Desarrollo de Algoritmo de Mezclado de Mapas por Ocupación de Celdas Aplicado a la Navegación y Exploración Colaborativa de Entornos Internos Desconocidos

Carlos Alberto Velásquez Hernández

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería Eléctrica y Electrónica
Bogotá D.C., Colombia
2017

Desarrollo de Algoritmo de Mezclado de Mapas por Ocupación de Celdas Aplicado a la Navegación y Exploración Colaborativa de Entornos Internos Desconocidos

Carlos Alberto Velásquez Hernández

Tesis de grado presentada como requisito para optar al título de:
Magíster en Ingeniería - Automatización Industrial

Director:
Ph.D. Ms.C.Flavio Augusto Prieto Ortiz

Línea de Investigación:
Robótica
Grupo de Investigación:
GAUNAL

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería Eléctrica y Electrónica
Bogotá D.C., Colombia
2017

”Nuestros sentidos nos permiten percibir solo una pequeña porción del mundo exterior”.

Nikola Tesla

Agradecimientos

A mi madre, Cenobia Susana Hernández Vásquez, a quien dedico este trabajo. Sin su apoyo y amor incondicional no habría sido posible alcanzar esta meta. A mi familia, fuente de inspiración para su realización.

A la Universidad Nacional de Colombia, mi alma mater, por haberme abierto las puertas del conocimiento científico y académico. A los Grupos de investigación GAUNAL y DIMA-UN por brindarme la oportunidad de conocer y profundizar mis conocimientos en el campo de la robótica autónoma móvil. A mi director, Flavio Augusto Prieto Ortiz, por su valiosa guía y dedicación durante el trascurso de mi trabajo de tesis.

Resumen

Durante la última década, el problema de SLAM ha sido ampliamente estudiado y divulgado. Han surgido técnicas que proponen una buena solución y nuevos desafíos que buscan ampliar el horizonte de aplicaciones y aproximaciones al problema de SLAM. Un reciente desafío propone la extensión del SLAM, aplicado a un único agente, al Multi-SLAM, donde 2 o más robots interactúan con el fin de cumplir una única tarea: la obtención de un mapa global único y consistente con el entorno explorado. Dicha obtención del mapa global, a partir de reconstrucciones parciales reportadas por los robots, es el problema de mayor relevancia en el campo del Multi-SLAM. Este trabajo investigativo propone una solución al problema del Multi-SLAM, abordandolo con las herramientas que brinda la Visión por Computador. En detalle, se expondrán conceptos básicos de SLAM y Multi-SLAM, el diseño y desarrollo de un algoritmo de mezcla de mapas por ocupación de celda y su validación, y la presentación de un detector de esquinas nuevo, el cual fue desarrollado en este trabajo. Así mismo, se mostrará el diseño de un algoritmo de toma de decisión para el control de robots móviles en el entorno y su integración con el algoritmo que mezcla mapas. Los resultados obtenidos demuestran que los algoritmos desarrollados en este trabajo son una aproximación consistente y aplicable al campo del Multi-SLAM para operaciones en tiempo real, lo que supone un notable avance en este campo.

Palabras clave: Robótica móvil, SLAM, Visión por Computador, Mezcla de mapas, Decision-making, RANSAC.

Contenido

Lista de Figuras

Lista de Tablas

1. Introducción

El problema de localización y posicionamiento autónomo de robots móviles en ambientes cambiantes es un tema de amplia investigación en el campo de la robótica actual. Estas investigaciones están siendo impulsadas por las potenciales aplicaciones que puede tener un vehículo robotizado con la capacidad de navegar autónomamente en entornos desconocidos. Transporte y almacenaje en la industria, transporte de pacientes o medicamentos en medicina, recolección y verificación de cosechas, transporte y almacenaje de equipajes en aeropuertos, búsqueda en áreas de desastre o de difícil acceso [52, 39, 47], exploración terrestre [24] y acuática [2], son sólo algunas de las aplicaciones en las cuales tiene campo este tipo de dispositivos.

Dependiendo del tipo y sector de aplicación, el uso de un solo vehículo robotizado no es suficiente y surge la necesidad de integrar múltiples vehículos que puedan realizar una misma tarea o trabajos diferentes de forma conjunta y coordinada, conformando así lo que se conoce como red colaborativa. El desarrollo de esta clase de redes propone nuevos y más exigentes retos, ya que no sólo se debe resolver el problema de navegación y posicionamiento para un solo robot, sino concebir un sistema que permita controlar un grupo de agentes móviles (robots) en un mismo entorno desconocido. En la actualidad se han desarrollado técnicas aplicables a la navegación y posicionamiento de plataformas robóticas móviles en entornos dinámicos, estáticos, interiores [24] y exteriores [20]. Estas técnicas se basan en la Localización y Mapeo Simultáneo o SLAM (Simultaneous Localization and Mapping). La mayoría de métodos de SLAM se centran en el problema de mapeo y localización para un único robot. Sin embargo, técnicas aplicables a grupos de robots están empezando a ser formuladas [42, 32].

La aparición de técnicas SLAM enfocadas en grupos de robots móviles dio surgimiento al campo llamado Multi Robot SLAM (MR-SLAM), cuyo foco central es la obtención de un método consistente que permita a un número cualquiera de agentes reconstruir un mismo entorno. Puntualmente, las técnicas MR-SLAM poseen tres grandes problemas [54, 28, 17, 40] a resolver con el fin de hacerlas aplicables computacionalmente a tareas de colaborativas en tiempo real. Dentro de ellos, el problema de mezclado de mapas es el más estudiado y abordado en la literatura [54, 47, 40, 53, 43, 51, 6]. De acuerdo a lo anterior, es evidente que el ánimo de las técnicas MR-SLAM es reducir los tiempos de mapeo de un ambiente sobre el cual se quiera realizar una operación. Una aplicación específica que requiere técnicas de

este estilo es la atención y rescate en zonas de desastres. En esta aplicación, el tiempo es la variable que juega en contra de quienes tienen la misión de localizar, en el menor tiempo posible, al mayor número de víctimas o heridos.

Bajo esta idea de crear redes autónomas de robot móviles, se han propuesto diferentes técnicas de localización para grupos de robots móviles. Como técnicas MR-SLAM pueden mencionarse 3 enfoques. El primer enfoque plantea el algoritmo MR-SLAM con condiciones de inicio homogéneas para todos los agentes de la red (posición de inicio y estrategia de navegación común). El segundo enfoque plantea la eliminación de condiciones de inicio para todos los agentes y lo reemplaza por encuentros reales de los robots móviles en el entorno. Una vez dos o más robots se encuentran en el entorno, determinan la posición relativa entre ellos con el fin de unir los mapas locales. El último enfoque, de mayor reto en MR-SLAM, supone la idea de no establecer ninguna condición de inicio de los agentes.

Este último enfoque basa su idea en los solapamientos entre dos o más mapas locales e intenta predecir la posibilidad de un encuentro entre robots. Si logra predecir de manera acertada posibles encuentros entre robots, alerta al sistema de monitoreo de lo sucedido con el fin de desviar o modificar la estrategia de navegación y así ahorrar tiempo de mapeo. Para ello, este enfoque hace uso de técnicas comúnmente utilizadas en el campo de la Visión por Computador. A través del análisis de mapas locales, que son tratados como imágenes, la técnica busca encontrar zonas o características comunes entre los mapas con el fin de encontrar una solución que permita unir o mezclar los mapas analizados. Este enfoque hace uso de técnicas de detección y descripción de características, así como de algoritmo de apareamiento de datos. Sin embargo, este enfoque supone una alta carga de procesamiento, ya que las técnicas aportadas por el campo de la Visión por Computador demandan gran cantidad de tiempo para el análisis de los datos de entrada al algoritmo. Por ende, implementaciones en tiempo real de estas técnicas son escasas debido a las grandes limitaciones que supone analizar un conjunto de n mapas aportados por n agentes o robots. A mayor número de agentes en el entorno, mayor es la carga computacional y por ende, menor es la probabilidad de obtener un algoritmo capaz de brindar una solución eficaz y en tiempo real.

El reporte de investigación del trabajo desarrollado propone una nueva técnica MR-SLAM basada en la mezcla de mapas. En esencia, la técnica MR-SLAM desarrollada se compone de 3 etapas que permiten llegar a una solución consistente y viable para aplicaciones en tiempo real. La primera etapa se encarga del análisis de mapas para encontrar zonas comunes entre dos o más mapas. La segunda se encarga del cálculo matemático que permita unir dos mapas locales una vez el algoritmo de detección de solapamientos lanza una posible solución. Finalmente, la última y tercera etapa se encarga de controlar la exploración de los agentes en el entorno con el fin de evitar encuentros físicos entre los mismos.

Los resultados obtenidos en este trabajo muestran que la técnica propuesta es adecuada para tareas de reconstrucción colaborativa de entornos desconocidos internos en tiempo real. Como primer resultado, la técnica presenta un desempeño eficiente en el tiempo de análisis y mezcla de mapas: menos de 280 milisegundos por pareja de mapas analizadas, es decir, 4 parejas de mapas analizados en aproximadamente 1 segundo. Por otro lado, la técnica MR-SLAM desarrollada también mostró una reducción del tiempo de exploración de un entorno y del consumo de energía cuando se usan 2 o más agentes: la reducción es cercana al 33 % del tiempo que le toma a un solo agente reconstruir el mismo entorno, aún cuando el algoritmo de toma de decisiones detiene o pausa la operación de uno o más agentes cuando lo considera necesario. Por lo cual, esta técnica presenta como ventaja no solo la reducción e el tiempo de exploración de un entorno, sino también la reducción del consumo energético de toda la red de agentes desplegada. Como resultado adicional, se presenta una nueva técnica de detección de esquinas, la cual fue desarrollada para mejorar la eficiencia de cómputo del algoritmo de mezcla en su etapa de detección de puntos característicos: el detector de esquinas desarrollado presentó el mejor tiempo de cómputo de todos los detectores estudiados para este trabajo, es decir, para una imagen de 512×512 píxeles, el detector demora cerca de 4.8 milisegundos, 3 veces menos que técnicas ampliamente conocidas como el detector de Shi Tomasi o de SIFT. Por lo anterior, es claro que la técnica MR-SLAM desarrollada en este trabajo presenta un avance en el del Multi Robot SLAM (MR-SLAM), ya que es la primera vez que se reporta el uso de una técnica en operaciones de reconstrucción en tiempo real con las características mencionadas anteriormente.

Finalmente, este documento expone cada una de las etapas involucradas en el proceso de diseño, desarrollo y prueba real de la técnica MR-SLAM. Para ello, en el Capítuo 1 se detalla el marco referencial en el cual se enmarca este trabajo (objetivos, problemática y alcance de la técnica desarrollada). En el Capítulo 2 se expone el marco teórico o conocimientos previos al diseño y desarrollo del algoritmo. En esta sección se aclaran y extienden conceptos relacionados a SLAM, MR-SLAM y filtros de datos comúnmente usados. Seguido, en el Capítulo 3 se presenta el algoritmo de detección de solapamientos o cruce entre mapas. En el Capítulo 4 se detallan los algoritmos de toma de decisiones y de cálculo de la trama para unir los mapas. Finalmente, el Capítulo 5 muestra las pruebas y simulaciones desarrolladas para la validación de la técnica, las consideraciones y condiciones de operación usadas en cada prueba, los escenarios seleccionados y los resultados más destacados de la validación de la técnica MR-SLAM desarrollada.

2. Marco teórico

Previo al diseño, desarrollo e implementación de los algoritmos usados, se hace necesario realizar una breve fundamentación teórica con el fin de aclarar conceptos usados comúnmente a lo largo de este documento.

2.1. Odometría

La odometría es el estudio de la estimación de la posición y orientación (pose) de un robot a través del tiempo, haciendo uso de las mediciones tomadas de la sensórica interna o externa del robot.
Comúnmente, esta estimación se realiza con sensores encoder, IMU, acelerómetros, magnetómetros, brújula digital, entre otros. Principalmente, la odometría fusiona la información entregada por cada uno de los sensores disponibles en el robot en aras de estimar, mediante su modelo cinemático y dinámico, el desplazamiento a lo largo del tiempo. Para el caso de robots móviles, la pose del robot está determinada por 3 variables: X, Y y θ; mientras que para robots acuáticos o aéreos su descripción se hace en 6 parámetros: X, Y, Z y θ_x, θ_y, θ_z.

2.2. Teorema fundamental de Bayes

El teorema de Bayes [35] es esencial en el desarrollo de algoritmos de mapeo, ya que es capaz de asociar la ocurrencia de dos eventos relacionados. El teorema de Bayes estipula que la probabilidad de que un suceso A suceda, conociendo la probabilidad condicional de dicho suceso con respecto a otro suceso B, siempre que se conozca la distribución de probabilidad marginal de A, Ecuación 2-1.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2-1}$$

Cuando se trata de aplicar este teorema a la resolución exitosa de SLAM, se puede considerar el suceso A como uno de los diferentes estados en los que se encontrará el sistema, basado en la información del suceso B, que para el teorema vienen siendo las medidas obtenidas por un sensor. La regla dice que se puede resolver este cálculo de manera sencilla multiplicando dos

términos: la probabilidad (en nuestro modelo) de obtener la medida B dado el estado A, y el grado de confianza que se le da al hecho de que A sea precisamente el estado del sistema antes de recibir los datos.

2.3. Filtros de Kalman

Como se expresa en [35], el filtro de Kalman, o estimación lineal cuadrática, es un algoritmo que usa una serie de medidas observadas a lo largo del tiempo, que contienen un cierto nivel de ruido (como las medidas de un sensor o la odometría del robot) y otras inexactitudes, para producir estimaciones de variables desconocidas, que tienden a ser más precisas de lo que serían si estuviesen basadas en una simple medición. Formalmente, un filtro de Kalman opera recursivamente en flujos de datos con ruido para producir una estimación estadísticamente óptima del estado del sistema.

El algoritmo funciona como un proceso en dos fases: en la fase de predicción, el filtro de Kalman produce estimaciones del estado actual de las variables, junto a algunas incertidumbres. Una vez capturada la salida del siguiente estado, el filtro actualiza sus estimaciones utilizando una media ponderada, con mayor peso a las predicciones de las cuales se tiene un mayor nivel de certeza. Este filtro asume que el sistema que intenta predecir se comporta de forma lineal y que todas las medidas de error tienen una distribución Gaussiana.

Existen diversas variaciones de este filtro, entre ellas el filtro extendido de Kalman (EKF), usado comúnmente en las técnicas de SLAM. Sin embargo, estas variaciones presentan complicaciones con respecto al filtro original, es decir, el filtro EKF al no ser un estimador estadístico óptimo, es sensible a datos de inicio erróneos, ya que las estimaciones iniciales del estado del sistema divergirán del comportamiento o estado real del sistema modelado. Esta situación es comprensible toda vez que este filtro realiza una linealización del modelo no lineal que pretender predecir a lo largo del tiempo.

2.4. Localización y mapeo simultáneo (SLAM)

SLAM (Simultaneous Localization and Mapping) es una técnica usada por robots y vehículos autónomos para construir una representación consistente de un entorno desconocido y, mediante el uso de la odometría del robot, lograr estimar su posición en cada instante de su recorrido. Si bien SLAM se aplica generalmente en vehículos robotizados, también se ha implementado en robots humanoides [38] y en grupos colaborativos de robots [32].

Los vehículos autónomos y robots perciben el entorno en que se encuentran a través de sensórica especializada que le permite captar información proveniente de su exterior. En primera medida, se encuentran los sensores que permiten realizar una primera estimación de la pose del robot (odometría), tales sensores son encoders, acelerómetros, giroscopio, IMU, entre otros. Éstos, a través del modelo cinemático y dinámico del robot permiten hacer una estimación gruesa de cuánto se ha movido el robot. Sin embargo, el error asociado (ruido) a estas medidas no permite confiar del todo en la precisión y exactitud de las estimaciones realizadas. Dicho ruido puede deberse, por ejemplo, a deslizamientos del robot que son contabilizados por el encoder como movimientos efectivos, señales débiles del campo magnético de la tierra o ruidos electromagnéticos en la lectura y estimación de la orientación y aceleración del robot. Es por esto que los robots requieren de una sensórica más especializada, que les permita captar de su entorno más información para corregir la primera estimación realizada. De ahí, surge la necesidad del uso de sensores como Laser de Rango, Sonares, Cámaras Estéreo o dispositivo RGB-D. Estos sensores toman muestras del entorno para incluirlas en el algoritmo de estimación de la pose y, a su vez, construir un mapa consistente a partir de dichas muestras. Las muestras o características representativas son conocidas como landmarks o puntos de referencias.

Los landmarks ayudan a ubicar al robot en todo instante y, como característica fundamental, deben ser distintivos dentro del entorno de tal manera que el robot en su recorrido pueda re-observarlos. De esta manera, el robot puede corregir la información de su pose y actualizar el mapa percibido. Otra característica fundamental de los landmarks es ser referencias estáticas, es decir, no pueden cambiar su posición relativa respecto al robot, ya que es este último quien cambia su pose y se vale de las referencias para saber cuánto se ha movido. Una analogía posible es el sistema de posicionamiento de un ser humano, en el cual nunca se selecciona como referencia un carro o un avión en movimiento, sino puntos fijos como edificaciones. Adicionalmente, estos puntos de referencia deben ser invariantes a orientación y traslación.

Fundamentalmente, SLAM sigue una secuencia de pasos para garantizar un mapeo y estimado de la pose (dupla de posición y orientación), lo más preciso posible. La Figura **2-1** muestra la secuencia de pasos más representativos del algoritmo general de SLAM. Primero, el algoritmo calcula su posición mediante odometría. Esa primera estimación de la pose puede ser considerada como una aproximación cruda o gruesa, debido a la gran incertidumbre asociada a las medidas entregadas por los sensores del robot. Sin embargo, el algoritmo procede a realizar la toma de una muestra del entorno para refinar su pose y así comenzar con la reconstrucción del entorno, Figura **2-1**.a . Como segundo paso, se predice la posición siguiente del robot mediante la introducción de un paso de control (de aquí la importancia de conocer el modelo cinemático y dinámico del robot), Figura **2-1**.b . Con este movimiento, se calcula por odometría la nueva pose del robot y finalmente, el robot debe volver a observar

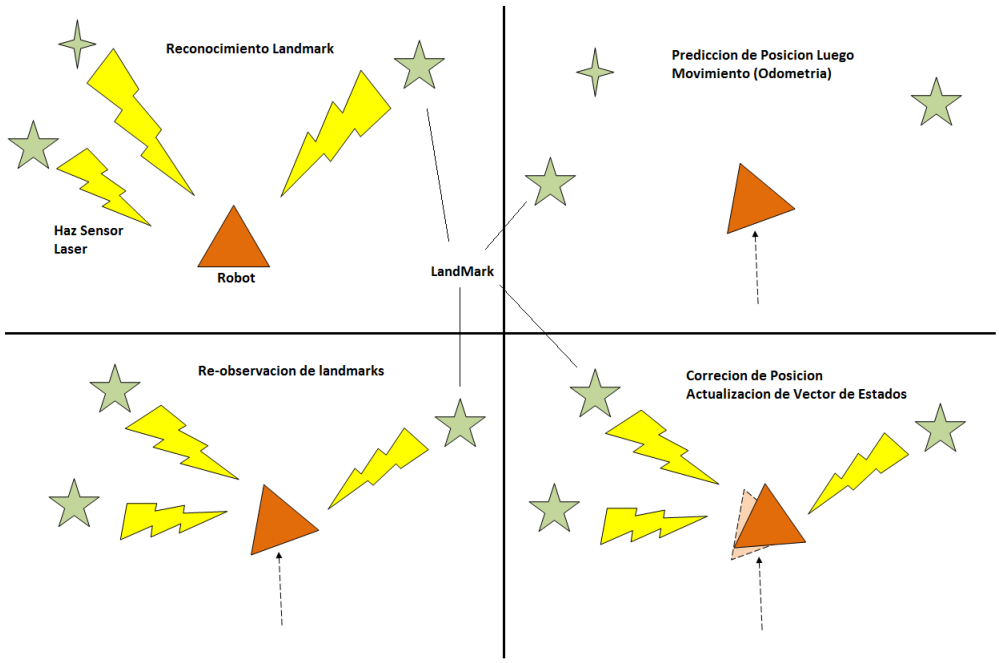

Figura 2-1.: Secuencia de pasos del algoritmo de SLAM para la estimación simultanea de la pose y mapa de un robot.

la muestra o landmark que vio en el primer paso para refinar aún más la posición actual del robot y actualizar el mapa que lleva reconstruido a este paso, Figura **2-1**.c-d . Es evidente que, el paso de re-observar las landmarks es fundamental para el algoritmo de SLAM, ya que de esta manera se estima cuánto se movió respecto a la landmark reconocida y cuál fue la incertidumbre asociada a su pose durante la iteración de estimación de su estado. Ahora bien, si el robot detecta una nueva landmark en su muestreo y esta cumple con las condiciones para ser considera como referencia, es incluida dentro del mapa creado por el robot y, adicionalmente, es usada para refinar aún más la estimación de su posición.

Todos los cálculos realizados por SLAM, se basan en métodos probabilísticos, es decir, calculan la probabilidad de la presencia de un objeto o no, luego de observarlo una y otra vez para usarlo como landmark. Además, hacen uso de estos métodos para estimar si su posición real y actual ha sido consistente con las medidas y cálculos realizados en iteraciones anteriores.

SLAM, como teoría y técnica de localización, es considerada como una técnica solucionada, es decir, como teoría se encuentra claramente definida. Sin embargo, los problemas actuales con esta técnica se basan en el procesamiento de la información captada por el robot, la validación (debido al ruido) de la misma, la capacidad y tiempo de computación usado para procesar la información que recibe el robot. Actualmente, pese a que se reconoce la impor-

tancia y aplicación de esta técnica en el campo de la robótica, las aplicaciones reales en robot son limitadas, ya que son pocas las aproximaciones de SLAM realmente optimizadas para aplicaciones en tiempo real. En el anexo *Técnicas de SLAM* se encuentra una revisión de las técnicas más destacadas de SLAM.

2.5. Multi Robot SLAM

Multi Robot SLAM (MR-SLAM) es la técnica extendida de SLAM y aplicada a una red o grupo de robots que se comunican entre sí de forma inalámbrica, principalmente, con el fin de darle autonomía a cada agente en su tarea de reconstruir el entorno. Esta técnica surge como necesidad de resolver ciertas deficiencias de SLAM, debido a las limitantes que posee el hacer uso de un solo robot en un entorno de grandes proporciones. Por un lado, se tiene el tiempo que emplea un robot para reconocer un entorno, es decir que el tiempo empleado por el robot es proporcional al tamaño del ambiente a mapear. Sin embargo, contar una red de n robots puede reducir en una proporción considerable el tiempo de reconstrucción del entorno requerido. De lo anterior, también se puede concluir que esta técnica permite abarcar grandes áreas sin un aumento proporcional directo del tiempo, debido a que cada robot puede cubrir un máximo de área estipulado, por lo cual el mapa global reconstruido crece de acuerdo al número de agentes usados en la red colaborativa.

Adicionalmente, tener un grupo de robots introduce redundancia a la red colaborativa [9] y la hace más tolerante a errores o fallos en la navegación de los agentes, debido a que refina las medidas percibidas por otro, ya que como se sabe en robótica móvil y SLAM, cada robot posee un error de estimación asociado a sí mismo y este error puede propagarse a lo largo de todo el mapa. Por esto, introducirle redundancia o re mapeo al mapa construido por un robot, mejora la precisión del mismo y, por ende, mejora el desempeño de todos los agentes al momento de realizar tareas colaborativas.

Adicionalmente, esta técnica provee la gran ventaja de reducir costos en la sensórica implementada en cada robot, debido a la introducción de redundancia en el mapeo del entorno que, a diferencia del single SLAM, usa sensórica especializada en un solo robot, ya que solo hay una posibilidad de mapear eficientemente el entorno. En MR-SLAM, la posibilidad de realizar tareas conjuntas, re-mapear zonas de otros robots para refinar el mapa global generado, son ventajas de esta técnica ampliamente estudiada en los últimos años.

2.5.1. Técnicas MR-SLAM

En MR-SLAM existen diversas formas de abordar el problema de cómo realizar un mapeo conjunto por parte de una red de robots móviles capaces de comunicarse entre sí. Diversas técnicas conocidas se pueden dividir en métodos centralizados y descentralizados. Evidentemente, las técnicas centralizadas tiene la desventaja de que deben ser capaces de manejar la información proveniente de cada agente en una sola terminal. Sin embargo, las técnicas descentralizadas representan un alto costo económico, debido a que los algoritmos y la capacidad computacional usada en cada agente es mayor y más compleja.

Dentro de las técnicas o estrategias MR-SLAM mayormente desarrolladas en la literatura se encuentran:

- Técnica MR SLAM usando posiciones iniciales conocidas: es la técnica más simple de abordar y, quizás, de resolver el problema de navegación y reconstrucción de un entorno por parte de un grupo de agentes. Considera esencial que cada robot parta de un punto inicial conocido por todos los robots y por ende, de acuerdo a un marco de referencia global, cada robot puede determinar dónde se encuentran los demás agentes. De esta manera, se puede hacer uso de esas estimaciones (transformaciones) para ir construyendo de manera centralizada el mapa global del entorno. Esta técnica al estar supeditada a la condición del previo conocimiento de las posiciones de cada robot, es sensible a una mala inicialización de la posición de un robot respecto al punto de origen en común. Por lo tanto, esto puede provocar un mal mezclado del mapa parcial reconstruido por cada robot.
- Técnica MR SLAM usando Rendezvous (Encuentros): esta técnica descentralizada consiste en realizar encuentros entre dos o más robots de la red. Una vez se genera un encuentro, es posible realizar el mezclado de mapas o iniciar en conjunto la técnica de mapeado y reconstrucción del entorno (SLAM). En [26] se puede ver una aplicación de la primera variante, donde cada robot cuenta con una cámara que detecta visualmente a otro robot. Cuando el algoritmo de visión logra determinar con precisión la presencia de un robot, realiza una estimación de la posición relativa entre ambos respecto al punto de encuentro dónde hubo la detección. Una vez realizada la estimación pueden intercambiar sus mapas. Debido a que cada robot sabe dónde está respecto a su mapa y al hacer el cálculo de la posición de un robot respecto a otro, éste puede aplicar una transformación para realizar un mezclado de los mapas de los agentes (robots) encontrados. Luego del encuentro, los robots pueden corregir sus caminos para no seguir encontrándose y dirigirse hacia zonas no mapeadas.

  La otra variante de esta técnica es desarrollada en [21], donde los autores exponen una manera de emular el single SLAM a través de una red de robots. La técnica considera un inicio de cada agente en el cual cada uno construye su mapa y almacena su propio vector de estados. Una vez se realiza un encuentro entre dos o más robots, inicia un

algoritmo de SLAM que ve a los agentes encontrados como uno solo y empieza a realizar el mezclado de los mapas previo y posterior al encuentro de los agentes.

- Técnica MR-SLAM para mezclado de mapas por solapamientos: está técnica es, quizás, la más compleja de todas. Busca realizar un mezclado de mapas evitando la menor área común entre los mapas de cualquier par de agentes de la red. Es decir, el algoritmo de mezclado verifica, a través de la aplicación de transformaciones a los mapas, cuales se superponen y que tan consistentes son en un posible mezclado. Una vez comprobada la validez de la transformación aplicada para el mezclado de mapas, se procede a la generación y construcción de un mapa global a partir de esas verificaciones. Esta técnica tiene como ventaja principal que no se generan encuentros reales entre robots ni re-mapeos en diferentes zonas de los mapas parciales generados, debido a que el algoritmo debe predecir cuando ocurre una superposición (solapamiento) de mapas. Aunque este enfoque es el más eficiente, requiere de un tipo de mapa gráfico para su buen funcionamiento. Para el desarrollo de este trabajo investigativo, este enfoque fue el seleccionado, debido a que resulta ser el más eficiente en el tiempo de reconstrucción si se evitan encuentros entre los robots o re-mapeos innecesarios de zonas del entorno.

2.5.2. Problemas MR-SLAM

Existen tres problemas o deficiencias conocidas de MR-SLAM que en la actualidad son estudiadas para encontrar una solución adecuada [54]:

1. Localización y Mapeo Incremental: se refiere al problema de análisis y procesamiento, en tiempo real, de toda la información adquirida por cada robot y que, al usar una técnica centralizada, tiende a aumentar el costo computacional de los algoritmos implementados. Quizás este inconveniente pueda ponerle un límite máximo de agentes a la red, debido a que puede desbordarse la capacidad operativa de la plataforma que controla los robots.
2. Loop Closure: esta deficiencia proviene del SLAM y se refiere a la terminación del ciclo de mapeo. Busca resolver cuando un robot concluye su labor de mapear porque ya reconstruyó el entorno posible a su alcance. ¿Cómo sabe el robot que mapeó todo el entorno disponible?, ¿cómo se garantiza que en realidad se haya mapeado todo el entorno?. Estos son algunos de los cuestionamientos involucrados en el análisis y solución del problema *loop closure*.
3. Mezclado de Mapas: se refiere al problema de cómo generar un mapa global a partir de mapas parciales lo más rápido y confiable posible, debido a que no es trivial resolver el problema de unir dos o más mapas. Por otro lado, el algoritmo debe encargarse de generar una nueva trayectoria para los robots, para que no prosigan con encuentros repetitivos. Además, el mapa global generado debe ser lo suficientemente consistente como para que un robot pueda navegar por todo el espacio mapeado y posicionarse de manera precisa aun cuando navegue por zonas que no fueron reconstruidas por él.

2.6. Mapas para navegación y exploración de entornos

Los mapas o representaciones del entorno son importantes en la robótica móvil, debido a su obligatorio uso para tareas de navegación, actualización y exploración del entorno mismo.

En la navegación de robots móviles, puede ser muy útil el uso de planos arquitectónicos de los espacios donde se supone navegará y actuará el robot. Sin embargo, estos planos tienen la gran limitante de no representar fielmente cómo es la edificación (entorno) por dentro, ya que en estos planos no se incluye el mobiliario y demás objetos presentes en un entorno real [44]. Es decir, los planos arquitectónicos no son mapas consistentes con la realidad del entorno y esto puede afectar enormemente la operación e integridad del robot. Además, muchas veces no se cuenta con un plano de ayuda para el robot, puesto que el entorno es totalmente desconocido. Es por esto que, en el campo de la navegación de la robótica móvil se abre un área de estudio enfocada netamente hacia la adquisición, asociación y reconstrucción de entornos. Reconstruir un mapa por parte del mismo robot, le da la capacidad de conocer por sí mismo el entorno en donde navega. Además, aprender a percibir el entorno entrena al robot hacia la toma de mejores decisiones a la hora de navegar, ya que lo vuelve más robusto frente a cambios repentinos dentro del ambiente, dado que el robot tiene la facultad de actualizar el mapa que ha reconstruido.

Los algoritmos de adquisición de mapas han sido propuestos desde hace décadas. Sin embargo, su aplicación e implementación en ambientes reales aun presenta limitantes que siguen siendo estudiadas. Los desafíos fundamentales para la reconstrucción o estimación del mapa de un entorno, según [44], son:

- La hipótesis del espacio: dada la inmensidad del espacio o entorno en el que se trabaja, las posibilidades de obtener un mapa son inmensas. Dado que los mapas son definidos sobre espacios continuos, los mapas tienen infinitas dimensiones por lo que el tratamiento y estimación del estado del mismo es compleja: los mapas pueden ser descritos por 10^5 o más variables [44].
- Aprendizaje de Mapas: para la adquisición de mapas se debe haber resuelto el problema de estimación de la pose del robot dentro del entorno. En la actualidad, este desafío ha presentado el mayor avance, dado que en el campo de la navegación de robots móviles a este problema se le ha resuelto con técnicas de SLAM.

Estos dos grandes desafíos, en los cuales se enmarca el problema de estimación de mapas (mapeo), es el resultado de varios factores [44]:

- Tamaño: mientras más pequeño sea el rango perceptual del robot respecto a su entorno relativo, más difícil es adquirir el mapa.
- Ruido en la percepción y actuación: en la práctica y en la vida real, todo sensor o actuador está expuesto a tener ruido en sus medidas. Los modelos cinemáticos y

dinámicos y los filtros de los sensores, buscan precisamente eliminar o reducir al más bajo nivel el ruido en la información que entra y sale del sistema, sin embargo, es inevitable su presencia. Por lo tanto, los desafíos en esta materia se han concentrado en desarrollar mejores sensores y actuadores que permitan llevar a la práctica lo que se plantea y supone en la teoría.

- Ambigüedad en la percepción: se refiere al hecho de que en un entorno dos o más espacios pueden ser similares y, por ende, pueden confundir al robot al momento de hacer sus cálculos. Por eso se busca que el entorno sea lo más distintivo posible. Como analogía, un ser humano dentro de un cuarto con 4 esquinas iguales no es capaz de posicionarse ni relacionar la información percibida a través de sus sentidos, a menos de que en el espacio él sea capaz de determinar diferencias únicas e irrepetibles en cada esquina. Este mismo problema lo sufren con mayor frecuencia los robots, más aún cuando cuentan con sensórica que introduce ruido a la información adquirida.

Dentro de las aproximaciones más notables y destacables se encuentran los mapas por ocupación de celda, que, a groso modo, estiman la probabilidad de ocupación de una celda sobre el mapa a estimar. Esta clase de mapa tiene una representación gráfica, visible e interpretable por un ser humano, Figura **2-2**.

También como técnicas de elaboración de mapas, se encuentran los mapas por extracción de características. Esta clase de mapas representa un entorno a través de un vector de estados, el cual contiene referencias (landmarks) y poses asociados a cada estado. Esto, tal como se puede notar, no tiene una representación gráfica, aunque garantiza la extracción de características únicas del entorno, asociadas a poses también consideradas únicas dentro del mapa. Sin embargo, el tiempo de generación de trayectorias con esta representación se hace más compleja y extensa, dado que los tiempos de búsqueda de una trayectoria segura se incrementan si el vector de estados estimados crece.

En este trabajo es de interés la reconstrucción de mapas por ocupación de celdas, debido a que es la técnica computacionalmente óptima y la más trabajada en el ámbito de mezclado de mapas en redes colaborativas de robot móviles. Además, dada su buena representación (Figura **2-2**), el trazado o generación de trayectorias es más sencillo, lo mismo que la evasión de obstáculos, dado que el mapa representa la probabilidad de existencia de obstáculos en cada una de las celdas del mapa.

2.6.1. Mapas por ocupación de celda

Los mapas por ocupación de celdas (o Occupancy Grid Map, OGM), son modelos determinísticos de representación de un ambiente, mediante la estimación de la probabilidad de que una celda tenga un estado de ocupación o no. Estas técnicas de representación de entor-

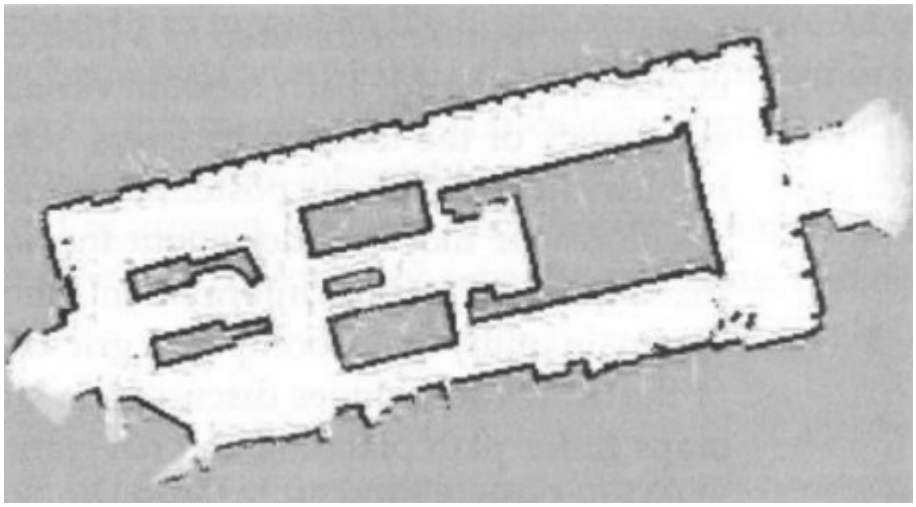

Figura 2-2.: Representación gráfica de un mapa por ocupación de celdas. Tomado de [44].

nos se popularizaron a partir de su primera publicación a cargo de Hans Moravec y Alberto Elfes [33] en los años 80's, cuando ambos plantearon la primera técnica, basados en modelos de sensores de percepción. En [33] se plantea una primera estimación del estado de un mapa sonar conociendo la posición desde la cual han sido tomadas cada una de estas medidas. Alberto Elfes [14] plantea que la representación de ambientes por ocupación de celdas emplean un mosaico multidimensional (sea 2D o 3D) del espacio en las celdas, donde cada una de ellas almacena la probabilidad estimada de su estado. En otras palabras, un mapa de esta naturaleza es una representación probabilística de la presencia o no de un objeto mediante la determinación de su estado. En esta clase de mapas existen 3 estados posibles para una celda: desconocido (UNK), ocupado (OCC) y vacío/libre (EMP).

2.6.2. Inconvenientes

Ahora bien, uno de los principales inconvenientes que afronta esta clase de representación de entornos es la reducción de la estimación de un entorno multidimensional a uno unidimensional, es decir, la estimación no se hace sobre un conjunto de celdas, sino que se realiza celda por celda. Esto provoca inconsistencia en el mapa generado, debido a que la representación pierde información relevante del entorno ya que cada celda representa una área del entorno. Esto podría acarrear problemas en cuanto a la definición de objetos sobre sus bordes, debido a que la determinación de sus bordes puede arrojar estados erróneos: el objeto puede expandirse o contraerse por este análisis individual. Sin embargo, en [23] se expone una técnica SLAM basada en la reconstrucción de mapas OGM, en la cual se ayuda a determinar el estado de una celda influenciada por sus vecinos más cercanos.

2.6.3. Aplicaciones

Los mapas OGM tienen la ventaja de ser los más adecuados para realizar tareas de SLAM, navegación individual y conjunta, exploración y reconstrucción de entornos, planeación y generación de trayectorias con evasión de obstáculos [3, 13, 55], debido al tipo de representación gráfica que usan. De aquí que esta clase de representación gráfica de entornos desconocidos sea la seleccionada para la técnica MR-SLAM desarrollada y basada en algoritmos de mezcla de mapas.

3. Algoritmo de mezclado de mapas por ocupación de celdas

El enfoque seleccionado para dar una solución consistente al problema de Multi Robot SLAM (MR-SLAM), fue el desarrollo de un Algoritmo de Mezcla de Mapas OGM por solapamientos, ya que, de acuerdo a la Sección 2.5.1, es el enfoque óptimo para la reconstrucción colaborativa de entornos desconocidos. Adicionalmente, como ventaja de este enfoque, no se requiere de ninguna condición inicial de los agentes para su puesta en operación. Sin embargo, es la aproximación que presenta mayores desafíos, puesto que su finalidad es encontrar zonas comunes entre dos mapas parciales (o solapamientos), en aras de calcular una transformada rígida que permita unir los mapas de manera consistente. Y es precisamente el cálculo de esta transformada el desafío más determinante, dado que el espacio de búsqueda entre los mapas es infinito y esto puede aumentar el tiempo de cómputo del algoritmo.

En la academia, pocas han sido las soluciones planteadas al problema de MR-SLAM con el enfoque escogido (mezcla de mapas por solapamientos). De acuerdo al estado del arte, solo 1 trabajo ha propuesto un algoritmo de mezcla para MR-SLAM. En [7], *Blanco et al.* hacen un estudio de las técnicas de detección y descripción de características en aras de seleccionar las que mejor se ajusten al algoritmo de mezcla propuesto en su artículo. Dicho trabajo científico es la base para el algoritmo de mezcla desarrollado y que en esta sección será detallado.

Como mejoras al trabajo propuesto en [7], se plantea reducir el tiempo del algoritmo de mezcla para el análisis de dos mapas OGM. En su artículo, Blanco et al. estimaron que el tiempo de su algoritmo toma alrededor de 1 segundo para analizar si dos mapas presentan solapamientos entre sí. Adicionalmente, la técnica propuesta por los autores solo aborda el problema de diseño del algoritmo y lo validan con grupos de mapas obtenidos de manera previa, es decir, validan su algoritmo con pruebas offline y no en tiempo de real. Contraria a este idea, el algoritmo de mezcla desarrollado en este trabajo se diseñó, desarrolló e implementó para ser validado con pruebas de ejecución en tiempo real, es decir, a medida que agentes van reconstruyendo su entorno de manera individual, el algoritmo de mezcla monitorea los mapas parciales de cada agente, los analiza y luego determina si son o no candidatos para ser mezclados de manera adecuada.

El algoritmo de mezcla desarrollado sigue el esquema o diagrama de flujo presentado en la Figura **3-1**. En él se muestra la etapa de verificación de mapas disponibles una vez son reportados por los agentes. Seguido a esto, inicia una etapa de detección, una etapa de descripción y luego una de apareamiento para determinar los posibles puntos en común entre dos mapas. Como se puede ver, el algoritmo sigue con una etapa de extracción de la transformada o modelo rígido de los mapas usando el algoritmo de RANSAC. Seguido, realiza un ajuste a los parámetros encontrados por el algoritmo de RANSAC, a fin de mejorar la consistencia del modelo hallado. Finalmente, ejecuta una etapa de verificación de la transformada calculada para saber si se acepta o rechaza. Una vez el algoritmo finalice esta etapa, puede continuar analizando los demás mapas que estén disponibles. En caso de que el algoritmo mezcle dos mapas, éste le informa a los robots involucrados en la operación sobre la actualización encontrada entre los mapas.

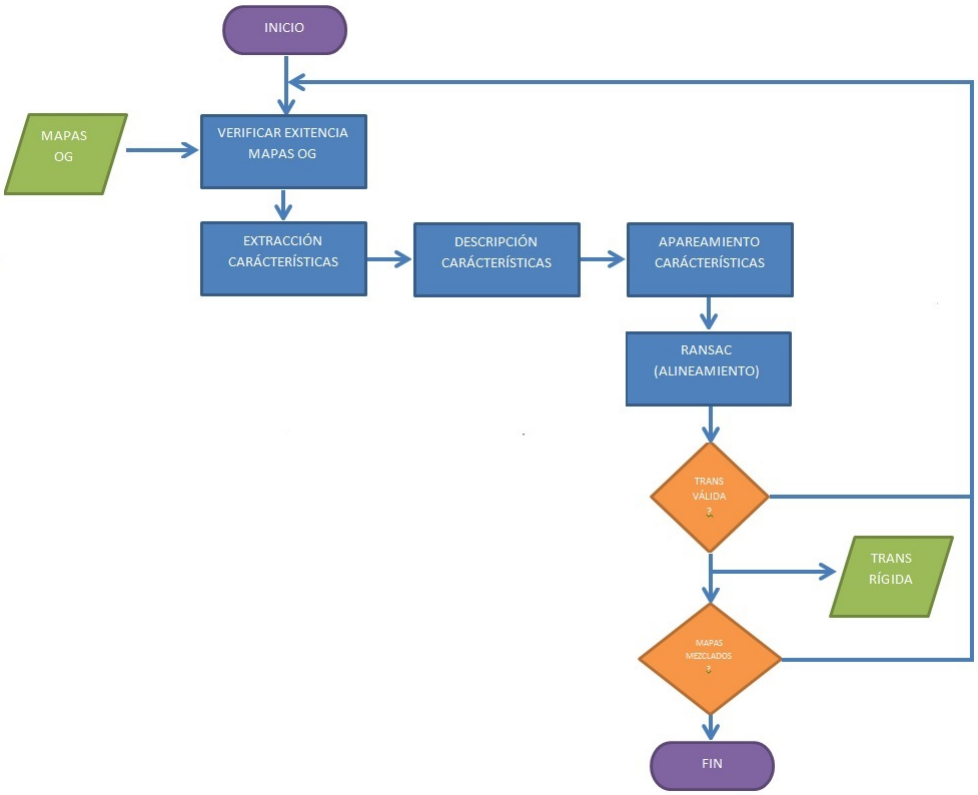

Figura 3-1.: Diagrama de flujo del algoritmo de mezcla de mapas para la técnica MR-SLAM desarrollada.

En las siguientes secciones se presenta el desarrollo de las etapas de extracción, descripción, apareamiento y alineamiento de puntos característicos del algoritmo de mezcla de mapas diseñado para la técnica MR-SLAM, Figura **3-1**. En la etapa de extracción de características, se muestra el trabajo desarrollado para seleccionar la mejor técnica de detección de puntos característicos. En especial, se propone y presenta una nueva técnica de detección de esquinas

diseñada para el algoritmo de mezcla. Para las secciones correspondientes a la descripción y apareamiento de características, se presenta el estudio, comparación y selección de la técnica más apropiada para la mezcla de mapas. Finalmente, se presenta el algoritmo de alineamiento de mapas a través del uso del algoritmo de RANSAC, así como una etapa de refinamiento de la transformada rígida hallada por este último.

3.1. Extractor de características

En el campo de *análisis y procesamiento de imagen*, los detectores de esquinas son los métodos más conocidos y usados para la detección de características. Trabajos recientes [10, 4] reportan pruebas y comparaciones entre diferentes técnicas. Sin embargo, el trabajo presentado en [27] presenta un análisis detallado de las técnicas más representativas que también son estudiadas y tomadas como base en este trabajo.

Los detectores de esquinas, como su nombre lo indica, son los responsables de seleccionar los puntos de mayor interés en una imagen, teniendo en cuenta cuán representativo es dicho punto para el área donde se encuentra. Esos puntos de interés pueden incluir esquinas, bordes de interés o regiones con alto grado de información para la imagen, es decir, áreas con un gradiente de imagen único o de interés. Con estos puntos de interés se pueden, principalmente, detectar líneas, figuras geométricas (cuadrados, triángulos, círculos, entre otras), formas definidas como cruces o letras, números, etc.

Ahora bien, como punto de partida del algoritmo de mezclado de mapas, se hace necesario aplicar una etapa de extracción de características en cada mapa generado por el algoritmo de SLAM. Esto se hace con el fin de reducir el espacio de búsqueda del algoritmo y garantizar una reducción en el tiempo de cómputo de los mapas, ya que se entiende que los puntos característicos de los mapas contienen la solución óptima para mezclarlos.

3.1.1. Detector de esquinas de Harris

Basado en el Detector de Esquinas de Moravec, en 1988 esta técnica fue propuesta por Harris y Stephens [18]. Esta técnica propone un análisis a través del gradiente de imagen y una métrica propia para aceptar/rechazar un píxel o punto de la imagen como una esquina o borde, Ecuación 3-1. Para el cálculo de dicha métrica, este método propone el uso de un tensor matricial calculado para cada píxel en la imagen, Ecuación 3-2. Este tensor incluye las derivadas de la imagen I_x, I_y en los ejes X y Y, a los cuales se les aplica un paso de convolución con un ventana Gaussiana de tamaño w.

$$m_h = det(M) - k * tr^2(M) \tag{3-1}$$

$$M = \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} = \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \tag{3-2}$$

3.1.2. Detector de esquinas de Shi-Tomasi

Esta técnica es una modificación del detector de esquinas de Harris. Básicamente, los autores modificaron la métrica establecida por Harris para identificar si un punto es o no una esquina. Mientras Harris usa la medida de existencia de esquinas de la Ecuación 3-1, Shi-Tomasi selecciona el valor mínimo de los eigenvalores del tensor matricial asociado a cada pixel [46]. Es decir, Shi-Tomasi modifica el segundo paso del algoritmo de detección de esquinas de Harris:

$$m_{shi} = min(\lambda_1, \lambda_2) \tag{3-3}$$

Esta simple modificación incrementó el desempeño del detector de esquinas, puesto que la medida calculada muestra información real y concisa de la distribución del gradiente de imagen sobre la imagen analizada. De ahi que la selección de puntos característicos como esquinas mejoró notablemente con esta modificación. Sin embargo, la técnica no logró resolver el problema relacionado con el paso de umbralización, porque el umbral seleccionado para el algoritmo es quien realmente se encarga de aceptar o rechazar puntos después de haber calculado las medidas de existencia de esquinas.

3.1.3. Detector de esquinas de Trajkovic-Hedley

Este detector fue propuesto por Miroslav Trajkovic y Mark Hedley en 1998 [48]. Su aproximación se conoce como detector de esquinas de Trajkovic-Hedley.

La técnica es un extractor de características basado en el análisis de los vecinos más cercanos a un punto (píxel), al analizar el nivel de gris de cada píxel vecino. Dicha técnica mostró un cambio sustancial en el análisis y detección de características en imágenes, ya que no usa una ventana o kernel que se mueve a través de todo el conjunto de datos de la imagen como lo hacen las anteriores técnicas. Sin embargo, Trajkovic y Hedley mantienen la misma definición de esquinas realizada por Moravec y Harris en sus técnicas: *"las esquinas son los puntos donde un cambio en la intensidad de imagen se presenta en todas las direcciones asociadas a ese punto"* [48].

3.1.4. Extractor de puntos característicos de SIFT

Como referencia en el campo del Registro y Correspondencia de Imágenes, Lowe [29] creó una técnica llamada Scale-invariant Feature Transform (SIFT) para la detección, descripción

y correspondencia de imágenes. Esta técnica probó ser más robusta que las descritas anteriormente, ya que Lowe logró demostrar que su extractor de características es *"invariante a escala, translación, rotación y parcialmente invariante a cambios en iluminación"* [29].

Como punto de partida, esta técnica inicia con una detección de las puntos más distintivos (extremos), también llamados puntos de interés, en el espacio multi-escala (space-scale extrema, en inglés). Para realizar esta operación, Lowe propuso usar los Diferenciales Gaussianos (DoG) a diferentes escalas, los cuales pueden ser calculados como los diferenciales entre dos imágenes cercanas en escala y ruido. Finalmente, la detección de puntos se realiza haciendo una búsqueda de puntos característicos en cada DoG resultante. Así mismo se calcula una orientación dominante sobre el punto encontrado y se procede a construir un descriptor por cada punto detectado.

Como limitaciones, el extractor de puntos de SIFT demanda tiempo para realizar la detección. Sin embargo, la idea usada en este extractor introdujo una buena aproximación hacia lo que debería ser el detector ideal. Por lo cuál, Lowe lanzó en 2004 una versión mejorada de su extractor de puntos para mejorar el tiempo de ejecución del mismo [29]. Adicionalmente, existen trabajos [12, 50] cuyo principal objetivo es hacer más eficiente el algoritmo de detección de puntos de SIFT.

3.1.5. Detector de esquinas desarrollado para SLAM

El detector de esquinas desarrollado se basa en la idea del detector Trajkovic-Hedley, el cual analiza los vecinos más cercanos a un píxel de interés, con el fin de establecer si este pertenece a un conjunto de píxeles que forman una característica (esquina). Este detector funciona con una umbralización básica en las imágenes para facilitar el análisis de los potenciales puntos de interés. Dado que los mapas generados por algoritmos de SLAM se componen de 3 estados (libre, ocupado y desconocido), ellos son representados con 3 niveles de gris: blanco (nivel 255), gris (nivel 127) y negro (nivel 0). De esta manera, el uso de gradientes o derivadas de imagen, filtros u otros pasos utilizados por otras técnicas, no se hacen necesarios. Solo basta con una umbralización para obtener la información relevante del mapa: los estados ocupados y libres de la imagen.

Algoritmo General

A través de un análisis de los 8 vecinos más cercanos de un píxel de interés, se ha desarrollado una estrategia que permite acelerar la búsqueda de esos puntos característicos en la imagen. Sea I la imagen de un mapa generado por SLAM e $I(x,y)$ la intensidad de imagen o nivel de gris del píxel (x,y). Un punto será considerado solo si su nivel de gris es 255 (blanco), $I(x,y)=1$, ya que este nivel representa el estado libre en el mapa de SLAM. Esta idea asegura que el punto analizado será el centro de la caracterśtica, puesto que el gradiente de imagen o

cambio en el nivel de intensidad será mayor cuando se esté en presencia de una característica real o esquina. La Figura **3-2** muestra el píxel analizado C y sus 8 vecinos más cercanos A, A', B, B', P, P', Q, Q'.

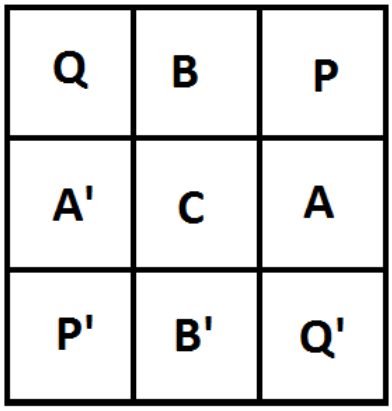

Figura 3-2.: El píxel analizado es C, los 4 vecinos director son A, A', B, B' y los 4 vecinos adyacentes son P, P', Q, Q'.

Con las consideraciones establecidas anteriormente, el algoritmo ejecuta:

1. Calcular las medidas r_S, r_D usando el conjunto de ecuaciones 3-4.

$$\begin{aligned} r_S &= A + A' + B + B' \\ r_2 &= P + P' \\ r_4 &= Q + Q' \\ r_D &= r_2 + r_4 \end{aligned} \tag{3-4}$$

2. Determinar la medida r que denota el número de píxeles con valor 0 en la escala de grises (color negro o estado ocupado en el mapa de SLAM).

$$r = r_S + r_D \tag{3-5}$$

3. Si r tiene un valor de 4, 5 o 6, calcular las medidas complementarias con las ecuaciones 3-6.

$$\begin{aligned} r_{sup} &= P + B + Q \\ r_{der} &= P + A + Q' \\ r_{inf} &= P' + B' + Q' \\ r_{esqP} &= A + P + B \end{aligned} \tag{3-6}$$

4. Analizar la medida r bajo las siguientes suposiciones:
   - Si r=*4*, el criterio de selección usado es:
     - *Si* $mod(r_{esqP}) == 0 \ \wedge \ r_S == 2 \ \wedge \ r_{sup} + r_{inf} == 3 \ \wedge \ r_2 != r_4$.
       Este criterio selecciona puntos con los patrones mostrados en la Figura **3-3**.

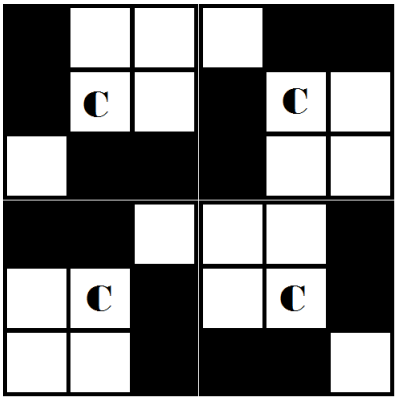

Figura 3-3.: Patrones o características extraídos con el criterio de selección aplicado para $r = 4$.

Para la asignación de la orientación de los patrones encontrados con estas características, se realizan asignaciones sobre la diagonal libre de estos patrones, es decir, sobre los píxeles color blanco. Los valores de orientación corresponden a los ángulos 45°, 135°, 225°y 315°. Esta asignación se hace teniendo en cuenta el sistema coordenado de la imagen. El usuario define cuáles son los puntos A y B (Figura **3-2**), los cuales corresponden a los ejes X y Y, respectivamente. El sentido de los ejes viene dado por los vectores $\vec{CA}$ y $\vec{CB}$.

- Si r=*5*, el criterio usado en este caso es:
  - *Si* $r_S == 2 \ \wedge \ mod(r_{sup} + r_{der}) == 0$.
    Este criterio selecciona puntos con los patrones mostrados en la Figura **3-4**.

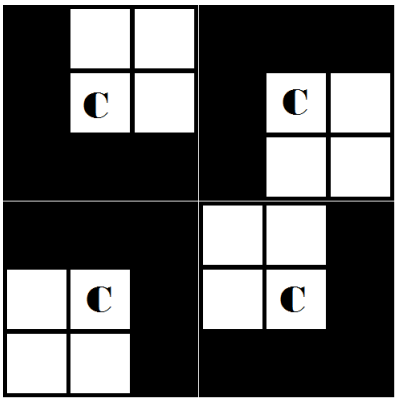

Figura 3-4.: Patrones o características extraídos con el criterio de selección aplicado para $r = 5$. Este patrón muestra la característica ideal alrededor del píxel C.

Nuevamente, para la asignación de la orientación de los patrones encontrados con estas características, se realizan asignaciones sobre la diagonal formada por los dos únicos píxeles libres o de color blanco de la región.

Estos ángulos corresponden a los valores de 45°, 135°, 225°y 315°. Esta asignación se hace teniendo en cuenta el sistema coordenado de la imagen, tal y como se explicó en el anterior inciso.

- Si *r=6*, el criterio seleccionado es:
  - *Si* $r_S == 3 \ \wedge \ r_D == 3$.
    En la Figura **3-5** se muestra los patrones extraídos por este criterio de selección. Las características con otra configuración de sus vecinos más cercanos no son consideradas por este criterio, ya que tienden a ser reconocidas como ruido por el algoritmo.

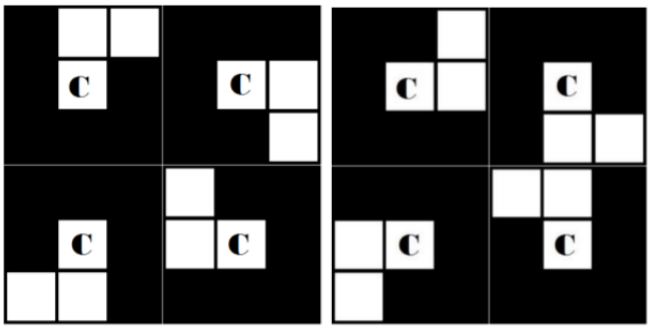

Figura 3-5.: Patrones extraídos con el criterio de selección aplicado para $r = 6$. Aunque estos patrones resulten extraños para ser tomados en cuenta como esquinas, son comunes en los mapas generados por SLAM.

Para la asignación de la orientación de los patrones encontrados con estas características, se realizan asignaciones sobre los dos únicos píxeles libres o de color blanco que están sobre el borde de la región. Estos ángulos corresponden a los valores de 22.5°, 67.5°, 112.5°, 157.5°, 202.5°, 247.5°, 292.5°y 337.5°. Esta asignación se hace teniendo en cuenta el sistema coordenado de la imagen, tal y como se explicó en el anterior inciso.

5. Los puntos seleccionados durante la aplicación de los criterios del paso 4, son considerados los puntos de interés.

Con estas simples reglas, el detector de esquinas desarrollado en este trabajo hace una rápida detección en su espacio de búsqueda, ya que reduce el número de píxeles analizados al no considerar los puntos con un nivel de gris diferente de 255 (color blanco) o 1, en una escala normalizada. Además, de las reglas aplicadas, el algoritmo asigna una orientación de la característica con una resolución angular de 22.5°, dado que es la menor resolución que se puede lograr con esta técnica.

Finalmente, el algoritmo puede ser configurado en el paso 4 de acuerdo a como el programador lo requiera. Es decir, es posible habilitar o deshabilitar las 3 condiciones aquí descritas para r ($r = 4, 5, 6$) o si se requiere, se puede adicionar una cuarta condición que únicamente evalúa

cuando $r = 7$. Sin embargo, con esta última condición se debe tener especial cuidado, ya que el algoritmo podría detectar más puntos de los necesarios y, por ende, perder su robustez. Las pruebas realizadas con esta última condición ($r = 7$) mostraron que puede ser usada en imágenes de alta dimensión en la escala de grises, mientras que para imágenes de mapas SLAM no provee mayor extracción de puntos bajo este patrón, ya que está condición en mapeo es poco frecuente por la manera como se genera un perfil o borde en el mapeo.

3.1.6. Selección de técnica de extracción de características

Para la selección adecuada de la técnica de extracción, se hizo necesario la realización de pruebas para comparar los resultados de cada una de las técnicas en términos de tiempo de ejecución, estabilidad y calidad de los puntos extraídos (falsos positivos detectados). En este trabajo, la selección basada en tiempo de cómputo es esencial ya que el algoritmo resultante será utilizado en una aplicación en tiempo real, de ahí la necesidad de establecer este parámetro como el primordial a evaluar entre las diferentes técnicas. Por otro lado, la estabilidad de las técnicas y la robustez a ruido de las técnicas (característica que incide en la calidad de los puntos extraídos), serán las medidas complementarias para decidir, en términos de eficiencia y precisión, cuál es el algoritmo de extracción de características más adecuado. Las técnicas usadas para la prueba y selección son: Detector de Harris, Detector de Shi-Tomasi, Detector Trajkovic-Hedley, Extractor de Puntos de SIFT y el detector de esquinas desarrollado en este trabajo.

Inicialmente, se dividió un conjunto de imágenes de prueba en 3 grupos. El primer grupo son imágenes binarizadas o en baja dimensión en la escala de grises (dos niveles son usados). La intención principal de este grupo de imágenes es probar que todos los extractores utilizados funcionan adecuadamente con imágenes cuyas características están bien acentuadas, ya que los cambios de intensidad o gradiente de imagen son altos. El segundo grupo está compuesto por imágenes de mapas SLAM, las cuales se conforman por 3 niveles en la escala de grises. En este tipo de imágenes se sabe que la presencia de ruido es alta, debido a que las técnicas SLAM reconstruyen mapas haciendo uso de métodos probabilísticos aplicados a un sensor láser que, comúnmente, también es sensible al ruido proveniente del entorno de trabajo. El tercer y último grupo se compone de imágenes en alta dimensión en la escala de grises, es decir, imágenes que utilizan gran parte de los diferentes niveles de gris y cuyas características pueden ser más difíciles de extraer, debido a que pueden no presentar grandes cambios en sus niveles de intensidad.

Cada técnica fue ejecutada 20 veces para probar la estabilidad de los algoritmos. Como se dijo anteriormente, esta propiedad es una métrica a tener en cuenta para la selección del detector ideal. A medida que se corría cada algoritmo, se medía el tiempo de cómputo de cada uno de ellos y siempre se seleccionaba el tiempo mínimo de ejecución del algoritmo. De

esta manera, se obtenía la información necesaria para seleccionar las técnicas más destacadas en cuanto a tiempo de ejecución y así cumplir con el criterio de selección de mayor peso para este trabajo. La tercera propiedad, calidad de puntos extraídos, se mide de acuerdo a los resultados obtenidos por estabilidad de cada una de las técnicas probadas. La estabilidad incide en la calidad de los puntos extraídos, ya que si un algoritmo no es estable, tendrá resultados diferentes en cada iteración y, por ende, será difícil determinar o analizar los resultados en cuanto a la calidad de los puntos extraídos. De ahí la importancia de medir la estabilidad de las técnicas aquí estudiadas.

a. Grupo de imágenes de prueba para los detector de características/esquinas

Como se dijo anteriormente, el primer grupo de imágenes usadas corresponde un grupo de imágenes de baja dimensión en la escala de grises. Con esta clase de imágenes, se espera que un detector común pueda funcionar de manera correcta, ya que las características presentes se encuentran bien demarcadas. La Figura **3-6** muestra las imágenes de prueba usadas en este grupo.

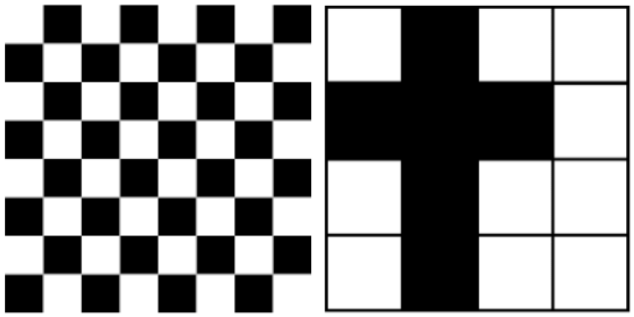

Figura 3-6.: Imágenes usadas en el primer grupo de prueba. Las imágenes poseen dos valores extremos en la escala de grises: negro y blanco. Imagen izquierda: ajedrez, tamaño 204×204; imagen derecha: cruz, tamaño 507×507.

El segundo grupo de imágenes consiste en imágenes generadas por algoritmos de Mapeo y Localización Simultanea (SLAM). Estas imágenes también son consideradas de baja dimensión en la escala de grises. Solo usan 3 niveles de gris. Para este grupo de imágenes, el algoritmo de detección propuesto no requiere de ninguna etapa de filtrado, reducción de ruido o extracción de borde. Sin embargo, para las demás técnicas, de acuerdo a lo propuesto por sus respectivos autores, si se hace necesario el uso de una o más de estas etapas. Las imágenes utilizadas para este grupo se muestran en la Figura **3-7**. Los niveles de gris usados en estas imágenes son: Negro (0), Gris (127) y Blanco (255).

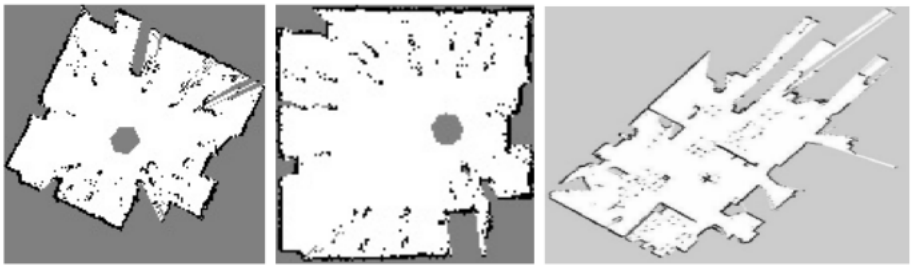

Figura 3-7.: Imágenes que componen el segundo grupo de prueba. Imagen izquierda: mapa 1, tamaño 146×144; imagen central: mapa 2, tamaño 113×113; imagen derecha: mapa 3, tamaño 398×402.

El tercer grupo se compone con imágenes en alta dimensión en la escala de grises. Esta es una extensión del trabajo inicial propuesto, ya que el objetivo central es el desarrollo de una técnica de extracción para imágenes de mapas SLAM (Figura **3-7**). Sin embargo, a lo largo de las pruebas realizadas al detector de esquinas desarrollado, se encontró que éste tiene un notable desempeño con esta clase imágenes. Cabe destacar que estas imágenes corresponden a imágenes reales, comúnmente capturadas por equipos fotográficos o de vídeo, de ahí la importancia de realizar una extensión y prueba del algoritmo propuesto. Adicionalmente, se toma la imagen de Lena como referente en este trabajo, ya que es la imagen más utilizada en los trabajos investigativos referentes al campo de Visión por Computador. La Figura **3-8** muestra las imágenes que componen este grupo.

Figura 3-8.: Imágenes del tercer grupo de prueba usado en las pruebas. Imagen izquierda: Lena, tamaño 521×512; imagen central: Ventana 1, tamaño 479×634; imagen derecha: Ventana 2, tamaño 480×352.

b. Configuraciones y consideraciones de programación

Dado que para una adecuada selección de la técnica de extracción más conveniente, se hizo necesaria una configuración de cada técnica con el fin de garantizar el mejor desempeño posible de cada algoritmo.

La gran mayoría de técnicas se programaron haciendo uso del conjunto de librerías OpenCv 3.0 y OpenCv Contrib, donde se encuentran los algoritmos oficiales y optimizados. La única técnica no disponible en los paquetes de OpenCv es el detector de Trajkovic-Hedley, luego esta técnica se programó de acuerdo al algoritmo base expuesto por los autores en [48]. El lenguaje de programación utilizado fue C++ y el Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) utilizado fue QT 4.8, el cual permite a través de su interfaz de programación el llamado y enlace de código nativo en C++ y de librerías como OpenCv. El equipo de cómputo utilizado cuenta con un procesador de 2.2GHz con turbo boost, 4GB de memoria RAM y sistema operativo Ubuntu 14.04 LTS.

Con base en trabajos previamente publicados [10, 4], la selección de parámetros para los detectores de Harris y de Shi-Tomasi fueron iguales, ya que ambas técnicas son similares y solo difieren en el criterio de selección de puntos característicos. Para ambas técnicas, fue seleccionado un kernel o ventana de tamaño 3x3. Esta ventana es usada para el filtro Gaussiano usado por ambas técnicas. De acuerdo a [10], un incremento en el tamaño de esta ventana no mejora el rendimiento de las técnicas, pero si incrementara significativamente el tiempo de cómputo empleado por cada técnica. Adicionalmente, para el algoritmo de Harris, el parámetro k, usado en el criterio de medida de existencia de un punto característico, se estableció en *0.04*, ya que es el valor recomendado por los autores. La desviación estándar del filtro Gaussiano utilizado fue establecida a *1*.

Por otro lado, para el algoritmo de Trajkovic-Hedley, fue seleccionada la técnica de análisis de los 4 vecinos directos. Los umbrales T_1 y T_2, que permiten la selección de puntos característicos y que son usados por la técnica, fueron establecidos de acuerdo a cada grupo de imagen. Este cálculo se hizo con el fin de obtener los mejores resultados del algoritmo para cada imagen de prueba. En las pruebas desarrolladas con esta técnica, los resultados mostraron que el análisis realizado por el algoritmo original de Trajkovic-Hedley en baja resolución puede provocar una pérdida de la precisión del algoritmo cuando realiza el análisis de los puntos en alta resolución, debido a que un punto en baja resolución podría no corresponder exactamente a la misma ubicación en alta resolución.

Finalmente, para el extractor de puntos de SIFT no se realizaron configuraciones adicionales, ya que el paquete OpenCv Contrib expone la mejor versión a usar de esta técnica. Como se sabe, la técnica SIFT esta patentada, sin embargo el paquete Contrib es la única versión disponible del algoritmo oficial de Lowe. Para el detector propuesto la única consideración que se tuvo fue el uso del extractor de borde de Sobel para las imágenes del tercer grupo. Cabe resaltar que en las pruebas realizadas solo se tiene en cuenta el tiempo de extracción de puntos de cada algoritmo, así que etapas de preparación, carga y visualización de la imagen de prueba, conteo o procesamiento adicional de las características extraídas no es tenido en cuenta para el análisis que se llevó a cabo. Se respetaron los algoritmos tal y como los

autores los plantearon.

c. Estabilidad de extractores de características – Resultados

Una de las propiedades a analizar en cada una de las técnicas estudiadas es la estabilidad de la técnica al momento de realizar la extracción de puntos de una imagen. Su importancia radica en que gracias a esta propiedad, se puede determinar si una técnica tiene un comportamiento estable y esperado bajo ciertas condiciones de operación, sobre todo si esta va a ser utilizada en aplicaciones en tiempo real.

La estabilidad de una técnica garantiza que si una misma imagen o conjunto de datos son aplicados a un algoritmo, este responda de manera estable y repetida, es decir, en cada iteración el algoritmo debería concluir la misma respuesta, dado que los datos de entrada siguen siendo los mismos. Esta propiedad va garantizar que a medida que un mapa es generado por un algoritmo de SLAM, los puntos extraídos en interacciones anteriores se conserven y no cambien, a menos que el algoritmo de SLAM cambie los datos del mapa ya reconstruido.

Los resultados de estabilidad de las técnicas analizadas dieron un 100 % de estabilidad para cada algoritmo. Por ende, todas las técnicas analizadas cumplen con el criterio de estabilidad, lo que quiere decir que son aptas para ser utilizadas en algoritmos de mezcla de mapas. Ninguna de las técnicas, durante las ejecuciones realizadas mostró cambios ni en el número de puntos extraídos ni en la posición de los mismos.

d. Calidad y cantidad de los puntos extraídos – Resultados

La calidad y cantidad de los puntos de interés detectados por cada una de las técnicas es una de las métricas escogidas para la selección de la técnica de detección de características, ya que incidirá directamente en la etapa de correspondencia de puntos de interés entre mapas. Si la técnica no logra extraer suficientes puntos de una imagen, la posibilidad de encontrar un solapamiento entre mapas disminuirá, ya que tendrá pocas opciones de encontrar parejas de puntos de dos mapas o imágenes diferentes. Por el contrario, si la técnica es muy sensible y detecta una gran cantidad de puntos interés, el algoritmo de apareamiento (correspondencia) iterará tratando de encontrar una posible pareja de puntos, ya que normalmente estos algoritmos iteran sobre todos los puntos de interés encontrados en las fases de detección y como se sabe, el objetivo principal de este trabajo es la implementación de un algoritmo de mezclado en tiempo real, luego si la etapa de apareamiento requiere mucho tiempo, el algoritmo no será el adecuado para su implementación en tareas de reconstrucción colaborativa de entornos.

Por otro lado, la calidad de los puntos detectados es una propiedad a tener en cuenta, ya que las imágenes generadas por algoritmos de SLAM contienen información ruidosa debido a los

métodos probabilísticos que reconstruyen esta clase de mapas. Cabe destacar que el ruido en los mapas de SLAM no tiene la misma concepción que en Visión por Computación, ya que en el campo de SLAM se refiere a la existencia probabilística o no de un obstáculo u objeto que es posicionado en el mapa como un píxel de color negro. Es decir que esta clase de mapas, la presencia de píxeles color negro de manera aislada es común y por ende, una técnica sensible a grandes cambios de gradiente podría detectar estos solitarios píxeles como puntos de interés. Ahora bien, para ambientes internos, los mapas de SLAM tienden a contener información ruidosa del mobiliario y de objetos cuyo tamaño no es apreciable para el sensor láser.

Los resultados en esta sección son mostrados por grupos de imágenes, todo con el fin de mostrar las diferencias entre las técnicas. Para el primer y tercer grupo de imágenes de prueba, los resultados se encuentran en el anexo *Calidad y cantidad de puntos extraídos para grupo 1 y 3 de imágenes de prueba*, ya que el objetivo principal de este desarrollo se concentra en el segundo grupo de imágenes

Segundo grupo de imágenes de prueba – Imágenes de SLAM

Este grupo de imágenes es el objetivo principal del estudio realizado para la selección del detector de características, ya que el algoritmo de mezclado solo trabajará con imágenes de mapas generados por SLAM.

Las figuras **3-9**, **3-10**, **3-11**, **3-12** y **3-13** muestran los resultados obtenidos de las imágenes originales de la Figura **3-7**. Las técnicas en su gran mayoría tienen un comportamiento destacado con este tipo de imágenes a excepción del detector Trajkovic-Hedley. Dada la tipología y el alto contenido de ruido en los datos de la imagen (producto del algoritmo de SLAM y el sensor láser), el detector Trajkovic-Hedley detecta todos los obstáculos del mapa como puntos característicos. Con este resultado se concluye que para imágenes en baja dimensión en la escala de grises, este detector no es el adecuado debido a los resultados obtenidos.

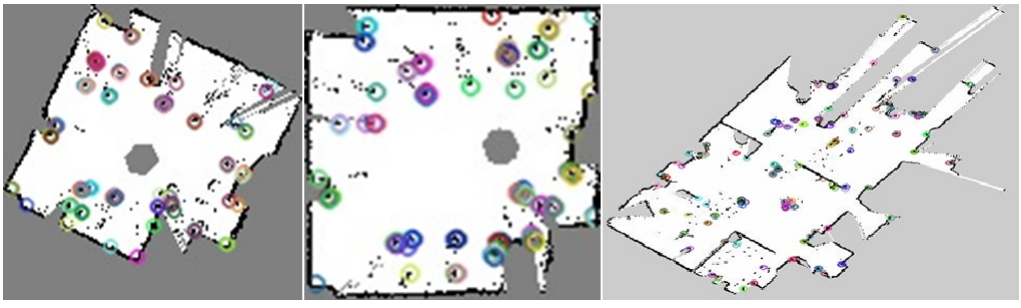

Figura 3-9.: Puntos de interés extraídos por el detector de Harris en imágenes generadas por algoritmos de SLAM.

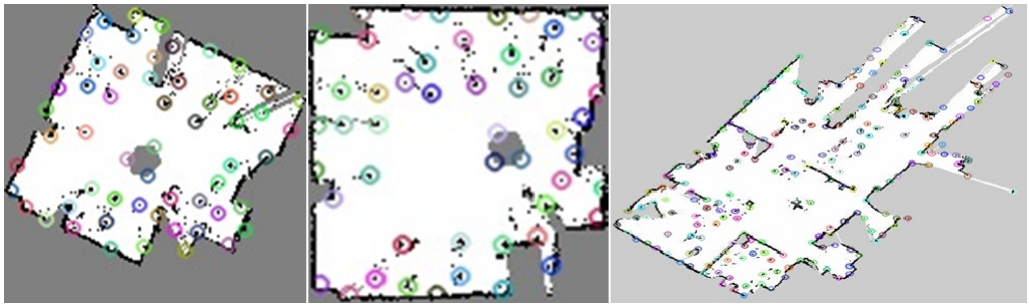

Figura 3-10.: Puntos Característicos extraídos por el detector Shi-Tomasi en imágenes generadas por algoritmos de SLAM.

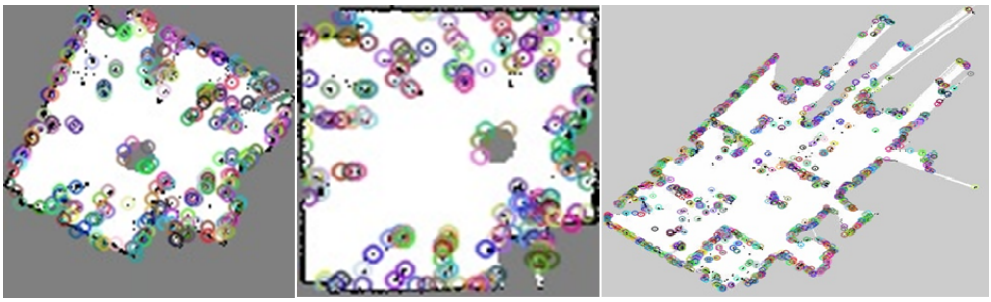

Figura 3-11.: Puntos de interés extraídos por el detector Trajkovic-Hedley en imágenes de prueba del grupo 2 (mapas de SLAM).

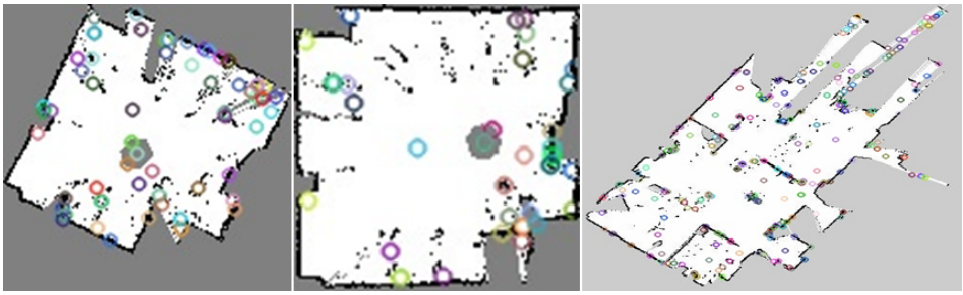

Figura 3-12.: Puntos de interés extraídos por el extractor de puntos de SIFT en imágenes generadas por algoritmos de SLAM.

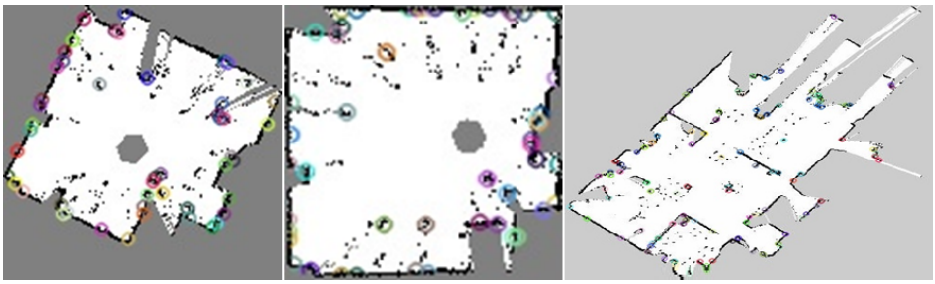

Figura 3-13.: Puntos de interés extraídos por el detector de esquinas desarrollado para imágenes generadas por SLAM.

Otro análisis que resulta de las figuras **3-9**, **3-10**, **3-11**, **3-12** y **3-13**, es la detección de esquinas o puntos de interés de la técnica desarrollada en este trabajo. Cabe resaltar que esta técnica centra la detección de puntos en las áreas de mayor presencia de bordes, sin caer en el problema de detección excesiva de puntos del detector Trajkovic-Hedley. Esto demuestra, a diferencia de las demás técnicas, que el algoritmo propuesto es robusto frente a la presencia marcada de bordes y de alto ruido en las imágenes de mapas SLAM, ya que si se analiza con detenimiento, esta técnica es la única que conserva las áreas de detección sobre bordes realmente característicos y no sobre áreas que pueden considerarse de alto ruido dentro del mapa. Como se mencionó anteriormente, en los mapas generados por SLAM es normal encontrar puntos solitarios que pueden no corresponder a obstáculos reales en el entorno, luego si una técnica llega a considerar estos puntos como características del mapa, es probable que solo estén presentes en ese mapa en específico y no en los demás, ya que al ser considerado un ruido producto de un método probabilístico, existe la alta posibilidad que esas mismas áreas no existan en otros puntos. Tal detección de características se puede encontrar en los resultados de los detectores de Harris y Shi-Tomasi, donde se evidencia que el detector de Harris se concentra en zonas específicas, mientras que el detector de Shi-Tomasi tiende a detectar en exceso características sobre áreas con alta presencia de ruido.

La Tabla **3-1** muestra el número de características halladas por cada técnica. En los resultados de la tabla se evidencia la tendencia analizada del detector de Trajkovic-Hedley, la robustez del algoritmo propuesto al ser el único que conservó la idea esencial de extracción de puntos realmente característicos.

	Mapa 1	**Mapa 2**	**Mapa 3**
Técnica	#Características Detectadas	#Características Detectadas	#Características Detectadas
Detector de Harris*	93	93	174
Detector de Shi-Tomasi	57	50	217
Detector de Trajkovic**	235	126	1117
Extractor de SIFT	92	49	414
Detector Desarrollado	83	81	103
*Umbral: 140		**T_1: 8000 y T_2: 4000	

Tabla 3-1.: Cantidad de Puntos detectados por cada Técnica en imágenes generadas por algoritmos de SLAM.

e. Tiempo de ejecución de los extractores de características – Resultados

En lo que respecta a los tiempos de ejecución de cada técnica, se seleccionó el tiempo de menor

ejecución en las 20 iteraciones realizadas, con el fin de tener el mejor tiempo de cómputo. Este es el parámetro que incidirá mayoritariamente en la selección del detector adecuado, ya que el resultado final del algoritmo de mezclado es su implementación en tiempo real en tareas de reconstrucción colaborativa de entornos. Los resultados obtenidos para cada técnica en cada grupo de imágenes se muestran en la Tabla **3-2**.

	Ajedrez	**Cruz**	**Mapa 1**	**Mapa 2**	**Mapa 3**	**Lena**	**Ventana 1**	**Ventana 2**
Técnica	**Tiempo (μsec)**	**Tiempo (μsec)**	**Tiempo (μsec)**	**Tiempo (μsec)**	**Tiempo (μsec)**	**Tiempo (μsec)**	**Tiempo (μsec)**	**Tiempo (μsec)**
Detector de Harris	2363	15085	1095	632	9106	14095	18012	9163
Detector de Shi-Tomasi	1365	10632	1095	632	9106	14095	18012	9163
Detector de Trajkovic	1831	3127	1393	625	15524	28857	14430	12175
Extractor de SIFT	17972	114290	11321	6510	75746	131277	156270	95303
Detector Desarrollado	454	2745	384	228	2349	4877	4816	2576

Tabla 3-2.: Tiempo de cómputo en microsegundos asociado a cada una de las técnicas analizadas. El tiempo mostrado corresponde el tiempo mínimo de cómputo seleccionado de 20 iteraciones para cada algoritmo.

Los resultados muestran que la técnica con el desempeño menos destacado fue el detector de SIFT. Este resultado era previsible si se tiene en cuenta que el detector de SIFT hace un análisis multi-escala de la imagen procesada para hacerla invariante a escala. El cálculo de las 4 Octavas, cada una conformada por 5 imágenes, requiere de un mayor tiempo si se compara con las demás técnicas que solo usan 1 o 2 versiones de la imagen original para realizar la extracción.

Por otro lado, pese a la divergencia en la cantidad y calidad de puntos de interés extraídos, las técnicas de Harris y Shi-Tomasi son comparativamente iguales en el tiempo de ejecución que requieren para realizar la detección. Este resultado tiene su explicación en que ambas técnicas utilizan los mismos pasos de cálculos de gradiente de imagen, derivadas de imagen y cálculo de la matriz Hessiana, pero difieren en el último paso de aceptación o rechazo del punto como característica. Probablemente, las operaciones involucradas en el cálculo de la métrica para cada una de las técnicas sea quien incida en los resultados. Es decir, mientras que Shi-Tomasi solo selecciona de dos posibles números, el valor mínimo, el detector de Ha-

rris calcula la traza, el determinante y finalmente la medida de aceptación del punto.

Finalmente, los resultados de la Tabla **3-2** muestran que la técnica desarrollada en este trabajo para mapas de SLAM, fue la de mejor rendimiento, es decir, fue la que requirió menos tiempo en cada uno de los grupos de imágenes utilizados. Con relación a la técnica Shi-Tomasi, que mostró un destacado desempeño a lo largo de las pruebas, el detector desarrollado es 2 veces más rápido en tiempo de ejecución y por ende su uso en aplicaciones en tiempo real se muestra como una real opción para algoritmos de mezclados.

f. Selección de extractor de esquinas

Con los resultados mostrados en las secciones anteriores, se puede resaltar que el detector desarrollado es equivalente a la técnica Shi-Tomasi en términos de tiempo de ejecución y cantidad de puntos extraídos para todos los grupos de imágenes utilizadas. Ambas técnicas extraen puntos en áreas características y también a lo largo de toda la imagen. Sin embargo, como se mostró en la sección 3.1.6.a, existe una diferencia considerable en las imágenes de mapas SLAM, ya que el detector de Shi-Tomasi es sensible al ruido normal contenido en esta clase de imágenes. Es inevitable no contar con la presencia de esta imperfección en los mapas generados por técnicas de SLAM. Adicionalmente, es notable el desempeño del detector Shi-Tomasi en imágenes en alta dimensión en la escala de grises, ya que su parámetro de dispersión provoca que el algoritmo no se concentre en zonas específicas como lo hace el detector de Harris. Sin embargo, el objetivo de estudio de estas técnicas es basado con las imágenes propias de los mapas de SLAM, por ende la sensibilidad excesiva a ruido en estas imágenes provocar que el algoritmo de mezclado converja a soluciones falsas o simplemente no converja.

Los resultados en tiempo de cómputo para el extractor de puntos de SIFT, hace que esta técnica no sea viable para una implementación real, ya que el análisis de un mapa de SLAM puede tomar hasta 75 milisegundos, mientras que para las demás técnicas el tiempo promedio es de 10 milisegundos. Ahora bien, los detectores de Harris y Trajkovic-Hedley mostraron una debilidad común que las demás técnicas no poseían, requieren ser configuradas dependiendo del tipo de imagen a analizar. Esta deficiencia hace poco probable que sean utilizadas en aplicaciones en tiempo real, puesto que requerirán de una etapa de ajustado de sus parámetros (umbrales) y esto, evidentemente, requerirá un tiempo adicional ya que las imágenes, como se realizó en este estudio, pueden ser de diferentes tipos.

Por otro lado, los resultados de tiempo de ejecución son claros en precisar que el detector propuesto es más rápido que los demás detectores. Por ende, y basados en el análisis mostrado a lo largo de las diferentes sub secciones, el algoritmo que se adecua a las necesidades del algoritmo de mezclado de mapas es el desarrollado y propuesto en este trabajo investi-

gativo. La explicación principal a este resultado es que la técnica fue desarrollada y pensada específicamente en resolver la detección eficaz de puntos en mapas generados por SLAM.

3.2. Descriptor de características

La etapa de detección de puntos característicos, el procedimiento común a seguir corresponde a describir o encontrar una representación adecuada para cada punto extraído. Como su nombre lo indica, el descriptor tiene como tarea principal describir un punto característico asociado a un conjunto de datos o, como lo es en este caso, a imágenes. El descriptor debe proporcionar información relevante (distintiva), única y robusta que permita diferenciar, lo más marcadamente posible, los puntos característicos entre sí. Para esto, se buscan patrones, regiones, momentos o texturas únicas alrededor del punto, con el fin de encontrar un vector característico que permita describir la región circundante del punto característico.

Esta etapa de descripción no se aplica directamente en la etapa de extracción, ya que el costo asociado a este análisis sobre cada píxel de la imagen, hace que el tiempo de respuesta del algoritmo sea lento. La finalidad con el descriptor es la de permitir que la fase de apareamiento (matching en inglés) sea lo más rápida y efectiva posible. De esta manera, un algoritmo de rastreo y seguimiento de puntos o un algoritmo de mezcla de datos puede operar de manera óptima y eficiente en aplicaciones en tiempo real.

Ahora bien, dada la cantidad de información contenida en la imagen, existen una gran variedad de descriptores útiles para diversas aplicaciones. Sin embargo, existen descriptores de referencia sobre los cuales se proponen nuevos y se realizan comparaciones para saber qué tan buena es la aproximación propuesta o cual se ajusta más al problema. Para este trabajo, se estudiaron, programaron y probaron diferentes descriptores de características comúnmente usados en el mundo académico y científico, dos de los cuales siguen siendo referentes en el campo de la Visión por Computador: el descriptor de SIFT y el descriptor de SURF. A estos descriptores, se le suman el descriptor circular o polar, el descriptor logarítmico y el descriptor ORB.

3.2.1. Descriptor de SIFT

Una vez el extractor de SIFT obtiene los puntos de interés, se les debe asignar una orientación dominante a lo largo de una región circundante al punto característico. Con esto, se toma una ventana de 16x16 píxeles y se crean sub regiones de 4x4, para tener un total de 16. En cada sub región se toma la orientación del gradiente de imagen calculada con anterioridad. Se crea un vector de 8 bins (o datos) donde se almacenan las orientaciones de cada píxel. Dado que para cada sub región se crea un vector de 8 datos, el descriptor final de SIFT

tiene un total de 128 datos. La extensión de este cálculo se encuentra detallada por Lowe en [29, 30].

3.2.2. Descriptor de SURF

En 2006, *Bay et al.* propusieron una técnica novedosa y comparable con el algoritmo de SIFT llamada Speeded Up Robust Features [5] o SURF por sus siglas en inglés. Esta técnica mostró ser un aporte significativo al Análisis y Procesamiento de Imágenes, dado que el tiempo de cómputo mejoró notablemente respecto al tiempo de cómputo de SIFT, sin poner en riesgo las bondades ampliamente conocidas de esta última. De hecho, los autores de SURF introdujeron el algoritmo como una versión mejorada de SIFT. En la actualidad, el algoritmo de SURF es ampliamente conocido y difundido, los trabajos relacionados con esta área de conocimiento deben incluirlo como referente. Sin embargo, al igual que SIFT, el algoritmo oficial se encuentra patentado. El trabajo completo de este descriptor se encuentra en [5, 45].

3.2.3. Descriptor ORB

En 2011, fue propuesto el algoritmo Oriented FAST and Rotated BRIEF (ORB) por *Rublee et al.* [36] como una alternativa a los algoritmos de SIFT y SURF, ya que reduce el costo computacional en su desempeño, pero sin perder las características presentes en SIFT y SURF. Adicionalmente, los autores proponen esta técnica como un algoritmo de libre acceso, puesto que una fuerte limitante de SIFT y SURF es que ambos algoritmos se encuentran patentados. Por otro lado, como su propio nombre lo indica, ORB hace una combinación de las técnicas FAST y BRIEF. La primera técnica es un detector de características binario, mientras que el segundo es el descriptor utilizado para detectores binarios. La novedad de esta técnica radica en el paso de rotación que aplica al descriptor de BRIEF para hacerlo invariante a rotación. De esta manera, hace distintivos los descriptores asociados a los puntos característicos detectados y ofrece una solución equiparable y rápida a las técnicas de SIFT y SURF.

3.2.4. Descriptor Circular

El Descriptor Circular o Descriptor Cilíndrico utiliza el concepto de integración sobre un área circular alrededor de un punto específico [8]. Este descriptor es ampliamente utilizado en aplicaciones que utilizan cámaras ojo de pescado, debido a que éstas presentan una distorsión circular respecto al centro del lente de la cámara.

Este descriptor, al hacer un barrido por una sección circundante a un punto, da como resultado un descriptor 2D de dimensiones $N \times M$, siendo N las filas y M las columnas. Ahora bien, la información contenida en dicho descriptor proviene de coordenadas polares, dado

que este espacio es la mejor representación de una sección circular cualquiera. En específico, las filas N representan el radio r de la coordenada y θ está representada en las columnas M.

La integración sobre el área circular corresponde a una doble integración sobre un rango $[R_{min}, R_{max}]$ para r y $[0, 2\pi]$ para θ. El acotamiento de r permite evitar cálculos sobre el punto de interés dado que la información puede no ser distintiva por cuenta de que el píxel de interés solo tiene 8 vecinos circundantes. Para r, se establece un Δr que determina el valor de las filas N del descriptor, es decir, $N = \frac{R_{max}-R_{min}}{\Delta r}$. Para θ se establece también un delta $\Delta\theta$ que determina el número de las columnas M, es decir, $M = \frac{2\pi}{\Delta\theta}$. Ahora bien, para el asocio de la información del descriptor, se definen los índices i, j cuyo rango está definido por $[0, M-1]$ y $[0, N-1]$, respectivamente. La Ecuación 3-7 representa función sobre la cual se calcula el descriptor circular,

$$\begin{aligned} f_a[i,j] &= \int_{\phi_j}^{\phi_{j+1}} \int_{r_i}^{r_{i+1}} m \begin{bmatrix} x_a + rcos(\theta) \\ y_a + rsin(\theta) \end{bmatrix} drd\theta \\ r_i &= R_{min} + i\Delta_r \\ \phi_j &= j\Delta_\phi \end{aligned} \tag{3-7}$$

Donde *m(x)* corresponde al valor de intensidad del píxel circundante de la imagen, *fa(i, j)* es el valor de intensidad de i y j. r_i representa el cambio de r en cada iteración, lo mismo que ϕ_j para la variable θ.

Al analizar cómo se construye el descriptor cilíndrico, salta a la vista la gran relevancia que tiene este descriptor dado que mapea un área circundante al punto de interés del espacio cartesiano al polar, Figura **3-14**, reduciendo a una sola variable el espacio de búsqueda del algoritmo de apareamiento. Se pasa de las variables X, Y y θ a solo θ, dado que el radio r que este descriptor usa es conocido, en magnitud, para todos los descriptores. Por lo tanto, la tarea de apareamiento se reduce a encontrar el desfase o corrimiento en entre dos descriptores y si *a priori* se tiene un conocimiento de la orientación del punto de interés, el algoritmo de apareamiento solo tiene que refinar este desfase en un pequeño rango de búsqueda sobre θ.

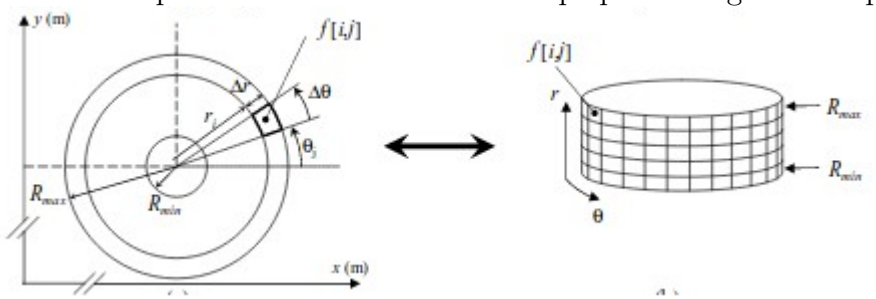

Figura 3-14.: Integración sobre el punto de interés para cálculo de descriptor cilíndrico. Tomado de [8].

Adicionalmente, este descriptor por la forma constructiva, es el único que representa informa-

ción que característica de la imagen original, ya que incluye información sobre la intensidad de los píxeles de la imagen original, formando así una marca visual única, característica distintiva de una región sobre un punto de interés, Figura **3-15**. Sin embargo, como desventaja se puede notar que al ser un descriptor 2D su cálculo puede requerir de un tiempo de cálculo mayor que los demás descriptores. Los deltas (Δr y $\Delta\theta$) que definen el tamaño del descriptor influyen directamente en el tiempo de cómputo del descriptor, pero también influyen en la calidad de la marca representada en el descriptor: si estos parámetros son grandes, el tiempo de computo crece, pero la marca es mucho más fina y única; mientras que un valor pequeño de estos parámetros disminuye el tiempo de cálculo, pero genera una marca gruesa que puede no ser tan única o distintiva. Depende del investigador la calibración de estos parámetros para obtener el mejor resultado.

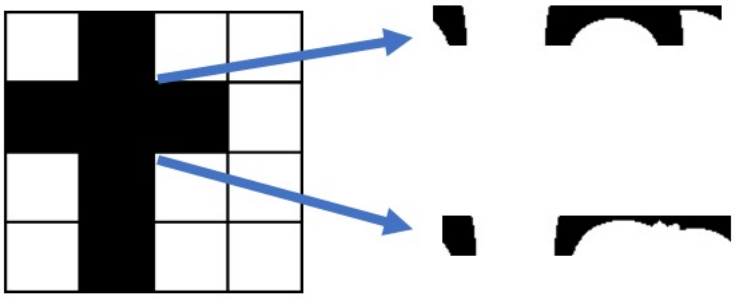

Figura 3-15.: Descriptores cilíndricos para dos puntos característicos de imagen de prueba.

Finalmente, vale la pena notar que este descriptor tiene una variante llamada Descriptor Circular Logarítmico, donde se realiza el mismo proceso de integración, salvo que en las filas se representa información en la escala logarítmica, es decir, r, R_{min}, R_{max} y Δr se representan en la escala logarítmica.

3.2.5. Selección de descriptor de característica

Para la selección del Descriptor de Característica idóneo para el Algoritmo de Mezcla de Mapas se realizaron pruebas sobre el banco de imágenes descritos en la sección 3.1.6.a. En específico, se realizaron 10 pruebas sobre cada imagen con cada descriptor. Con esto se mide el tiempo de cómputo del descriptor, que es la variable apropiada para establecer cuál descriptor escoger. En este punto, cabe recordar que el tiempo de cálculo del algoritmo de mezcla es lo que se intenta reducir, ya que lo que se busca es obtener el menor tiempo de cómputo del algoritmo de mezcla de mapas.

a. Consideraciones

Como se mencionó anteriormente, el banco de prueba seleccionado se compone de 8 imágenes

divididas en 3 grupos (ver Sección 3.1.6.a). El grupo objetivo de este análisis es el conformado por imágenes que representan Mapas por Ocupación de Celdas.

Por otra parte, en lo que respecta a los descriptores analizados, se evalúa el tiempo de cómputo del descriptor, no del algoritmo de detección. Ahora bien, en aras de respetar las técnicas analizadas, los algoritmos de descripción usados son los publicados por cada autor. Solo para el caso del descriptor circular y descriptor logarítmico se utiliza el algoritmo desarrollado para este trabajo. Dado que cada descriptor viene con un extractor de características único, el combinar extractores y descriptores no compatibles puede ir en detrimento del descriptor, puesto que cada técnica de descripción puede verse sesgada en sus cálculos sobre un punto de interés que realmente no es compatible con el tipo de información o dato que utiliza el descriptor. Parar citar un par de ejemplos, se utilizó el algoritmo de detección desarrollado con todos los descriptores. Sin embargo, con los descriptores SURF y ORB los cálculos presentaron errores con los puntos de interés o no se pudieron calcular los descriptores asociados a los puntos de interés extraídos. Esto se explica dado que el algoritmo de SURF hace cálculos basados en puntos de interés con una ventana de tamaño 20, por lo que el algoritmo tiende descartar puntos cercanos a los bordes de la imagen (contrario a lo que hace el algoritmo desarrollado en este trabajo). Con esto, se le induce un error al descriptor de SURF que no puede acceder a datos de la imagen. Para el caso de ORB, el descriptor nunca logró calcular el mismo número de descriptores para igual número de puntos de interés. Una explicación a esto es que el algoritmo de detección de ORB realiza un cálculo de orientación fino que no es proporcionado por el algoritmo propuesto.

b. Tiempo de cómputo de descriptores

Para cada imagen se realizaron 10 pruebas con cada descriptor, es decir, se realizaron 400 pruebas con el conjunto de imágenes y descriptores. El tiempo de cómputo que se selecciona es el menor tiempo obtenido por cada descriptor en cada una de las imágenes. La Tabla **3-3** muestra los resultados obtenidos por los descriptores durante las pruebas.

	Ajedrez	**Cruz**	**Mapa 1**	**Mapa 2**	**Mapa 3**	**Lena**	**Ventana 1**	**Ventana 2**
Técnica	**Tiempo (μsec)**	**Tiempo (μsec)**	**Tiempo (μsec)**	**Tiempo (μsec)**	**Tiempo (μsec)**	**Tiempo (μsec)**	**Tiempo (μsec)**	**Tiempo (μsec)**
Descriptor Cilíndrico	5248	15586	904	425	1808	219748	276314	123647
Descriptor Logarítmico	5012	14095	829	605	1050	197004	237953	119118
Descriptor SIFT	69677	19993	9176	5407	61673	130276	99989	117250
Descriptor SURF	20955	18810	8775	4804	41978	110375	189497	113541
Descriptor ORB	2002	5259	1164	315	8704	10737	11226	9470

Tabla 3-3.: Tiempo de cómputo en microsegundos asociado a cada una de las técnicas analizadas. El tiempo mostrado corresponde el tiempo mínimo de cómputo seleccionado de 10 iteraciones para cada algoritmo.

De la Tabla **3-3**, los resultados demuestran el buen desempeño del descriptor ORB frente a las demás técnicas. Esto refleja que en efecto, tal y como lo plantearon sus desarrolladores, ORB es un algoritmo equiparable y con menor tiempo de computo que los descriptores de SIFT y SURF. El descriptor de ORB se mostró como la técnica de mejor desempeño en los 3 grupos de imágenes a los que fue sometidos.

SIFT, tal y como está reportado en la literatura, es una de las aproximaciones de mejor desempeño cualitativo, pero con un tiempo de cómputo elevado debido a que éste realiza una serie de pasos para garantizar el mejor resultado en cada una de sus etapas. Por otro lado, se observa que SURF presenta una leve mejora respecto al tiempo de cómputo de SIFT.

Finalmente, los descriptores Cilíndricos y Logarítmico, pese a que desarrollan el mismo proceso de cálculo del descriptor cambiando solo una escala, es notable que el descriptor Logarítmico presenta un tiempo de cómputo menor que el reportado por el descriptor Cilíndrico en lo que respecta a los grupos de imágenes 1 y 3. Sin embargo, como se observa en el grupo de imágenes 2, que es el grupo objetivo de este trabajo, el descriptor cilíndrico presentó un mejor desempeño en el mapa2, mientras que para las imágenes mapa1 y mapa3 los tiempos de computo son equiparables, es decir, su diferencia es menor a 1 milisegundo. Ahora bien, en lo que respecta a estos dos descriptores respecto a las demás técnicas, presentan un mayor tiempo de cómputo, dado que son las únicas técnicas que calculan un descriptor bidimensional, de ahí que en todos sus cálculos requieran más tiempo que los demás descriptores que

son unidimensionales.

c. Selección de descriptor de características
Basado en los resultados cuantitativos de la sección anterior, donde se vislumbra el descriptor de ORB como una técnica aplicable para cualquier tipo de imagen, y donde se observa que el descriptor Cilíndrico tiene también un desempeño destacado para el grupo de imágenes objetivo de este trabajo, se procedió a seleccionar a este último descriptor como la técnica seleccionada para al Algoritmo de Mezcla de Mapas.

En primera medida, pese a que los tiempos de cómputo benefician al descriptor de ORB para la mayoría de imágenes de prueba, los tiempos de cálculo son menores para el descriptor Cilíndrico en el grupo de imágenes representadas por mapas de ocupación de celda (mapas generados por algoritmos de SLAM – grupo 2 de imágenes de prueba).

Como segundo punto, en la sección de Consideraciones se expuso el problema encontrado con el Detector de Característica seleccionado para el Algoritmo de Mezcla y los descriptores SURF y ORB. Por ende, una combinación del Detector Propuesto con el Descriptor de ORB no es posible, dadas las limitantes que surgieron cuando se intentó combinarlas: el descriptor no logra calcular para todos los puntos de interés un descriptor.

Finalmente, dada la limitante que tienen los mapas por ocupación de celda (son mapas en baja escala de grises), un descriptor como el calculado por SIFT, SURF u ORB no sería lo suficientemente distintivo como para evitar que en la etapa de correspondencia de puntos se presenten malos apareamientos. La razón principal se debe a que los mapas OGM tienen marcados cambios de gradiente, lo cual haría que los descriptores anteriores reflejen estos cambios, provocando que dos o más puntos se parezcan erróneamente. El problema detallado anteriormente no se presenta con el descriptor Cilíndrico, dado que éste calcula una huella o parche circundante al punto de interés. Es decir, es el único de los descriptores (además del Logarítmico) que refleja información precisa del vecindario del punto de interés.

3.3. Correspondencia de características

La correspondencia o apareamiento de características (matching en inglés) se encarga de la búsqueda y selección de la pareja correspondiente o más parecida de un punto característico en un conjunto de datos. Básicamente, el apareamiento de características desarrolla estrategias para comparar los puntos característicos de dos o más conjuntos de datos en aras de poder encontrar una transformación que permita a dichos conjuntos de datos mezclarse en un solo.

En esta etapa, por lo general los algoritmos que implementan descriptores propios tienden a proponer una técnica de apareamiento propia, dado que el tamaño específico de los descriptores hace única la estrategia de aparear puntos. Por ejemplo, un descriptor de SIFT de 128 datos no puede mezclarse con un descriptor de SURF de 64 o con el descriptor cilíndrico. Sin embargo, en el framework OpenCV se listan dos colecciones o conjuntos de algoritmos que recogen las técnicas de apareamiento más comunes y optimizadas para descriptores conocidos como SIFT, SURF, ORB, entre otros. El algoritmo más conocido es FLANN (Fast Approximate Nearest Neighbor Search Library), propuesto por David Lowe en 2009 [34]. Por otro parte, existe el algoritmo Bruce-Force Matcher el cual es una versión básica de FLANN, dado que realiza comparaciones punto a punto entre los conjuntos de datos. Esta técnica compara un punto con todos sus posibles correspondientes y solo toma el de menor distancia para el punto tomado como referencia.

Ahora bien, para el caso del algoritmo de mezcla desarrollado, los algoritmos FLANN y Bruce-Force Matcher no fueron programados y ni evaluados, dado que estas dos técnicas no son aplicables a descriptores bi-dimensionales como lo es el descriptor cilíndrico. Dado que este descriptor fue el seleccionado, la técnica de apareamiento seleccionada es un algoritmo clásico de comparación de descriptores. Adicionalmente, y para contrastar esta técnica versus otros algoritmos de correspondencia, se implementaron algoritmos de clustering de puntos característicos y distancia entre puntos. Es por esto que en esta sección, se describen las técnicas anteriormente mencionadas para su estudio y análisis. Finalmente, la sección culmina con la selección de la técnica de apareamiento idónea para el algoritmo de mezcla de mapas SLAM desarrollado.

3.3.1. Correspondencia de características basada en distancia entre puntos

Es la técnica apareamiento más simple de las conocidas. Básicamente establece relaciones espaciales entre dos puntos característicos pertenecientes a un mismo conjunto de datos. Como restricciones, esta distancia solo puede ser calculada y comparada entre conjuntos de datos invariantes a escala, rotación, traslación y cizallamiento (punto de vista). Ahora bien, dada la naturaleza de cómo se adquieren las imágenes de mapas SLAM, se sabe que dichas imágenes no tienen cambios de escala, resolución ni cizallamiento (puntos de vista), por cuenta de la información contenida en los mapas representa el estado de un ambiente estático real que no sufre de elongamientos, ni cambios estructurales a lo largo de un determinado tiempo. Por ende, esta simple técnica es aplicable al caso de estudio en este trabajo.

La técnica inicia con el cálculo de todas las posibles relaciones espaciales que se puedan tener entre los puntos característicos de un conjunto de datos. Sea n el número de puntos característicos de una imagen A, las distancias calculadas entre puntos estarán determinadas

por: sea el punto P_1, perteneciente a la imagen A,

$$\forall i \in n : \ {}^{A}d_{1i} = \sqrt{(P_i(x,y) - P_1(x,y))^2}. \tag{3-8}$$

Una vez determinadas las distancias para un conjunto de puntos característicos, se realiza el mismo procedimiento para las demás imágenes en análisis. Ahora bien, dos características serán correspondientes si,

$$\sqrt{(\ {}^{A}d_{ik} - \ {}^{B}d_{jl})^2} < umbral, \tag{3-9}$$

donde ${}^{A}d_{ik}$ es la distancia entre los puntos i y k de la imagen A, mientras que ${}^{B}d_{jl}$ es la distancia entre los puntos j y l de la imagen B. El umbral dependerá de tanto error se permita en los cálculos de las distancias. Para este trabajo, se determinó un error porcentual del 8 %, debido a que las distancias calculadas pueden sufrir variaciones dado que representan un mapa reconstruido por un algoritmo de SLAM el cual tiene un error probabilístico asociado a sus cálculos de alrededor de 4 cms. Por ende, un punto puede ser visto por dos agentes con un error máximo de 8cm entre las medidas reportadas por cada agente.

Ahora bien, el error establecido en el umbral influye directamente en la aceptación de falsas parejas de puntos, dado que dos o más distancias pueden cumplir con esta condición. De ahí la importancia de conocer el problema que se aborda y de seleccionar el mejor parámetro que permita seleccionar correctamente la mayor cantidad de puntos correspondientes.

3.3.2. Correspondencia de características basada en Clustering

Esta técnica fundamentalmente busca formar grupos o clústeres de características basado en una condición o conjunto de reglas, de tal forma que cada clúster tenga una identidad única que permita aparearlos con otros clústeres.

Las reglas relacionadas para formar clústeres van desde establecer un radio sobre un punto característico hasta establecer relaciones matemáticas entre los puntos para determinar la pertenencia o no a un grupo. Si se selecciona un radio de cobertura r_{clus}, se establece un algoritmo de búsqueda que agregue puntos característicos que estén por debajo de este radio, es decir,

$$\sqrt{(x_1 - x_c)^2 + (y_1 - y_c)^2} < r_{clus}, \tag{3-10}$$

donde (x_c, y_c) son las coordenadas del punto central del clúster y (x_1, y_1) corresponde a las coordenadas de un punto característico perteneciente al mismo conjunto de datos del punto (x_c, y_c). Ahora bien, una vez se han calculado todos los posibles grupos, se procede a aparearlos comparando al información contenida en cada uno de ellos. Como se tiene un

descriptor asociado a cada punto característico, la comparación entre grupos puede arrojar que tan distantes o parecidos se encuentran ambos clústeres.

Sea $C_1(n)$ el clúster asociado a la imagen A y $C_2(m)$ el clúster asociado a la imagen B, donde n y m corresponden al número de puntos característicos asociados a cada clúster. Se tiene que la distancia d que representa la correspondencia entre los clústeres es:

$$d = \sum_{i=0}^{n} \sum_{j=0}^{m} \parallel C_2(j) - C_1(i) \parallel . \tag{3-11}$$

donde $C_2(j)$ corresponde al punto característico j con descriptor $Desc_B(j)$ en el conjunto B y $C_1(i)$ corresponde al punto característico i con descriptor $Desc_A(i)$ en el conjunto A. Si la distancia d está por debajo de un umbral programado, se considera una pareja de clústeres correspondientes, de otra manera se descarta. Sin embargo, por cuenta de lo distante que pueden resultar dos clústeres, se calculan todas las posibles distancias entre descriptores y se selecciona un porcentaje de las mismas, partiendo de la distancia mínima calculada.

Este estrategia de clustering, al comparar conjuntos de clústeres mediante descriptores, presenta un alto consumo de tiempo de cómputo (muy por encima de la técnica de apareamiento por comparación de descriptores), por cuenta de que realiza cálculos adicionales a la comparación implicita de los descriptores. Es decir, es una extensión de la técnica de apareamiento de comparación de descriptores. Es por esto que para este trabajo, se seleccionó una técnica de clúster basada en la distancia euclidiana entre puntos característicos, es decir, se establecen relaciones espaciales entre los puntos característicos, ya que éstas permanecen invariantes a rotación y traslación.

La condición utilizada para la variante de clustering utilizada, selecciona los dos puntos característicos P_1 y P_2 más cercanos a un punto analizado P_C. Una vez se tienen esos puntos, se calculan las distancias entre los puntos que forman un clúster:

$$d_{C1} = \sqrt{(P_1(x,y) - P_C(x,y))^2} \tag{3-12}$$

$$d_{C2} = \sqrt{(P_2(x,y) - P_C(x,y))^2} \tag{3-13}$$

$$d_{12} = \sqrt{(P_2(x,y) - P_1(x,y))^2} \tag{3-14}$$

Estas tres distancias permanecen invariantes por lo tanto forman un clúster distintivo, que tiene como característica fundamental no tener un radio r_{clus} predeterminado. Adicionalmente, los grupos formados con este variante no excluyen que un punto solo pueda pertenecer a un clúster. De hecho, un punto puede pertenecer a máximo 3 grupos por cuenta de las relaciones espaciales d_{C1}, d_{C2} y d_{12} establecidas.

3.3.3. Correspondencia de características basada en comparación de descriptores

Esta técnica de apareamiento entre puntos característicos basa su desempeño en la comparación punto a punto entre descriptores. Es útil para realizar apareamientos o correspondencias entre conjuntos de datos cuyos puntos característicos o descriptores son desarrollos únicos y no compatibles con las técnicas de correspondencia comúnmente conocidas. Básicamente, esta técnica compara un descriptor del conjunto A con todos los descriptores del conjunto B, por ejemplo, y establece una métrica sobre la cual se seleccionarán las parejas correspondientes que permitirán que ambos conjuntos de datos puedan unirse.

El tiempo de cómputo es elevado en comparación con las demás técnicas de apareamiento (FLANN, Bruce-Force Matcher, entre otros), por cuenta de que hace cálculos sobre todos los datos de todos los descriptores. En términos generales, este enfoque es similar al algoritmo Bruce-Force Matcher, salvo que este último está optimizado en su versión oficial de OpenCV. Caso contrario con el algoritmo basada en comparaciones de descriptores, donde primero se analizan todas las posibles combinaciones y luego se selecciona un conjunto determinado de las mismas.

Por otra parte, esta técnica tiene dos variantes. La primera utiliza una función de maximización donde se analizará que tan parecidos son descriptores. En esta variante, el apareamiento ideal se da cuando los conjuntos de datos son totalmente iguales, por ende, la función que calcula su igualdad es igual 1, en una escala normalizada. La selección de las parejas se hace teniendo en cuenta un umbral como condición de rechazo (si la distancia es menor o igual a este umbral) o aceptación (si la distancia es mayor a este umbral).

Por otra lado, el segundo enfoque usa una función de minimización que establece que una pareja de puntos es correspondiente si la distancia o diferencia entre sus descriptores es tendiente a 0, donde una distancia igual 0 equivale a que dos descriptores son iguales, y por ende los puntos son idealmente correspondientes. Al igual que la primera variante, una vez se tiene una distancia entre descriptores, está se somete a un umbral el cual hace las veces de condición de aceptación (si la distancia es menor a este umbral) o rechazo (si la distancia es mayor o igual a este umbral).

3.3.4. Análisis y selección de algoritmo de apareamiento

Previo a la selección de la técnica de apareamiento idónea, se seleccionó un conjunto de tres imágenes (Figura **3-16**) que muestran diferentes mapas reconstruidos por diferentes agentes en diferentes tiempos. Las pruebas que se realizaron fueron: apareamiento entre $Img1$ e $Img2$ y apareamiento entre $Img1$ e $Img3$. La finalidad con estas imágenes es mostrar que

tan robusta es cada una de las técnicas analizadas, en aras de seleccionar la que mejor se adapte al problema de mezcla de mapas SLAM.

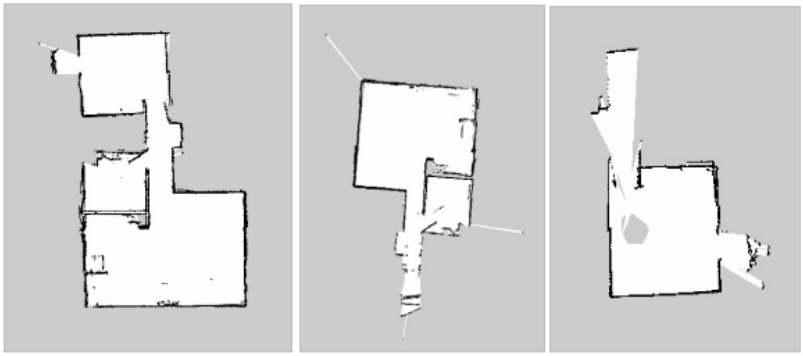

Figura 3-16.: Mapas SLAM de prueba. La imagen izquierda es denominada $Img1$, la imagen central es denominada $Img2$ y la imagen derecha es denominada $Img3$.

Una vez seleccionadas las imágenes se tomaron muestras de tiempo y se contó, manualmente, el número de parejas correspondientes para determinar cuál técnica presenta mejor desempeño. Adicionalmente, es de notar que cada secuencia de apareamiento fue probada en oportunidades en aras de seleccionar el tiempo mínimo registrado por cada algoritmo y para probar la estabilidad de los mismos. En lo que respecta a la estabilidad, todas las técnicas presentaron el desempeño esperado, por ende este análisis se omite teniendo en cuenta que todas las técnicas son estables y realizan el mismo cálculo, aun cuando las imágenes sufrieron rotaciones de 180°.

a. Algoritmo basado en distancias

Para la técnica basada en distancias, los resultados que se obtuvieron se muestran a continuación:

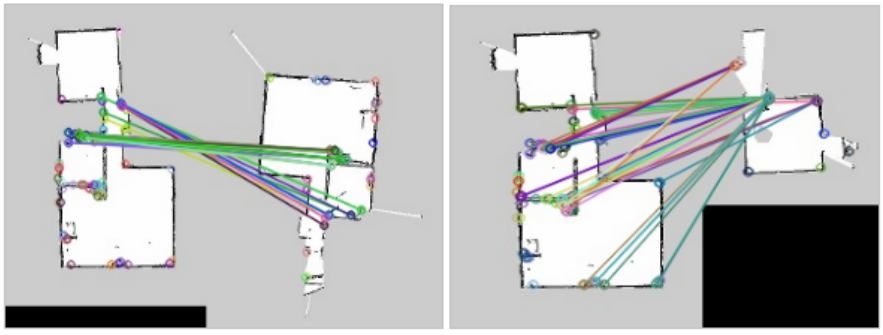

Figura 3-17.: Resultados de apareamiento basado en Distancias entre $Img1$ e $Img2$ (imagen izquierda), y $Img1$ e $Img3$(imagen derecha).

En la Figura **3-17** se nota como el algoritmo encuentra pareja en pares de puntos que realmente no son correspondientes. Este resultado no esperado, tiene una explicación si se analiza detalladamente el funcionamiento del algoritmo. Dado que se obtiene información en un plano de un ambiente, muchos de los espacios característicos tienden a ser similares

en un plano por cuenta de la falta de información de la dimensión z (si hablamos que el marco del mapa y de la imagen está en el plano x-y). Esta es una limitante a tener en cuenta, dado que los espacios empiezan a ser muy parecidos en el plano. Por ejemplo, una puerta puede tener puntos característicos en sus extremos. Este objeto se ve representado como una línea en el plano del mapa SLAM y como las puertas tienden a tener un ancho estándar, se ven como puntos distanciados iguales, pero en espacios diferentes. Si se tuviera información de altura, se podría saber qué tan altas son las puertas o si se tuviera información RGB se podría analizar el color y por ende, saber que se tratan de puertas diferentes.

b. Algoritmo basado en Clustering

Para la técnica basada en clustering, la Figura **3-18** muestra los grupos formados en cada imagen.

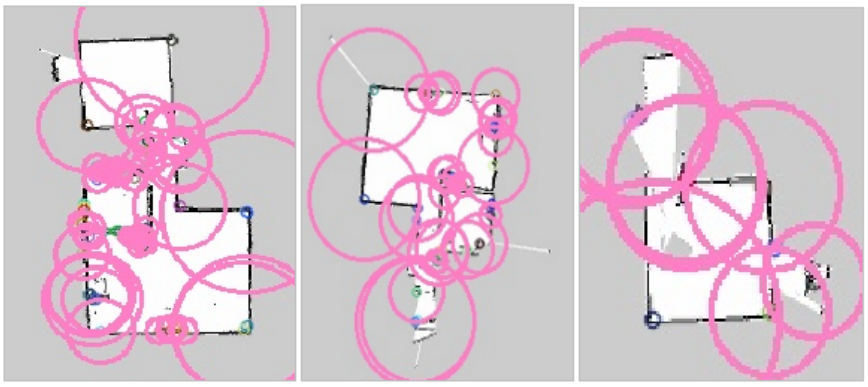

Figura 3-18.: Algoritmo de Clustering aplicado a imagen $Img1$ (izquierda), imagen $Img2$ (centro) e imagen $Img3$ (derecha).

En la Figura **3-19** se muestran los resultados de apareamiento para las 2 pruebas realizadas.

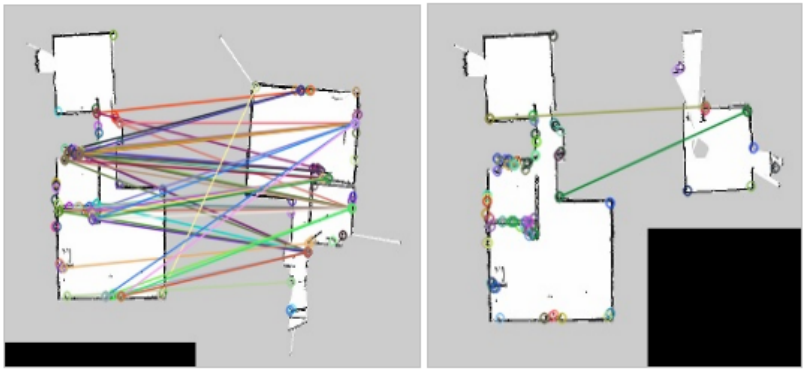

Figura 3-19.: Resultados de apareamiento basado en Clustering entre $Img1$ e $Img2$ (izquierda) y $Img1$ e $Img3$ (derecha).

De las figuras **3-18** y **3-19** se puede decir que esta técnica tuvo un pobre desempeño. Salta a la vista el hecho de los clústeres formados poseen pocas correspondencias. Sin embargo,

por el umbral establecido, es muy probable que el algoritmo haya encontrado como correspondientes las parejas mostradas en las anteriores figuras. Este comportamiento se explica dado el principio de funcionamiento del clúster seleccionado. Al crear clústeres de 3 puntos, el algoritmo calcula las distancias de las ecuaciones 3-12, 3-13 y 3-14. Sin embargo, estas distancias pueden sufrir del problema encontrado en la técnica basada en distancias, aunque ésta (clustering) sea más restrictiva al contar con 3 dimensiones. Además en diferentes pruebas realizadas, se encontró que esta técnica sufre con el estado del mapa SLAM determinado en un instante de tiempo, dado que los clústeres se recalculan con cada actualización del mapa y por ende, los clústeres cambian. Esto hace que sea imposible tener unos clústeres invariantes en el tiempo, lo que hace que sea poco probable que el algoritmo converja a la respuesta correcta.

c. Algoritmo basado en comparación de descriptores

Para la técnica basada en distancias, los resultados que se obtuvieron se muestran a continuación:

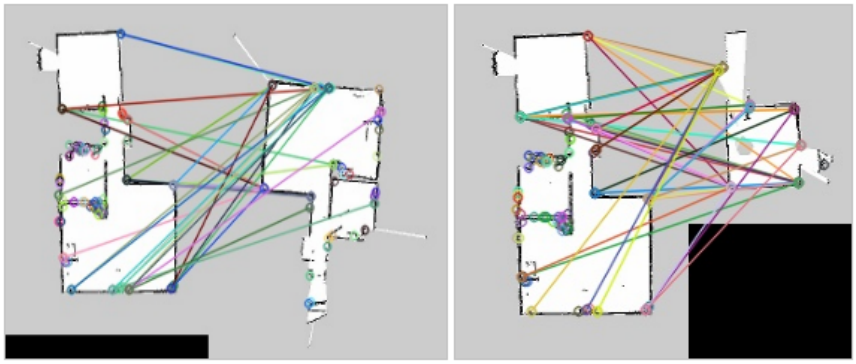

Figura 3-20.: Resultados de apareamiento basado en comparación de descriptores entre $Img1$ e $Img2$ (izquierda) y $Img1$ e $Img3$ (derecha).

En la Figuras **3-20**, se nota que esta técnica logra encontrar puntos correspondientes, pese a que también sufre de los malos apareamientos. La razón principal para los malos apareamientos sigue siendo la poca información del entorno contenida en el mapa, ya que es un plano. Sin embargo, también se nota que dentro de los resultados obtenidos, hay parejas que si corresponden entre sí. Esto se debe fundamental a que la huella calculada por el descriptor, pese a tener ruido, coincide. Adicionalmente, dado que el algoritmo detector de puntos calcula una orientación con resolución angular entre 22.5°y 45°, el error en el desfase de los descriptores incide directamente sobre la distancia entre los mismos. Ahora bien, en pruebas desarrolladas con otro conjunto de imágenes se encontró en que más del 85 % de apareamientos entre diferentes imágenes, esta técnica logró encontrar al menos 2 parejas correspondientes que sirven como puntos de referencia para mezclar las dos imágenes.

d. Tiempo de cómputo de las técnicas de apareamiento

La Tabla **3-4** muestra los tiempos de cómputos obtenidos para cada una de las técnicas analizadas.

	Apareamientos	
	$Img1$-$Img2$	$Img1$-$Img3$
Técnica	**Tiempo (μsec)**	**Tiempo (μsec)**
Apareamiento Distancias	3044	1802
Apareamiento Clustering	1530	989
Apareamiento Comp. Descriptores	208145	85120

Tabla 3-4.: Tiempo de cómputo asociado a cada técnica en cada una de las pruebas de apareamiento realizadas. Para cada prueba, cada algoritmo fue probado 10 veces para cuantificar su tiempo de cómputo mínimo.

Como se puede notar, la técnica de mejor desempeño es la clustering, dado que es la que menos cálculos realizas entre todas las técnicas. Por otra parte, la técnica que mostró el desempeño más lento es la de apareamiento basado en comparación de descriptores. Este resultado era de esperarse, dado que es la única técnica que realiza una comparación dato por dato entre dos descriptores.

e. Selección de Técnica de Apareamiento

Basado en los resultados expuestos en las secciones anteriores, es de notar que la única técnica capaz de encontrar 2 o más puntos correspondientes es la técnica basada en la comparación de descriptores. Por ende, y obviando los resultados en tiempo de cómputo, se selecciona la técnica de apareamiento basada en la comparación de descriptores, aún cuando tenga el tiempo de cómputo más elevado.

Su elección se fundamenta en que es la única de las técnicas capaz de aparear de manera correcta puntos característicos de conjuntos de datos diferentes. Pese a que también selecciona parejas erróneamente, mostró un alto desempeño frente a las demás técnicas. Ahora bien, es de notar que para mejorar los resultados de esta etapa de apareamiento, se hace necesaria una etapa adicional de cálculo de la transformada rígida y de alineación entre los conjuntos de datos. Estas 2 últimas etapas, permitirán obtener la transformación lineal rígida adecuada para mezclar dos mapas SLAM. Estos algoritmos se detallan en la Sección 4.1, donde se detalla el proceso de depuración de las parejas mal apareadas y la selección de las que sí corresponden adecuadamente.

4. Algoritmo de obtención de mapa global y control de robots móviles aplicado a MR-SLAM

Para obtener un mapa global construido a partir de mapas OGM, se hace necesario desarrollar, y complementar al algoritmo de mezcla descrito en la sección anterior, un algoritmo de alineación entre mapas que permita encontrar una transformada rígida consistente. Una vez se determinan todas las transformaciones rígidas necesarias para unir un número determinado de mapas parciales, se construye un árbol de tramas o sistemas referenciales para conocer las posiciones de un robot respecto a otro. De esta manera, el algoritmo de mezcla aplicado a MR-SLAM puede generar órdenes de control a cada robot en el ambiente de acuerdo a un algoritmo de toma de decisiones simples, basadas en la información proporcionada tanto por el mapa global como por cada agente.

En esta sección, se detalla el desarrollo del algoritmo de cálculo de la transformación rígida que permite unir a dos mapas, basado en la información proporcionada por el algoritmo de correspondencia, sección 4.1. Esta transformación es sometida a un procedimiento de refinamiento del ángulo de alineación para mejorar la mezcla de los mapas. Una vez se tiene la transformación, se empieza a construir un árbol de tramas encargado de proveer la trama global desde la cual se unen todos los mapas. Seguido a esto, en la sección 4.2 se detalla el algoritmo de toma de decisiones desarrollado en este trabajo para el control óptimo de los robots en el ambiente de trabajo.

4.1. Algoritmo de cálculo de tramas entre mapas locales para obtención de mapa global

Como cada mapa parcial posee su propio marco referencial, se hace necesario calcular una matriz de transformación rígida y consistente que permita unirlos. Dicha transformación rígida entre mapas debe ser considerada como una trama local adicional entre los mapas a mezclarse, cuyo origen se encuentra en los puntos obtenidos por el algoritmo de correspondencia. A medida que el algoritmo de mezcla encuentra correspondencias y transformaciones, las tramas locales entre mapas aumenta, de ahí surge la necesidad de construir un árbol de

tramas o sistemas referenciales que permitan, desde un único marco global, mezclar todos los mapas.

Adicional a la utilidad descrita anteriormente, el cálculo de la trama global permite que el marco móvil de cada mapa parcial, y que representa la posición del robot que está construyendo dicho mapa, sea conocida. Si desde un solo sistema referencial es posible conocer las posiciones de los robots, localmente un robot puede conocer las posiciones de sus compañeros, dado que el árbol de tramas puede proveerle de dicha información para, por ejemplo, ejecutar tareas cooperativas. Con esto, el algoritmo de toma de decisiones puede optimizar la reconstrucción del entorno, ya que puede generar trayectorias focalizadas hacia las zonas que deben ser reconstruidas y que vistas desde un mapa parcial no es posible.

El algoritmo de cálculo de la trama global parte de un algoritmo de alineación entre mapas basado en RANSAC (RANdom Sample Consensus). RANSAC provee una transformación rígida inicial que luego es sometida a un procedimiento de refinamiento. Dicha transformación es sometida un procedimiento de validación para aceptarla o no como una transformación consistente entre los mapas. Si la trama es aceptada, el algoritmo construye el árbol de tramas. A medida que se encuentra una nueva transformación, el algoritmo vuelve a realizar los procedimientos descritos anteriormente.

4.1.1. RANSAC

Ransac fue propuesto por Fischler y Bolles en 1981 [16] para el campo de análisis de imagen y cartografía automática. Es un método iterativo que calcula el modelo matemático de un conjunto de datos que pueden estar sujetos a cierta cantidad de ruido, el cual muchas veces es indeterminado. Introduce los conceptos de inliers y outliers. Los primeros corresponden a datos que cumplen con el modelo matemático puesto a prueba; mientras que los segundos corresponden a datos que no cumplen con dicho modelo, Figura **4-1**.

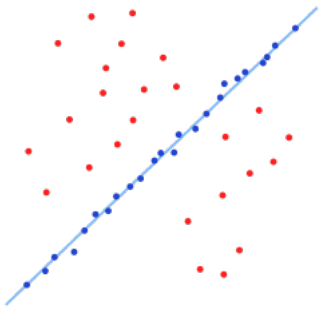

Figura 4-1.: Modelo matemático encontrado por Ransac dentro del conjunto de datos de prueba. La línea y puntos azules corresponden al modelo matemático hallado y a los inliers que lo satisfacen, respectivamente.

Ransac posee dos parámetros de configuración: n, que corresponde al número de iteraciones que debe realizar el algoritmo para intentar calcular un modelo matemático dentro del conjunto de datos analizados y m el conjunto mínimo de inliers para los cuales un modelo se considera adecuado. En [11] se detalla un estudio amplio y detallado de la configuración adecuada de dichos parámetros.

Ahora bien, para el problema en específico, el espacio de búsqueda entre mapas bidimensionales se reduce a búsquedas en x, y y θ. Estos 3 parámetros son los que Ransac intenta establecer y que sirven para determinar las tramas del árbol del sistema referencial global. Adicionalmente, para este problema en específico, el número de iteraciones n corresponde al número de parejas correspondientes entregadas por el algoritmo de correspondencia. Un modelo se considera válido si logra establecer al menos 3 inliers dentro de las parejas correspondientes.

Para una pareja correspondiente P_{M1} y P_{M2}, de los mapas M_1 y M_2, se cumple que:

$$P_{M2} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} P_{M1} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \tag{4-1}$$

donde $\theta = \theta_{PM2} - \theta_{PM1}$. Por lo tanto,

$$\begin{bmatrix} t_x \\ t_y \end{bmatrix} = P_{M2} - \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} P_{M1}. \tag{4-2}$$

Una vez calculados los parámetros de la transformación, se verifican si las demás parejas satisfacen el modelo (inliers) si,

$$\frac{P_{M1trans}}{P_{M2}} > 0{,}92 \tag{4-3}$$

De esta manera, si se encuentran 3 inliers para el modelo hallado, el algoritmo toma esa transformada y sigue iterando hasta encontrar un modelo mejor (que tenga más inliers del modelo encontrado anteriormente).

4.1.2. Refinamiento de transformación

Una vez Ransac obtiene un modelo válido, la transformación es sometida a un refinamiento con el fin de mejorar el desempeño del mezclado. Como se observa, el parámetro θ se obtiene la resta de los ángulos de cada punto característico de la pareja correspondiente. Estos ángulos tienen una resolución de 22.5°, resolución de la orientación asignada a los puntos característicos. Por lo tanto, el algoritmo de mezcla puede verse afectado, ya que puede existir una transformación con una rotación de menor resolución a la obtenida (22.5°). Para

ello, se hace uso de un algoritmo de refinamiento que maximiza la correspondencia entre los puntos cuando se someten a la Ecuación 4-3: un resultado de la división entre los parámetros $P_{M1trans}$ y P_{M2} igual 1 establece una transformación ideal entre puntos.

El algoritmo de refinamiento básicamente desarrolla un procedimiento parecido al de Ransac: itera sobre el ángulo θ en un rango de [-7°, 7°] y con una resolución de 0.5°, recalcula los parámetros t_x y t_y, y selecciona el que maximice la correspondencia de los inliers hallados por Ransac.

4.1.3. Validación de transformación

Para validar una transformación, se debe verificar sobre las parejas inliers y sus respectivos mapas, si la información circundante a estos puntos concuerda o no. Es decir, los puntos inliers de un mapa se transforman al espacio del otro mapa candidato a mezclarse. Se seleccionan un radio sobre cada uno de los puntos y se realizan test binarios para saber si la información de un mapa es igual a la del otro. Se acepta la transformación rígida encontrada si de los test realizados entre los mapas, más del 90 % son válidos. Es decir, si un punto $M_1(x_1, y_1)$ contiene la misma información que $M_2(x_{1trans}, y_{1trans})$.

4.1.4. Cálculo de trama global

Una vez se ha validado o aceptado la trama, se comienza a armar el árbol de tramas que contiene la trama global del algoritmo de mezcla. El número de transformaciones a encontrar corresponde al número de mapas parciales disponibles. El algoritmo de RANSAC itera hasta que el número de transformaciones necesarias se encuentra.
En el árbol de tramas, por cada transformación hallada entre mapas, se incluyen 3 transformaciones: una por cada mapa y la transformación entre mapas. Adicionalmente, para el cálculo de posiciones de los robots se incluye una cuarta, la trama de posición del robot, Figura **4-2**.

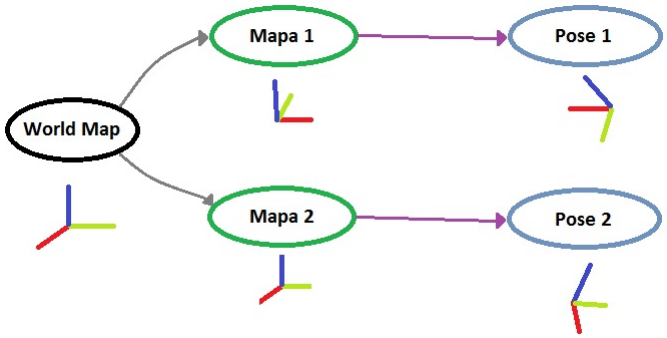

Figura 4-2.: Árbol de tramas entre mapas parciales para obtención de trama global. En esta figura, la trama World Map es la trama formada por la transformación rígida encontrada entre los mapas 1 y 2.

En total, el árbol de tramas contiene $4xO$ tramas, donde O es el número de mapas parciales.

4.1.5. Resultados de algoritmo de cálculo de tramas

Para mostrar el desempeño del algoritmo de RANSAC y de refinamiento, las figuras **4-3**, **4-4** y **4-5** muestran los resultados obtenidos con estos algoritmos.

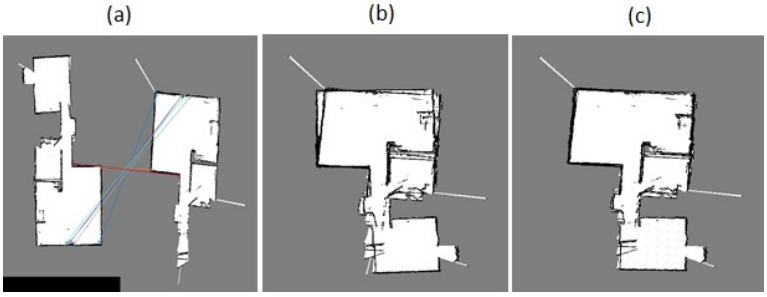

Figura 4-3.: Calculo de transformación rígida usando RANSAC (a) y el resultado de su mezcla (b), y el resultado obtenido por la etapa de refinamiento (c). En la figura a se observa como sobre un punto se cruzan todas las parejas correspondientes inliers.

Figura 4-4.: Calculo de transformación rígida usando RANSAC (a) y el resultado de su mezcla (b), y el resultado obtenido por la etapa de refinamiento (c). En la figura a se observa como sobre un punto se cruzan todas las parejas correspondientes inliers.

Figura 4-5.: Calculo de transformación rígida usando RANSAC (a) y el resultado de su mezcla (b), y el resultado obtenido por la etapa de refinamiento (c). En este resultado, se observa que ya no existe un punto o vértice común.

Como se puede ver, el algoritmo de RANSAC logra encontrar transformaciones entre los mapas, imagen a de las figuras **4-3**, **4-4** y **4-5**. El algoritmo de refinamiento muestra un mejoramiento en el cálculo de la transformada, imágenes $b - c$ de las figuras **4-3**, **4-4** y **4-5**, lo cual muestra que esta etapa era necesaria para mejorar el desempeño del algoritmo de mezcla.
Ahora bien, en la etapa de validación de los ejemplos mostrados en las figuras **4-3**, **4-4** y **4-5** determinaron que la mezcla de la imagen c de la Figura **4-5** no representaba una mezcla óptima, dado que aún cuando presenta un mejoramiento en la mezcla de los mapas, no logra ser lo suficientemente consistente. La imagen c de las figuras **4-3** y **4-4** presentan una correspondencia de 0,956 y 0,903, respectivamente, mientras que la correspondencia de la Figura **4-5** es de 0,867. Este parámetro de aceptación se estipuló en 0,9 (90 %), dado que al unir mapas se puede presentar información que aún no es conocida en uno de los dos mapas y, por ende, los test binarios realizados no permitirán alcanzar una correspondencia de 1 (100 %). Adicionalmente, como los mapas son obtenidos por algoritmos de SLAM, estos están sujetos a ruidos propios de la naturaleza de SLAM y por ende la correspondencia ideal del 100 % es poco probable que se obtenga. Ahora bien, durante pruebas realizadas, se encontró que el algoritmo logra encontrar transformaciones erradas con un valor cercano a 0,87. De ahí que se estableciera una frontera de 0,03 o 3 % por encima de dicho valor, dando como resultado un parámetro de 0,9, el cual como se evidencia en las figuras anteriores, muestra un resultado destacable.

4.2. Algoritmo de toma de decisiones para control de navegación de robot móviles

Para un adecuado control de los robots en operación, una vez el algoritmo de mezcla empieza a construir el mapa global de un entorno desconocido, se hace necesario optimizar la tarea de exploración y reconstrucción llevada a cabo por cada agente de forma independiente. Si un robot continua con su tarea de navegación autónoma sin tener en cuenta a los demás agentes, es muy probable que el objetivo del algoritmo de mezcla no se cumpla, dado que los robots terminarían con mapas casi idénticos y el tiempo de operación de los mismos no se vería reducido por cuenta de que no existe un algoritmo capaz de redirigirlos cuando dos o más agentes se dirigen hacía un mismo punto en el espacio. Es en este punto donde se hace necesario el desarrollo de un algoritmo de toma de decisiones que complemente el trabajo realizado por el algoritmo de mezcla.

Como su nombre lo indica, el algoritmo de toma de decisiones se encarga de analizar un conjunto de estados o procesos a los cuales tiene acceso para tomar un decisión ajustada y adecuada al estado actual del sistema. De esta manera, el sistema puede reaccionar y redirigir su estrategia hacia puntos o procesos que requieran de mayor prioridad o ajustes. Con esto, se

busca optimizar o mejorar el rendimiento general del sistema. El proceso toma de decisiones (o decision-making process, en inglés) requiere de una descripción de proceso y de objetos [49]. La descripción de procesos puede estar representada por modelos basados en grafos que incluyen desde arboles de decisión, redes de Petri, autómatas simples hasta cadenas o procesos de Markov. Por otra parte, los objetos pueden estar representados por modelos probabilísticos o difusos, como es el caso de SLAM y MR-SLAM. En ocasiones, la descripción de objetos esta implícita dentro de la descripción del proceso, dado que estas se encuentran interconectadas [41], por lo que en esos casos no se hace necesario hacer ambas descripciones.

En robótica, la toma de decisiones ha tenido amplia acogida en los robots de fútbol [1, 49]. Dado que en el terreno de juego se tiene a un conjunto de agentes, se hace necesario saber cómo manipularlos para cumplir un solo objetivo, marcar un gol en la portería del oponente. Para esto, el uso de árboles de decisión ayuda a que un agente sepa cómo actuar y qué objetivo cumplir en el terreno (patear el balón, defender su terreno, meter un gol) para que en su conjunto, el equipo obtenga el resultado esperado. Los arboles de decisión en esta aplicación de la robótica ayudan a la generación de un comportamiento emergente del equipo de robots en su conjunto, dado que el árbol de decisión le ayuda a cada agente a saber que tarea desarrollar en un instante de tiempo, sin necesidad de interferir con las tareas desarrolladas por sus compañeros. En la Figura **4-6** se muestra un árbol de decisión para el proceso de generar un pase del balón a un compañero del equipo.

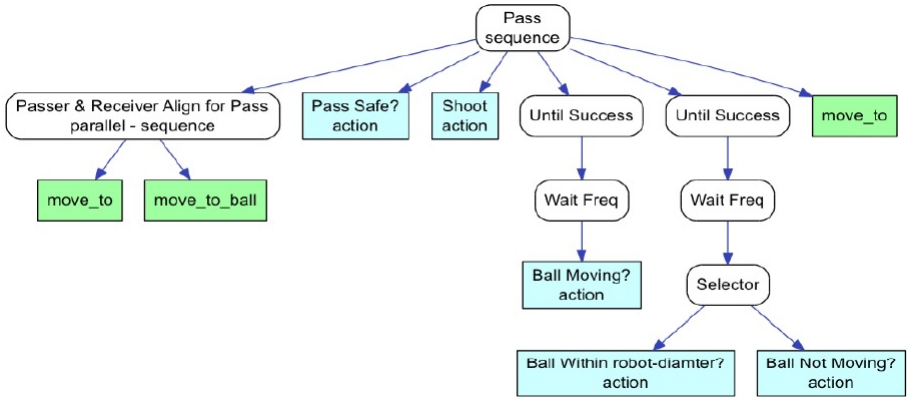

Figura 4-6.: Árbol de decisión para el proceso de pase del balón a un compañero. Tomado de [49].

En dicho árbol, se puede notar la inclusión de estados o procesos de éxito de haber pasado o no el balón a un compañero. Para un robot, el árbol de decisión mostrado en la Figura **4-6** corresponde un solo estado o proceso de un árbol de decisión mucho más extenso y grande, ya que este solo incluye el proceso de pasar un balón.

De lo anterior, al campo del MR-SLAM se puede extrapolar la idea de organizar a un conjunto

de robots para que trabajen en equipo y así conseguir un objetivo en común, obtener un mapa global de navegación de un entorno desconocido. En este caso, el uso de un árbol de decisiones ayuda al algoritmo central de supervisión (que también se encarga de la mezcla de los mapas) a direccionar o cambiar de rumbo a los robots una vez se han mezclado dos o más mapas parciales. Con este proceso de toma de decisiones, el algoritmo de mezcla aplicado a MR-SLAM se complementa para ofrecer una solución completa al problema del MR-SLAM referido en la Sección 2.

4.2.1. Parámetros de algoritmo de toma de decisiones

El árbol de decisión desarrollado para este trabajo requiere de unos parámetros o variables sobre los cuales se generan comandos de control de los robots. Una vez una transformada rígida o trama entre mapas parciales ha sido encontrada, el algoritmo toma una decisión basándose en los parámetros descritos a continuación:

- Distancia restante a objetivo actual (d_{rest}): esta distancia medida en metros provee la distancia restante para que el robot llegue al objetivo actual trazado. Esta distancia se calcula sobre la trayectoria que sigue el robot y se tiene en cuenta como parámetro principal, ya que una vez el robot llega a su objetivo, genera una nueva trayectoria.
- Estado de operación de robot (rob_{sta}): esta variable contiene el estado del robot durante su operación. Dentro de los estados considerados para el robot se cuentan: operando, pausado, detenido, fallo.
- Porcentaje mapa conocido (map_{kno}): como los mapas OGM poseen tres niveles de gris que representan los estados libre, ocupado y desconocido, este parámetro se encarga de llevar cuánto del mapa por conocer ha sido determinado por el robot. De esta manera, un valor cercano a 1 quiere decir que el mapa fue conocido o construido en su totalidad.
- Distancia recorrida (d_{trav}): este parámetro provee la distancia en metros que ha recorrido el robot a lo largo de su tarea de reconstrucción.

Con estos parámetros se garantiza que el árbol de decisión genere la orden adecuada para el control de los robots móviles en el entorno que se está reconstruyendo cooperativamente.

4.2.2. Algoritmo de Toma de Decisiones aplicado al problema de MR-SLAM

Con los parámetros descritos anteriormente, el algoritmo de toma de decisión se basa en el árbol mostrado en la Figura **4-7**.

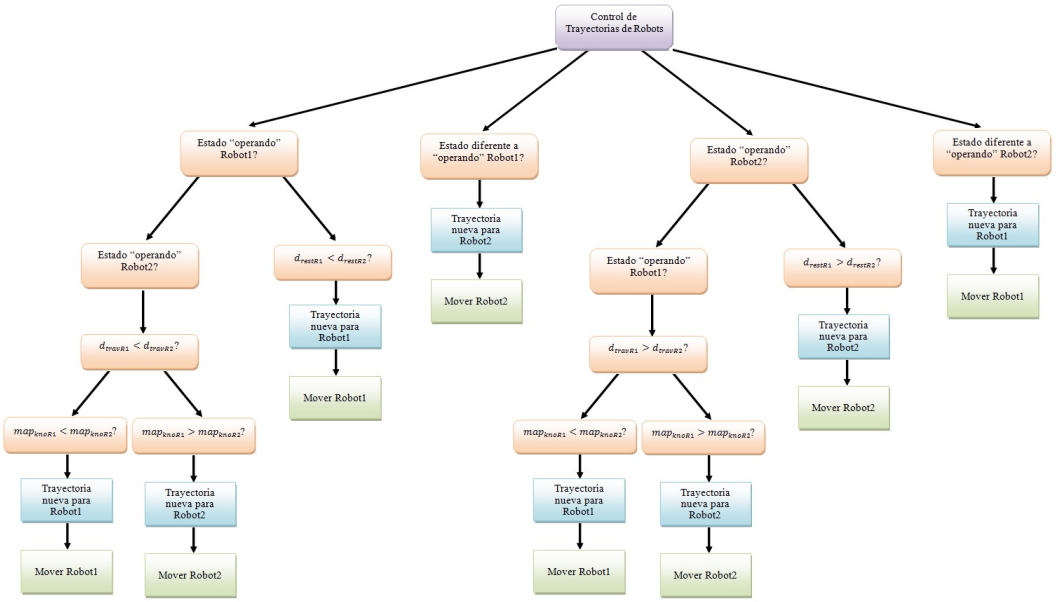

Figura 4-7.: Árbol de decisión del algoritmo de toma de decisiones desarrollado para el algoritmo de mezcla aplicado a MR-SLAM.

Como se puede notar, el parámetro de mayor importancia es rob_{sta}. Lo anterior se sustenta en que para que exista mezcla de mapas, el robot debe tener un estado diferente a fallo. Los demás estados proveen de mapas para el algoritmo de mezcla y cálculo de tramas (estado operando, pausado, detenido). Ahora bien, si un robot posee el estado detenido o pausado, no puede ser re direccionado, ya que no se tiene certeza de lo que podría pasar al robot con ese estado si se le ordena cambiar su rumbo. Para ambos estados, es necesario adquirir el tipo de acción que desencadenó dicho estado y si es seguro que el algoritmo de supervisión lo saque del mismo.

Seguido a lo anterior, el algoritmo de toma de decisiones analiza el parámetro d_{rest}. En el caso que dos robots se dirijan a un mismo punto, se prioriza el robot que esté a punto de llegar a su objetivo, mientras que el otro debe ser detenido para cambiarle su trayectoria hacia un nuevo punto. La anterior decisión se toma basado en el estado actual del robot (parámetro rob_{sta}). Este parámetro marca la pauta para optimizar el tiempo de reconstrucción cooperativa del entorno, ya que antes de que los robots finalicen con una tarea, el algoritmo de supervisión puede darles nuevas órdenes de movimiento a los agentes en el entorno.

Por otro lado, el algoritmo puede generar decisiones basados en los parámetros map_{kno} y d_{trav}. Estos dos parámetros ayudan al sistema a saber cuál robot acaba de iniciar su tarea

de reconstrucción y cuánto tiempo de operación lleva en el entorno. Si un robot tiene un mapa parcial conocido más grande que otro, el parámetro map_{kno} indicará que dicho robot está cerca de culminar con su tarea, mientras que al otro robot le hace falta mucho más por reconstruir. Por lo tanto, el algoritmo seleccionará al robot con menor indicador para desviarlo hacia una nueva ruta sobre la cual pueda mapear mucha más área y así tratar de igualar dichos parámetros. Ahora bien, dado que el problema de loop closure en SLAM puede afectar la lectura correcta de este parámetro, se complementa con el parámetro d_{trav}. Si un robot encuentra limites en su entorno (obstáculos) quedará encerrado y solo determinara una pequeña porción que será reflejada con un valor de map_{kno} bajo. El parámetro d_{trav} ayuda a saber cuánto tiempo de operación lleva un robot en el entorno y si la distancia viajada corresponde, proporcionalmente, con el mapa reconstruido: a mayor distancia viajada, mayor debería ser el mapa reconstruido. De esta manera, el algoritmo de toma de decisiones procura optimizar el tiempo de operación de los robots en un entorno desconocido.

4.2.3. Prueba de algoritmo de toma decisión

Para probar el funcionamiento del algoritmo de toma de decisión se llevaron a cabo pruebas con un robot móvil Kobuki en tareas de exploración individual y de movimientos simples del robot. Dicho robot cuenta con sensores de bumper, de acantillado, de caída de llanta y de proximidad. La idea de estas pruebas era inducir al algoritmo la generación de una decisión de acuerdo al estado del robot. Las pruebas buscaban verificar que todas las posibles decisiones que se pudieran tomar, se generaran de acuerdo al análisis de los 4 parámetros que tiene en cuenta el árbol de decisión.

Por un lado, en las pruebas mostraron que no es posible ni seguro agregar ordenes para sacar a los robots de su estado pausado o detenido, básicamente porque el primer estado puede desencadenarse por la detección de obstáculos cercanos al robot (accionamiento de sensores de bumper, acantilado o caída de llantas). En una prueba desarrollada, al robot se le cruza un obstáculo dinámico que acciona un sensor de bumper. Éste reacciona y detiene su tarea, por lo que decide poner su estado en pausado. Una vez se le induce al algoritmo de toma de decisión que ejecute una acción sobre el robot pausado, este le envía una orden de reanudar su tarea de exploración y le envía un nuevo objetivo. Al recibir dichas ordenes, el robot intentar generar una nueva trayectoria, pero no puede dado que el obstáculo sigue impidiéndole su movimiento. Como el algoritmo de SLAM discrimina obstáculos dinámicos y estáticos, al moverse remapea la zona donde se encuentra el obstáculo y lo incluye dentro de su mapa de navegación, condición indeseada dado que dicho obstáculo no hace parte del entorno. Con este evento, el algoritmo de generación de trayectorias pierde la posibilidad de generar rutas seguras para el robot, puesto que ha quedado dentro del radió de seguridad trazado para los obstáculos estáticos contenidos en el mapa. Por esta razón, se decide no introducir ordenes o decisiones cuando el robot se encuentre en estos estados, ya que puede

inducirse comportamientos indeseados al robot. Para incluirlas, se hace necesario que el robot exponga mucha más información de las razones que indujeron su estado. Por ejemplo, no es lo mismo para el robot pausar su movimiento por un objeto detectado por el láser a pausar su movimiento por accionamiento del sensor de bumper. El primero es un estado pausado que permite un margen de maniobra mucho mejor que el segundo, dado que el sensor de bumper implica contacto directo con un obstáculo en una zona ciega del láser. Sin embargo, por la información expuesta por el robot, ambas situaciones son informadas como un estado de pausa y no de fallo o detención.

Por otro lado, durante las pruebas desarrolladas, se encontró una nueva orden o decisión que puede tomar el algoritmo de toma de decisión. Cuando el algoritmo intenta buscar un nuevo objetivo para uno de los robots, recurre al algoritmo de generación de trayectorias con el mapa mezclado. Como este mapa es una actualización de los mapas parciales de los robots, pueden darse casos en los que no se trace trayectoria alguna dado que el mapa puede no tener fronteras de exploración, ya que la unión de mapas permitió completarlo en su totalidad y se considera que la tarea de reconstrucción ha terminado. Este comportamiento también se da cuando uno de los robots posee un objetivo de exploración y el algoritmo decide detener al otro robot para no mandarlos al mismo punto de exploración.

Con las pruebas desarrolladas al algoritmo, se encontró que el esquema o árbol seleccionado cumple con todas las expectativas esperadas para este trabajo. Adicionalmente, estas pruebas permitieron deducir que existe otra razón para utilizar algoritmos de mezcla y control de robots en MR-SLAM, la optimización de recursos energéticos de la red al tener la capacidad de detener adecuadamente a agentes. Este proceso, dentro del algoritmo de mezcla, se ejecuta con un periodo de 10 segundos, tiempo en el cual se actualizan los mapas mezclados y el algoritmo actualiza sus decisiones. Una extensión de los resultados y comportamientos encontrados con este algoritmo es mostrado en la sección 5 de Resultados, donde se muestra el caso donde la unión de dos mapas, completa en su totalidad la reconstrucción del entorno de prueba.

5. Resultados

Durante el proceso de diseño, desarrollo y programación del algoritmo de mezcla de mapas por ocupación de celdas (o mapas OGM) aplicado a MR-SLAM, se llevaron a cabo simulaciones y pruebas en tiempo real del algoritmo con entornos desconocidos simulados, en aras de depurar y validar el funcionamiento de cada una de las fases que componen el algoritmo obtenido. Es necesario aclarar también que la técnica MR-SLAM desarrollada se compone de dos grandes módulos: el algoritmo de mezcla de mapas y el algoritmo de control y toma de decisiones de los robots móviles.

En este capítulo, se detalla desde el banco de pruebas utilizado para las pruebas simuladas que se llevaron a cabo, hasta el sistema de navegación usado por cada robot móvil de prueba. En la Sección 5.1, se muestra el banco de mapas utilizados en las pruebas de simulación del algoritmo de mezcla. Como característica fundamental de este banco, se menciona la importancia de que los mapas utilizados son reconstrucciones semi estructuradas de un entorno real que fueron obtenidas con el algoritmo de SLAM HectorMapping [22]. En la Sección 5.2, se describen los resultados de las pruebas simuladas con el banco de mapas de la Sección 5.1. Las simulaciones mostradas en esta sección son ejecuciones sobre mapas obtenidos, es decir, se supuso que el algoritmo de SLAM culminó con su tarea y, por ende, el mapa obtenido no variará respecto al tiempo de ejecución de la exploración autónoma del robot.

Ahora bien, en la Sección 5.3 se exponen brevemente los escenarios de pruebas escogidos para validar el algoritmo de mezcla en tiempo real. Estos espacios cuentan con el área suficiente para que dos o más robots móviles autónomos puedan desarrollar sus tareas de exploración individualmente, mientras que el algoritmo de mezcla y control, monitorea los robots y ejecuta acciones en tiempo real. La Sección 5.4 detalla el sistema de navegación autónomo sobre el cual trabajará el algoritmo desarrollado. Esta sección es relevante ya que precisa cómo a partir del funcionamiento del robot, el algoritmo puede generar órdenes de movimientos sobre el agente para optimizar la tarea de exploración y reconstrucción del entorno de trabajo.

Sin embargo, al ser una implementación en tiempo real sobre un sistema previamente desarrollado, el algoritmo presenta unas limitantes propias del robot escogido, de los equipos con que trabaja el sistema y de los algoritmos de mezcla de mapas. Estos pormenores son descritos en la Sección 5.5, donde se precisan las consideraciones y limitaciones del algoritmo desarrollado, dadas las condiciones de operación sobre el sistema de robots móviles utilizado.

Finalmente, en las Secciones 5.6 y 5.7 se detallan, respectivamente, los resultados obtenidos de las pruebas realizadas y la eficiencia del sistema en cuanto a su funcionamiento en tiempo real. En estas dos secciones se amplían los resultados más destacados que se obtuvieron a lo largos de las pruebas desarrolladas y se analiza el desempeño del algoritmo en términos de precisión, estabilidad y eficiencia del mismo en su ejecución en tiempo real.

5.1. Banco de mapas de prueba

Dado que el algoritmo de mezcla desarrollado se enmarca en el análisis de mapas OGM generados por algoritmos SLAM 2D, las imágenes de prueba para este algoritmo contienen reconstrucciones en un plano de corte de los entornos reales. El sensor utilizado para la adquisición de estos mapas es un láser Hokuyo URG04LX-UG01. Para probar el desempeño del algoritmo de mezcla se realizaron 2 reconstrucciones completas del entorno mostrado en la Figura **5-1**, desde 3 puntos diferentes de partida.

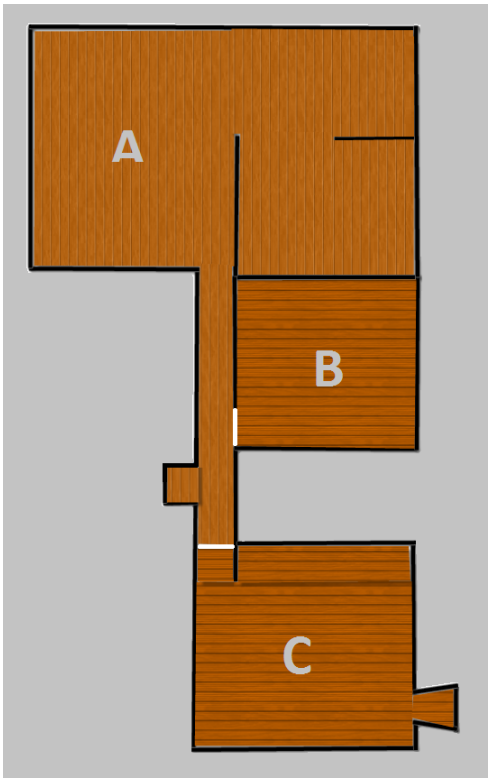

Figura 5-1.: Representación gráfica del entorno real utilizado con los puntos de partida A, B y C. El ambiente corresponde a un apartamento de $65m^2$.

Cada punto de partida representa la posición inicial de un robot en el entorno y por cada reconstrucción se tomaron 3 muestras de mapas a distintos tiempos de reconstrucción del entorno, dando como resultado la toma de 18 mapas de pruebas. Esta idea simula el hecho de

que los mapas están siendo aportados por un robot en un instante determinado. El tiempo de reconstrucción del entorno desde cada punto de inicio no es de interés en esta prueba, lo mismo que el tiempo en que fueron tomadas muestras en cada reconstrucción. La idea central es probar el desempeño del algoritmo de mezcla con mapas reales, sin importar el tiempo en que fueron adquiridas. En la vida real, no se puede garantizar que todos los robots partan al mismo tiempo desde sus posiciones de inicio, ya que un robot puede iniciar en cualquier momento. Cabe recordar que el algoritmo de mezcla y de control desarrollado no requiere de ninguna condición de inicio. Las figuras **5-2**, **5-3** y **5-4** muestran los mapas de prueba.

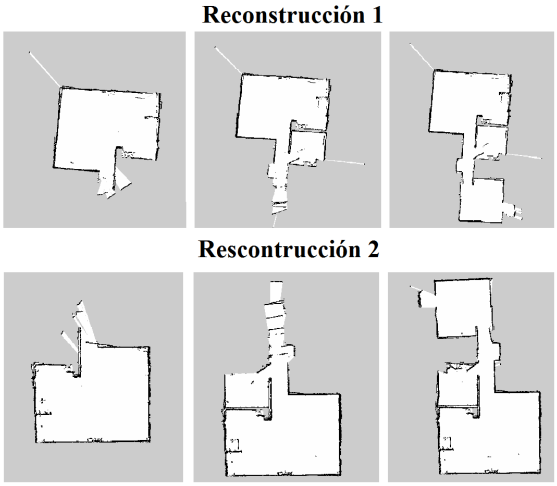

Figura 5-2.: Reconstrucción del entorno desde el punto de partida A.

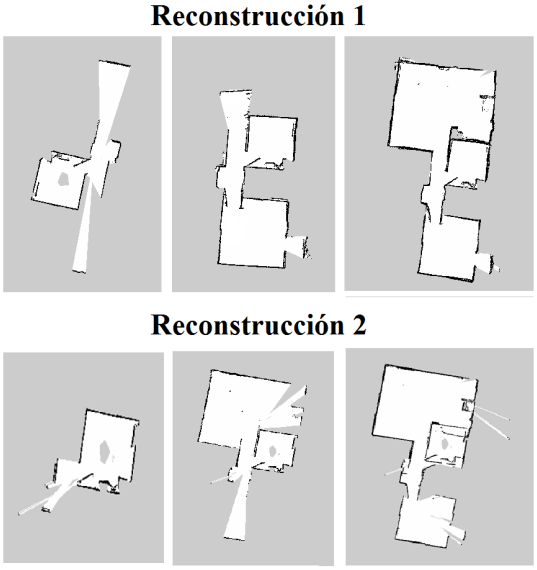

Figura 5-3.: Reconstrucción del entorno desde el punto de partida B.

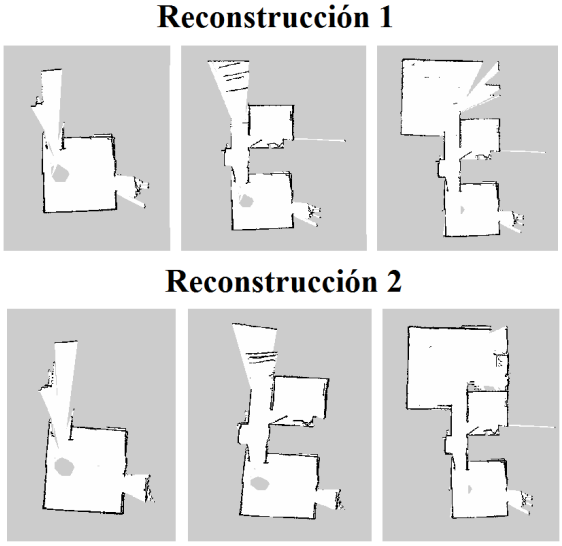

Figura 5-4.: Reconstrucción del entorno desde el punto de partida C.

Con estas imágenes, el algoritmo de mezcla desarrollado fue sometido a diversas pruebas para probar su desempeño. De las 18 reconstrucciones, se tomaron y analizaron las 6 imágenes más representativas del grupo de prueba para mostrar los resultados obtenidos con el algoritmo de mezcla desarrollado, Figura **5-5**.

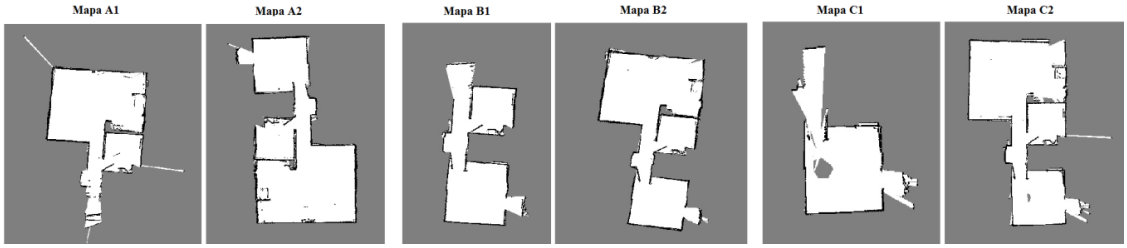

Figura 5-5.: Imágenes seleccionadas para la Sección 5.2. Se tomaron 2 imágenes por cada grupo mostrado en las figuras **5-2**, **5-3** y **5-4**

5.2. Simulaciones de algoritmo de mezclado de mapas

De acuerdo al algoritmo de mezcla y cálculo de transformadas, presentado en la Sección 3 y 4.1, a todos los mapas de pruebas mostrados en la sección anterior se les calcularon los puntos característicos, sus respectivos descriptores y tiempos de cómputo.

Las Figura **5-6** muestra los resultados para las imágenes de la Figura **5-5**.

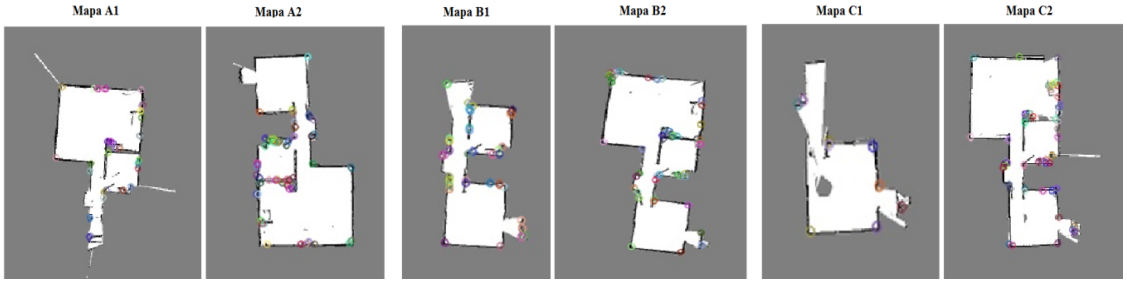

Figura 5-6.: Puntos característicos de cada imagen de prueba usada para la simulación del algoritmo de mezcla.

Con esta clase de imágenes (mapas OGM), el algoritmo de detección de esquinas desarrollado para este trabajo tiene un desempeño destacado, es decir, logra detectar las características de los mapas con gran precisión. Por la topología del entorno representada en esta clase de imágenes, las esquinas y paredes tienden a ser lo más representativo del mapa, aunque siempre se encuentren sometidas a ruidos en la captura, cálculo y proyección del entorno. Adicionalmente, por los estados representados en los píxeles de la imagen o mapa, el uso de un detector de mayor complejidad (Harris, SIFT, por ejemplo), no es necesario, dado que como se mostró y sustentó en la sección de 3.1, el algoritmo de detección desarrollado es el adecuado para este tipo de imágenes. La Tabla **5-1** muestra además los tiempos de cómputo asociados a cada imagen de prueba, resultados que sustentan y validan la elección del detector usado.

Resultados detección de puntos característicos		
	Tiempo (μsec)	**# Puntos**
Mapa A1	1731	31
Mapa A2	1523	60
Mapa B1	3101	37
Mapa B2	4852	49
Mapa C1	827	9
Mapa C2	2914	53

Tabla 5-1.: Puntos característicos y tiempo de cómputo para las 6 imágenes de prueba.

Sin embargo, es también necesario destacar el hecho de que a lo largo de los diferentes estados del mapa, los puntos característicos suelen no mantenerse. Lo anterior se sustenta en la naturaleza estocástica de los algoritmos de SLAM: en un paso de cómputo pone o quita datos y en el siguiente paso los elimina o refuerza, dado que el sensor puede haber entregado una información diferente. Frente a esta situación, se decidió permitirle al algoritmo calcular los puntos característicos solo del estado actual del mapa y no llevar un historial o cola que almacene todos los puntos característicos encontrados. Esto por el hecho de que los algoritmos de SLAM, por ejemplo, pueden poner objetos dinámicos dentro del mapa como

obstáculos estáticos y en pasos posteriores puede mejorar esa estimación y eliminar dicha información.

Ahora bien, en lo que respecta a los descriptores circulares hallados para cada punto característico, estos se calcularon con un radio de 25 píxeles y un paso angular de 1°. En la Tabla **5-2** presenta los resultados en tiempo de cómputo para cada uno de los 6 mapas que se muestran en la Figura **5-6**.

Resultados descripción de puntos característicos		
	Tiempo (μsec)	**# Descriptores**
Mapa A1	11171	31
Mapa A2	24335	60
Mapa B1	15320	37
Mapa B2	17891	49
Mapa C1	3004	9
Mapa C2	26048	53

Tabla 5-2.: Tiempo de cómputo de los descriptores asociados a los puntos característicos de cada una de las 6 imágenes de prueba.

La configuración seleccionada (radio 25 y paso angular de 1°) responde a un proceso de pruebas realizadas con la respectiva variación de dichos parámetros. Se encontró que para un radio mayor a 30 píxeles, el tiempo de cómputo crece considerablemente, poniendo en riesgo la implementación en tiempo real del algoritmo. Adicionalmente, el incremento de este parámetro no representa una diferencia significativa respecto al parámetro seleccionado en cuanto a discriminabilidad del descriptor. Por otra parte, radios menores a 18 píxeles provocan una aceleración del tiempo de cómputo, pero los descriptores tienden a no ser únicos, ya que la huella seleccionada no representa de manera adecuada el área circundante a un punto característico. Esto se traduce en una alta tasa de falsa correspondencia entre puntos característicos, razón por la cual está situación debe evitarse a toda costa, puesto que no se desea encontrar un mal emparejamiento que puede desencadenar una mala mezcla de mapas. El parámetro de paso angular tiene el mismo comportamiento que el radio del descriptor: a mayor paso angular, menor discriminabilidad tendrá el descriptor, mientras que a menor paso angular, se obtiene un descriptor único y característico, pero con un alto costo de cómputo.

Esta prueba finaliza con el algoritmo de correspondencia, cálculo y refinamiento de la transformada entre las imágenes de prueba. En la Tabla **5-3** se muestran los resultados asociados a esta etapa, donde se destacan 4 resultados que obtienen un porcentaje de solapamiento superior al 90 % (umbral seleccionado para la aceptación/rechazo de una transformada encontrada entre mapas). Las Figura **5-7** muestra las imágenes mezcladas de los casos encontrados. En total se realizaron 12 experimentos (cada mapa se comparó con el resto de mapas pertenecientes a otros robots).

Resultados apareamiento entre mapas de prueba					
Pareja 1	**Pareja 2**	**Parejas iniciales**	**Parejas RANSAC**	**% Solapamiento**	**Tiempo (μsec)**
Mapa B1	Mapa C1	10	0	0.000	27319
Mapa B1	Mapa C2	21	5	90.161	152583
Mapa B1	Mapa A1	15	3	77.420	94425
Mapa B1	Mapa A2	49	3	84.519	179721
Mapa C1	Mapa A1	27	7	55.841	21912
Mapa C1	Mapa A2	38	5	90.314	43468
Mapa C1	Mapa B2	42	4	73.741	33754
Mapa A1	Mapa B2	27	6	91.054	114585
Mapa A1	Mapa C2	35	5	82.417	125171
Mapa B2	Mapa C2	58	4	68.106	194303
Mapa B2	Mapa A2	126	6	74.941	193064
Mapa C2	Mapa A2	52	9	90.170	247168

Tabla 5-3.: Resultados del algoritmo de correspondencia y cálculo de la transformada. Se muestran los datos relacionados al número de parejas encontradas por el algoritmo de correspondencia, las parejas que encuentra RANSAC y la etapa de refinamiento, y finalmente el valor de aceptación/rechazo de la transformada encontrada.

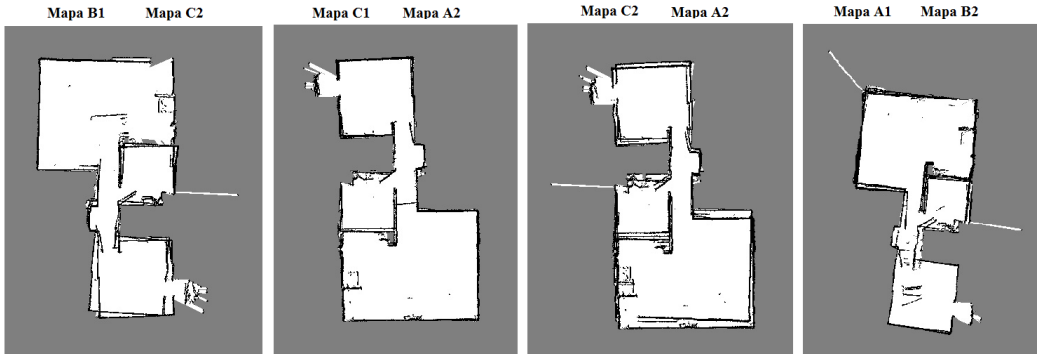

Figura 5-7.: Mezclas encontradas dentro de los 12 experimentos realizados para la prueba del algoritmo de mezcla.

Como se puede notar de la Tabla **5-3**, de las 12 comparaciones realizadas, el algoritmo encontró 4 transformadas válidas, las cuales involucran a por lo menos un mapa de cada robot (un mapa global donde todos los robots aportan información). Por otro lado, el algoritmo nunca realizó mezclas entre mapas que no presentaban áreas en común, lo que muestra que el algoritmo es robusto a la falsa aceptación de transformadas. Ahora bien, 2 de las 12 comparaciones no presentan solapamientos, mientras que el resto, 10 comparaciones, si posee zonas en común entre los mapas. Esto quiere decir que dentro de las 10 comparaciones que si presentan solapamientos, el algoritmo encontró 4, lo que se traduce en que el algoritmo tuvo una efectividad en 6 comparaciones de 12 posibles (4 transformadas que encontró más

las 2 comparaciones que no presentaban solapamientos y que fueron clasificadas de manera correcta). Las otras 6 comparaciones podrían ser consideradas como errores del algoritmo de mezcla. Sin embargo, un análisis a profundidad permite develar que la gran mayoría de falsos rechazos corresponden a imágenes que tienen el mapa completo o en un estado avanzado de exploración, mientras que de las 4 transformadas válidas, todas involucran un mapa en un estado inicial.

De los anteriores resultados, se puede mencionar también el hecho de que los falsos rechazos se dieron con mapas del entorno en un estado avanzado de exploración y no en etapas iniciales. Esto quiere decir que, al tener mucha más información, los mapas pueden contener mucho más ruido y, en consecuencia, el fallo en la mezcla puede verse afectado dado que la etapa de descripción es sensible a este ruido, al igual que la etapa de aceptación/rechazo de la transformada calculada.

Finalmente, se puede mencionar como resultado los tiempos de cómputo del algoritmo. Si se toman los tiempos de cómputo más grandes de cada etapa, el tiempo máximo de ejecución del algoritmo es de 280 milisegundos. Este tiempo máximo de cómputo se muestra como resultado destacado de la técnica MR-SLAM desarrollada en este trabajo en comparación con la técnica de *Blanco et al.* [7].

5.3. Escenario de pruebas

Para validar el funcionamiento del algoritmo de mezcla de mapas y control de robots, se desarrollaron 2 pruebas que demuestran el funcionamiento de los algoritmos diseñados. Estas pruebas fueron llevadas a cabo en 2 escenarios simulados (Figura **5-8**).

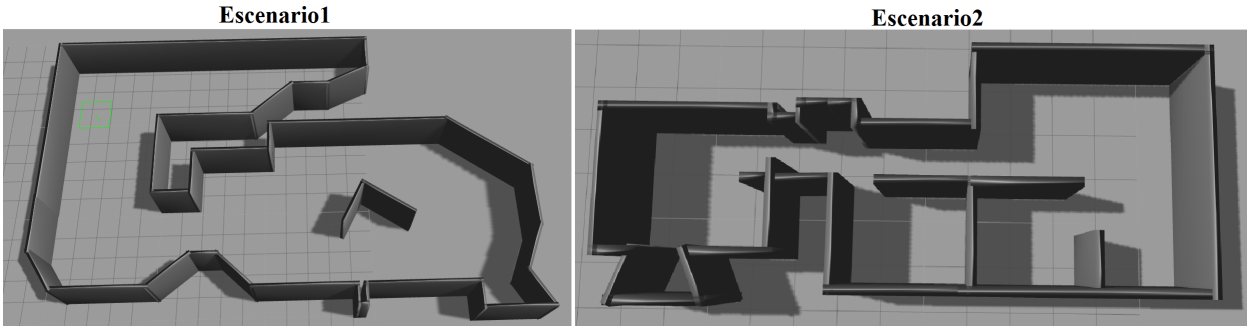

Figura 5-8.: Escenarios de pruebas para la validación en tiempo real del algoritmo de mezcla y control de robots.

Como características relevantes de estos ambientes, se menciona el hecho de que son entornos estructurados (escenarios conformados por figuras geométricas regulares), delimitados espacialmente (ambientes cerrados e internos) y con superficies mayores da $65m^2$, lo que

permite contener 2 o más robots móviles debidamente equipados. Lo que se buscó con estos escenarios fue demostrar que el algoritmo de mezcla funciona correctamente y bajo condiciones de inicio no controladas. Además, se buscó que los robots pudieran realizar un mapeo inicial individualmente y antes de mapear zonas comunes entre ellos, de ahí la importancia del tamaño escogido para los escenarios.

5.4. Sistema de navegación autónomo

Cada robot móvil está equipado de un sistema de navegación que consta de un algoritmo de SLAM (provee el mapa y pose del robot), uno de generación de trayectorias, uno de control de movimiento y seguimiento de trayectorias, así como una interfaz de operación para recibir órdenes y proveer información del estado del robot. El sistema fue desarrollado para poder validar la técnica MR-SLAM propuesta en este trabajo: para poder probar un algoritmo de mezcla aplicado a MR-SLAM, se necesita de un sistema de navegación apropiado que provea de la información requerida por éste.
El sistema de navegación fue programado bajo la plataforma de desarrollo ROS (Robot Operating System) en C++. En la actualidad se encuentra implementado en 3 robots móviles reales Kobuki. En la Figura **5-9** se muestra el esquema del sistema de navegación desarrollado y usado en este trabajo.

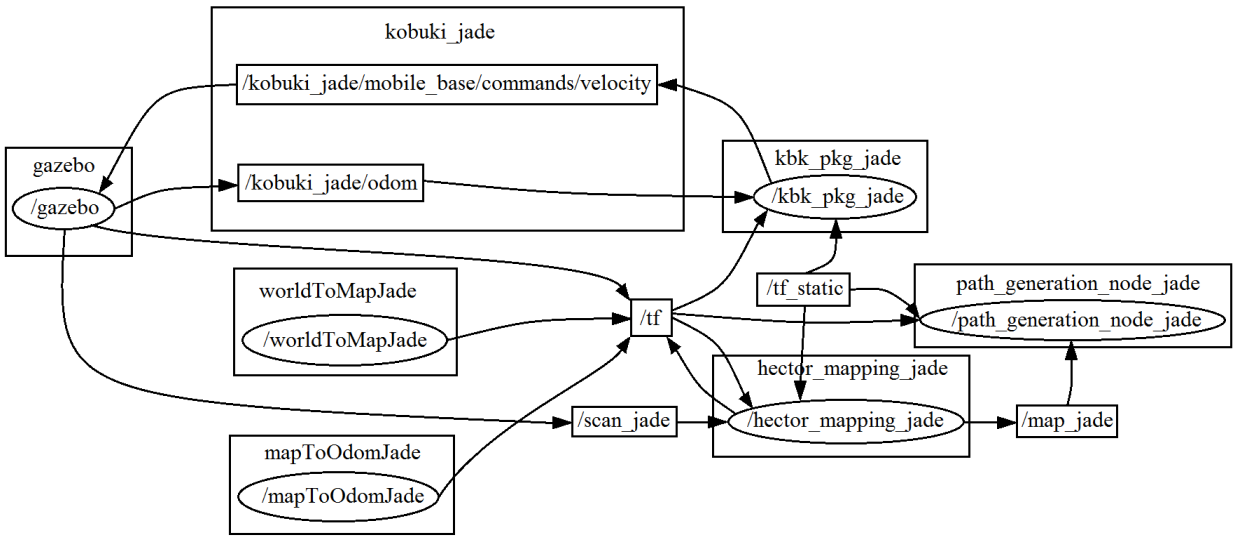

Figura 5-9.: Sistema de navegación autónomo desarrollado para robots móviles terrestres. En este esquema se muestra el sistema de navegación del robot Jade usado en este trabajo como robot de prueba.

En ROS, las soluciones o desarrollos se componen principalmente de topics y nodos. Un topic representa un canal de comunicaciones entre nodos con un tipo de mensaje y se nombran con

un "/", mientras que los nodos no poseen este carácter especial. Con lo anterior, en la Figura **5-9** se tienen como nodos principales *kobuki_jade*, *kbk_un_pkg*, *hector_mapping_jade*, *path_generation_jade* (el nombre jade corresponde al nombre de unos de los 3 robots usados en este trabajo: Jade, Luna y Mars). El primer nodo mencionado provee toda la interfaz con los sensores del robot Kobuki: encoders, IMU, infrarrojos, sensor de acantilado, entre otros. *kbk_un_pkg* provee los algoritmos necesarios para control de movimiento y seguimiento de trayectoria, así como recibir órdenes y exponer información del robot al usuario. Este nodo será el que reciba las órdenes generadas por el algoritmo de control y toma de decisiones. Las órdenes que este nodo puede recibir son: pausar, reanudar, detener, navegar autónomamente y navegar a un objetivo dado. *hector_mapping_jade* es el algoritmo de SLAM, que como se muestra en la imagen, recibe el topic */scan_jade* que le provee la información del láser y publica un topic llamado */map_jade* donde publica los resultados de SLAM. Por otro lado, como su nombre lo indica, *path_generation_jade* se encarga de generar las trayectorias que el robot debe seguir para explorar o navegar de forma segura y óptima en un entorno desconocido.

Cabe destacar que del sistema de navegación mostrado anteriormente, todos los algoritmos y nodos que lo conforman fueron desarrollados a excepción del algoritmo de SLAM que es propiedad del equipo Hector Team de la Universidad Técnica de Darmsdadt.

5.5. Condiciones y aspectos generales a la validación en tiempo real de algoritmo de mezcla de mapas

Previo al desarrollo de las pruebas de validación, se requiere contextualizar los aspectos más relevantes que interfieren en la realización de las pruebas. Los robots, los algoritmos desarrollados, los sensores láser usados y las limitaciones propias de la validación y algoritmos obtenidos, son los aspectos más relevantes para este trabajo.

Este trabajo se desarrolló en Ubuntu 14.04 LTS como sistema operativo base en un computador con procesador Intel i5 5200 @ 2.20Ghz x4, con 4 Gb RAM, 150 Gb disco duro. El lenguaje de programación usado fue C++, haciendo uso de la plataforma de desarrollo ROS Indigo y OpenCV 3.0. A continuación, se mencionan aspectos relevantes para tener en cuenta en este trabajo:

Robots móviles

Los robots móviles Kobuki son ampliamente conocidos en el mundo académico, aunque su versión comercial más conocida es TurtleBot. Las siguientes son las características más relevantes de estos equipos:

- Exactitud en posición: ± 15mm.
- Velocidad crucero: 33 cm/s.
- Velocidad máxima: 77 cm/s.
- Autonomía: 4.5 horas.
- Batería: 2400 mAh.
- Tamaño del robot: 36cm diámetro × 52cm alto.
- Radio de seguridad para evasión de obstáculos estáticos: 21 centímetros.
- Distancia detección de obstáculos: 20 centímetros.
- 3 sensores de accionamiento físico o bumpers: izquierdo, frontal, derecho.
- 3 sensores de acantilado: izquierdo, frontal, derecho.
- 3 sensores infrarrojos: izquierdo, frontal, derecho.
- IMU de 2DoF: orientación y velocidad angular sobre eje Z.
- 2 encoders incrementales de 1024 pulsos/revolución.
- 2 sensores de caída de llantas.

Sensor láser

Es el sensor que obtiene la información relevante del entorno. Realmente, este sensor delimita las características principales del sistema de navegación, del algoritmo de SLAM y, por ende, del algoritmo de mezcla de mapas.

Para este trabajo se usaron sensores Lidar Hokuyo URG-04LX UG01, cuyas características más esenciales son:

- Alcance máximo de detección: 5.6 metros.
- Alcance mínimo de detección: 0.2 metros.
- Frecuencia de escaneo: 10 hz.
- Precisión: ± 30 mm.
- Resolución: 1mm.
- Resolución angular: 0.33°.
- Interfaz de comunicación: USB 2.0

Sistema de navegación

El sistema de navegación, junto con el algoritmo de SLAM poseen las siguientes características que implícitamente permiten vislumbrar limitaciones sobre las cuales este trabajo se enmarcó:

- Alcance máximo de mapeo: 5.6 metros.
- Tamaño de mapa reconstruido: 169 metros cuadrados.
- Tamaño de mapa en celdas: 512×512.
- Resolución de mapa: 2.5 centímetros por cada celda del mapa.
- Reconstrucción de entorno/mapa en 2D.

- Operación en ambientes internos y semi estructurados.
- Frecuencia SLAM: 0.5 hz.
- Frecuencia control movimiento: 30 hz.

Modos de navegación:

- Navegación autónoma: exploración, generación de trayectorias, evasión de obstáculos, localización y mapeo autónomo.
- Navegación dirigida: envío y seguimiento de objetivos específicos, evasión de obstáculos y generación de trayectorias.

Algoritmo de mezcla

El algoritmo de mezcla desarrollado tiene como características principales:

- Análisis y control de 5 robots móviles en tiempo real.
- Mezcla de mapas OGM para ambientes semi estructurados.
- Tiempo de detección y cálculo de trama entre mapas: 300 ms.
- Frecuencia de ejecución de algoritmo: 1 hz.
- Algoritmo de mezcla y control centralizado de robots.

Limitaciones

Para validad el algoritmo de mezcla se desarrollaron pruebas simuladas y en tiempo real con mapas adquiridos de escenarios reales y simulados. Para las pruebas principales se seleccionaron 2 escenarios simulados en el software Gazebo, uno de los cuales es una representación del entorno real utilizado para las pruebas simuladas del algoritmo, sección 5.1. Este limitante surge dada la necesidad de requerir un entono lo suficientemente grande como para probar de forma adecuada el concepto de análisis, mezcla de mapas y control de robot móviles. Además, con el uso de un entorno real de este tamaño, se requiere de una infraestructura de comunicaciones inalámbrica robusta y que actualmente es inexistente en los laboratorios de la Universidad Nacional de Colombia, dado que el volumen de datos a transmitir por un agente es alto.

Sin embargo, este limitante no interfiere con la validación real de los algoritmos desarrollados, dado que del simulador la única información que se obtiene es la del láser Hokuyo, lo que quiere decir que los algoritmos del sistema de navegación y mezcla de mapas se ejecutan y funcionan en tiempo real: el algoritmo de SLAM está sujeto a ruido propio, la pose y control de movimiento está sujeta a deslizamientos o imprecisiones en el seguimiento de trayectoria, entre otras características comunes y normales a la ejecución de estos algoritmos. Adicionalmente, para mejorar aún más los datos del láser, se hizo uso de características propias del simulador Gazebo, el cual permite inducirle ruido a los datos que adquiere el láser simulado, es decir, al sensor láser Hokuyo se le introdujo un ruido gaussiano con desviación estándar de 0.3. Durante las pruebas desarrolladas, se encontró que este valor es el más cercano al ruido que se pudo observar en los sensores que se tienen en los robots móviles reales.

Por otra parte, al simular un sensor láser en un equipo de cómputo, la prueba del algoritmo de mezcla se delimitó al uso de 3 robots móviles, dado que los algoritmos del sistema de navegación consumen al rededor del 50 % de un procesador, por lo que el uso de más de 4 robots en una misma computadora hace imposible que se simule tanto la red colaborativa de robots, como el algoritmo de mezclado de mapas.

5.6. Resultados de pruebas de algoritmo de mezcla en tiempo real

De acuerdo a lo detallado en las anteriores secciones, se realizaron pruebas de validación del algoritmo de mezcla de mapas, así como del algoritmo de control de robots móviles (que incluye el módulo de toma de decisiones). En específico, se desarrollaron 3 pruebas para el escenario de la Figura **5-8** llamado *escenario*1 y 2 pruebas para el escenario de la misma figura llamado *escenario*2.

Las pruebas consistieron en una reconstrucción individual del entorno por parte de cada robot para medir la consistencia del algoritmo SLAM. Una segunda prueba consistió en usar 2 agentes en el entorno y verificar el funcionamiento del algoritmo de mezcla y su consistencia en la obtención de un mapa global de navegación. La tercera prueba consistió en usar los 3 robots en un mismo entorno para probar el desempeño del algoritmo de mezcla. Esta última prueba solo se realizó en el *escenario*1, dado que el tamaño de este era suficiente para albergar 3 robots navegando.

5.6.1. Prueba 1 – Reconstrucción individual

Los 3 robot móviles Jade, Luna y Mars fueron sometidos a 3 pruebas de reconstrucción para medir su desempeño en ambos escenarios.

Para el *escenario*1, los robots partieron de diferentes posiciones, tal y como se muestran en la Figura **5-10**, mientras que la Figura **5-11** muestra los resultados de los mapas obtenidos por cada robot en dicho ambiente.

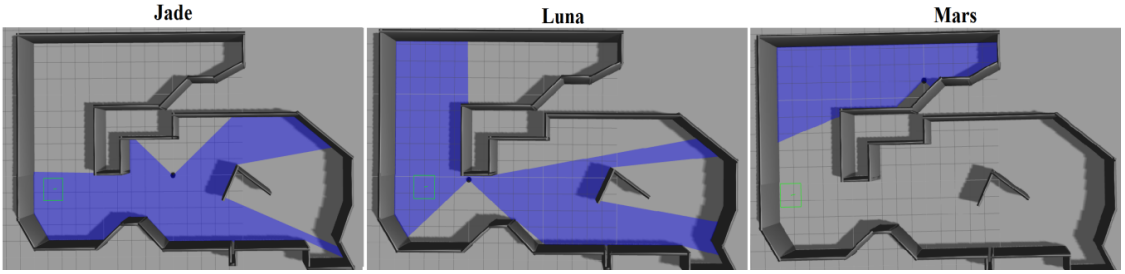

Figura 5-10.: Puntos de inicio para cada uno de los robots en el *escenario*1.

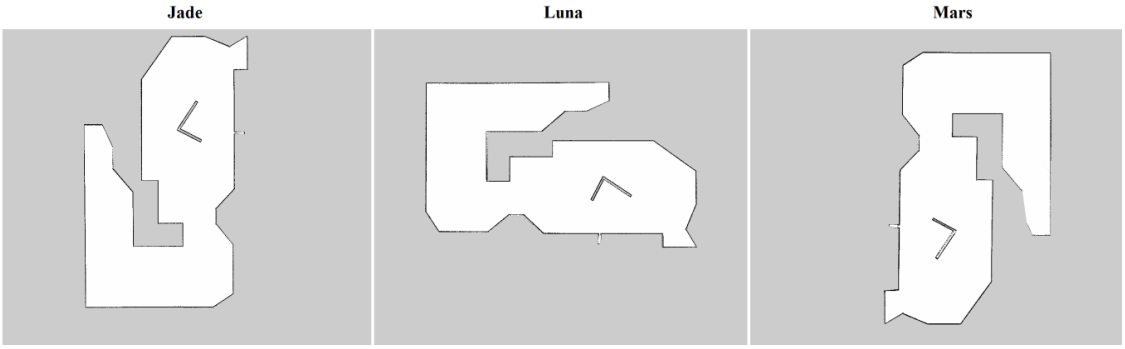

Figura 5-11.: Mapas obtenidos por el algoritmo de SLAM para cada uno de los robots de prueba en el *escenario*1.

Como se puede observar, los robots lograron de manera adecuada obtener un mapa consistente del *escenario*1. Durante las pruebas de exploración, los robots no presentaron problemas en sus recorridos y lograron culminar de manera exitosa con las 3 pruebas a las que fueron sometidos. Con esto, las pruebas del sistema de navegación dieron como resultado una efectividad del 100 % en tareas de reconstrucción de entornos desconocidos internos.

Ahora bien, para el *escenario*2 se realizaron las mismas pruebas desarrolladas con el *escenario*1 para conocer la efectividad del sistema de navegación usado. La Figura **5-12** muestra los puntos de partida para cada robot, mientras que la Figura **5-13** muestra los resultados.

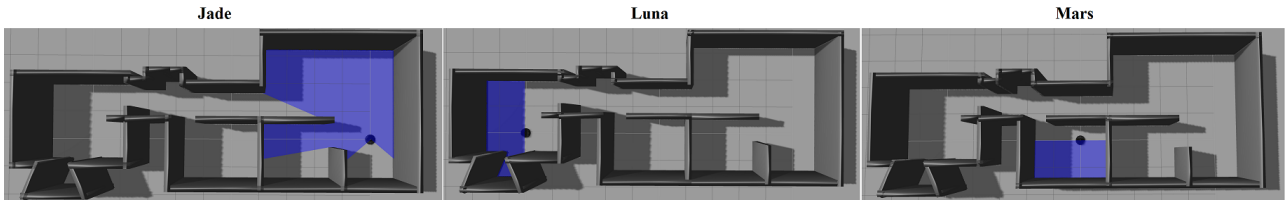

Figura 5-12.: Puntos de inicio de los robots en el *escenario*2.

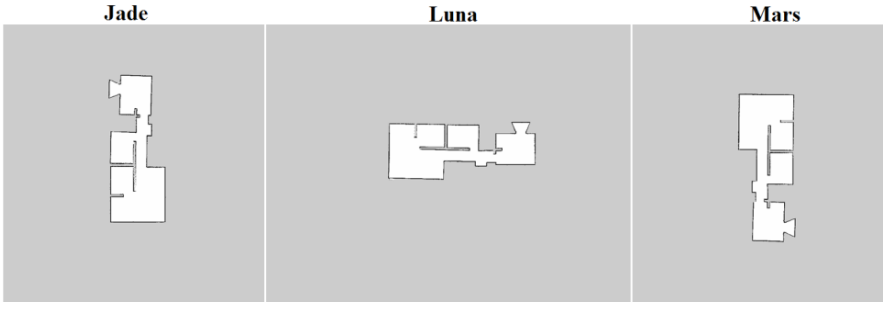

Figura 5-13.: Mapas obtenidos por el algoritmo de SLAM para cada uno de los robots de prueba en el *escenario*2.

Al igual que los resultados obtenidos para el *escenario*1, los robots pudieron obtener de manera adecuada una representación bidimensional del entorno de prueba. Esto demuestra

la robustez y utilidad del sistema de navegación implementado para este trabajo. Estos resultados, sirven como punto de partida para las siguientes pruebas desarrolladas, donde se espera que el algoritmo de mezcla logre obtener resultados similares a los mapas obtenidos en esta primera prueba.

5.6.2. Prueba 2 – Reconstrucción colaborativa

Esta prueba consistió en el uso de 2 robots para reconstrucción colaborativa de los entornos de pruebas. En específico, se desarrollaron 4 pruebas, 2 para cada escenario.

*escenario*1
En el *escenario*1, se utilizaron los robots Jade-Luna y Jade-Mars para las pruebas de reconstrucción del entorno. En la Figura **5-14** se muestra las posiciones iniciales para los casos escogidos, en ellas se evidencia que los robots inician en posiciones separadas y sin referencia alguna entre ellas.

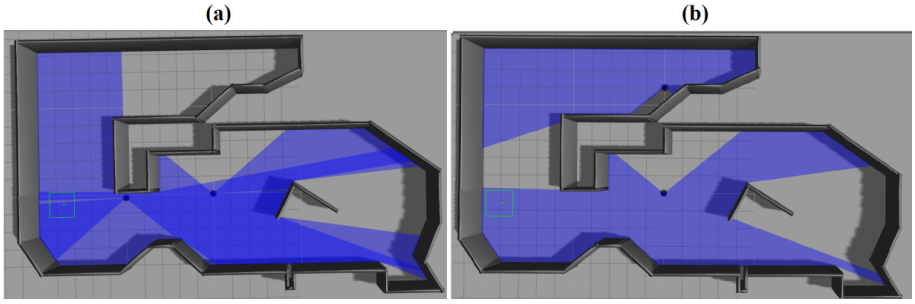

Figura 5-14.: Puntos de inicio de los robots en el *escenario*1 para los casos Jade-Luna (a) y Jade-Mars (b).

Una vez los robots iniciaron con su tarea de exploración, se ejecutó el algoritmo de mezcla. Dicho algoritmo logró encontrar un solapamiento entre los mapas de cada robot y construir un mapa global con esa información. La Figura **5-15** muestra el estado de los mapas justo antes de su mezcla.

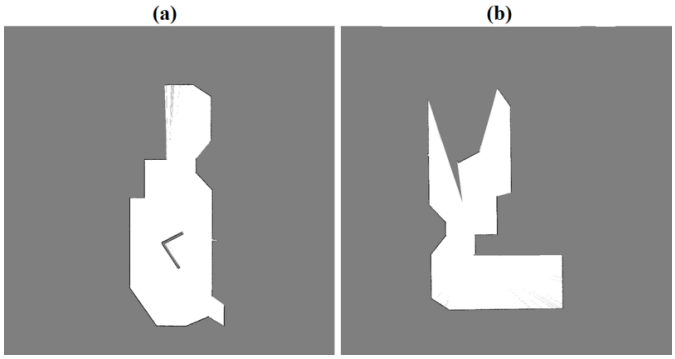

Figura 5-15.: Estado de los mapas de los robots Jade (a) y Luna (b) al momento de su análisis y posterior mezcla.

Una vez los puntos característicos y descriptores son encontrados, el algoritmo de mezcla los utiliza para encontrar un modelo consistente entre los mapas. El resultado de la mezcla se muestra en la Figura **5-16**.

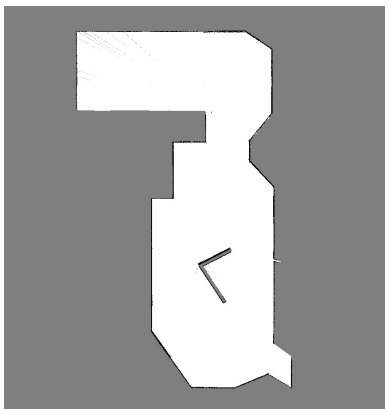

Figura 5-16.: Resultado de la mezcla de los mapas mostrados en la Figura **5-15**.

Como se puede notar, el algoritmo de mezcla logró fusionar o mezclar de manera consistente los mapas parciales aportados por los robots Jade y Luna. El algoritmo solo necesitó de 3 puntos para encontrar un modelo rígido entre los mapas. Con esa información, el algoritmo de RANSAC logró encontrar la transformación entre los mapas. El resultado, tal y como se detalla en la Figura **5-16**, muestra un desempeño óptimo del algoritmo de mezcla desarrollado.

Por otro lado, el algoritmo de control de robots móviles decidió detener el robot Jade y darle un nuevo objetivo (punto sobre el mapa) al robot Luna. De la sección 4.2 se sabe que el algoritmo de toma de decisiones analiza las distancias viajadas, el estado de operación, el porcentaje de exploración del entorno y la distancia a los objetivos. En este caso, ambos robots se encontraban operando, las distancias restantes a objetivos eran $0{,}66m$ y $0{,}45m$ para Jade y Luna, respectivamente. Con esta información, el algoritmo tomó la decisión de desviar a Luna hacia un nuevo objetivo y permitir que Jade continuara con su recorrido. Como el algoritmo de mezcla actualiza los mapas mezclados con un periodo de 10 segundos, cuando el algoritmo actualizó los mapas, decidió detener a Jade y permitir que Luna continuara con su recorrido. Este comportamiento era de esperarse dado que las fronteras de exploración del entorno se encontraban hacia Luna, pero como ambos robots no deben tener un mismo objetivo y Jade estaba más distante que Luna, la detección de Jade era la decisión adecuada.

Ahora bien, para el segundo caso se tienen posiciones entre los robots mucho más distantes en el entorno. La Figura **5-17** muestra el estado de los mapas de los robots justo antes de su mezcla. La Figura **5-18** muestra los resultados de la mezcla de los mapas, luego de que el algoritmo de apareamiento y RANSAC encontraron un modelo válido.

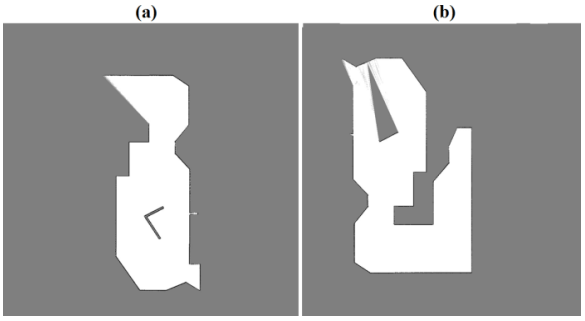

Figura 5-17.: Estado de los mapas de los robots Jade (a) y Mars (b) al momento de su análisis.

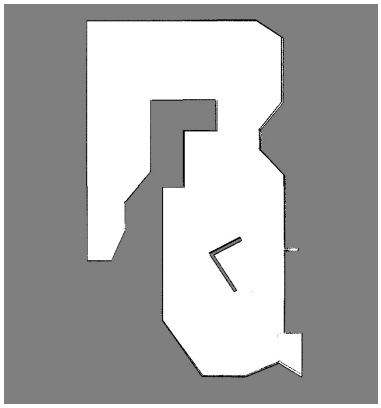

Figura 5-18.: Resultado de la mezcla de los mapas mostrados en la Figura **5-17** en el *escenario*1.

Como se puede notar, el comportamiento del algoritmo de mezcla fue adecuado y logró obtener un mapa mezclado consistente con el entorno de prueba. Para este caso, el algoritmo logró extraer 5 parejas que fueron usadas para encontrar el modelo rígido entre los mapas. En este caso, dado el estado avanzado de uno de los mapas, era normal esperarse encontrar un número mayor de puntos correspondientes. Una vez el algoritmo de mezcla logra informar a los robots de la actualización del mapa de exploración, el algoritmo de toma decisiones, decide darle un objetivo a Jade y paralizar a Mars. La decisión se sustenta en que para el estado de los mapas, Mars había recorrido mucho más trayecto que jade y su mapa se encontraba en un porcentaje de exportación mucho mayor al de jade. Por lo tanto, es normal que Jade haya sido el robot desviado.

*escenario*2

Para este entorno, se utilizaron los robots Luna-Jade y Luna-Mars para las pruebas de reconstrucción del entorno. En la Figura **5-19** se muestra las posiciones iniciales para los casos escogidos.

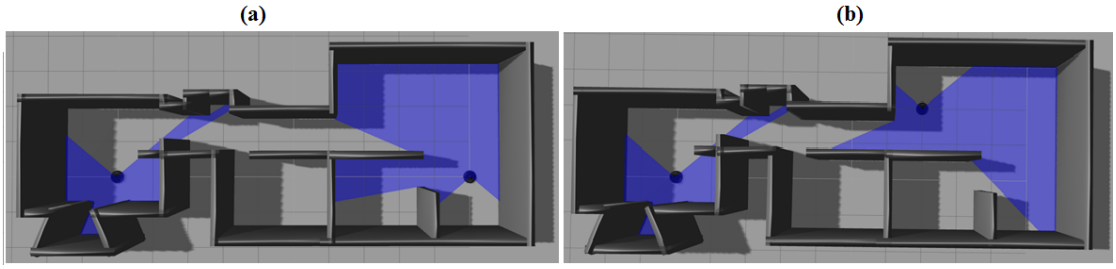

Figura 5-19.: Puntos de inicio de los robots en el *escenario*1 para los casos Luna-Jade (a) y Luna-Mars (b).

Para estos dos casos, el algoritmo de mezcla encontró solapamientos entre los mapas de los robots. Para el primer caso, en la Figura **5-20** se muestra el estado de los mapas justo antes de su mezcla.

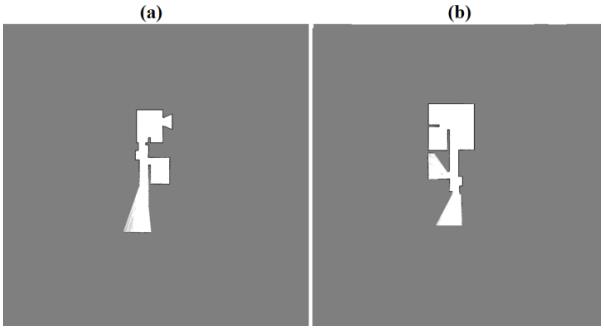

Figura 5-20.: Estado de los mapas de los robots Luna (a) y Jade (b) al momento de su mezcla.

Una vez se tienen los puntos característicos y sus descriptores, el algoritmo de apareamiento y RANSAC lograron encontrar un modelo consistente entre los mapas, los algoritmos hallaron 3 parejas correspondientes. El resultado de la mezcla de los mapas se muestra en la Figura **5-21**.

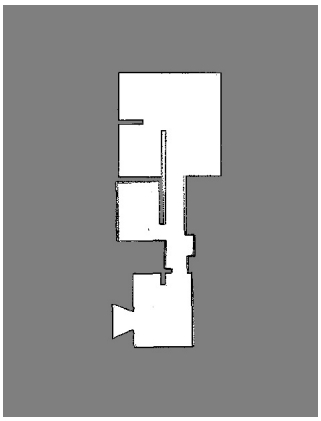

Figura 5-21.: Resultado de la mezcla de los mapas mostrados en la Figura **5-20**.

Seguido a la mezcla de los mapas, el algoritmo de control de robots móviles decidió detener a los robots involucrados en la prueba, dado que por el estado de los mapas una vez unidos ya no quedaban fronteras de exploración. Al no contar con fronteras de exploración, se asume que el entorno ha sido completado. Si se compara el resultado de esta mezcla con los mapas obtenidos de manera individual, en efecto el mapa se completa, por lo tanto ya no es necesario seguir explorando. Esta acción implica un ahorro de energía y tiempo de exploración del entorno, dos de los objetivos de hacer uso de algoritmos de mezcla aplicado a MR-SLAM.

Ahora bien, para el segundo caso la posición de Luna no varía, mientras que la de Mars es cercana a la posición de Jade en el caso anterior. La Figura **5-22** muestra el estado de los mapas de los robots Luna y Mars, mientras que la Figura **5-23** muestra el resultado de la mezcla para este caso.

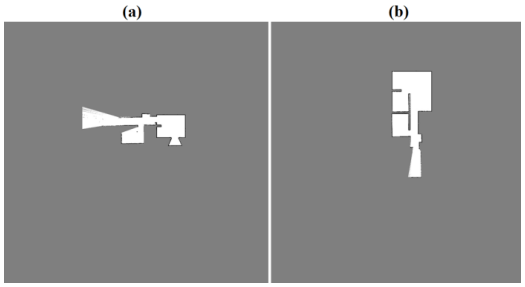

Figura 5-22.: Mapas de los robots Luna (a) y Mars (b) justo antes de su mezcla.

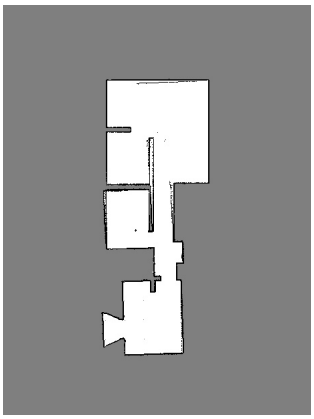

Figura 5-23.: Resultado de la mezcla de los mapas mostrados en la Figura 5.25 en el *escenario*2.

Para este segundo, el algoritmo de mezcla tuvo un comportamiento adecuado y logró obtener un mapa mezclado consistente con el entorno de prueba. Para este caso, el algoritmo de apareamiento y RANSAC lograron extraer 3 parejas que fueron usadas para encontrar el modelo rígido entre los mapas. Al igual que el caso anterior, por el estado avanzado de los mapas, el algoritmo de toma de decisiones determinó dejar que los robots culminaran con sus

rutas actuales de exploración y luego pararlos, ya que no existían fronteras de exploración luego de la mezcla de los mapas.

Finalmente, para los 4 casos detallados, los algoritmos funcionaron con un 100 % de efectividad, no hubo errores en la ejecución y cálculos imprecisos, mostrando la solidez del algoritmo de mezcla propuesto. Se evidenció durante las pruebas del algoritmo que hubo casos en los que el porcentaje de mezcla de dos mapas alcanzaba el 82 % para parejas correspondientes mal clasificadas. Sin embargo, como se mostró en la Sección 4.1 y 5.2, el porcentaje de aceptación/rechazo de una transformación se seleccionó en 90 %, precisamente para evitar esta situación. Con estas 4 pruebas, la validación del algoritmo de mezcla culminó satisfactoriamente y cumplió con las expectativas y objetivos planteados para este trabajo.

5.6.3. Prueba 3 – Reconstrucción colaborativa de *escenario*1

Esta prueba consistió en el uso de 3 robots para la reconstrucción colaborativa del escenario. Lo que quiere validar es la extensión del algoritmo de mezcla a un grupo de más de 2 agentes dispuestos en un mismo entorno y con una tarea en común: obtener colaborativamente un mapa global de navegación. En la etapa inicial de la prueba, los robots fueron dispuestos en 3 puntos diferentes y distantes entre sí, ver Figura **5-24**. Cada robot recibe una orden de explorar autónomamente su entorno, tal y como ocurrió en todas las anteriores pruebas.

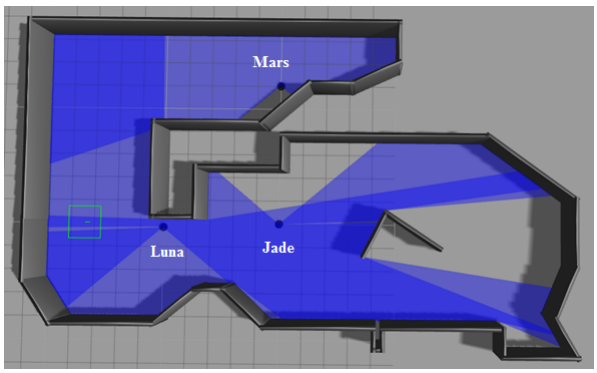

Figura 5-24.: Posiciones iniciales para los robots Jade, Luna y Mars en el *escenario*1.

El primer solapamiento encontrado por el algoritmo de mezcla fue entre los robots Jade y Luna. El estado de los mapas en ese instante es mostrado en la Figura **5-25**. Con dichos mapas, el algoritmo de apareamiento y RANSAC logra encontrar un modelo válido entre los mapas y permite realizar la mezcla de los mismos, Figura **5-26**.

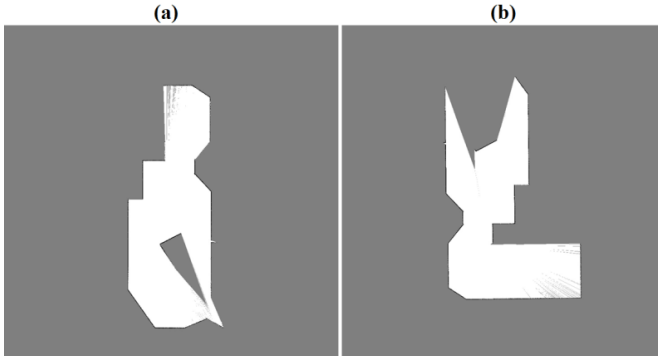

Figura 5-25.: Mapas de los robots Jade (a) y Luna (b) justo antes de su mezcla.

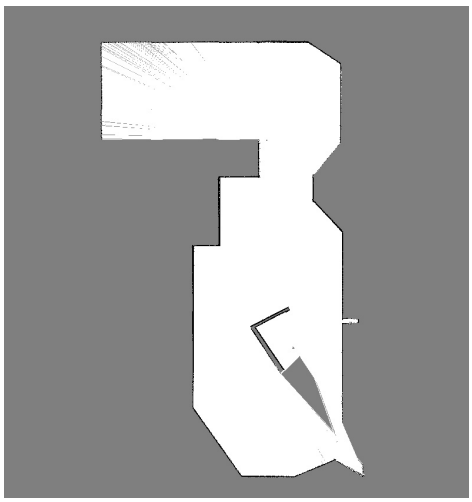

Figura 5-26.: Resultado de la primera mezcla de los mapas de Jade y Luna en el *escenario*1.

Para esta primera mezcla se usaron 3 parejas correspondientes. Nuevamente, el algoritmo de toma de decisiones determinó desviar a Luna hacia un nuevo objetivo y a Jade le permitió culminar con su recorrido. En actualizaciones posteriores de este mapa, Jade es detenido por el algoritmo de control, dado que las fronteras de exploración corresponden a los objetivos asignados a Luna. En este punto, solo Luna y Mars siguen activos. De esta manera, el algoritmo de mezcla y control optimiza el rendimiento general de la red de robots en tareas de exploración colaborativa.

Una vez Mars se acerca hacia el mapa explorado por Jade y Luna, el algoritmo de mezcla detecta un solapamiento entre los mapas. En este caso, el solapamiento se da entre los robots Jade y Mars, dado que el algoritmo de mezcla elimina de su cola de análisis a Luna puesto que ésta ya se encuentra asociada a Jade por la mezcla realizada con anterioridad. En este punto, el mapa de Jade corresponde a la mezcla de Jade y Luna. En la Figura **5-27** se muestran los estados de los mapas a mezclarse.

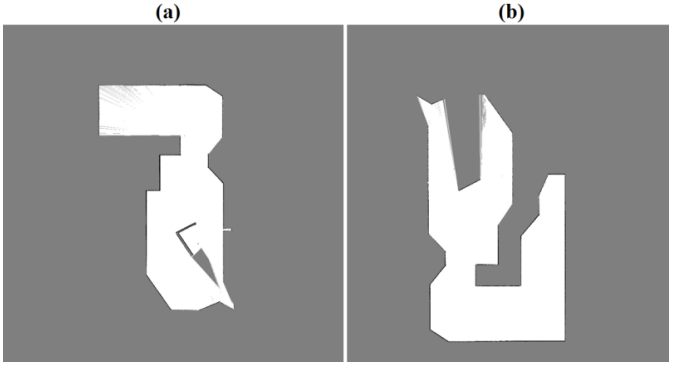

Figura 5-27.: Mapas de los robots Jade-Luna (a) y Mars (b) justo antes de la segunda mezcla.

Con la información contenida y aportada por los mapas de la figura anterior, el algoritmo de correspondencia encuentra un modelo o transformada rígida válida conformada por 3 parejas correspondientes. El resultado final de la segunda mezcla realizada se muestra en la Figura **5-28**.

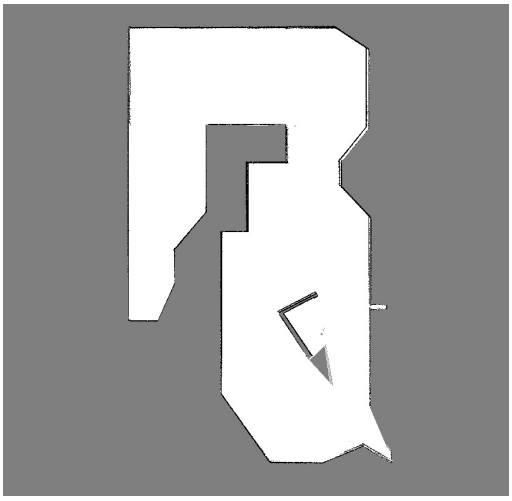

Figura 5-28.: Resultado de la mezcla de los 3 mapas en el *escenario*1.

Una vez la segunda mezcla se concreta, el algoritmo de toma de decisiones detiene al robot Luna y permite Mars continúe con su trayectoria. Luego de esa señal, Mars genera 2 trayectorias más y finaliza con su exploración, dado que el algoritmo de mezcla le envía la orden de detención.

Finalmente, como se pudo ver esta prueba, las mezclas resultaron adecuadas y consistentes. Los porcentajes de aceptación para ambas mezclas fueron de 95,236 % y 91, 013 %, para la primera y segunda mezcla. El árbol de tramas final para esta prueba se muestra en la Figura **5-29**.

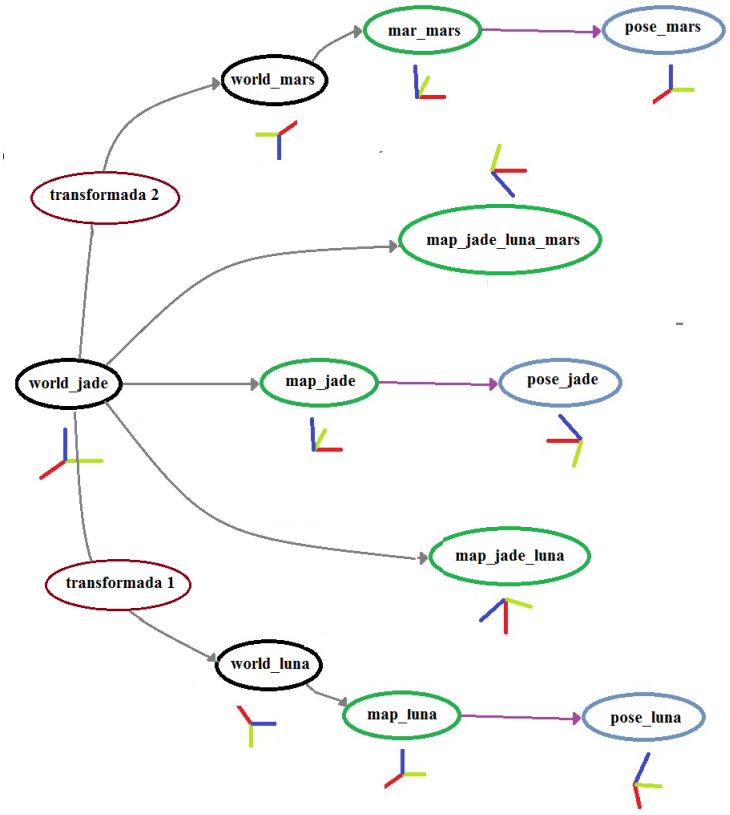

Figura 5-29.: Árbol de tramas construido por el algoritmo de mezcla para la prueba 3 en el *escenario*1.

En la anterior figura, se refleja como en el árbol de tramas el marco de referencia global del mapa mezclado es $world_jade$. Este marco se determina ya que las mezclas se hicieron respecto al robot Jade. Otro punto a detallar es que el algoritmo construye las tramas de los mapas mezclados Jade-Luna y Jade-Luna-Mars para ser consultadas por el algoritmo de toma decisiones. Siempre se usa el último mapa mezclado para la toma de decisiones. Con estas pruebas, se demuestra la validez y funcionamiento del algoritmo de mezcla desarrollado, siendo éste una técnica aplicable al campo del MR-SLAM.

5.6.4. Eficiencia del sistema

Como punto final para probar la validez de la técnica desarrollada en este trabajo. Se tomaron muestras de tiempos de cómputo en cada una de las pruebas para determinar si el uso de un algoritmo de mezcla reduce o no el tiempo de exploración colaborativa de un entorno.

La Tabla **5-4** muestra los tiempos de exploración para la prueba 1. Estos tiempos sirven de base para medir el desempeño de tareas de exploración colaborativa. Los tiempos relacionados con las pruebas 2 y 3 se muestran en la tablas **5-5** y **5-6**, respectivamente.

Tiempos de exploración para la prueba 1						
	Escenario 1			**Escenario 2**		
	Tiempo (sec)	**Tiempo (sec)**	**Tiempo (sec)**	**Tiempo (sec)**	**Tiempo (sec)**	**Tiempo (sec)**
Robot Jade	322	324	327	165	167	168
Robot Luna	352	350	355	148	149	153
Robot Mars	433	435	435	186	185	190

Tabla 5-4.: Tiempo de exploración para cada robot en ambos escenarios. Estos datos corresponden a la prueba 1.

Tiempos de exploración para la prueba 2				
	Escenario 1		**Escenario 2**	
	Tiempo (sec)	**Tiempo (sec)**	**Tiempo (sec)**	**Tiempo (sec)**
Robot Jade-Luna	293	289	127	125
Robot Jade-Mars	268	271	133	136

Tabla 5-5.: Tiempo de exploración para la prueba 2

Tiempos de exploración para la prueba 3	
	Escenario 1
	Tiempo (sec)
Robot Jade-Luna-Mars	239

Tabla 5-6.: Tiempo de exploración para la prueba 3

De las anteriores tablas se puede destacar el decaimiento de los tiempos de exploración con el uso de más agentes en el entorno. Puntualmente, para las pruebas del *escenario*1, los tiempos de exploración decayeron un 24 % entre las pruebas 1 y 2, mientras que los tiempos decayeron un 35.2 % entre los tiempos de exploración de las pruebas 1 y 3. Por otro lado, los tiempos de exploración para el escenario 2 mostraron una reducción del 21.8 % entre las pruebas 1 y 2.

Inicialmente se esperaba que el tiempo de exploración se redujera proporcionalmente al número de agentes en el entorno, sin embargo, los resultados mostrados en las tablas anteriores no muestran una relación directa entre estas dos variables. Esto se explica por el hecho de

que no se puede predecir cómo un robot ejecuta su tarea de exploración. Como se puede ver en la Tabla **5-4**, los tiempos de exploración individual para un mismo robot varían. Incluso, éstos varían de robot a robot (los tiempos de Mars difieren en una cantidad considerable a los tiempos de Jade y Luna). Por otro lado, como la reconstrucción está sujeta a ruidos percibidos durante la exploración, no siempre se tendrán los mismos puntos característicos y por ende, los resultados del algoritmo de mezcla cambiaran de una mezcla a otra.

Por otra parte, es necesario citar el hecho de que el algoritmo de toma decisiones determinó en varias ocasiones paralizar la operación de un agente en el entorno, quitando tiempo de exploración para los agentes y dejando que uno solo terminara con la exploración. Por ende, la eficiencia del algoritmo de mezcla no solo debe ser medida en términos de reducción de tiempo de exploración, sino en ahorro de energía dado que el algoritmo puede determinar cómo optimizar la exploración de un entorno teniendo en cuenta el tiempo y la energía.

Con el análisis desarrollado a lo largo de este capítulo, se puede determinar que la técnica desarrollada y validada a lo largo de este trabajo cumple con los requisitos para ser usada como una aproximación consistente al problema de MR-SLAM, dado que es capaz de mezclar adecuadamente mapas parciales de diferentes agentes en un mismo entorno y sin condiciones iniciales. Adicionalmente, el algoritmo desarrollado es capaz de controlar a dichos robots, mediante la toma de decisiones y generación de objetivos de exploración.

6. Conclusiones

Como ha quedado validado en este trabajo, el desarrollo de técnicas de mezcla de mapas aplicables al campo del MR-SLAM, sin condiciones iniciales de operación, es posible. La variante utilizada en este trabajo ofrece una válida y eficiente aproximación a este campo, ya que los resultados obtenidos sustentan dicha hipótesis. Haber abordado el problema de mezcla de mapas desde la perspectiva de la Visión por Computador clásica, permitió desarrollar la técnica con un desempeño en tiempo real destacable.

Por otro lado, la metodología seguida en este trabajo permitió el desarrollo de los algoritmos que conforman la técnica de mezcla de mapas propuesta. A través de un estudio, análisis a profundidad y validación de cada una de las técnicas existentes en la literatura, se pudo construir de manera adecuada el algoritmo propuesto. Además, es claro que esta metodología permitió el desarrollo de un nuevo detector de características, toda vez que en la etapa de detección de características, se optó por el diseño de un nuevo detector que, como se demostró, es extensible a imágenes en alta escala de gris. En gran medida, la rapidez del algoritmo de mezcla se debe a la detección adecuada de puntos de interés para mapas OGM. De ahí que la técnica propuesta presente como gran avance su alto desempeño en tareas en tiempo real.

El buen funcionamiento del algoritmo de cálculo de tramas entre mapas mezclados. El uso de la técnica RANSAC junto con uno de refinamiento, permitió encontrar la métrica adecuada para saber cuando una transformada rígida debe ser aceptada o rechazada por el algoritmo de mezcla. Los resultados mostraron un alto rechazo hacia transformadas inválidas y una buena aceptación de transformadas válidas. Además, el costo computacional de estas operaciones no supuso un incremento del tiempo de ejecución del algoritmo de mezcla, toda vez que los resultados mostraron que en menos de 300 milisegundos se podía construir el árbol de tramas válido para dos o más mapas mezclados.

Ahora bien, es de suma importancia mencionar el papel del algoritmo de control de robots complementario al algoritmo de mezcla de robot. Sin lugar a dudas, el árbol de decisión construido para este trabajo jugó un papel importante en la optimización de los recursos dispuestos para la exploración colaborativa de un entorno. El desarrollo y los resultados obtenidos con este algoritmo de control permiten afirmar que para el óptimo funcionamiento de un algoritmo de mezcla o de una técnica MR-SLAM, se hace necesario incluir un algoritmo

con estas características, puesto que de nada sirve mezclar eficientemente mapas OGM sino se sabe qué hacer con la información resultante de dicha mezcla. Esta idea refuerza aún más el éxito del trabajo planteado y desarrollado, ya que se obtuvo una técnica completa, que abordó más allá de los objetivos y resultados esperados.

Sin embargo, como todo desarrollo, la técnica aquí expuesta está sujeta a una validación más extensa, en aras de mejorar u optimizar lo concebido hasta el momento. Como trabajo a futuro, se plantea hacer extensivo el algoritmo de mezcla a más de 8 agentes en tiempo real y así hacerlo útil a colmenas de robots. Adicionalmente, extender la técnica a mapas tridimensionales puede mejorar la mezcla de mapas, ya que éstos contienen mucha más información que los mapas bidimensionales y, por ende, los descriptores tienden a ser mucho más característicos y únicos que los obtenidos en este trabajo. Es necesario recordar que un problema recurrente con representaciones bidimensionales de los entornos, es que son sensibles a que dos o más espacios sean similares y, en consecuencia, mal apareados.

En conclusión, el aporte con esta técnica al campo de MR-SLAM permite vislumbrar que nuevos y mejores desarrollos se puedan obtener, partiendo de lo expuesto y desarrollado a lo largo de este trabajo. Sin lugar a dudas, el desarrollo de algoritmos aplicables a tareas de exploración colaborativas es posible y valioso, toda vez que su utilidad está sustentada y sus propiedades demostradas.

A. Técnicas de SLAM

Las técnicas SLAM pueden ser clasificadas en dos grupos: los métodos basados en filtros y los métodos basados en correspondencias. Las aproximaciones basadas en filtros abordan el SLAM como un problema probabilístico y aplican filtros para estimar un vector de estado que contiene la pose (dupla de posición y orientación) del vehículo y la posición de puntos de referencia. Los filtros más aplicados son el filtro extendido de Kalman (EKF), los filtros de partículas (PF), los filtros UKF (Unscented Kalman Filter) y filtros de información extendida.

A.1. SLAM basado en filtro extendido de Kalman

El filtro extendido de Kalman (Extended Kalman Filter - EKF), fue la primera técnica más utilizada en las aproximaciones al problema de SLAM. Este se basa en el uso de un vector de estados para estimar la posición del robot, junto con un conjunto de características del entorno y con una matriz de covarianza que relaciona el error asociado a cada una de las estimaciones. En estas aproximaciones, el vector de estado y matriz de covarianza se actualizan utilizando EKF a medida que el vehículo se mueve en el entorno. El SLAM con EKF asume una representación basada en características del entorno, en la cual los objetos del entorno pueden ser representados como puntos en un espacio de parámetros y la pose del robot y localización de las características forman una red de relaciones espaciales.

Dentro de las debilidades de los EKF, se encuentran: *i*) el uso de modelos linealizados para modelos de movimientos y percepción de obstáculos no lineales, *ii*) la complejidad computacional, debido a la matriz de covarianza que va siendo actualizada y a los cálculos que debe hacer el algoritmo para actualizar y refinar la posición y el mapa del robot. Debido a la introducción de una nueva marca o referencia del entorno, los datos asociados a este nuevo estado deben ser incluidos dentro de la matriz de covarianzas y, adicionalmente, el algoritmo debe calcular la relación de este nuevo estado con los demás estados introducidos con anterioridad para medir la consistencia de las estimaciones hechas y actualizar la pose y mapa reconstruido.

- Hector SLAM [22]: técnica de SLAM basada en el filtro extendido de Kalman para estimación 3D del mapa y posición del robot en cada instante. Es una técnica SLAM

2D basada en un método de correspondencia robusto y una técnica de navegación 3D usando un sistema de sensado inercial.

Hector SLAM basa su proceso de localización en tiempo real, haciendo uso exclusivamente de la alta frecuencia de muestreo de los sensores de percepción para procesar y estimar la pose del robot. Esta técnica no tiene en cuenta la estimación por odometría, debido a que los autores consideran que la baja precisión de este proceso le resta exactitud y desempeño al algoritmo. Al contar con esta novedad, este algoritmo es muy usado para robots UAV (aviones no tripulados en español).

- KartoSLAM: técnica de SLAM basada en grafos, desarrollada por la empresa SRI International's Karto Robotics y liberada su versión libre en ROS (Robot System Operating). Fundamentalmente, difiere de otras técnicas dado que representa el mapa captado a través de nodos y arcos que los conectan. Cada nodo representa una Pose del robot incluyendo las medidas de los sensores y la incertidumbre asociada a dicha información, mientras que los arcos o conexiones entre nodos representan las posibles trayectorias de robot de un punto a otro, al igual que representan las restricciones de espacio por la cuales dicho robot puede moverse.

  Esta aproximación permite desarrollar el mapeo y localización del robot en grandes superficies, ya que un nuevo nodo es almacenado cuando su información ha sido verificada y consistente con el entorno (ese nuevo nodo debe poseer conexiones con los nodos anteriormente procesados). Sin embargo, el consumo de memoria para almacenar su información y el tiempo de computación al tener un grafo tan grande son sus limitantes.
  KartoSLAM ha demostrado ser una aproximación de SLAM precisa y escalable dentro de las técnicas de SLAM conocidas.

A.2. SLAM basado en filtro de partículas

Los problemas anteriormente descritos con los EKF han permitido el surgimiento de otras técnicas para resolver el problema de estimación de la pose y mapa de un entorno. Este otro enfoque usa filtros de partículas. Estas técnicas modelan un entorno mediante pequeñas partículas que, probabilísticamente, contienen una estimación del entorno y una posible pose del robot respecto a la marca percibida. En esta técnica, para cada partícula, se aplica el EKF con el fin de refinar la información contenida por dicha partícula. Sin embargo, esta no está correlacionada con las demás partículas, por lo que su cálculo computacional no es tan alto como el EKF puro.

Esta técnica resuelve el problema de trabajar con modelos no lineales al basarse en distribuciones no gaussianas. Sin embargo, su complejidad y limitación está en el hecho de que para reconstruir el entorno o estimar una posición consistente del robot, se requieren de muchas partículas y realizar cálculos sobre cada una de ellas es algo dispendioso, ya que se corre el riesgo de que existan partículas con información falsa o con alta presencia de ruido. Por lo tanto, el análisis de estas partículas no solo aumenta el tiempo de cómputo del algoritmo, sino que induce error en la estimación de la pose y del mapa construido.

En años recientes, se han venido desarrollando técnicas adicionales para resolver este inconveniente asociado a las partículas. La aparición del filtro Rao Blackwelized ha permitido la reducción y complejidad computacional de los filtros de partículas, ya que las factoriza con el fin de reducir la redundancia de partículas captadas y con información falsa. Adicionalmente, técnicas bio inspiradas como Optimización por Enjambre de Partículas (PSO) también han ayudado a eliminar partículas inadecuadas del proceso de estimación de la pose del robot.

- FastSLAM: la técnica más ampliamente conocida en SLAM basada en filtro de partículas. FastSLAM divide el problema de SLAM en dos partes: la primera relacionada con la posición del vehículo y la segunda con la posición de marcas de referencia dentro del entorno. En esta aproximación se utiliza un filtro de partículas (Rao-Blackwellized), para estimar la pose del vehículo y un EKF para estimar la posición de los puntos de referencia (landmarks). Esta formulación hace al algoritmo más robusto y le da la capacidad de manejar modelos de movimiento no lineales. A pesar de sus grandes ventajas, FastSlam también tiene inconvenientes, siendo el más significativo su degeneración con el tiempo (se relaciona con la pérdida de la diversidad de las partículas). Sin embargo, se han reportado soluciones eficientes a tales inconvenientes, lo que ha permitido implementaciones exitosas de FastSlam como el caso reportado en [19].

A.3. SLAM basado en Correspondencia

Los métodos basados en correspondencias toman dos conjuntos de datos consecutivos, generalmente una nube de puntos, e intentan estimar una transformación que permita asociar dichos datos para luego obtener el mapa del entorno junto con la pose del vehículo. Estos métodos son ampliamente utilizados en las aproximaciones basadas en visión. Se han reportado implementaciones con cámaras monoculares [37], estéreo [25] y cámaras infrarrojas de profundidad (RGB-D) [15]. Uno de los mecanismos de asociación de datos más utilizado en este tipo de métodos es ICP (Iterative Closest Point), que es una técnica que permite la alineación de conjuntos de datos [15].

El algoritmo más representativo de este tipo de técnicas SLAM es conocida como RGB-DSLAM. Sin embargo, debido al costo computacional relacionado a la asociación y corres-

pondencia de datos, esta técnica sigue siendo de amplio interés investigativo.

A.4. Resumen técnicas de SLAM

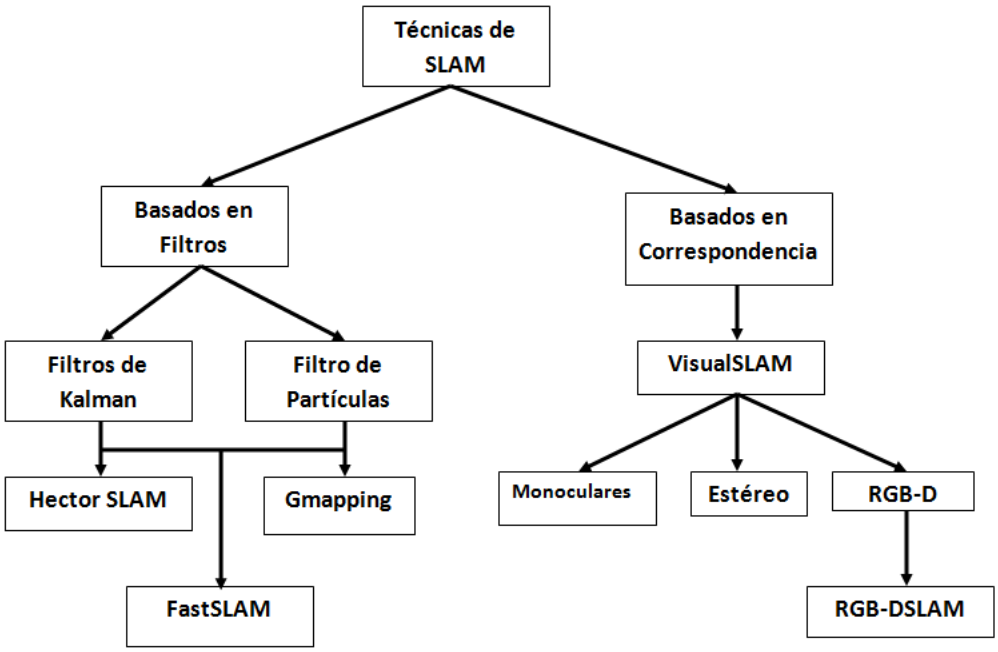

Figura A-1.: Técnicas de SLAM más representativas por ramas.

Basados en los resultados expuestos y analizados en [31], las técnicas que presentan un mejor desempeño, en términos de consistencia y velocidad del algoritmo, son Hector SLAM y Gmapping. Sin embargo, dada la pobreza relacionada con la calidad de la estimación ofrecida por la odometría interna de un robot, la técnica Hector SLAM se presenta como la mejor opción para la reconstrucción de mapas por ocupación de celda y, por ende, para ser implementada en el sistema de navegación de autónomo desarrollado para los robots de este trabajo.

B. Calidad y cantidad de puntos extraídos para grupo 1 y 3 de imágenes de prueba

B.1. Primer grupo de imágenes de prueba

Este grupo corresponde a imágenes binarizadas con presencia marcada de características y esquinas, dado que solo presentan dos tonalidades extremas en la escala de grises.

Las figuras **B-1**, **B-2**, **B-3**, **B-4** y **B-5** muestran los resultados para cada una de las técnicas. De los resultados mostrados en las figuras, todos los algoritmos tienen un comportamiento destacable. Todas las técnicas fueron capaces de detectar el 100 % de las posibles esquinas presentes en las imágenes de prueba. Esto demuestra que para imágenes en baja dimensión en la escala de grises, estos algoritmos son adecuados. Como punto a denotar, la técnica de detección propuesta no selecciona el punto de interés sobre el borde o sobre la característica, sino a un costado de ella. Como se explicó en la Sección 3.1.5, *"Detector de esquinas propuesto para SLAM"*, la detección siempre se hará sobre un píxel cuyo nivel de gris sea 255 (color blanco), ya que se considera que es en ese punto donde se encuentra en centro real de la característica detectada.

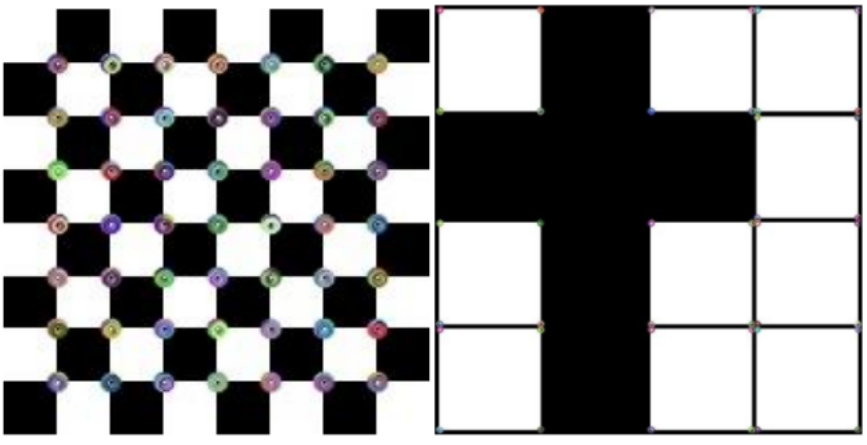

Figura B-1.: Puntos característicos extraídos por el detector de Harris. Es notable el desempeño de esta técnica con este tipo de imágenes binarizadas.

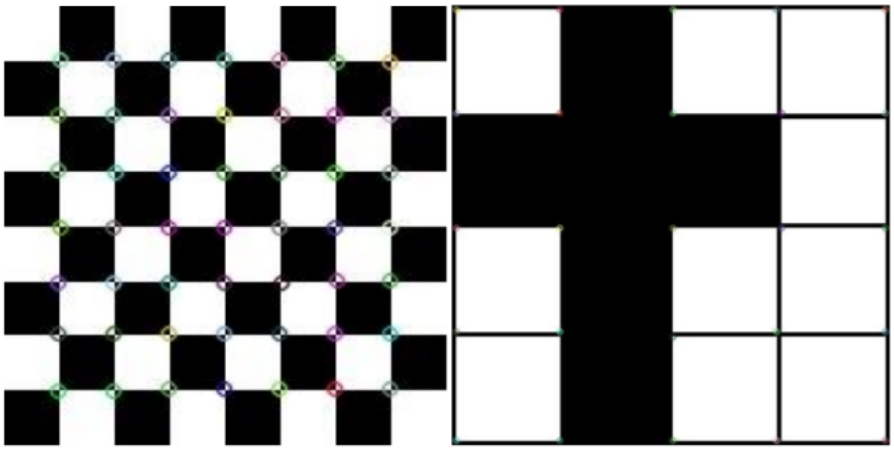

Figura B-2.: Puntos característicos extraídos por el detector de Shi-Tomasi. Como era de esperar, esta técnica muestra un mejoramiento de los puntos extraídos respecto al detector de Harris ya que no forma aglomeraciones en ciertas regiones.

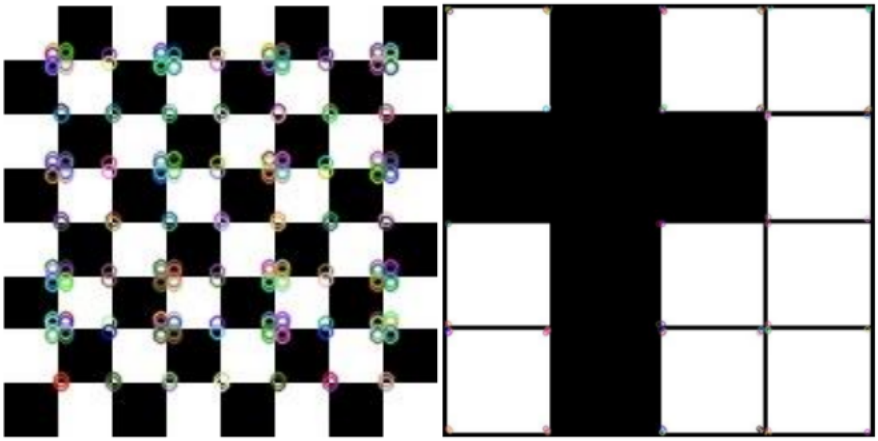

Figura B-3.: Puntos característicos extraídos por el detector de Trajkovic-Hedley Detector.

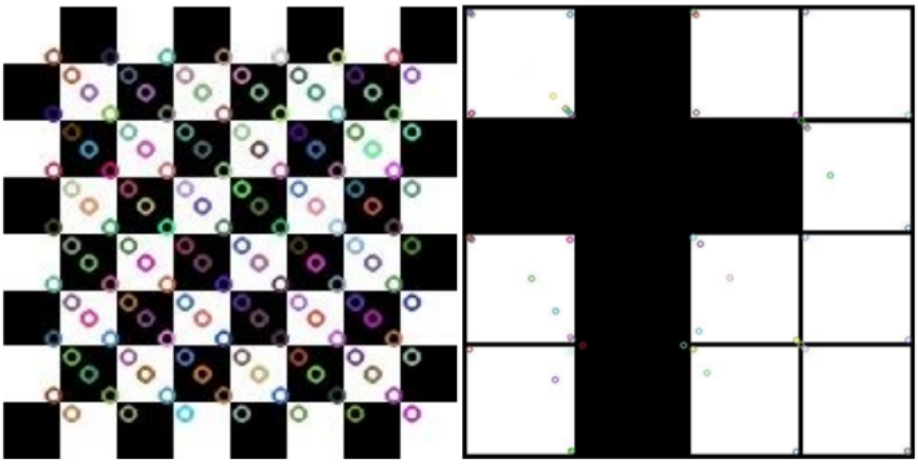

Figura B-4.: Puntos de interés extraídos por el Extractor de Puntos de SIFT. Esta técnica también muestra un buen desempeño.

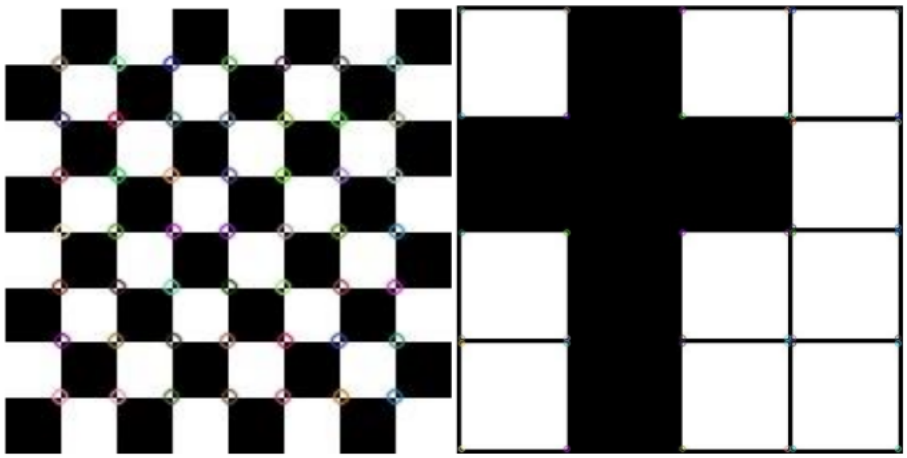

Figura B-5.: Puntos de interés detectados por el algoritmo de detección desarrollado para imágenes (mapas) generados por técnicas de SLAM.

De los resultados mostrados, es claro que el detector de Shi-Tomasi representa un mejoramiento del detector de Harris, ya que Shi-Tomasi utiliza un parámetro de dispersión de los puntos detectados y por ende no focaliza la detección en una misma área como si lo hace el detector de Harris. También es posible de denotar el rendimiento de la técnica desarrollada en este trabajo, ya que tiene un desempeño similar al detector Shi- Tomasi en el número de puntos detectados, Tabla **B-1**.

	Ajedrez		
Técnica	#Características Detectadas	#Características Reales	#Características Falsas
Detector de Harris*	441	49	392
Detector de Shi-Tomasi	49	49	0
Detector de Trajkovic**	194	49	145
Extractor de SIFT	326	49	277
Detector Desarrollados	49	49	0
	Cruz		
Técnica	#Características Detectadas	#Características Reales	#Características Falsas
Detector de Harris*	120	25	95
Detector de Shi-Tomasi	25	25	0
Detector de Trajkovic**	73	25	48
Extractor de SIFT	99	25	74
Detector Desarrollado	40	25	15
*Umbral: 140		**T_1: 125 y T_2: 112	

Tabla B-1.: Cantidad de puntos detectados por cada técnica versus el número de características reales en cada imagen de prueba.

Ahora bien, como lo muestra la Tabla **B-1**, es claro que solo la técnica propuesta y el detector Shi-Tomasi logran cumplir con el objetivo de detectar los puntos necesarios sobre la imagen, sin aumentar la presencia de falsos positivos en el vector de características de salida. Sólo Shi-Tomasi logró detectar el número exacto de características en ambas imágenes, mientras que el detector propuesto logra detectar 15 falsas características en la imagen 1 (Cruz). En el resto de las técnicas, la presencia de falsos positivos supera el 100 % de los puntos característicos reales de cada imagen. Esto, sin lugar a dudas, es una ventaja en términos computacionales, ya que demuestra que los algoritmos sobresalientes son precisos en los puntos que extraen y por lo tanto, la información que le entregan a las etapas de descripción y correspondencia del algoritmo de mezclado es la adecuada.

B.2. Tercer Grupo de Imágenes de Prueba

Este grupo corresponde a imágenes en alta dimensión en la escala de grises. Son imágenes comunes, que normalmente encontramos en equipos fotográficos o de vídeo cámara. Esta clase de imágenes presenta el desafío de obtener el mayor y mejor número de puntos de interés o puntos característicos, ya que ellas pueden contener información con alto ruido, diferentes focos de iluminación y perspectiva de los objetos, pero con diversos valores de intensidad de imagen (nivel de gris). Aquí el problema del alto contenido de ruido puede no ser tan traumático como si lo es para las imágenes de mapas de SLAM, ya que los gradientes o cambios de intensidad sobre estas áreas pueden no ser lo suficientemente relevantes para el algoritmo ya que no están tan marcadas como si lo están en los mapas.

En este tipo de imágenes, un paso extra es usado en la técnica desarrollada. Se trata de un extractor de borde común, Filtro de Sobel o Filtro de Scharr. En estas imágenes es necesario el uso de este extractor ya que la técnica trabaja en baja dimensión en la escala de grises y por ende ser hace necesario reducir la escala de la imagen original.

De las figuras **B-6**, **B-7**, **B-8**, **B-9** y **B-10** se puede deducir que Shi-Tomasi y el algoritmo desarrollado en este trabajo tienen el mejor desempeño en términos de dispersión y cobertura de los puntos de interés extraídos, ya que ambas técnicas no se concentran en áreas específicas, sino que exploran todo el espacio de búsqueda en aras de extraer la mayor y mejor cantidad de puntos característicos. Para este grupo de imágenes, la etapa de dispersión de los puntos de interés aplicada por Shi-Tomasi ayuda a mejorar el rendimiento de la técnica. Sin embargo, como se vio con el grupo anterior de imágenes, no siempre esta etapa es adecuada para todas las imágenes.

Figura B-6.: Características detectadas por el algoritmo de detección de Harris para el tercer grupo de imágenes.

Figura B-7.: Puntos Característicos extraídos por el detector Shi-Tomasi para el tercer grupo de imágenes.

Figura B-8.: Características detectadas por el algoritmo de detección de Trajkovic-Hedley para imágenes en alta dimensión en la escala de grises (tercer grupo de imágenes).

Figura B-9.: Características detectadas por el Extractor de Puntos de SIFT. Pese a los buenos resultados obtenidos, esta técnica requiere de un tiempo de ejecución alto.

Figura B-10.: Puntos Característicos extraídos por el Detector Desarrollado en este trabajo investigativo. Pese a no ser ideado para esta clase de imágenes, muestra un desempeño destacable en términos la dispersión y calidad de puntos extraídos. Fue la única técnica capaz abarcar su detección a lo largo de toda la imagen.

Otra análisis que se puede realizar es la notable diferencia entra las técnicas de Harris y Shi-Tomasi. De las figuras **B-6** y **B-7**, es claro que el detector de Harris solo concentra su detección en áreas con un cambio de gradiente grande, luego la técnica tiende a formar aglomeraciones de puntos de interés innecesarias, lo que repercute en una alta tasa de detección de puntos (ver Tabla **B-2**). Este diferencia en los resultados tiene una posible explicación, la aplicación de la métrica para determinar si un punto es o no característico. Mientras que el detector de Harris calcula su medida basado en el determinante y traza de la matriz Hessiana (usa todos los datos de la matriz), el detector de Shi-Tomasi únicamente usa el valor mínimo de los eigenvalores de esta misma matriz.

En lo que respecta al detector de puntos de SIFT, ésta presenta un comportamiento también destacable ya que los puntos extraídos no sobrepasan en cantidad (ver Tabla **B-2**) a los puntos de las técnicas hasta ahora más sobresaliente en los 3 grupos de imágenes, el detector Shi-Tomasi y el detector desarrollado. Este comportamiento es de esperarse para esta clase de técnicas, ya que el objetivo principal de este detector es trabajar con este tipo de imágenes.

	Lena	**Ventana 1**	**Ventan 2**
Técnica	#Características Detectadas	#Características Detectadas	#Características Detectadas
Detector de Harris*	2391	2503	3684
Detector de Shi-Tomasi	588	477	540
Detector de Trajkovic**	1725	1447	1195
Extractor de SIFT	1248	606	1470
Detector Desarrollado	1208	1327	652
	*Umbral: 130 **T_1: 1495 y T_2: 523	*Umbral: 130 **T_1: 3738 y T_2: 4000	*Umbral: 177 **T_1: 1196 y T_2: 822

Tabla B-2.: Cantidad de puntos detectados por cada técnica en imágenes de alta dimensión en la escala de grises.

Bibliografía

[1] Abiyev, Rahib H. ; Bektas, Şenol ; Akkaya, Nurullah ; Aytac, Ersin: Behaviour Trees Based Decision Making for Soccer Robots / Recent Advances in Mathematical Methods, Intelligent Systems and Materials. 2008. – Informe de Investigación

[2] Aditya, Andra: Implementation of a 4D fast SLAM including volumetric sum of the UAV. En: *Sixth International Conference on Sensing Technology (ICST)*. Kolkata, India, 2013

[3] Adluru, Nagesh ; Latecki, Longin J. ; Sobel, Marc ; Lakaemper, Rolf: Merging Maps Of Multiple Robots. En: *International Conference on Pattern Recognition (ICPR)*. Tampa, USA, 2008

[4] Bagrowski, Grzegorz ; Luckner, Marcin: Comparison of Corner Detectors for Revolving Objects Matching Task. En: *Artificial Intelligence and Soft Computing* 7267 (2012), p. 459–467

[5] Baya, H ; Essa, A. ; Tuytelaars, T. ; Van Gool, Luc: Speeded-Up Robust Features (SURF). En: *Computer Vision and Image Understanding* 110 (2006), p. 346–359

[6] Birk, Andreas ; Carpin, Stefano: Merging Occupancy Grid Maps From Multiple Robots. En: *Proceedings of the IEEE* 94 (2006), p. 1384–1397

[7] Blanco, J. ; González-Jiménez, J. ; Fernández-Madrigal, J.: A robust, multi-hypothesis approach to matching occupancy grid maps. En: *Robotica* 31 (2013), p. 687–701

[8] Blanco, J.L. ; Gonzalez-Jimenez, J. ; Fernandez-Madrigal, J.A.: A new method for robust and efficient occupancy grid-map matching / 3rd. Iberian Conference on Pattern Recognition and Image Analysis (IBPRIA'07). 2007. – Informe de Investigación

[9] Burgard, Wolfram ; Moors, Mark ; Fox, Dieter ; Simmons, Reid ; Thurn, Sebastian: Collaborative Multi-robot Explotarion. En: *International Conference on Robotics Automation*. San Francisco, USA, 2000

[10] Chen, Jie ; Zou, Li-hui ; Zhang, Juan ; Dou, Li-hua: The Comparison and Application of Corner Detection Algorithms. En: *Journal of Multimedia* 4 (2009), p. 435–441

[11] Chum, O. ; Matas, J.: Randomized ransac with t(d,d) test. En: *British Machine Vision Conference*, 2002, p. 448–457

[12] Daixian, Zhu: SIFT algorithm analysis and optimization. En: *International Conference on Image Analysis and Signal Processing (IASP)*, 2010

[13] Dinnissen, Pierre ; Givigi, Sidney N. ; Schwartz, Howard M.: Map Merging of

Multi-Robot SLAM using Reinforcement Learning. En: *International Conference on Systems, Man and Cybernetics (SMC)*. Seoul, South Korea, 2012

[14] Elfes, Alberto: Using Occupancy Grids for Mobile Robot Perception. En: *Computer* 22 (1989), p. 46–57

[15] Engelhard, Nikolas ; Endres, Felix ; Hess, Lürgen ; Sturm, Jürgen ; Burgard, Wolfram: Real-time 3D visual Slam with a hand-held RGB-D camera / European Robotics Forum. 2011. – Informe de Investigación

[16] Fischler, M.A. ; Bolles, R.C.: Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. En: *Communications of the ACM* 24 (1981), p. 381–395

[17] Fox, Dieter ; Ko, Jonathan ; Konolige, Kurt ; Limketkai, Benson ; Schulz, Dirk ; Stewart, Benjamin: Distributed Multi-Robot Exploration and Mapping. En: *2nd Canadian Conference on Computer and Robot Vision (CRV'05)*. Victoria, Canada, 2006

[18] Harris, Chris ; Stephens, Mike: A combined corner and edge detector. En: *In Proc. of Fourth Alvey Vision Conference*, 1988, p. 147–151

[19] Havangi, Ramazan ; Nekoui, Mohammad ; Teshnehlab, Mohammad: An improved FastSLAM framework using soft computing. En: *Turkish Journal of Electrical Engineering Computer Sciences* 20 (2012), p. 25–46

[20] He, Bo ; Zhang, Shujing ; Yan, Tianhong ; Zhang, Tao ; Liang, Yan ; Zhang, Hongjin: A Novel Combined SLAM Based on RBPF-SLAM and EIF-SLAM for Mobile System Sensing in a Large Scale Environment. En: *Sensors* 11 (2011), p. 10197–10219

[21] Howard, Andrew: Multi-robot Simultaneous Localization and Mapping using Particle filters / Jet Propulsion Laboratory, NASA. 2006. – Informe de Investigación. – 1–8 p.

[22] Kohlbrecher, S. ; Meyer, J. ; Von Stryk, O. ; Klingauf, U.: A Flexible and Scalable SLAM System with Full 3D Motion Estimation. En: *International Symposium on on Safety, Security and Rescue Robotics (SSRR)*, 2011, p. 155–160

[23] Kohlbrecher, S. ; Stryk, O. ; Meyer, J. ; Klingauf, U.: A Flexible and Scalable SLAM System with Full 3D Motion Estimation. En: *International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. Kyoto, Japan, 2011

[24] Lee, Jung-Suk ; Nam, Sang Y. ; Chung, Wan K.: Robust RBPF-SALM for Indoor Mobile Robots Using Sonar Sensors in Non-Static Environments. En: *Advanced Robotics* 25 (2012), p. 1227–1248

[25] Lemaire, Thomas ; Berger, Cyrille ; Jung, Il-Kyun ; Lacroix, Simon: Vision-Based SLAM: "Stereo and Monocular Approaches". En: *International Journal of Computer Vision* 74 (2007), p. 343–364

[26] León, A. ; Barea, R ; Bergasa, L. M. ; Lopez, E. ; Ocaña, M. ; Schleicher, D.: SLAM and Map Merging. En: *Journal of Physical Agents* 3 (2009), p. 13–23

[27] Li, Jing ; Allinson, Nigel M.: A comprehensive review of current local features for computer vision. En: *Neurocomputing* 71 (2008), p. 1771–1787

[28] Liu, Yanli ; Fan, Xiaoping ; Zhang, Heng: A Fast Map Merging Algorithm in the Field of MultiRobot SLAM. En: *Sensors* 2013 (2013), p. 1–8

[29] Lowe, David G.: Object recognition from local scale-invariant features / International Conference on Computer Vision. 1999. – Informe de Investigación

[30] Lowe, David G.: Distinctive image features from scale-invariant keypoints. En: *International Journal of Computer Vision* 60 (2004), p. 99–110

[31] Machado Santos, Joao ; Portugal, David ; Rocha, Rui P.: An Evaluation of 2D SLAM Techniques Available in Robot Operating System. En: *International Symposium on on Safety, Security and Rescue Robotics (SSRR)*, 2013

[32] Moratuwage, D. ; Vo, Ba-Ngu ; Wang, Danwei ; Wang, Han: Extending Bayesian RFS SLAM to Multi-Vehicle SLAM. En: *Control Automation Robotics Vision (ICARCV)*. Guangzhou, China, 2012

[33] Moravec, Hans ; Elfes, Alberto: High Resolution Maps from Wide Angle Sonar. En: *International Conference on Robotics Automation*. St. Louis, USA, 1985

[34] Muja, Marius ; Lowe, D.G.: Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. En: *International Conference on Computer Vision Theory and Application VISSAPP'09)*, 2009, p. 331–340

[35] iñals Pons, Joaquín: *Localización y generación de mapas de entorno (SLAM) de un robot por medio de una Kinect*. Valencia, Escola Tecnica Superior d'Enginyeria Informatica, Universitat Politècnica de València, Tesis de Pregrado, 2012

[36] Rublee, Ethan ; Rabaud, Vincent ; Konolige, Kurt ; Bradski, Gary: ORB: an efficient alternative to SIFT or SURF. En: *International Conference on Computer Vision (ICCV)*, 2011

[37] Solà, Joan ; Vidal-Calleja, Teresa ; Civera, Javier ; Martínez Montiel, José M.: Impact of landmark Parametrization on Monocular EKF-SLAM with Points and Lines. En: *International Journal of Computer Vision* 97 (2012), p. 155–160

[38] Stasse, Oliver ; Davison, Andrew J. ; Sellaouti, Ramzi ; Yokoi, Kazuhito: Real-time 3D SLAM for Humanoid Robot considering Pattern Generator Information. En: *International Conference on Intelligent Robots and Systems, IROS 2006*. Beijing, China, 2006

[39] Sugiyama, H. ; Tsujioka, T. ; Murata, M.: Collaborative movement of rescue robots for reliable and effective networking in disaster area. En: *Collaborative Computing: Networking, Applications and Worksharing*. San Jose, CA, 2005

[40] Sun D., Kleiner A. ; Wendt, T.M.: Multi-Robot Range-Only SLAM by Active Sensor Nodes for Urban Search and Rescue. En: *RoboCup 2008: Robot Soccer World Cup XII* 5399 (2009), p. 318–330

[41] Swere, Erick ; Mulvaney, David: Robot Navigation Using Decision Trees / Electronic Systems and Control Division Research. 2003. – Informe de Investigación

[42] Thorpe, Jeremy ; McEliece, Robert: Data Fusion Algorithms for Collaborative Ro-

botic Exploration / Jet Propulsion Laboratory, NASA. 2002. – Informe de Investigación. – 1–14 p.

[43] Thrun, S. ; Liu, Y.: Multi-Robot SLAM with Sparse Extended Information Filers. En: *The Eleventh International Symposium on Robotics Research* 15 (2005), p. 254–266

[44] Thurn, Sebastian ; Fox, Dieter ; Burgard, Wolfram: *Probabilistic Robotics*. Cambridge : The MIT Press, 2005

[45] Toft Pedersen, Jacob: SURF: Feature detection description / Department of Computer Science, Aarhus University. 2011. – Informe de Investigación

[46] Tomasi, Carlo ; Kanade, Takeo: Detection and Tracking of Point Features / International Journal of Computer Vision. 1991. – Informe de Investigación

[47] Topal, Sebastian ; Erkmen, Ismet ; Erkmen, Aydan M.: A Novel Map Merging Methodology for Multi-Robot Systems. En: *Proceedings of the World Congress on Engineering and Computer Science*, 2010

[48] Trajkovic, M. ; Hedley, M: Fast corner detection. En: *Image and Vision Computing* 16 (1998), p. 75–87

[49] Vascák, Ján: Decision-Making Systems in Mobile Robotics / Technical University of Košice, Slovakia. 2008. – Informe de Investigación

[50] Wang, L. ; Ma, S. ; Xue, H. ; Hou, Z.: An improved SIFT algorithm for feature points matching of dairy cow images / World Automation Congress (WAC). 2010. – Informe de Investigación

[51] Xu, Weijun ; Jiang, Rongxin ; Chen, Yaowu: Map alignment based on PLICP algorithm for multi-robot SLAM. En: *2012 IEEE International Symposium on Industrial Electronics (ISIE)* 1 (2012)

[52] Zhang, Zhe. ; Nejat, Goldie: Intelligent Sensing Systems for Rescue Robots: Landmark Identification and Three-Dimensional Mapping of Unknown Cluttered Urban Search and Rescue Environments. En: *Advanced Robotics* 23 (2009), p. 1179–1198

[53] Zhou, Xun S. ; Roumeliotis, Stergios I.: Multi Robot SLAM Map Alignment with Rendezvous / Department of Computer Science Engineering, University of Minnesota. 2005. – Informe de Investigación. – 1–12 p.

[54] Zhou, Xun S. ; Roumeliotis, Stergios I.: Multi-robot SLAM with Unknown Initial Correspondence: The Robot Rendezvous Case. En: *International Conference on Intelligent Robots and Systems*. Beijing, China, 2006

[55] Özkucur, N.E. ; Akin, H.L.: Cooperative Multi-Robot Map Merging Using FastSLAM. En: *RoboCup 2009: Robot Soccer World Cup XIII* 5949 (2010), p. 449–460