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

Maestría en Ciencias en Ciencias de la Computación

Identificación de nubes de puntos utilizando etiquetado parcial

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de Maestro en Ciencias

Presenta:

Arturo Sair García Amador

Ensenada, Baja California, México

2021

Tesis defendida por

Arturo Sair García Amador

y aprobada por el siguiente Comité

Dr. Edgar Leonel Chávez González

Director de tesis

Dra. Mónica Tentori Espinoza

Dra. Rufina Hernández Martínez

Dr. Pedro Gilberto López Mariscal

Coordinador del Posgrado en Ciencias de la Computación

Dr. Pedro Negrete Regagnon

Director de Estudios de Posgrado

Resumen de la tesis que presenta Arturo Sair García Amador como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación.

Identificación de nubes de puntos utilizando etiquetado parcial

Resumen aprobado por:

Dr. Edgar Leonel Chávez González
Director de tesis

La identificación de nubes de puntos es un problema central en computación, con aplicaciones que van desde identificadores biométricos hasta aplicaciones en astronáutica y medicina. RANSAC es esencialmente el único algoritmo que existe para emparejar una nube de puntos con un modelo específico. En esta tesis se proponen modificaciones al algoritmo RANSAC basadas en la siguiente hipótesis de trabajo: los puntos correspondientes de un modelo tienen una mayor densidad relativa, en comparación con puntos que corresponden a ruido que normalmente se encuentran más dispersos. Esta hipótesis se traduce a tres propuestas algorítmicas descritas en el trabajo. Estas nuevas versiones del algoritmo son presentadas como: DD-RANSAC, siglas del término en inglés 'Density Driven RANSAC'. Los tres algoritmos derivados de esta investigación fueron medidos y comparados con el algoritmo base de RANSAC en términos de eficiencia y trabajo, con tres modelos diferentes; cada uno con diferente dimensionalidad. A partir de los resultados obtenidos, se concluye que la heurística de procesar primero las zonas con mayor densidad resulta efectiva en términos de eficiencia y trabajo del algoritmo, siempre y cuando los puntos tengan coherencia espacial.

Palabras clave: Emparejamiento, Nube de puntos, RANSAC, índice, Densidad

Abstract of the thesis presented by Arturo Sair García Amador as a partial requirement to obtain the Master of Science degree in Computer Science..

Point cloud registration via partial labelling

Abstract approved by:

Dr. Edgar Leonel Chávez González
Thesis Director

Point cloud registration is a central problem in computing, with applications from awide range of biometric identifiers to astronautics and medicine. RANSAC is essentially the only existing algorithm to match a point cloud to a specific model. This work proposes different modifications to the RANSAC algorithm, based on the following hypothesis: Inlier points model has higher relative density than the usually more dispersed outliers. The above work hypothesis translates to three algorithmic proposals described herein. We present these new versions of RANSAC as 'Density Driven RANSAC' or DD-RANSAC. We compared three algorithms derived from this thesis with the base RANSAC algorithm for efficiency and performance, testing three different models, each with different dimensionality. From the results of the experiments, we conclude our heuristic, i.e., processing first areas with the highest density, is effective in terms of efficiency and performance if the points have spatial coherence.

Keywords: Registration, Point Cloud, RANSAC, Index, Density

Dedicatoria

A mi familia...

este triunfo es gracias a ustedes.

Agradecimientos

Al Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE), gracias por confiar en mi.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT), gracias por brindarme el apoyo económico para realizar mis estudios de maestría. No. de becario: 19264494

Dr. Edgar Chávez, gracias por el tiempo dedicado y por todas sus enseñanzas. Su trabajo y disciplina son un ejemplo para mi.

Dra. Mónica Tentori y Dra. Rufina Hernández, gracias por las observaciones y aportaciones en esta tesis.

A mis compañeros de posgrado que compartimos momentos y circunstancias muy especiales estos últimos años.

A todas las personas que hicieron posible esta victoria, *gracias*.

Tabla de contenido

Página

Tabla de contenido (continuación)

Lista de figuras

Figura Página

Lista de figuras (continuación)

Lista de tablas

Tabla Página

Capítulo 1. Introducción

En el siglo XIX el daguerrotipo[1] se utilizaba para reconocer delincuentes o para descartar algún sospechoso en los departamentos de policía de aquella época. Se necesitaban alrededor de cuatro horas para tomar una imagen, revelarla, compararla con las imágenes previamente guardadas y arrojar un resultado. Aunque este método ya era innovador para su época, tenía dos problemas, requerir un tiempo de exposición de tres a quince minutos y el ángulo con el que se tomaban las imágenes siempre era de frente. Esto dificultaba la tarea de reconocimiento para el equipo encargado. Muchos sospechosos podían fingir algún gesto con el objetivo de confundir al equipo y eludir los castigos más severos por recurrencia. También existían casos donde las sombras y luces diferían con las imágenes registradas, por lo tanto, se ocultaban rasgos importantes que dejaban en libertad a prófugos o atenuaban la pena de reincidentes.

De acuerdo con Maltoni *et al.* (2009), en 1880 Alphonse Bertillon, antropólogo e integrante de la Prefectura de París, propuso la 'antropometría'; un procedimiento para la autentificación[2] de personas sospechosas, este consistía en una medición precisa de todos los huesos, la cabeza, marcas individuales, tatuajes, cicatrices y características personales del sospechoso. Este método fue llamado 'El Bertillonaje' y obtuvo muchos aciertos porque podía distinguir entre individuos, aún disfrazados, maquillados o con gestos fingidos, sin embargo, el método fue puesto en duda cuando se encontraron dos personas distintas con las mismas medidas de Bertillonaje. En 1892 apareció un método nuevo que resolvía el problema por medio de la identificación de huellas dactilares. Propuesto por el británico Francis Galton y posteriormente mejorado por el policía argentino Juan Vucetich. Poco después, en 1893, la Oficina del Ministerio del Interior del Reino Unido aceptó que no había dos personas que tuvieran las mismas huellas dactilares, por esta razón, muchos de los principales departamentos de policía vieron el potencial del uso de huellas dactilares como método preciso para la identificación de infractores y reincidentes que usaban un alias para evadir las penas más severas; ya que eran reservadas para los reincidentes.

La identificación por huellas dactilares encontró gran demanda en la medicina fo-

[1] Procedimiento fotográfico que utilizaba una placa metálica para plasmar las imágenes captadas. Fue desarrollado y perfeccionado por Louis Daguerre y ampliamente difundido en 1839.

[2] Autentificación y autenticación son empleados como sinónimos; este proceso recupera un solo registro de la base de datos para compararlo contra una consulta o muestra. El resultado puede ser positivo si ambos registros coinciden o negativo si difieren.

rense; al analizar las marcas dejadas en diferentes objetos de una escena del crimen, además, dicha demanda llevo a otro problema; el de la identificación[3], donde una huella 'A' se compara con todas las huellas almacenadas hasta encontrar una 'B' igual. El problema que salió a relucir en poco tiempo fue que entre más grande es la colección de huellas almacenadas, mayor es el tiempo para obtener una respuesta. Para solucionar este problema, se hicieron muchos esfuerzos para hacer eficiente el proceso, ordenando los registros, pero era notorio que esta tarea resultaba cada vez más tardada, a pesar de los esfuerzos realizados. Por esta razón se iniciaron programas de investigación para automatizar y digitalizar esta tarea.

Digitalizar es una palabra muy interesante que parece reciente, del siglo XXI, pero adquiere su significado actual en los años 60´s del siglo pasado; cuando el desarrollo de la computación con microprocesadores se comercializa y sirve como herramienta útil en tareas que van desde el conteo de votos en elecciones hasta los cálculos previos de misiones espaciales. Digitalizar quiere decir, generar una versión discreta de algo, en otras palabras, es una copia numéricamente representada. Un dígito numérico es un símbolo único que se usa solo o en combinaciones, para representar números en un sistema numérico posicional, sin embargo, el significado actual de *dígito* proviene del latín 'digiti' que significa *dedos* y hace referencia al hecho de que los diez dedos de las manos corresponden a los diez símbolos del sistema numérico de base común diez.

En las últimas décadas, la preocupación por la seguridad, fraude y robo de identidad ha creado una gran demanda del uso de huellas dactilares y de otros datos como métodos de identificación y autentificación. La diferencia entre ambas podría parecer mínima, pero computacionalmente, la primera requiere de n registros a comparar, mientras que la segunda se realiza un único registro. Por otro lado, es importante mencionar que la manera de digitalizar una huella dactilar y la mayoría de datos biométricos es muy similar. Generalmente se utiliza un plano donde se representan las medidas importantes de cada rasgo biométrico formando un conjunto de puntos o coordenadas en el plano. Este conjunto de datos también es conocido como *nube de puntos*.

[3]El problema de identificación involucra comparar una consulta con toda la base de datos y recuperar aquellos registros que hacen 'match' con una consulta dada.

Además de la identificación y/o autentificación por biométricos, existe una gran variedad de problemas que pueden ser planteados como una nube de puntos. Por ejemplo, para identificar melodías o pistas de audio se requiere una ’huella sonora’ que caracteriza cada una. En astronáutica[4] se debe determinar la actitud[5] de los satélites, sondas o naves espaciales utilizando un mapa estelar como nube de puntos.

En el año 2000 se fundó la empresa que ahora conocemos como Shazam, con sede en Londres, dicha empresa ofrecía el innovador servicio de identificación de pistas musicales llamado ‘2580’, por ser el número a donde se llamaba para recibir el servicio. Cualquier persona podía marcar, desde un teléfono móvil para obtener información de alguna canción, mediante una *huella digital acústica*. A partir de un segmento breve de una pista de audio se buscaban sus posibles coincidencias en la base de datos y posteriormente el resultado llegaba en forma de mensaje de texto con el título de la canción y su autor. Con la llegada de los teléfonos inteligentes Shazam migró su servicio y creo una aplicación donde se ofrecía el mismo servicio que permite identificar pistas de audio, pero con la diferencia de que ahora las bases de datos son más robustas y se incluyen películas y vídeos identificados por audio, casi al instante, en la misma aplicación (Wang, 2006).

En astronáutica el término ’actitud’ proviene del inglés ‘Attitude control’ . Este término se refiere al proceso de controlar la orientación de un vehículo aeroespacial con respecto a un marco de referencia inercial. Desde los inicios de la aviación se necesitaba aproximar la orientación con pequeñas marcas en las aeronaves para emparejarlas con el ’horizonte’ mientras volaban. Esto deja de ser práctico al volar de noche o con poca visibilidad; por esta razón fue necesario idear instrumentos precisos, capaces de brindar esta información. Para resolver el control de actitud se utilizó el giroscopio[6]. Con este instrumento se logró determinar la orientación o actitud de las aeronaves e incluso misiles en condiciones atmosféricas adversas. En los viajes espaciales, el principal problema con los instrumentos inerciales, como el giroscopio, es que se acumulan pequeños errores y esto requiere de actualizaciones periódicas basadas en alguna fuente externa. Estas correcciones periódicas se hacen gracias a la combinación de

[4] Teoría y práctica de la navegación más allá de la atmósfera terrestre por parte de objetos artificiales, ya sean tripulados o no tripulados.

[5] Orientación de un objeto con respecto a un sistema de referencia inercial.

[6] Es una rueda o disco giratorio donde el eje de rotación o eje de giro es libre de asumir cualquier orientación por sí mismo. Al girar, la orientación de este eje no se ve afectada por la inclinación o rotación del soporte, debido a la conservación del momento angular.

giroscopios y sensores de estrellas. El uso eficiente de los sensores de estrellas generalmente requiere una estimación de actitud inicial bastante precisa y posteriormente es refinada por los datos del sensor de estrellas. Cuando una nave espacial se somete a una maniobra desde una orientación a otra, los giroscopios proporcionan una medida precisa del cambio de orientación y una buena estimación inicial de la nueva actitud. Esta estimación se refina utilizando los datos del sensor de estrellas. La actitud refinada se utiliza como una medida de la verdadera orientación de la nave espacial y como dato para la actualización de los giroscopios, corrigiendo los errores acumulados (Wertz, 1978).

Aparentemente ya está solucionado el problema, sin embargo, existía un detalle a tomar en cuenta. La combinación de instrumentos inerciales y sensores de estrellas pueden llegar a ser muy pesada, es decir que llegan a tener una masa de hasta 20 Kg. En el caso de la navegación espacial, se necesita reducir la masa al mínimo posible porque en aquella década cada kilogramo puesto en órbita costaba entre 50 mil y 100 mil dólares[7] (Liebe *et al.*, 2004).

Conforme se incrementó el poder de cómputo en la década de los 80´s y 90´s, se desarrollaron los 'giroscopios estelares'. De acuerdo con Liebe *et al.* (2004), un giroscopio estelar es un rastreador de estrellas (Star Tracker)[8] que puede operar a altas velocidades de respuesta y actualización. Los giroscopios estelares pueden eliminar potencialmente la necesidad de giroscopios inerciales convencionales a bordo de una nave espacial, ya que reducen el tamaño y la masa del equipo, además el control de actitud puede llegar a ser mejor en precisión.

El problema del emparejamiento de nubes de puntos se ha intentado resolver de una u otra manera desde el siglo XIX. Actualmente el mismo problema ha visto soluciones innovadoras de acuerdo a las necesidades específicas del área de aplicación. Esperamos que esta tesis contribuya a dicha colección de soluciones, pero primero es necesario describir el problema detalladamente.

[7] De acuerdo con Jones (2018), hasta el año 2020 los costos de poner en órbita un satélite se han reducido a dos mil dólares por kilogramo y se espera que esta tendencia continué a la baja durante todo el siglo XXI.

[8] Un rastreador de estrellas es un dispositivo óptico que mide las posiciones de las estrellas mediante fotoceldas o cámaras. Los astrónomos han medido las posiciones de muchas estrellas con un alto grado de precisión, con estas medidas se generan mapas estelares que ayudan a un rastreador de estrellas montado en satélites o naves espaciales para determinar la actitud de la nave con respecto a las estrellas.

1.1. Emparejamiento de nubes de puntos

Definición 1 Datos biométricos *Los datos biométricos son las características fisiológicas, de comportamiento o de rasgos de personalidad medibles que se atribuyen a una sola persona. Por ejemplo: el reconocimiento por huellas digitales, reconocimiento facial, por retina o por iris, reconocimientos de voz, reconocimiento de escritura en teclado e incluso por el modo de caminar o de correr. Todo estos forman vectores que pueden ser representados como nubes de puntos, además, estos vectores tienen la peculiaridad de están vinculados a una única persona; por lo tanto son útiles como método de identificación.*

Definición 2 Nube de puntos *Una nube de puntos* $P = \{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_{|P|}\}$ *es un subconjunto finito del plano* $P \subset \mathbb{R}^2$.

Definición 3 Emparejamiento *Sean* $P = \{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_k, \ldots, \vec{P}_{|P|}\}$ *y* $Q = \{\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_k, \ldots, \vec{q}_{|Q|}\}$ *nubes de puntos, de tal manera que* $\|\vec{p}_i - \vec{q}_i\|_2 \leq \delta$ *para* $1 \leq i \leq k$ *y* $\|\vec{p}_i - \vec{q}_i\|_2 > \delta$ *para* $i > k$ *para cierta* $\delta \geq 0$, *reordenando los índices si es necesario. Decimos que* P *empata con* Q *bajo cierto* δ *con un soporte de tamaño* k.

El problema del emparejamiento de nubes de puntos consiste en encontrar una transformación que empate o que mantenga una correspondencia punto a punto de un subconjunto de una nube de puntos con otro subconjunto de una nube de puntos.

Las nubes de puntos pueden estar sujetas a inserciones, oclusiones y transformaciones (ejemplificado en la Figura 1). Las oclusiones hacen referencia a puntos que ocurren en P pero no ocurren en Q, en el caso contrario, las inserciones son puntos presentes en Q, pero no en P (nube de puntos original). En la Figura 1 podemos observar las inserciones como puntos de color azul y las oclusiones como puntos de color rojo.

Como ya mencionamos, además de inserciones y oclusiones, las nubes de puntos están sujetas a transformaciones, estas son aplicadas al conjunto de puntos completo, es decir que las transformaciones son operadores que actúan sobre cada uno de los

elementos $\{\vec{p}_1, \vec{p}_2, \vec{p}_3 ..., \vec{p}_n\}$, sin afectar la cardinalidad del conjunto (a diferencia de las inserciones y oclusiones). Existen diversos tipos de transformaciones, pero las de mayor interés para esta tesis son las *transformaciones afines* denotadas como funciones de tipo $f : \mathbb{R}^2 \to \mathbb{R}^2$.

En la ecuación 1 observamos una función $f(x, y)$ que depende de seis parámetros.

$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} r \\ s \end{pmatrix} \tag{1}$$

En la ecuación anterior, los parámetros (a, b, c, d, r, s) definen cualquier transformación afín, para cada vector (x, y), pero ahora las reescribiremos en términos de α, β y γ de la siguiente manera.

$$\alpha = \frac{a+b}{2} + i\frac{c-d}{2}$$
$$\beta = \frac{a-d}{2} + i\frac{c+b}{2}$$
$$\gamma = r + is$$

Una transformación afín también se puede representar en el plano complejo como una función $f(z)$, como se muestra en la Ecuación 2.

$$f(z) = \alpha z + \beta \bar{z} + \gamma \tag{2}$$

Donde $\alpha, \beta, \gamma \in \mathbb{C}$ y $|\alpha|^2 - |\beta|^2 = \det f \neq 0$ y $\bar{z}$ es el complejo conjugado de z. Si $\beta = 0$ entonces la función $f(z)$ corresponde a una transformación de similitud , en donde están incluidas las rotaciones, cambios de escala y las traslaciones.

Para aclarar la relación de los puntos representados el plano complejo $\mathbb{C}$ con los puntos en $\mathbb{R}^2$, decimos que cada $\vec{p} = (x, y)$ será identificado con el número complejo $p = x + iy$. De esta manera, P se convierte en un conjunto de números complejos. Para una nube de puntos $P = \{\vec{p}_1, \vec{p}_2, \vec{p}_3, \ldots, \vec{p}_{|P|}\}$ transformada por una función f se denota

como $f(P) = \{f(p_1), f(p_2), f(p_3), \ldots, f(p_{|P|})\}$. En otras palabras, dicha transformación opera sobre cada uno de los puntos de conjunto.

Ahora supongamos que $Q = f(P)$, es decir una nube de puntos *transformada*. Entonces, P y Q son dos nubes de puntos aparentemente diferentes, sin embargo, sabemos que existe una función que transforma cada punto de P a un punto en Q y viceversa. De este mismo modo, se dice que dos nubes de puntos P y Q *empatan* cuando existe una función que puede transformar una fracción de los puntos de P en puntos de Q bajo cierta tolerancia δ (ejemplo en la Figura 1). Cabe aclarar que cuando nos referimos a 'una fracción de puntos', es porque cabe la posibilidad de inserciones y oclusiones, dejando ciertos puntos sin ninguna relación en su contraparte, además, a esta 'fracción' de puntos coincidentes también se le llama *soporte*.

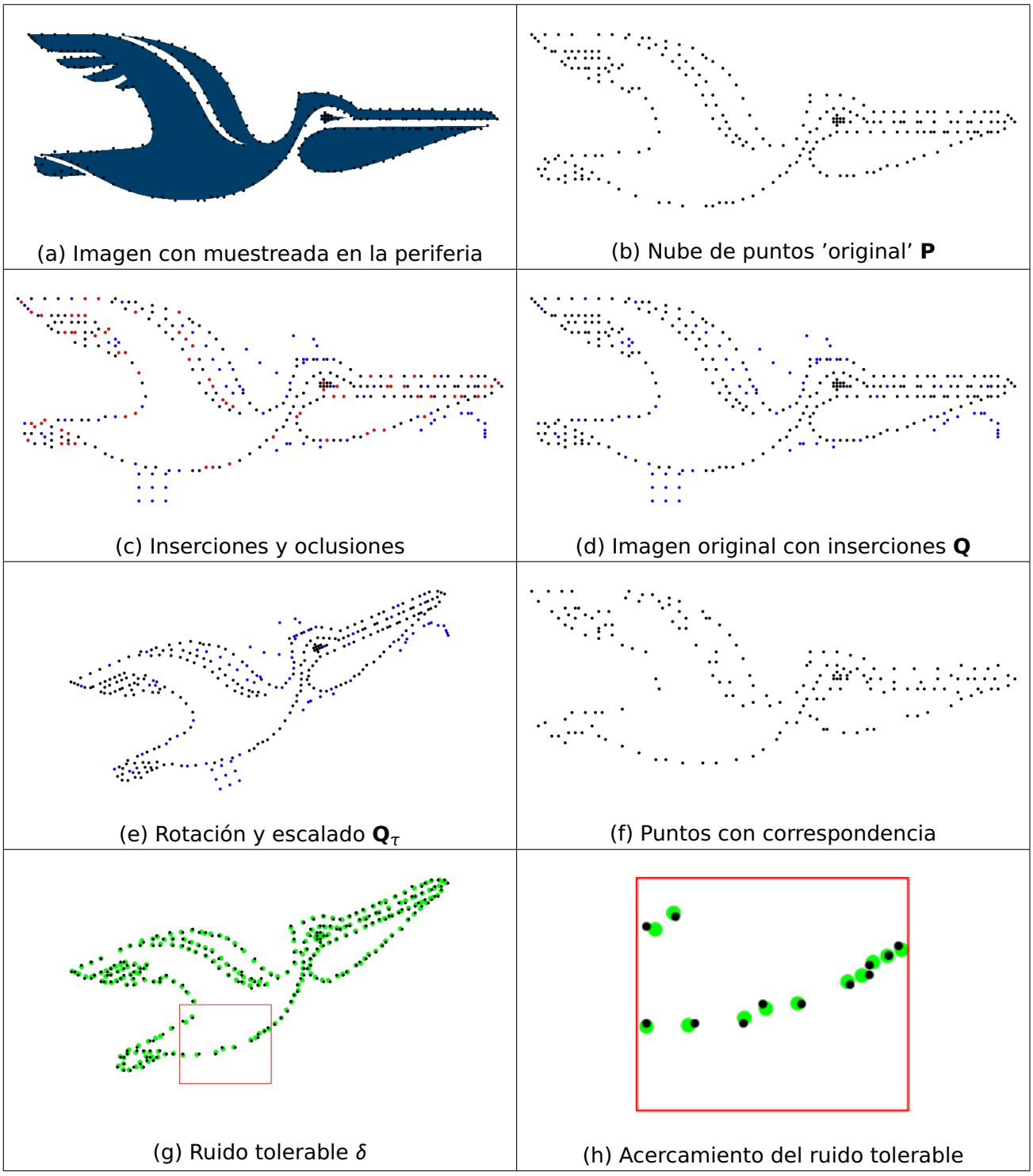
Es importante subrayar, que las nubes de puntos en dos dimensiones las podemos generalizar a dimensiones superiores y de igual modo es posible encontrar emparejamiento en dichas nubes de puntos.

Existen muchos problemas que se pueden abstraer como el problema de identificación de nubes de puntos, es decir, encontrar los parámetros de una función que transforma un conjunto de puntos en otro. Estos puntos serán vectores de una cierta dimensión.

A continuación, discutiremos con detalle algunos de los problemas más representativos relacionados con la identificación de nubes de puntos.

1.1.1. Identificación de huellas dactilares

De acuerdo con Feng y Jain (2011) "la mayoría de los sistemas de reconocimiento de huellas dactilares están basados en cuatro tipos de representaciones (Figura 2), escala de grises, imagen de fase, imagen tipo esqueleto y minucias". Es posible identificar una nube de puntos en el plano, cuando se emparejada con otra nube de puntos, aún ante ciertas transformaciones y ruido. Esto quiere decir, que el problema de identificación de huellas dactilares se puede abstraer y convertir en el mismo problema del emparejamiento de nubes de puntos.



(a) Imagen con muestreada en la periferia

(b) Nube de puntos 'original' **P**

(c) Inserciones y oclusiones

(d) Imagen original con inserciones **Q**

(e) Rotación y escalado $\mathbf{Q}_\tau$

(f) Puntos con correspondencia

(g) Ruido tolerable δ

(h) Acercamiento del ruido tolerable

Figura 1. Emparejamiento de nubes de puntos en dos dimensiones.

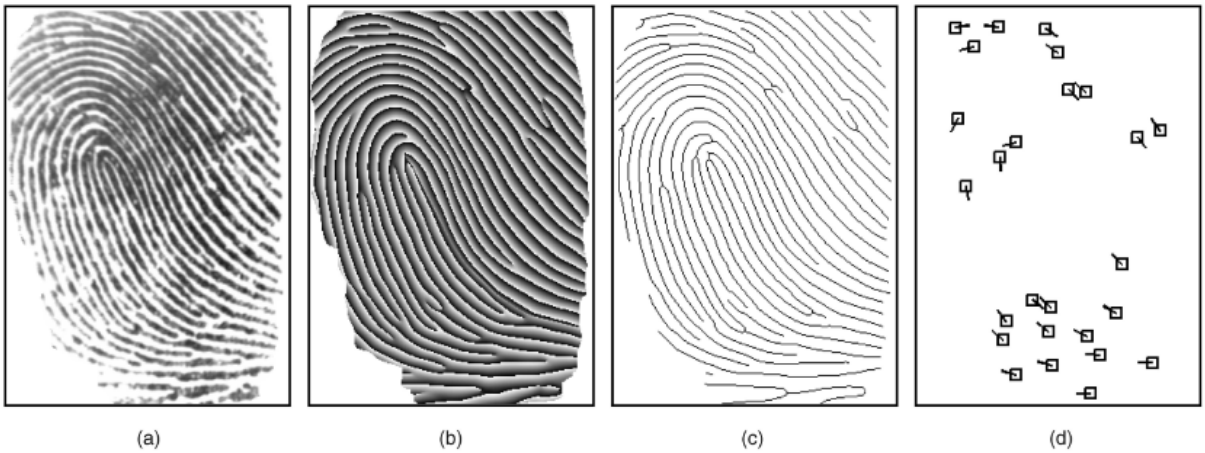


Figura 2. Representación de huellas dactilares: a) escala de grises, b) imagen de fase, c) imagen tipo esqueleto, d) minucias (Feng2011).

1.1.2. Reconocimiento por iris y retina

El sistema de reconocimiento o identificación por registro de iris o retina es similar al de huellas dactilares pues a partir de las características físicas de una persona se obtiene un vector de cierta dimensión (representado como nube de puntos), que se asocia a una persona. De acuerdo con las primeras técnicas descritas por Wildes *et al.* (1994) Rathgeb *et al.* (2012) para el reconocimiento de iris y por Akram *et al.* (2011) para el reconocimiento de retina, se requiere comparar esta información con una consulta y se debe encontrar la correspondencia entre los puntos de interés.

1.1.3. Reconocimiento facial

Por otro lado, el reconocimiento facial se logra gracias a una representación del rostro como nube de puntos en el plano, mediante la detección de ciertos puntos de interés, algunos trabajos utilizan el algoritmo SIFT (scale-invariant feature transform) y sus mejoras como (Geng y Jiang, 2009) para generar dichos ’keypoints’, correspondientes, por ejemplo, las esquinas de los ojos y labio o ciertas peculiaridades en las cejas y barbilla. Esto en conjunto dibuja una nube de puntos en el plano que está asociada al rostro de una persona. Posteriormente si la información se almacena y se le emparejar con otra nube de puntos de consulta, se puede determinar si son la misma nube de puntos o no, incluso ante un cierto nivel de ruido.

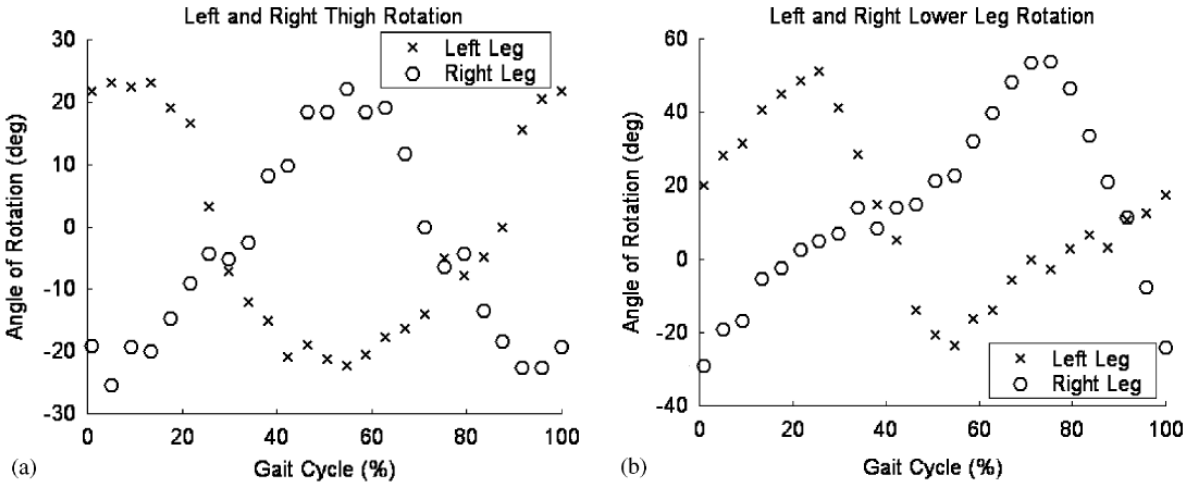


Figura 3. Rotación del muslo y la parte inferior de las piernas izquierda y derecha con un cambio de fase de medio período.

1.1.4. Identificación de caminado y corrido

Otro caso realmente interesante de la identificación por datos biométricos es el reconocimiento por el modo de correr y/o caminar. Este modelo básicamente toma en cuenta el ángulo de rotación entre algún par de huesos en las piernas a través del tiempo. En la Figura 3 obtenida de (Yam *et al.*, 2004) se explica el modelo que permite utilizar los datos (ángulo de rotación) del movimiento de las piernas derecha e izquierda, dibujando claramente una función, a priori desconocida, pero que se puede determinar con diferentes métodos. De tal modo que cada persona está asociada a una *manera de caminar* y este es descrito por ciertos parámetros muy específicos que describen el propio modelo.

Adicionalmente (Yam *et al.*, 2004) menciona que: ”las características biométricas tienen ventajas y desventajas únicas cada una, por ejemplo: los patrones de huellas dactilares, iris y retina pueden ser únicos en poblaciones grandes, pero presentan el problema de ser difíciles de recolectar, pues requieren de la cooperación del sujeto, por otro lado, la marcha puede tener el potencial de superar estas limitaciones, porque se puede obtener información desde la distancia, además , el hecho de que el cuerpo es de mayores proporciones que la cara, orejas y dedos permite la posibilidad de obtener información biométrica del caminado sin necesidad de cámaras de alta resolución”.

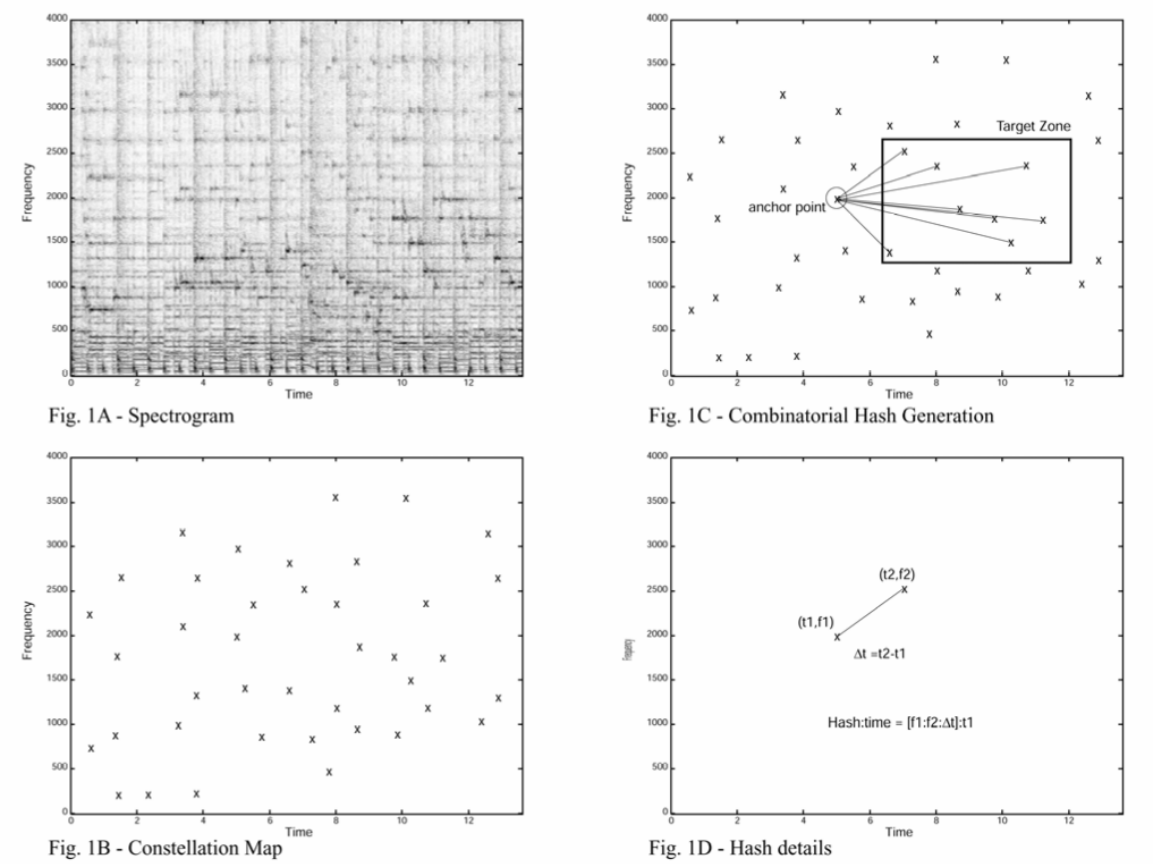


Figura 4. Espectrograma de una señal de audio y su representación como nube de puntos. Imagen obtenida de Wang (2003).

1.1.5. Identificación de audio

Más allá del reconocimiento de datos biométricos, también es posible sacar provecho de la identificación de nubes de puntos para representar una señal de audio. De acuerdo con Wang (2003), ”las pistas de audio pueden ser representadas como si se tratara de huellas digitales, además resultan invariantes ante algunas transformaciones/filtros y muestran robustez ante cierto nivel de ruido, sin embargo, resultan sensibles a la entropía de la pista”. Como se puede apreciar en la Figura 4, el espectrograma de una señal de audio en un plano relaciona el tiempo de la señal con sus frecuencias, allí se pueden identificar claramente una serie de puntos destacados, formando una nube de puntos, si a este conjunto de puntos se le puede encontrar la correspondencia de cada punto con otra nube de puntos consulta, es posible identificar estas pistas aún cuando presenten cierto nivel de ruido.

1.1.6. Realidad aumentada

En el área de visión computacional, incluyendo realidad aumentada, reconstrucción 3D basado en imágenes y auto-localización se encuentra una característica en común; estimar con precisión la relación que existe entre dos imágenes de una misma superficie plana (esto es llamado homografía), con esta información también se puede estimar el movimiento de una cámara (es decir el desplazamiento de una cámara entre un punto *a* y un punto *b*), esto último se puede determinar encontrando tan solo cuatro puntos correspondientes en el mismo plano; por otro lado, es bien sabido que la estimación del movimiento de la cámara es realmente sensible a cualquier tipo de ruidos y valores atípicos (Lee y Kim, 2007).

Hablando de realidad aumentada, la detección de planos es un prerrequisito básico para todas las demás tareas en el área, de acuerdo con Yang y Förstner (2010). Para realizar esta tarea en tiempo real, se requiere identificar un conjunto de puntos muestra (nube de puntos) y establecer cuantos y cuales puntos encajan con el modelo de un plano (en este caso), y determinar los parámetros de dicho plano que vive en un espacio 3D. Además de la detección de un plano, muchas otras tareas fundamentales en realidad aumentada recaen en encontrar la homografía entre dos imágenes, i.e., dadas dos imágenes se debe encontrar la función que transforma la perspectiva de una en la siguiente, por ejemplo, para acoplar alguna animación en la pantalla de un celular; con el movimiento del dispositivo, es necesario saber si se aleja o acerca (por ejemplo) para hacer las correcciones correspondientes, pero lo mismo vale para cualquier movimiento de cámara.

Hasta aquí, la identificación de nubes de puntos es el problema central de esta tesis, sin embargo, aún no se menciona como se puede solucionar este problema, es decir, falta saber ¿qué se necesita para lograr identificar la correspondencia entre dos conjuntos de puntos dados? (si es que existen) y una vez encontrada dicha correspondencia, obtener los parámetros que describen la transformación de un conjunto de puntos en otro. RANSAC[9] es el algoritmo más usado para resolver este problema, por lo que dedicaremos el Capítulo 2 para abordar todos los detalles al respecto.

[9] Random sample consensus

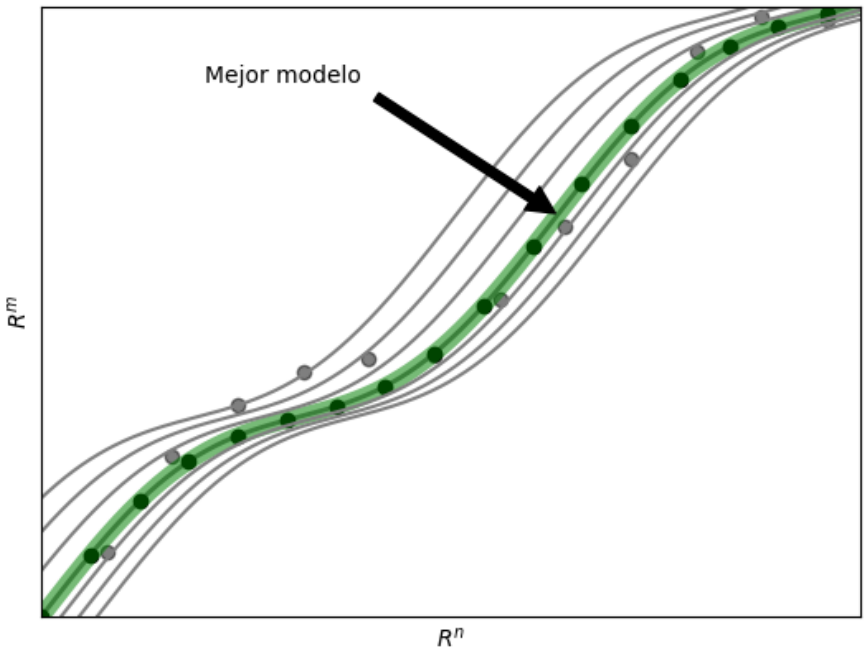


Figura 5. Representación 2D de un conjunto de vectores de $n + m$ dimensiones.

Hasta este momento se han expuesto: la importancia de la identificación de nubes de puntos, los principales ejemplos y las aplicaciones más recientes, por lo tanto, es muy importante resaltar que se puede generalizar la solución de toda esta variedad de problemas. En la Figura 5 se tiene representado un conjunto de vectores (cada punto) de $n + m$ dimensiones. Es posible encontrar una función que se ajuste a la mayoría de los demás vectores, siempre y cuando se elijan los parámetros adecuados que definen el modelo, sin embargo, existe toda una familia de posibles funciones que podrían ajustarse a una fracción de los datos, aunque no todas las funciones generan un consenso con la mayor parte de los vectores; encontrar los parámetros que definen el modelo con mayor consenso se puede lograr con al algoritmo RANSAC.

Esta tesis tiene como objetivo:
proponer una mejora sustancial al algoritmo RANSAC, a partir de un índice de densidad obtenido de la propia distribución espacial de los datos.

1.2. Sobre esta tesis

1.2.1. Justificación

Contribuir a la mejora en los tiempos de ejecución del estado del arte de RANSAC, utilizando un índice relacionado directamente con la densidad de vecinos de cada punto.

1.2.2. Objetivo general

Diseñar e implementar una heurística capaz de mejorar los tiempos de ejecución de RANSAC, guiando la sección de muestreo del algoritmo a partir de un sesgo de densidad.

1.2.3. Objetivos específicos

Los objetivos específicos de este trabajo que están *enumerados* a continuación se validarán con cada uno de los siguientes modelos:

- Ajuste de una línea recta a partir de una nube de puntos en el plano, conformada por inliers y outliers.
- Ajuste de una circunferencia a partir de una nube de puntos en el plano, conformada por inliers y outliers.
- Emparejamiento de dos nubes de puntos en el plano, bajo transformaciones afines, inserciones y oclusiones.

    1. Desarrollar un método que genere un etiquetado parcial en las nubes de puntos y determinar si esto implica una mejora en los tiempos de ejecución del algoritmo.
    2. Analizar la robustez del método, tal que soporte un porcentaje de eliminaciones y oclusiones de puntos, además determinar dicho valor.
    3. Analizar la tolerancia del método frente a un cierto nivel de ruido.
    4. Analizar el rendimiento del método y hacer una comparación con los trabajos previos encontrados en la literatura.

1.3. Organización

- *Capitulo 1:* Breve introducción de los conceptos principales alrededor del problema de *emparejamiento de nubes de puntos* a través de las aplicaciones más destacadas de las últimas décadas.
- *Capitulo 2:* Descripción del algoritmo RANSAC y el estado del arte alrededor de dicho algoritmo, además se señala gráficamente que lugar ocuparía este trabajo dentro de la familia RANSAC.
- *Capitulo 3:* Desarrollo del algoritmo propuesto, con tres variantes, denominado 'DD-RANSAC'.
- *Capitulo 4:* Metodología de los experimentos que permiten medir el comportamiento de DD-RANSAC.
- *Capitulo 5:* Análisis y presentación de resultados.
- *Capitulo 6:* Conclusiones y trabajo futuro.

Capítulo 2. Antecedentes

Estadística, proviene del término italiano *statista*, que significa 'hombre de Estado'. De acuerdo con la enciclopedia británica, estadística es la ciencia de recopilar, analizar, presentar e interpretar datos. Actualmente, las grandes cantidades de datos disponibles en el mundo han facilitado el proceso de convertir dichos datos en información[1] útil, por lo tanto, los últimos avances teóricos y prácticos de esta área, no han sido menores; una manera de obtener información relevante o parámetros que describan a un cierto conjunto de datos es a través de estimadores[2].

Este capítulo tratará una familia de algoritmos de *consenso de muestra aleatoria* o también llamada RANSAC[3] por sus siglas en inglés. De acuerdo con Hartley y Zisserman (2003), RANSAC se ha convertido en uno de los más populares y robustos estimadores, desarrollado por la comunidad de visión computacional.

En muchos problemas prácticos de estimación de parámetros que contienen errores graves de datos, estos no son compensados. Para hacer frente a esta situación, se han formulado varias heurísticas que generalmente usan todos los datos para obtener los parámetros de un modelo, después se ubica el dato más alejado de dicho modelo y se asume que es un error grave y se elimina, este proceso se puede repetir hasta que la desviación máxima del conjunto de datos restantes sea menor que un cierto umbral preestablecido o hasta que ya no hay datos suficientes para continuar. Los puntos más distantes del modelo se les llama 'puntos envenenados' y en conjunto con los datos correctos del modelo, con frecuencia suelen hacer que la heurística anterior falle (ver Figura 6), en otras palabras, el promedio no es una técnica apropiada para aplicar a un conjunto de datos no verificados (Fischler y Bolles, 1981).

Es importante subrayar la diferencia entre los errores en el modelo o también llamados errores de clasificación y los errores de medición. De acuerdo con Fischler y Bolles (1981), estos últimos errores siguen una distribución normal, por lo tanto, se les puede aplicar el supuesto del suavizado sin mayor problema. Por otro lado, los errores de clasificación son errores graves, tienen un efecto significativamente mayor que los errores de medición y no se promedian. A diferencia de los estimadores desarrollados

[1] Información es el nombre que damos a un conjunto organizado de datos procesados.
[2] Algunos ejemplos de estimaciones son: la media, varianza, proporción y máxima verosimilitud.
[3] Random sample consensus

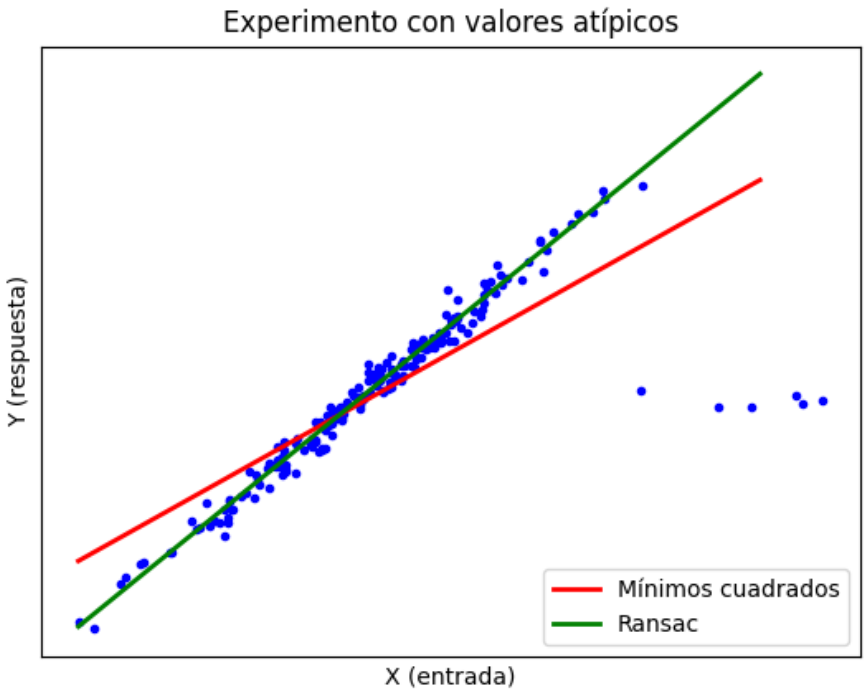


Figura 6. Ajuste de los parámetros de un modelo matemático a partir de los valores correspondientes a un experimento con 3 % de valores atípicos, obtenidos con dos métodos diferentes, mínimos cuadrados y RANSAC.

desde la estadística y de las heurísticas que tratan con el problema de los errores de clasificación/modelo, RANSAC no necesita algoritmos complejos de optimización, ni enormes cantidades de memoria. Por el contrario, RANSAC tiene una implementación simple y robusta. El algoritmo es iterativo y consiste en dos pasos o secciones principales: generación de hipótesis y evaluación de hipótesis (Choi *et al.*, 1997).

RANSAC vio la luz con el artículo titulado: ’Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography’, de Fischler y Bolles (1981). De acuerdo con este artículo, RANSAC fue creado con la capacidad de suavizar conjuntos de datos con un gran porcentaje de error, por lo tanto, resulta un algoritmo ideal para aplicaciones en análisis automatizado de imágenes, donde los sensores que proporcionan los datos son muy propensos al error. En dicho trabajo se resolvió con éxito el problema de determinación de la ubicación o LDP[4] por sus siglas en inglés. Y por si esto fuera poco, esto ha motivado la investigación en esta área de las Ciencias de la Computación, donde muchos investigadores han hecho

[4]Location Determination Problem

importantes aportes en los últimos cuarenta años.

2.1. Vanilla-RANSAC

En tecnología e informática se le da el adjetivo de ’vanilla’ (pronunciado: va-ni-la) a la versión más simple, básica o sin características especiales de un producto, método o técnica. El término se basa en el hecho de que ’vanilla’ es el sabor de helado más popular, básico o al menos el que se sirve con más frecuencia, de acuerdo con Eric Raymond, editor de: ’The New Hacker’s Dictionary’.

Como ya se mencionó, el algoritmo base de RANSAC consta de dos pasos o secciones principales, generación de hipótesis y evaluación de hipótesis. La generación de hipótesis es un conjunto de parámetros que definen al modelo. Para esto, a partir de un conjunto de puntos dado D, se requiere seleccionar aleatoriamente, un ’subconjunto mínimo muestra’ de *puntos* $\overline{S_i}$ de tamaño m que sean capaces de satisfacer la cantidad de *parámetros* necesarios para generar una hipótesis H_i. Para el caso del modelo del ajuste de una línea recta, el mínimo número de parámetros es dos y se pueden satisfacer eligiendo dos puntos, pero si se conociera la pendiente de la recta, *per se*, con tan solo un punto se podría generar una hipótesis. Por otro lado, el modelo del círculo requiere de tres parámetros, estos se pueden satisfacer conociendo el centro del círculo y un radio, si este no fuera el caso, también se pueden satisfacer estos tres parámetros con tres puntos que formen parte de la circunferencia. Es importante subrayar que elegir tres puntos proporciona un total de seis parámetros (dos por cada punto), pero con estos pueden ser determinados los tres parámetros necesarios del modelo del círculo, del mismo modo, en el modelo de la línea recta vamos a seleccionar pares de puntos que proporcionan cuatro parámetros en total, pero con esta información se pueden calcular los dos parámetros necesarios para dicho modelo. En la Figura 7 está ilustrada la selección de puntos $\overline{S_i}$ para el modelo del círculo y de la línea recta.

El primer paso de RANSAC es escoger aleatoriamente un subconjunto mínimo de puntos $\overline{S_i}$, suficientes para satisfacer los parámetros del modelo en cuestión y proponer un modelo o hipótesis H_i. Las hipótesis generadas pueden ser de dos tipos de datos diferentes, los contaminados y los *no contaminados*. Este último es aquel dato que pertenece al modelo, también se le conoce en la literatura como ’**inlier**’. Por otro lado,

los datos contaminados son los que anteriormente llamamos ’errores de clasificación o de modelo’, este tipo de dato también es conocido como ’**outlier**’ (Chum, 2005; Fischler y Bolles, 1981).

El segundo paso de RANSAC es la evaluación de hipótesis, esta utiliza los parámetros de la hipótesis generada anteriormente para calcular su función de costo L o también llamado ’soporte’. El soporte se calcula con la cantidad de elementos del conjunto D que se ajustan dentro de la hipótesis, con un cierto margen de error δ (Figura 7). Esta hipótesis junto con el soporte obtenido se guardan como el ’mejor modelo’ y se repiten los dos pasos anteriores (generación de hipótesis y evaluación de hipótesis) hasta que se encuentre otro ’mejor modelo’ con mayor soporte que pueda sustituir al anterior (Chum, 2005).

En la Figura 7 se ilustran los pasos básicos de RANSAC, generación de hipótesis, evaluación de hipótesis y como resultado; el mejor modelo. Del lado izquierdo se observa el modelo de la línea recta con un bajo porcentaje de outliers. Del lado derecho de la figura se encuentra el modelo del círculo con un alto porcentaje de outliers. Las muestras de datos utilizada para la generación de una hipótesis (puntos en color rojo) S_l^0 para la línea recta y S_c^0 para el círculo son de tamaño $m_l = 2$ y $m_c = 3$ respectivamente. Las hipótesis correspondientes a las muestras S_l^0 y S_c^0 son modelos representados como una las franjas verdes de la ’Evaluación de Hipótesis’. Estas franjas tienen una anchura característica, determinada por el margen de error δ. Cada punto que se encuentra dentro de las líneas color verde se pintó de color rojo para representar que está dentro del modelo propuesto. Finalmente, la cantidad de puntos dentro de cada modelo propuesto es igual al tamaño del soporte correspondiente L_l^0 y L_c^0.

Nótese que las muestras S_l^0 y S_c^0 *no son muestras libres de outliers*, por lo tanto, los soportes correspondientes L_l^0 y L_c^0 se pueden apreciar muy bajos en relación con la cantidad de puntos de cada ejemplo/modelo. Por otro lado, los mejores modelos se obtuvieron de las muestras que efectivamente son libres de outliers, S_l^k y S_c^k respectivamente. Además estás muestras libres de outliers generan modelos con un alto soporte, L_l^k y L_c^k respectivamente. Esto último se aprecia claramente con la cantidad de puntos iluminados en rojo que están dentro de cada figura; línea recta y círculo, ambos en color verde (Figura 7).

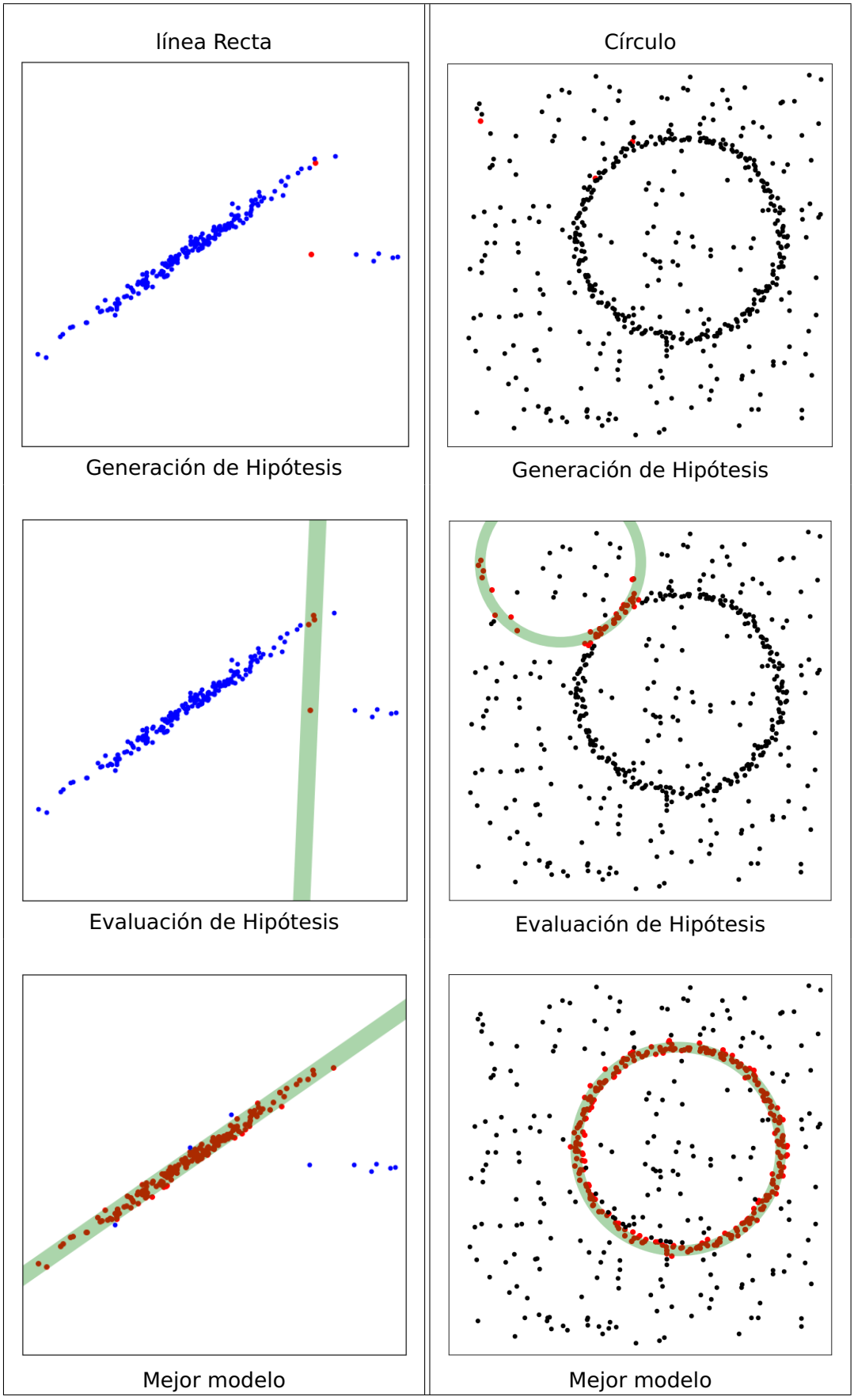


Figura 7. Pasos clave de Vanilla-RANSAC para el ajuste de una *Línea Recta* y una *Circunferencia*.

El conjunto de puntos seleccionado aleatoriamente, de tamaño m, con que se obtienen los parámetros que describen el modelo propuesto, puede estar conformado únicamente por inliers, únicamente por outliers o una combinación de los dos. De acuerdo con Fischler y Bolles (1981) y con Chum (2005), una hipótesis generada a partir de una muestra libre de outliers, produce un modelo acertado y con un soporte alto. Por el contrario, una muestra que contiene uno o más outliers, es decir una muestra contaminada, va a producir un modelo con bajo soporte.

Suponiendo que existe un único modelo de mayor soporte en la configuración de nubes de puntos, el ciclo de generación-evaluación de hipótesis de RANSAC, es decir una única muestra solución, en el peor de los casos el ciclo de generación-evaluación se podría repetir hasta evaluar todas las combinaciones de puntos posibles, aún cuando el número de combinaciones es enorme siempre se puede encontrar dicho mejor modelo. De acuerdo con Fischler y Bolles (1981) se dice que el tiempo de respuesta de RANSAC crece como N^m siendo N en número de datos o puntos del conjunto D y m es el número mínimo de parámetros del modelo. Por esta razón se necesita calcular cual sería el número de iteraciones a evaluar para encontrar una hipótesis generada a partir de una muestra formada únicamente de inliers.

De acuerdo con Fischler y Bolles (1981); Chum (2005); Hartley y Zisserman (2003), siendo $P(I)$ la probabilidad de que una muestra no contaminada de tamaño m sea seleccionada aleatoriamente del conjunto D de cardinalidad $|D| = N$. Dicha probabilidad está determinada por la relación entre las combinaciones de puntos libres de outliers y las combinaciones de puntos posibles, es decir:

$$P(I) = \frac{\binom{I}{m}}{\binom{N}{m}} \tag{3}$$

Sea $P(I) = u$ y su complemento $v = 1 - u$, es decir v es la probabilidad de encontrar al menos un ’outlier’ al escoger aleatoriamente una muestra. El número de intentos k necesarios para dar con una muestra libre de outliers con una probabilidad p (normalmente del 0.99) es:

$$1 - p = (1 - u^m)^k \tag{4}$$

Despejando k se obtiene:

$$k = \frac{\log(1-p)}{\log(1-(1-v)^m)} \tag{5}$$

Por lo tanto, el tiempo de ejecución del algoritmo RANSAC queda descrito de la siguiente manera:

$$T = k\,(T_G + N\,T_E) \tag{6}$$

donde T_G es el tiempo de la generación de hipótesis, T_E el tiempo de evaluación de la hipótesis de cada uno de los N puntos y k es el número de muestras tomadas para encontrar al menos una muestra libre de ouliers con probabilidad p.

A continuación, la estructura de algoritmo RANSAC de Fischler y Bolles (1981).

Algorithm 1: Vanilla-RANSAC

input : $D = \{o_1, o_2, \ldots, o_n\}$: Nube de puntos o base de datos
H: Hipótesis generada, basada en un modelo matemático
m: Tamaño mínimo de muestra
p: Probabilidad de obtener una muestra libre de outliers
δ: Umbral de error

output: H^*: Mejor hipótesis evaluada
L^*: Mejor soporte

$k \leftarrow f(m, p)$: (ecuación 5)
$H^* \leftarrow \emptyset$
$L^* \leftarrow \emptyset$

for $i \leftarrow 1$ **to** k **do**
G_H $\quad \overline{S_i} \leftarrow$ *Aleatoriamente*, selecciona m puntos de D como *i-esima* muestra
$\quad H_i \leftarrow$ Genera una hipótesis del modelo H, a partir de $\overline{S_i}$
E_H $\quad L_i \leftarrow$ Calcular el soporte de la hipótesis H_i con tolerancia δ
$\quad$ **if** $L_i > L^*$ **then**
$\quad\quad L^* \leftarrow L_i$
$\quad\quad H^* \leftarrow H_i$

Vanilla-RANSAC se puede mejorar en términos del tiempo de ejecución, robustez y/o precisión. Dichas mejoras mantienen la misma estructura de este algoritmo, pero se modifica la generación de hipótesis G_H y/o la evaluación de hipótesis E_H.

2.2. Familia de algoritmos RANSAC

De acuerdo con Choi *et al.* (1997) la familia de algoritmos RANSAC puede ser clasificada por una de las siguientes característica: precisión, robustez o rapidez, además, en la Figura 8 (cortesía del mismo autor). La familia RANSAC está agrupada de acuerdo a las mejoras realizadas al algoritmo, en comparación con Vanilla-RANSAC, estas mejoras son las siguientes:

1. ***Mejoras en Robustez***

   - **Finalización adaptada:** El número de iteraciones k influye directamente en el tiempo de ejecución y en la robustez del algoritmo. Este número se *adapta* de acuerdo a la calidad de las muestras obtenidas. Esto último se puede determinar con el valor γ (relación inliers/outliers), σ (tamaño del ruido) y/o Σ (error medio) obtenidos del muestreo.
     - Feng&Hung MAPSAC (Feng y Hung, 2003)
     - AMLESAC (Konouchine *et al.*, 2005)
     - uMLESAC (Choi y Kim, 2008)
   - **Evaluación adaptada:** La evaluación de una hipótesis H_i necesita de ciertos parámetros dados desde un principio, como la relación inliers/outliers y el tamaño del ruido. Cuando no se tienen estos parámetros *per se*, generar estimaciones de dichos parámetros conforme las muestras $\overline{S_i}$ son evaluadas, mejora la robustez del algoritmo.
     - FengHung MAPSAC (Feng y Hung, 2003)
     - AMLESAC (Konouchine *et al.*, 2005)
     - pbM-estimator (Subbarao y Meer, 2006)
     - uMLESAC (Choi y Kim, 2008)

2. ***Mejoras en Precisión***

   - **Optimización local:** Se agrega un paso extra a RANSAC, una vez encontrada una hipótesis H_i con un alto soporte. El objetivo de los esquemas de optimización es mejorar la estimación de los parámetros del modelo.
     - LO-RANSAC (Chum *et al.*, 2004)

- pbM-estimator (Subbarao y Meer, 2006)

- **Selección de Modelo:** La selección entre evaluar una muestra $\overline{S_i}$ de acuerdo con un modelo u otro se puede incorporar y automatizar con RANSAC. Esto resulta muy útil, sobre todo cuando se trata con datos degenerados. El 'switcheo' del modelo se produce en la parte final de la evaluación de hipótesis.
  - MAPSAC (Torr, 2002)
  - QDEGSAC (Frahm y Pollefeys, 2006)
- **Función de perdida:** Modifica el cálculo del soporte. En vez de utilizar el número de puntos acordes con la hipótesis propuesta, la función de costo se obtiene con otra medida. Por ejemplo, la estimación de máxima verosimilitud o la estimación de probabilidad máxima *a posteriori*.
  - MSAC (Torr y Murray, 1997)
  - MLESAC (Torr y Zisserman, 2000)
  - MAPSAC (Torr, 2002)

3. ***Mejoras en rapidez***

- **Evaluación parcial:** Evalúa parcialmente cada hipótesis propuesta, es decir, calcula el soporte L_i de cada modelo propuesto H_i utilizando solo una fracción de los datos de conjunto D inicial, en vez de utilizar todo el conjunto. Esto reduce el tiempo de ejecución.
  - RANSAC with Bail-out Test (Capel, 2005)
  - R-RANSAC with SPRT (Matas y Chum, 2005)
  - R-RANSAC with T(d,d) Test (Chum y Matas, 2002)
  - Preemptive RANSAC (Nistér, 2003)
- **Muestreo guiado:** Obtiene los subconjuntos muestra $\overline{S_i}$, a partir de un cierto orden o sesgo, en vez de hacer una selección aleatoria de puntos.
  - PROSAC (Chum y Matas, 2005)
  - GASAC (Rodehorst y Hellwich, 2006)
  - Guided MLESAC (Tordoff y Murray, 2005)
  - NAPSAC (Myatt *et al.*, 2002)

El algoritmo propuesto a partir de este trabajo de investigación es una heurística que guía el proceso de generación de hipótesis a través de un sesgo dependiente de la densidad de puntos.

- *DD-RANSAC (Density-Driven RANSAC)*

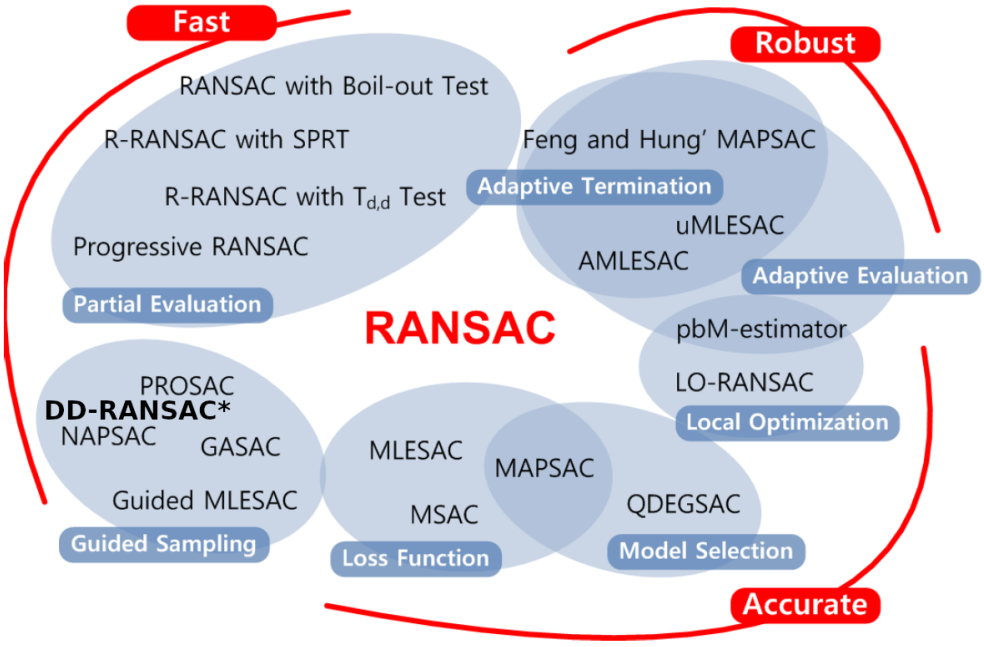


Figura 8. Familia RANSAC. Mapa de Choi *et al.* (1997) con DD-RANSAC añadido.

A continuación, se abordarán algunas de las variantes de RANSAC relevantes para esta tesis.

2.2.1. LO-RANSAC

El algoritmo 'RANSAC con optimización local' (LO-RANSAC) fue propuesto por Chum *et al.* (2004). Está mejora al original Vanilla-RANSAC es sencilla, pero con muy buenos resultados. LO-RANSAC propone agregar un paso de optimización del modelo generalizado después de la evaluación de hipótesis, es decir, después de obtener un buena hipótesis H_i muy bien evaluada o con alto soporte, se procede con la optimización de los parámetros de la misma. De acuerdo con el autor, el algoritmo resulta menos sensible ante el ruido y el mal condicionamiento, además esta estrategia garantiza que el paso de optimización tenga un impacto mínimo en el tiempo de ejecución.

2.2.2. MLESAC

La propuesta de Torr y Zisserman (2000) titulada: ”MLESAC[5]: A new robust estimator with application to estimating image geometry”, adopta la misma estrategia de muestreo para generar hipótesis H_i, pero evalúa el soporte de dicha hipótesis con la estimación de máxima verosimilitud o MLE[6], en lugar de evaluar con el tamaño del soporte con el número de inliers acordes con una cierta hipótesis.

2.2.3. MAPSAC

En estadística bayesiana, la estimación de ’probabilidad máxima *a posteriori*’, MAP[7] por sus siglas en inglés, es la estimación de una cantidad desconocida, que es igual a la moda de la distribución *a posteriori*. Esta se puede utilizar para obtener una estimación puntual de una cantidad no observada a partir de datos empíricos, es decir, la distribución *a priori*. MAP está estrechamente relacionado con la MLE, pero emplea un objetivo de optimización aumentado, por lo tanto, se puede decir que MLE es un caso especial de MAP, cuando la distribución de probabilidad *a priori* sigue una distribución uniforme.

Torr (2002) propuso MAPSAC, utilizando el enfoque bayesiano del MAP. De acuerdo con el mismo autor, a veces es posible ajustar varios modelos o hipótesis diferentes a un mismo conjunto de datos, por lo tanto, se debe tomar una decisión sobre cuál es el más apropiado. Su trabajo citado explora la manera de automatizar el proceso de selección de modelos con este mismo enfoque bayesiano. Si los datos de una muestra pudieran haber surgido de uno de varios modelos, el procedimiento de ajuste clasifica todos los modelos potenciales. MAPSAC puede seleccionar el modelo que mejor se ajusta a los datos, a esto se le llama ’el problema de la selección de modelos robustos’.

2.2.4. QDEGSAC

De acuerdo con Frahm y Pollefeys (2006), la degeneración de los datos significa que estos no proporcionan suficientes restricciones para determinar una solución única, por lo tanto, solo se puede determinar una familia de soluciones. En el caso de los

[5] Maximum Likelihood Estimation SAmple Consensus
[6] Maximum Likelihood Estimation
[7] Maximum *a posteriori* probability

datos *quasi-degenerados*, se tiene un conjunto de datos degenerados, más una pequeña fracción de datos que proporcionan las restricciones restantes para determinar una solución única.

El algoritmo Vanilla-RANSAC tiene una baja probabilidad de calcular la solución correcta ante el problema de datos quasi-degenerados, además, dicho algoritmo no verifica que sea única la solución resultante. Por otro lado, el algoritmo 'RANSAC para datos quasi-degenerados' (QDEGSAC por sus siglas en inglés), propone un enfoque basado en jerarquías sobre el número de restricciones proporcionadas por los datos, por lo tanto, es aplicable sobre diversos modelos y presenta la misma robustez que Vanilla-RANSAC (Frahm y Pollefeys, 2006).

2.2.5. AMLESAC

'AMLESAC: A New Maximum Likelihood Robust Estimator' lo publicó Konouchine *et al.* (2005), en este artículo se propone una variante adaptativa al ruido de MLESAC que adopta la misma estrategia de muestreo y también busca una solución que maximiza la probabilidad de los parámetros de un modelo adecuado. AMLESAC, por otro lado, estima simultáneamente la proporción de valores atípicos γ y el nivel de ruido interno σ, además propone una optimización local y una selección de subconjuntos para mejorar la precisión y la aumentar su velocidad.

2.2.6. R-RANSAC con T(d,d) Test

El algoritmo *RANSAC aleatorizado*, mejor conocido como R-RANSAC[8], fue desarrollado por Chum y Matas (2002). Esta versión de RANSAC propone mejorar el proceso de evaluación de hipótesis, imponiendo un 'test' o prueba a la hipótesis generada H_i. Cuando se trata con nubes de puntos muy grandes, el tiempo utilizado para evaluar la hipótesis generada con cada uno de los puntos de D puede ser enorme, así que generar una prueba que permita decidir entre hacer evaluación de hipótesis completa o no, *optimiza* el tiempo de ejecución del algoritmo. Primero se selecciona un tamaño de muestra óptimo d que sea representativo del conjunto D, el 'test' se aprueba únicamente si todos los puntos de esta muestra representativa, aleatoriamente seleccionados, son acordes a la hipótesis generada. Sí el 'test' es aprobado, entonces

[8] Randomized RANSAC

se procede con la evaluación de hipótesis completa y se obtiene el soporte de dicha hipótesis.

2.2.7. Preemptive-RANSAC

RANSAC preferente fue propuesto por Nistér (2003); esta modificación de RANSAC explota la idea de aleatorizar el paso de evaluación de la hipótesis. La velocidad general de RANSAC es un producto del número de hipótesis generadas y el tiempo invertido en procesar cada hipótesis (Ecuación 6). El número de hipótesis generadas depende del tamaño de la muestra y el tiempo requerido para procesar cada hipótesis consiste en el tiempo necesario para generar cada hipótesis H_i, más el tiempo de evaluación de la función de costo L_i sobre todos los puntos del conjunto D. De este modo, la evaluación de la función de costo depende linealmente en $|D|$. Para nubes de puntos muy grandes, el algoritmo puede pasar la mayor parte del tiempo evaluando funciones de costo o soportes de hipótesis contaminadas, es decir, hipótesis que se generaron a partir de muestras $\overline{S_i}$ que contienen al menos un outlier. Por lo tanto, rechazar hipótesis contaminadas lo antes posible mejora el tiempo de ejecución del algoritmo. RANSAC preferente se basa en comparar hipótesis después de evaluar cada una con solo una fracción de los puntos y rechazar las que tienen peores puntuaciones. Primero se elige un punto de los datos para cada hipótesis activa y la función de costo se actualiza utilizando dicho punto, las mejores hipótesis se mantienen y se descartan las de menor soporte. Este procedimiento permite maximizar la función de costo.

2.2.8. PROSAC

El algoritmo de consenso por muestreo progresivo o PROSAC[9] por sus siglas en inglés, fue propuesto por Chum y Matas (2005). De acuerdo con el mismo autor, esta propuesta algorítmica explota un orden lineal definido sobre el conjunto de correspondencias mediante una función de similitud. A diferencia de Vanilla-RANSAC, que trata todas las correspondencias por igual y extrae muestras aleatorias del conjunto completo, el muestreo de PROSAC se obtiene de conjuntos progresivamente más grandes de correspondencias mejor clasificadas. Bajo la suposición de que la medida de similitud predice una coincidencia mejor que la conjetura aleatoria, PROSAC muestra que

[9] Progressive Sample Consensus

se logran grandes ahorros computacionales. Derivado del conjunto de correspondencias, el número de muestras extraídas en PROSAC converge hacia Vanilla-RANSAC en el peor de los casos. En el artículo de Chum y Matas (2005), la similitud se definió de dos maneras; con la distancia, en el espacio de SIFT[10], con el mejor y segundo mejor 'match' y con la distancia euclidiana de los primeros quince coeficientes de la Transformada de Coseno Discreta o DCT[11].

2.2.9. *NAPSAC*

El algoritmo de "consenso de muestra por N puntos adyacentes" o NAPSAC[12] por sus siglas en inglés fue propuesto por Myatt *et al.* (2002). Esta versión de RANSAC está centrada en la eficiencia de la estrategia de muestreo. La hipótesis principal es que los inliers tienden a estar más cercanos entre sí que de los outliers. En este algoritmo, la estrategia de muestreo se modificó para aprovechar la proximidad entre inliers. De acuerdo con Chum (2005), NAPSAC podría tener un desempeño eficiente y robusto en altas dimensiones, donde la probabilidad de extraer una muestra *no contaminada* se vuelve muy baja, incluso para conjuntos de datos con pocos outliers.

[10] Scale-invariant feature transform
[11] Discrete Cosine Transform
[12] N Adjacent Points Sample Consensus

Específicamente, NAPSAC procede de la siguiente manera:

1. Selecciona un punto x_0 de la nube de puntos D .
2. Encuentra un conjunto de puntos C_{x_0} dentro de una bola de radio r y centro x_0. Estos conjuntos también pueden ser vistos como pequeños 'clusters'.
3. Si la cardinalidad de C_{x_0} es menor a cierta medida ρ, dicho conjunto se descarta.
4. A partir de todos los elementos de C_{x_0}, selecciona conjuntos $\overline{S_i}$ de tamaño m para generar hipótesis H_i del modelo.
5. Evalúa H_i maximizando la función de costo de acuerdo con Vanilla-RANSAC de Fischler y Bolles (1981) .

De acuerdo con Myatt *et al.* (2002), los conjuntos $\overline{S_i}$ obtenidos de una C_{x_0} con cardinalidad mayor a ρ tienen más probabilidades de ser un conjunto *no contaminado*. Sin embargo, no se determina cuál es el valor óptimo de r.

Es muy importante para esta tesis resaltar la variante NAPSAC de Myatt *et al.* (2002), ya que nuestra hipótesis inicial es la misma: *"los inliers tienden a estar más cercanos entre sí que de los outliers"*, sin embargo, el enfoque y la solución misma son muy diferentes. En el siguiente capítulo se explicará cada detalle de nuestra propuesta algorítmica.

A continuación, se presentan los artículos más recientes que se relacionan directamente con el uso del algoritmo RANSAC, guiado por la densidad de puntos en un determinado espacio.

2.3. *Histology to μCT Data Matching Using Landmarks and a Density Biased RANSAC*

De acuerdo con Chicherova *et al.* (2014), la necesidad de una histología 2D razonable para el registro de datos 3D se ha vuelto cada vez más importante, gracias a la disponibilidad de dispositivos de microtomografía computarizada (μCT) asequibles, de alta resolución espacial y con contraste de tejidos. La combinación de la información funcional de la histología con los datos de imágenes estructurales del μCT proporciona

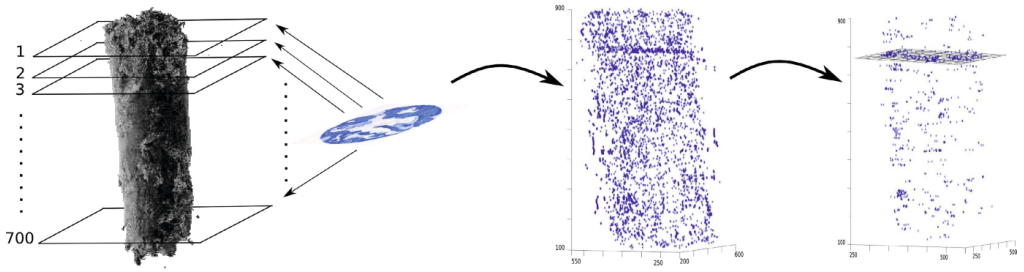


Figura 9. Secuencia: (*izquierda*) Coincidencia de características entre los datos de μCT y una imagen histológica, (*centro*) Nube de puntos 3D de los puntos coincidentes, (*derecha*) Ajuste del plano, optimizado con RANSAC. Imagen obtenida de Chicherova *et al.* (2014).

una mejor comprensión para identificar las características anatómicas de los tejidos duros y blandos.

Chicherova *et al.* (2014) propone hacer coincidir imágenes histológicas *2D* con datos de microtomografía computarizada *3D*, basándose en puntos de referencia (en este caso se utilizó SURF[13]) y un ajuste plano con RANSAC, pero este último con un sesgo de densidad de puntos, permitiendo una localización el 75 % de las imágenes histológicas de la base de datos con los datos 3D (μCT) en menos de cuatro minutos.

El método se divide en 4 secciones, adquisición de datos, detección y coincidencia de funciones, generación de nube de puntos 3D y RANSAC sesgado por su densidad para un ajuste del plano robusto. Esta última sección es de mayor interés para nosotros porque se utiliza RANSAC para ajustar el modelo del plano entre puntos 2D con una nube de puntos 3D. En la Figura 9 se ilustran estas secciones. De acuerdo con Chicherova *et al.* (2014), RANSAC se acelera guiando la generación de hipótesis H_i del algoritmo a través del sesgo de densidad local. Sin embargo, el método desarrollado en dicho artículo necesita el parámetro de la *densidad límite* como un dato de entrada y no ofrece detalles sobre el cálculo de la densidad de cada punto, solo se explica que los puntos con mayor densidad tienen más probabilidades de ser elegidos como parte de una muestra $\overline{S_i}$ de RANSAC.

[13] Speeded-Up Robust Features

2.4. *A Density-Based Recursive RANSAC Algorithm for Unmanned Aerial Vehicle Multi-Target Tracking in Dense Clutter*

El seguimiento de objetivos es un tema actual y de relevancia para la vigilancia de vehículos aéreos no tripulados. Recientemente, se ha utilizado ’Recursive RANSAC’ (R-RANSAC) para esta aplicación y ha mostrado un buen rendimiento en el seguimiento en un entorno denso y desordenado. Sin embargo, la pesada carga computacional limita su uso en vehículos aéreos no tripulados. Yang *et al.* (2017) propone una modificación a R-RANSAC basada en la densidad de puntos, utilizando esta propiedad como un sesgo en el muestreo. De acuerdo con el autor, a esta modificación se le llamó ’Density-Based Recursive RANSAC’ (DBR-RANSAC).

En Yang *et al.* (2017) se explica que la densidad está estrechamente relacionada con subconjuntos de puntos llamados ’clusters’. Para generar estos ’clusters’, primero se eligen un par de puntos (p_1, p_2) no visitado y se obtiene la distancia euclidiana entre ambos puntos $d(p_1, p_2)$. Si la distancia calculada es menor que cierto umbral Δ *predeterminado*, los puntos forman parte del mismo ’cluster’, en caso contrario cada punto genera un ’cluster’ nuevo. Finalmente, la densidad de cada punto está dada por la cardinalidad del ’cluster’ al cual pertenece. Esta medida genera un sesgo de selección en el muestreo del algoritmo R-RANSAC.

De acuerdo con Yang *et al.* (2017), es congruente que los puntos con mayor densidad sean seleccionados con preferencia porque la medición de los objetivos tienden a reunirse en una región vecina a la anterior. Esto se ve reflejado en la continuidad del movimiento de los objetivos. El aprovechamiento de esta propiedad se puede observar con claridad en la Figura 10 .

De acuerdo con Yang *et al.* (2017), el sesgo de densidad en RANSAC puede evitar la aleatoriedad del muestreo y esto puede reducir la complejidad del cálculo. Por último, los resultados de la simulación muestran la validez del algoritmo propuesto y específicamente del uso de la densidad de puntos como una guía útil para el algoritmo RANSAC (Figura 10), al menos en este caso, para el problema del *seguimiento de objetivos*.

Hasta ahora se han mencionado las variantes de la familia RANSAC más relevantes

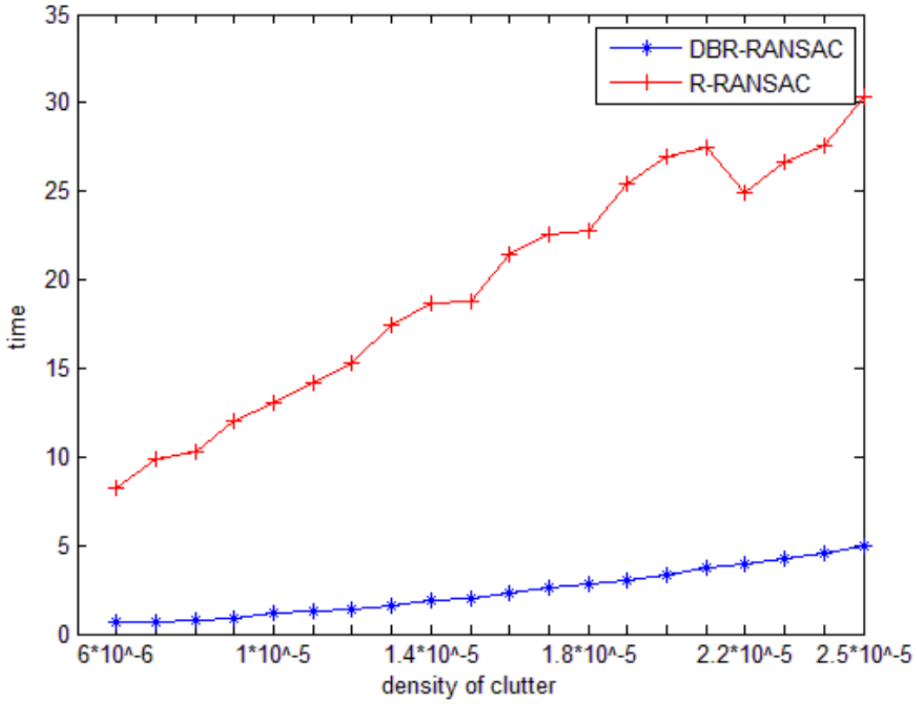


Figura 10. Desempeño de DBR-RANSAC y R-RANSAC, rastreando múltiples objetivos con diferentes valores de *densidad de desorden*. Imagen obtenida de Yang *et al.* (2017).

para esta investigación con el objetivo de enmarcar y contextualizar nuestra propuesta algorítmica, misma que será presentada en el próximo capítulo.

Capítulo 3. DD-RANSAC

Nuestra propuesta algorítmica tiene como base, modificar Vanilla-RANSAC y utilizar la densidad de puntos como guía para la selección de muestras $\overline{S_i}$. Esta propuesta cobra mucho sentido después de plantear la siguiente hipótesis:
"los inliers tienden a estar más cercanos entre sí que de los outliers".
Pensamos que al utilizar la densidad de puntos se pueden encontrar con mayor probabilidad las muestras $\overline{S_i}$ libres de outliers y en consecuencia, esto ayudaría a reducir el tiempo de ejecución del algoritmo.

Descubrir las regiones de mayor *densidad* de puntos se puede lograr de diferentes formas. Yang *et al.* (2017) implementó la localización de regiones densas gracias a una distancia definida o radio alrededor de cada punto, generando 'clusters'. En este trabajo utilizaremos la consulta de los *k vecinos más cercanos reversos* para identificar aquellos puntos que tengan más vecinos a su alrededor, en otras palabras, con esta consulta se puede saber si un punto se encuentra en una región densa o no, si es el caso, también es posible ordenar por densidad los elementos del conjunto y darles preferencia al momento de ser seleccionados.

Para explicar con claridad la propuesta algorítmica y la forma de encontrar las regiones de mayor densidad en una nube de puntos es necesario abordar y enfatizar las diferencias entre una consulta por rango, una consulta de los k vecinos más cercanos y la consulta de los k vecinos más cercanos reversos.

3.1. Consulta por rango

De acuerdo con Zezula *et al.* (2006), la consulta por rango $R(q, r)$ es una de las consultas más comunes. Esta consulta se hace mediante un objeto de referencia q con un respectivo radio r como restricción de distancia y una colección de objetos o nube de puntos D en donde se realiza la búsqueda. Dicha consulta recupera todos los objetos $o_i \in D$ cercanos a q que se encuentran a menos de r *distancia*[1] de q.

Formalmente definida de la siguiente manera:

[1] En este caso, se entiende que la función de distancia es la *distancia Euclidiana*, sin embargo, existen otras funciones de distancia en el *Apéndice*; aunque estas funciones no son relevantes para los objetivos de esta tesis, también es importante mencionarlas.

$$R(q,r) = \left\{ o \in D : d(o,q) \leq r \right\} \tag{7}$$

De ser necesario, la lista de objetos individuales arrojados como respuesta se pueden ordenar de acuerdo a su distancia respecto a q. La imagen 11 ilustra con claridad la consulta por rango, definida con la ecuación 7.

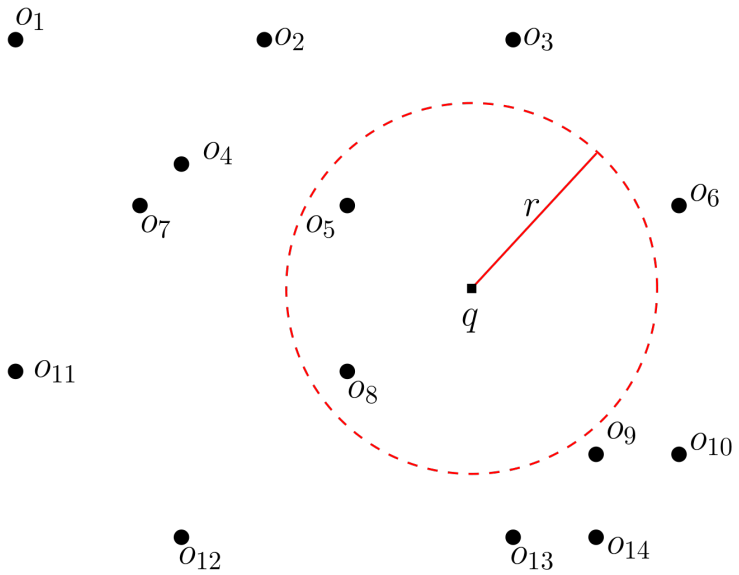


Figura 11. Consulta por rango, $R(q,r) = \{o_5, o_8\}$.

Nótese que el objeto de consulta q no necesita estar dentro de la colección D para ser referenciado en $R(q,r)$. Por otro lado, cuando el radio de búsqueda es cero, la consulta de rango es denominada 'coincidencia exacta', es decir: $R(q,0)$. En este caso, se estaría buscando una copia idéntica del objeto de consulta q. El uso más habitual de este tipo de consultas es para algoritmos de eliminación, es decir, cuando se desea localizar un objeto para quitarlo de la colección.

3.2. Consulta de *k vecinos más cercanos* (NN)

Para buscar objetos similares por rango, en una consulta dada, es necesario especificar una distancia máxima r. Pero puede ser difícil especificar el radio sin algún conocimiento previo de los datos. Cuando se especifica un radio de consulta demasiado pequeño, el resultado puede ser un conjunto vacío, esto requerirá de una o más nuevas búsquedas con radio mayor para obtener algún resultado diferente. Por

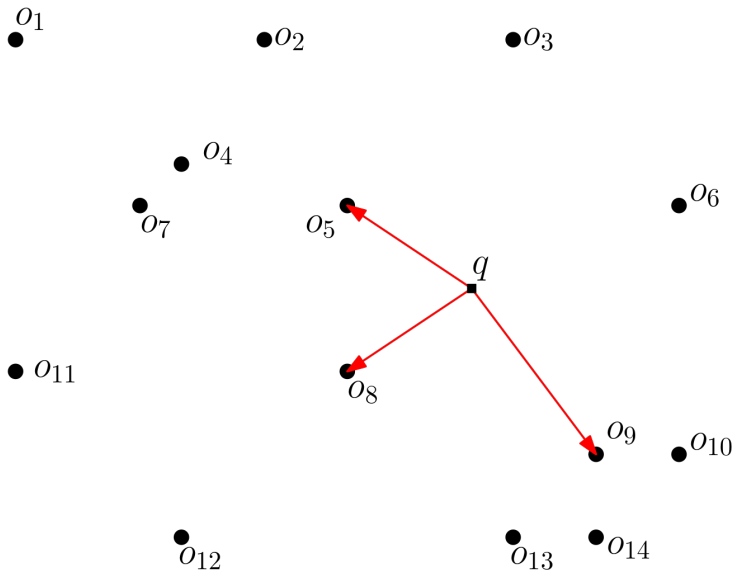


Figura 12. Consulta de los k vecinos más cercanos $NN_k(q)$, donde $k = 3$. $NN_3(q) = \{o_5, o_8, o_9\}$.

otro lado, si los radios de consulta son demasiado grandes, las consultas pueden ser computacionalmente muy costosas y los conjuntos de respuestas podrían contener elementos no significativos o incluso podría ser la base de datos completa (Zezula *et al.*, 2006).

De acuerdo con Chávez *et al.* (2001) la consulta de los k vecinos más cercanos $NN_k(q)$ recupera los k elementos más cercanos de q que pertenecen a la base de datos D, definida de la siguiente manera:

$$NN_k(q) = \left\{A \subseteq D \mid |A| = k \,\&\, \forall u \in A, v \in D - A,\ d(q,u) \leq d(q,v)\right\} \tag{8}$$

Los k elementos que se encuentran a la menor distancia de la consulta q pueden ser ordenados de acuerdo a su distancia respecto de q. La figura 12 nos da ejemplo de esto.

A diferencia de la consulta por rango, $NN_k(q)$ no necesita ningún parámetro de distancia como entrada, sin embargo, esta consulta posee una distancia mínima implícita en el *k-esimo* vecino más cercano. Por otro lado, es importante notar que el objeto de consulta q no necesita estar dentro de la colección D para ser referenciado (figura 12). Además, si la colección D donde se realiza la búsqueda contiene menos de k objetos, la consulta devolverá la base de datos completa como respuesta (Chávez *et al.*, 2001).

Una manera un tanto *ingenua* de implementar la consulta $NN_k(q)$ es utilizando un método de búsqueda secuencial que calcule la distancia desde la consulta q hasta cada uno de los elementos del conjunto de datos y finalmente devolviendo un conjunto de k elementos con menor distancia. Este enfoque funciona, incluso encuentra los vecinos más cercanos exactos, sin embargo, esta solución *no escala* ya que al duplicar el tamaño de la base de datos el tiempo de búsqueda también se duplica. Para resolver esto, se han propuesto algunas estructuras de datos como *KD-Tree*, *Ball-Tree* y *LSH* que permiten realizar búsquedas en *tiempo logarítmico*. Esto quiere decir, que al *duplicar* el tamaño de la base de datos D, el tiempo de búsqueda aumenta en una unidad de tiempo en promedio. Además, este tipo de estructuras de datos permiten por sí mismas implementar la consulta $NN_k(q)$ (Mahapatra y Chakraborty, 2015).

De acuerdo con Chávez *et al.* (2001), uno de los mayores obstáculos para las búsquedas eficientes en espacios métricos es la existencia y ubicuidad de aplicaciones reales para espacios de alta dimensión. Las técnicas tradicionales de indexación para espacios vectoriales, como KD-Tree, tienen una dependencia exponencial de la dimensión de representación del espacio, ya que el espacio que contiene las respuestas a las consultas crece exponencialmente con el número de dimensiones; a esto se le conoce como *la maldición de la dimensionalidad*. En algunos casos, donde la dimensión intrínseca es muy alta, el problema se vuelve intratable para los algoritmos exactos y se tiene que recurrir a algoritmos aproximados o probabilísticos.

En esta tesis requerimos de la consulta $NN_k(q)$ para calcular los vecinos más cercanos *reversos* de cada elemento de D, por lo tanto, calcular $NN_k(q)$ resulta *indispensable*. La implementación de la consulta $NN_k(q)$ se desarrolló con la estructura de datos *KD-Tree*; esta fue la más adecuada ya que trabajamos con conjuntos de datos de baja dimensionalidad, es decir, vectores *2D*.

En el apéndice se encuentran los detalles sobre la maldición de la dimensionalidad, sobre las implicaciones de utilizar *KD-Tree* para consultas $NN_k(q)$ en espacios de baja dimensionalidad y sobre las demás estructuras de datos mencionadas como *Ball-Tree* y *LSH*. Las dos últimas no tienen relevancia técnica para esta tesis, sin embargo, es importante abordarlas para dar mayor contexto a esta investigación.

3.3. Consulta de los vecinos más cercanos *reversos* (RNN)

De acuerdo con Du y Xie (2012), la consulta del *vecino más cercano reverso* es el problema más importante en una base de datos *espacial*, $RNN_k(q)$ es un tipo de consulta basada en $NN_k(q)$. Esta función se puede utilizar para evaluar la influencia de algún objeto de la consulta q en una colección de datos D. En muchas situaciones, es interesante saber cómo se percibe o clasifica un objeto específico en términos de su distancia con otros objetos del conjunto de datos, es decir:
¿qué objetos ven a q como su vecino más cercano?.

De acuerdo con Zezula *et al.* (2006), la consulta de los k vecinos más cercanos reversos, $RNN_k(q)$ se puede definir de la siguiente manera:

$$RNN_k(q) = \left\{ A \subseteq D \mid \forall u \in A : q \in NN_k(u) \wedge \forall v \in D - A : q \notin NN_k(v) \right\} \tag{9}$$

Nótese que la cantidad de elementos del conjunto resultante de $RNN_k(q)$ puede ser diferente de k, ya que este parámetro solo se utiliza dentro de la función $NN_k(q)$ para determinar la cantidad de elementos que devuelve. Por otro lado, la consulta $RNN_k(q)$ no es simétrica. Esto último quiere decir que dados dos objetos o_α y $o_\beta \in D$; el resultado de $RNN_k(o_\alpha)$ puede contener a o_β, pero no necesariamente la consulta $RNN_k(o_\beta)$ contiene a o_α. La Figura 13 y Tabla 1 dan ejemplo de estas observaciones.

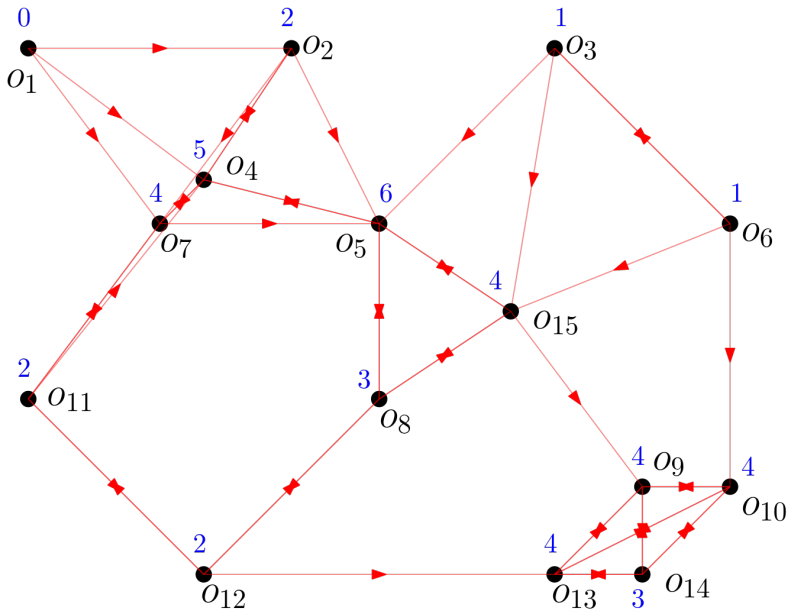


Figura 13. Consulta de vecinos más cercanos *reversos*, con $k = 3$.
El número de RNN_k de cada punto está marcado en color azul. De cada punto salen tres flechas rojas, indicando la consulta $NN_{k=3}$ de cada uno.

Tabla 1. Detalles de la consulta RNN_k de la Figura 13).

D	$NN_k(o_i)$	$\lvert NN_k(o_i)\rvert$	$RNN_k(o_i)$	$\lvert RNN_k(o_i)\rvert$
o_1	$\{o_4, o_7, o_2\}$	3	$\{\ \}$	0
o_2	$\{o_4, o_5, o_7\}$	3	$\{o_1, o_4\}$	2
o_3	$\{o_5, o_6, o_{15}\}$	3	$\{o_6\}$	1
o_4	$\{o_7, o_2, o_5\}$	3	$\{o_1, o_2, o_5, o_7, o_{11}\}$	5
o_5	$\{o_{15}, o_8, o_4\}$	3	$\{o_2, o_3, o_4, o_7, o_8, o_{15}\}$	6
o_6	$\{o_3, o_{15}, o_{10}\}$	3	$\{o_3\}$	1
o_7	$\{o_4, o_5, o_{11}\}$	3	$\{o_1, o_2, o_4, o_{11}\}$	4
o_8	$\{o_{15}, o_5, o_{12}\}$	3	$\{o_5, o_{12}, o_{15}\}$	3
o_9	$\{o_{10}, o_{14}, o_{13}\}$	3	$\{o_{10}, o_{13}, o_{14}, o_{15}\}$	4
o_{10}	$\{o_9, o_{14}, o_{13}\}$	3	$\{o_6, o_9, o_{13}, o_{14}\}$	4
o_{11}	$\{o_7, o_{12}, o_4\}$	3	$\{o_7, o_{12}\}$	2
o_{12}	$\{o_8, o_{11}, o_{13}\}$	3	$\{o_8, o_{11}\}$	2
o_{13}	$\{o_{14}, o_9, o_{10}\}$	3	$\{o_9, o_{10}, o_{12}, o_{14}\}$	4
o_{14}	$\{o_9, o_{13}, o_{10}\}$	3	$\{o_9, o_{10}, o_{13}\}$	3
o_{15}	$\{o_5, o_8, o_9\}$	3	$\{o_3, o_5, o_6, o_8\}$	4

En la Figura 13 se observan un conjunto de datos representados en el plano como una base de datos $D = \{o_1, 0_2, 0_3, \ldots, 0_{15}\}$ y las consultas de los $k = 3$ vecinos más cercanos de cada uno de los elementos de D, es decir: $NN_k(o_i)|o_i \in D$. El resultado de cada una de estas consultas está representado con las $k = 3$ flechas rojas que salen de cada punto. Estas flechas están enlistadas en la segunda columna de la Tabla 1, para observar el orden con el que son devueltas. Entonces, la consulta RNN_k consiste en identificar en cuales de las consultas $NN_k(o_i)$ aparece cada o_i como respuesta. Por ejemplo, en la Figura 13, o_5 aparece seis veces como resultado de las consultas $NN_k(o_i)$ (segunda columna de la Tabla 1), por lo tanto los vecinos más cercanos reversos de o_5 son el conjunto de los elementos o_i que tienen a o_5 como parte de sus $k = 3$ vecinos más cercanos, es decir, el conjunto $RNN_k(o_5) = \{o_2, o_3, o_4, o_7, o_8, o_{15}\}$. Esto se puede apreciar en la Figura 13 ya que los vecinos reversos de o_5 son las flechas rojas que apuntan hacia este mismo punto.
Finalmente, esta cantidad de flechas que apuntan hacia determinado elemento o_i es su número de vecinos más cercanos reversos y se puede observar con color *azul* a un lado de cada elemento de la Figura 13.

La información más relevante de la Figura 13, ejemplo de una consulta RNN_k, se encuentra en la cuarta y quinta columna de la Tabla 1, sin embargo, es importe resaltar algunos aspectos de esta consulta; para introducir el *índice de densidad* propuesto en esta tesis.

1. La cuarta columna de la tabla 1 contiene la consulta de los vecinos más cercanos *reversos* de cada elemento de la base de datos, es decir,
   $RNN_k(o_i)|\forall o_i \in D$.

2. Existen puntos con más vecinos más cercanos *reversos* que otros, e incluso el punto o_1 no contiene a *ninguno*. Esto último ejemplifica con claridad que tanto la consulta $NN_k(q)$ como la consulta $RNN_k(q)$ *no* son simétricas. Por ejemplo, la consulta $NN_k(o_1) = \{o_4, o_7, o_2\}$, pero ninguna de las consultas $NN_k(o_4)$, $NN_k(o_7)$ o $NN_k(o_2)$ contienen a o_1 como parte de sus vecinos más cercanos (observar Tabla 1).

3. A pesar de que el número de vecinos más cercanos reversos puede variar entre los diferentes elementos, el número *total* de vecinos reversos es $n \times k$, es decir, el número total de vecinos más cercano de todo el conjunto de puntos D es igual al número total de vecinos más cercanos *reversos*. Esta *redistribución* se observa con claridad al comparar las columnas dos y cuatro de la Tabla 1.

4. Finalmente, se observa que los puntos con menor número RNN_k como o_1, o_3 y o_6 se encuentran ubicados en la periferia del conjunto, por el contrario, los puntos más céntricos como o_4, o_5 y o_{15} tienen los números de RNN_k más altos.

Como ya se ha mencionado, la hipótesis de trabajo de esta tesis es que los ’inliers’ tienden a estar más cercanos entre sí. Por el contrario, los ’outliers’ son más dispersos. En consecuencia, los puntos con mayor número de vecinos más cercanos reversos tienen una mayor probabilidad de generar muestras libres de ’outliers’. Lo anterior se traduce en una mejora en los tiempos de ejecución del algoritmo.

Dicho lo anterior, nuestro primer objetivo es *ordenar* todos los puntos por densidad, por esta razón se utilizó el número de vecinos más cercanos reversos de cada punto como ’*proxy*’ de densidad. Para ejemplificarlo tenemos la Figura 13 donde está anotado en color azul el número de RNN_k de cada punto. Esto permite generar un conjunto conformado por subconjuntos, cuyos elementos sean de igual densidad, dichos subconjuntos son ordenados de mayor a menor densidad.
Un ejemplo de lo anterior es la ecuación 12.

3.4. Propuestas algorítmicas

El objetivo de este trabajo es reducir el tiempo de ejecución del algoritmo RANSAC, modificando el muestreo. Bajo la hipótesis de que los puntos con mayor densidad tienen mayor probabilidad de generar muestras libres de 'outliers', primero se *genera un índice* que logre identificar los puntos con mayor densidad utilizando la consulta RNN_k en cada punto, después se *ordenan* los elementos del conjunto a modo de lista, de acuerdo a la densidad obtenida (ecuación 12) y finalmente, se seleccionan muestras de dicha lista, al recuperar en orden los elementos de la lista se prioriza los elementos con la mayor densidad. Este ordenamiento se puede expresar de la siguiente forma:

$$E = \Pi_\rho(D') \tag{10}$$

donde D' es equivalente al conjunto D, pero con el índice $RNN_k(o_i)$ de cada o_i ligado a cada elemento del mismo conjunto D y Π_ρ es la *permutación* que ordena el conjunto D', de acuerdo con su densidad. Este conjunto *ordenado* E también se puede expresar de la siguiente manera:

$$D \xrightarrow{orden} E \tag{11}$$

La notación anterior (ecuación 11) será utilizada en los algoritmos propuestos de las siguientes subsecciones, para facilitar la representación de un conjunto ordenado de acuerdo al índice calculado con el número de k vecinos más cercanos reversos, $RNN_k(o_i)$.

Tabla 2. Densidad de cada elemento de la Figura 13.

D	$\lvert RNN_k(o_i)\rvert$
o_1	0
o_2	2
o_3	1
o_4	5
o_5	5
o_6	1
o_7	4
o_8	2
o_9	3
o_{10}	4
o_{11}	2
o_{12}	2
o_{13}	4
o_{14}	3
o_{15}	4

$\xrightarrow{orden}$

Densidad	E
6	o_5
5	o_4
4	o_{15}
4	o_7
4	o_9
4	o_{10}
4	o_{13}
3	o_8
3	o_{14}
2	o_2
2	o_{11}
2	o_{12}
1	o_3
1	o_6
0	o_1

Para ejemplificar el ordenamiento de la ecuación 11 vamos a retomar la Figura 13, donde se tiene un conjunto de puntos presentados en el plano. Para ordenar todos los puntos o_i, se pueden agrupar en subconjuntos de acuerdo a su número de vecinos más cercanos reversos de la siguiente manera:

$$E = \big\{\{o_5\}, \{o_4\}, \{o_{15}, o_7, o_9, o_{10}, o_{13}\}, \{o_8, o_{14}\}, \{o_2, o_{11}, o_{12}\}, \{o_3, o_6\}, \{o_1\}\big\} \quad (12)$$

Nótese que la ecuación anterior tiene el mismo orden que resultante de la Tabla 2. Sin embargo, es importante destacar que los subconjuntos obtenidos están agrupados por densidad, es decir, que todos los elementos que pertenecen a un mismo subconjunto tiene el mismo número de vecinos más cercanos reversos. Se dice que estos subconjuntos son ’subconjuntos *no* ordenados’, por lo tanto, cualquier *permutación* en los elementos de un mismo subconjunto es equivalente con cualquier otra. Además, el conjunto completo E también es equivalente a otro, siempre y cuando se respete el orden de los subconjuntos. Para dar ejemplo de lo anterior, presentamos el siguiente conjunto E que es equivalente al conjunto anterior de la ecuación 12 aun después de haber hecho permutaciones dentro de cada subconjunto:

$$E = \big\{\{o_5\}, \{o_4\}, \{o_{10}, o_{15}, o_{13}, o_9, o_7\}, \{o_{14}, o_8\}, \{o_{11}, o_{12}, o_2\}, \{o_6, o_3\}, \{o_1\}\big\} \quad (13)$$

Una vez definido el conjunto *ordenado* E, se debe plantear alguna forma de seleccionar sus elementos; puede ser en estricto orden a modo de lista, seleccionando elementos con cierta probabilidad de acuerdo al subconjunto que pertenecen o incluso generando nuevos subconjuntos y reordenándolos en función de la *suma* total de densidad de todos sus elementos. Estas tres posibilidades se propusieron para esta investigación, cada una como un método para generar muestras $\overline{S_i}$ de un cierto tamaño m correspondiente al *tamaño mínimo de muestra* que es capaz de satisfacer el número de parámetros de un cierto modelo[2]. Por lo tanto, se obtuvieron tres propuestas algorítmicas que serán presentadas a continuación.

3.4.1. DD-RANSAC (Uniforme)

Hasta aquí, el algoritmo base de RANSAC consta de dos pasos o secciones principales, la generación de hipótesis (G_H) y evaluación de hipótesis (E_H). En esta propuesta algorítmica vamos a añadir un paso previo a (G_H) llamado preprocesado de datos (P_D) (observe el algoritmo 2).

El preprocesado de datos es el núcleo de este algoritmo, por lo tanto, nos vamos a centrar en esta sección. Las otras partes del algoritmo permanecen igual al original Vanilla-RANSAC a diferencia de un pequeño detalle que también será abordado y explicado con claridad.

A partir de un conjunto de puntos dado D, se genera un segundo conjunto D' que contiene el número de vecinos más cercanos reversos de cada uno de los elementos de D, es decir, D' se obtiene realizando la consulta: $RNN_k(o_i)$.

El conjunto D' debe ser ordenado de acuerdo con la ecuación 11 ($E \overset{orden}{\longleftarrow} D'$), como ya se mencionó, este orden jerarquiza los puntos de acuerdo a su densidad. Si se recorren los puntos a modo de lista, se observaría que el primer elemento recuperado tendría el mayor número de RNN_k y el último elemento tendría el menor número.

Es importante recordar que la fase de muestreo en RANSAC necesita de una serie de subconjuntos $\overline{S_i}$ de tamaño m (el *tamaño mínimo de muestra* m es la cantidad de

[2] En el siguiente capítulo abordaremos los modelos utilizados en esta tesis, cada uno con un tamaño mínimo de muestra distinto; $m = 2$, $m = 3$ y $m = 6$).

Tabla 3. Combinaciones de tres en tres del conjunto $L = \{a, b, c, d, e, f\}$.

G	i	j	k
g_1	a	b	c
g_2	a	b	d
g_3	a	b	e
g_4	a	b	f
g_5	a	c	d
g_6	a	c	e
g_7	a	c	f
g_8	a	d	e
g_9	a	d	f
g_{10}	a	e	f
g_{11}	b	c	d
g_{12}	b	c	e
g_{13}	b	c	f
g_{14}	b	d	e
g_{15}	b	d	f
g_{16}	b	e	f
g_{17}	c	d	e
g_{18}	c	d	f
g_{19}	c	e	f
g_{20}	d	e	f

puntos necesarios que en conjunto son capaces de satisfacer los parámetros de un determinado modelo). Por lo tanto, el siguiente paso es totalmente dependiente del *modelo* que se quiere recuperar de la nube de puntos; por ser dependiente de m.

Ahora vamos a presentar el conjunto G, llamado generador, este conjunto tienen como elementos a $\overline{g_i}$ que posteriormente serán requeridos para el muestreo. Estos subconjuntos son resultado de las *combinaciones* de los elementos de E en grupos de tamaño m. Las combinaciones de objetos son cualquier selección de ellos en la que *no* importa el orden. El número de combinaciones de n objetos tomados de m en m se calculan de la manera siguiente:

$$ {}_nC_m = \frac{n!}{m!\,(n-m)!} \tag{14} $$

Para aclarar esta idea, vamos a tomar como ejemplo las combinaciones posibles de tres en tres del conjunto $L = \{a, b, c, d, e, f\}$, representadas en la Tabla 3.

Nótese que el número muestras obtenidas en la Tabla 3 es ${}_6C_3 = 20$.

También se puede observar que cada S_i de la tabla es un subconjunto $\{i, j, k\} \mid i < j < k$ donde i, j, k hacen referencia al índice de los elementos ordenados de L. Además, el índice k no necesariamente tiene que llegar hasta el último elemento del conjunto, es decir, k podría ser truncado hasta un cierto número para evitar muestrear los últimos elementos del conjunto; ya que estos son de menor interés de acuerdo con nuestra hipótesis de trabajo.

Retomado el primer algoritmo propuesto, el conjunto generador G podría llegar a tener ${}_nC_m$ elementos, donde $n = |E|$ y m es el tamaño mínimo de muestra. Cada elemento de G es una muestra diferente, de tal manera que en esta etapa de RANSAC las muestras ya están preseleccionadas y ordenadas, sin embargo, el conjunto G se va generando conforme se consulta cada uno de sus elementos; ya que calcular el conjunto completo antes de consultar el primer elemento, podría ser realmente costoso para nubes de puntos muy grandes.

Los últimos pasos de esta primera variante de RANSAC se mantienen igual que Vanilla-RANSAC. Es decir, la generación de hipótesis H_i a partir de una muestra dada S_i, la evacuación L_i de la hipótesis generada y en la retención de la mejor hipótesis, *no* se sufren modificación alguna.

El Algoritmo 2 resume las modificaciones descritas arriba, al algoritmo Vanilla-RANSAC. Esta primera propuesta sigue un sesgo en la selección de muestreo, dando prioridad a los puntos con mayor densidad. Los puntos destacados se generan a partir de un preprocesado de datos que arroja un ’proxy’ de densidad, para cada punto. El ’proxy’ se genera a partir de la consulta de los k vecinos más cercanos *reversos* o RNN_k.

El término ’proxy’ hace referencia a un representante de una medida o propiedad, en este caso nos referimos a la densidad local.

Algorithm 2: DD-RANSAC (uniforme)

input : $D = \{o_1, o_2, \ldots, o_n\}$: Nube de puntos o base de datos
H: Hipótesis generada, basada en un determinado modelo
m: Tamaño mínimo de muestra
p: Probabilidad de obtener una muestra libre de outliers
δ: Umbral de error
k': Número de vecinos más cercanos

output: H^*: Mejor hipótesis evaluada
L^*: Mejor soporte

$k \leftarrow f(m, p)$ // ecuación 5
$H^* \leftarrow \emptyset$
$L^* \leftarrow \emptyset$

P_D /* Pre-procesado de los datos */
$D' \longleftarrow RNN_k(o_i, k') | \forall o_i \in D$ // ecuación 9
$E \overset{orden}{\longleftarrow} D'$
$G \longleftarrow {}_nC_m$ // n es el número de elementos de E

G_H /* Generación de Hipótesis */
for $i \leftarrow 1$ **to** k **do**
  $\overline{S_i} \leftarrow$ Selecciona el i-esimo elemento de G como muestra
  $H_i \leftarrow$ Calcular los parámetros del modelo, a partir de $\overline{S_i}$
E_H /* Evaluación de Hipótesis */
  $L_i \leftarrow$ Calcular el soporte de la hipótesis H_i con tolerancia δ
  if $L_i > L^*$ **then**
    $L^* \leftarrow L_i$
    $H^* \leftarrow H_i$

3.4.2. DD-RANSAC (Cargado)

De acuerdo con la RAE, un dado es: un objeto generalmente cúbico en cuyas caras aparecen puntos, que representan distintos números o figuras. Normalmente usado en juegos de azar.

Un dado convencional tiene forma poliédrica. Al arrojarlo, la cara que voltea hacia

arriba se considera el resultado de un ’tiro’. Cada una de las caras del dado tiene la misma oportunidad de caer hacia arriba, es decir, la probabilidad de que resulte una u otra cara es la misma. Esto se puede verificar tirando un dado repetidas veces y midiendo la distribución de probabilidad del resultado. Sin embargo, existe una forma de modificar esta distribución de probabilidad y sesgar un cierto resultado, es decir, el dado se modifica para que una cara especifica tenga mayor oportunidad de ser el resultado de un tiro. A esto se le conoce normalmente como un dado cargado.

Continuando con la misma hipótesis de trabajo, en esta segunda propuesta algorítmica vamos a utilizar el concepto del ’dado cargado’ para sesgar el muestreo de RANSAC en favor de los puntos con mayor densidad. De acuerdo con nuestra hipótesis de trabajo, esto brinda mayores probabilidades de encontrar muestras libres de ’outliers’ con un menor número de intentos.

Al igual que en el primer algoritmo propuesto, también vamos a utilizar la consulta de los k vecinos más cercanos *reversos* para obtener la densidad de cada punto de la base de datos. Una vez ordenada la base de datos, de mayor a menor densidad, se obtiene el conjunto E, ejemplificado con la ecuación 12.

A continuación, el objetivo es escoger como muestras, preferentemente, los elementos de los subconjuntos de E de mayor densidad. Para esto, seleccionaremos uno de los subconjuntos aleatoriamente, pero con una función de probabilidad *no uniforme*. Una vez escogido uno de los subconjuntos de E, ahora sí, procedemos a seleccionar aleatoriamente alguno de los elementos de dicho subconjunto. Esta vez, la selección aleatoria entre los elementos del mismo subconjunto tiene una función de probabilidad *uniforme*, ya que todos los puntos que pertenecen al mismo subconjunto poseen la misma densidad.

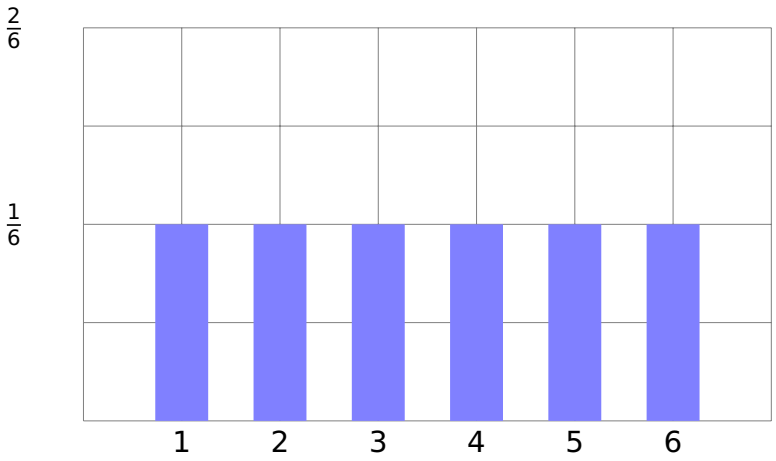


Figura 14. Función de probabilidad de un dado justo de seis caras.

La distribución uniforme, discreta, describe el comportamiento de una variable que puede tomar *n* valores distintos con la misma probabilidad cada uno de ellos. La Figura 14 representa la función de probabilidad de un dado convencional (de seis caras), donde cada una de las caras tiene la misma probabilidad de ser elegida al tirar dicho dado. En este caso se tiene la función $P(X) = 1/6$, con valores de $x = \{1, 2, \ldots, 6\}$

Sin embargo, una distribución *no uniforme* puede tener diferentes formas, por ejemplo, lineal, cuadrática o exponencial. En esta tesis y específicamente en esta propuesta algorítmica, donde utilizamos el modelo de un dado cargado, necesitamos proponer un tipo de función de probabilidad que va a adoptar la selección aleatoria de puntos muestrales. Dicho en otras palabras, el sesgo de densidad en el muestreo, propuesto como hipótesis de trabajo, está representado en la función de probabilidad del muestreo aleatorio.

En las Figuras 15 y 16 se observa una distribución *no uniforme*, discreta, del comportamiento de un ’dado’ generado a partir del conjunto *E* de la Figura 13 (ecuación 12). En dichas figuras se tienen tantas caras como subconjuntos tenga *E*, es decir, se tiene un dado de siete caras: (0, 1, 2, 3, 4, 5, 6). Estos números, mejor conocidos como *espacio muestral*, representan los índices del número de vecinos más cercanos reversos, acomodados en orden ascendente. Sin embargo, ahora estos valores tienen una probabilidad diferente cada uno.

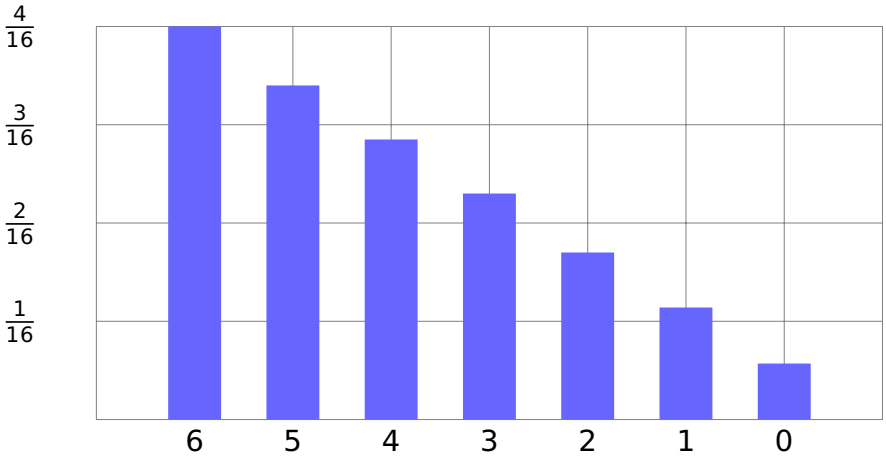


Figura 15. Función de probabilidad de un dado cargado, basado en el conjunto de la ecuación 12.

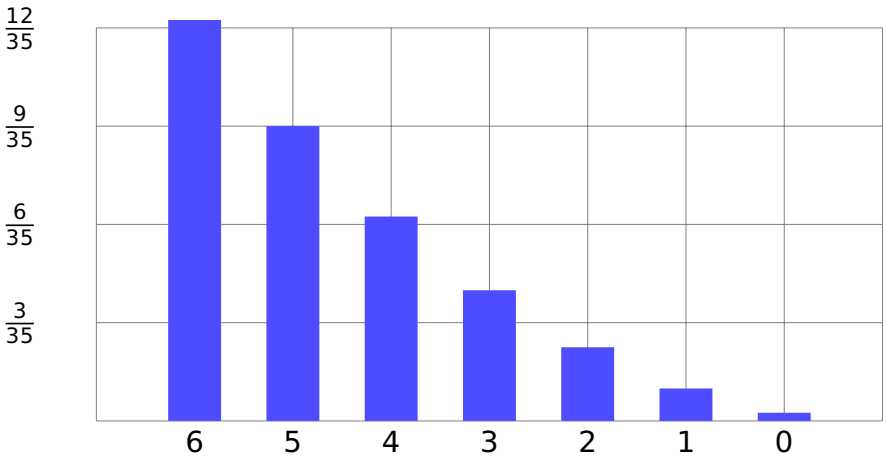


Figura 16. Función de probabilidad cuadrática de un dado cargado, basado en la ecuación 12.

La probabilidad de cada elemento del espacio muestral está asociada a la frecuencia con las que aparece dicho elemento como resultado de una selección aleatoria. Los elementos con mayor probabilidad son los que van a aparecer primero resultado y por lo tanto van a ser los primeros puntos seleccionados como muestra en nuestro algoritmo RANSAC. Este sesgo coincide plenamente con nuestra hipótesis de trabajo. Sin embargo, la forma de la función de probabilidad marca una clara diferencia entre las oportunidades que tiene una muestra u otra de ser seleccionada. En las Figuras 15 y 16 se aprecian claramente estas diferencias, es decir, cada elemento del espacio muestral tienen un cierto valor que representa su probabilidad de resultar seleccionado, dada una muestra aleatoria.

A continuación, vamos a retomar la Figura 13 para dar un ejemplo. Al calcular los vecinos más cercanos reversos de cada elemento de esta figura se obtiene un conjunto como el siguiente:

$$E = \{\{o_5\}, \{o_4\}, \{o_{15}, o_7, o_9, o_{10}, o_{13}\}, \{o_8, o_{14}\}, \{o_2, o_{11}, o_{12}\}, \{o_3, o_6\}, \{o_1\}\}$$

El primer subconjunto de E es el de mayor densidad, en este caso con valor de ’6’, el segundo subconjunto tiene densidad ’5’, el tercero aparece con densidad ’4’, el cuarto con densidad ’3’, el quito con densidad 2, el sexto con densidad ’1’ y finalmente el séptimo subconjunto con densidad ’0’. Esta cantidad de subconjuntos y sus respectivas etiquetas, son el espacio muestral de las Figuras 15 y 16. El objetivo es proponer una función de probabilidad que dé preferencia a los primeros subconjuntos, es decir, a los de mayor densidad. En la Figura 15 se tiene una función de probabilidad lineal y en la Figura 16 se tiene una función de probabilidad cuadrática. Ambas dan preferencia a los subconjuntos con mayor densidad. Sin embargo, el sesgo en la función cuadrática es más drástico.

Finalmente, cuando seleccionamos una muestra aleatoria de este ’dado cargado’, se espera obtener con mayor frecuencia un resultado de alta densidad. Estos resultados se utilizarían como muestra y si nuestra hipótesis de trabajo es correcta, entonces se podrían encontrar muestras libres de ’outliers’ con pocos intentos.

Algorithm 3: DD-RANSAC (cargado)

input : $D = \{o_1, o_2, \ldots, o_n\}$: Nube de puntos o base de datos
H: Hipótesis generada, basada en un determinado modelo
m: Tamaño mínimo de muestra
p: Probabilidad de obtener una muestra libre de outliers
δ: Umbral de error
k': Número de vecinos más cercanos

output: H^*: Mejor hipótesis evaluada
L^*: Mejor soporte

$k \leftarrow f(m, p)$ // ecuación 5
$H^* \leftarrow \emptyset$
$L^* \leftarrow \emptyset$

P_D /* Pre-procesado de los datos */
$D' \longleftarrow RNN_k(o_i, k') | \forall o_i \in D$ // ecuación 9
$E \overset{orden}{\longleftarrow} D'$

G_H /* Generación de Hipótesis */
def *tirarDado* *(E)*
  $Dado \longleftarrow |E|$ = número de caras
  $Dado \longleftarrow$ cargar $(Dado)$ // función cuadrática
  $\eta \longleftarrow$Aleatoriamente, elegir una cara del $Dado$ // distribución sesgada
  $s \longleftarrow$Aleatoriamente, elegir un elemento de e_η // distribución uniforme
  $E \longleftarrow E \setminus s$
  return s

for $i \leftarrow 1$ **to** k **do**
  $\overline{S_i} \leftarrow$ **tirarDados** (E_m) // tirar m dados diferentes
  $H_i \leftarrow$ Calcular los parámetros del modelo, a partir de $\overline{S_i}$
E_H /* Evaluación de Hipótesis */
  $L_i \leftarrow$ Calcular el soporte de la hipótesis H_i con tolerancia δ
  if $L_i > L^*$ **then**
    $L^* \leftarrow L_i$
    $H^* \leftarrow H_i$

Particularmente, el Algoritmo 3 está enmarcado en nuestra hipótesis de trabajo (los 'inliers' tienden a estar más cercanos entre sí y por el contrario, los 'outliers' son más dispersos; por lo tanto los puntos con mayor densidad tienen mayor probabilidad de generar muestras libres de 'outliers'), este algoritmo utiliza una variable aleatoria uniforme y una función de probabilidad no uniforme que aumente las posibilidades de recuperar los puntos con mayor densidad. En comparación con el Vanilla-RANSAC, el Algoritmo 3 posee una etapa de *preprocesado*, anterior a la generación de hipótesis, que permite identificar los puntos con mayor densidad y una etapa de *generación de hipótesis* (descrita en este capítulo), basada en una función de probabilidad *no*

uniforme que se actualiza con cada muestra obtenida.

3.4.3. DD-RANSAC (Dirigido)

En esta última propuesta algorítmica, también se utiliza la densidad de los puntos como guía para la obtención de muestras, priorizando las combinaciones de puntos con mayor densidad y dejando hasta el final todos aquellos puntos con baja densidad.

Se podría decir que esta propuesta es una modificación a la primer propuesta algorítmica (sección 3.4.1). En la implementación de los dos algoritmos anteriores se observó que existe un gran número de muestras obtenidas a partir de puntos de baja densidad, antes de dar con una muestra libre de 'outliers'. Esto es normal, pero la idea es reducir al mínimo las muestras con puntos de baja densidad. Para esto, se identificaron los subconjuntos de puntos de mayor densidad y se recuperaron *únicamente* las combinaciones de puntos dentro de estos subconjuntos más densos.

Para explicarlo mejor, supongamos que se tiene el F conjunto formado por cuatro subconjuntos: $F = \{A, B, C, D\}$. Estos subconjuntos se encuentran en orden de densidad, es decir, los elementos de A tienen la mayor densidad, los elementos de B y C tienen una densidad intermedia, pero siempre son más densos los elementos de B que de C y finalmente, los elementos de D poseen la densidad más baja. Solo por dar un ejemplo, supongamos que se tienen $\rho_A = 4$, $\rho_B = 3$, $\rho_C = 2$ y $\rho_D = 1$, de la siguiente manera:

$$A = \{a_1, a_2, a_3\}$$
$$B = \{b_1, b_2, b_3\}$$
$$C = \{c_1, c_2, c_3\}$$
$$D = \{d_1, d_2, d_3\}$$

El objetivo es dejar al final todas las muestras que impliquen a los elementos del subconjunto D e iniciar con las muestras donde estén implicados los elementos del subconjunto A. Para lograr esto, primero obtenemos las combinaciones de los subconjuntos:

$$_FC_2 = AB, AC, AD, BC, BD, CD$$

Además, se identifica la sumas de sus densidades con el propósito de ordenarlos:

$$AB = 12$$
$$AC = 8$$
$$BC = 6$$
$$AD = 4$$
$$BD = 3$$
$$CD = 2$$

A continuación, se procede a recuperar las combinaciones de tamaño m de los elementos pertenecientes a $\{A \cup B\}$ $\{A \cup C\}$, $\{B \cup C\}$, $\{A \cup D\}$, $\{B \cup D\}$ y $\{C \cup D\}$, en ese orden. Este nuevo *orden* cumple con el objetivo de dejar al final las muestras que impliquen a los elementos del subconjunto menos denso D.

Lo que nos quiere decir esta nueva forma de obtener un muestreo ordenado, además de nuestra hipótesis de trabajo, es que las muestras de mediana densidad tienen mayores posibilidades de ser libres de ’outliers’ que aquellas muestras formadas con una combinación de elementos densos y poco densos; ya que estos elementos de baja densidad podrían ser ’outliers’.

Las combinaciones de tamaño m de elementos de $\{A \cup B\}, \{A \cup C\}, \ldots, \{C \cup D\}$, generan todas de muestras S_i, utilizadas para su posterior evaluación en el algoritmo RANSAC. En este algoritmo, presentado a continuación (Algoritmo 4), las muestras *generadas* están representadas en el conjunto G. Cabe mencionar, que la cantidad de combinaciones de puntos posibles en G puede ser enorme, para conjuntos de puntos muy grandes. Por lo tanto, no se calcula *ninguna* combinación de puntos antes de empezar con el muestreo, por el contrario, se van calculando conforme el algoritmo solicita una nueva muestra.

Algorithm 4: DD-RANSAC (dirigido)

input : $D = \{o_1, o_2, \ldots, o_n\}$: Nube de puntos o base de datos
H: Hipótesis generada, basada en un determinado modelo
m: Tamaño mínimo de muestra
p: Probabilidad de obtener una muestra libre de outliers
δ: Umbral de error
k': Número de vecinos más cercanos

output: H^*: Mejor hipótesis evaluada
L^*: Mejor soporte

$k \leftarrow f(m, p)$ `// ecuación 5`
$H^* \leftarrow \emptyset$
$L^* \leftarrow \emptyset$

P_D `/* Pre-procesado de los datos */`
$D' \longleftarrow RNN_k(o_i, k') | \forall o_i \in D$ `// ecuación 9`
$E \overset{orden}{\longleftarrow} D'$
$F \overset{orden}{\longleftarrow} {}_E C_2$ `// el orden esta dado por la suma de densidad de los subconjuntos`
$G \longleftarrow {}_F C_m$ `// se obtienen las combinaciones de cada subconjunto de F`

G_H `/* Generación de Hipótesis */`
for $i \leftarrow 1$ **to** k **do**
  $\overline{S_i} \leftarrow$ Selecciona el i-esimo elemento de G como muestra
  $H_i \leftarrow$ Calcular los parámetros del modelo, a partir de $\overline{S_i}$
  E_H `/* Evaluación de Hipótesis */`
  $L_i \leftarrow$ Calcular el soporte de la hipótesis H_i con tolerancia δ
  if $L_i > L^*$ **then**
    $L^* \leftarrow L_i$
    $H^* \leftarrow H_i$

Capítulo 4. Metodología

Para medir el desempeño de las propuestas algorítmicas de esta tesis (descritas en el capítulo anterior), es necesario proponer un modelo que se quiera recuperar, por lo tanto, se propusieron tres modelos, con diferente dimensionalidad cada uno, para esta fase de experimentación.

1. Emparejamiento de una *línea recta*, con un cierto rango de error δ, a partir de una nube de puntos (modelo $2D$).
2. Emparejamiento de una *circunferencia*, con un cierto rango de error δ, a partir de una nube de puntos (modelo $3D$).
3. *Emparejamiento de dos nubes de puntos*, ante transformaciones afines, inserciones, oclusiones y un cierto rango de error δ (modelo $6D$).

Los modelos anteriores se eligieron para mostrar las propiedades de nuestras tres propuestas algorítmicas y para comparar cada uno de los resultados con el algoritmo base *'Vanilla-RANSAC'*.

4.1. Modelos para RANSAC

RANSAC utiliza muestreo para encontrar los parámetros que describan al modelo que alcance el mayor consenso entre los elementos de una base de datos. Esto significa que se debe de tener un modelo definido al cual se quiere llegar. Esto mismo supone cierto conocimiento previo de la base de datos.

Es importante subrayar que los algoritmos propuestos en el capitulo 3 no alteran su forma, cuando se aplica un modelo u otro. Sin embargo, la cardinalidad de las muestras tomadas cambia conforme al modelo utilizado. Esto repercute en la cantidad de combinaciones posibles de elementos de una base de datos, por lo tanto, también modifica el número de intentos necesarios para encontrar una probable muestra libre de 'outliers'. A continuación, los detalles de los tres modelos utilizados.

4.1.1. Emparejamiento de una línea recta

La ecuación general de una línea recta tiene la siguiente forma:

$$y = m\,x + b \tag{15}$$

donde m y b son constantes por definir.

Dada la ecuación de la línea recta, se observa que es necesario encontrar los parámetros m y b para definir una línea recta especifica de toda la familia. Por otro lado, una manera de encontrar estas incógnitas es utilizar las coordenadas de dos puntos que formen parte de la línea recta. En otras palabras, se necesitan pares de puntos (p_1, p_2) para determinar los parámetros de la ecuación de una línea recta única, en el plano.

Dados dos puntos, $P_1 = (x_1, y_1)$ y $P_2 = (x_2, y_2)$, se puede obtener obtener como primer parámetro la pendiente de la recta, con la siguiente expresión:

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

Una vez calculada la pendiente, ahora se puede obtener el segundo parámetro b, conocido como el desplazamiento respecto al origen:

$$b = y_2 - m\,x_2$$

En el experimento con RANSAC, cada par de puntos se obtiene como parte del muestreo de la nube de puntos, la obtención de los parámetros (m, b) representa la generación de una hipótesis H_i del algoritmo y finalmente, el soporte de cada hipótesis se obtiene a partir del consenso de toda la base de datos con la hipótesis generada. En la Figura 7 se puede observar un ejemplo de las nubes de puntos generadas para este experimento.

4.1.2. Emparejamiento de una circunferencia

La ecuación general de un círculo tiene la siguiente forma:

$$x^2 + y^2 + 2gx + 2fy + c = 0 \tag{16}$$

donde g, f y c son constantes.

Las coordenadas del centro del círculo son $(-g, -f)$ y la expresión para determinar el radio está dada por:

$$r^2 = g^2 + f^2 - c$$

Por lo tanto, la Ecuación 16 y los parámetros g, f y c , definidos, describen un círculo *único* en el plano. Una de las formas para satisfacer estos parámetros es mediante las coordenadas de tres puntos que formen parte del círculo. Dicho de otro modo, las coordenadas de tres puntos en el plano son suficientes para describir un círculo único en el mismo plano; debido a que la información de la terna de puntos satisface los parámetros de la ecuación general de dicho círculo.

Dados tres puntos en el plano, se evalúa la Ecuación 16 para cada uno de los puntos. Esto forma un sistema de tres ecuaciones con tres incógnitas, g, f y c. Posteriormente se resuelve el sistema de ecuaciones y se obtiene el valor de cada uno de los parámetros.

En este experimento con RANSAC, el ajuste de una circunferencia, se utilizarán ternas de puntos como muestras, después, con cada terna de puntos se obtendrán los parámetros que describen el modelo como hipótesis H_i y finalmente se evaluará cada hipótesis generada conforme a la misma premisa de los algoritmos abordados en el capítulo anterior; es decir, se realiza un consenso sobre toda la nube puntos, para obtener la cantidad de elementos que son afines con una hipótesis determinada. En la Figura 7 se puede observar un ejemplo del emparejamiento de una circunferencia en una nube de puntos generadas para este experimento.

4.1.3. Emparejamiento de dos nubes de puntos

Los detalles sobre el problema del emparejar dos nubes de puntos se encuentran en la Sección 1.1. De acuerdo con la Ecuación 1, presentada en esa misma sección, para lograr el emparejamiento de dos nubes de puntos ante transformaciones afines, inserciones, oclusiones y un cierto ruido δ se requieren de seis parámetros (a, b, c, d, r, s). Para satisfacer todos los parámetros, primero observemos que sucede dados dos puntos, $P' = (x', y)$ y $P = (x, y)$.
En este caso, P' es igual a P transformada, ante una determinada transformación afín. De acuerdo con la Ecuación 1, obtenemos la siguiente expresión.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} r \\ s \end{pmatrix}$$

Este par de puntos genera dos ecuaciones lineales, sin embargo, se tienen seis incógnitas (a, b, c, d, r, s); ahora tomaremos tres pares de puntos, es decir, una terna del conjunto original P_1, P_2, P_3 y una terna del conjunto transformado P'_1, P'_2, P'_3 para obtener un conjunto de seis ecuaciones lineales que permitan encontrar los seis parámetros (a, b, c, d, r, s). Juntando las ecuaciones y reescritas en forma matricial, se obtiene la siguiente expresión:

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \\ y'_1 \\ y'_2 \\ y'_3 \end{pmatrix} = \begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ r \\ c \\ d \\ s \end{pmatrix}$$

De esta forma es fácil de identificar los parámetros dados y los parámetros desconocidos de la transformación.

Recapitulando, dadas dos terna de puntos $\{P_1, P_2, P_3\}$ y $\{P'_1, P'_2, P'_3\}$, es posible obtener una función que transforma las coordenadas una terna a las coordenadas de la otra. Esta función representa una hipótesis H_i del algoritmo RANSAC, la cual es evaluada en cada elemento de la nube de puntos original. Esto quiere decir a cada punto de la nube de puntos original se le aplica la función obtenida y se determina si coincide con algún otro punto de la nube transformada a menos de una distancia δ, si existe una coincidencia, entonces suma un punto al soporte calculado de dicha hipótesis de acuerdo con el algoritmo RANSAC.

Es importante subrayar que los experimentos propuestos tienen que evaluar los modelos dentro de las mismas condiciones de la hipótesis de trabajo. En el caso del ajuste de la línea recta y la circunstancia, ambos cumplen con la hipótesis, ya que los puntos más densos se encuentran normalmente muy cercanos al modelo. Para el caso del registro de dos nubes de puntos, la hipótesis no se cumple cuando se generan nubes de puntos con datos aleatorios, por lo tanto, para desarrollar este experimento y cumplir con la hipótesis de trabajo, se generaron diez letreros con un patrón determinado a los cuales se les pueden aplicar transformaciones afines, inserciones y oclusiones. En la Figura 17 se muestra un ejemplo de este experimento; generación de una nube de puntos a partir de un letrero, transformación y emparejamiento.

Se dice que tienen coherencia espacial, todas aquellas instancias donde se pueden descubrir patrones de una cierta figura. La coherencia espacial de los letreros generados para este experimento permitirá evaluar el emparejamiento de dos nubes de puntos bajo las mismas condiciones de nuestra hipótesis de trabajo.

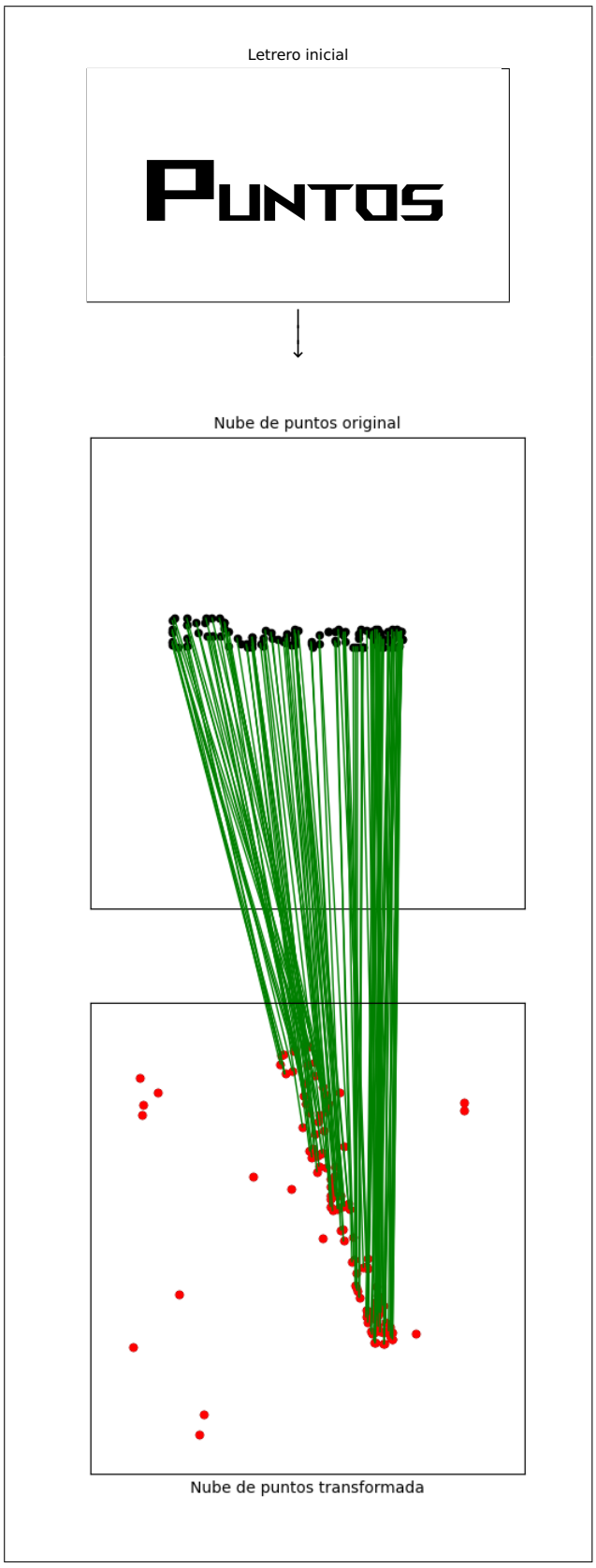


Figura 17. Emparejamiento de dos nube de puntos de tamaño $|D| = 120$, generadas a partir de la detección de bordes de un letrero (recuadro superior); la nube de puntos negros es la 'original' y la nube de puntos rojos (recuadro inferior), fue sometida a una transformación afín, 15 % de inserciones y 15 % de oclusiones. Las líneas verdes representan las coincidencias encontradas a partir de los parámetros del mejor modelo encontrado.

4.2. Evaluación de las propuestas algorítmicas

Dados los tres modelos anteriores (emparejamiento de una *línea recta*, emparejamiento de una *circunferencia* y emparejamiento de dos nubes de puntos), así como cada uno de los algoritmos propuestos en el Capítulo 3; se realizaron los siguientes experimentos:

1. Evaluación del número de muestras tomadas antes de encontrar una muestra libre de ’outliers’, cuya evaluación de la hipótesis generada alcance un determinado soporte, es decir, el *trabajo del algoritmo*.
2. Dado un número mínimo k de hipótesis generadas, se evalúa el cociente de los experimentos exitosos entre el número de experimentos realizados, es decir, la *eficiencia del algoritmo*.

Los algoritmos propuestos en el capitulo 3 se evaluaron con cada uno de los modelos presentados (emparejamiento de una línea recta, de una circunferencia y de dos nubes de puntos). Las pruebas se implementaron en el lenguaje de programación Python 3.6, en un servidor del Departamento de Ciencias de la Computación del CICESE con sistema operativo GNU/Linux, Ubuntu 21.04 y con las siguientes características de ’hardware’:

- Procesador Intel(R) Xeon(R) CPU E7-4809 v3 @ 2.00GHz.
- CPU habilitada x8
- Memoria RAM de 256 GB.

4.2.1. Trabajo del algoritmo

Se pusieron a prueba los tres algoritmos propuestos en el Capítulo 3 más el algoritmo base ’Vanilla-RANSAC’, con cada uno de los modelos de la Sección 4.1. Se generaron 50 configuraciones diferentes de cada modelo y se resolvieron con diferentes porcentajes de ’inliers’ (30 %, 45 %, 60 %, 75 %, 90 %). En este experimento se busca obtener el número de iteraciones necesarias para llegar a una primera solución; cuya evaluación arroje un soporte igual o mayor al porcentaje de ’inliers’ con la que fue construida la nube de puntos, en otras palabras, se quiere saber:

1. ¿Cuál de los algoritmos propuestos llega más rápido a una solución?
2. ¿Cuál es la ventaja relativa de este algoritmo con respecto al algoritmo ’Vanilla’?

4.2.2. Eficiencia del algoritmo

Se pusieron a prueba los tres algoritmos propuestos en el Capítulo 3 más el algoritmo base, ’Vanilla-RANSAC’, con cada uno de los modelos de la Sección 4.1. En cada modelo de este experimento se generaron *100* nubes de puntos diferentes de *5* diferentes tamaños y se resolvieron con los siguientes porcentajes de ’inliers’: 30 %, 45 %, 60 %, 75 % y 90 %. En este experimento existe una cantidad k de muestras que se deben evaluar antes de que el algoritmo se detenga (Ecuación 5), si dentro de estas primeras k muestras se obtiene una hipótesis H_i con el suficiente soporte, se añade un punto porcentual a la medida de eficiencia, por el contrario, si no se llega al soporte necesario; la respuesta del algoritmo es nula. La eficiencia de cada uno de los algoritmos es igual a la cantidad de puntos porcentuales alcanzados en cada configuración, es decir, se generaron $5 \times 5 \times 100 = 2500$ nubes de puntos diferentes, en cada uno de los siguientes modelos de la Sección 4.1.

En el siguiente capítulo presentaremos los resultados obtenidos de la fase experimental de esta tesis, no sin antes mencionar, que cada una de las métricas propuestas (trabajo del algoritmo y eficiencia del algoritmo) se evaluó con las combinaciones de cuatro algoritmos y tres modelos, es decir, doce experimentos para cada métrica.

Capítulo 5. Resultados

Los resultados de la fase experimental de este trabajo de tesis se presentan de dos manera diferentes; ’gráfico de lineas’ para el trabajo realizado por el algoritmo y ’mapa de calor’ para la eficiencia del algoritmo.

5.1. *Trabajo del algoritmo*

5.1.1. Emparejamiento de una línea recta

Los resultados del emparejamiento de una línea recta se encuentran en la Figura 18, ahí se observa la cantidad de muestras realizadas por cada algoritmo propuesto, para obtener una primer hipótesis H_i que alcance un soporte igual o mayor al que se indica en la variable independiente. De los cuatro algoritmos probados, el más rápido fue el algoritmo *’Dirigido’*, seguido por el algoritmo ’Uniforme’, después el algoritmo ’Cargado’ y finalmente ’Vanilla’. La diferencia entre el más rápido y el más tardado es de dos ordenes de magnitud (esta diferencia se observa claramente en la Figura 19).

Como era de esperarse, en ’Vanilla-RANSAC’ se observa una disminución en el número de muestras tomadas aleatoriamente conforme aumenta el soporte de la nube de puntos. Al tener más puntos pertenecientes al modelo, es lógico que la probabilidad de obtener una muestra libre de ’outliers’ aumente y el número de hipótesis a evaluar disminuya, por esta razón se observa una pendiente negativa en ’Vanilla-RANSAC’; sin embargo, en el algoritmo *’Dirigido’* no se observa con claridad dicha pendiente.

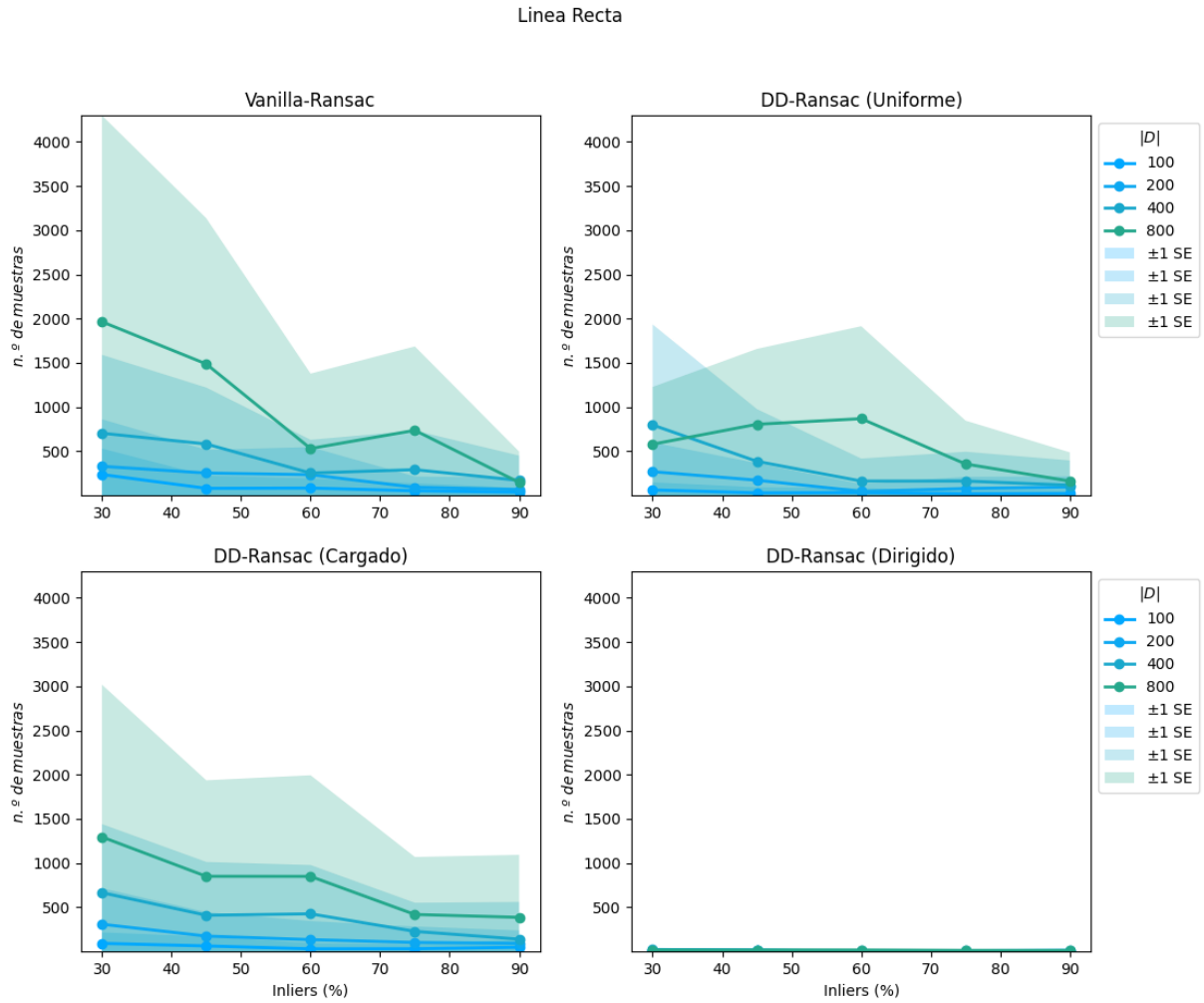


Figura 18. Comparación del trabajo de los cuatro algoritmos presentados. Evaluación de cuatro diferentes tamaños de nube de puntos $|D|$ con cinco diferentes porcentajes de 'inliers' cada uno. Cada punto representa el promedio de 50 casos y tienen ±1 error estándar asociado (transparencias de color).

Para apreciar con mayor claridad la diferencia que existe entre los algoritmos de la Figura 18, a continuación, en la Figura 19 se comparan únicamente las nubes de puntos más grandes, es decir, $D = 800$ y se cambia la escala a logarítmica. Esto resalta la diferencia en el orden de magnitud de los resultados.

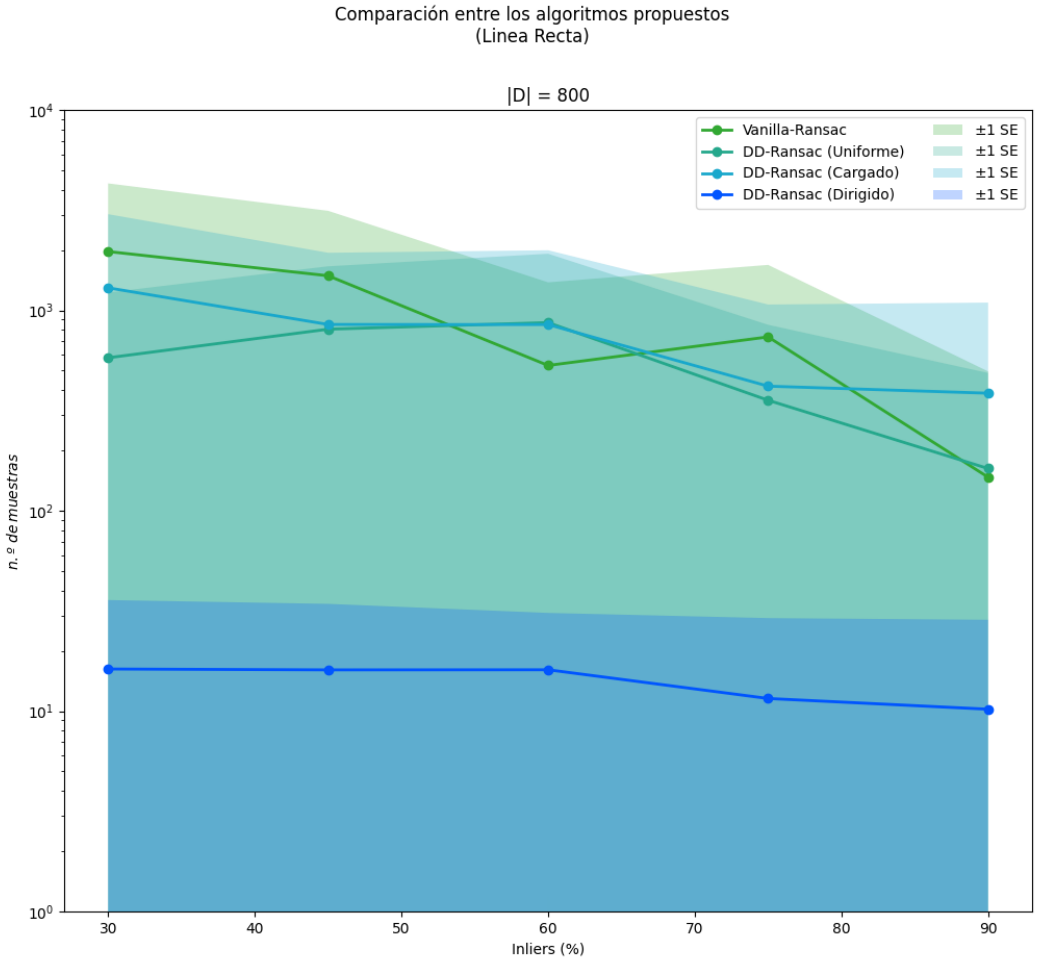


Figura 19. Comparación del trabajo realizado en cada uno de los cuatro algoritmos presentados en la Figura 18, en escala *logarítmica*, se incluye únicamente el tamaños de nube de puntos $|D| = 800$.

5.1.2. Emparejamiento de una circunferencia

En la Figura 20 se observa el tamaño del muestreo que cada algoritmo necesitó para obtener una primera hipótesis H_i con un soporte igual o mayor al indicado por la variable independiente. En orden descendente, el trabajo de los cuatro algoritmos puestos a prueba resultó de la siguiente forma: algoritmo *'Dirigido'*, 'Vanilla', 'Cargado' y finalmente la variante 'Uniforme'. Esto quiere decir, que el algoritmo *'Dirigido'* fue el más rápido de todos; con una diferencia de *tres ordenes de magnitud* contra 'Vanilla-RANSAC' y de *cuatro ordenes de magnitud* en comparación con el algoritmo el más lento.

Al igual que el modelo anterior (emparejamiento de una línea recta), en este modelo se observa que ’Vanilla-RANSAC’ disminuye el número de muestras tomadas conforme aumenta el soporte de la nube de puntos (variable independiente), además, esto mismo se observa con más claridad en los casos del algoritmo ’Cargado’ y ’Uniforme’; no así, para el algoritmo más rápido.

En la Figura 21 se comparó el tamaño de nube más grande ($D = 800$) de cada algoritmo en este modelo. En la Figura 20 se hace notar la diferencia que existe en el orden de magnitud de los resultados.

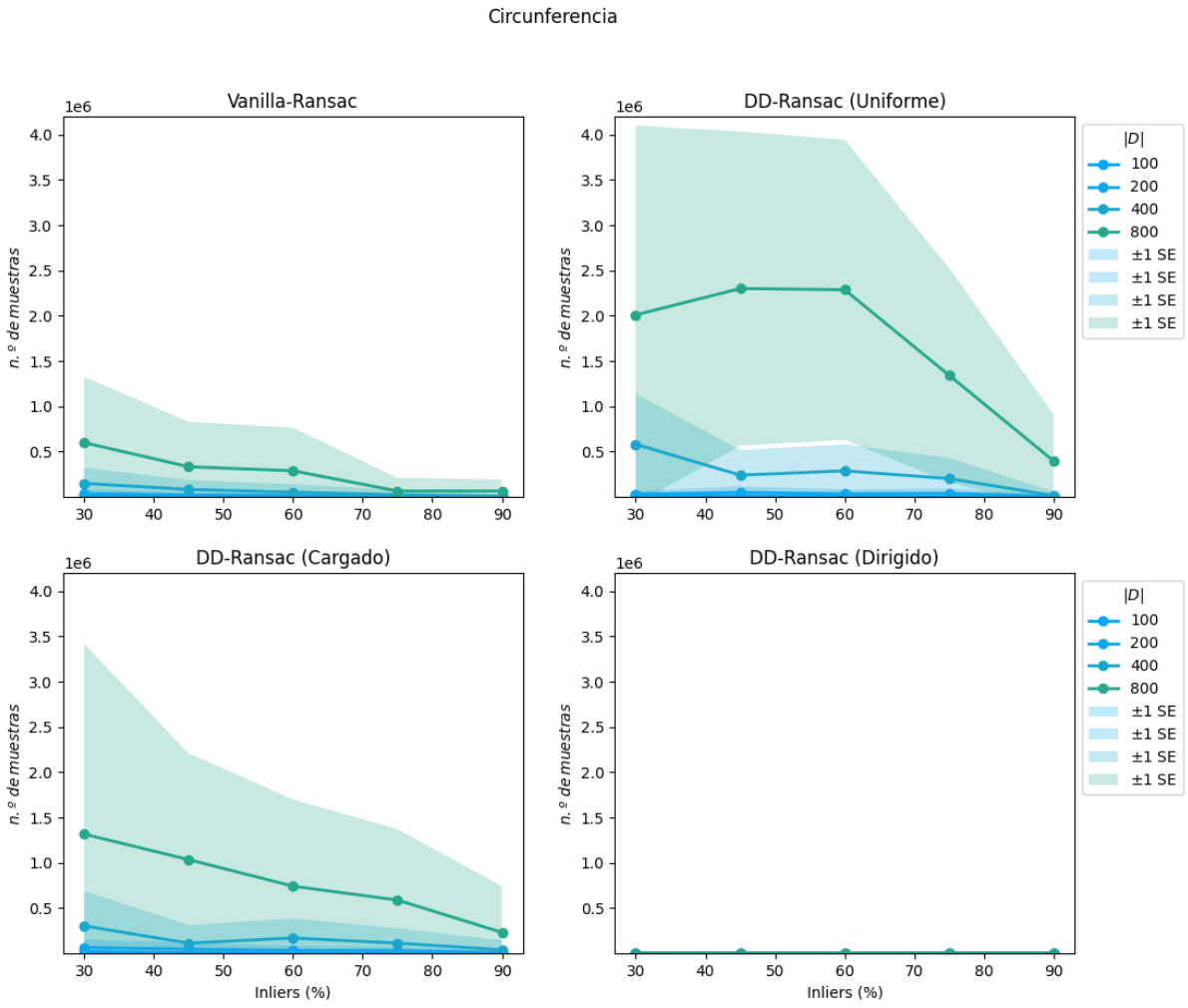


Figura 20. Comparación del trabajo de los cuatro algoritmos presentados al emparejar la ecuación de una circunferencia en una nube de puntos. Se evaluaron cuatro diferentes tamaños de nube de puntos $|D|$ con cinco diferentes porcentajes de ’inliers’ en cada tamaño de nube. Cada punto de la gráfica representa el promedio de 50 casos y tienen ±1 error estándar asociado (transparencias de color).

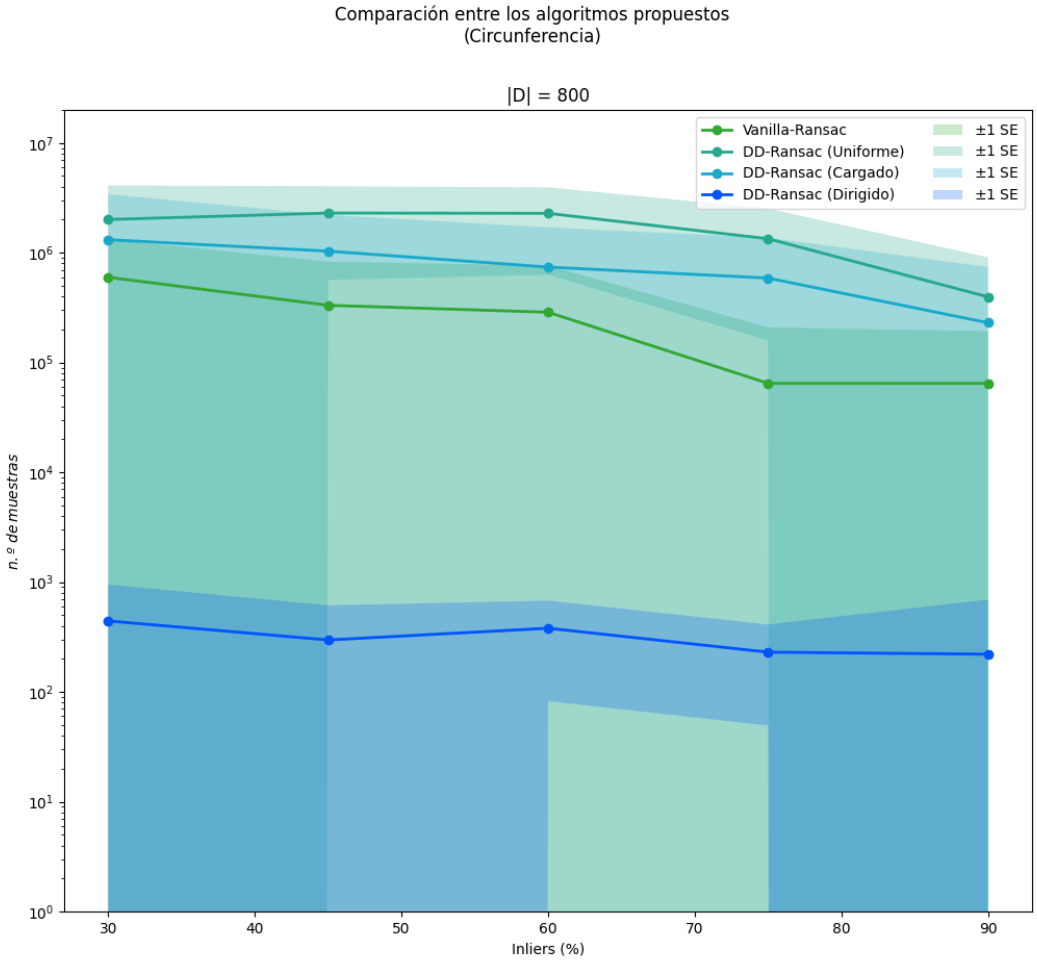


Figura 21. Comparación del trabajo realizado en cada uno de los cuatro algoritmos presentados en la Figura 20 en escala *logarítmica*. Únicamente se incluyó el tamaños de nube $|D| = 800$. Cada punto de la gráfica representa el promedio de 50 casos y tienen ±1 error estándar asociado (transparencias de color).

5.1.3. Emparejamiento de dos nubes de puntos

En este experimento se cambió la escala de las nubes de puntos propuestas inicialmente; resultaba enorme el tiempo de ejecución de los algoritmos, así que finalmente se probaron conjuntos de puntos de los siguientes tamaños: 20, 40, 80 y 160.
Estas nubes de puntos puestas a prueba también permiten distinguir el comportamiento de cada algoritmo propuesto en comparación con los demás.

Al igual que en los modelos anteriores, en la Figura 22 se tiene el número de muestras necesarias para obtener una primera hipótesis H_i, que empareje dos nubes de puntos con un soporte igual o mayor al indicado por la variable independiente. Cada

uno de los algoritmos resolvió todos los letreros generados y se obtuvo el promedio de estos, además se calculó el error estándar asociado de cada punto para representarlo en la gráfica con color transparente.

En este experimento también comprueba que la hipótesis de trabajo es correcta, ya que el algoritmo *'DD-RANSAC Dirigido'* (gráfica inferior derecha) necesita menor número de muestras para resolver el mismo problema que los demás algoritmos, además, el error estándar asociado de cada punto se reduce con este mismo algoritmo, en comparación con los otros algoritmos.

Posteriormente se compararon únicamente las nubes de puntos de mayor tamaño para apreciar el cambio de escala entre los resultados de cada uno de los algoritmos propuestos para el emparejamiento de dos nubes de puntos. Estos resultados se presentan en la Figura 23.

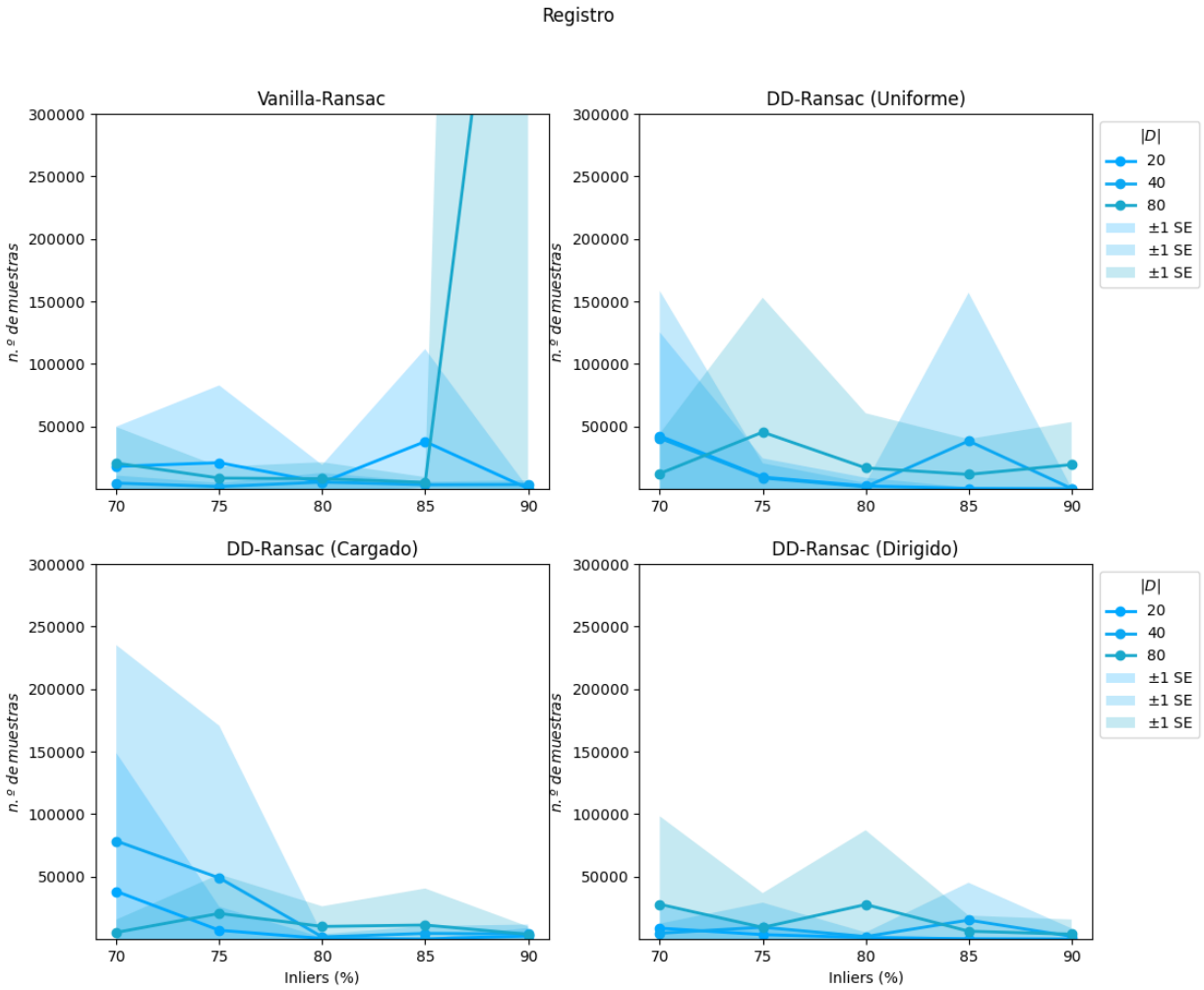


Figura 22. Comparación del trabajo de los cuatro algoritmos presentados al emparejar dos nubes de puntos, una original y la otra sometida a una transformación afín, inserciones y oclusiones. Se evaluaron cuatro diferentes tamaños de nube de puntos $|D|$ con cinco diferentes porcentajes de 'inliers' cada uno. Cada punto de la gráfica representa el promedio de los diez letreros generados con ±1 error estándar asociado (transparencias de color).

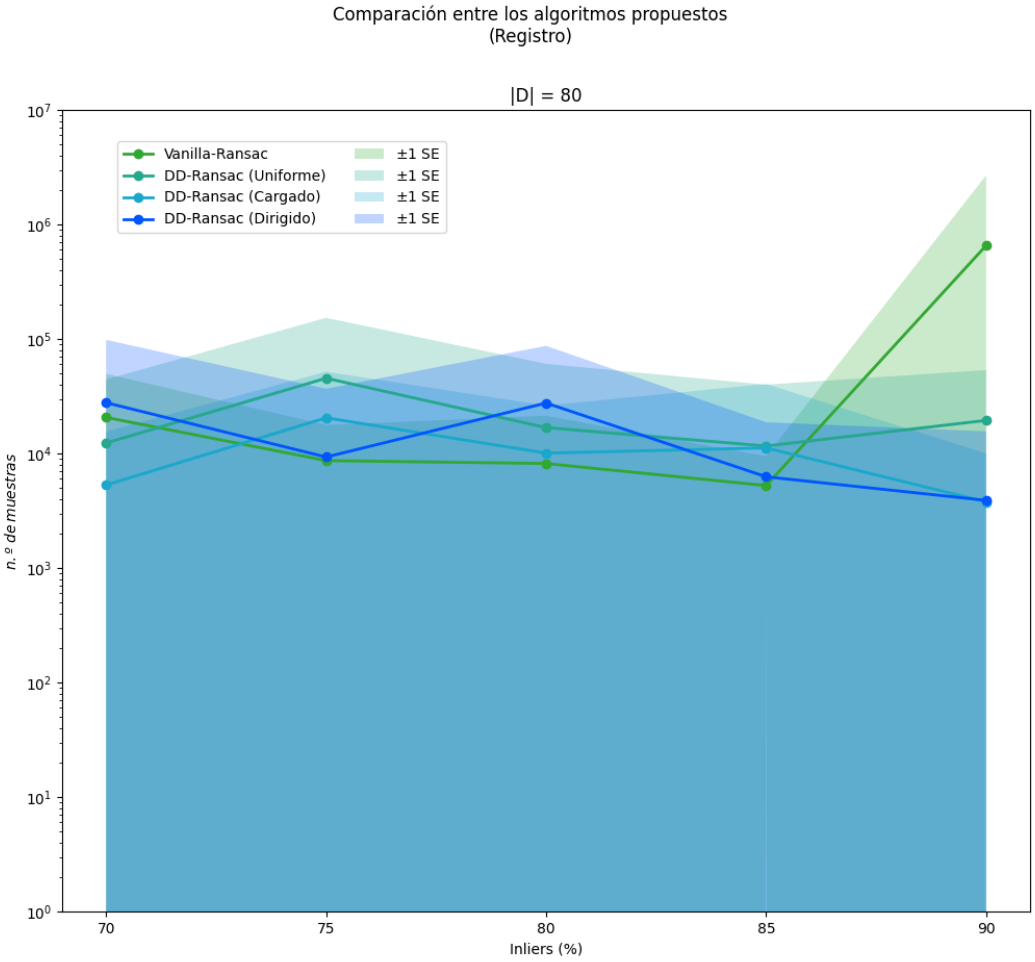


Figura 23. Comparación del trabajo realizado en cada uno de los cuatro algoritmos presentados en la Figura 22, presentado en escala *logarítmica*, únicamente se incluyen las nubes de puntos de tamaño $|D| = 80$.

5.2. *Eficiencia del algoritmo*

5.2.1. Emparejamiento de una línea recta

En la Figura 24 se observan los resultados de cada uno de los algoritmos propuestos para el modelo de la línea recta. 'Vanilla-RANSAC' es la referencia de la eficiencia alcanza como resultado de un muestreo aleatorio. Como ya se esperaba, entre mayor cantidad de 'inliers' tenga una nube de puntos, se obtiene una mayor eficiencia; ya que se tiene mayor probabilidad de encontrar una muestra libre de 'outliers' en los primeros k intentos. Sin embargo, ninguna configuración probada con 'Vanilla-RANSAC' alcanza el 100 % de eficiencia en este modelo.

Los algoritmos *'DD-RANSAC Uniforme'* y *'DD-RANSAC Cargado'* presentan más cuadros con tonalidades verdes y con mayor porcentaje de eficiencia en comparación con 'Vanilla-RANSAC', sin embargo, ninguna configuración logra el 100 % en cualquiera de los algoritmos mencionados.

El algoritmo *'DD-RANSAC Dirigido'* (mapa derecho inferior) de la Figura 24 es claramente el más eficiente; tiene la mayor cantidad de casillas en color verde y el 40 % de las configuraciones logran un 100 % de eficiencia, es decir, *'DD-RANSAC Dirigido'* alcanza el mayor soporte de cada configuración dentro de los primeros k intentos (la mayoría de las veces), en la peor de las configuraciones probadas, el algoritmo arroja una respuesta nula el 19 % de las veces. Resulta interesante observar que estas 'peores configuraciones' son las pertenecientes a nubes de puntos con mayor cantidad de 'inliers', es decir, 75 % y 90 %; este comportamiento resulta contraintuitivo, ya que es opuesto al comportamiento de los demás algoritmos probados.

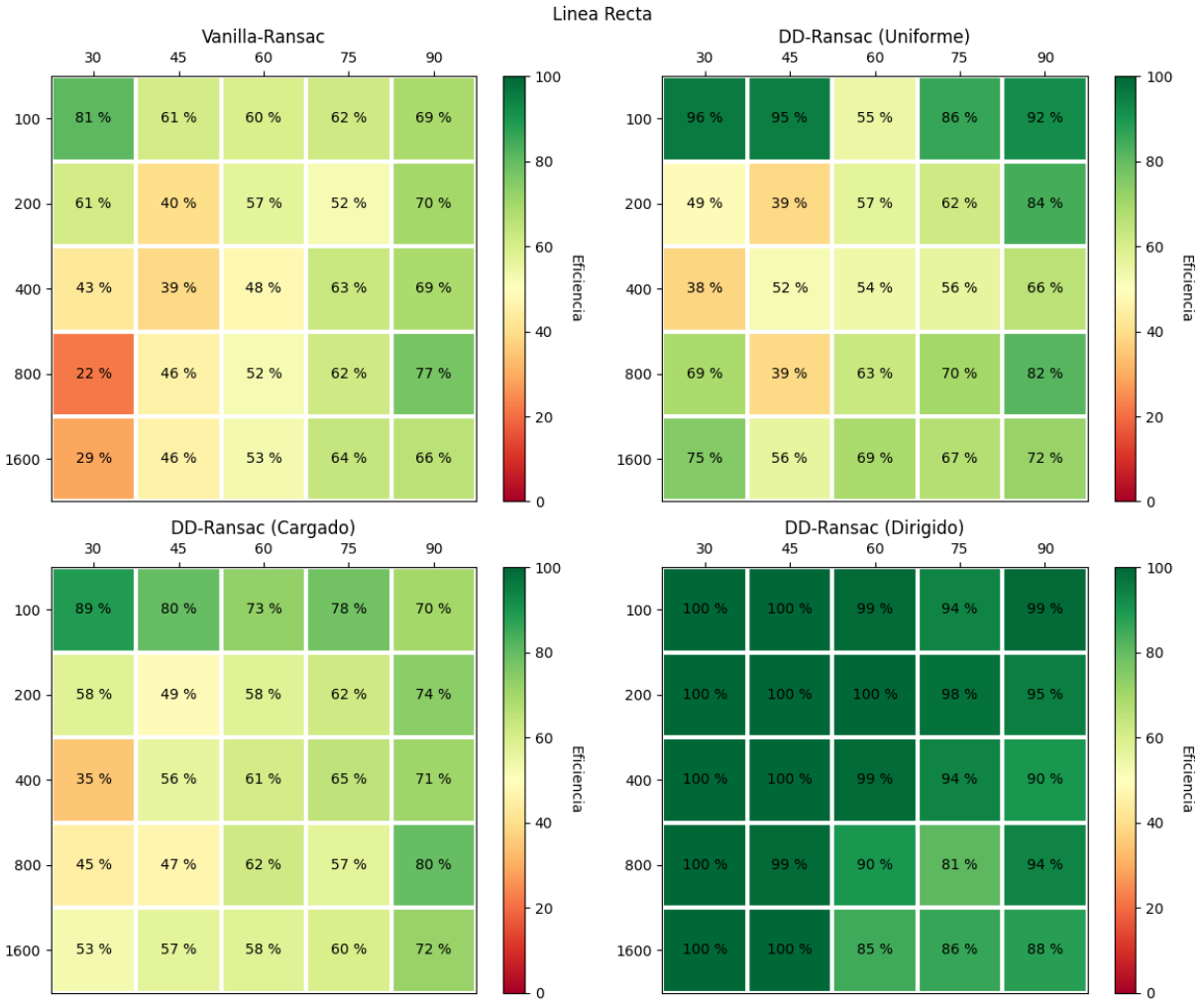


Figura 24. *Eficiencia* de los algoritmos propuestos, probados con el modelo del *emparejamiento de una línea recta* con diferentes configuraciones de nubes de puntos, presentados como mapas de calor.

5.2.2. Emparejamiento de una circunferencia

En la Figura 25 se observan los resultados de cada uno de los algoritmos propuestos para el modelo de la Circunferencia. En este caso, los algoritmos ’DD-RANSAC Uniforme’, ’DD-RANSAC Cargado’ y ’Vanilla-RANSAC’ no presentan mayor diferencia, cualitativamente hablando, estos tres algoritmos arrojan un mapa de calor predominantemente rojo o con baja eficiencia. Alrededor del 94 % de las configuraciones no superan el 50 % de eficiencia. A pesar de la baja eficiencia obtenida, estos tres algoritmos, en general, se comportan de acuerdo a lo esperado: ”entre mayor cantidad de ’inliers’ tenga una nube de puntos, se obtiene una mayor eficiencia”, aunque no es tan marcada la diferencia en algunos cosos.

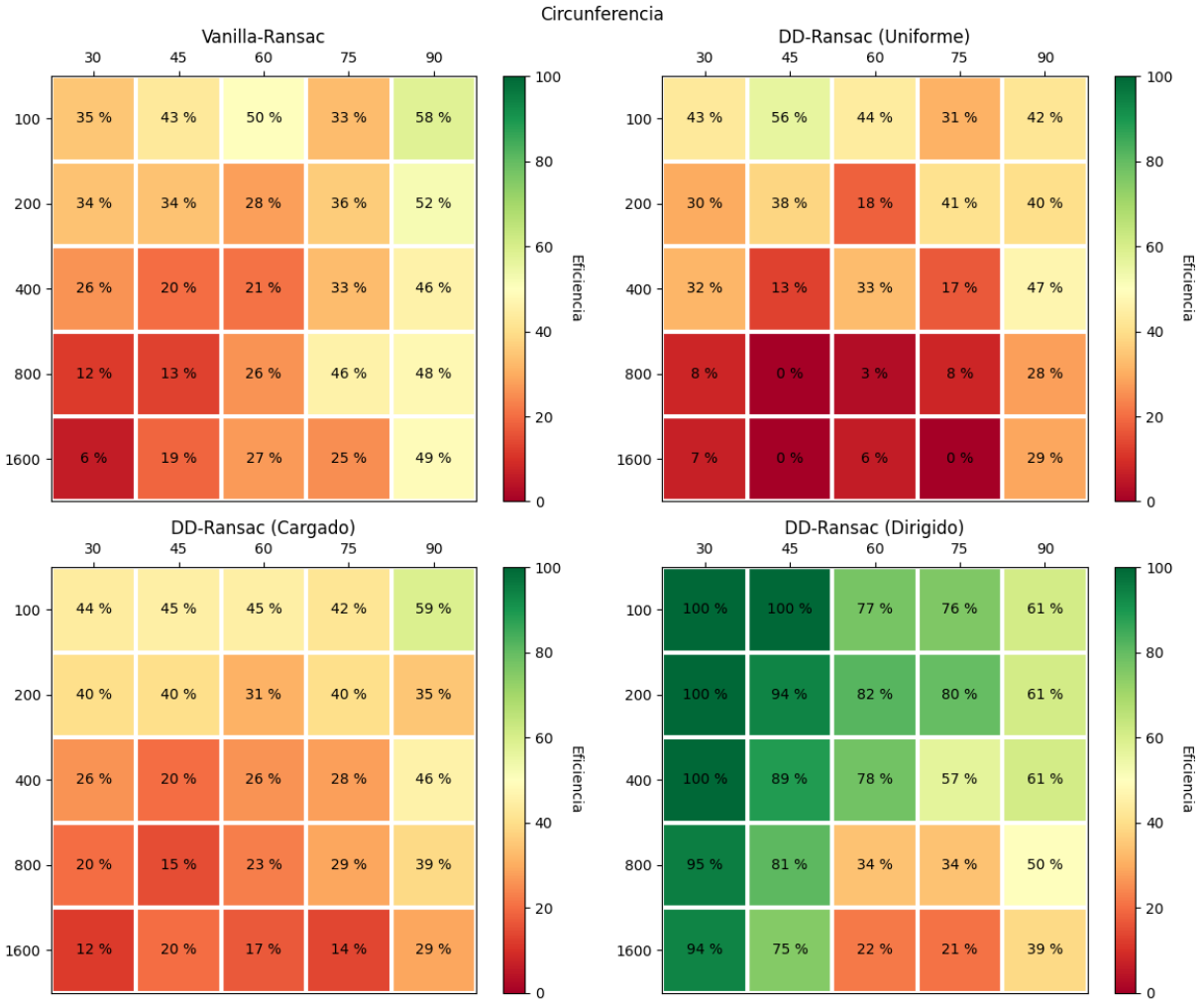


Figura 25. *Eficiencia* de los algoritmos propuestos, probados con el modelo del *emparejamiento de una circunferencia* con diferentes configuraciones de nubes de puntos, presentados como mapas de calor.

En este modelo, se observa muy claramente cuál es el algoritmo más eficiente; *'DD-RANSAC Dirigido'* una vez más. Sin embargo, una vez más se observa en este experimento que las configuraciones que obtienen menor eficiencia son las nubes de puntos con mayor cantidad de 'inliers': 75 % y 90 %. Nuevamente, este comportamiento resulta *contraintuitivo*, ya que es opuesto al comportamiento de los otros tres algoritmos probados, es decir, el comportamiento normalmente esperado.

5.2.3. Emparejamiento de dos nubes de puntos

Este modelo de *6 dimensiones* se ha cambiado de escala en el tamaño de las nubes de puntos para obtener resultados en el menor tiempo posible, de tal forma que se

generaron $5 \times 5 \times 100 = 2500$ nubes de puntos diferentes para ser resueltas con cada uno de los algoritmos propuestos en el Capitulo 3.

En la Figura 26 se presentan los resultados de cada uno de los algoritmos propuesto, puestos a prueba con el modelo del *emparejamiento de dos nubes de puntos*. En los mapas superior e inferior izquierdos, de la figura, se tiene expresada la eficiencia de los algoritmos ’Vanilla-RANSAC’ y ’DD-RANSAC Cargado’ respectivamente. Cualitativamente, no presentan mayor diferencia; estos algoritmos arrojan un mapa de calor predominantemente rojo y amarillo. En estos dos algoritmos se observa que las configuraciones con menor eficiencia son las de mayor cantidad de ’inliers’, es decir, 85 % y 90 %. En el lado derecho de la Figura 26 se observan los mapas de calor de los algoritmos *’DD-RANSAC Uniforme’* y *’DD-RANSAC Dirigido’*; en ambos algoritmos se observa un sesgo, muy ligero, donde las configuraciones con mayor eficiencia son las de menor cantidad de ’inliers’, es decir, los bloques más rojos de cada mapa de la figura se colocan preferentemente del lado derecho.

Hay que mencionar que existen dos parámetros en estos algoritmos, que hasta ahora simplemente han pasado a segundo plano. Probablemente, el aparente sesgo mencionado en el párrafo anterior esté relacionado con alguno los siguientes dos parámetros.

1. La k de la consulta kNN, este parámetro pertenece al número de vecinos más cercanos que arroja la consulta; hasta ahora este parámetro está definido como un pequeño porcentaje de la cardinalidad de la nube de puntos a tratar. El resultado de kNN influye directamente en la consulta RNN y por lo tanto en el orden del muestreo de los tres algoritmos propuestos ’DD-RANSAC’.
2. La Ecuación 5 calcula el número de muestras necesarias para obtener con cierta certeza al menos una muestra libre de ’outliers’, este número disminuye considerablemente, conforme aumenta la cantidad de ’inliers’ de la nube en cuestión.

Cualquiera de estos dos parámetros mencionados podría ser una de las razones por las que se observa un aparente sesgo en los mapas de calor.

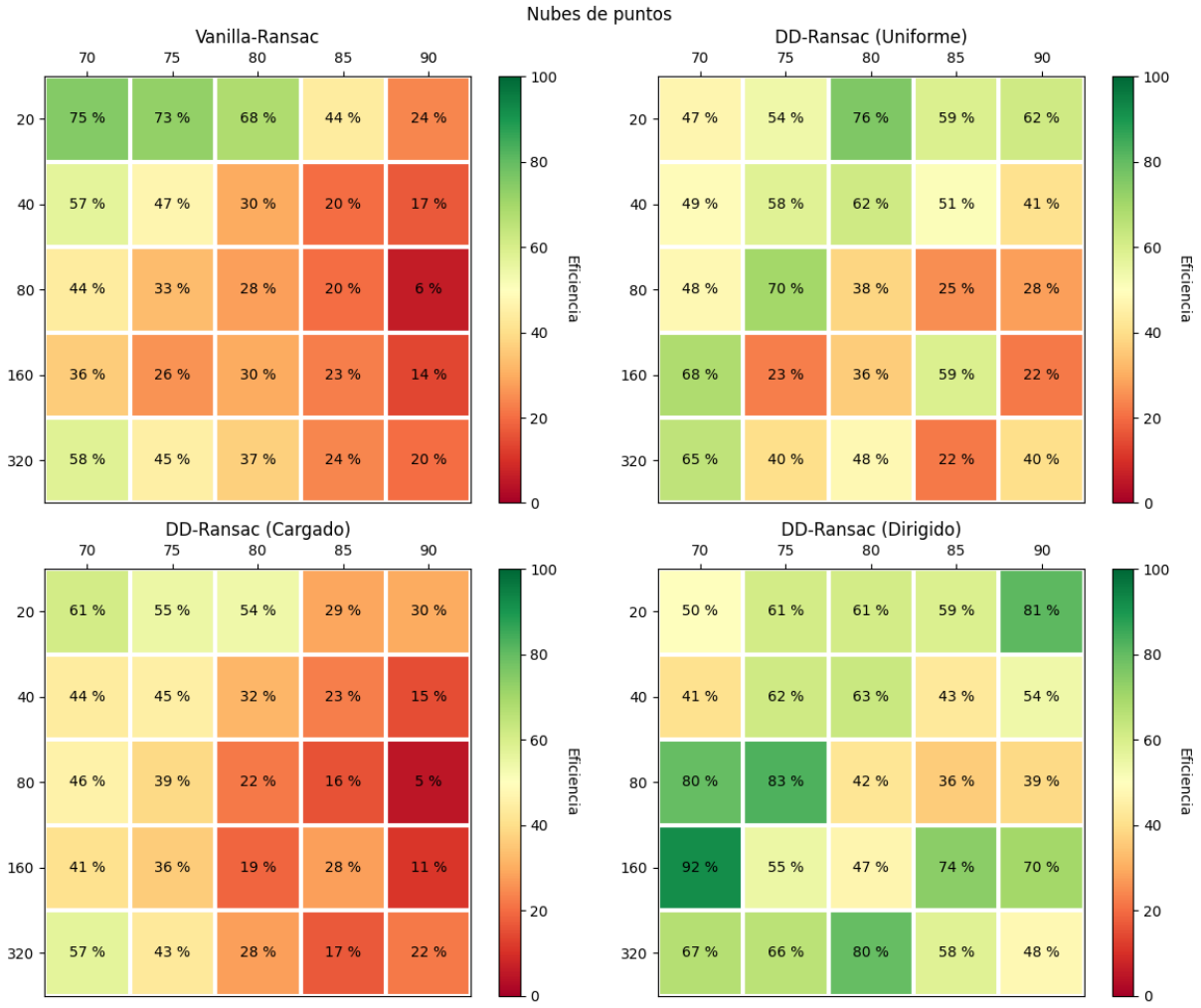


Figura 26. *Eficiencia* de los algoritmos propuestos, probados con el modelo del *emparejamiento de dos nubes de puntos* (basadas en letreros). Se generaron diferentes configuraciones de nubes de puntos con una transformación afín, inserciones y oclusiones. Los resultados están presentados como mapas de calor.

Capítulo 6. Conclusiones

6.1. Resumen

En esta tesis se trabajó el problema de la ’identificación de nubes de puntos’ en diferentes dimensiones. RANSAC es esencialmente el único algoritmo que existe para esta tarea, por lo tanto, se propusieron tres modificaciones diferentes a la sección de muestreo de dicho algoritmo. Los tres algoritmos propuestos en esta tesis surgieron a partir de siguiente hipótesis: *”los inliers tienen una mayor densidad relativa, en comparación con los outliers que se encuentran normalmente, más dispersos”*, dicho con otras palabras, los puntos que pertenecen al modelo tienden a estar más cercanos de otros puntos que también pertenecen al modelo, por lo tanto, se observa que los ’inliers’ tienden a encontrarse en zonas más densas, por el contrario, los puntos que no pertenecen al modelo, como el ruido, tienden a estar dispersos o en zonas de baja densidad local.

Para poner a prueba los *algoritmos propuestos*, se utilizaron los modelos del *emparejamiento de una línea recta* con un cierto rango de error δ, *emparejamiento de una circunferencia* con un cierto rango de error δ y *emparejamiento de dos nubes de puntos*, ante transformaciones afines, inserciones, oclusiones, más un cierto pequeño nivel de error δ. Con estos modelos de idearon dos experimentos que midieran el *trabajo* y la *eficiencia* de cada algoritmo, para compararlos dentro de las mismas condiciones y así poder identificar similitudes y cuantificar sus diferencias.

Se encontró que la relación que guardan los inliers con la densidad local de los datos es *verdadera*, por lo tanto, la obtención de una muestra libres de outliers aumenta su probabilidad cuando se da preferencia a los datos con mayor densidad local. Al evaluar los algoritmos propuestos bajo las mismas condiciones de la hipótesis inicial, el algoritmo que genera muestras en estricto orden de densidad (*’DD-RANSAC Dirigido’*), tiene la mejor evaluación en ambas métricas; *trabajo* y *eficiencia*.

6.2. Conclusiones

1. La heurística propuesta de procesar primero las zonas más densas resulta más efectiva en términos del trabajo realizado y la eficiencia del algoritmo, es decir, muestrear primero los puntos de mayor densidad representa una mejora al algoritmo RANSAC.

2. La hipótesis de trabajo es correcta en los casos donde se tiene coherencia espacial.

3. Un sesgo adecuado en la sección de muestreo del algoritmo RANSAC reduce la cantidad de muestras necesarias para encontrar una primera muestra libre de 'outliers'.

6.3. Trabajo futuro

- Modificar la evaluación o cálculo del soporte a una evaluación parcial, para mejorar el tiempo de ejecución.

- Explorar el efecto del tamaño k de la consulta kNN sobre la consulta RNN y sobre esta heurística.

- Explorar el desempeño de esta heurística ante el problema de las configuraciones quasi-degeneradas.

  Recordando que en la sección 2.2.4 se menciona que en el problema de los datos *quasi-degenerados*, se tiene un conjunto de datos degenerados (aquellos que no proporcionan suficientes restricciones para determinar una solución única), más una pequeña fracción de datos que proporcionan las restricciones restantes para determinar una solución única.

Literatura citada

Akram, M. U., Tariq, A., y Khan, S. A. (2011). Retinal recognition: Personal identification using blood vessels. En: *2011 International Conference for Internet Technology and Secured Transactions*. pp. 180–184.

Bellman, R. (1961). Curse of dimensionality. *Adaptive control processes: a guided tour. Princeton, NJ*, **3**(2).

Capel, D. P. (2005). An effective bail-out test for ransac consensus scoring. En: *Proceedings of the British Machine Conference*. BMVA Press, pp. 78.1–78.10.

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

Chicherova, N., Fundana, K., Müller, B., y Cattin, P. C. (2014). Histology to μct data matching using landmarks and a density biased ransac. En: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, pp. 243–250.

Choi, S. y Kim, J.-H. (2008). Robust regression to varying data distribution and its application to landmark-based localization. En: *2008 IEEE International Conference on Systems, Man and Cybernetics*. pp. 3465–3470.

Choi, S., Kim, T., y Yu, W. (1997). Performance evaluation of ransac family. *Journal of Computer Vision*, **24**(3): 271–300.

Chum, O. (2005). *Two-View Geometry Estimation by Random Sample and Consensus*. Tesis de doctorado, Department of Cybernetics. Czech Technical University in Prague.

Chum, O. y Matas, J. (2002). Randomized ransac with t d,d test. En: *IMAGE AND VISION COMPUTING*. pp. 448–457.

Chum, O. y Matas, J. (2005). Matching with prosac-progressive sample consensus. En: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. IEEE, Vol. 1, pp. 220–226.

Chum, O., Matas, J., y Obdrzalek, S. (2004). Enhancing ransac by generalized model optimization. En: *Proceedings of the ACCV*. Vol. 2, pp. 812–817.

Cover, T. y Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, **13**(1): 21–27.

Du, X. y Xie, T. (2012). Based on the half-space pruning to continuously monitoring reverse knn. En: *Proceedings of the 2012 International Conference on Computer Application and System Modeling*. Atlantis Press, pp. 1527–1531.

Feng, C. y Hung, Y. (2003). A robust method for estimating the fundamental matrix. En: *DICTA*. Citeseer, pp. 633–642.

Feng, J. y Jain, A. K. (2011). Fingerprint reconstruction: From minutiae to phase. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **33**(2): 209–223.

Fischler, M. A. y Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, **24**(6): 381–395.

Frahm, J.-M. y Pollefeys, M. (2006). Ransac for (quasi-)degenerate data (qdegsac. En: *In Proc.of CVPR*. pp. 453–460.

Geng, C. y Jiang, X. (2009). Sift features for face recognition. En: *2009 2nd IEEE International Conference on Computer Science and Information Technology*. pp. 598–602.

Hartley, R. y Zisserman, A. (2003). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, 2nd ed.

Jones, H. W. (2018). The recent large reduction in space launch cost. En: *International Conference on Environmental Systems (ICES)*, Jun. NASA Ames Research Center Moffett Field, CA, USA.

Kaspi, O., Yosipof, A., y Senderowitz, H. (2017). Random sample consensus (ransac) algorithm for material-informatics: application to photovoltaic solar cells. *Journal of Cheminformatics*, **9**: 34.

Konouchine, A., Gaganov, V., y Veznevets, V. (2005). Amlesac: A new maximum likelihood robust estimator. En: *Proc. Graphicon*. Citeseer, Vol. 5, pp. 93–100.

Korn, F. y Muthukrishnan, S. (2000). Influence sets based on reverse nearest neighbor queries. *ACM Sigmod Record*, **29**(2): 201–212.

Lee, J. j. y Kim, G. (2007). Robust estimation of camera homography using fuzzy ransac. En: O. Gervasi y M. L. Gavrilova (eds.), *Computational Science and Its Applications – ICCSA 2007*, Berlin, Heidelberg. Springer Berlin Heidelberg, pp. 992–1002.

Liebe, C. C., Gromov, K., y Meller, D. M. (2004). Toward a stellar gyroscope for spacecraft attitude determination. *Journal of Guidance, Control, and Dynamics*, **27**(1): 91–99.

Mahapatra, R. P. y Chakraborty, P. S. (2015). Comparative analysis of nearest neighbor query processing techniques. *Procedia Computer Science*, **57**: 1289–1298.

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

Matas, J. y Chum, O. (2005). Randomized ransac with sequential probability ratio test. En: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. IEEE, Vol. 2, pp. 1727–1732.

Myatt, D. R., Torr, P. H. S., Nasuto, S. J., Bishop, J. M., y Craddock, R. (2002). Napsac: high noise, high dimensional robust estimation. En: *In BMVC02*. pp. 458–467.

Nistér, D. (2003). Preemptive ransac for live structure and motion estimation. En: *ICCV*. IEEE Computer Society, pp. 199–206.

Rathgeb, C., Uhl, A., y Wild, P. (2012). *Iris biometrics: from segmentation to template security*, Vol. 59. Springer Science & Business Media.

Rodehorst, V. y Hellwich, O. (2006). Genetic algorithm sample consensus (gasac)-a parallel strategy for robust parameter estimation. En: *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*. IEEE, pp. 103–103.

Saad, F., Freer, C., Rinard, M., y Mansinghka, V. (2020). The fast loaded dice roller: A near-optimal exact sampler for discrete probability distributions. En: S. Chiappa y R. Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 26–28 Aug. PMLR, Vol. 108, pp. 1036–1046.

Subbarao, R. y Meer, P. (2006). Subspace estimation using projection based m-estimators over grassmann manifolds. En: *European Conference on Computer Vision*. Springer, pp. 301–312.

Tordoff, B. J. y Murray, D. W. (2005). Guided-mlesac: Faster image transform estimation by using matching priors. *IEEE transactions on pattern analysis and machine intelligence*, **27**(10): 1523–1535.

Torr, P. y Zisserman, A. (2000). Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, **78**: 138–156.

Torr, P. H. y Murray, D. W. (1997). The development and comparison of robust methods for estimating the fundamental matrix. *International journal of computer vision*, **24**(3): 271–300.

Torr, P. H. S. (2002). Bayesian model estimation and selection for epipolar geometry and generic manifold fitting. *International Journal of Computer Vision*, **50**(1): 35–61.

Wang, A. (2006). The shazam music recognition service. *Commun. ACM*, **49**: 44–48.

Wang, A. L. (2003). An industrial-strength audio search algorithm. *CiteSeerX*, pp. 7–13.

Wertz, J. R. (1978). *Spacecraft attitude determination and control*. Springer Science Business Media.

Wildes, R., Asmuth, J., Green, G., Hsu, S., Kolczynski, R., Matey, J., y McBride, S. (1994). A system for automated iris recognition. En: *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*. pp. 121–128.

Yam, C., Nixon, M. S., y Carter, J. N. (2004). Automated person recognition by walking and running via model-based approaches. *Pattern Recognition*, **37**(5): 1057–1072.

Yang, F., Tang, W., y Lan, H. (2017). A density-based recursive ransac algorithm for unmanned aerial vehicle multi-target tracking in dense clutter. En: *2017 13th IEEE International Conference on Control & Automation (ICCA)*. IEEE, pp. 23–27.

Yang, M. Y. y Förstner, W. (2010). Plane detection in point cloud data. En: *Proceedings of the 2nd int conf on machine control guidance, Bonn*. Vol. 1, pp. 95–104.

Zezula, P., Amato, G., Dohnal, V., y Batko, M. (2006). *Similarity search: the metric space approach*, Vol. 32. Springer Science & Business Media.

Apéndice

Función de distancia

De acuerdo con Zezula *et al.* (2006), una función de distancia de un *espacio métrico*. Típicamente, una función de distancia debe cumplir con las siguientes propiedades:

1. No ser negativa: $\forall x, y \in \mathcal{U},\ d(x, y) \geq 0$
2. Simétrica: $\forall x, y \in \mathcal{U},\ d(x, y) = d(y, x)$
3. Reflexiva: $\forall x \in \mathcal{U},\ d(x, x) = 0$
4. Positiva: $\forall x, y \in \mathcal{U},\ x \neq y \Rightarrow d(x, y) > 0$
5. Desigualdad del triangulo: $\forall x, y, z \in \mathcal{U},\ d(x, z) \leq d(x, y) + d(y, z)$

Además, existen algunas variantes de las funciones de distancia, tales casos son:

→ *pseudo-métrica*: cuando la propiedad 4 no se satisface.

→ *quasi-métrica*: cuando la propiedad 2 no se satisface.

En la siguiente tabla, obtenida de Mahapatra y Chakraborty (2015), se enlistan algunas de las funciones de distancia más utilizadas junto con la ecuación que calcula cada una.

Tabla 4. Distancias más conocidas.

Distancia	Espacio	Ecuación
Manhattan	Vectorial	$\sum \lvert x_i - y_i \rvert$
Euclidiana	Vectorial	$\sqrt{\sum \lvert x_i - y_i \rvert^2}$
Minkowski	Vectorial	$\sqrt[p]{\sum \lvert x_i - y_i \rvert^p}$
Chebychev	Vectorial	$\max \lvert x_i - y_i \rvert$
Canberra	Vectorial	$^n\sum_{i=1} \left(\lvert x_i - y_i \rvert \,/\, (\lvert x_i \rvert + \lvert y_i \rvert) \right)$
Hamming	Cadena	$\sum x_i \oplus y_i$
Edición	Cadena	$\lvert A \rvert + \lvert B \rvert - 2 \times \lvert LCS(A,B) \rvert$
Jaccard	Conjunto	$1 - \left(\lvert A \cap B \rvert / \lvert A \cup B \rvert \right)$

Espacio Métrico

Un espacio métrico es un par $(\mathcal{U}, d)$ donde $\mathcal{U}$ es el universo de objetos válidos, $D \subseteq U$ es un subconjunto finito de $\mathcal{U}$ (una base de datos) y $d : \mathcal{U} \times \mathcal{U} \longrightarrow R^+$ una función de distancia ó métrica.

Índice Alfabético