $(\mathbb{U}, d)$ donde $\mathbb{U}$ es el universo de objetos y d una función definida como $d : \mathbb{U} \times \mathbb{U} \to \mathbb{R}^+$ que denota una medida de distancia entre objetos. La función de distancia tiene las siguiente propiedades:

- $d(x, y) \geq 0 \quad x, y \in \mathbb{U}$;
- $d(x, y) = 0 \quad$ si y solo si $\quad x = y$;
- $d(x, y) = d(y, x) \quad x, y \in \mathbb{U}$;
- $d(x, y) \leq d(x, z) + d(z, y) \quad x, y, z \in \mathbb{U}$;

2.1.1. Medidas de distancia

La función de distancia de un espacio métrico representa una manera de medir la cercanía de objetos en un dominio dado. Dependiendo del tipo de valor devuelto, las funciones de distancia se pueden dividir en dos grupos:

- **discreta :** $d : u \times u \to \mathbb{Z}^+$ devuelve enteros positivos
- **continua :** $d : u \times u \to \mathbb{R}^+$ devuelve reales positivos

Calcular una distancia puede ser computacionalmente costoso. Si se escala a volúmenes de objetos muy grandes el comparar distancias entre todos los objetos no es viable.

2.1.2. Consultas por similitud

Una consulta por similitud se define por el objeto de consulta y una restricción para la proximidad que se requiere. Esta restricción usualmente se expresa como una función de distancia. La consulta regresa todos los objetos que satisfacen el criterio de proximidad al objeto de consulta.

2.1.2.1. Consulta en rango

Es el tipo de consulta por similitud más común, está especificado por un objeto de consulta $q \in \mathbb{U}$, junto con un radio de consulta r como medida de restricción. La consulta recupera todos los objetos encontrados dentro del radio r respecto a q. De manera formal una consulta en rango se define como:

$$R(q, r) = \{o \in X, d(o, q) \leq r\}$$

Observese que el objeto de consulta q puede o no existir en la colección $X \subseteq \mathbb{U}$ a ser buscado, la única condición es que el objeto de consulta pertenezca al dominio métrico $\mathbb{U}$. Si el radio de consulta es 0 entonces se trata de una consulta exacta. En la *figura 4* se muestra que para un objeto de consulta q los objetos o_2, o_5 y o_6 se encuentran dentro del radio de consulta r.

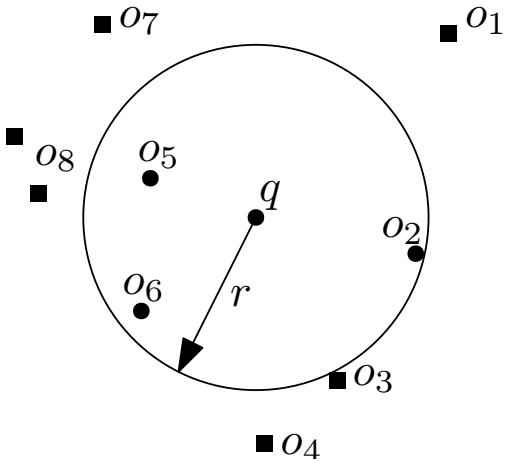

Figura 4. Consulta en rango R(q,r)

2.1.3. Consulta por vecino más cercano

En la consulta por rango se desconoce la distribución de los objetos dentro de la colección, si se proporciona un rango pequeño se puede devolver el conjunto vacío, y si el rango es muy grande la cardinalidad del conjunto devuelto puede llegar a ser el tamaño de la colección. Una alternativa para buscar objetos similares es utilizar la consulta por vecinos más cercanos. La versión fundamental de esta consulta encuentra el objeto más cercano al objeto de consulta. Este concepto se puede generalizar al caso donde nos interesa recuperar los k vecinos más cercanos al objeto q. Formalmente la consulta de k-vecinos más cercanos se define como sigue:

$$kNN(q) = \{R \subseteq X, |R| = k \wedge \forall x \in R, y \in X - R : d(q,x) \leq d(q,y)\}$$

Si múltiples objetos se encuentran a la misma distancia la selección es arbitraria. La *figura 5* muestra el caso de una consulta 3NN(q) en la cual los objetos o_3, o_4 y o_5 corresponden a los tres vecinos más cercanos.

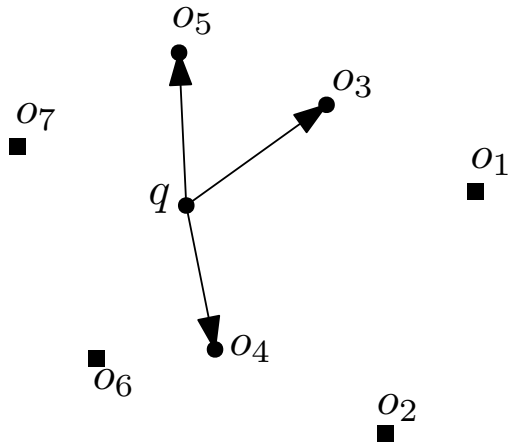

Figura 5. Consulta de 3 vecinos más cercanos 3NN(q)

2.1.4. Consulta inversa de vecino más cercano

Existen casos en los cuales interesa saber la distribución que tiene un objeto dentro de una colección; por ejemplo, cuales objetos tienen a q como su vecino más cercano. Esta consulta se conoce como búsqueda inversa de vecino más cercano. Es llamada $kRNN(q)$ y regresa todos los objetos con q entre sus k vecinos más cercanos. De manera formal se define como:

$$kRNN(q) = \{R \subseteq X, \forall x \in R : q \in kNN(x) \wedge \forall x \in X - R : q \notin kNN(x)\}$$

En la *figura 6* se muestra un ejemplo de consulta inversa de vecino más cercano, los objetos o_3, o_5 y o_6 tienen a q como su vecino más cercano.

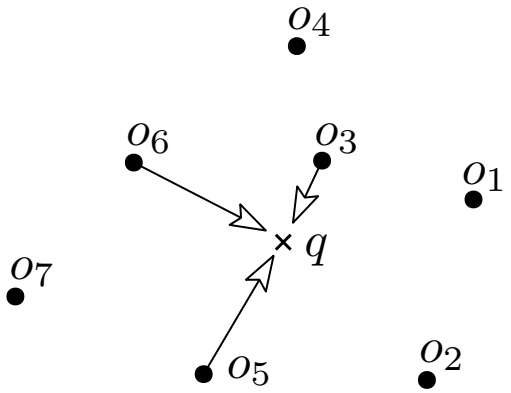

Figura 6. Consulta de vecino más cercano inversa RNN(q)

2.1.5. Principios básicos de partición

La partición es una de las estrategias que más se utilizan en cualquier estructura de almacenamiento, permite dividir el espacio de búsqueda en subgrupos de tal manera que cuando se realiza una consulta únicamente se tenga que buscar en algunos de esos grupos.

2.1.5.1. Partición en bola

La partición en bola parte el conjunto S en subconjuntos S_1 y S_2 usando un corte esférico con respecto a $p \in X$, donde p es llamado pivote y se selecciona de manera arbitraria. Sea d_m la distancia media de $\{d(o_i, p), \forall o_i \in S\}$. Entonces todo o_i se encuentra en S_1 o S_2 dependiendo si $d(o_i, p)$ es mayor o menor a la distancia media d_m. En la *figura 7* se muestra una partición en bola generando los conjuntos S_1 y S_2.

2.1.5.2. Partición en hiperplanos

Esta partición divide el conjunto S en los subconjuntos S_1 y S_2. La partición se logra utilizando dos objetos de referencia (pivotes) $p_1, p_2 \in X$. Los objetos se asignan a S_1 o S_2 de acuerdo a qué pivote sea el más cercano. Un ejemplo es mostrado en la *figura 8*

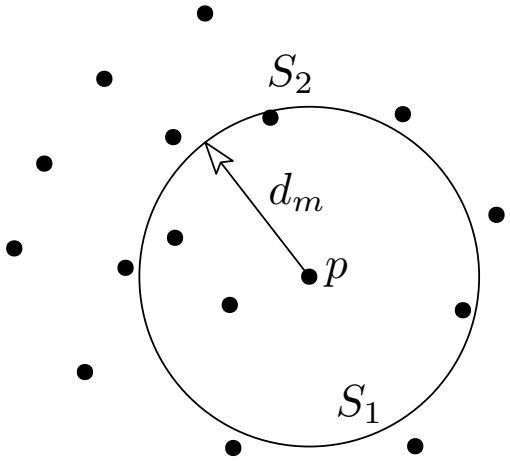

Figura 7. Ejemplo de partición en bola.

. Esta técnica no logra una partición balanceada y la elección de puntos de referencia que permita esta división es un reto.

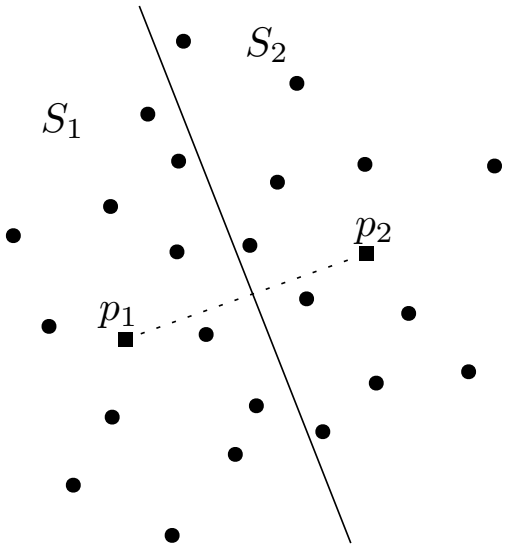

Figura 8. Ejemplo de partición en hiperplano.

2.1.5.3. Partición con medio excluido

Esta partición divide el conjunto S en tres subconjuntos S_1, S_2 y S_3. Es una extensión de la partición en bola, en esta técnica existe un umbral(δ) para la distancia media que mantiene los objetos cercanos a esta distancia excluidos de los otros dos subconjuntos. Esto permite ignorar alguno de los subconjuntos al realizar una consulta, la partición se puede definir como sigue:

- $S_1 \leftarrow \{o_j | d(o_j, p) \leq d_{m-\delta}\}$,

- $S_2 \leftarrow \{o_j | d(o_j, p) > d_{m+\delta}\}$,
- $S_3 \leftarrow$ otro caso.

En la *figura 9* se muestra un ejemplo de una partición con medio excluido, los círculos punteados delimitan el subconjunto que es excluido y se considera como un subconjunto más.

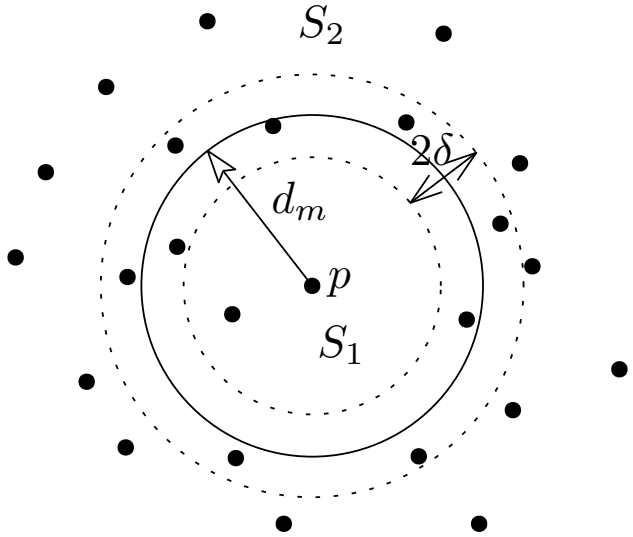

Figura 9. Ejemplo de partición con medio excluido.

2.1.6. Búsqueda de similitud aproximada

En general la búsqueda por similitud es costosa y los métodos en el estado del arte en ocasiones no proveen un tiempo de respuesta aceptable. Es por ello que se realiza una búsqueda por similitud aproximada la cual obtiene un conjunto de resultados inexactos pero suficiente para muchas aplicaciones. Lo interesante de realizar una búsqueda por similitud aproximada se debe a que se realiza de forma más rápida.

2.1.7. Medidas para evitar calcular distancias

El rendimiento de los algoritmos de búsqueda por similitud no se limita a las operaciones E/S (entrada/salida) sino también al procesador por lo cual es importante reducir tanto como sea posible la cantidad de cálculos de distancia. Para lograr esto se deben aplicar condiciones que permitan no solo evitar acceder a conjuntos irrelevantes sino también minimizar la cantidad de distancias que se calculan. La estrategia

es utilizar propiedades de los espacios métricos y de manera específica las características de la distancia tales como la simetría, no negatividad o la desigualdad del triángulo, que permiten determinar cotas a la distancia entre los objetos. Las técnicas de *poda* se utilizan en lo general en estructuras de índices para espacios métricos.

Considérese el siguiente ejemplo, en la *figura 10a* se muestra una partición en hiperplanos recursiva, y en la *figura 10b* el árbol binario correspondiente, donde cada nivel corresponde a un hiperplano. Cuando se realiza una consulta se determina en que lado del hiperplano está el objeto de esta manera se puede evitar revisar subconjuntos. Debe notarse que al ser una búsqueda con rango(r) se debe comprobar si el rango no excede el subconjunto en el que se ubica el objeto, si este es el caso entonces se deberán revisar ambos subconjuntos generados por el hiperplano.

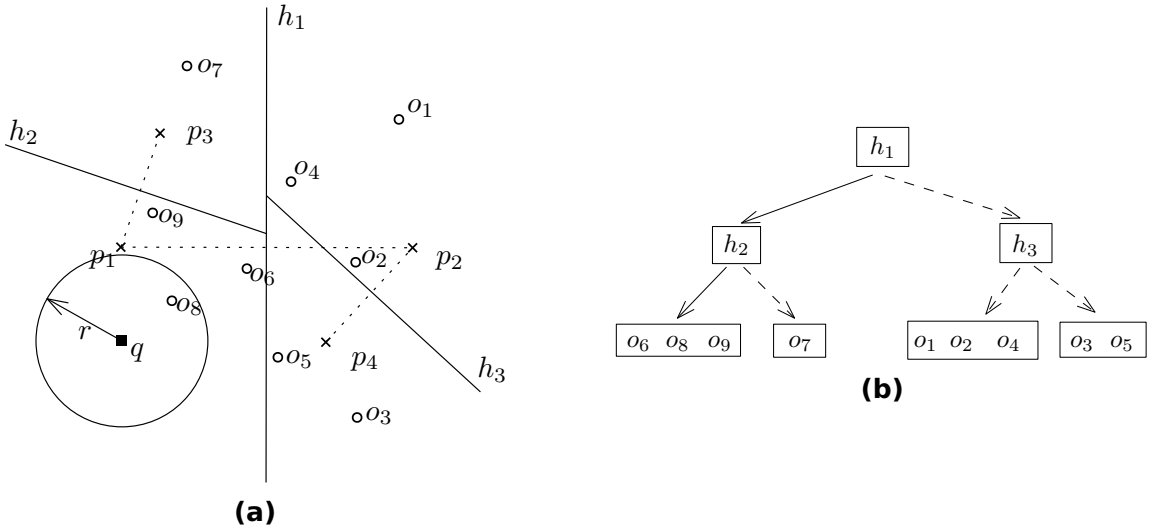

Figura 10. Búsqueda en rango para la consulta R(q,r), (a) Partición recursiva en hiperplanos, (b) Recorrido para determinar el subconjunto de *q*

2.1.8. Medidas de rendimiento

En la búsqueda por similitud existe un intercambio natural entre eficiencia de consultas y fiabilidad en los resultados. Para comparar dos algoritmos de búsqueda por similitud es importante conocer la relación entre esas dos medidas. Un buen algoritmo de búsqueda mostrará una buena eficiencia manteniendo resultados fiables. Por lo general los algoritmos de búsqueda por similitud se evalúan por el tiempo total que toma realizar una consulta, el tiempo total se puede descomponer en la fórmula (Chávez *et al.*, 2001a) :

$$T = \text{evaluaciones de distancia} \times \text{complejidad de } d() + \text{tiempo extra de CPU} + \text{tiempo E/S}$$

y se busca minimizar T. En algunas aplicaciones, el calcular $d()$ es la operación más costosa por lo que los demás componentes no se toman en cuenta. Sin embargo el tiempo de E/S puede ser un factor determinante para algunos casos.

2.1.8.1. Precisión y exhaustividad(recall)

Existen dos medidas que se utilizan para evaluar la calidad de los resultados regresados por una consulta. La precisión es la razón de los objetos recuperados y que son relevantes entre la cantidad total de objetos recuperados. La exhaustividad esta determinada por la cantidad de objetos relevantes que se recuperan entre la cantidad total de objetos relevantes para la consulta. Sea **A** el conjunto de objetos relevantes recuperados para un a consulta , sea **B** el conjunto de objetos no relevantes recuperados y sea **C** el conjunto de objetos relevantes pero no recuperados, la precisión y exhaustividad se definen formalmente como:

$$\text{Precisión} = \frac{|A|}{|A \cup B|}$$

y

$$\text{Exhaustividad} = \frac{|A|}{|A \cup C|}$$

Capítulo 3. Planteamiento del problema

Diversos problemas computacionales representan los objetos como conjuntos de puntos. Al trabajar con conjuntos de puntos se requiere determinar la similitud que existe entre dos conjuntos ya sea para realizar comparaciones o búsquedas. Sin embargo, los conjuntos de puntos no estan siempre fijos, pueden estar bajo alguna operación de transformación lineal, pueden existir inserciones y eliminaciones y puede existir ruido. El establecer un emparejamiento entre dos conjuntos de puntos bajo alguna de las operaciones anteriores ya no es una tarea trivial, por ello se necesita una representación que sea robusta ante este tipo de operaciones.

Los conjuntos de puntos como tal proporcionan poca información ya que cualquier punto del conjunto es igual a otro o se puede transformar mediante alguna función. Para diferenciar los puntos y poder determinar un emparejamiento se debe agregar información adicional que le proporcione contexto a cada punto. Una forma de lograr esto es generar un grafo sobre el conjunto de puntos. El grafo que se genere debe mantener un tamaño de orden lineal con respecto al tamaño del conjunto, un ejemplo de grafo que logra lo anterior es el que se obtiene al aplicar la triangulación de Delaunay, en la *figura 11* se muestra un ejemplo. El grafo que se obtiene al aplicar esta triangulación tiene como características adicionales el mantenerse invariante ante cambios de escala o rotaciones sobre el conjunto de puntos. Otro ejemplo de grafo es el que se obtiene de utilizar la técnica de los k-vecinos en la cual para cada punto se agregan aristas que lo unen con los k puntos que son sus vecinos más cercanos, en la *figura 12* se muestra un ejemplo de este tipo de grafo. El problema de utilizar un grafo obtenido de utilizar la técnica de los k-vecinos reside en la selección del valor de k. Para valores de k muy grandes el grafo tiende a ser uno completo y su tamaño llega al orden cuadrático. La importancia de mantener el tamaño de la representación en el orden lineal se debe a que en la práctica se comparan millones de objetos y existen limitaciones en el tamaño de memoria disponible.

Una vez que se obtiene el grafo tenemos información de contexto para los puntos, cada punto tiene información de su vecindad. Un punto se representa por lo que se llamará *espiga*. Una espiga esta compuesta por el punto y las aristas que son incidentes, la representación del objeto pasa de ser un conjunto de puntos a un conjunto de

espigas, ver *figura 13*.

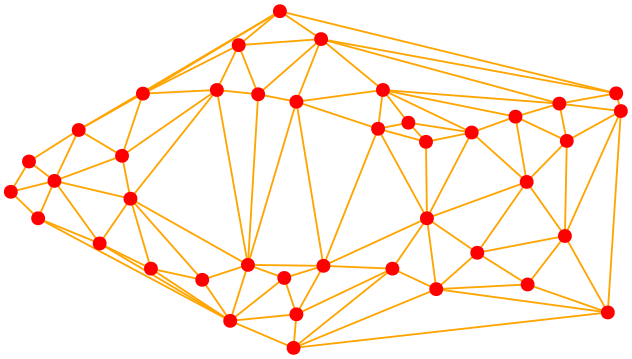

Figura 11. Ejemplo de grafo obtenido de aplicar la triangulación de Delaunay sobre el conjunto de puntos.

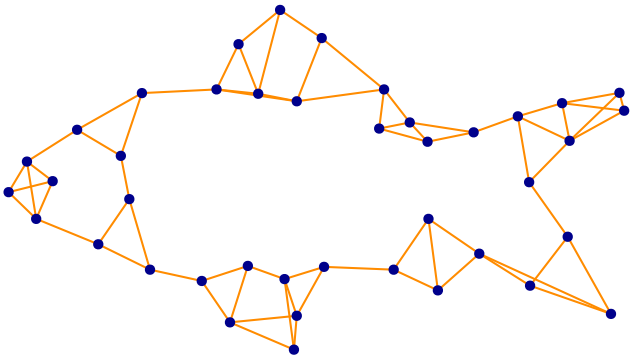

Figura 12. Ejemplo de grafo obtenido de aplicar la técnica de k-vecinos sobre el conjunto de puntos.

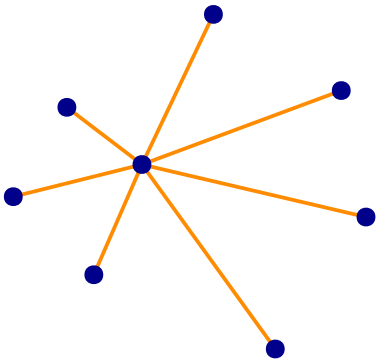

Figura 13. Ejemplo de una espiga obtenida de un grafo.

Debe notarse que por la forma de construcción, el grafo generado y las espigas se mantienen aún cuando el conjunto de puntos se encuentra rotado o bajo algún cambio de escala. Sin embargo, la presencia de inserciones de puntos ocasiona que se agreguen nuevas aristas que unen al nuevo punto insertado y las eliminaciones ocasionan que se eliminen las aristas que estaban unidas al punto que se eliminó. La cantidad de aristas que un punto tendrá de más o de menos depende de las características de su vecindad. Aún bajo inserciones y borrados las espigas tendrán cierta similitud con

respecto a su forma original ya que algunas de sus aristas se conservarán y otras estarán de más o menos. La similitud entre dos conjuntos de puntos estará dada por la cantidad de espigas iguales encontradas, para determinar si dos espigas son iguales se requiere establecer un criterio de similitud por lo que es necesario establecer una métrica de distancia.

3.1. Distancia entre dos espigas

La distancia entre dos espigas esta dada por la cantidad de operaciones de inserción y eliminación necesarias para igualar las espigas. Nótese que al insertar o eliminar aristas, los ángulos entre las aristas que no se modificaron se mantienen, mientras que los ángulos formados por un par de aristas, entre las cuales se insertó o eliminó alguna arista, corresponden a la suma de los ángulos inducidos por las aristas modificadas. Si se inserta una nueva arista ocasiona que el ángulo en el que se insertó se divida en dos nuevos ángulos, y si se elimina una arista los ángulos adyacentes a la arista se unen en un nuevo ángulo. Al utilizar los ángulos se garantiza que las espigas se mantienen iguales aún cuando el conjunto de puntos se encuentra bajo alguna operación de rotación o cambio de escala.

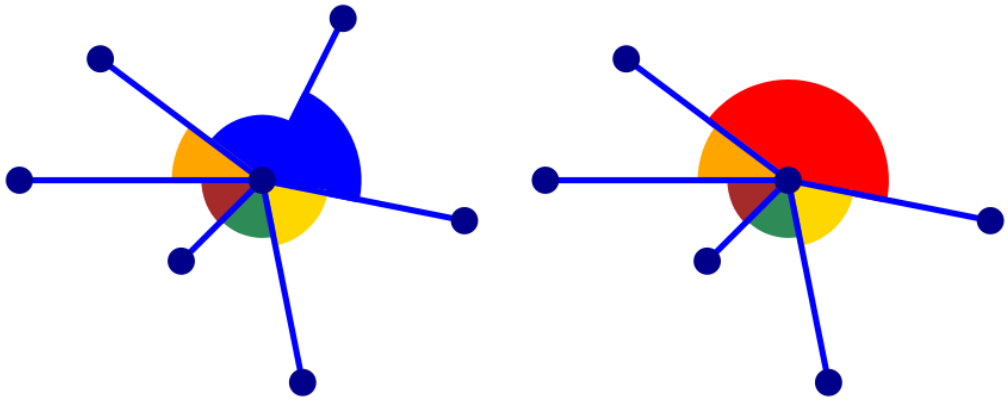

Figura 14. Ejemplo de correspondencia de ángulos entre espigas, esta dado por el color de los ángulos; el ángulo de color rojo(arista eliminada) corresponde a la unión de los ángulos de color azul.

Para calcular la distancia entre dos espigas se debe encontrar el emparejamiento óptimo de sus ángulos, de está manera se puede determinar la cantidad de aristas que se deben eliminar o insertar y que hacen las dos espigas iguales. Debe notarse que si se van sumando los ángulos ya sea en dirección de las manecillas del reloj o en dirección contraria, al final la suma será de 360°. Las sumas parciales en cualquier de las

dos direcciones serán monótonamente crecientes hasta llegar a 360°. Si dos espigas son similares entonces ocurren dos cosas, los valores de sus ángulos son iguales ó el valor de un ángulo de una espiga corresponde a la suma de dos ángulos de la otra espiga. Si se conoce donde empezar a emparejar las sumas parciales serán iguales para los casos anteriores. Sin embargo, cuando no se conoce donde empezar a emparejar se deben probar todos los posibles emparejamientos con el objetivo de encontrar las sumas parciales que logren la mayor cantidad de valores iguales para ambas espigas.

La distancia de un emparejamiento entre dos espigas esta dada por la formula, $(n+m)-2*(e)$, donde n y m representan la cantidad de aristas que contienen las espigas y e es la cantidad de valores iguales que se encontraron. La distancia entre dos espigas corresponde a la mínima distancia obtenida de entre todos sus emparejamientos posibles.

Ejemplo : Sean *spk1* y *spk2* las espigas que se muestran en la *figura 15a* y *15b* respectivamente. Se puede observar que *spk2* es la *spk1* con una arista insertada que divide el ángulo de *90* en dos nuevos ángulos. Cuando se conoce donde empezar a emparejar las sumas parciales de los ángulos corresponderá en ambas espigas. En la *figura 16* se muestra el emparejamiento que se logra cuando se realizan las sumas parciales a partir del ángulo 85. Aunque los valores individuales de los ángulos no correspondan las sumas si lo hacen. Al aplicar la formula tenemos que $n=5$ para *spk1*, $m=6$ para *spk2* y $e=5$. La distancia entre las espigas (a) y (b) es de $(5+6)-2*(5)=1$.

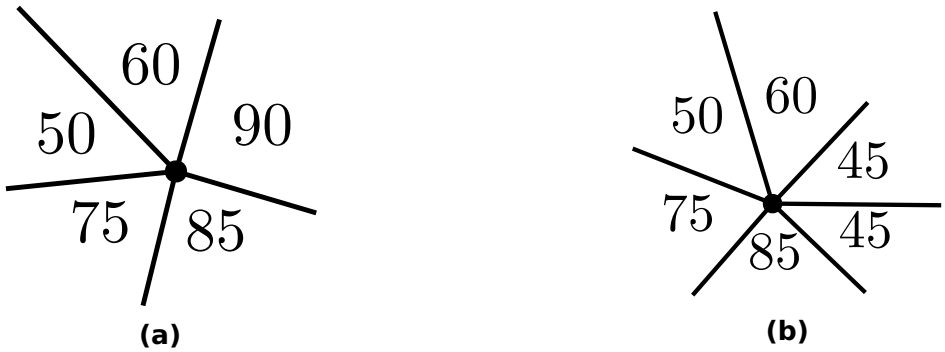

Figura 15. Ejemplo de espigas con valor de ángulo entre sus aristas.

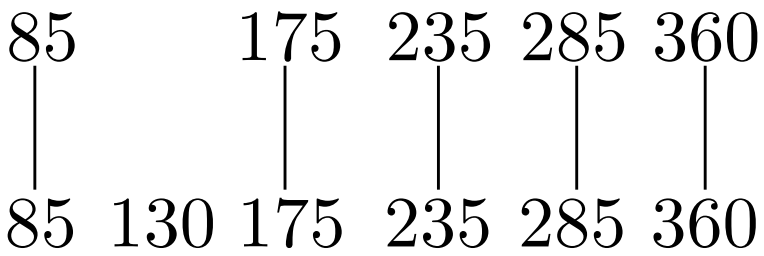

Figura 16. Emparejamiento de las sumas parciales a partir del ángulo de 85 en sentido contrario a las manecillas del reloj.

Capítulo 4. Índices Métricos

La idea detrás de los algoritmos de indexado es reducir la cantidad de distancias que se calculan al realizar una consulta, una manera de lograr esto es partir el conjunto de datos en subconjuntos de tal manera que cuando se realice la consulta se busque en los subconjuntos relevantes y se evite revisar aquellos que no lo son. Los algoritmos de indexado se pueden clasificar de diferentes maneras ya sea por el tipo de función de distancia ó por tipo de partición utilizada, por el tipo de almacenamiento utilizado por decir algunos. Aquí se utilizará la división en cuatro grupos de acuerdo a la partición del espacio como presenta (Zezula *et al.*, 2006).

4.1. Métodos de partición en bola

Las características de la partición en bola son, solo se necesita un pivote y genera subconjuntos balanceados utilizando la distancia media d_m. Numerosos trabajos han utilizado esta técnica de partición.

4.1.1. Árbol Burkhard-Keller

Esta solución se presenta por (Burkhard y Keller, 1973), generalmente conocido como árbol BKT, se utiliza para funciones de distancia discretas y se construye de manera recursiva como sigue: de un conjunto de objetos X, se selecciona un objeto arbitrario $p \in X$ el cual será la raíz del árbol. Para cada distancia $i \geq 0$, los subconjuntos $X_i = \{o \in X, d(o,p) = i\}$ se definen como el grupo de objetos que se encuentran a distancia i de p. Para cada subconjunto no vacío se construye un nodo hijo de p, el proceso de partición se repite de manera recursiva para cada subconjunto. El algoritmo para búsquedas en rango es simple. La búsqueda en rango R(q,r) inicia en el nodo raíz y compara el objeto p con el objeto consulta q si $d(p,q) \leq r$ entonces p es parte del conjunto devuelto, se continua revisando todos los nodos hijos de p cuya distancia a q se encuentre dentro del rango.

En la *figura 17* se muestra un ejemplo de un espacio métrico y el árbol BKT construido sobre el. En la *figura 17a* se muestran los objetos a distintas distancias del pivote p y la consulta en rango q, en la *figura 17b* se ilustra el árbol BKT generado a partir del

conjunto de objetos, el recorrido de la consulta *q* está marcado por las lineas continuas y los objetos del conjunto respuesta están marcados por un rectángulo.

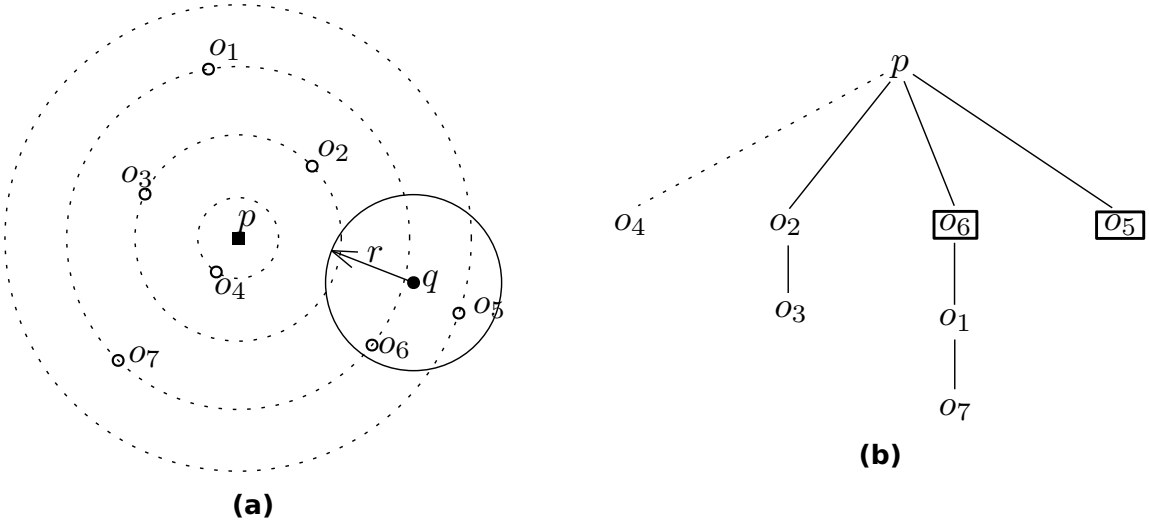

Figura 17. Búsqueda en rango para la consulta R(q,r) en BKT, (a) Conjunto de puntos y consulta *q*, (b) árbol construido sobre el conjunto de objetos

4.1.2. Árbol de Consultas Fijas

Es conocido como FQT por sus siglas en inglés (Fixed Queries Tree) fue propuesto por (Baeza-Yates *et al.*, 1994), presenta modificaciones al BKT que permite reducir el número de distancias que se calculan. La diferencia se debe a que los objetos son almacenados en hojas y los nodos intermedios solo ayudan a navegar a través del árbol. La búsqueda en rango se realiza de la misma manera que en el BKT. La característica de esta estructura es que todos los nodos hoja se encuentran en el mismo nivel del árbol. Existe una variante al FQT, llamada FHFQT(Fixed-Height Fixed Queries Tree) la cual limita la altura de los árboles construidos.

4.1.3. Arreglo de Consultas Fijas

Es llamado FQA por sus siglas en inglés(Fixed Queries Array), se presenta por (Chávez *et al.*, 2001a). Es una estructura muy relacionada al FHFQT con la diferencia que el FQA utiliza arreglos en vez de árboles para almacenar los objetos. Se crea para reducir la cantidad de memoria utilizada. No presenta buenos resultados para distancias continuas.

4.1.4. Árbol Vantage Point

El árbol Vantage Point se presenta por (Yianilos, 1993) y está diseñado para funciones de distancia continuas aunque se puede utilizar para funciones de distancia discretas. El árbol se crea utilizando una partición en bola recursiva. Una consulta en rango R(q,r) recorre el árbol desde la raíz hasta las hojas. En cada nodo intermedio se calcula la distancia entre el pivote y el objeto consulta para visitar los subárboles que correspondan.

En la *figura 18* se muestra un ejemplo de VPT, se selecciona un pivote del conjunto de puntos y se calcula d_m y se divide en dos subconjuntos, posteriormente el proceso se repite una vez más para cada subconjunto. Como resultado se genera una partición en cuatro subconjuntos. Dada la consulta q se puede observar que se revisan tanto el conjunto S_0 como el conjunto S_1 a pesar de que la intersección con el conjunto S_1 es muy pequeña.

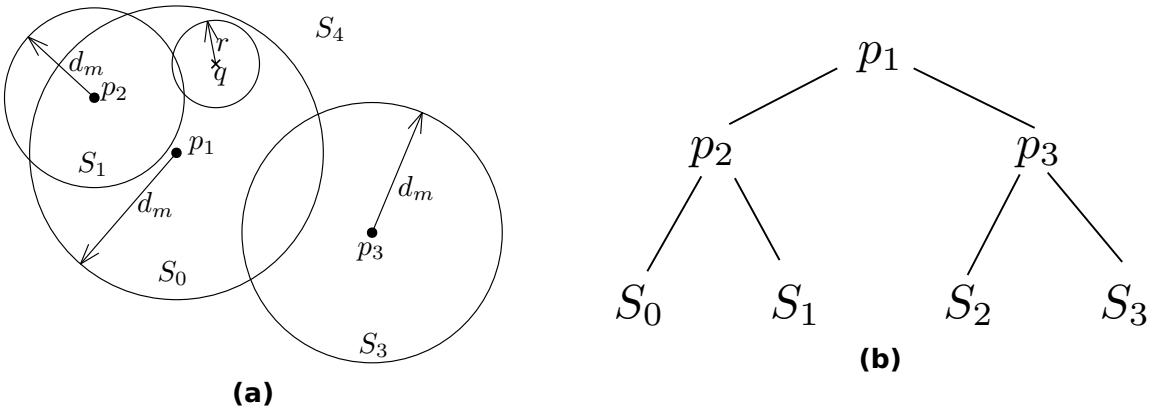

Figura 18. Búsqueda en rango para la consulta R(q,r) en BKT, (a) Conjunto de puntos y consulta q, (b) árbol construido sobre el conjunto de objetos

4.2. Enfoque de partición utilizando hiperplanos

A continuación se presentan métodos de indexado basados en el enfoque de partición por hiperplanos.

4.2.1. Árbol Bisector

Fue la primer estructura de índice en utilizar la partición de hiperplanos, es conocido como BST y fué propuesto por Kalantari y McDonal en 1983. El BST se construye

partiendo el conjunto recursivamente utilizando hiperplanos, cada subconjunto formado por el hiperplano corresponde a un subárbol. El algoritmo de búsqueda en rango $R(q, r)$ revisa un subárbol si $d(q, p_i) - r$ está dentro del subconjunto cubierto por el pivote p_i, el área de cobertura se define por la mayor distancia del pivote a un elemento del subconjunto.

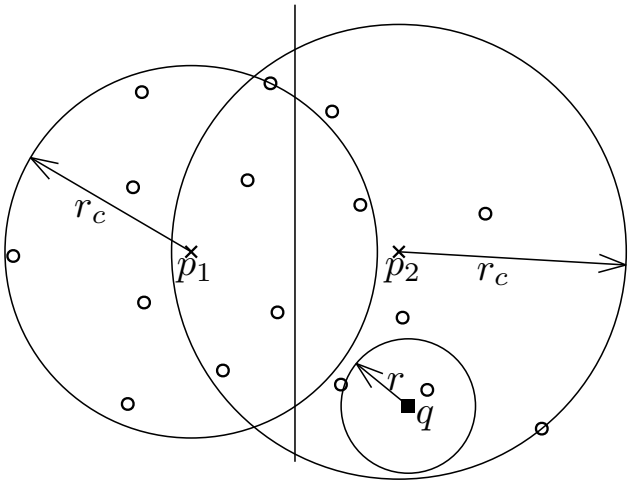

Figura 19. Ejemplo de partición utilizando el BST con consulta en rango R(q,r).

4.2.2. Árbol de hiperplano generalizado

Esta técnica es muy similar al BST, realiza la partición del conjunto de datos utilizando hiperplanos de manera recursiva. La diferencia se debe a que en vez de utilizar el área de cobertura de los pivotes para evitar revisar subconjuntos utiliza el hiperplano entre los pivotes para decidir qué subárbol se debe consultar.

4.3. Explotando distancias pre-calculadas

Cuando el cálculo de distancias se vuelve muy costoso uno busca reducir la cantidad de distancias a calcular al mínimo. En (Shasha y Wang, 1990) se propone utilizar un conjunto de distancias precalculadas. Para un conjunto de datos de tamaño n se genera una tabla de tamaño $n \times n$ donde se almacenan las distancias entre todos los objetos, cuando se busca conocer la distancia contra un objeto no presente en la tabla se utilizan los intervalos entre los objetos ya presentes para estimarla. En esta sección se presentan algunos algoritmos que hacen uso de la matriz de distancias.

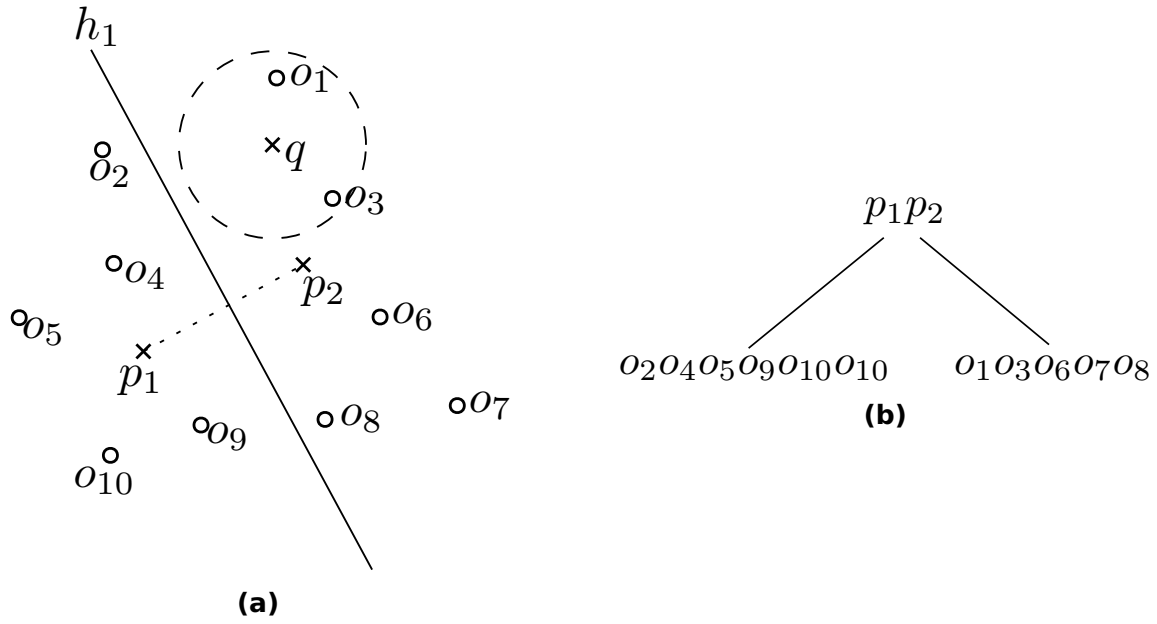

Figura 20. Árbol de hiperplano generalizado(GHT): (a) Consulta en rango R(q,r) sobre el conjunto de objetos, (b) Representación en árbol de la partición generada

4.3.1. AESA

Su nombre viene por las siglas en inglés (Approximating and Eliminating Search Algorithm), fue propuesto por (Ruiz, 1986) con el objetivo de reducir los tiempos de consultas de vecino más cercano. Utiliza una matriz de distancias que se calcula durante la creación de la estructura. A diferencia con los métodos antes mencionados, en AESA cada objeto representa un pivote. Para una consulta en rango R(q,r) se selecciona un objeto p de manera aleatoria y se utiliza como pivote. Se calcula $d(q, p)$ la cual se utilizará para reducir la cantidad de elementos a revisar. Un objeto o no debe ser revisado si $|d(q, p) - d(p - o)| > r$, se selecciona entonces otro pivote entre los objetos restantes y se repite el proceso hasta que el tamaño del conjunto de objetos no descartados es lo suficientemente pequeño. Al final se revisa la distancia entre q y los objetos o no descartados. Los objetos para cuales la distancia $d(q, o) \leq r$ se cumple son regresados.

La principal limitante de este método es la matriz de distancias que se tiene que almacenar, para tamaños de conjuntos pequeños puede ser viable sin embargo para conjuntos de datos muy grandes el tiempo de construcción y el tamaño del almacenamiento resulta no ser práctico.

4.3.2. AESA Lineal

Esta técnica surge como una solución al problema de almacenamiento de orden cuadrático que presenta AESA. Fue propuesto por (Mico *et al.*, 1992) y funciona generando una matriz en la que se calculan distancias entre una serie de pivotes elegidos hacia todos los objetos, la matriz resultante es de tamaño $n \times m$, el crear la matriz de esta manera genera otro problema, como elegir los pivotes para reducir la cantidad de distancias a calcular. Una técnica propuesta por el mismo autor consiste en seleccionar los objetos pivotes que se encuentren lo más alejados posibles unos de otros. El algoritmo de búsqueda es similar al del AESA con la diferencia de que no todos los objetos serán pivotes

4.4. Enfoque de indexado hibrido

El utilizar distancias precalculadas reduce el costo computacional; sin embargo, su desventaja es la cantidad de espacio requerido. Por ello surge la estrategia de combinar métodos de partición y métodos de distancias precalculadas.

4.4.1. Árbol multi Vantage Point

Es conocido como MVPT por sus siglas en inglés (Multi Vantage Point Tree) es una extensión del VPT propuesta por (Bozkaya y Ozsoyoglu, 1997). MVPT reduce el número de pivotes usados para construir el árbol. Se utilizan dos pivotes en cada nodo interno en vez de uno.

4.4.2. GNAT

Su nombre viene de las siglas (Geometric Near-neighbor Access Tree), es una estructura de datos que refleja la geometría de los datos, fue propuesta por (Brin, 1995). La forma de construcción es como sigue, se selecciona un conjunto de objetos separadores(pivotes), $p_1, ..., p_k$ de manera aleatoria y manteniendo los objetos lo suficientemente separados. Los objetos restantes son agrupados de acuerdo a cual pivote se encuentran más cerca. El árbol es construido de manera recursiva generando la partición del conjunto de datos hasta llegar a un tamaño determinado. La consulta se

realiza comparando la distancia $d(q, p_i) \leq r$ en cuyo caso p_i se agrega a la solución y se procede a revisar su subárbol.

4.4.3. Árbol de Aproximación Espacial

Este índice esta inspirado en el diagrama de Voronoi, en particular utiliza la característica de vecino más cercano que presenta el grafo de Delaunay, es propuesto por (Navarro, 2002). Para un conjunto de datos X, el árbol de aproximación espacial (SAT) se define como: la raíz del árbol está representada por un objeto arbitrario p y el conjunto de vecindad($N(p)$) más pequeño del objeto p está determinado por:

$$o \in N(p) \Longleftrightarrow \forall o' \in N(p)/\{o\} : d(o, p) < d(o, o')$$

En la *figura 21* se muestra un ejemplo de árbol de aproximación espacial, en la *figura 21b* se muestra el árbol generado a partir del objeto o_5 como raíz. Para construir el árbol una vez encontrado el conjunto $N(p)$, para cada vecino de p se genera un subárbol repitiendo el proceso de encontrar su vecindad más pequeña. Encontrar $N(p)$ es un problema difícil de calcular por lo que se utilizan heurísticas para construir este conjunto, sin embargo no hay garantía de encontrar el conjunto mínimo. Para realizar una consulta en rango $R(q, r)$ para la vecindad de cada punto se encuentran los objetos cercanos a q y sus respectivos subárboles son visitados.

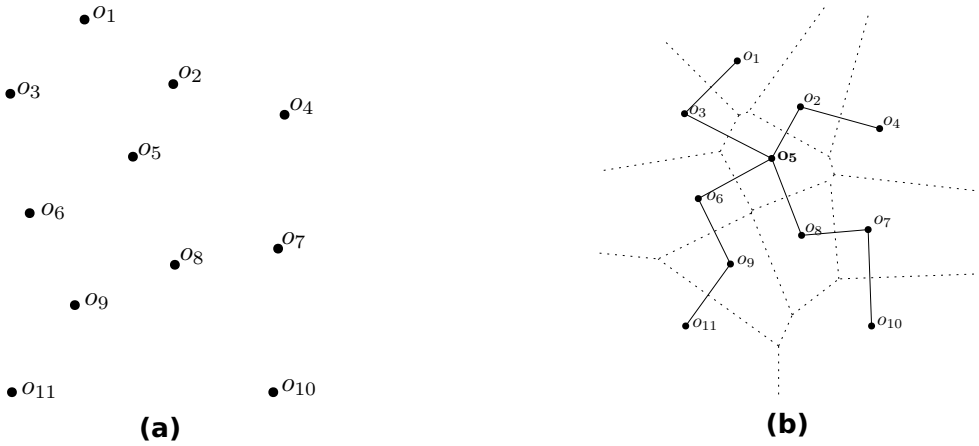

Figura 21. Árbol de aproximación espacial: (a)Conjunto de objetos, (b) Representación en árbol a partir de objeto o_5

4.4.4. Árbol M

En (Ciaccia *et al.*, 1997) se propone una estructura dinámica llamada árbol métrico (metric tree) o comúnmente conocida como M-tree. El M-tree está diseñado para trabajar con archivos y soporta archivos de datos cuyo tamaño es variable, esto supone una gran ventaja cuando la inserción o eliminación de objetos son operaciones frecuentes. La construcción se realiza desde las hojas hasta llegar a la raíz, los objetos son organizados en conjuntos de tamaño fijo que corresponden a regiones del espacio. Cada nodo hoja almacena un identificador de objeto, un vector de características y la distancia a su nodo padre. Los nodos intermedios almacenan un objeto enrutador(O_r), un vector de características, un radio de cobertura y la distancia al nodo padre. El radio de cobertura(r_c) define la región en el espacio métrico centrada en el objeto O_r y con radio r_c. Una consulta en rango $R(q, r)$ revisa todo subárbol para el cual $d(q, O_r) \leq r$. En la *figura 22* se muestra un ejemplo de una partición en regiones las cuales se encuentran dentro de otra región. La región esta acotada por el radio de cobertura y se encuentra centrada en o_r.

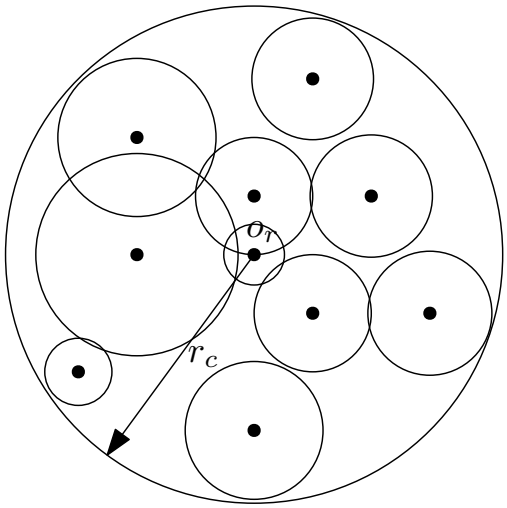

Figura 22. Ejemplo de región dividida que se mantiene en un nodo interno.

4.4.5. Hash por similitud

El utilizar hash es una solución útil cuando se quiere realizar una búsqueda de vecinos más cercanos de una manera aproximada. El hashing es un enfoque para reducir datos a una representación de menor dimensión o una representación equivalente de menor tamaño, por lo general se utilizan códigos cortos los cuales están formados por una secuencia de bits. El algoritmo de hash convencional intenta evitar colisio-

nes mientras que el enfoque de hash por similitud tiene como objetivo maximizar la probabilidad de colisión de los elementos cercanos (Indyk *et al.*, 1997). El principal esfuerzo al trabajar con hash por similitud es encontrar las funciones que maximicen la probabilidad de colisión.

En la *figura 22* se muestra la taxonomía de los algoritmos de indexado presentada por (Chávez *et al.*, 2001b), se presenta además en la *tabla 1* un breve resumen de las principales características de utilizar cada uno de los algoritmos de indexado presentes en la literatura.

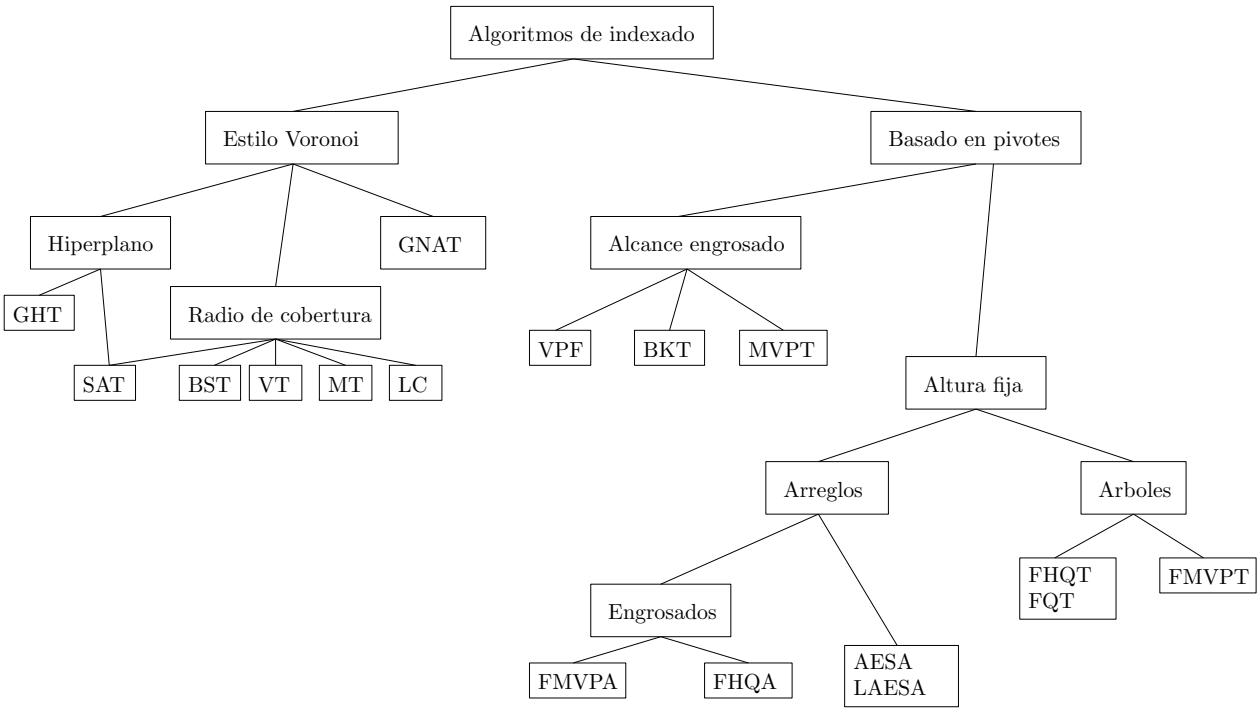

Figura 23. Taxonomia de algoritmos de indexación presentada por (Chávez *et al.*, 2001b), Fig. 20

Tabla 1. Resumen de características de los algoritmos de indexado.

Clasificación	Algoritmo	Características
Partición en bola	BKT	No hay un método de selección para los puntos pivotes de referencia
	FQT	
	FQA	
	VPT	
Partición en hiperplanos	Bisector	Selección de puntos de referencia que generen una partición balanceada
	Hiperplano generalizado	
Distancias precalculadas	AESA	Alto uso de memoria
	LAESA	Tiempo de preprocesamiento
Hibridos	MVPT	Árboles generados no son balanceados
	GNAT	
	SAT	
	Árbol M	Es creado en disco, complemento a sistemas de manejo de bases de datos.
Aproximación	Hash	Útil para consultas de similitud aproximada

Capítulo 5. Nubes de puntos

Como se ha venido mencionando en el desarrollo de este trabajo, los conjuntos de puntos se pueden utilizar para representar una gran variedad de problemas computacionales. La navegación astronómica es uno de esos problemas ya que podemos comparar las estrellas en el espacio como un conjunto de puntos. Las naves espaciales que orbitan alrededor de la tierra utilizan una serie de dispositivos para poder orientarse tales como GPS, magnetómetros o giroscopios. Sin embargo cuando se quiere navegar en el espacio alejado de la Tierra todos los dispositivos antes mencionados no se pueden utilizar.

Los cambios en las estrellas requieren mucho tiempo por lo cual parece que se mantienen estáticas y es posible utilizar esta característica para poder orientar naves espaciales en el espacio profundo. En la literatura se pueden encontrar distintos métodos para identificar estrellas, algunos de ellos utilizan técnicas geométricas al representar las estrellas mediante polígonos, otros utilizan redes neuronales e incluso se encuentran trabajos con algoritmos genéticos. (Hernández *et al.*, 2017) utilizan un método basado en polígonos para encontrar estrellas. Cada estrella tiene un polígono asociado el cual se construye de la siguiente manera. Primero las estrellas son proyectadas al plano xy, despues para una estrella determinada v_1 se selecciona una vecindad de tamaño k. Después se busca la estrella v_2 que se encuentre más cercana a v_1 dentro de la vecindad seleccionada. El resto de estrellas $v_3, \ldots, v_k$ en la vecindad se ordenan de acuerdo al ángulo positivo θ que forman con respecto a v_1 y v_2. En la *figura 24* muestra un ejemplo de polígono construido para la estrella v_1. El método que proponen garantiza que los polígonos sean invariantes ante rotaciones. Para mantener los polígonos invariantes ante transformaciones de similitud se convierten a un número complejo mediante una serie de funciones ϕ_l. El trabajo antes mencionado tiene como ventajas, la rapidez obtenida al utilizar números complejos para las consultas y el soporte ante transformaciones de similitud; sin embargo, por las características del problema tratado no se aborda un problema presente al trabajar con nubes de puntos el cual son las inserciones y eliminaciones. El problema resuelto en el trabajo presentado por (Hernández *et al.*, 2017) es el del buscar un punto dentro de un conjunto; se busca identificar una estrella utilizando su vecindad como contexto.

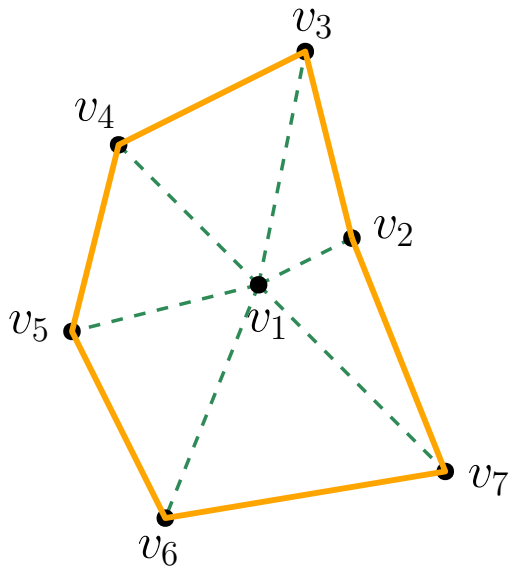

Figura 24. Construcción de polígono sobre vecindad del punto v_1.

En el trabajo presentado por (Ramírez-Chacón *et al.*, 2018) se proponen una serie de estructuras de datos para búsqueda de nubes de puntos cuando se presenta ruido o inserciones y borrados. El problema que se resuelve es el de buscar una nube de puntos dentro de una colección de nubes de puntos. En ese trabajo se supone que los puntos de una nube de puntos fueron extraídos utilizando un método que garantiza una invarianza a transformaciones de similitud. Los experimentos se realizaron utilizando una base de datos sintética de 10 millones de nubes de 1000 puntos cada una bajo una distribución uniforme. Se presenta una representación basada en rejillas la cual utiliza una matriz de bits. Para generar la matriz de bits se divide el espacio en celdas, se asigna 1 a las celdas donde se encuentra algún punto y 0 en caso contrario, un ejemplo de esta construcción se muestra en la *figura 25*. La selección del tamaño de las celdas es un factor determinante, un tamaño de celda grande genera una matriz de bits pequeña y los puntos cercanos es probable se asignen a una misma celda. Si muchos puntos son asignados a la misma celda los resultados se ven afectados ya que el incremento de colisiones incrementa la cantidad de falsos positivos. Sin embargo, cuando se utilizan tamaños de celda pequeños se genera una matriz de tamaño grande lo cual se traduce en un decremento de colisiones. El decremento de colisiones genera mejores resultados al disminuir la cantidad de falsos positivos pero al costo de utilizar mayor cantidad de memoria.

En este trabajo de tesis se busca presentar una solución a los dos problemas que

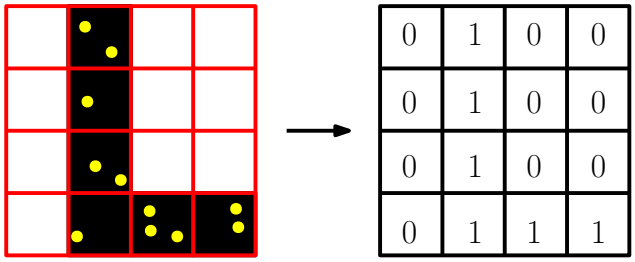

Figura 25. Representación de nubes de puntos mediante mapa de bits(*derecha*) utilizando rejillas(*izquierda*).

tratan los trabajos presentados anteriormente, esto es poder buscar un punto dentro de un conjunto de puntos y poder buscar conjuntos de puntos dentro de una colección.

5.1. Casos de aplicaciones prácticas

En la literatura se pueden encontrar trabajos en áreas diversas que utilizan conjuntos de puntos como representación, a continuación se presentan algunos trabajos con esta característica.

5.1.1. Audio

En el trabajo presentado por (chun Wang, 2003) se hace uso de nubes de puntos para representar archivos de audio. Las nubes de puntos son utilizadas para poder realizar consultas de archivos de audio sobre una base de datos. De acuerdo a datos que se presentan por (chun Wang, 2003), el uso de los picos de espectrogramas tiene gran robustez ante el ruido. Un punto tiempo-frecuencia se considera candidato si tiene mayor energía que la vecindad sobre la cual se encuentra centrado. El espectrograma está representado por tres dimensiones y se reduce a un espacio tiempo-frecuencia eliminando así la amplitud. Una vez que se reduce la dimensión de los puntos se transforman a códigos hash. Para generar los códigos hash se seleccionan puntos ancla a los cuales se les asocia una región objetivo, para cada punto en la región de un punto ancla se crea una tupla con las frecuencias de ambos puntos y su diferencia en el tiempo, cada tupla se convierte a un código hash. En la *figura 26* se muestra un ejemplo de un espectrograma y un punto ancla seleccionado junto con su región asociada

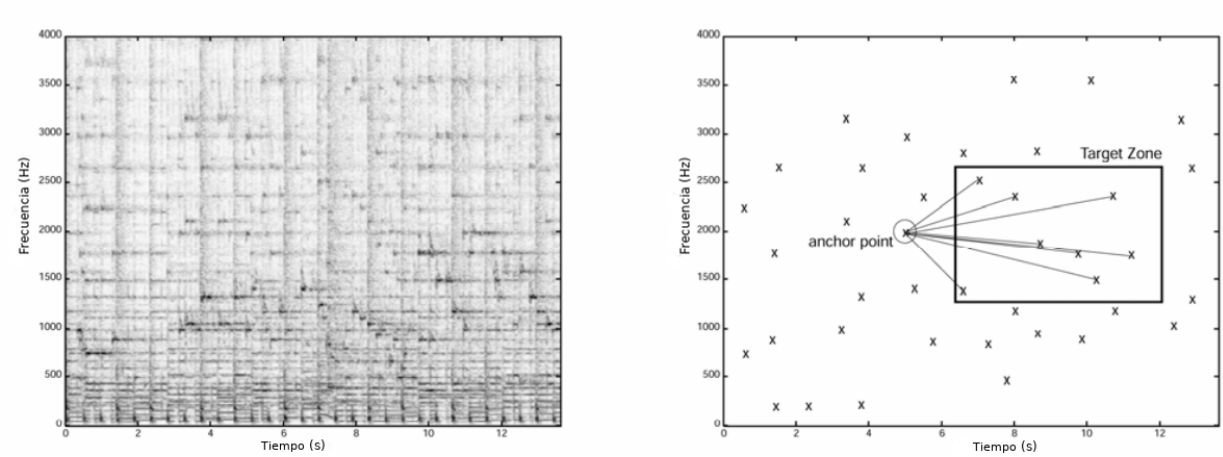

Figura 26. Ejemplo de espectrograma(*izquierda*) y el punto ancla seleccionado(*derecha*), (chun Wang, 2003), Fig. 1A, 1D

5.1.2. Recuperación de imágenes

En el campo de indexado y recuperación de imágenes, las características que se extraen de las imágenes son un factor importante. Dos tipos de características son utilizadas dependiendo de la estrategia que se quiere utilizar, las características globales y locales. Las características globales utilizan información general de la imagen como puede ser la forma, el color o la textura. Las características locales en cambio consisten de puntos de interés que están representadas por información específica tal como esquinas o aristas. Al extraer puntos de interés se considera que sean repetibles como un criterio de estabilidad para el método ya que permite encontrar los mismos puntos de interés aún cuando la imagen esta bajo alguna transformación de cambio de escala, rotación o iluminación. En la literatura se pueden encontrar diversos algoritmos detectores de puntos de interés, en (Bahroun *et al.*, 2014) se presenta un trabajo para realizar consultas en imágenes satelitales utilizando puntos de interés que se obtienen utilizando tres algoritmos muy conocidos, Harris, SIFT y SURF. Harris es un detector de esquinas y lleva el nombre de su creador. SIFT viene de las sigla en ingles de *Scale Invariant Feature Transform* y es un detector/descriptor de regiones que extrae características de imágenes las cuales son invariantes a cambios de iluminación y transformaciones afines. SURF (Speeded Up Robust Features) es un detector y descriptor de puntos de interés que son invariantes ante cambios de escala y rotación.

El enfoque utilizado por (Bahroun *et al.*, 2014) para emparejar puntos de interés

extraídos de imágenes satelitales tiene como primer paso, extraer un descriptor local alrededor de los puntos de interés utilizando el código LBP. LBP es el acrónimo de Local Binary Pattern y es un descriptor visual el cual genera un vector de características. LBP funciona de la siguiente manera, en la imagen se selecciona una ventana la cual se divide en celdas de cierto tamaño *figura 27*, cada pixel en la celda se compara con cada uno de los ocho vecinos que se encuentran alrededor, en la *figura 28* se muestra la vecindad para el punto a comparar. Si el valor del pixel del centro es mayor que el valor de su vecino se asigna un 0 y 1 en caso contrario. Una vez comparado el pixel central con sus vecinos se genera un número binario de 8 dígitos. Los números binarios generados se utilizan para calcular un histograma de números binarios sobre la celda. El descriptor de la ventana es una concatenación de los histogramas de todas las celdas. Una vez que se han calculado los descriptores para los puntos de interés se realiza un emparejamiento basándose en sus códigos LBP calculados previamente, después se reduce la cantidad de falsos candidatos realizando una prueba espacial donde para cada punto de interés se compara el ángulo que forma con sus dos vecinos más cercanos y la relación de distancia que mantienen.

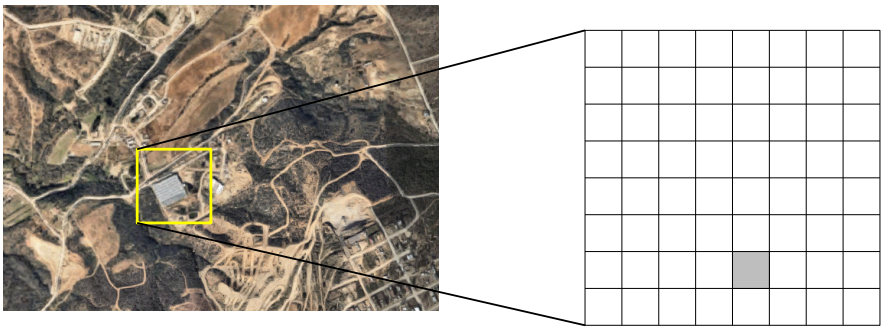

Figura 27. Selección de ventana sobre una imagen(*izquierda*) división de ventana en celdas(*derecha*).

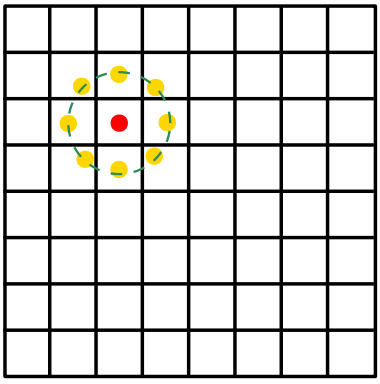

Figura 28. Vecindad(puntos amarillos) para el punto de interés(punto rojo).

Capítulo 6. Validación experimental

En este capítulo se describen las características de los experimentos realizados así como los resultados obtenidos.

6.1. Diseño de experimentos

A continuación se exponen los experimentos realizados para obtener y evaluar las características de las espigas.

Los algoritmos se implementaron en el lenguaje de programación c++11 utilizando la librería de geometría computacional CGAL (Yvinec, 2018). Se utilizó un sistema operativo Ubuntu Linux versión 14.04.5 LTS. Las características técnicas del servidor donde se realizaron los experimentos son las siguientes:

- Procesador Intel(R) Xeon(R) CPU E7-4809 v3 @ 2.00GHz.
- Memoria RAM 1.5 TB.
- 64 núcleos.

La mayor parte de experimentos descritos a continuación(excepción caso de imágenes) se realizaron utilizando puntos generados de manera aleatoria con una distribución uniforme en el rango [1, 1024] para las coordenadas x, y. Por cuestiones del tiempo de ejecución y para mantener una consistencia, cada experimento se realizó 10 veces y sus resultados fueron promediados para el resultado final.

6.1.1. Evaluación del método de comparación de espigas

Una vez implementado el método de comparación de espigas se procedió a obtener la precisión, con el objetivo de conocer que tanto se parecen entre si las espigas. Sea A una nube de 300 puntos y sea B una nube de puntos obtenida de A con un porcentaje p de inserciones y borrados. Para cada espiga $s \in A$ se compara contra toda espiga $s^{'} \in B$. Los porcentajes de inserciones y borrados utilizados para este experimento fueron de 0, 5, 10, 20, 30 y 50 %.

Un siguiente experimento se realizó con objetivo de obtener la cantidad de espigas que se recuperan ante operaciones de inserción y eliminación. Utilizando información espacial de los puntos basta con encontrar tres pares de puntos emparejados entre dos conjuntos de puntos para determinar si los dos conjuntos son el mismo. De los tres puntos se puede obtener información geométrica como los ángulos que forman o la relación de distancia que mantienen los puntos, si esta información es la misma en ambos conjuntos entonces se consideran iguales. En el experimento se considera el conjunto de puntos como encontrado si se logra un emparejamiento de tres puntos o más, en caso contrario se considera como no encontrado. Para esto se generaron mil conjuntos de puntos donde cada conjunto contiene 300 puntos. Estos mil conjuntos de puntos forman la base de datos, la consulta se creó eligiendo de forma aleatoria un conjunto de puntos y realizando inserciones y eliminaciones con porcentajes de 0, 5, 10, 20, 30 y 50 %.

Otro aspecto a evaluar en el método de recuperación de espigas es la tolerancia que presenta ante el ruido. El experimento realizado consistió en generar una nube de 300 puntos y a partir de esta nube generar una nueva con ruido, donde cada punto fue movido un pixel en cualquier dirección de manera aleatoria.

Lo siguiente que se realizó fue probar la robustez que ofrece esta representación en la práctica, para ello se utilizaron imágenes de la base de datos Calthec101 (Li Fei-Fei *et al.*, 2004) de las cuales se obtuvieron sus respectivos conjuntos de puntos mediante la utilización del algoritmo SIFT, este procedimiento se realizó en python 2.7 utilizando la librería OpenCV la cual incorpora diversas funciones para visión por computadora. De la base de datos de imágenes se seleccionó una de manera arbitraria, la imagen fue rotada cada 30 grados y se almacenó como una nueva imagen. La imagen original se comparó contra las imágenes rotadas para determinar si las espigas se mantenían bajo la operación de rotación, la cual es más sencilla que la inserción y eliminación de puntos.

Hasta este punto, las espigas se obtienen de un grafo plano generado al aplicar la triangulación de Delaunay sobre el conjunto de puntos. Por lo que se propuso utilizar otro grafo plano, el cual se obtiene utilizando la técnica de los k-vecinos, donde para cada punto se agrega una arista con los k puntos que son su vecino más cercano. De esta manera se generan espigas de un mismo tamaño. El tamaño de las espigas se

vuelve una variable por lo que es necesario determinar con cual valor de k se obtienen mejores resultados.

Lo primero que se realizó fue obtener el promedio del tamaño de las espigas obtenidas del grafo generado mediante triangulación de Delaunay. Este promedio fue utilizado como referencia para los valores de k. Una vez obtenido este promedio, se evalúan la cantidad de espigas encontradas y se comparan los valores de k utilizados, se utilizó $4 \leq k \leq 8$.

Una vez encontradas las características de las espigas se seleccionó un método de indexado para determinar cuales eran los tiempos de consulta que se pueden logran. La decisión fue utilizar una estructura de datos tipo árbol, la selección fue el árbol BK que como se describió en el marco teórico es aplicable para funciones de distancia discretas. Para realizar el experimento primero se generaron 1000 nubes de 300 puntos, después se extrajeron las espigas para ser indexadas en el árbol. Se seleccionó una nube dentro del conjunto de manera aleatoria sobre la cual se insertaron y eliminaron distintos porcentajes de puntos. Por último se realizó la consulta de cada una de las espigas de la nube modificada sobre el índice.

6.1.2. Representación de espigas mediante cadenas binarias

Hasta este punto se conocen dos problemas presentes al utilizar la representación de espigas. El primer problema es la alta complejidad de la función de distancia y el segundo problema es la sensibilidad de las espigas al ruido. Se propone otro método para representar espigas utilizando cadenas binarias con el objetivo de solucionar los dos problemas antes mencionados. El método para generar la representación binaria a partir de una espiga es como sigue.

Sea s una espiga de tamaño m donde $\alpha_1, \ldots, \alpha_m$ representan los ángulos entre sus aristas adyacentes. La espiga s se transforma a una representación vectorial dada por el vector $v =< v_1, \ldots, v_m >$ de dimensión m. Cada ángulo α_i corresponde con el componente v_i, en la *figura 29* se muestra la conversión de la espiga iniciando desde el ángulo más pequeño y siguiendo el sentido de las manecillas del reloj. Una vez generado el vector se normalizan sus componentes a un valor en el intervalo $[0, 1]$, donde 1 equivale a los 360 grados. Después de normalizar los vectores se seleccionan

l dígitos decimales significativos. Se recorre el punto hacia la derecha hasta tener los l dígitos en la parte entera y se descarta la parte decimal. Hasta este punto las espigas están representadas por vectores de números enteros, lo siguiente es transformar el vector a la representación mediante una cadena binaria.

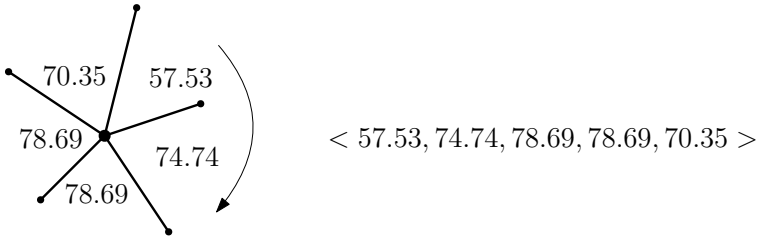

Figura 29. Representación mediante espigas(izquierda) y representación vectorial(derecha).

El valor de los componentes de los vectores se puede encontrar en el intervalo $[0, 10^l]$, este intervalo se divide en t intervalos. Sea t_i el intervalo donde se encuentra el valor del componente v_i, se genera una cadena binaria de tamaño t donde cada bit en $[t_1, t_i]$ se asigna 1 y 0 para los bits restantes. La asignación de bits se realiza de derecha a izquierda. La representación mediante cadena binaria que corresponde al vector v estará formada por la concatenación de las cadenas binarias de cada uno de sus componentes. Para calcular la similitud entre cadenas binarias se utiliza la distancia de Hamming. El proceso completo para transformar una espigas a su representación mediante cadenas binarias se ilustra en la *figura 30*.

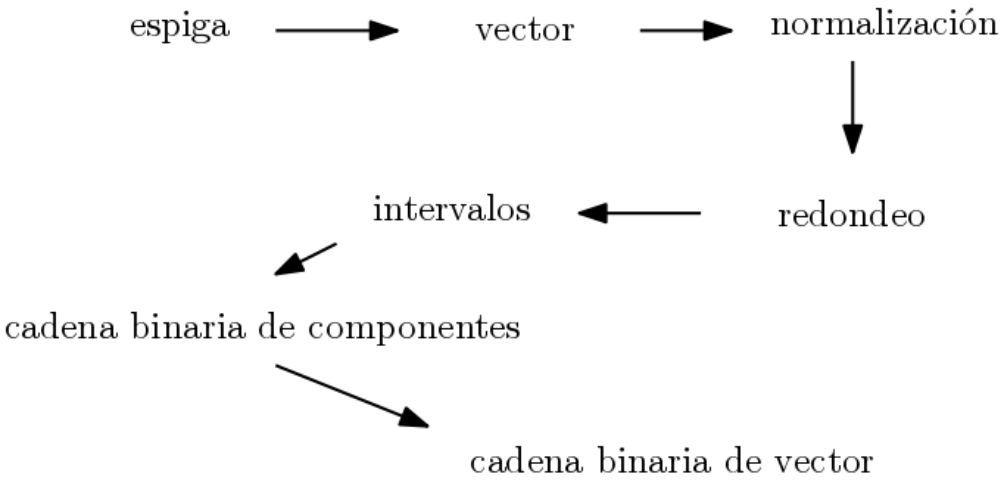

Figura 30. Metodología para conversión de espiga a cadena binaria.

La tolerancia al ruido se logra utilizando los intervalos, si los componentes de dos vectores que se encuentran en un mismo intervalo se consideran similares, en la *figura*

31 se muestra que los componentes v_i y u_i se encuentran en el mismo intervalo. Si los componentes de dos vectores se encuentran en distintos intervalos entonces no se consideran iguales debido a que sus cadenas binarias tendrán un bit de diferencia, en la *figura 32* se muestra un ejemplo de componentes cuyo valor se encuentra en intervalos distintos. El experimento realizado para obtener la tolerancia al ruido se realizó utilizando dos dígitos, 32 intervalos y tamaños de espiga de 4,6,8 y 10.

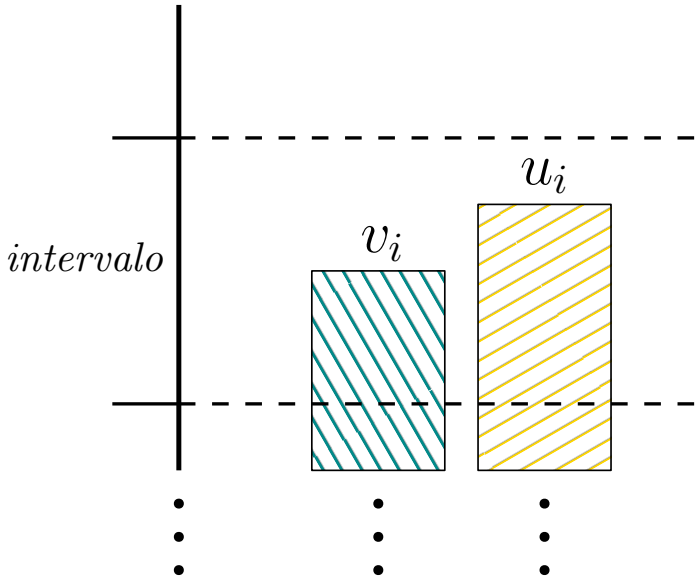

Figura 31. Los componentes de dos vectores se consideran iguales si se encuentran en un mismo intervalo.

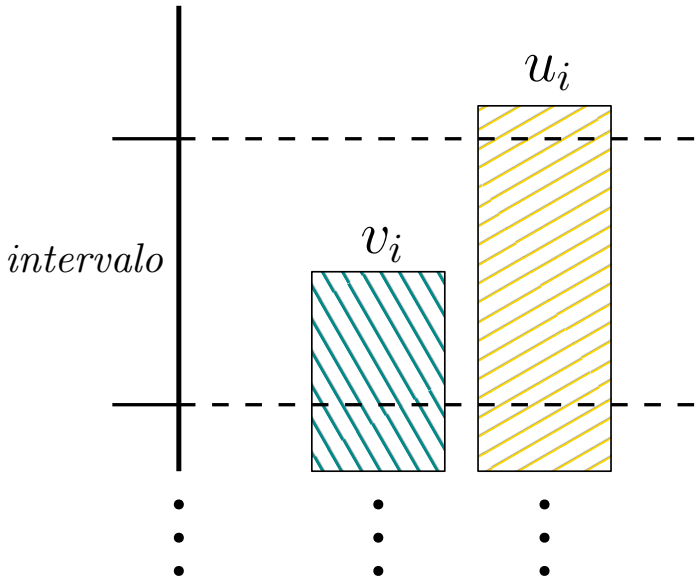

Figura 32. Los componentes de dos vectores no se consideran iguales si se encuentran en distintos intervalos.

Utilizando la idea de cadenas binarias y tomando en cuenta la relación de los ángulos de una espiga surge la idea de utilizar una matriz de números binarios para almacenar esa relación. La matriz de números binarios que se genera se reduce luego a una representación de cadena binaria. El procedimiento para generar la representación es el siguiente. Para una espiga s de tamaño k se genera una matriz m de tamaño $k \times k$. Una vez que se ha creado la matriz se compara cada ángulo con los restantes, se inicia con el ángulo más pequeño y se recorre la espiga ya sea en sentido de las manecillas del reloj o sentido contrario. Sea α_i el ángulo actual, se realiza la comparación contra los ángulos $\alpha_{i+1}, \ldots, \alpha_k$, si el ángulo α_i es mayor que el ángulo α_{i+1} se asigna un 1 en la matriz en la posición $m_{i,i+1}$ o 0 en caso contrario. En la *figura 33* se muestra que al comparar los ángulos 70.35 y 78.69 se asigna un 0 en la matriz en la celda $m_{2,3}$, donde 2 es la posición del ángulo 70.35 siguiendo la dirección contraria a las manecillas del reloj. Como cada ángulo se compara con los siguientes, solamente la matriz triangular superior contiene información de la relación entre los ángulos. La matriz triangular superior se utiliza para genera la representación binaria, se recorre de arriba hacia abajo y de izquierda a derecha. Los dígitos más significativos en la cadena binaria corresponden a los elementos más arriba y a la izquierda de esta matriz triangular superior. Debe notarse que la primera fila en la matriz se descarta debido que al iniciar con el ángulo menor el resultado de comparar contra los demás ángulos será cero, un ejemplo de creación de cadena binaria a partir de la matriz triangular superior se muestra en la *figura 34*. Para probar la tolerancia de está técnica ante el ruido se realizó un experimento con niveles de ruido entre 0 y 5 con tamaños de espiga de $4, 6, 8,$ y10.

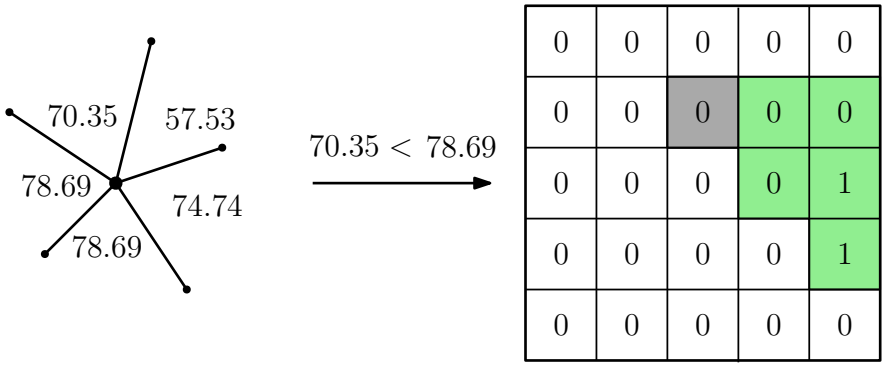

Figura 33. Representación mediante espigas(izquierda), representación matricial(derecha).

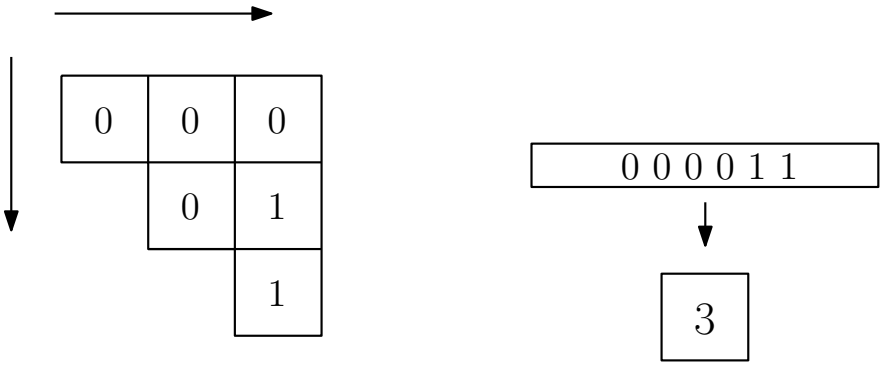

Figura 34. Representación mediante matriz triangular superior(izquierda), representación binaria(derecha).

6.2. Resultados

6.2.1. Método de comparación de espigas

El método de recuperación utilizando distancia entre espigas tiene una precisión del 100 %, es decir no se presentan falsos positivos. Sin embargo, conforme se insertan y eliminan puntos la cantidad de espigas que se mantienen tiende a disminuir. De acuerdo al experimento realizado se encontró que se pueden recuperar conjuntos de puntos con hasta un 25 % de inserciones y borrados. Además es posible recuperar conjuntos de puntos en un 80 % de los casos con un 30 % de inserciones y borrados. En la *figura 35* se muestra con que porcentaje se encuentran las espigas suficientes para distintos porcentajes de inserciones y borrados.

Del experimento realizado para determinar la unicidad de las espigas se encontró que no se parecen entre si, esto concuerda con el primer experimento donde la cantidad de falsos positivos encontrados fue de 0, por lo que la idea de utilizar estructuras de índices invertidos fue descartada.

De acuerdo al experimento realizado para determinar la tolerancia de las espigas al ruido, se encontró que las espigas no presentan tolerancia alguna, basta con mover los puntos una unidad para que el valor de los ángulos de las espigas sea diferente. Al momento de realizar el experimento utilizando imágenes se encontró que el algoritmo para extraer características presentaba una variación en las coordenadas de los puntos regresados, esto es comparable a presentar ruido en el conjuntos de puntos por lo que el método utilizando espigas no era aplicable y es necesario mejorar el método de

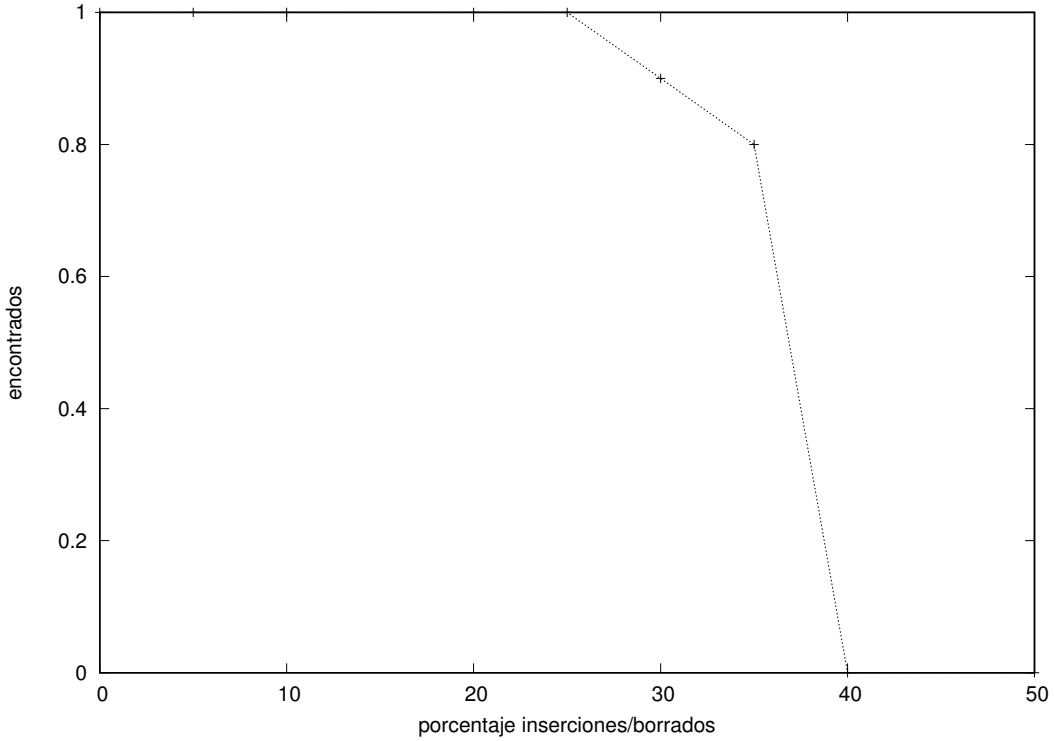

Figura 35. Espigas encontradas para una nube de puntos como consulta.

manera que presente mayor tolerancia ante ruido.

La idea de utilizar espigas obtenidas de un grafo distinto surge con el objetivo de obtener características de las espigas y tener más control al momento de crearlas. Se propone utilizar un grafo que genera espigas de un mismo tamaño y se crea utilizando la técnica de los k-vecinos. Surge la interrogante de determinar que tamaños de espigas daban mejores resultados ante las operaciones de interés. Lo primero a realizar fue calcular el tamaño promedio de las espigas que se obtienen al utilizar un grafo generado mediante la triangulación de Delaunay. El conocer este promedio, permite establecer un rango de tamaños de espigas con los cuales trabajar. La *figura 36* muestra los tamaños mínimo, medio y máximo promedio de las espigas de cada una de las nubes. En la *tabla 2* se muestran los promedios generales de las espigas pertenecientes a las 1000 nubes del experimento.

La selección de tamaños de espigas para utilizar en la técnica de k vecinos fue de 4 a 8, utilizando como referencia el tamaño promedio general presentado anteriormente. Dos problemas se presentan dependiendo de la cantidad de aristas que forman

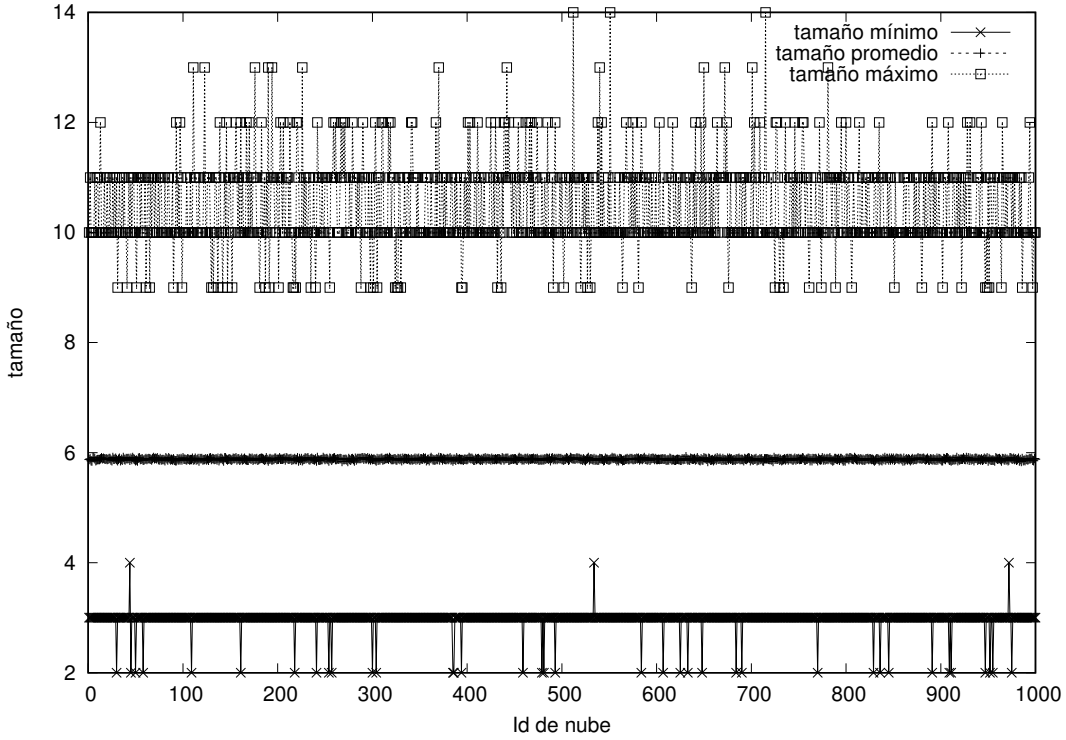

Figura 36. Resultados del grado promedio de las espigas.

Tabla 2. Promedio y varianza obtenidos

tamaño	**promedio**	**varianza**
mínimo	2.976	0.18821
promedio	5.88037	0.0150626
máximo	10.445	0.760904

las espigas. El primer problema es que el tamaño de las espigas esta directamente relacionado con el costo de la distancia y el segundo problema es que a mayor tamaño de espiga se reduce la tolerancia ante operaciones de inserciones y borrados. La reducción de la tolerancia ante inserciones y borrados se debe a que las espigas están formadas por una mayor cantidad de aristas y la probabilidad de que sean modificadas cuando se inserta o elimina algún punto es mayor. Como resultado de este experimento se encontró que las espigas de tamaño 4 y 5 presentan una mayor tolerancia ante inserciones y borrados. En la *figura 37* se muestra la comparación de casos encontrados para distintos porcentajes de inserciones y eliminaciones.

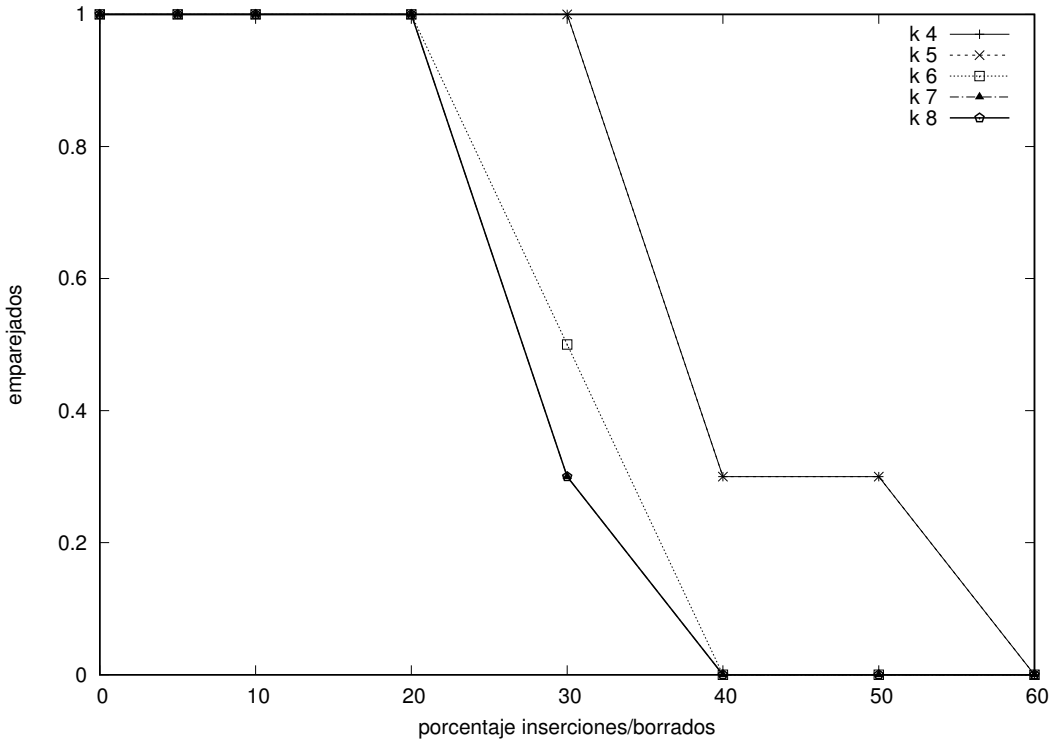

Figura 37. Comparativa de espigas encontradas para distintos tamaños de espigas.

6.2.2. Resultados de utilización de índices

El utilizar una estructura de índice reduce en gran medida los tiempos de consulta sobre todo cuando nuestra representación propuesta tiene un costo computacional muy alto, los resultados experimentales muestran que una consulta de búsqueda lineal utilizando distancia entre espigas toma en promedio 43.14 segundos mientras que al utilizar el índice el tiempo de consulta se ve reducido a 700 mili-segundos. Debe notarse que los tiempos de consulta son para una base de datos formada por 1000 nubes de 300 puntos, en la práctica las bases de datos están formadas por millones de nubes de puntos.

6.2.3. Resultados de representación mediante cadenas binarias

Al experimentar con el ruido en las nubes utilizando cadenas binarias, se encontró que se obtienen espigas suficientes para determinar similitud entre conjuntos de puntos con una precisión del 99 % utilizando espigas de tamaño 10 y con hasta 6 pixels de ruido. En la *figura 38* se muestra la precisión obtenida para los distintos tamaños de espigas y en la *figura 39* se muestra la exhaustividad lograda.

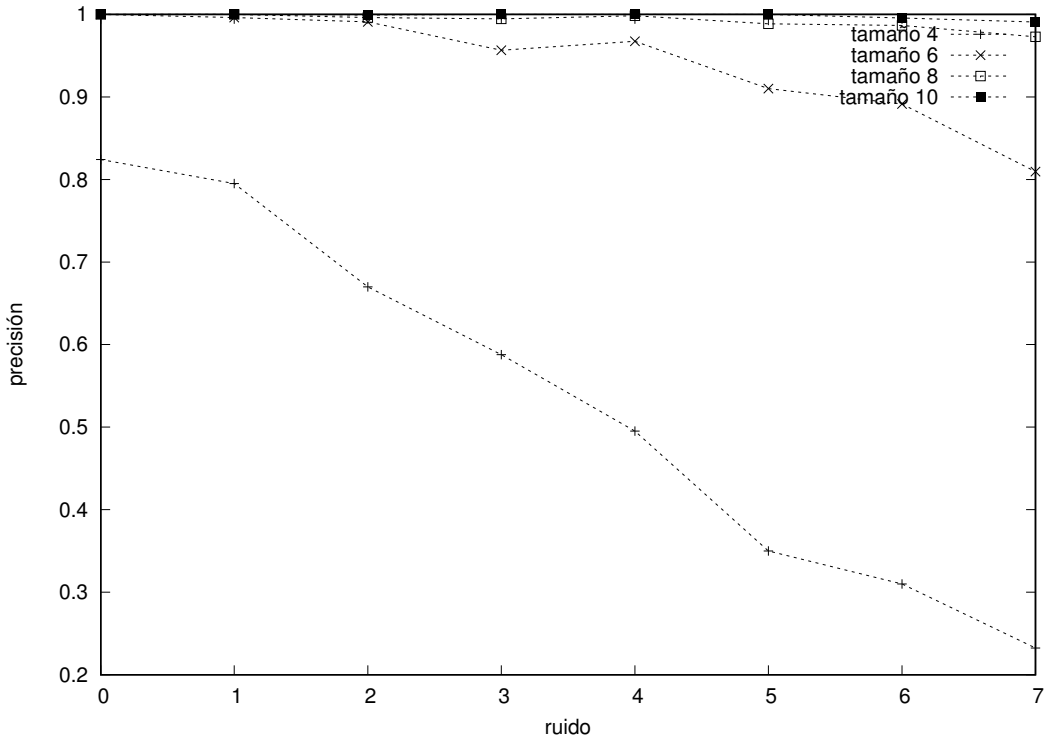

Figura 38. Precisión obtenida de utilizar una representación de vectores binarios para distintos tamaños de espigas.

Adicionalmente se calculó el tamaño promedio de las aristas de las espigas cuando varía su tamaño, también se calculó el cambio que sufren sus ángulos cuando se aplican distintos niveles de ruido, los resultados anteriores se presentan en tablas como anexos. En la *tabla 4*, se puede apreciar que a mayor tamaño de las espigas el ruido afecta en menor medida la diferencia entre los ángulos.

Como resultado del experimento de utilizar una matriz para generar una representación binaria, se obtuvo que presenta una buena precisión y exhaustividad al igual que la técnica anterior. Los resultados mostrados el la *figura 40* y *figura 41* muestran que si bien la exhaustividad que presenta es baja aún se recuperan la cantidad de elementos necesarios con una alta precisión. El tamaño de las espigas influye en gran medida en los resultados obtenidos debido a que tamaños de espiga pequeños de igual manera generan una representación binaria pequeña lo cual se traduce a una alta colisión en los códigos generados. Se complementa además con el resultado de un experimento anterior donde se determinó que espigas de mayor tamaño presentan mayor tolerancia ante el ruido.

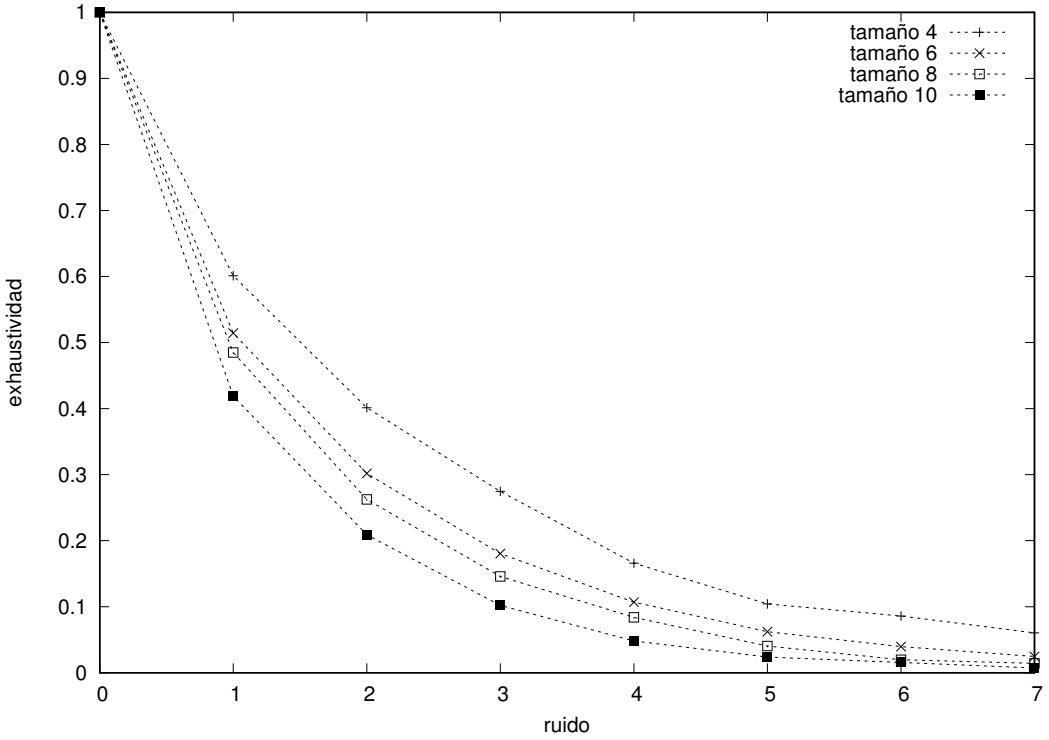

Figura 39. Exhaustividad obtenida de utilizar una representación de vectores binarios para distintos tamaños de espigas.

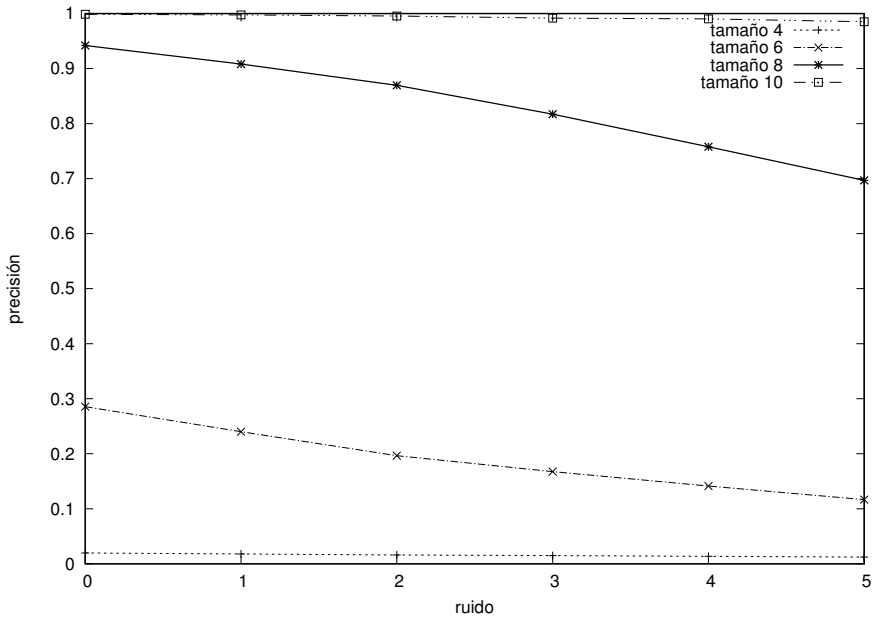

Figura 40. Precisión de la representación binaria para distintos tamaños de espigas y bajo distintos niveles de ruido.

Adicionalmente se realizó la evaluación de la tolerancia de esta representación ante inserciones y borrados, se obtuvo como resultado que hasta con 20 % de inserciones y borrados se encuentran las espigas suficientes con una precisión del 92 % utilizando

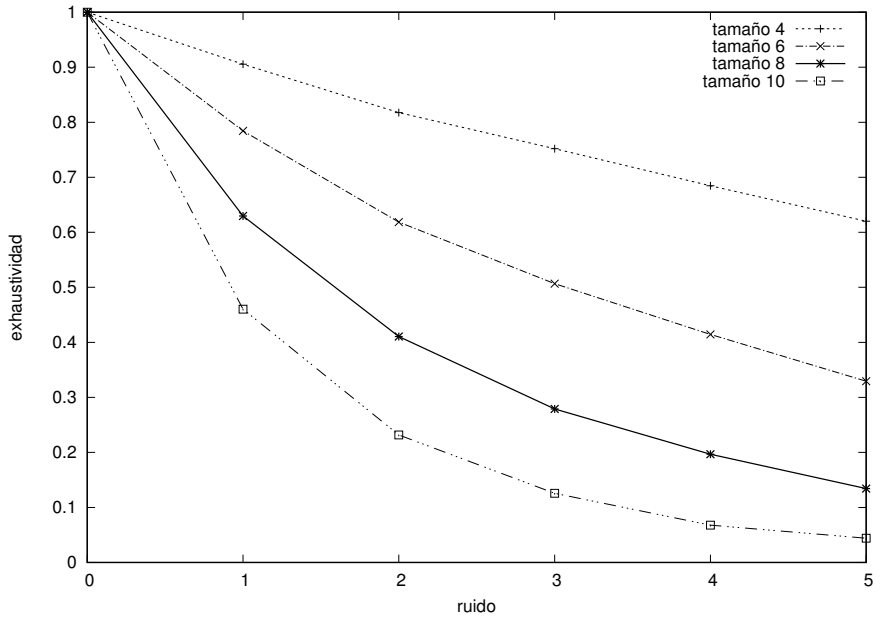

Figura 41. Exhaustividad de la representación binaria para distintos tamaños de espigas y bajo distintos niveles de ruido.

tamaños de espigas de 10. En la *figura 42* y *figura 43* se muestran los resultados de utilizar tamaños de espigas de 4,6,8 y 10.

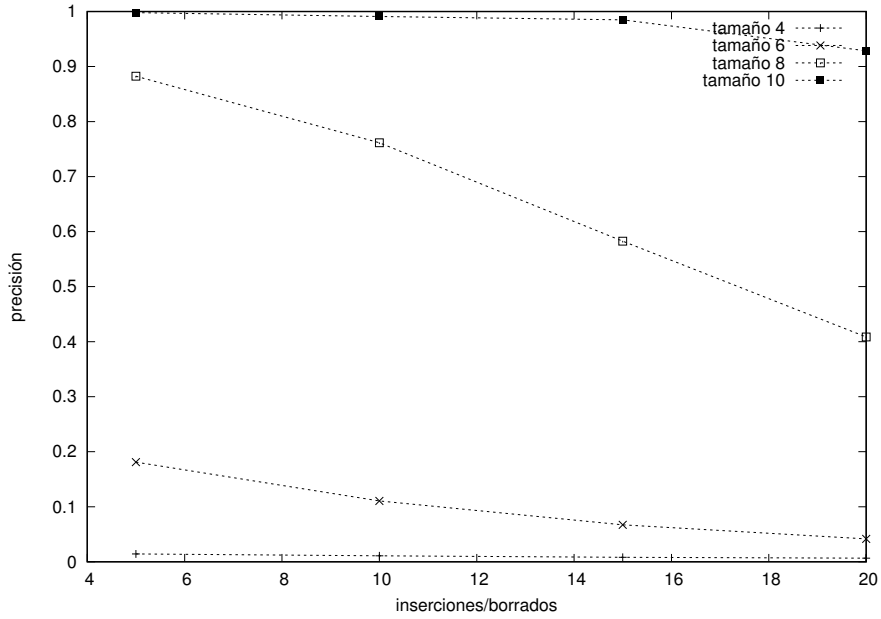

Figura 42. Precisión de la representación binaria para distintos tamaños de espigas y bajo distintos niveles de inserciones y borrados.

Como último experimento se compararon la precisión y la exhaustividad de las dos técnicas utilizando cadenas binarias, los resultados de la comparación se muestran en

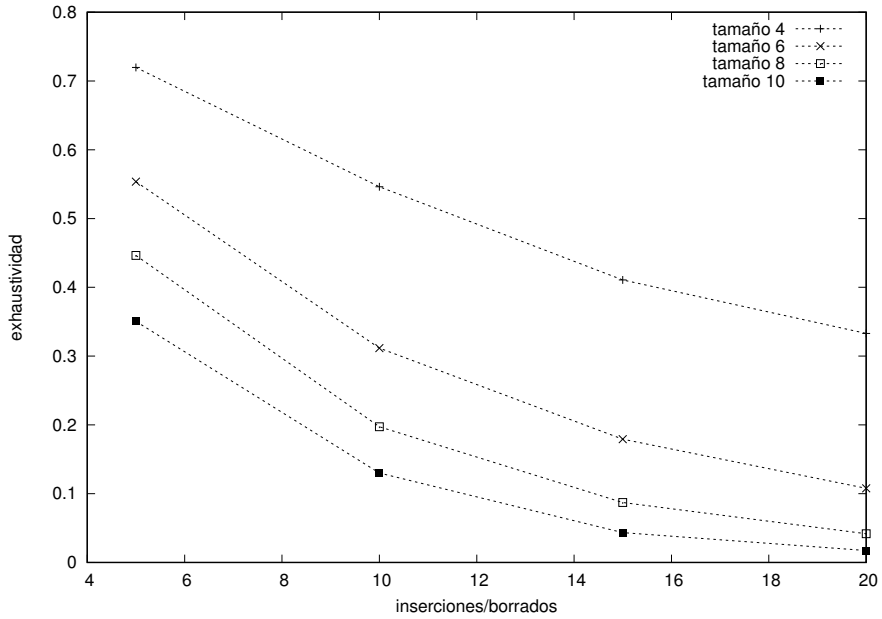

Figura 43. Exhaustividad de la representación binaria para distintos tamaños de espigas y bajo distintos niveles de inserciones y borrados.

la *figura 44* y *figura 45*, se puede apreciar que al utilizar la representación mediante matriz se obtiene una mejor precisión y al utilizar la representación mediante vectores se logra una mayor exhaustividad.

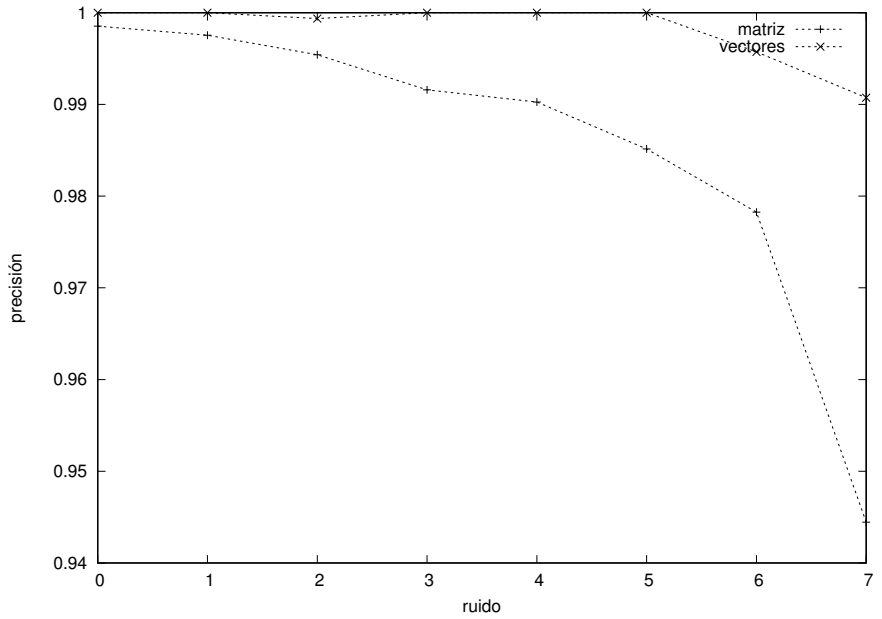

Figura 44. Comparación de la precisión obtenida al utilizar las representaciones binarias.

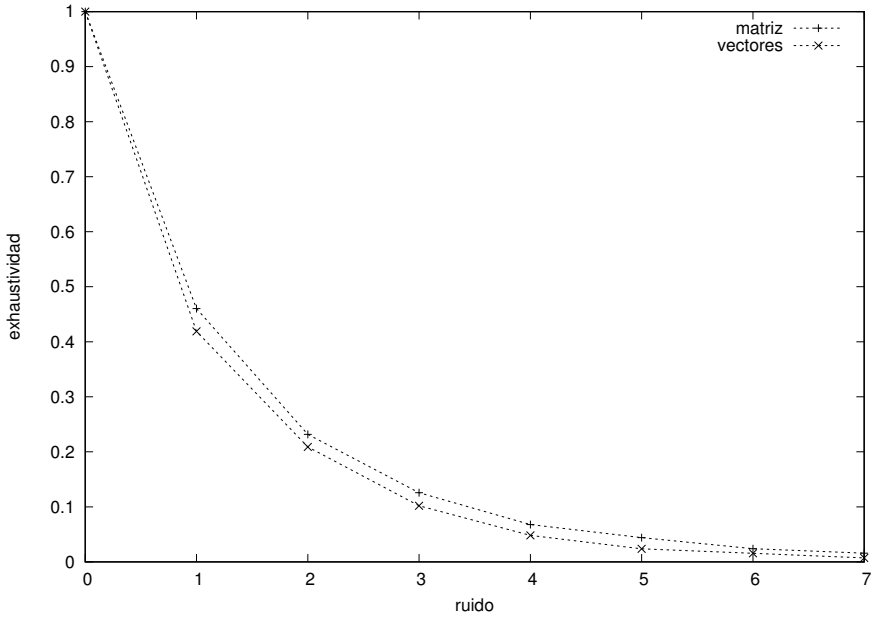

Figura 45. Comparación de la exhaustividad obtenida al utilizar las representaciones binarias.

Capítulo 7. Conclusiones

7.1. Resumen

En este trabajo de tesis se aborda el problema de recuperación de conjuntos de puntos y se enfoca en desarrollar una estructura que sea robusta ante transformaciones lineales, inserciones y eliminaciones y ruido en los conjuntos de puntos. Se propone una representación de espiga, con características interesantes, así como diferentes técnicas utilizando la representación propuesta que mejoran los resultados inicialmente obtenidos.

Se realizaron distintos experimentos con el objetivo de conocer características de la representación que pudieran ser utilizadas para obtener buenos resultados. Se realizaron experimentos utilizando una técnica de indexado con el objetivo de reducir los altos tiempos de consulta. Los resultados iniciales muestran que la representación de espigas es robusta ante las transformaciones lineales además de inserciones y eliminaciones, sin embargo, se encontró que la representación de espigas es sensible al ruido además de tener tiempos de consulta muy altos. Se identificó que los métodos de extracción de características de imágenes utilizados generan coordenadas con variación para una misma imagen. La variación generada es similar a la presencia de ruido por lo que no se obtuvieron buenos resultados para recuperación de imágenes multimedia. Debido a la sensibilidad al ruido de la representación y los altos tiempos de consulta se plantearon estrategias adicionales transformando la representación de espiga a una representación de cadena binaria. La representación mediante cadenas binarias mantiene la robustez ante inserciones y eliminaciones, además se logra recuperar nubes de puntos con niveles de ruido de hasta 6 pixels. Los tiempos de consulta fueron reducidos en gran medida debido a que la función de distancia tiene menor complejidad computacional.

7.2. Conclusiones

A continuación se presentan las conclusiones obtenidas durante el desarrollo de este trabajo de tesis:

- La representación de espigas es robusta ante operaciones afines y ante inserciones y eliminaciones de puntos.
- La complejidad de la función de distancia es muy alta, aún con técnicas de indexado los tiempos de consulta son altos.
- La representación mediante espigas es sensible al ruido, no se recomienda utilizar en problemas donde se presente ruido en los datos.
- La representación de espigas mantiene información local de los conjuntos de puntos que puede ser utilizada para transformar a otra representación.
- El transformar la representación de espigas a una representación de cadenas binarias soluciona las desventajas que presentan las espigas.
- Las representación de cadenas binarias utilizan menor cantidad de información que las espigas, lo cual se traduce a un menor consumo de memoria.

7.3. Trabajo futuro

A continuación se presentan aspectos para temas y trabajos que pueden ser desarrollados a partir de este tema de tesis:

- Recuperación de imágenes multimedia utilizando representación de cadenas binarias para características obtenidas de los distintos métodos presentes en la literatura.
- Utilización de técnicas Hash para indexado de las cadenas binarias, con el objetivo de reducir los tiempos de consulta a un tiempo constante.
- Construcción de un sistema para usuarios finales basado en el estilo arquitectónico REST, utilizando índices en memoria secundaria y que presente alta escalabilidad.

Literatura citada

Baeza-Yates, R., Cunto, W., Manber, U., y Wu, S. (1994). Proximity matching using fixed-queries trees. En: M. Crochemore y D. Gusfield (eds.), *Combinatorial Pattern Matching*, Berlin, Heidelberg. Springer Berlin Heidelberg, pp. 198–212.

Bahroun, S., Gharbi, H., y Zagrouba, E. (2014). Local query on satellite images based on interest points. En: *2014 IEEE Geoscience and Remote Sensing Symposium*, jul. IEEE, pp. 4508–4511.

Bozkaya, T. y Ozsoyoglu, M. (1997). Distance-based indexing for high-dimensional metric spaces. *SIGMOD Rec.*, **26**(2): 357–368.

Brin, S. (1995). Near neighbor search in large metric spaces. pp. 574–584.

Burkhard, W. A. y Keller, R. M. (1973). Some approaches to best-match file searching. *Commun. ACM*, **16**(4): 230–236.

Chávez, E., Marroquín, J. L., y Navarro, G. (2001a). Fixed Queries Array: A Fast and Economical Data Structure for Proximity Searching. *Multimedia Tools and Applications*, **14**(2): 113–135.

Chávez, E., Navarro, G., Baeza-Yates, R., y Marroquín, J. L. (2001b). Searching in metric spaces. *ACM Computing Surveys*, **33**(3): 273–321.

chun Wang, A. L. (2003). An industrial-strength audio search algorithm. En: *Proceedings of the 4 th International Conference on Music Information Retrieval*.

Ciaccia, P., Patella, M., Rabitti, F., y Zezula, P. (1997). Indexing metric spaces with m-tree. En: *PROC. QUINTO CONVEGNO NAZIONALE SEBD*. pp. 67–86.

Cormen, T. H. (2009). *Introduction to algorithms*. MIT press.

Hernández, E. A., Alonso, M. A., Chávez, E., Covarrubias, D. H., y Conte, R. (2017). Robust polygon recognition method with similarity invariants applied to star identification. *Advances in Space Research*, **59**(4): 1095–1111.

Indyk, P., Motwani, R., Raghavan, P., y Vempala, S. (1997). Locality-preserving hashing in multidimensional spaces. En: *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, New York, NY, USA. ACM, STOC '97, pp. 618–625.

Li Fei-Fei, Fergus, R., y Perona, P. (2004). Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. En: *2004 Conference on Computer Vision and Pattern Recognition Workshop*. IEEE, pp. 178–178.

Maltoni, D., Maio, D., Jain, A. K., y Prabhakar, S. (2009a). Fingerprint Analysis and Representation. En: *Handbook of Fingerprint Recognition*. Springer London, London, pp. 97–166.

Maltoni, D., Maio, D., Jain, A. K., y Prabhakar, S. (2009b). *Handbook of Fingerprint Recognition*. Springer London. London.

Mico, L., Oncina, J., y Vidal, E. (1992). An algorithm for finding nearest neighbours in constant average time with a linear space complexity. En: *Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol.II. Conference B: Pattern Recognition Methodology and Systems*. IEEE Comput. Soc. Press, pp. 557–560.

Navarro, G. (2002). Searching in metric spaces by spatial approximation. *The VLDB Journal*, **11**(1): 28–46.

Ning Liu, Yilong Yin, y Hongwei Zhang (2005). A Fingerprint Matching Algorithm Based On Delaunay Triangulation Net. En: *The Fifth International Conference on Computer and Information Technology (CIT'05)*. IEEE, pp. 591–595.

Ramírez-Chacón, M., Hidalgo-Silva, H., y Chávez, E. (2018). Indexing and searching point clouds. **34**: 3349–3358.

Ruiz, E. V. (1986). An algorithm for finding nearest neighbours in (approximately) constant average time. *Pattern Recognition Letters*, **4**(3): 145–157.

Shasha, D. y Wang, T.-L. (1990). New techniques for best-match retrieval. *ACM Trans. Inf. Syst.*, **8**(2): 140–158.

Tuytelaars, T. y Mikolajczyk, K. (2008). Local invariant feature detectors: A survey. *Found. Trends. Comput. Graph. Vis.*, **3**(3): 177–280.

Wang, J., Shen, H. T., Song, J., y Ji, J. (2014). Hashing for similarity search: A survey. *CoRR*, **abs/1408.2927**.

Wang, J., Liu, W., Kumar, S., y Chang, S. (2016). Learning to hash for indexing big data—a survey. *Proceedings of the IEEE*, **104**(1): 34–57.

Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. En: *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics, SODA '93, pp. 311–321.

Yvinec, M. (2018). {2D} Triangulation. En: *{CGAL} User and Reference Manual*. CGAL Editorial Board, cuarta edición.

Zaeri, N. (2011). Minutiae-based Fingerprint Extraction and Recognition. En: *Biometrics*. InTech.

Zezula, P., Amato, G., Dohnal, V., y Batko, M. (2006). *Similarity Search The Metric Space Approach*, Vol. 32 de *Advances in Database Systems*. Kluwer Academic Publishers. Boston, pp. 67–99.

Anexo

Tabla 3. Valores mínimo, promedio y máximo de tamaños de aristas de las espigas.

tamaño de espiga	distancia mínima	desv. estándar	distancia promedio	desv. estándar	distancia máxima	desv. estándar
4	30.4354	0.896666	50.5518	0.754606	67.8049	0.970384
5	30.2321	0.993702	55.514	0.980164	76.2736	1.36679
6	30.3018	0.91625	60.5087	0.850243	84.675	1.0772
7	30.4046	0.840277	64.8372	0.970796	91.7284	1.4572
8	30.186	1.09086	69.1784	1.11291	99.0111	1.73295
9	30.0704	1.0152	72.8973	1.23441	105.208	1.70908
10	30.2114	0.977598	77.0015	1.01611	111.622	1.54703

Tabla 4. Promedio de la diferencia en los ángulos entre la misma espiga bajo distintos niveles de ruido

tamaño de espigas	variación	ruido				
		1	2	3	4	5
4	mínimo	0.421767	0.627815	0.734176	0.798297	0.835722
	promedio	7.70444	10.9757	17.8702	19.6917	24.9084
	máximo	14.8721	20.9862	34.2869	37.4206	47.2474
5	mínimo	0.328101	0.543889	0.655553	0.736202	0.779359
	promedio	7.15139	12.9229	15.4281	22.7018	24.3285
	máximo	15.6424	28.3471	33.4492	49.4842	52.6943
6	mínimo	0.2638	0.45374	0.596317	0.679403	0.734321
	promedio	7.22136	10.8655	16.6561	21.3105	23.9467
	máximo	17.4123	25.617	38.9296	50.7704	56.6464
7	mínimo	0.224006	0.412487	0.537504	0.628885	0.677286
	promedio	7.14324	11.4286	18.0672	21.0232	23.5599
	máximo	18.801	29.7404	46.5175	53.9648	60.1486