



Máster Universitario en Visión Artificial

# SD-SLAM+: Mejora de un algoritmo de Visual SLAM mediante información de profundidad con cámara RGBD

Memoria del Trabajo Fin de Máster en Visión Artificial

Autor: Omar Garrido Martín

Tutores:

Jose María Cañas Plaza

Diego Martín Martín

Junio 2020





# **Resumen**

TODO

TODO

TODO

## RESUMEN

# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Visión artificial . . . . .	1
1.2. Visual SLAM . . . . .	4
1.2.1. Localización visual . . . . .	5
1.2.2. Mapeado . . . . .	6
1.3. Cámaras de profundidad . . . . .	7
1.3.1. Realsense D435 . . . . .	9
1.3.2. Ventajas de la información de profundidad en sistemas SLAM . . . . .	12
1.4. Estructura del documento . . . . .	12
<b>2. Objetivos</b>	<b>13</b>
2.1. Descripción del problema . . . . .	13
2.2. Objetivos . . . . .	15
<b>3. Estado del arte</b>	<b>17</b>
3.1. SLAM con información Visual (RGB) . . . . .	17
3.2. SLAM con información Visual (RGBD) . . . . .	17
3.3. SLAM con solo información de profundidad . . . . .	17
<b>4. Mejoras sobre SD-SLAM</b>	<b>19</b>
4.1. Introducción . . . . .	19
4.2. ROSificación de DIFODO . . . . .	19
4.3. Herramienta odometry_evaluation_file_creator . . . . .	20
4.4. SD-SLAM+: Integración de DIFODO sobre SD-SLAM . . . . .	21

## ÍNDICE GENERAL

---

<b>5. Experimentos</b>	<b>31</b>
5.1. El conjunto de datos . . . . .	31
5.2. Métricas de calidad . . . . .	32
5.3. Evaluación de DIFODO . . . . .	33
5.3.1. Efecto de la resolución de las imágenes de entrada . . . . .	34
5.3.2. Efecto del nivel de la pirámide . . . . .	36
5.3.3. Conclusiones . . . . .	39
5.4. Comparativa entre DIFODO y SD-SLAM . . . . .	39
5.4.1. Precisión de las estimaciones . . . . .	40
5.4.2. Conclusiones . . . . .	44
5.5. SD-SLAM frente a SD-SLAM+ . . . . .	44
5.5.1. El conjunto de datos . . . . .	45
5.5.2. Experimento: DIFODO como apoyo a SD-SLAM . . . . .	45
5.5.3. Experimento: Rendimiento de SD-SLAM+ en entornos difíciles . . . . .	52
5.5.4. Experimento: Rendimiento de SD-SLAM+ en entornos con zonas de baja textura . . . . .	62
5.5.5. Evaluación de tiempo de procesamiento . . . . .	64
5.5.6. Conclusiones . . . . .	66
<b>6. Conclusiones</b>	<b>67</b>
6.1. Objetivos conseguidos . . . . .	67
6.2. Discusión . . . . .	67
6.3. Trabajos futuros . . . . .	67
<b>Bibliografía</b>	<b>69</b>

# Índice de figuras

1.1.	La visión artificial es una disciplina que engloba a muchas otras. . . . .	2
1.2.	Histórico del error en la competición ImageNet . . . . .	3
1.3.	Aplicaciones de realidad aumentada que usan SLAM . . . . .	4
1.4.	Emparejamiento de puntos desde dos vistas de una cámara. . . . .	6
1.5.	Cierre de bucle: Comparativa antes y después del cierre de bucle . . . . .	7
1.6.	Imagen de profundidad y de color para una cámara realsense . . . . .	8
1.7.	Cámara D435 . . . . .	10
1.8.	Nube de puntos con textura . . . . .	10
1.9.	Cámara D435: <b>(a)</b> emisor infrarrojo activo, <b>(b)</b> emisor infrarrojo desactivado . .	11
2.1.	Momento en el cual la textura decae en la secuencia <b>rgbd_dataset_freiburg3_floor.bag</b> . Imagen perteneciente a la GUI de SD-SLAM+ . . . . .	14
2.2.	Efecto de el movimiento sobre las imágenes capturadas por una cámara. <b>(a)</b> imagen tomada con baja velocidad, <b>(b)</b> es el resultado de un rápido movimiento efectuado sobre la cámara . . . . .	15
4.1.	Máquina de estados de SD-SLAM . . . . .	22
4.2.	Máquina de estados de SD-SLAM+ . . . . .	23
4.3.	Cálculo del desplazamiento entre dos poses . . . . .	26
4.4.	Inversa de una pose . . . . .	27
4.5.	Dirección y sentido de los ejes de los sistemas de coordenadas . . . . .	28
4.6.	Adición del desplazamiento a la última pose conocida I . . . . .	29
4.7.	Adición del desplazamiento a la última pose conocida II . . . . .	30

## ÍNDICE DE FIGURAS

---

5.1. Trajetorias de los algoritmos para la secuencia <i>floor</i> . En verde se muestra la trayectoria verdadera, en azul la trayectoria estimada en cada caso. La figura (a) corresponde a la estimación de DIFODO y la figura (b) a la estimación de SD-SLAM. . . . .	42
5.2. SD-SLAM aplicado en <b>rgbd_dataset_freiburg1_desk.bag</b> . . . . .	46
5.3. Trajetorias de los algoritmos mostradas en el programa slam-testbed. En verde se muestra el <i>ground-truth</i> , en azul la trayectoria estimatda en cada caso. Las figura (a) corresponde a la versión original de SD-SLAM y la figura (b) a la versión modificada de SD-SLAM, SD-SLAM+ . . . . .	48
5.4. Trajetorias de los algoritmos mostradas en el programa slam-testbed. En verde se muestra el <i>ground-truth</i> , en azul la trayectoria estimatda en cada caso. Las figura (a) corresponde a la estimación de SD-SLAM+ en el test 2 y la figura (b) a la la estimación de SD-SLAM+ en el test 4. . . . .	52
5.5. SD-SLAM aplicado en <b>rgbd_dataset_freiburg3_nostructure_texture_far.bag</b>	53
5.6. Final de la secuencia <b>rgbd_dataset_freiburg3_nostructure_texture_far.bag</b> .	54
5.7. Trajetorias de los algoritmos mostradas en el programa slam-testbed. En verde se muestra el <i>ground-truth</i> , en azul la trayectoria estimatda en cada caso. Las figura (a) corresponde a la estimación de SD-SLAM y la figura (b) a la la estimación de SD-SLAM+. . . . .	56
5.8. Trajetorias de los algoritmos mostradas en el programa slam-testbed. En verde se muestra el <i>ground-truth</i> , en azul la trayectoria estimatda en cada caso. Las figura (a) corresponde a la estimación de SD-SLAM y la figura (b) a la la estimación de SD-SLAM+. . . . .	57
5.9. Trajetorias y reconstrucción de los algoritmos en la interfaz SD-SLAM. Las figura (a) corresponde a D-SLAM y la figura (b) a SD-SLAM+. . . . .	58
5.10. Trajetorias de los algoritmos mostradas en el programa slam-testbed. En verde se muestra el <i>ground-truth</i> , en azul la trayectoria estimatda en cada caso. Las figura (a) corresponde a la estimación de SD-SLAM+ para el test 6 y la figura (b) a la la estimación de SD-SLAM+ para el test 7. . . . .	60

5.11. Comparativa de las trayectorias de los algoritmos mostradas en el programa slam-testbed. En verde se muestra el <i>ground-truth</i> , en azul la trayectoria esti- matda en cada caso. . . . .	61
5.12. Momento en el cual la textura decae en la secuencia <b>rgbd_dataset_freiburg3_floor.bag</b> . Imagen perteneciente a la GUI de SD-SLAM+ . . . . .	62
5.13. Comparativa de las trayectorias y reconstrucción de los algoritmos SD-SLAM y SD-SLAM+. . . . .	64
5.14. Comparativa del tiempo de procesamiento por imagen para la secuencia <b>rgbd_dataset_freiburg3_flo</b>	

## ÍNDICE DE FIGURAS

---

# Índice de tablas

5.1.	Tiempos de procesamiento para distintas resoluciones y distintos procesadores . . . . .	35
5.2.	RMSE en metros para el ATE (Absolute Trajectory Error) en distintas resoluciones de DIFODO. . . . .	36
5.3.	Tiempos de procesamiento para distintos niveles de pirámide en AMD-FX 8370 . . . . .	37
5.4.	Comparación de tiempos de procesamiento para distintos niveles de pirámides y procesadores. . . . .	38
5.5.	RMSE en metros para el ATE (Absolute Trajectory Error) en distintos niveles de pirámide de DIFODO. . . . .	39
5.6.	Comparación del ATE (Absolute Trajectory Error) en metros para DIFODO y SD-SLAM en secuencias de Freiburg 1. . . . .	40
5.7.	Comparación del ATE en metros para DIFODO y SD-SLAM en secuencias de Freiburg 3: Estructura vs Textura. En verde el menor error en cada secuencia. . . . .	43
5.8.	Comparación del ATE en el Test 1: SD-SLAM vs SD-SLAM+ . . . . .	46
5.9.	Comparación del ATE en el test 2: SD-SLAM vs SD-SLAM+ . . . . .	49
5.10.	Test 2: Evolución del ATE en SD-SLAM+ . . . . .	49
5.11.	Comparación del ATE en el test 3: SD-SLAM vs SD-SLAM+ . . . . .	50
5.12.	Test 3: Evolución del ATE en SD-SLAM+ . . . . .	50
5.13.	Comparación del ATE en el Test 4: SD-SLAM vs SD-SLAM+ . . . . .	51
5.14.	Test 4: Evolución del ATE en SD-SLAM+ . . . . .	51
5.15.	Comparación del ATE en el test 5: SD-SLAM vs SD-SLAM+ . . . . .	54
5.16.	Comparación del ATE en el test 6: SD-SLAM vs SD-SLAM+ . . . . .	56
5.17.	Test 6: Evolución del ATE en SD-SLAM+ . . . . .	59
5.18.	Comparación del ATE en el test 7: SD-SLAM vs SD-SLAM+ . . . . .	59

## ÍNDICE DE TABLAS

---

5.19. Test 8: Evolución del ATE en SD-SLAM+ . . . . .	61
5.20. Comparación del ATE en el test 9: SD-SLAM vs SD-SLAM+ . . . . .	63

# **Capítulo 1**

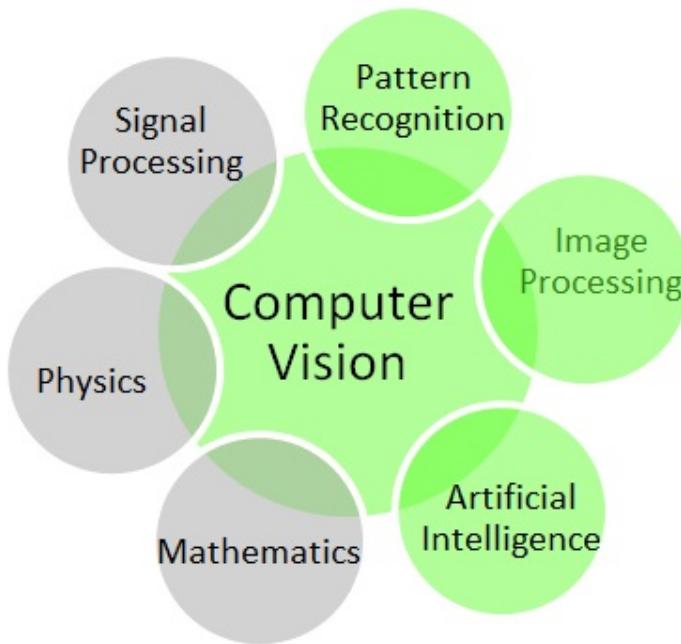
## **Introducción**

En este capítulo se hace un repaso por la historia de la visión artificial. Después se habla de la técnica SLAM(localización y mapeado simultaneo en español), muy utilizada en robótica y que entra dentro de la visión artificial. Finalmente se explicará que son las cámaras de profundidad y porqué son de importancia en este proyecto. El objetivo de este capítulo es el de introducir una serie de conceptos básicos al lector para facilitar la lectura.

### **1.1. Visión artificial**

La visión artificial es una disciplina científica que busca la obtención de información a través del análisis de la imagen. La visión artificial es un campo enorme que incluye diversos subcampos, que van desde la parte más física como puede ser: el estudio de la formación de la imagen, el desarrollo de sensores para la adquisición de imagen, sistemas de iluminación, etc., hasta la parte más cercana a la computación software como es el desarrollo de algoritmos para el procesado y análisis de la imagen.

Esta es a su vez, un subcampo de la inteligencia artificial, entendiéndose la inteligencia artificial como la idea de que las máquinas sean capaces de pensar como los humanos. La visión artificial esta formada o utiliza otros campos como son: la matemática, estadística, informática, aprendizaje máquina, procesamiento de imagen, procesamiento de señal...entre muchos otros, figura 1.1.



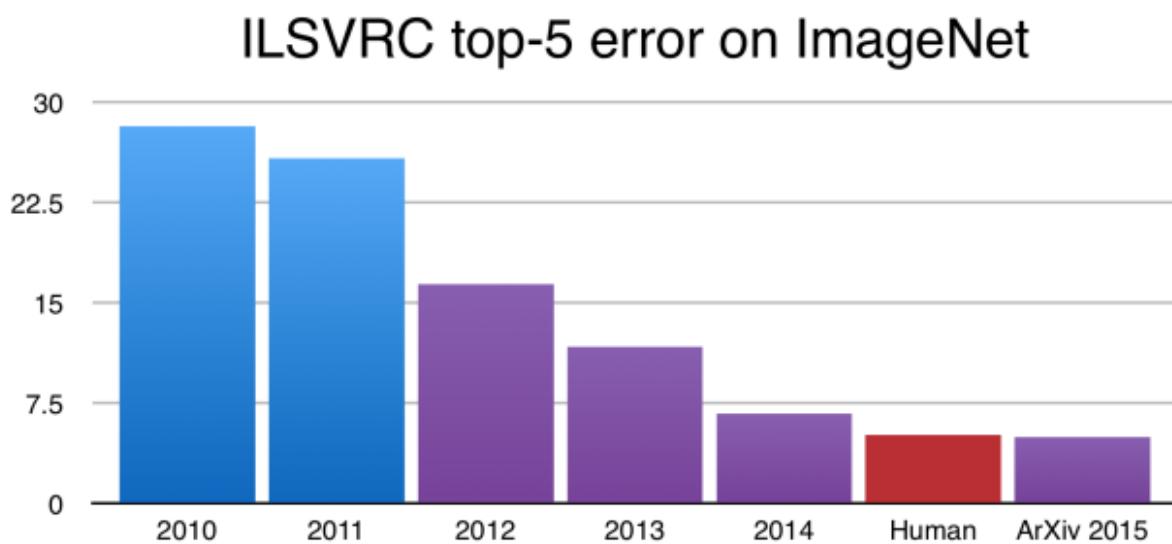
**Figura 1.1:** La visión artificial es una disciplina que engloba a muchas otras.

La visión artificial nace en la década de los 60, en las universidades, como un subcampo de la inteligencia artificial. Ambas tuvieron un fuerte empuje por parte tanto de la comunidad científica como de inversión pública-privada, debido al interés que generó y se consiguieron muchísimos avances que a día de hoy son los fundamentos de la visión artificial que conocemos.

En la década de los 80 este campo se ve empujado por el desarrollo de la ingeniería informática y la creación de procesadores más rápidos, que permiten captar, procesar y reproducir imágenes tomadas por una cámara que podía estar conectada de forma remota. Ambos campos, la visión artificial y la informática, están estrechamente relacionadas pues el cómputo de imagen es muy costoso computacionalmente ya que la imagen es una estructura de información de gran tamaño en comparación con otro tipo de señal, por lo que es necesario la optimización de los algoritmos desarrollados de visión.

La visión artificial, cae un poco en el olvido con el denominado invierno de la Inteligencia Artificial hacia la década de los 90. Aunque esto no significa que el campo pare su avance y muchos desarrollos importantes se dieron durante este tiempo, sobre todo en el aprendizaje máquina, un campo del cual se alimenta la visión artificial. Ya por estas fechas, se considera un campo mucho más maduro y se utiliza con éxito en el ámbito industrial para automatizar tareas en la industria como por ejemplo el análisis por imagen de calidad en piezas fabricadas.

2012 es el año en que AlexNet, una red neuronal convolucional profunda, gana la competición *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)*, la mayor competición de reconocimiento de objetos en el mundo de la visión artificial. Sin embargo el mayor logro no fue que ganase la competición, sino de que lograse tan solo un 15 % de error frente al 25 % del año anterior, como muestra la figura 1.2. Tan solo unos años después, en 2015, se consigue superar la precisión que tiene un ser humano, marcando todo un hito.



**Figura 1.2:** Histórico del error en la competición ImageNet

AlexNet supone el inicio de la época dorada de las redes neuronales profundas hasta el día de hoy, con las cuales se han conseguido verdaderos logros en numerosas aplicaciones y campos como: El coche autónomo, avances en imagen de diagnóstico médico, sistemas biométricos como el reconocimiento facial, entendimiento del entorno a nivel semántico, etc., entre muchas otras.

Sin embargo, no todo se soluciona actualmente con *deep learning*, donde queda mucho por explorar, y todavía hay problemas como SLAM, donde se siguen usando los métodos tradicionales o mejor dicho donde el *deep learning* todavía no ha presentado mejoras respecto a estos.

## 1.2. Visual SLAM

SLAM es las siglas de **Localización y Mapeado Simultáneo** en inglés. Es uno de los problemas abiertos más importantes de la robótica y la visión artificial si hablamos de SLAM visual. El objetivo con el que nace el SLAM visual es la navegación autónoma de robots. Si se cuenta con un mapa del entorno sobre el que el robot va a navegar, es relativamente sencillo localizar a este en ese entorno y establecer rutas de un punto a otro. El problema llega cuando no existe un mapa existente del entorno sobre el que se pretende navegar. Es entonces cuando nace la necesidad de realizar la tarea de la localización y el mapeado de forma simultánea. Esto se acaba traduciendo en sistemas que precisan de ejecutarse en tiempo real, añadiendo más complejidad a un problema ya de por sí difícil.

Por el camino los algoritmos de SLAM también han mostrado esfuerzos en la reconstrucción de entornos 3D, aunque para reconstrucciones muy detalladas se sigue utilizando lo que se conoce como *Structure from Motion* [11], estructura a partir del movimiento. La diferencia entre el SFM y SLAM es que SFM trabaja con imágenes desordenadas, idealmente distintas vistas del objeto a reconstruir, y es un proceso costoso y que se realiza de forma *offline*, llegando a tardar horas o días, por otro lado visual SLAM trabaja con imágenes consecutivas, ordenadas temporalmente, y precisa de funcionar en tiempo real.

Si bien la navegación robótica fue la que inició el desarrollo de algoritmos de SLAM, en los últimos años campos como la realidad virtual/aumentada y la reconstrucción 3D de objetos han sido los grandes promotores de los últimos avances en SLAM. Distintas sectores como el de los videojuegos ha visto un aumento en las aplicaciones desarrolladas que utilizan SLAM, pero también en sectores más tradicionales como la decoración y sectores industriales, figura 1.3, entre muchos otros.



**Figura 1.3:** Aplicaciones de realidad aumentada que usan SLAM

### 1.2.1. Localización visual

La diferencia entre la localización en un mapa conocido con la localización cuando no se cuenta con mapa, es que en el primer caso, se utiliza la información del mapa para determinar la posición y orientación dentro de este y es independiente de posiciones anteriores, mientras que en el caso de no contar con un mapa lo que se calcula es el desplazamiento relativo con la posición anterior, por lo que hay una dependencia temporal entre estimaciones a diferencia de cuando se cuenta con un mapa.

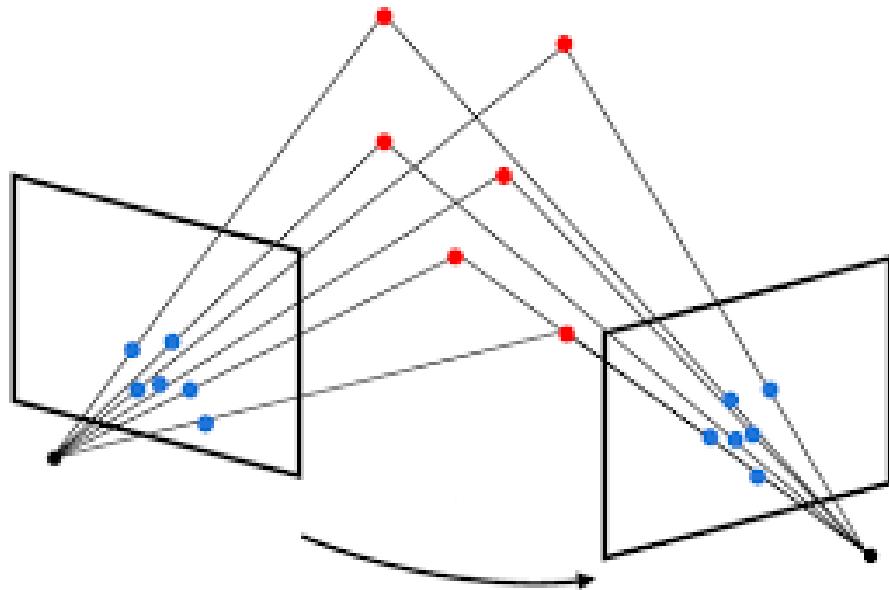
La parte correspondiente a la localización en SLAM se denomina odometría, cuando la información es visual se le denomina odometría visual. El término de odometría surge en el campo de la robótica, refiriéndose a la estimación incremental o desplazamiento de un robot sobre ruedas, utilizando para este cálculo la geometría del robot. Se habla de odometría visual por primera vez en 2004 [4], en este caso en lugar de la geometría del robot se utilizan imágenes adquiridas por cámaras para esta estimación de pose incremental.

Cuando se utiliza una única cámara para realizar la odometría visual se habla entonces de visión monocular. Para poder realizar la estimación de la pose se precisa de observaciones consecutivas de la misma cámara, obteniendo diferentes perspectivas del mismo entorno, simulándose la captura con numerosas cámaras y perspectivas utilizada en el *Structure from Motion*. Al contar con numerosas perspectivas de la misma escena se puede obtener tanto la reconstrucción 3D como la posición de cada perspectiva, que es el objetivo de los algoritmos de SLAM. El único inconveniente de esta técnica es que no permite obtener una escala real, por lo que los objetos reconstruidos así como los desplazamientos se encontrarán en una escala aleatoria, que no sigue el sistema métrico utilizado normalmente. Este problema de la escala se soluciona con la inclusión de información de profundidad con los nuevos sensores RGBD de los cuales se habla más adelante. Los algoritmos de odometría visual se pueden dividir en tres categorías:

- **Métodos directos:** Los métodos directos utilizan todo la información de intensidad de los píxeles de la imagen en la estimación del desplazamiento. Esto les permite ser más robustos que los métodos basados en puntos característicos en casos donde hay falta de textura.
- **Métodos basados en puntos característicos:** Estos métodos buscan puntos característicos en la escena, normalmente puntos muy ricos en textura y que son muy representativos.

tativos y cuentan con mucha información. Se hace un seguimiento de estos puntos a lo largo de las distintas imágenes, mediante técnicas de emparejamiento. Con un número suficiente de puntos emparejados, figura 1.4, se busca estimar la pose de la cámara y la pose 3D de los puntos emparejados mediante la reducción del error de reproyección. El algoritmo más usado para esta optimización y estimación de los parámetros es *Bundle Adjustment* [10].

- **Métodos híbridos:** Aquellos métodos que combinan tanto métodos directos como basados en características [7].

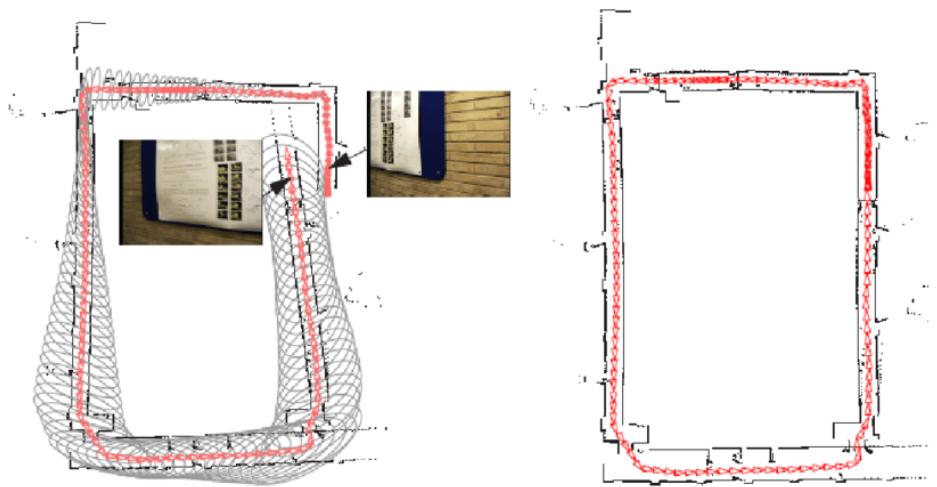


**Figura 1.4:** Emparejamiento de puntos desde dos vistas de una cámara.

### 1.2.2. Mapeado

El mapeado consiste en la creación de un mapa tridimensional, enfocado en aportar la información necesaria al algoritmo de SLAM para permitirle localizarse. Si bien la odometría visual no precisa de un mapa como tal, el mapeado es una característica de SLAM que le permite utilizar esta información para ser más preciso y robusto en la estimación de la pose. El contar con información de puntos anteriores le permite estimar mejor las nuevas posiciones, reduciendo la deriva incremental del error que se produce con cada nueva estimación en la odometría, por

otro lado, una de las características claves de los algoritmos de SLAM es la posibilidad de relocalizarse en caso de perderse y también el cierre de bucle. El cierre de bucle es una técnica que consiste en la optimización global de la trayectoria al pasar por una zona ya visualizada previamente y de la que se tiene información visual en el mapa. Al optimizar la trayectoria también se optimiza al mismo tiempo la reconstrucción 3D por lo que esta técnica es muy útil y favorece la robustez de los algoritmos de SLAM. En la figura 1.5, se aprecia la deriva o acumulación del error en la odometría y como el cierre de bucle optimiza la trayectoria.



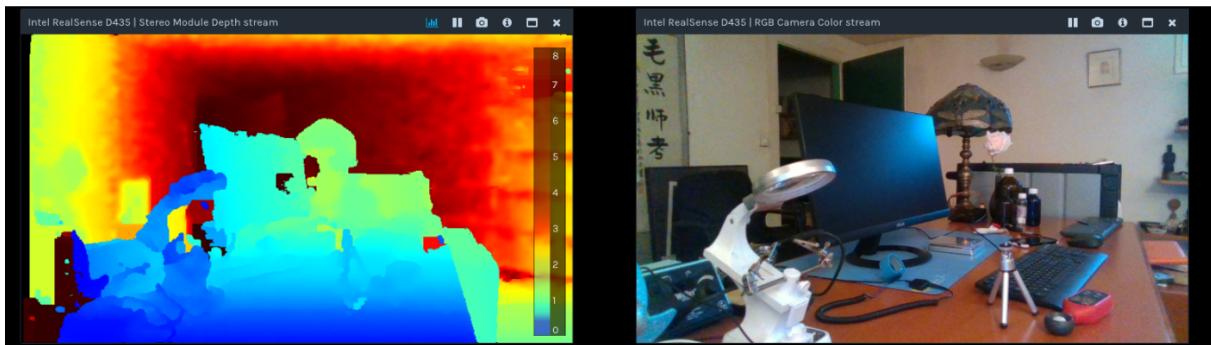
**Figura 1.5:** Cierre de bucle: Comparativa antes y después del cierre de bucle

### 1.3. Cámaras de profundidad

En la realización de este trabajo se ha utilizado una cámara que pertenece a la categoría de cámara de profundidad. Una cámara de profundidad es toda cámara que es capaz de obtener una imagen de profundidad, es decir, una imagen donde el valor de cada uno de sus píxeles corresponde a la distancia entre la cámara y el objeto que se encuentra en ese píxel. Normalmente la representación visual de estas imágenes se hace con colores que van desde los tonos fríos, para objetos cercanos, a los tonos calientes para los objetos más alejados, como se puede apreciar en la figura 1.6.

### 1.3. CÁMARAS DE PROFUNDIDAD

---



**Figura 1.6:** Imagen de profundidad y de color para una cámara realsense

Dentro de esta categoría de cámaras de profundidad, se encuentran distintos tipos de cámaras en función de su funcionamiento físico a la hora de obtener la imagen de profundidad de la escena. Las cámaras de profundidad se agrupan en:

- **Cámaras estereoscópicas:** Las cámaras estereoscópicas se caracterizan por poseer dos lentes, simulando al ojo humano. Conocida la geometría entre las dos cámaras se puede crear un mapa de disparidad, que permite conocer a partir de dos imágenes RGB la distancia a cada uno de los píxeles. En esta categoría se puede englobar tanto a las cámaras con dos lentes, como a varias cámaras con captura sincronizada y cuya posición relativa es conocida. También se puede incluir en esta categoría a una única cámara que se desplaza entre tomas, esta última sería el caso de SLAM monocular.
- **Escáneres de luz estructurada:** Este tipo de cámaras forman parte de los sensores activos, ya que a diferencia de los anteriores, estos actúan sobre la imagen, en este caso proyectando un patrón de luz conocido. Este patrón de luz visible o no visible en el caso del infrarrojo, permite que la cámara añada más información a la escena que le permita realizar la reconstrucción 3D. Dependiendo del uso de este patrón de luz encontramos varias categorías:
  1. **Par de cámaras estéreo y luz estructurada visible:** El computo del mapa de disparidad se ve mejorado utilizando un patrón de luz estructurada que permite hacer una buena reconstrucción 3D incluso en zonas donde se presentan colores homogéneos, donde los sistemas estéreo basados solo en RGB fallan.
  2. **Triangulación activa:** La proyección de un plano de luz que barre el objeto y la

toma de imágenes permite hacer una reconstrucción 3D del objeto gracias a la deformación del plano de luz al incidir sobre el objeto y la triangulación usando la geometría cámara-emisor de luz.

3. **Una cámara RGB y patrón de luz estructurada infrarroja:** El modelo de cámara representativo de este categoría es el *Microsoft Kinect™ V1*. Un dispositivo inicialmente pensado para videojuegos pero que permitió a la comunidad científica hacerse con un sensor de bajo coste, con una capacidad de obtener imágenes de profundidad de gran calidad. En este caso, se cuenta con un proyector de luz infrarroja, un receptor de luz infrarroja y una cámara RGB. El patrón de luz estructurada también es conocido y es usado directamente para la creación del mapa de disparidad, sin necesidad de usar un par estéreo pues el proyector de luz estructurada se modela como una lente, sustituyendo a una de las cámaras del par estéreo.
  - **TOF (Time of Flight o tiempo de vuelo):** Las cámaras de tiempo de vuelo son el último avance en cuanto a imágenes de profundidad. De nuevo, se cuenta con la proyección de un patrón de luz, típicamente infrarrojo, pero en este caso, lo que se mide es el tiempo que tarda la luz en rebotar contra un objeto y volver a la cámara. Conocida la velocidad de la luz y el tiempo que ha tardado ese rayo en volver desde que salió del proyector IR se puede obtener la distancia del objeto contra el que ese rayo rebotó. En este grupo se encuentra la segunda generación del Kinect, *Microsoft Kinect™ V2*.

### 1.3.1. Realsense D435

La cámara realsense D435 de Intel es la cámara utilizada a lo largo de este proyecto. Esta cámara cae en la categoría de cámaras de luz estructurada usando un par estéreo. En concreto el dispositivo cuenta con tres cámaras, dos infrarrojas y una de color, y un emisor del patrón de luz estructurada infrarroja.

Las características de este sensor y su bajo coste, así como tamaño, lo hacen ideal para proyectos de robótica, siendo un sensor que podría usarse para vehículos no tripulados como es el caso de los drones.

De entre las características a destacar de este sensor, figura 1.7, destaca su pequeño tamaño

### 1.3. CÁMARAS DE PROFUNDIDAD

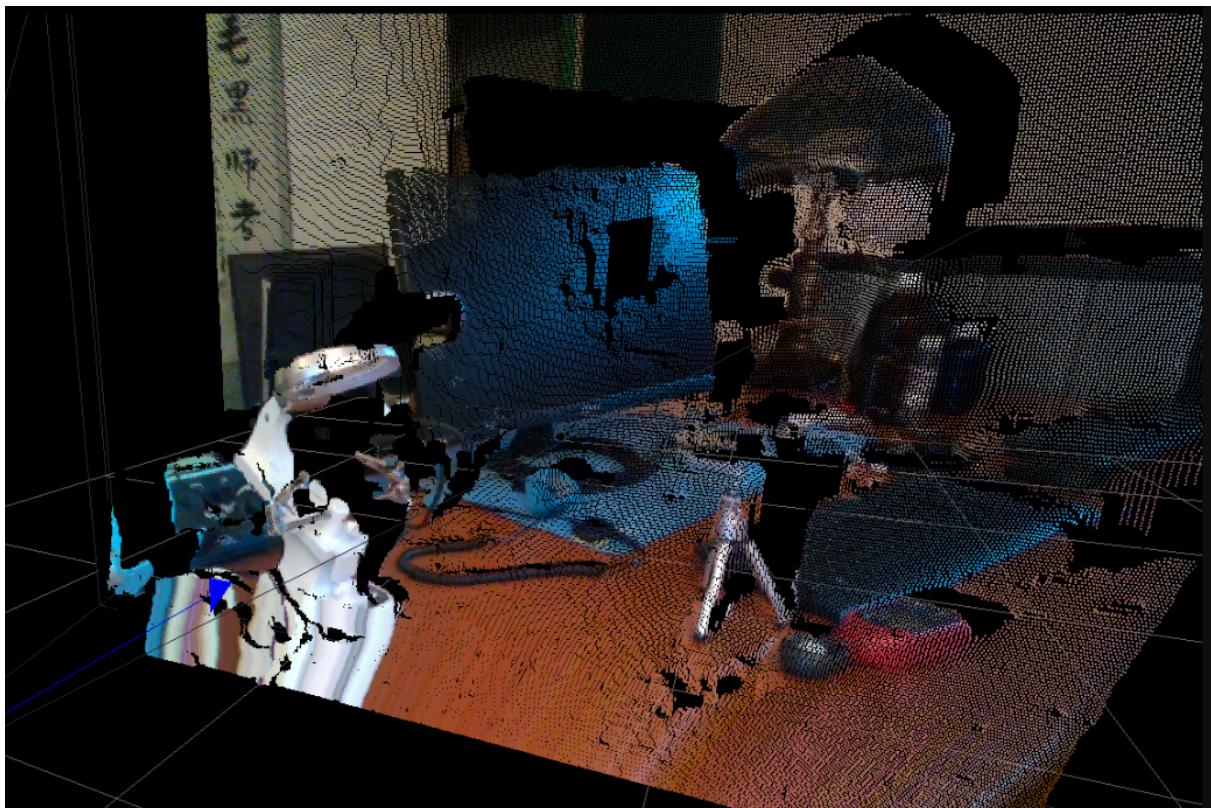
---

de tan solo 90x25x25mm y un peso de 72g. En el apartado de imagen es capaz de ofrecer hasta 90 imágenes de profundidad por segundo a baja resolución y, de hasta 30 imágenes por segundo en calidad HD. En el apartado de imagen a color la resolución aumenta hasta *full HD*.



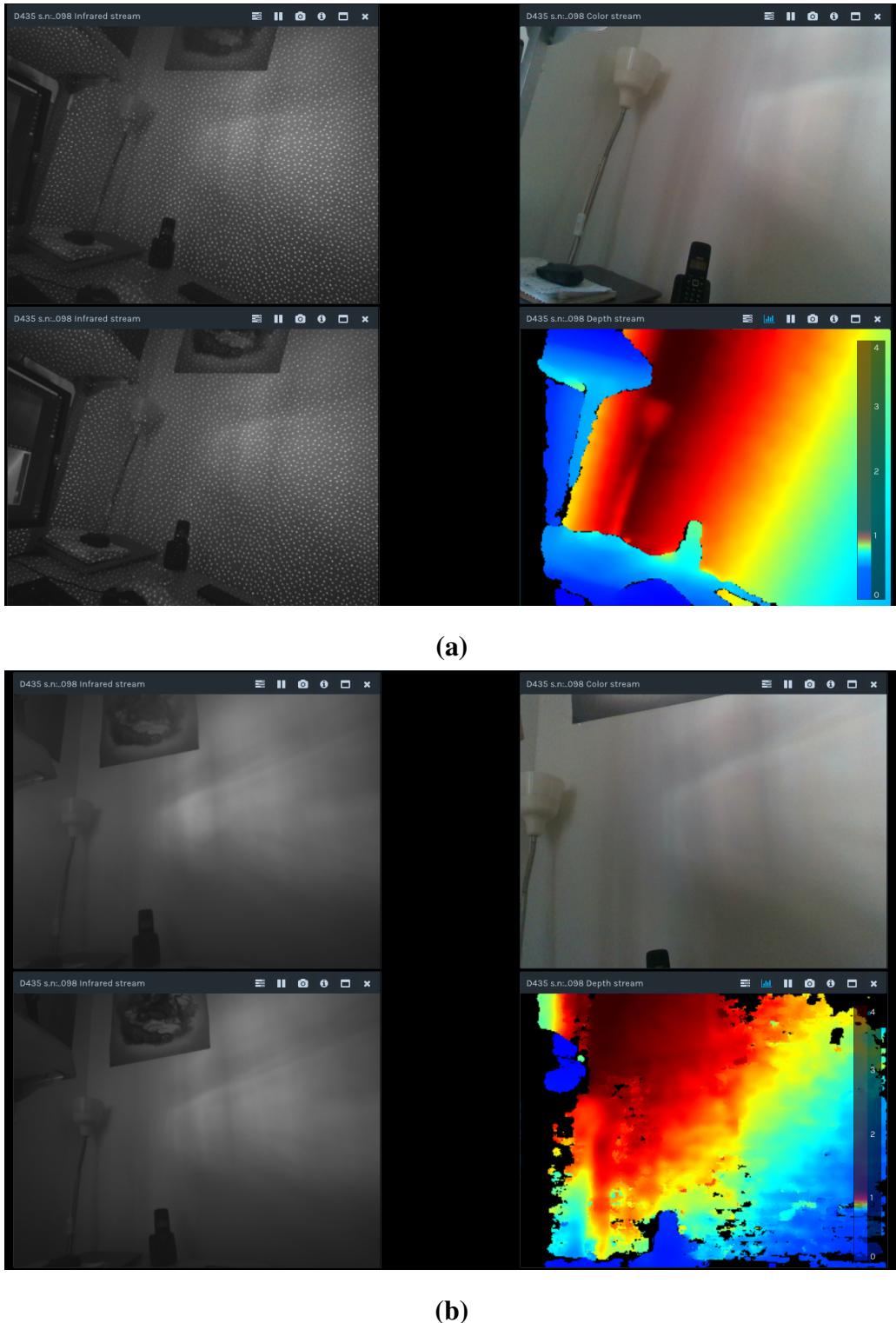
**Figura 1.7:** Cámara D435

El mapa de profundidad se crea a partir de la información del infrarrojo de la escena y del patrón emitido. Por otro lado, se cuenta con la imagen a color, que es independiente de los otros sensores en su funcionamiento. Solo se combina la información de la imagen de profundidad y la de color cuando se registran ambas imágenes, pudiendo obtener una nube de puntos 3D con información de color, como se aprecia en la figura 1.8.



**Figura 1.8:** Nube de puntos con textura

El efecto de crear un mapa de disparidad, con solo la información de la propia escena y sin aplicar el patrón sobre esta o aplicando de forma activa un patrón, se puede observar en la figura 1.9.



**Figura 1.9:** Cámara D435: **(a)** emisor infrarrojo activo, **(b)** emisor infrarrojo desactivado

## 1.4. ESTRUCTURA DEL DOCUMENTO

---

En **(b)** se observa como la imagen de profundidad presenta alteraciones y zonas donde la profundidad no se ha podido recuperar correctamente. La presencia del patrón de luz infrarrojo sobre la escena que se aprecia en **(a)**, favorece enormemente la generación del mapa de profundidad.

### 1.3.2. Ventajas de la información de profundidad en sistemas SLAM

Estas cámaras, que normalmente aportan también información de color, presentan una serie de ventajas para los algoritmos de SLAM. Entre estas ventajas destaca la posibilidad de contar con escala verdadera. Por otro lado gracias a esta nueva información, se puede optimizar algunos procesos de algoritmos existentes basados tan solo en información RGB, haciendo que estos sean computacionalmente más eficientes, por ejemplo al utilizar la información de profundidad en el paso del *Bundle Adjustment*. Si en BA se tiene una inicialización en el valor de la profundidad de cada uno de los puntos, se están eliminando parámetros de la ecuación a resolver, permitiendo que el algoritmo pueda converger más rápido. Otra mejora inmediata es la posibilidad de obtener mapas densos, al no tener que triangular cada uno de los puntos de la escena o crear un mapa de disparidad, procesos computacionalmente costosos, sino simplemente recuperando esa información del sensor de forma inmediata. Gracias a esta nueva información de profundidad también empiezan a surgir algoritmos de SLAM híbridos que utilizan puntos característicos 3D y 2D en el cálculo de la odometría, así como algoritmos basados solo en la información de profundidad.

## 1.4. Estructura del documento

El presente documento está dividido en secciones para favorecer su lectura y entendimiento. Tras esta primera introducción para poner en conocimiento del lector conceptos básicos que le permitirán seguir el desarrollo de la memoria con mayor facilidad, se procede a una sección donde se enumeran los objetivos a lograr. Tras una revisión sobre el estado del arte, se pasa al desarrollo de la mejora del algoritmo SD-SLAM. Finalmente se presentan una serie de experimentos que sirven para validar y demostrar el funcionamiento del nuevo algoritmo desarrollado. La última sección corresponde a las conclusiones a las que se han llegado así como posibles mejoras futuras que se pueden explorar para continuar con la mejora de SD-SLAM.

# **Capítulo 2**

## **Objetivos**

En esta sección se describe el problema que se busca resolver, y se establecen unos objetivos para lograrlo.

### **2.1. Descripción del problema**

Como se ha visto en el anterior capítulo, los algoritmos de SLAM buscan la creación de un mapa representativo del entorno que le rodea así, como una estimación de la posición que se ocupa dentro de este. Para lograr esto es necesario utilizar algún tipo de información del entorno que permita representarlo, normalmente esta información es visual y se obtiene mediante cámaras.

Entre estos algoritmos de SLAM se encuentra SD-SLAM [5], desarrollado por el grupo de robótica de la universidad Rey Juan Carlos I, JdeRobot<sup>1</sup>. Este algoritmo es una mejora sobre el algoritmo de SLAM, ORB-SLAM2 [3]. Ambos son capaces de funcionar con información únicamente visual, RGB, y también con información de profundidad, RGBD. Sin embargo, como la mayoría de algoritmos de SLAM que utilizan información RGBD, estos solo utilizan la parte de la profundidad para obtener escala real en el mapa y facilitar las operaciones de triangulación al conocer la posición exacta 3D de los puntos. En general incorporar esta información hace a estos algoritmos más eficientes computacionalmente y les proporciona una mejor estimación de la pose.

---

<sup>1</sup><https://jderobot.github.io>

## 2.1. DESCRIPCIÓN DEL PROBLEMA

---

Sin embargo, existen momentos donde estos algoritmos no funcionan adecuadamente, llegando incluso a no poder dar una estimación de la pose ni de los puntos 3D de la escena en el mapa. Esto ocurre ante la falta de textura en la imagen. La textura en una imagen viene definida por la cantidad de bordes y esquinas que hay en la escena. Si es una escena que presenta homogeneidad en la intensidad de sus píxeles, se dice que hay poca textura. Un ejemplo real donde SD-SLAM se pierde y no consigue estimar la pose se puede apreciar en la figura 2.1.



**Figura 2.1:** Momento en el cual la textura decae en la secuencia `rgbd_dataset_freiburg3_floor.bag`. Imagen perteneciente a la GUI de SD-SLAM+

Estas escenas sin textura pueden darse debido a la homogeneidad en el color de los objetos como es el caso anterior, o también se puede producir cuando se hacen movimiento rápidos o bruscos, produciendo un efecto de emborronamiento en la imagen que imposibilita la percepción de los puntos característicos como se aprecia en la figura 2.2. Cuando la imagen sufre esta degradación, los bordes de la escena desaparecen, difuminándose la separación entre los distintos objetos de la escena. Otro caso, si bien el más extremo y quizás el menos común donde se puede observar perdida de textura, se da cuando la iluminación de la escena no es suficiente o directamente no hay luz.



**Figura 2.2:** Efecto de el movimiento sobre las imágenes capturadas por una cámara. **(a)** imagen tomada con baja velocidad, **(b)** es el resultado de un rápido movimiento efectuado sobre la cámara

## 2.2. Objetivos

En definitiva la perdida de textura en la imagen puede producirse por diversas causas produciendo el fallo de los algoritmos de SLAM visual como es el caso de SD-SLAM. Este trabajo busca precisamente evitar esto. En concreto, el objetivo principal será el de evitar que la odometría o estimación de la pose de SD-SLAM falle debido a la falta de textura. Independientemente de como se haya producido, la falta de textura imposibilita la obtención de puntos característicos sobre la imagen RGB, sin embargo la imagen de profundidad todavía aporta mucha información que puede ser usada para conseguir la odometría. Los objetivos de este trabajo son:

- **Evitar que SD-SLAM falle al estimar pose cuando la textura en la imagen es escasa.**
- **Buscar un algoritmo basado en solo información de profundidad que permita la odometría.**
- **Integración de SD-SLAM con un algoritmo de odometría basado solo en profundidad.**
- **Tiempo de procesamiento real:** SD-SLAM es un algoritmo que se caracteriza por poder funcionar en tiempo real, el nuevo algoritmo habrá de seguir manteniendo esta característica.

## **2.2. OBJETIVOS**

---

# **Capítulo 3**

## **Estado del arte**

TODO: Que se va a analizar, de que va la sección

### **3.1. SLAM con información Visual (RGB)**

### **3.2. SLAM con información Visual (RGBD)**

### **3.3. SLAM con solo información de profundidad**

Llegar a DIFODO, y exponer que parece que cumple con todos los requisitos para ser un algoritmo a integrarse con SD-SLAM pero solo nombrar esto pues, es el estado del arte esta sección

### 3.3. SLAM CON SOLO INFORMACIÓN DE PROFUNDIDAD

---

# Capítulo 4

## Mejoras sobre SD-SLAM

A lo largo de esta sección, se van a describir los distintos desarrollos realizados a lo largo de este trabajo de fin de máster.

### 4.1. Introducción

En este capítulo se va a desarrollar fundamentalmente, la integración de el algoritmo DIFODO junto a SD-SLAM para crear un algoritmo nuevo, denominado SD-SLAM+, cuyas características mejoren a las de su predecesor.

Durante la realización de este trabajo ha sido preciso el diseño de otros programas de software, de entre los cuales merece la pena destacar dos de ellos, por su importancia en el desarrollo de este trabajo.

### 4.2. ROSificación de DIFODO

Como se ha descrito en el estado del arte, en la sección 3.3, DIFODO es un algoritmo de odometría visual basado únicamente en información de profundidad. A priori, por los resultados presentados en el artículo de DIFODO [2], este parece cumplir con los requisitos para integrarse con SD-SLAM, que eran procesamiento en tiempo real y odometría con solo información de profundidad.

La *rosificación* de DIFODO, hace referencia al proceso de integración de el algoritmo de DI-

#### 4.3. HERRAMIENTA ODOMETRY\_EVALUATION\_FILE\_CREATOR

---

FODO en ROS (Robot Operating System)<sup>1</sup>. El primer paso hacia la integración, es verificar experimentalmente que este algoritmo, de verdad es capaz de estimar con precisión y en tiempo real la pose a partir de imágenes de profundidad.

SD-SLAM es un algoritmo que es compatible con ROS (Robot Operating System), lo que le permite un acceso a multitud de herramientas así como distintos *datasets* internacionales. ROS tiene la ventaja de permitir utilizar distintos conjuntos de datos de una forma sencilla, sin necesidad de tener que hacer un desarrollo específico para adaptarse al formato de cada uno de esos conjuntos de datos. De entre los distintos conjuntos de datos, se encuentra **RGB-D SLAM Dataset and Benchmark** [9], el cual cuenta con unas características que lo hacen idóneo para evaluar todos los experimentos que se realizarán durante este trabajo. Los detalles sobre este conjunto de datos se encuentran en la sección 5.1.

Debido a que el conjunto de datos a usar se encuentra en un formato compatible con ROS y también a que SD-SLAM está *rosificado*, surge la necesidad de *rosificar* DIFODO con el objetivo de evaluarlo y facilitar su posterior integración con SD-SLAM.

Por estas razones y para evaluar DIFODO, se ha creado un software que permite usar DIFODO como un nodo de ROS. La implementación es pública y se encuentra disponible en un repositorio en github<sup>2</sup>.

### 4.3. Herramienta odometry\_evaluation\_file\_creator

Una de las peculiaridades del conjunto de datos elegido, es que para cada secuencia se establece un sistema de coordenadas en un punto de la escena que no tiene que coincidir con el inicio de la primera imagen de esta. Por otro lado, estas secuencias no presentan información visual durante los primeros cuatro segundos. Esto se realiza con el objetivo de que la transmisión de datos sea constante, permitiendo a la cámara un tiempo de calentamiento. Todo esto, produce que la estimación de pose de algoritmos como DIFODO y SD-SLAM, no se pueda comparar directamente con la pose verdadera que provee el *dataset*, ya que hay un desplazamiento en translación y rotación entre el origen de coordenadas que usan las poses verdaderas y las de las estimaciones de los algoritmos DIFODO y SD-SLAM.

---

<sup>1</sup><https://www.ros.org/>

<sup>2</sup>[https://github.com/RoboticsLabURJC/2019-tfm-omar-garrido/tree/master/code/roify\\_difodo](https://github.com/RoboticsLabURJC/2019-tfm-omar-garrido/tree/master/code/roify_difodo)

Por otro lado, a la hora de comparar las estimaciones de pose con la pose verdadera, las distintas herramientas de comparación precisan que los sellos temporales, instante de tiempo preciso en el que se toma cada imagen, sean los mismos entre las poses a comparar. En concreto las secuencias usadas se grabaron en los años 2011-2012, por lo que los sellos temporales datan de esa fecha y no corresponden con el de nuevas estimaciones. En conclusión, las poses verdadera y absoluta precisan de estar registradas tanto temporalmente como espacialmente.

Para poder evaluar el rendimiento de DIFODO y SD-SLAM así como el nuevo SD-SLAM+, se ha desarrollado esta herramienta<sup>3</sup> que permite de forma automática obtener la relación de transformación entre el sistema de coordenadas del conjunto de datos y de el algoritmo siendo evaluado y transformar el de este último al del conjunto de datos. Esta herramienta, genera un fichero de estimaciones de pose en el mismo formato usado por el conjunto de datos **RGB-D SLAM Dataset and Benchmark** permitiendo la comparación de ambas trayectorias, la estimada y la verdadera.

## 4.4. SD-SLAM+: Integración de DIFODO sobre SD-SLAM

Una vez se ha comprobado que DIFODO es un algoritmo capaz de funcionar en tiempo real en la sección 5.3 y de comprobar que es capaz de funcionar mejor que SD-SLAM en aquellas secuencias o momentos donde no hay suficiente textura en la sección 5.4, el diseño y desarrollo del nuevo SD-SLAM+ comienza.

Los objetivos que se buscan lograr con este nuevo algoritmo es conseguir que SD-SLAM no pierda la estimación de pose cuando no haya suficiente textura en la escena. Para ello se va a integrar DIFODO en SD-SLAM, con el objetivo de que cuando SD-SLAM no sea capaz de estimar una nueva pose, se empiece a usar la odometría de DIFODO hasta que vuelva a haber suficiente textura en la imagen y se pueda recuperar la odometría visual original de SD-SLAM. Una vez se han definido los objetivos a lograr y la lógica general de funcionamiento del nuevo algoritmo SD-SLAM+, es necesario bajar al detalle de la programación y ver cómo funciona el algoritmo actual.

El funcionamiento del algoritmo original SD-SLAM se puede resumir con una máquina de estados como muestra la figura 4.1. El algoritmo original precisa de una inicialización, que

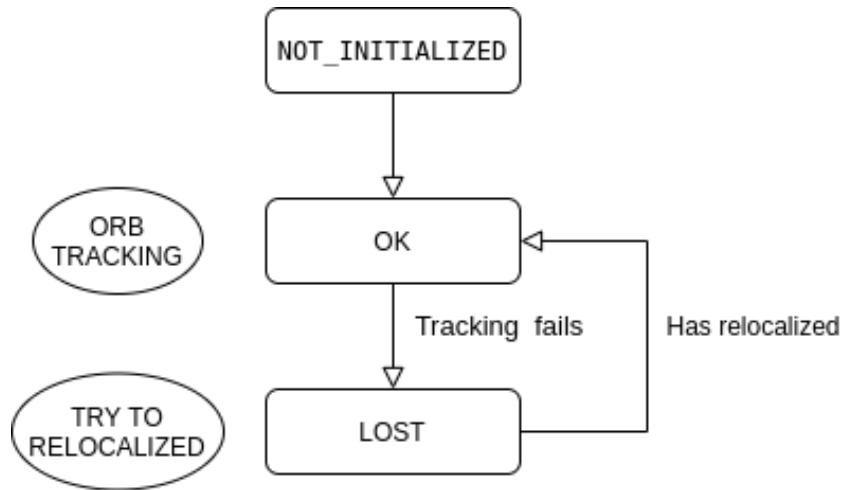
---

<sup>3</sup>[https://github.com/RoboticsLabURJC/2019-tfm-omar-garrido/tree/master/code/odometry\\_evaluation\\_file\\_creator](https://github.com/RoboticsLabURJC/2019-tfm-omar-garrido/tree/master/code/odometry_evaluation_file_creator)

#### 4.4. SD-SLAM+: INTEGRACIÓN DE DIFODO SOBRE SD-SLAM

---

para el caso RGBD se realiza con la primera imagen que cuente con al menos 500 puntos característicos o *keypoints*. Una vez el sistema se ha inicializado pasa al estado **OK** donde se utiliza el *tracking* de ORB [6] o estimación de la pose con ORB. En este estado la estimación de la pose se realiza mediante el emparejamiento de puntos característicos y sus descriptores asociados obtenidos mediante el algoritmo ORB. Para que la estimación de pose sea considerada como correcta, existe un límite inferior en cuanto al mínimo número de puntos característicos que han de ser emparejados. Actualmente para SD-SLAM este límite está en 20 pares de puntos emparejados.



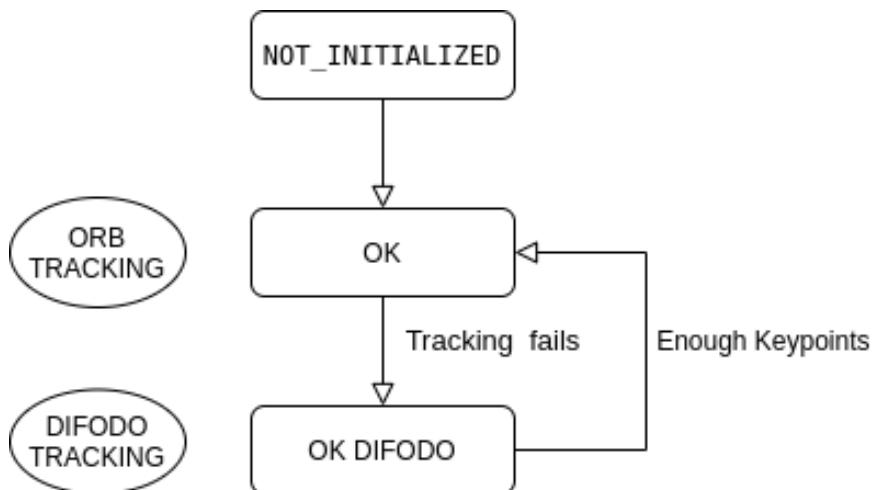
**Figura 4.1:** Máquina de estados de SD-SLAM

Cuando el número de pares de puntos emparejados no es suficiente, el *tracking* falla y el sistema pasa a un estado **LOST**. En este estado el sistema intentará relocalizarse con la llegada de cada nueva imagen. Este proceso de relocalización es similar al de estimación de pose. Se buscan emparejar puntos característicos extraídos de la nueva imagen con cada uno de los puntos característicos extraídos en cada uno de los *keyframes* o imágenes clave, que son imágenes que se han guardado durante el trayecto ya realizado para poder usarlas en la relocalización, en caso de que el sistema se pierda. De nuevo, al menos 20 pares de puntos han de ser emparejados para que la relocalización se considere válida y se vuelva al estado **OK**.

Una vez se conoce en detalle como funciona internamente SD-SLAM, se procede a diseñar como se podría integrar DIFODO. El primer objetivo es establecer cuando se utilizará la odometría de DIFODO, es decir, cuando se considera que la estimación de SD-SLAM no

es suficientemente buena o ha fallado. La primera idea, fue utilizar el número de puntos característicos extraídos por imagen y establecer un umbral, sin embargo, el número de puntos característicos no es una métrica totalmente válida, pues el hecho de que haya pocos puntos característicos no implica necesariamente que la imagen no cuente con suficiente textura para estimar las pose con precisión. Una escena con no muchos puntos característicos, podría darse en una escena cercana, por ejemplo, observando una hoja de papel blanca con unos caracteres de color negro en ella . El número de puntos característicos puede no ser muy alto pero si ser de suficiente calidad como para ser emparejados en su mayoría, lo cual produciría una estimación correcta de la pose. El contrajeemplo sería comparar dos escenas con muchos puntos característicos pero que a la hora de emparejarlos, apenas se emparejan puntos por que son distintas escenas. En conclusión el número de puntos característicos sobre una escena no es una métrica lo suficientemente buena como para establecerla como umbral.

El siguiente paso, fue evaluar usar el mismo criterio que ya estaba siendo usado por SD-SLAM, el número de puntos emparejados entre dos imágenes consecutivas. Ya que SD-SLAM usa este criterio para considerar si la estimación de la pose es correcta o no, tiene sentido que también sea el criterio utilizado para pasar a usar DIFODO. Es por eso que este sería el criterio elegido para pasar a usar DIFODO. Quedando la nueva máquina de estados de la siguiente manera, figura 4.2



**Figura 4.2:** Máquina de estados de SD-SLAM+

En SD-SLAM+ no existe el estado de **LOST**, ya que el algoritmo ya no se pierde, siempre estima pose ya sea con la estimación original de SD-SLAM o con la odometría de DIFODO.

#### 4.4. SD-SLAM+: INTEGRACIÓN DE DIFODO SOBRE SD-SLAM

---

En lugar de este estado, cuenta ahora con un estado denominado **OK DIFODO**, donde la estimación de las nuevas poses se realiza con DIFODO.

El siguiente paso es definir como volver de este estado donde se usa DIFODO al *tracking* original de SD-SLAM una vez que se ha recuperado la información de textura, ya que como se puede observar en la sección 5.4, la estimación de pose usando el emparejamiento de puntos extraídos con ORB comete menos error de media que la estimación de DIFODO, por lo que siempre que se disponga de suficiente textura en la imagen, se dará preferencia al *tracking* original de SD-SLAM frente al de DIFODO.

El primer impulso para comprobar si hay suficiente textura de nuevo, es usar exactamente la misma lógica de emparejamiento de pares de puntos mediante descriptores ORB y cuando el umbral establecido sea superado, volver al *tracking* de ORB. Aunque esta opción parece la más lógica en un primer momento, esto implicaría intentar el emparejamiento entre imágenes consecutivas en todo momento en el estado de **OK DIFODO**. Esto computacionalmente sería muy costoso pues no solo se estaría estimando la pose con DIFODO sino también se estaría intentando hacer lo mismo con ORB, imposibilitando el funcionamiento en tiempo real, el cual es uno de los objetivos de este trabajo fin de máster.

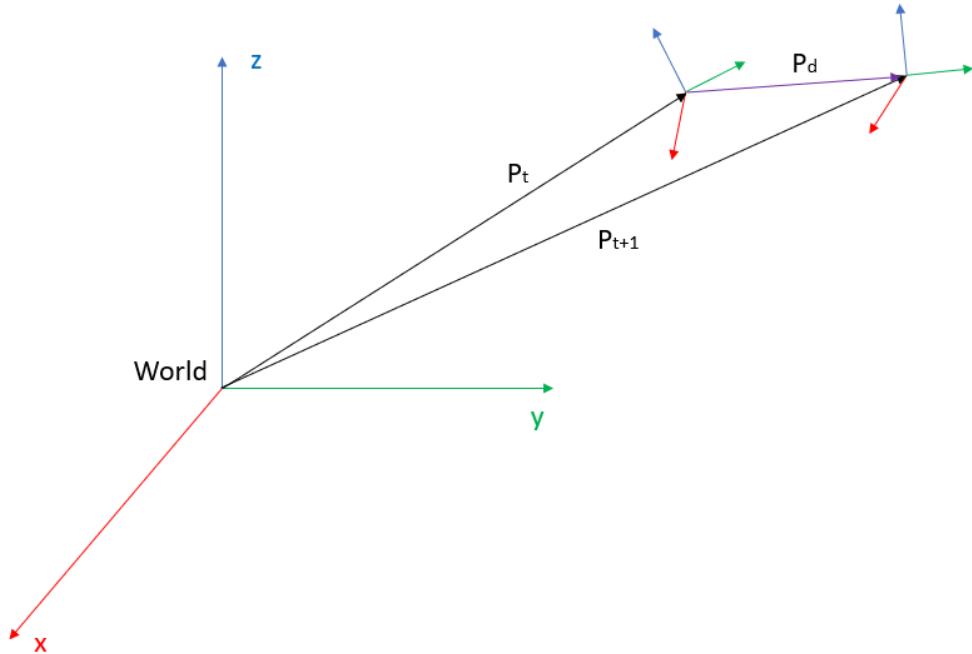
Descartada esta opción, se evalúa la posibilidad de utilizar el número de puntos característicos en la imagen, que como se vio anteriormente, no era la mejor opción a la hora de definir si se debía pasar a usar DIFODO. Este caso es el inverso del anterior estudiado, pues se pasa de DIFODO al *tracking* de ORB. De hecho se observa cierta similitud con el paso de **NOT INITIALIZED** a **OK**. En el paso del estado de no inicialización del sistema al estado de *tracking*, lo único que se comprueba es que haya un número suficiente de puntos característicos en la escena, en concreto 500 puntos, un valor obtenido de forma experimental y que ha demostrado ser suficiente como criterio de inicialización. Se opta pues, usar una lógica similar para volver a la estimación de la pose original de SD-SLAM desde el estado de DIFODO. Se crea un mecanismo de reinicialización que es muy similar a la función de inicialización original con la excepción de que el mapa no se vuelve a establecer sino que se usa toda la información acumulada anteriormente. Mediante los distintos experimentos de la sección 5.5 queda demostrado que este es un criterio suficiente para el correcto funcionamiento del nuevo algoritmo. La ventaja de este método frente al de emparejamiento de pares de puntos, es que este método de reinicialización es muy rápido, pues la extracción de estos puntos característicos en la

imagen es muy rápida mientras que el emparejamiento de puntos característicos es un proceso computacionalmente más costoso, permitiendo el funcionamiento en tiempo real.

El nuevo algoritmo SD-SLAM+ es capaz de pasar de un modo de estimación de la pose a otro eficientemente mientras el funcionamiento en tiempo real se mantiene.

Los criterios para pasar de un estado a otro no son el único reto de esta integración, también se necesita estudiar a más bajo nivel como se realiza esta unión. Uno de los problemas encontrados es que cada sistema de odometría, SD-SLAM y DIFODO, tienen su propio sistema de referencia, por lo que añadir la estimación de una nueva pose de uno al otro no es una operación directa sino que precisa de una transformación previa. El objetivo es, que el sistema de coordenadas de SD-SLAM sea el principal y, que el sistema de referencia de DIFODO, se use única y exclusivamente para obtener la nueva pose que después habrá de ser llevada al sistema de referencia principal.

Por otro lado, DIFODO es un algoritmo de odometría del cual se obtiene como salida la nueva posición. Sin embargo, en este caso es preferible contar con el desplazamiento entre la anterior pose y la nueva pose, para poder añadir este desplazamiento a la última pose conocida de SD-SLAM antes de que se perdiera, de una forma sencilla. La figura 4.3 representa el sistema de referencia de coordenadas de DIFODO, siendo el mundo, *World*, el origen del sistema de referencia de coordenadas,  $P_t$  la posición de la cámara en un instante de tiempo  $t$  y  $P_{t+1}$  la posición de la cámara en un instante de tiempo  $t+1$ . Este ejemplo representa las poses de dos imágenes consecutivas siendo  $P_d$  el desplazamiento entre ambas poses. Este desplazamiento es el que se quiere obtener para luego añadirlo a la última pose conocida por SD-SLAM.



**Figura 4.3:** Cálculo del desplazamiento entre dos poses

Para obtener  $P_d$  siendo conocidas las poses  $P_t$  y  $P_{t+1}$  hay que llevar la pose  $P_{t+1}$  al sistema de referencia de coordenadas de  $P_t$ , o bien si se enfoca como un problema de vectores, hay que restar al punto  $P_{t+1}$  el punto  $P_t$ . Esta operación de sustracción en poses (matrices con orientación y posición) se consigue mediante la aplicación de la inversa:

$$(P_t)^{-1}P_{t+1} = P_d \quad (4.1)$$

La pose es una matriz en coordenadas homogéneas de tamaño total de 4x4 que esta formada por una matriz de rotación  $\mathbf{R}$  3x3 y un vector de traslación  $\mathbf{t}$  3x1. La inversa de una pose se obtiene como se muestra en la figura 4.4:

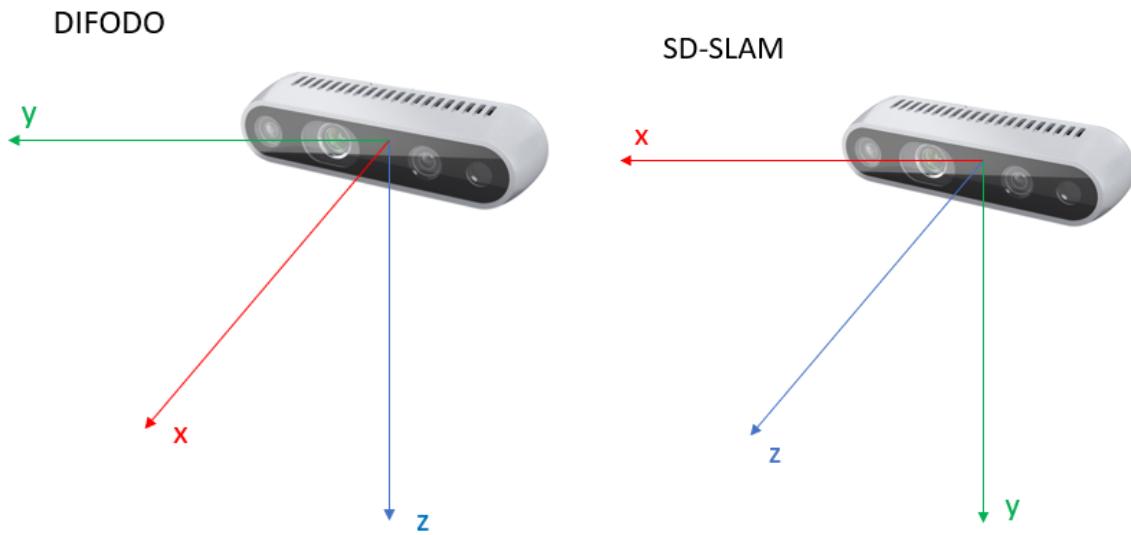
$$\left( \begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline \mathbf{0}^T & 1 \end{array} \right)^{-1} = \left( \begin{array}{c|c} \mathbf{R}^T & -\mathbf{R}^T \mathbf{t} \\ \hline \mathbf{0}^T & 1 \end{array} \right)$$

**Figura 4.4:** Inversa de una pose

De esta forma se consigue el desplazamiento que ha estimado DIFODO con la llegada de la nueva imagen. A continuación, este desplazamiento se habrá de aplicar a la última posición conocida de SD-SLAM, pero para esto es preciso encontrar la relación entre los sistemas de referencia de coordenadas de ambos sistemas primero.

El desplazamiento obtenido depende de la orientación y posición de los ejes de DIFODO con respecto a la imagen o cámara. Que se haya detectado un desplazamiento de 2cm en el eje y (hacia la derecha respecto de la imagen) no implica necesariamente que haya que hacer ese desplazamiento en el eje y en el sistema de referencia original de SD-SLAM. Es posible que en SD-SLAM esa dirección este representada por otro eje distinto al y, incluso puede que este rotado con respecto al de DIFODO, por lo que hay que encontrar la relación de transformación entre ambos sistemas de referencias.

La forma de obtener el sistema de referencias de coordenadas de cada algoritmo es manual. Si un desplazamiento hacia arriba se transmite como un movimiento positivo en el eje z, esa será la dirección y el sentido del eje z. Al menos dos ejes es lo que se necesita saber, pudiendo derivarse el tercero pues sabemos que son sistemas orto-normales.



**Figura 4.5:** Dirección y sentido de los ejes de los sistemas de coordenadas

La figura 4.5 muestra la dirección y el sentido de cada uno de los ejes del sistema de referencia de coordenadas de DIFODO y SD-SLAM. En ella se puede observar como ambos sistemas comparten dirección y sentido de sus ejes con respecto a la cámara solo que existe una relación de rotación entre ellos. Conociendo esta relación entre los ejes es fácil agregar el desplazamiento de DIFODO a la última pose de SD-SLAM. Un desplazamiento en el eje  $y$  en DIFODO se traducirá a un desplazamiento en el eje  $x$  para SD-SLAM. Lo mismo para los otros ejes.

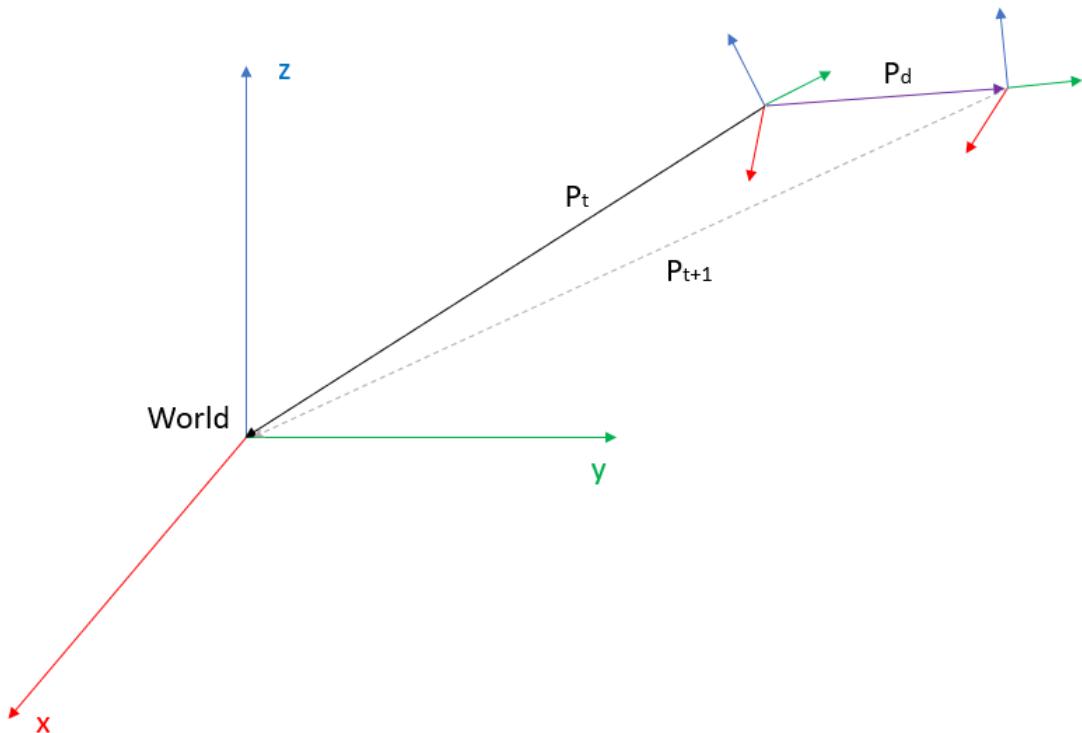
$$X_{DIFODO} = Z_{SD-SLAM}$$

$$Y_{DIFODO} = X_{SD-SLAM}$$

$$Z_{DIFODO} = Y_{SD-SLAM}$$

Como último paso, hay que añadir el desplazamiento ahora según los ejes de SD-SLAM, a la última pose conocida. Este paso es directo salvo por un detalle en la implementación de SD-SLAM. Como se observa en la figura 4.6, la última posición conocida  $P_t$  no se encuentra referenciada al mundo, sino que se conoce la posición del mundo respecto a la pose  $P_t$ . Esto aunque en un primer momento pueda parecer extraño, se debe a que la posición de los distintos puntos característicos encontrados en la imagen se calculan en primera instancia con respecto

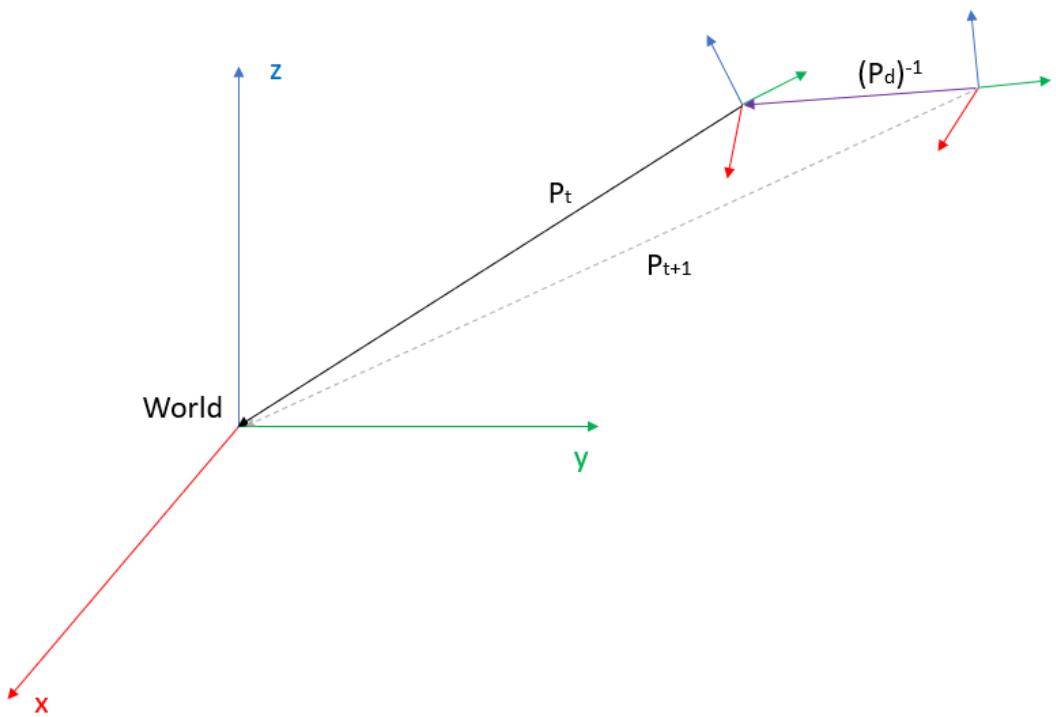
a la vista de la cámara y es a posteriori que estos son transformados al sistema de coordenadas mundo. El origen del sistema de coordenadas mundo, actúa en este caso como un punto más, visto desde la perspectiva de la cámara. De forma que para añadir el desplazamiento y calcular la nueva pose  $P_{t+1}$  es necesario primero cambiar el sistema de referencia de  $P_d$ .



**Figura 4.6:** Adición del desplazamiento a la última pose conocida I

Como la pose de la cámara en SD-SLAM se expresa de esta manera, lo que se desea conocer no es la pose de  $P_{t+1}$  respecto del mundo, sino, la pose del mundo desde  $P_{t+1}$  es decir vista desde la cámara. En este caso, en la ecuación 4.2 es el desplazamiento el que se invierte y se aplica a la última pose conocida. Esto se observa geométricamente en la figura 4.7.

$$(P_d)^{-1}P_t = P_{t+1} \quad (4.2)$$



**Figura 4.7:** Adición del desplazamiento a la última pose conocida II

Finalmente, con esta última operación se obtiene la nueva pose  $P_{t+1}$  que representa la nueva pose estimada, habiéndose usado DIFODO para lograrla. Todas estas operaciones se han traducido en el código de SD-SLAM+ a partir del código ya existente de SD-SLAM.

# Capítulo 5

## Experimentos

A lo largo del desarrollo de este proyecto se han realizado numerosos estudios sobre distintos algoritmos de autolocalización visual, teniendo cada uno de estos experimentos un objetivo concreto. En esta sección se detallan en profundidad, aunque primero se detallan las características del conjunto de datos usado durante la mayoría de experimentos.

### 5.1. El conjunto de datos

El conjunto de datos para evaluar estos algoritmos precisa de unas características específicas, en concreto ha de provenir de un sistema de cámaras con información tanto de color o RGB como de información de profundidad o D de *depth*, profundidad. Estas son las características de la cámara utilizada como apoyo durante el desarrollo de este proyecto, la cámara *Intel® RealSense™ Depth Camera D435*<sup>1</sup>.

El conjunto de datos **RGB-D SLAM Dataset and Benchmark** [9] cumple las características necesarias. Este *dataset* de la universidad técnica de Munich (Technical University of Munich or TUM), contiene más de 50 secuencias de vídeo grabadas con una cámara *Microsoft Kinect™ VI*. Las secuencias están formadas por secuencias de video RGBD con una resolución de 640x480 píxeles a una frecuencia de 30Hz. Las secuencias cuentan con poses consideradas como el *ground-truth* que se pueden usar para comparar las estimaciones de los algoritmos usados. También cuenta con información del acelerómetro incorporado de la Kinect, aunque esta no es usada en este proyecto, así como parámetros de calibración de la cámara o parámetros

---

<sup>1</sup><https://www.intelrealsense.com/depth-camera-d435/>

## 5.2. MÉTRICAS DE CALIDAD

---

intrínsecos en caso de ser necesarios. En el caso de SD-SLAM estos parámetros son necesarios para el proceso de estimación de pose.

El *dataset* se ha grabado en interiores, en habitaciones de oficina, dentro de la propia universidad, y presenta numerosos entornos con características distintas que lo hacen ideal para la comparación de distintos algoritmos. Una de estas características es la de contar con secuencias sin textura, secuencias sin estructura y secuencias que no cuentan con textura ni estructura. Las distintas secuencias están disponibles en dos formatos:

- Como un conjunto de imágenes que pueden ser cargadas desde disco.
- Como *rosbag*, un formato utilizado en el ámbito de ROS.

El formato usado en la realización de estos experimentos ha sido el de ROS ya que ambos algoritmos DIFODO y SD-SLAM están preparados para trabajar con este formato lo que facilita su uso.

## 5.2. Métricas de calidad

Entre las distintas formas de evaluar dos trayectorias para medir cual es la diferencia entre ellas hay dos que destacan y que se han popularizado entre los algoritmos de SLAM, son el ATE y el RPE [9] . El ATE o error de trayectoria absoluto está más enfocado en medir el rendimiento de algoritmos de visual SLAM. Por otro lado, el RPE o error de posición relativa está más enfocado en medir el rendimiento de algoritmos de odometría visual.

El ATE primero hace un alineamiento de ambas trayectorias en caso de ser necesario, para luego evaluar la distancia geométrica entre cada una de las posiciones de ambas trayectorias. En el cálculo de la diferencia entre poses solo se usa la traslación y no la orientación y, aunque pueda parecer que se está perdiendo la información de la orientación, esta se tiene en cuenta finalmente ya que un error en la orientación se acaba traduciendo con el tiempo en un error en la posición en el espacio.

El RPE calcula diferencias en el movimiento entre dos instantes temporales o sellos temporales al ser las estimaciones una señal no continua. El RPE puede y suele calcularse usando como unidad las estimaciones con cada nueva imagen. Se puede comparar el movimiento entre imágenes consecutivas, o cada X intervalo de imágenes.

Otro criterio a tener en cuenta es la eficiencia computacional, el tiempo de cómputo, que determina si el algoritmo podrá ejecutarse en robots y dispositivos con capacidad de cómputo limitada. Durante el desarrollo de este trabajo, el hardware sobre el que se estaba desarrollando se vio sustituido por uno más actual. Esto ha permitido que se obtengan tiempos en el procesamiento de DIFODO tanto para el hardware antiguo como para el nuevo. DIFODO principalmente utiliza el procesador en su implementación y, aunque la memoria RAM y la placa base también afectan al rendimiento en cierta medida, se va a utilizar el procesador como referencia en este estudio. La CPU más antigua era el modelo AMD FX-8370, una CPU del Q3 de 2014 que cuenta con 8 Núcleos, 8 Hilos a una frecuencia de 4.0GHz. La nueva CPU es un AMD Ryzen 5 2600, una arquitectura más nueva del segundo cuarto de 2018 que cuenta con 6 Núcleos, 12 Hilos a una frecuencia de 3.4GHz. El AMD FX-8370 es una CPU que en su día estuvo considerado alta gama mientras que el Ryzen 5 2600 es una gama media. Cuatro años separan a estos procesadores, siendo el Ryzen 5 uno de los procesadores más usados actualmente en el ámbito *gaming* por su relación calidad precio.

Por otro lado, también se cuenta con información de tiempos en el artículo original así como de la CPU utilizada, un Intel Core i7-3820 del final de 2013 con 4 Núcleos, 8 Hilos y frecuencia de 3.6 GHz. Un procesador cuyo ciclo de vida coincidió con el del AMD-FX 8370 por lo que será útil para comparar estos tiempos. Finalmente otra característica importante en SLAM es la robustez del sistema, entendiéndose por robustez la capacidad de perderse con menor o mayor frecuencia. Esta característica se reflejará en los experimentos en el error del ATE y de forma cualitativa.

## 5.3. Evaluación de DIFODO

DIFODO es la piedra angular de la mejora sobre SD-SLAM realizada en este TFM pero para evaluar si su integración tenía sentido, se realizaron una serie de experimentos para analizar su rendimiento y tiempos de procesamiento en función de cada uno de los hiperparámetros que permite modificar el algoritmo. Tras la realización de estos experimentos se alcanzaron unas conclusiones que son las que llevaron a integrarlo con SD-SLAM para mejorarlo.

DIFODO cuenta con una serie de hiperparámetros que permiten modificar cómo se comporta el algoritmo [2]. Estos parámetros afectan principalmente a la estimación de la odometría

## 5.3. EVALUACIÓN DE DIFODO

---

pero también hay algunos parámetros que afectan al tiempo de procesamiento. El parámetro que más afecta a la estimación y al tiempo de procesamiento es el número de imágenes usadas en la pirámide de imágenes. Por otro lado, la resolución de entrada de las imágenes, a pesar de no ser un parámetro directo de DIFODO, afecta en gran medida al rendimiento y tiempo de procesamiento del algoritmo por lo que también se estudiará en esta sección.

El tiempo de procesamiento es uno de los puntos más importantes de cara a cumplir con el requisito de tiempo real para el nuevo algoritmo desarrollado SD-SLAM+. Es por eso que es clave su estudio en DIFODO. En función del artículo original [2], DIFODO es capaz de funcionar a 60Hz en un procesador de un solo núcleo. Aunque no se especifica en qué resolución, más adelante se observarán los tiempos expuestos en el artículo de DIFODO así como los obtenidos experimentalmente.

### 5.3.1. Efecto de la resolución de las imágenes de entrada

La resolución afecta al rendimiento de DIFODO y a la estimación de la odometría. Los resultados mostrados en el artículo original [2], muestran como a mayor resolución, mayor es la precisión de la estimación así como el tiempo de procesamiento. Se han realizado una serie de experimentos sobre el mismo conjunto de datos pero usando la versión de DIFODO integrada con ROS que es como finalmente funcionará el algoritmo.

La tabla 5.1 muestra los tiempos de procesamiento de DIFODO a distintas resoluciones para los distintos procesadores. En ella se observa cómo se ha conseguido una reducción en el tiempo de procesamiento de casi un factor de 3 para la nueva CPU, el Ryzen 5 2600. Para las resoluciones menores a 320x240 el tiempo de procesamiento es menor que 33.3ms, tanto para la nueva CPU como la antigua, por lo que DIFODO a esas resoluciones funciona a tiempo real, cumpliendo el requisito de procesamiento en tiempo real a 30 imágenes por segundo. Esto es de especial importancia pues uno de los objetivos de este trabajo es conseguir un algoritmo que funcione en tiempo real. La resolución VGA presenta tiempos superiores al límite de los 33.3ms por lo que no se podrían considerar en casos donde la CPU no sea especialmente potente.

Resolución	Ryzen 5 2600	AMD-FX 8370	Intel Core i7-3820
40x30	-	-	3.85ms
80x60	-	-	5.18ms
160x120	3.5ms	9ms	10.04ms
320x240	12.75ms	30ms	28.61ms
640x480	55.5ms	130ms	-

**Tabla 5.1:** Tiempos de procesamiento para distintas resoluciones y distintos procesadores

En cuanto a la relación tiempo/resolución, se observa una clara tendencia donde cada vez que se aumenta en dos veces la resolución, cuatro veces si nos basamos en el número de píxeles totales, el tiempo de procesamiento aumenta de forma exponencial. Resoluciones similares o mayores a VGA se tienen que descartar si no se cuenta con un procesador de gran potencia o si se pretende trabajar a tiempo real o 30 imágenes por segundo. Esto es crucial en los algoritmos de SLAM ya que trabajar a menor tasa de imágenes por segundos dificulta la tarea de la estimación de la pose al presentar mayor desplazamiento entre imágenes.

En cuanto a los tiempos obtenidos, se puede observar cómo los tiempos expuestos en el artículo original son verosímiles puesto que los procesadores antiguos, AMD-FX 8370 e Intel Core i7-3820 son de un rendimiento similar y se obtienen tiempos similares. No se han realizado experimentos de tiempos a resoluciones menores como sí que se hizo en el artículo original, ya que no tiene sentido ir mucho más abajo en resolución pues a 160x120 hay mucho espacio hasta llegar a los 33.3ms. Lo que sí se ha hecho es comprobar que a resoluciones algo más grandes como VGA, el tiempo real deja de ser posible.

Una vez se han observado los tiempos de procesamiento, se puede concluir que DIFODO es un algoritmo que puede integrarse junto a SD-SLAM al menos en cuanto al requisito de procesamiento en tiempo real. El siguiente paso es observar cómo afecta la resolución a la estimación de la pose.

Lo esperado en base a los resultados reportados en el artículo original [2] es que el algoritmo a mayor resolución tenga un menor error en la estimación de la pose. La tabla 5.2 muestra el error de trayectoria absoluto para distintas resoluciones y para distintas secuencias del conjunto de datos. En rojo se muestra la resolución que arroja el mayor error y en verde la que consigue el menor. La resolución que parece obtener mejores resultados de forma general es la resolu-

### 5.3. EVALUACIÓN DE DIFODO

---

ción QVGA (320x240). Mientras que los peores resultados se obtienen en la mayor resolución 640x480. Estos resultados son a priori contrarios a lo esperado y a las conclusiones del artículo original, donde se espera que a mayor resolución, menor sea el error. Sin embargo, esto tiene una explicación, que tiene que ver con el diseño de la rosificación de DIFODO que se ha realizado para evaluar el algoritmo. Este software está diseñado para trabajar a la mayor frecuencia de imágenes posible o bien se puede fijar una frecuencia objetivo. En este caso, la frecuencia de las secuencias de vídeo del conjunto de datos es de 30 imágenes por segundo, lo que implica que la máxima frecuencia está limitada por la secuencia. Como se observó en la tabla 5.1, las resoluciones 160x120 y 320x240 pueden trabajar a 30 imágenes por segundo sin problema, incluso a frecuencias mayores, pero la resolución 640x480 apenas llega a las 20 imágenes por segundo. Esto produce que una de cada tres imágenes se pierdan al no poder seguir DIFODO a la resolución de 640x480 la tasa de imágenes por segundo de la secuencia, provocando una perdida de información que se traduce en un desplazamiento mayor entre las imágenes sobre la que estimar la odometría que afecta de forma negativa a la estimación del algoritmo.

Secuencia	160x120	320x240	640x480
desk2	0.2620	0.2094	0.2026
floor	0.8663	0.8844	0.8814
360	0.4496	0.4339	0.5692
rpy	0.0793	0.1175	0.1318
xyz	0.0805	0.0613	0.0756
desk	0.1476	0.1441	0.1573
room	0.4131	0.3705	0.4839
Total	2.2984	2.2211	2.5018

**Tabla 5.2:** RMSE en metros para el ATE (Absolute Trajectory Error) en distintas resoluciones de DIFODO.

#### 5.3.2. Efecto del nivel de la pirámide

DIFODO utiliza una pirámide de imágenes donde la imagen de mayor resolución es la imagen original y las imágenes sucesivas se submuestrean, obteniendo lo que es conocido como

pirámide de imágenes en visión artificial. Este mecanismo se utiliza para aplicar algoritmos de procesamiento a distintos niveles de resolución para después juntar la información obtenida en distintos niveles, obteniendo resultados mucho mejores que en la evaluación a una única resolución. Esta técnica se usó en el cálculo del flujo óptico [1]. DIFODO también la usa pero para el cálculo del *range flow* [8], técnica derivada del flujo óptico para ser aplicada en imágenes de profundidad en lugar de RGB. Esta pirámide le permite observar tanto movimientos de pequeños objetos como de objetos más grandes, lo que le permite obtener una mejor estimación que usando solo la imagen original. Esto afecta directamente a la precisión de la estimación obtenida y también al tiempo de procesamiento, pues a menor número de imágenes en la pirámide menos son las veces que se tendrá que calcular el *range flow*.

De nuevo se analiza tanto a nivel de tiempos como a nivel de error en la estimación el efecto de este parámetro.

Resolución	Niveles de Pirámide	Tiempo (ms)
160x120	1	4
160x120	3	8.5
160x120	5	9
320x240	1	15
320x240	3	27
320x240	5	30
640x480	1	80
640x480	3	130
640x480	5	130

**Tabla 5.3:** Tiempos de procesamiento para distintos niveles de pirámide en AMD-FX 8370

En la tabla 5.3 se observa cómo a mayor número de niveles usados en la pirámide mayor es el tiempo de procesamiento. Como se puede apreciar, la diferencia entre calcular 3 o 5 niveles es casi despreciable. La razón es que en cada nuevo nivel se está reduciendo la imagen original a la mitad del tamaño original. Se consigue pues un efecto similar al de elevar al cuadrado de forma continua. Por ejemplo, para la resolución de 640x480 los niveles del resto de la pirámide serían 320x240, 160x120, 80x40 y 40x20 respectivamente. La resolución de los últimos dos niveles

### 5.3. EVALUACIÓN DE DIFODO

---

de la pirámide es muy pequeña en comparación con el primer nivel por lo que la diferencia entre usar 3 y 5 niveles es computacionalmente despreciable. Sin embargo, usar únicamente la imagen de entrada supone un ahorro computacional cercano al 50 %. Finalmente, en la tabla 5.4 se aprecia cómo la disminución de tiempos con la nueva CPU mantiene la proporción de 3 veces inferior, independientemente del número de niveles de la pirámide.

Resolución	Niveles de Pirámide	AMD-FX 8370	AMD Ryzen 5 2600
320x240	1	15ms	5.85ms
320x240	3	27ms	12.29ms
320x240	5	30ms	12.75ms

**Tabla 5.4:** Comparación de tiempos de procesamiento para distintos niveles de pirámides y procesadores.

Se ha visto cómo la diferencia entre usar 3 o 5 niveles de pirámide es computacionalmente despreciable, mientras que se ahorra un 50 % de tiempo si solo usamos un nivel. La siguiente duda a resolver es comprobar cómo afecta esto a la precisión en la estimación de la pose.

Mediante un rápido vistazo a la tabla 5.5 se puede apreciar claramente que a mayor número de imágenes en la pirámide menor es el error cometido en la estimación de la pose. La diferencia es muy notable cuando se usa un solo nivel de pirámide siendo el error total en torno a 2.5 veces mayor que cuando se usan 5 niveles, como era de esperar pues una única resolución solo es capaz de encontrar movimientos muy locales y no globales o grandes en la imagen de profundidad.

Solo hay una secuencia donde los resultados son muy parejos en los 3 casos, en la secuencia titulada *floor*. Esta secuencia además presenta un error muy alto comparado con el resto de secuencias, esto es así puesto que como la secuencia indica, es una secuencia donde la cámara mira al suelo. Esto produce que haya una ausencia de estructura en la imagen lo que dificulta a DIFODO la estimación. Es el caso equivalente a una escena que es homogénea en los niveles de intensidad de sus píxeles para el cálculo del flujo óptico. Independientemente del nivel de pirámide no se puede hacer una estimación precisa y los nuevos niveles de pirámide no aportan información al no haber estructura ya sea de objetos pequeños o grandes.

Secuencia	1	3	5
desk2	1.1880	0.2955	0.2094
floor	0.8877	0.9473	0.8844
360	1.204	0.6992	0.4339
rpy	0.3438	0.1496	0.1175
xyz	0.1165	0.0645	0.0613
desk	0.7288	0.2997	0.1441
room	1.2819	0.4188	0.3705
Total	5.7507	2.8746	2.2211

**Tabla 5.5:** RMSE en metros para el ATE (Absolute Trajectory Error) en distintos niveles de pirámide de DIFODO.

### 5.3.3. Conclusiones

Se ha comprobado cómo DIFODO es un algoritmo capaz de ejecutarse a 30 imágenes por segundo en procesadores no muy exigentes, e incluso a mayor frecuencia en otros procesadores o a resoluciones inferiores sin comprometer la estimación de la pose. Esto es así cuando se trabaja a una resolución menor o igual a 320x240. Cumple por lo tanto con la necesidad de trabajar a tiempo real.

Queda demostrado que el nivel de pirámide afecta a la estimación de la pose y al tiempo de procesamiento, y se llega a la conclusión de que el número óptimo de niveles de pirámide es 5, ya que a 3 niveles de pirámide el tiempo que se reduce es despreciable y se produce una perdida de precisión en la estimación de la pose.

Por otro lado, se observa cómo las estimaciones de DIFODO son bastante buenas siempre y cuando haya suficiente estructura en la escena. Una comparación más exhaustiva del error cometido se verá en el siguiente capítulo donde se compara con SD-SLAM.

## 5.4. Comparativa entre DIFODO y SD-SLAM

En esta sección se hace una evaluación cuantitativa de el algoritmo DIFODO y de SD-SLAM. Hay que tener en cuenta que DIFODO es un algoritmo únicamente de odometría visual,

## 5.4. COMPARATIVA ENTRE DIFODO Y SD-SLAM

---

por lo que se obtiene una estimación con cada nueva imagen en forma de desplazamiento con respecto a la imagen anterior. Sin embargo SD-SLAM es un algoritmo de SLAM completo, no solo hace odometría o *tracking*, sino que hace mapeado también. Sin embargo, lo más importante de cara a esta comparación es que el mapeado le permite hacer un seguimiento de aquellos lugares que se han visto previamente y usarlos para optimizar de forma global las siguientes estimaciones así como las anteriores mediante el mecanismo denominado cierre de bucle. Esto le brinda una mayor robustez en la estimación de pose frente a DIFODO donde solo se utiliza la información del fotograma actual y el anterior en las predicciones. Es de esperar que SD-SLAM tenga unas mejores prestaciones que DIFODO.

### 5.4.1. Precisión de las estimaciones

Para hacer esta comparación entre SD-SLAM y DIFODO se va a usar el software de SD-SLAM y la versión *rosificada* de DIFODO en conjunto con la herramienta descrita en la sección 4.3, `odometry_evaluation_file_creator`. El dataset para el estudio de nuevo es **RGB-D SLAM Dataset and Benchmark** [9] detallado anteriormente en la sección 5.1. En cuanto a DIFODO, los parámetros usados durante la realización de estos experimentos son 320x240 de resolución y un nivel de pirámide de cinco niveles, pues se ha demostrado en la sección anterior, que es la mayor resolución que permite tiempo real de procesamiento y la que menor error presenta en la estimación.

Secuencia	DIFODO RMSE	SD-SLAM RMSE	Mejora relativa del error (%)
desk2	0.2094	<b>0.1716</b>	18.01
floor	0.8844	<b>0.3982</b>	54.96
360	0.4339	<b>0.3864</b>	10.92
rpy	0.1175	<b>0.0281</b>	76.02
xyz	<b>0.0613</b>	0.0626	-2.27
desk	0.1441	<b>0.1235</b>	14.26
room	0.3705	<b>0.2909</b>	21.46
<b>Valor medio</b>	<b>0.3173</b>	<b>0.2088</b>	<b>34.18</b>

**Tabla 5.6:** Comparación del ATE (Absolute Trajectory Error) en metros para DIFODO y SD-SLAM en secuencias de Freiburg 1.

#### 5.4. COMPARATIVA ENTRE DIFODO Y SD-SLAM

---

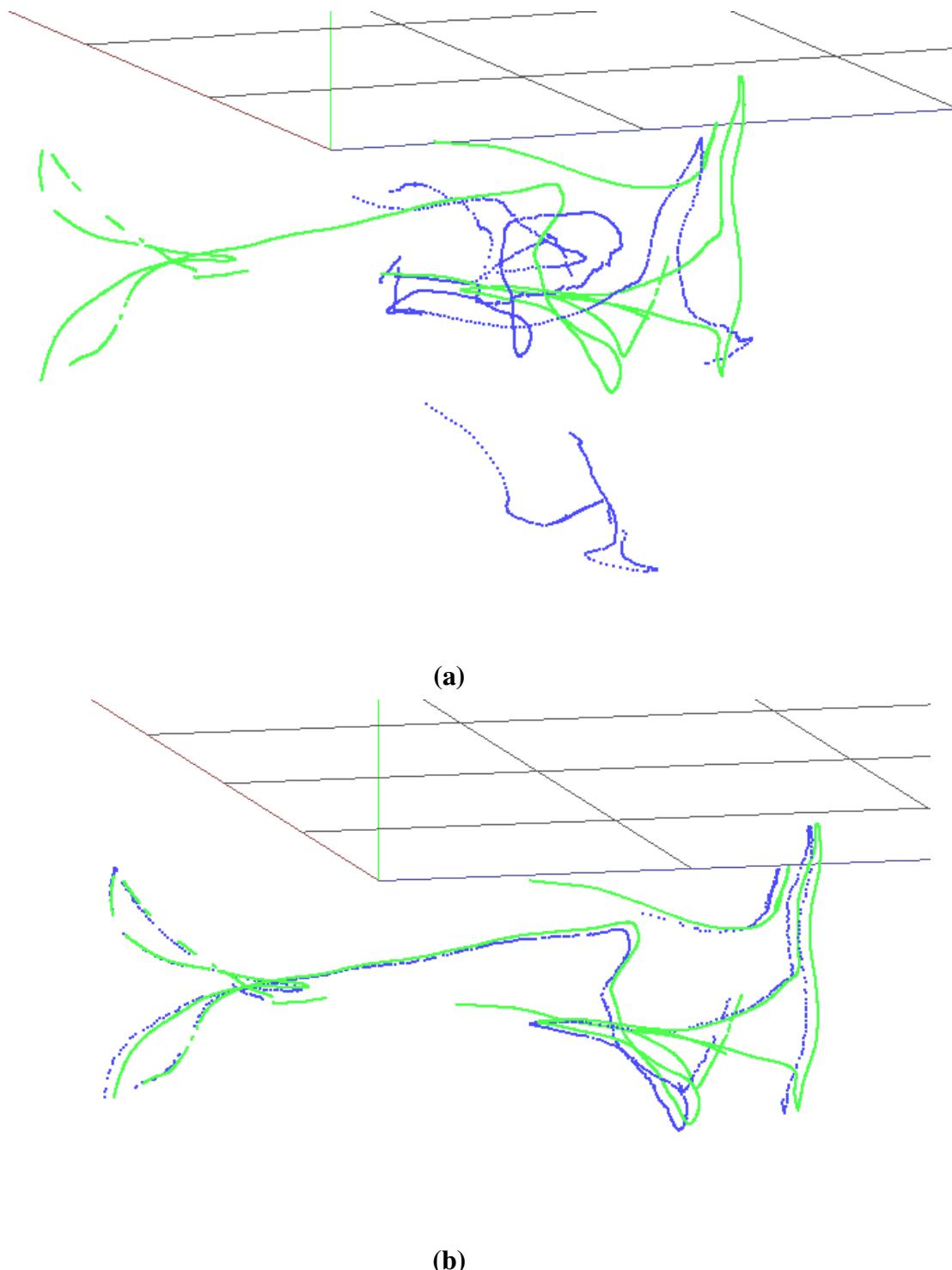
En la tabla 5.6 se compara el error cometido por cada algoritmo en las secuencias de tipo *Freiburg 1*. Como era de esperar el error es menor en casi todos los casos para SD-SLAM en comparación con DIFODO. SD-SLAM reduce el error en la estimación de la pose en un 34 % de media en comparación con DIFODO para estas secuencias. Sin embargo hay una serie de datos a tener en cuenta. En muchas secuencias como *desk2*, *360*, *xyz*, *desk* la diferencia en la estimación es mucho menor que ese 34 %, en torno a un 15-20 %, siendo muy parejos en ambos casos incluso llegando a ser superior DIFODO en la secuencia de *xyz*.

De entre todas estas secuencias hay dos valores atípicos claros, que se aprecian para las secuencias *floor* y *rpy*. En la secuencia *floor* la estimación de DIFODO contiene mucho error al no contar con apenas estructura como ya se discutió anteriormente en la sección ???. Se puede apreciar la mala estimación de DIFODO en la figura 5.1.

En el caso de la secuencia *rpy* la diferencia apreciada no radica en el hecho de que DIFODO tenga una mala estimación sino que SD-SLAM obtiene una muy buena estimación en este caso, de hecho, es la secuencia donde SD-SLAM presenta un menor error de todas las presentadas al igual que DIFODO.

## 5.4. COMPARATIVA ENTRE DIFODO Y SD-SLAM

---



**Figura 5.1:** Trayectorias de los algoritmos para la secuencia *floor*. En verde se muestra la trayectoria verdadera, en azul la trayectoria estimada en cada caso. La figura (a) corresponde a la estimación de DIFODO y la figura (b) a la estimación de SD-SLAM.

## 5.4. COMPARATIVA ENTRE DIFODO Y SD-SLAM

---

Tipo	Secuencia	DIFODO	SD-SLAM
Sin estructura	nostructure_texture_near	2.2992	<b>0.0665</b>
Sin textura	structure_notexture_near	<b>0.2903</b>	2.005
Sin textura	structure_notexture_far	<b>0.2708</b>	2.6243
Sin estructura/textura	nostructure_notexture_near_withloop	2.3351	2.2091
Sin estructura/textura	nostructure_notexture_far	1.9627	1.7486

**Tabla 5.7:** Comparación del ATE en metros para DIFODO y SD-SLAM en secuencias de Freiburg 3: Estructura vs Textura. En verde el menor error en cada secuencia.

En la tabla 5.7 se encuentran los resultados del error de la estimación pero en este caso para secuencias de la categoría *Freiburg 3*. Estas secuencias se caracterizan por ser más complejas y presentar diversidad de entornos. En la tabla 5.7 se han dividido las secuencias por tipos que corresponden a:

- **Sin estructura:** Secuencias donde no hay información de estructura pero sí de textura visual. Por ejemplo una pared lisa pero con dibujos o un folio sobre una superficie plana con texto.
- **Sin textura:** Secuencias donde no hay información de textura pero sí de estructura. Por ejemplo distintos objetos pero todos monocromos, uniformes visualmente. Un ejemplo serían varias paredes lisas blancas colocadas paralelas entre sí.
- **Sin estructura/textura:** Secuencias donde no hay información ni de estructura ni de textura. Por ejemplo una pared lisa.

Del tipo sin estructura, se ha evaluado la secuencia titulada *nostructure\_texture\_near*. Al no haber estructura el error de DIFODO es muy grande, de más de 2m, mientras que para SD-SLAM se consigue 0.06m de error, apenas unos centímetros.

Del tipo sin textura, se han evaluado dos secuencias: *structure\_notexture\_near* y *structure\_notexture\_far*. Al no haber textura SD-SLAM obtiene un error de varios metros, ya que en esas secuencias SD-SLAM no consigue ni llegar a inicializar, por lo que no se consigue ni siquiera una estimación de trayectoria. Por otro lado DIFODO permanece en su línea de error, al contar con suficiente información de estructura su estimación es similar a la encontrada en las secuencias anteriores,

## 5.5. SD-SLAM FRENTE A SD-SLAM+

---

*Freiburg 1.*

Finalmente hay un tipo de secuencia donde no se cuenta con textura y tampoco con estructura. Como es de esperar ambos algoritmos obtienen un error muy alto en torno a los 2m.

En todos los casos el comportamiento y los resultados son los esperados, pues ambos algoritmos dependen de la información visual ya sea en textura o en profundidad.

SEGUIR EN LA PAGINA 14 DEL PDF AÑADIENDO LA NUEVA SECCION DE EFICIENCIA COMPUTACIONAL QUE ME ESTAN PIDIENDO

### 5.4.2. Conclusiones

DIFODO ha demostrado ser un algoritmo de odometría visual con un buen rendimiento. SD-SLAM es de media un 20 % aproximadamente mejor que DIFODO en la estimación de la pose. DIFODO solo presenta mejor estimación de pose que SD-SLAM cuando hay poca textura visual. Sin embargo DIFODO, cuando no hay estructura no es capaz de ofrecer una buena estimación. Ambos algoritmos precisan pues, de información visual, en un caso textura y en el otro estructura. El uso de uno cuando el otro falla tiene sentido pues se puede utilizar como mecanismo de apoyo para no perder el *tracking*. Tiene sentido por los resultados vistos usar siempre que haya suficiente información visual SD-SLAM, pues de media, la estimación obtenida es mejor y se pueden usar otros mecanismos propios de un algoritmo de SLAM como el cierre de bucle. En aquellos casos donde no haya suficiente textura en la imagen y donde SD-SLAM normalmente se perdería o estimaría una mala posición, se usará DIFODO para suplir esta falta de información.

Se observa también que cuando no hay información ni de textura ni de estructura, el error en ambos algoritmos es muy alto, pero estamos hablando de el caso más difícil que pueden afrontar los algoritmos de visual SLAM. La solución en este caso sería utilizar información de otra fuente que no sea visual, como por ejemplo inercial.

## 5.5. SD-SLAM frente a SD-SLAM+

En esta sección se va a evaluar el error en la estimación de la pose para el algoritmo original, SD-SLAM y su evolución SD-SLAM+. Para ello se van a tener que simular modificaciones sobre el conjunto de datos originalmente usado y descrito en Sección 5.1.

### 5.5.1. El conjunto de datos

De nuevo el conjunto de datos usado es **RGBD TUM dataset** [9]. Sin embargo, a pesar de que el *dataset* contiene secuencias diversas como: secuencias de solo textura, secuencias de solo estructura, secuencias con textura y estructura, etc., no existe ninguna secuencia donde se tengan textura y estructura y, además, en algún momento se pierda la textura. Con el objetivo de probar el funcionamiento y el rendimiento del nuevo SD-SLAM+, es necesario simular perdidas de textura en las distintas secuencias. Para esto se ha creado una versión de ambos algoritmos con un nuevo parámetro en el fichero de configuración donde se pueden configurar los intervalos de tiempo donde se simulará esta perdida de textura. Esto permite que el algoritmo en los intervalos configurados reciba imágenes RGB completamente negras, creando una matriz con la misma resolución que el resto de imágenes de la secuencia donde el valor de intensidad de todos los pixeles se encuentra a cero.

### 5.5.2. Experimento: DIFODO como apoyo a SD-SLAM

Estos experimentos se centran en la idea de demostrar que la integración de DIFODO en SD-SLAM, mejora al algoritmo original, dando lugar a SD-SLAM+. Para ello se han realizado una serie de test utilizando la misma secuencia de video y cambiando los intervalos donde se simula la perdida de textura, para tener diferentes escenarios.

La secuencia usada es **rgbd\_dataset\_freiburg1\_desk.bag**. Esta secuencia muestra una trayectoria semicircular a lo largo de dos escritorios de un laboratorio de la universidad de Munich como se puede apreciar en Figura 5.2.

La trayectoria es de ida, yendo desde uno hacia el otro de los escritorios y luego vuelve por el mismo lado. Es por lo tanto una secuencia que permite relocalizaciones y cierres de bucles en algoritmos de SLAM, al pasar por zonas que ya ha recorrido previamente. La secuencia es de 23.4 segundos de duración, sin embargo, aproximadamente los primeros 4 segundos de este secuencia y del resto, carecen de imágenes. Esto es así para que a la hora de grabar el hardware tenga un tiempo de estabilización o calentamiento en el arranque, por lo que la duracion real sería de 20 segundos aproximadamente. Cerca de la mitad de la secuencia se emplea en ir de un escritorio al otro y la otra mitad de la secuencia en volver, por lo que a partir de la segunda mitad de la secuencia se pueden producir cierres de bucles.

## 5.5. SD-SLAM FRENTE A SD-SLAM+

---



**Figura 5.2:** SD-SLAM aplicado en `rgbd_dataset_freiburg1_desk.bag`

### Test 1

- **Intervalos sin textura simulados (segundos): [4-5]**
- **Descripción:** Se ha simulado una pérdida de luz del segundo 4 al 5 con el objetivo de ver como el nuevo SD-SLAM+ es capaz de manejar esta situación en comparación a la versión anterior.

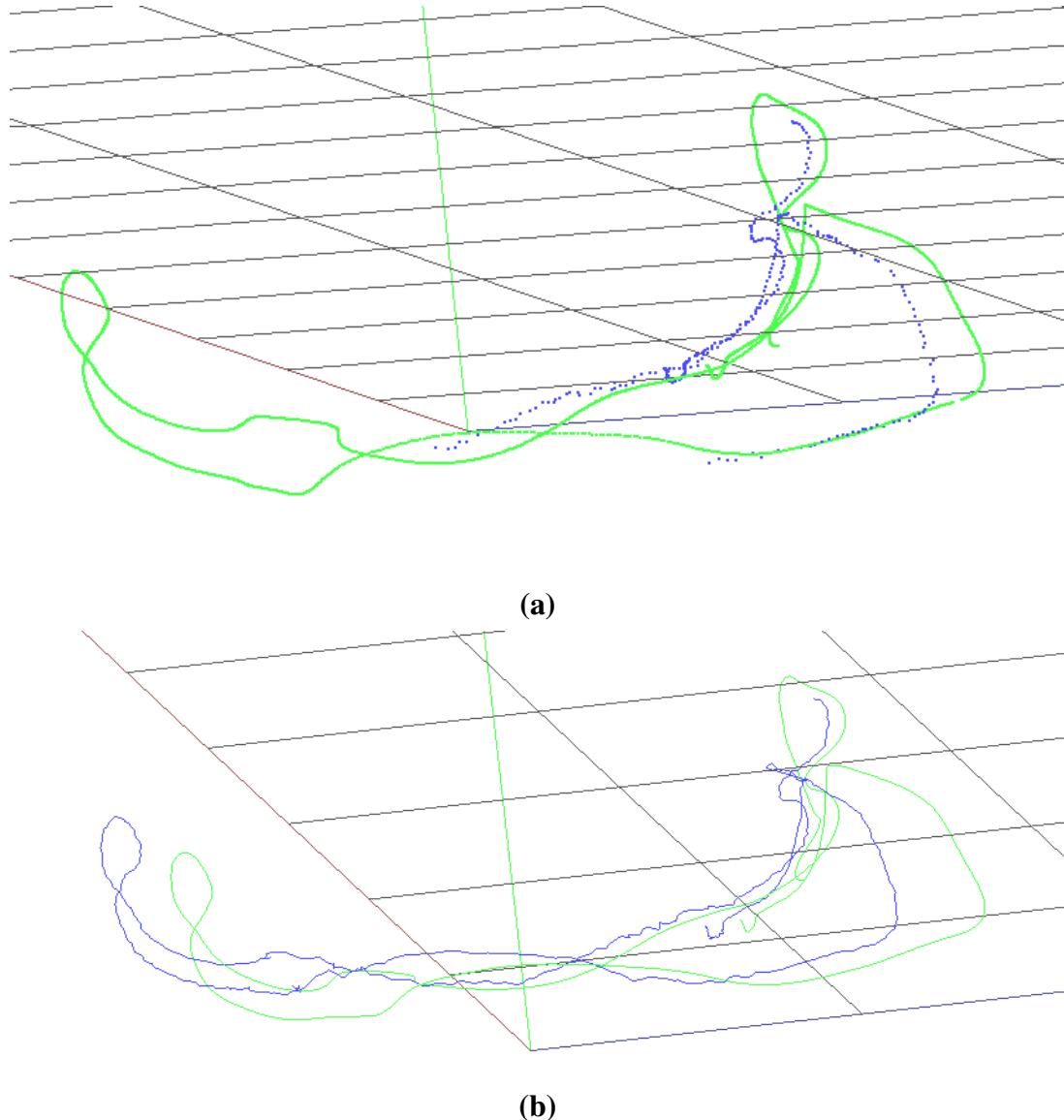
Version	rmse (m)	mean (m)	median (m)	std (m)	min (m)	max (m)	sse (m)
SD-SLAM	1.2528	0.8894	0.1415	0.8823	0.0350	2.1653	866.4533
SD-SLAM+	0.1863	0.1673	0.1199	0.0820	0.0352	0.3153	19.1323

**Tabla 5.8:** Comparación del ATE en el Test 1: SD-SLAM vs SD-SLAM+

Los resultados de la anterior Tabla 5.8 muestran como el error de la nueva versión es mucho más pequeño que la versión original. Mientras que la versión original de SD-SLAM tiene un *rmse* de 1.25m la nueva versión, SD-SLAM+, tiene un *rmse* de 0.18m, un error 6 veces más pequeño en este caso. Estos resultados eran esperables pues como se aprecia en la Figura 5.3, en la imagen (a), la trayectoria estimada solo se encuentra en la primera mitad de la trayectoria. Esto es debido a que a partir del segundo 4 y hasta el 5, desaparece la textura, lo que provoca

que el algoritmo de SD-SLAM no pueda continuar con la estimación de la pose y entre en un estado de relocalización. El algoritmo seguirá en este estado hasta la vuelta donde volverá a pasar por zonas conocidas y prodrá relocalizarse, volviendo a estimar la pose en los últimos tramos. Todo esto se traduce en un alto error ATE, ya que hay zonas durante las cuales no se ha producido estimación alguna. En el caso de la versión modificada, Figura 5.3 **(b)**, como era esperable, se encuentra estimación de la pose a lo largo de toda la trayectoria. Esto es gracias a que durante ese intervalo de 1 segundo donde no se cuenta con textura el nuevo algoritmo pasa a estimar la pose usando DIFODO para ello. Cuando hay textura de nuevo el algoritmo recupera el *tracking* original usando ORB. Es por ello que se encuentra un error mucho más pequeño y cercano al que se obtendría en la secuencia original sin este intervalo sin textura.

## 5.5. SD-SLAM FREnte A SD-SLAM+



**Figura 5.3:** Trajetorías de los algoritmos mostradas en el programa slam-testbed. En verde se muestra el *ground-truth*, en azul la trayectoria estimada en cada caso. Las figura (a) corresponde a la versión original de SD-SLAM y la figura (b) a la versión modificada de SD-SLAM, SD-SLAM+

### Test 2

- **Intervalos sin textura simulados (segundos):** [4-5, 6-7, 8-9]
- **Descripción:** En esta ocasión se aumenta el número de intervalos sin textura durante la secuencia. El objetivo es ver como se comporta SD-SLAM+ cuando se intercambia entre un algoritmo y otro.

Version	rmse (m)	mean (m)	median (m)	std (m)	min (m)	max (m)	sse (m)
SD-SLAM	1.2528	0.8894	0.1415	0.8823	0.0350	2.1653	866.4533
SD-SLAM+	0.1960	0.1781	0.1609	0.0819	0.0335	0.3265	21.2211

**Tabla 5.9:** Comparación del ATE en el test 2: SD-SLAM vs SD-SLAM+

Los resultados para SD-SLAM siguen siendo los mismos pues desde que deja de hacer estimaciones de pose, en el primer intervalo donde se pierde la textura, no es capaz de relocalizarse. El resto de intervalos de perdida de textura se producen antes de que la relocalización sea posible por lo que no afectan al error de trayectoria absoluto, siendo el de la Tabla 5.8 el mismo que en la Tabla 5.9.

Una vez más se observa en la Tabla 5.9 como el error en el caso de SD-SLAM+ es bastante menor, gracias a que consigue seguir haciendo *tracking* en los momentos donde no hay textura consiguiendo una trayectoria más parecido a la real.

Test Number	rmse (m)	mean (m)	median (m)	std (m)	min (m)	max (m)	sse (m)
Test 1	0.1863	0.1673	0.1199	0.0820	0.0352	0.3153	19.1323
Test 2	0.1960	0.1781	0.1609	0.0819	0.0335	0.3265	21.2211

**Tabla 5.10:** Test 2: Evolución del ATE en SD-SLAM+

La tabla anterior, Tabla 5.10, muestra como ha evolucionado el error desde el test 1 al test 2. Se aprecia como a pesar de introducir más intervalos donde el *tracking* de DIFODO tiene que entrar en juego apenas el error se ha visto alterado y los resultados son prácticamente similares. Esto es lógico pues ambos algoritmos DIFODO y SD-SLAM tenían un rendimiento similar para esta secuencia como se vió en la sección 5.4 por lo que usar un algoritmo u otro no debería alterar el error en gran medida.

### Test 3

- **Intervalos sin textura simulados (segundos): [4-8]**
- **Descripción:** En esta ocasión se va a analizar el comportamiento cuando no hay textura de forma prolongada en el tiempo. El objetivo es ver como se comporta SD-SLAM+ cuando DIFODO se usa de forma prolongada.

## 5.5. SD-SLAM FRENTE A SD-SLAM+

---

Version	rmse (m)	mean (m)	median (m)	std (m)	min (m)	max (m)	sse (m)
SD-SLAM	1.2528	0.8894	0.1415	0.8823	0.0350	2.1653	866.4533
SD-SLAM+	0.1289	0.1117	0.0895	0.0643	0.0330	0.3389	9.1240

**Tabla 5.11:** Comparación del ATE en el test 3: SD-SLAM vs SD-SLAM+

Los resultados para SD-SLAM siguen siendo los mismos por las razones que se vieron en el anterior test.

En la Tabla 5.11 podemos volver a observar como los resultados del SD-SLAM+ son mucho mejores que el algoritmo original.

Test Number	rmse (m)	mean (m)	median (m)	std (m)	min (m)	max (m)	sse (m)
Test 1	0.1863	0.1673	0.1199	0.0820	0.0352	0.3153	19.1323
Test 2	0.1960	0.1781	0.1609	0.0819	0.0335	0.3265	21.2211
Test 3	0.1289	0.1117	0.0895	0.0643	0.0330	0.3389	9.1240

**Tabla 5.12:** Test 3: Evolución del ATE en SD-SLAM+

La tabla anterior, Tabla 5.12, muestra como ha evolucionado el error desde el test 1 al test 3. Es curioso observar la relación del error entre el test 2 y el test 3 pues los intervalos de tiempo en ambos donde no hay textura suman un total de 3 y 4 segundos respectivamente durante un intervalo similar, segundos 4-9 en el test 2 y segundos 4-8 en el test 3. A pesar de ser tan parecidos el error es un 30 % menor en el test 3 frente al test 2. Como se pudo observar en la sección 5.4, en esta secuencia el rendimiento de ambos algoritmos, DIFODO y SD-SLAM es similar por lo que una posible explicación es que en ese intervalo entre los segundos 4-8 de esta secuencia, las estimaciones de pose de DIFODO son mejores o suplen las deficiencias posteriores en la estimación de la pose.

## Test 4

- **Intervalos sin textura simulados (segundos):** [4-5, 8-9, 12-13, 16-17]
- **Descripción:** En esta ocasión se aumenta el número de intervalos sin textura durante la secuencia, a lo largo de toda esta. El objetivo es ver como se comporta SD-SLAM+ en las partes finales de la secuencia.

Version	rmse (m)	mean (m)	median (m)	std (m)	min (m)	max (m)	sse (m)
SD-SLAM	1.2528	0.8894	0.1415	0.8823	0.0350	2.1653	866.4533
SD-SLAM+	0.3109	0.2845	0.2816	0.1253	0.03501	0.4830	54.5273

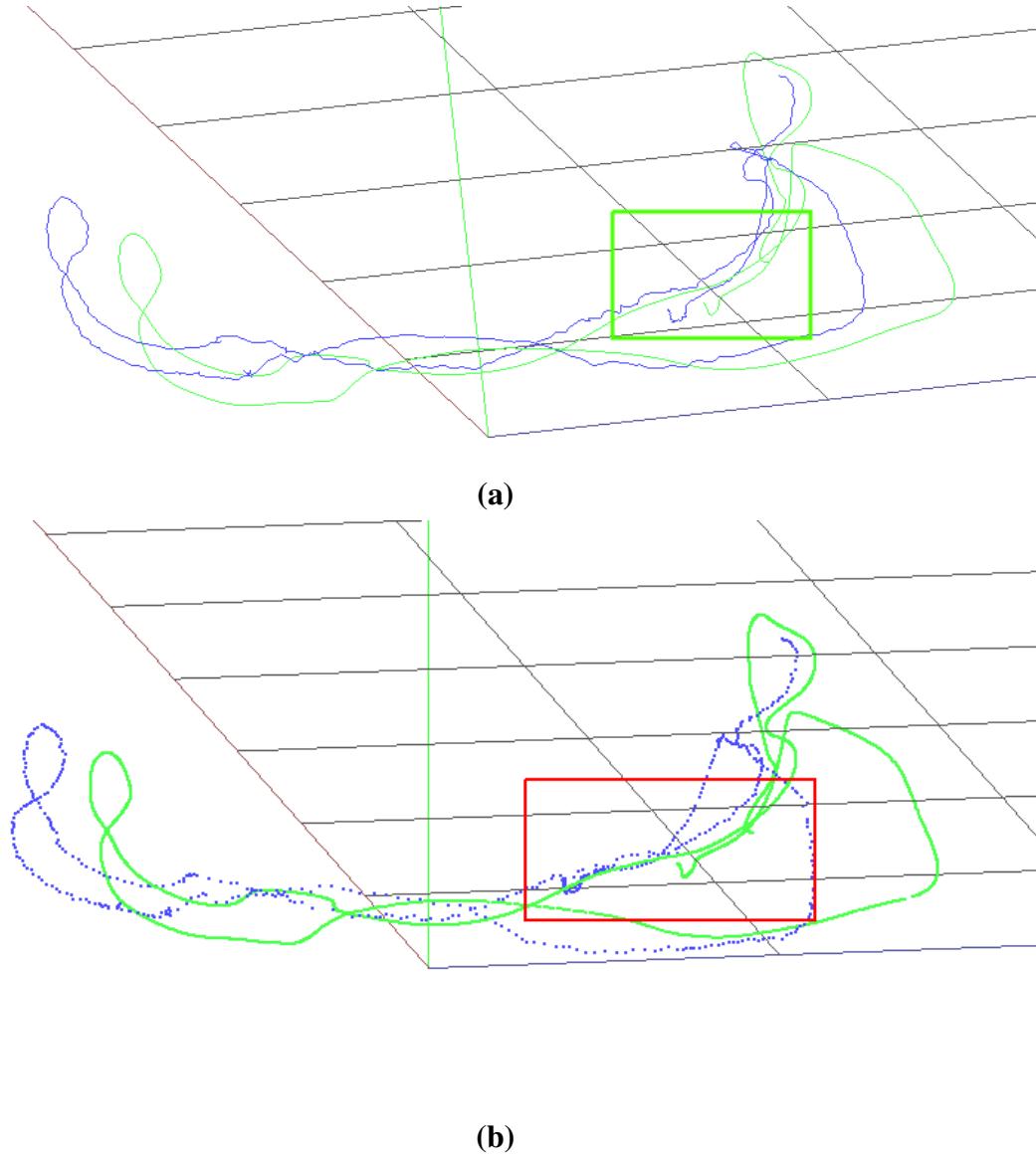
**Tabla 5.13:** Comparación del ATE en el Test 4: SD-SLAM vs SD-SLAM+

Se observa de nuevo en la Tabla 5.13 como el error en el caso de SD-SLAM+ es bastante menor, sin embargo se aprecia un aumento en el error que merece ser estudiado con más detalle comparandolo con los anteriores resultados, esto se aprecia en la siguiente Tabla 5.14:

Test Number	rmse (m)	mean (m)	median (m)	std (m)	min (m)	max (m)	sse (m)
Test 1	0.1863	0.1673	0.1199	0.0820	0.0352	0.3153	19.1323
Test 2	0.1960	0.1781	0.1609	0.0819	0.0335	0.3265	21.2211
Test 3	0.1289	0.1117	0.0895	0.0643	0.0330	0.3389	9.1240
Test 4	0.3109	0.2845	0.2816	0.1253	0.0350	0.4830	54.5273

**Tabla 5.14:** Test 4: Evolución del ATE en SD-SLAM+

En el test 4 se aprecia un aumento del error frente al test 2 de un 50 %. De los cuatro intervalos sin textura de este test, los dos primeros, son parte de los intervalos usados en el test 2, por lo que el aumento del error se debe a los intervalos posteriores. En la Figura 5.4 se puede apreciar como en la parte final la estimación no es tan buena en el test 4, imagen **(b)** comparado con el test 2 en **(a)**. El uso de DIFODO en la parte final parece que no es tan bueno como el de SD-SLAM original, probablemente debido al hecho de que SD-SLAM en esta zona final se apoya más en los cierres de bucle para mejorar sus estimaciones, algo que DIFODO no puede hacer al ser solo odometría.



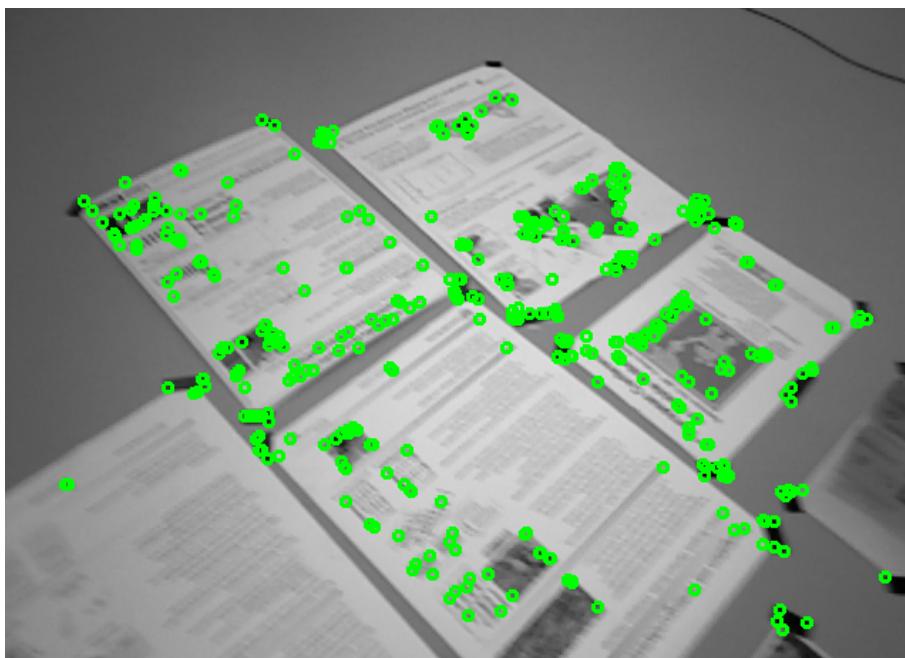
**Figura 5.4:** Trajetorías de los algoritmos mostradas en el programa slam-testbed. En verde se muestra el *ground-truth*, en azul la trayectoria estimada en cada caso. Las figura (a) corresponde a la estimación de SD-SLAM+ en el test 2 y la figura (b) a la la estimación de SD-SLAM+ en el test 4.

### 5.5.3. Experimento: Rendimiento de SD-SLAM+ en entornos difíciles

En la sección anterior, se ha evaluado SD-SLAM+ en un entorno donde había mucha información de textura y también de estructura o textura de profundidad. Esto daba como resultado que independientemente del método usado para la estimación de la pose el rendimiento fuera bueno en todos los casos. En concreto en los anteriores test se comprobaba como la falta de textura podía ser suplida por la estructura o información de profundidad de la escena.

En este nuevo experimento, se va a comprobar como se desenvuelve el nuevo SD-SLAM+ en un entorno mucho más difícil donde no hay estructura y se simularán intervalos donde tampoco haya textura. De forma que habrá intervalos de tiempos donde no habrá ni textura ni estructura, un entorno de lo más desafiante.

**rgbd\_dataset\_freiburg3\_nostructure\_texture\_far.bag** es el nombre de la secuencia que se usará durante los siguientes experimentos. La secuencia consiste, como se puede ver en Figura 5.5, en una serie de papeles con información en ellos que aportan mucha textura en la escena. Por otro lado la única estructura que se encuentra en la escena es un plano que es el suelo. Un plano en estructura es el sinónimo para la textura de una escena completamente homogénea por lo que las estimaciones de DIFODO en este caso tendrán bastante error, como ya se vió en la sección 5.4.



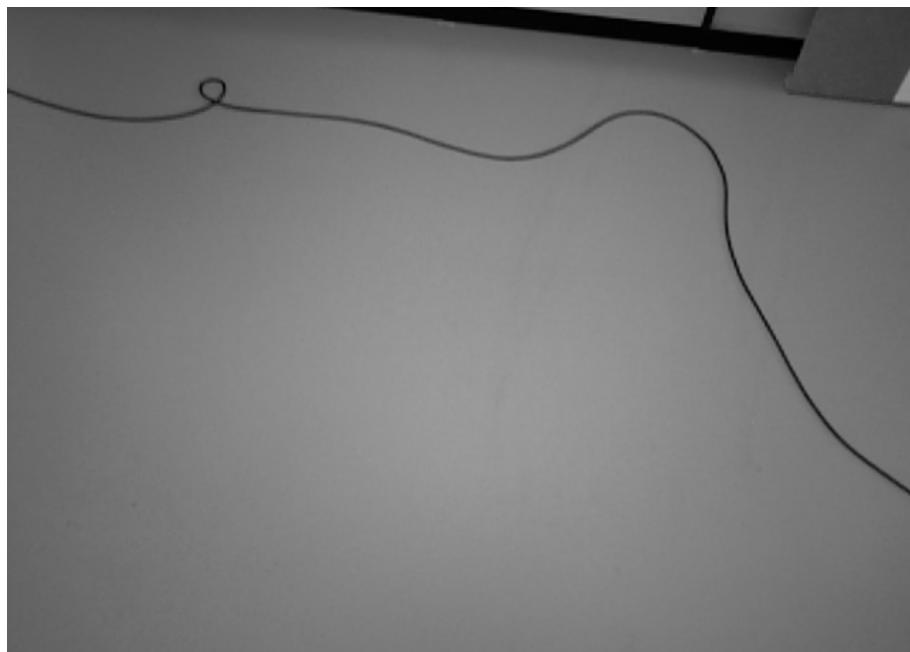
**Figura 5.5:** SD-SLAM aplicado en **rgbd\_dataset\_freiburg3\_nostructure\_texture\_far.bag**

Es una trayectoria rectilínea, empieza en el punto A y acaba en el punto B sin que haya ninguna relocalización ni posibilidad de ello. Hacia el final de la trayectoria, Figura 5.6, los papeles se pierden de la vista de la cámara quedando solo un cable y el suelo homogéneo. Esto como veremos a continuación supone un problema al *tracking* original de SD-SLAM pues la ausencia de textura impide que este consiga estimar la posición. La secuencia es de 16.1 segundos de duración, sin embargo, aproximadamente los primeros 4 segundos de este secuencia y del resto,

## 5.5. SD-SLAM FREnte A SD-SLAM+

---

carecen de imágenes. Esto es así para que a la hora de grabar el hardware tenga un tiempo de estabilización o calentamiento en el arranque, por lo que la duración real sería de 12 segundos aproximadamente.



**Figura 5.6:** Final de la secuencia `rgbd_dataset_freiburg3_nostructure_texture_far.bag`

### Test 5

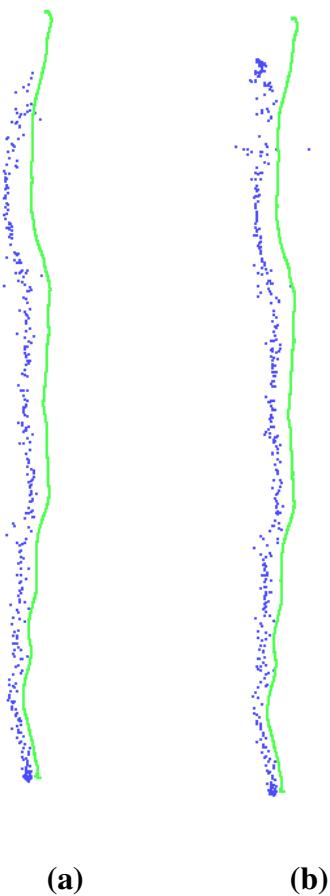
- **Intervalos sin textura simulados (segundos): []**
- **Descripción:** No hay intervalos sin textura simulados, ya que la propia secuencia ya presenta una zona hacia el final donde SD-SLAM no encuentra textura suficiente para estimar la posición por lo que DIFODO se usará. Es un buen ejemplo real para comparar sin necesidad de simulaciones el rendimiento del nuevo SD-SLAM+.

Version	rmse (m)	mean (m)	median (m)	std (m)	min (m)	max (m)	sse (m)
SD-SLAM	1.3445	0.5417	0.0890	1.2306	0.0136	3.9141	705.07
SD-SLAM+	0.1273	0.1072	0.0941	0.0686	0.0102	0.3038	6.7991

**Tabla 5.15:** Comparación del ATE en el test 5: SD-SLAM vs SD-SLAM+

Como se observa en la Tabla 5.15, el error es mucho menor en el caso de SD-SLAM+, sin embargo la estimación no es mucho más distinta si nos fijamos visualmente en la Figura 5.7. Unos puntos al final de la Figura 5.7 **(b)** que representan las estimaciones de DIFODO cuando los cables desaparecen de escena es la única diferencia, sin embargo hay una gran diferencia en el error entre ambos algoritmos. Esto se debe a que cuando SD-SLAM se pierde debido a la falta de textura, las siguientes posiciones que estima no se encuentran al final de la secuencia, sino que publica como pose el origen. Esto provoca que cuando se calcule al ATE se estén comparando posiciones del final de la secuencia con el inicio por lo que el error crece muchísimo. Esto se aprecia en el valor máximo de 3.9m de la Tabla 5.15, que corresponde a la comparación entre las posición finales y las poses del origen cuando SD-SLAM está perdido. También se aprecia en como el error medio de SD-SLAM es muy pequeño y similar al de SD-SLAM+ pero al contar con estas errores finales tan grandes la media y el rmse se ven afectados.

SD-SLAM+ ayuda a seguir estimando posiciones en el final de la trayectoria lo que evita que el error crezca y se mantenga estable. Aunque, como se aprecia en la Figura 5.7 **(b)**, las estimaciones no son muy acertadas como en el experimento anterior Subsección 5.5.2 debido a la ausencia de estructura de esta secuencia.



**Figura 5.7:** Trajetorías de los algoritmos mostradas en el programa slam-testbed. En verde se muestra el *ground-truth*, en azul la trayectoria estimada en cada caso. Las figura (a) corresponde a la estimación de SD-SLAM y la figura (b) a la estimación de SD-SLAM+.

## Test 6

- **Intervalos sin textura simulados (segundos): [4-5]**
- **Descripción:** En este nuevo test se pretende poner aún más a prueba a SD-SLAM+ añadiendo un intervalo durante el cual no hay textura.

Version	rmse (m)	mean (m)	median (m)	std (m)	min (m)	max (m)	sse (m)
SD-SLAM	2.5749	2.0802	2.2577	1.5174	0.0140	4.3386	2731.6205
SD-SLAM+	0.3109	0.2845	0.2816	0.1253	0.0350	0.4830	54.5273

**Tabla 5.16:** Comparación del ATE en el test 6: SD-SLAM vs SD-SLAM+

De nuevo el error cometido por SD-SLAM+ es mucho menor. En este caso SD-SLAM no ha sido capaz de continuar con la estimación a partir del intervalo en el que la textura se pierde. Como la secuencia no presenta posibles relocalizaciones no se vuelve a estimar la posición como se aprecia en la figura Figura 5.8 y la Figura 5.9 de forma visual.

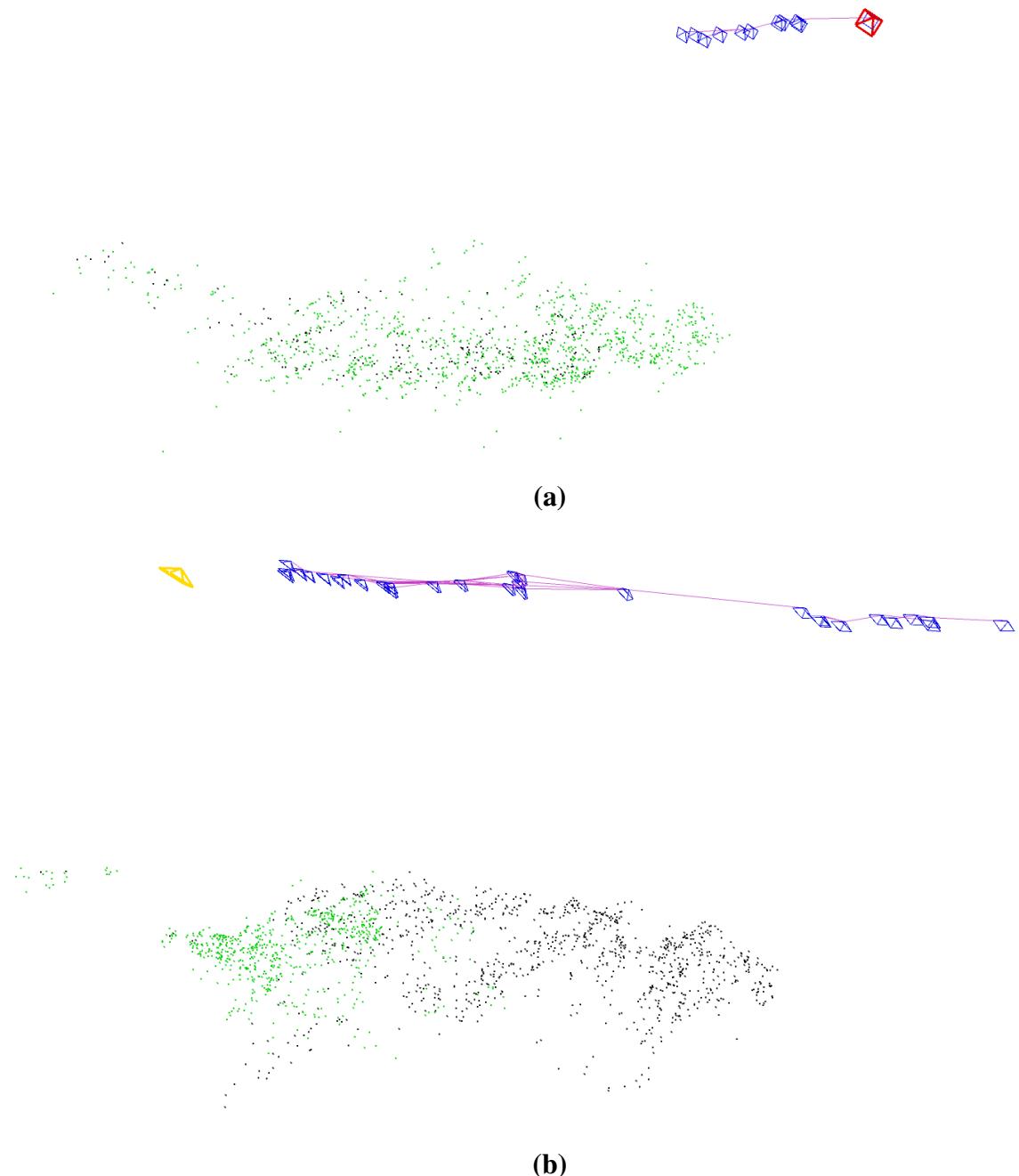
En el caso de SD-SLAM+ el error cometido es mucho menor, sin embargo se pueden apreciar en la zona resaltada de la Figura 5.8 **(b)**, una serie de estimaciones que se acumulan. Al no contar con más estructura que el plano del suelo, las estimaciones de DIFODO no son muy fiables y las nuevas estimaciones no siguen la trayectoria sino que se acumulan en torno al último punto conocido. Sin embargo como es solo un segundo cuando se recupera la textura es capaz de seguir utilizando el *tracking* original de SD-SLAM y la estimación global es mucho mejor que con SD-SLAM.



**Figura 5.8:** Trajetorías de los algoritmos mostradas en el programa slam-testbed. En verde se muestra el *ground-truth*, en azul la trayectoria estimada en cada caso. La figura **(a)** corresponde a la estimación de SD-SLAM y la figura **(b)** a la estimación de SD-SLAM+.

## 5.5. SD-SLAM FRENTE A SD-SLAM+

---



**Figura 5.9:** Trajetorías y reconstrucción de los algoritmos en la interfaz SD-SLAM. La figura (a) corresponde a D-SLAM y la figura (b) a SD-SLAM+.

En la Tabla 5.17 se compara el rendimiento de SD-SLAM+ en el test 5 y el test 6. En esta se observa como introducir tan solo un segundo en el *tracking* aumenta el error en casi un 300 %.

El error medio pasa a ser de casi 0.30m frente a los 0.10m de la secuencia original en el test 5. Este aumento de 20cm en el error viene dado por ese segundo de intervalo donde la posición casi no avanza debido a la mala estimación de DIFODO, mientras que en la secuencia original durante ese segundo se avanzarán estos 20cm. Esto produce que las estimaciones de SD-SLAM vayan por detrás de la secuencia original 20cm para el resto de estimaciones aumentando el error en esta cantidad.

Test Number	rmse (m)	mean (m)	median (m)	std (m)	min (m)	max (m)	sse (m)
Test 5	0.1273	0.1072	0.0941	0.0686	0.0102	0.3038	6.7991
Test 6	0.3109	0.2845	0.2816	0.1253	0.0350	0.4830	54.5273

**Tabla 5.17:** Test 6: Evolución del ATE en SD-SLAM+

## Test 7

- **Intervalos sin textura simulados (segundos):** [4-5, 6-7, 8-9]
- **Descripción:** Se añaden hasta 3 intervalos donde no hay textura.

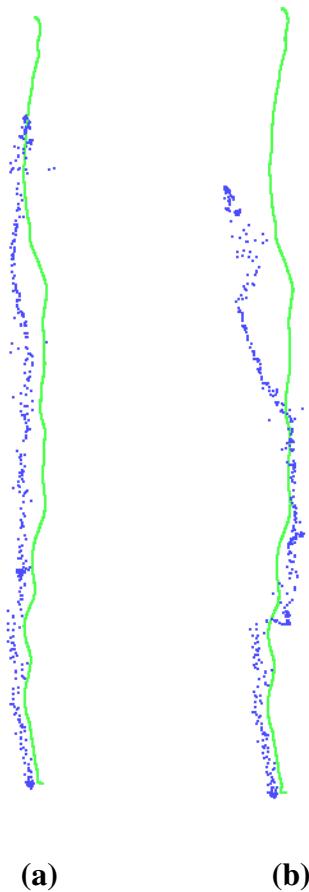
Version	rmse (m)	mean (m)	median (m)	std (m)	min (m)	max (m)	sse (m)
SD-SLAM	2.5749	2.0802	2.2577	1.5174	0.0140	4.3386	2731.6205
SD-SLAM+	0.5854	0.4761	0.5655	0.3406	0.0098	1.0465	148.4094

**Tabla 5.18:** Comparación del ATE en el test 7: SD-SLAM vs SD-SLAM+

El rendimiento de SD-SLAM sigue siendo el mismo que en el test anterior como era de esperar al no poder relocalizarse. De nuevo SD-SLAM+ lo hace mejor como era de esperar. Aunque el rendimiento no es para nada bueno, casi 0.5m de error medio por posición, algo que para algoritmos de SLAM es demasiado. Se puede visualizar en la Figura 5.10 (b) como la trayectoria que se sigue esta vez es mucho más irregular que la que se observaba en el test anterior Figura 5.10 (a). Cada vez que hay un intervalo donde se utiliza DIFODO, se observa o bien una acumulación de puntos o un desvío en la trayectoria original. Es interesante observar como en el último intervalo, se produce una rotación en la orientación por parte de DIFODO que produce que el *tracking* de SD-SLAM, una vez recuperada la información visual, estime

### 5.5. SD-SLAM FREnte A SD-SLAM+

un desplazamiento rectilíneo pero en una dirección incorrecta, alejando aún más las siguientes estimaciones.



**Figura 5.10:** Trajetorías de los algoritmos mostradas en el programa slam-testbed. En verde se muestra el *ground-truth*, en azul la trayectoria estimada en cada caso. Las figura **(a)** corresponde a la estimación de SD-SLAM+ para el test 6 y la figura **(b)** a la la estimación de SD-SLAM+ para el test 7.

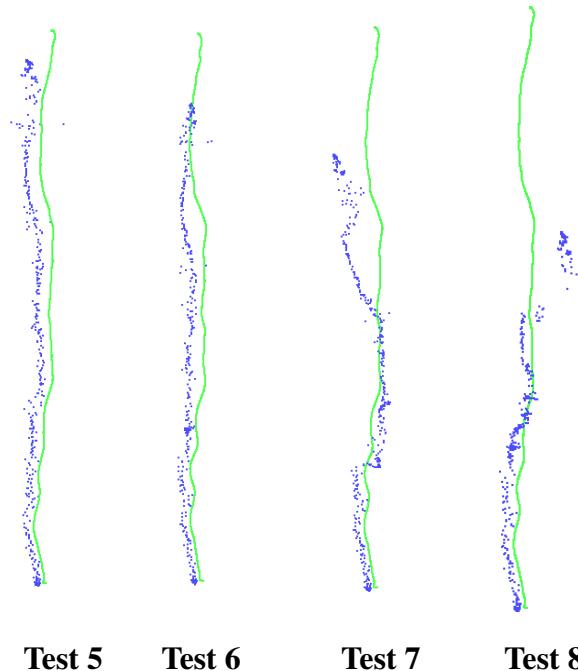
Test 8

- **Intervalos sin textura simulados (segundos):** [4-9]
  - **Descripción:** Esta vez se va a probar como afecta un largo intervalo durante el cual no hay textura en esta secuencia que tampoco cuenta con estructura.

Test Number	rmse (m)	mean (m)	median (m)	std (m)	min (m)	max (m)	sse (m)
Test 5	0.1273	0.1072	0.0941	0.0686	0.0102	0.3038	6.7991
Test 6	0.3109	0.2845	0.2816	0.1253	0.0350	0.4830	54.5273
Test 7	0.5854	0.4761	0.5655	0.3406	0.0098	1.0465	148.4094
Test 8	0.9999	0.7783	0.7649	0.6277	0.0131	1.6505	383.9896

**Tabla 5.19:** Test 8: Evolución del ATE en SD-SLAM+

El error que se obtiene en este test 8 es de cerca del 0.8m de media como se aprecia en Tabla 5.19. Como era de esperar para esta secuencia cuanto más tiempo se esté estimando posiciones con DIFODO mayor va a ser el aumento del error. En la Tabla 5.19 se aprecia como según se ha ido aumentando el tiempo donde se utilizaba DIFODO el error ha ido aumentando. En la Figura 5.11 se puede visualizar como con la inclusión en cada nuevo test de un mayor intervalo de tiempo sin textura, los estimaciones han ido degradandose.

**Figura 5.11:** Comparativa de las trayectorias de los algoritmos mostradas en el programa slam-testbed. En verde se muestra el *ground-truth*, en azul la trayectoria estimada en cada caso.

#### 5.5.4. Experimento: Rendimiento de SD-SLAM+ en entornos con zonas de baja textura

En este experimento se comprueba SD-SLAM en un entorno real, donde en determinado momento, el algoritmo original se pierde, siendo incapaz de estimar nuevas posiciones debido a la falta de textura. Este experimento muestra un caso real donde se pierde la textura de forma natural algo que ocurre con cierta frecuencia en ciertos entornos.

**rgbd\_dataset\_freiburg3\_floor.bag** es el nombre de la secuencia evaluada en este experimento. Es una secuencia de larga duración, de 49.3 segundos donde los primeros 4 segundos de nuevo no cuentan con imágenes. La secuencia es una trayectoria circular mientras se recorre una habitación con diversos objetos que aportan mucha textura y estructura. Sólo hay un momento donde el aporte de textura decrece como se aprecia en . Al ser una trayectoria circular, por el final de la secuencia se vuelve a visitar una zona ya conocida buscando producir un cierre de bucle.



**Figura 5.12:** Momento en el cual la textura decae en la secuencia **rgbd\_dataset\_freiburg3\_floor.bag**. Imagen perteneciente a la GUI de SD-SLAM+

**Test 9**

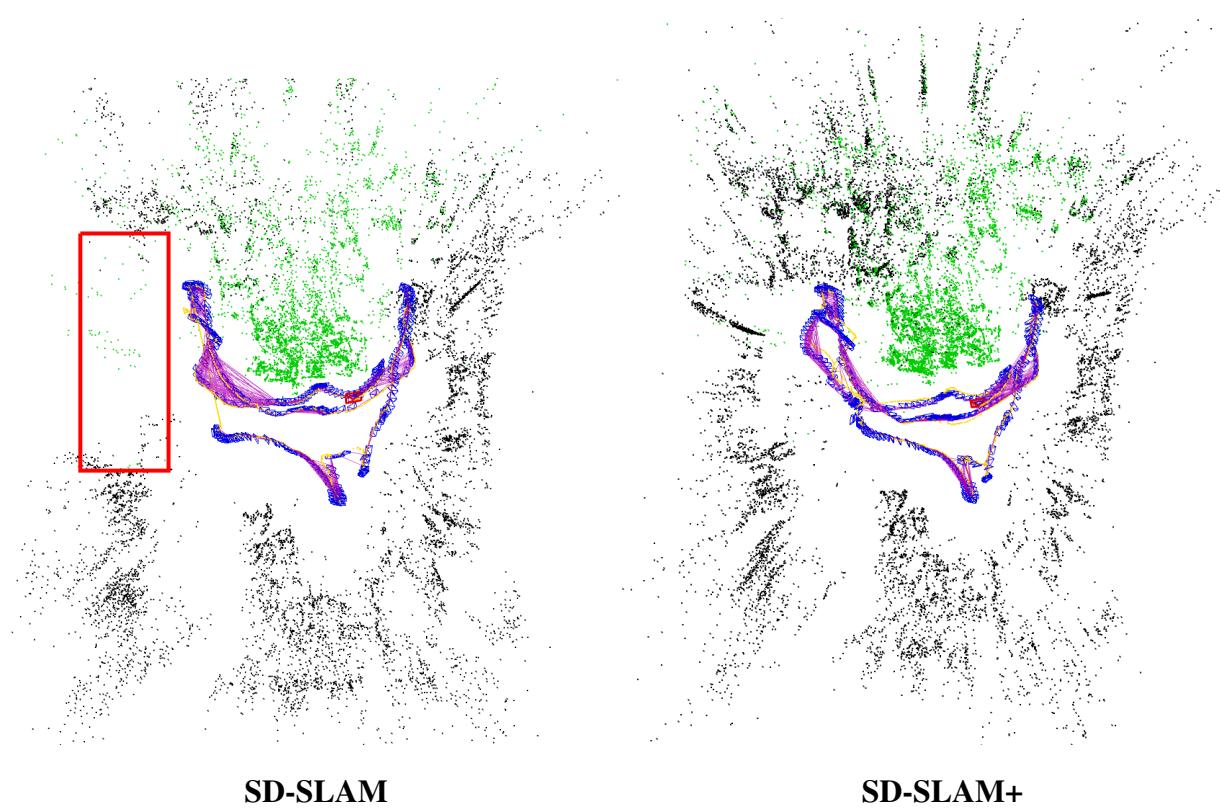
Es este test no ha habido necesidad de simular la perdida de textura como en los tests anteriores, sino, que como se ha mostrado en la figura anterior, de por si la secuencia cuenta con una zona donde la textura es bastante pobre. Esto produce que el algoritmo original SD-SLAM se pierda desde ese momento hasta volver a una zona conocida unos segundos después.

Version	rmse (m)	mean (m)	median (m)	std (m)	min (m)	max (m)	sse (m)
SD-SLAM	0.2909	0.1705	0.1022	0.2192	0.0195	1.2202	98.2218
SD-SLAM+	0.1111	0.1042	0.1011	0.0385	0.0267	0.2188	15.9222

**Tabla 5.20:** Comparación del ATE en el test 9: SD-SLAM vs SD-SLAM+

En la tabla 5.20 se hace una comparación entre las dos versiones de SD-SLAM. Como consecuencia de esta perdida de textura hacia el final de la secuencia, SD-SLAM tiene un error medio y de mínimos cuadrados mucho más alto que la versión mejorada SD-SLAM+. Sin embargo, si uno se fija en la mediana se obtienen valores casi idénticos. Esto es así puesto que ambos algoritmos utilizan la información o la estimación de pose de SD-SLAM hasta que se llega a la zona de la figura 5.12. En ese punto, SD-SLAM se pierde incapaz de hacer una estimación correcta en la pose debido a la ausencia de textura y empieza a acumular error hasta el momento de relocalizarse hacia el final de la secuencia. Esto se puede observar en el error máximo para SD-SLAM de 1.22m frente a los 0.21m de error máximo de SD-SLAM+. Por otro lado, SD-SLAM+ mantienen constante su estimación, siendo la media, mediana y el error por mínimos cuadrados muy similares, en torno a los 0.10m de error. Esto lo consigue utilizando la estimación de pose de DIFODO en el momento donde SD-SLAM no es capaz de estimar la pose.

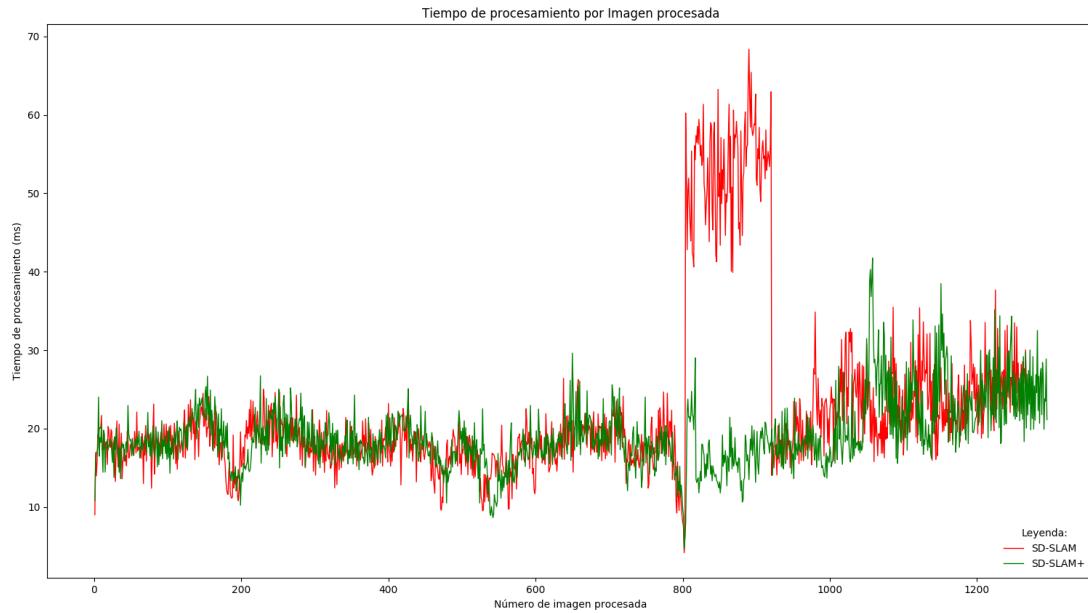
No solo el error se ve afectado, pues mientras que SD-SLAM se encuentra perdido, este no es capaz de hacer el mapeado hasta que se reencuentra de nuevo, por lo que perdidas de textura momentáneas afectan negativamente a la reconstrucción de entornos 3D también, como se puede ver en la figura 5.13. Existe una zona recuadrada en rojo en la cual no hay puntos en el espacio puesto que durante esa parte el algoritmo se encontraba perdido intentando relocalizarse.



**Figura 5.13:** Comparativa de las trayectorias y reconstrucción de los algoritmos SD-SLAM y SD-SLAM+.

### 5.5.5. Evaluación de tiempo de procesamiento

A continuación, en esta sección se va a evaluar el coste computacional de añadir DIFODO a SD-SLAM, comparando los tiempos de procesamiento de SD-SLAM y SD-SLAM+. Para ello se va a utilizar la secuencia utilizada en el anterior **Test 9, rgbd\_dataset\_freiburg3\_floor.bag**. Para este experimento, se ha usado DIFODO con una resolución de 320x240 y cinco niveles de pirámide.



**Figura 5.14:** Comparativa del tiempo de procesamiento por imagen para la secuencia `rgbd_dataset_freiburg3_floor.bag`

La figura anterior 5.14, representa el tiempo de procesamiento por imagen para la secuencia `rgbd_dataset_freiburg3_floor.bag`, con una duración de 1360 imágenes o *frames*. El tiempo de procesamiento por *frame* es similar para ambos, SD-SLAM y SD-SLAM+, manteniéndose durante casi toda la secuencia en torno a los 20ms. Sin embargo, se observa un periodo donde hay un aumento importante del tiempo de procesamiento para SD-SLAM, de la imagen 800 a la 950 aproximadamente. Durante estos 5 segundos, SD-SLAM pasa a tener un tiempo de procesamiento del doble con respecto al resto de la secuencia, sobre pasando los 30ms que precisa el tiempo real. La razón de este aumento es que SD-SLAM se pierde, imposibilitandole la estimación de nuevas poses, debido a la falta de textura en la escena, como se discutió en el anterior **Test 9**.

Cuando SD-SLAM se pierde, este pasa al modo de relocalización, intentando relocalizarse con las nuevas imágenes que le llegan, utilizando todos los *keyframes* guardados que tiene en el mapa para buscar emparejarlos. Este es un proceso costoso pues si hay muchos *keyframes* con características similares, el proceso de intentar emparejar la imagen actual con cada uno de esos puede llegar a consumir más tiempo del deseado, como en este caso.

Por otro lado SD-SLAM+ al no perderse por usar DIFODO durante ese periodo donde no hay textura, evita entrar en ese modo de relocalización. Se puede apreciar el tiempo de procesamien-

## 5.5. SD-SLAM FRENTE A SD-SLAM+

---

to de DIFODO a partir de la imagen número 800 durante unas 30 imágenes aproximadamente, con un tiempo medio de procesamiento de unos 20ms, similar al que se obtiene con la estimación original de SD-SLAM.

Integrar DIFODO junto a SD-SLAM, no solo no incrementa el coste computacional, sino que lo reduce al evitar entrar en este modo de relocalización. Mientras que la textura se mantenga, el algoritmo original no se verá afectado computacionalmente pues DIFODO solo añade tiempo computacional cuando el *tracking* original de SD-SLAM falla. En la práctica cuando se usa DIFODO el tracking de SD-SLAM original no se usa, por lo que al sustituir uno con otro el tiempo de procesamiento se mantiene estable.

### 5.5.6. Conclusiones

En este capítulo se ha visto como DIFODO es un algoritmo que puede integrarse con SD-SLAM para hacer a este último más robusto frente a situaciones donde falta textura visual. Esto es así, pues cumple con los requisitos de ser un algoritmo de odometría visual que sólo usa información de profundidad y cumple con el requisito de procesamiento en tiempo real.

Se ha comprobado como SD-SLAM es mejor que DIFODO en la mayoría de los casos siempre y cuando la imagen presente información de textura y de estructura suficientes. Sin embargo cuando hay ausencia de textura en la imagen, pero suficiente estructura, DIFODO estima la posición con un menor error que SD-SLAM, el cual directamente no estima posición alguna. Por otro lado, ambos algoritmos fallan cuando no hay estructura ni textura por lo que SD-SLAM+ precisará de al menos estructura o textura en la escena en todo momento para funcionar correctamente.

Finalmente, se ha comparado SD-SLAM contra SD-SLAM+, viendo como en aquellos casos donde hay ausencia de textura en algún momento, SD-SLAM+ tiene un mejor rendimiento al ser capaz de estimar la posición frente a un SD-SLAM que se pierde y solo es capaz de volver a estimar posiciones si se da un cierre de bucle.

En conclusión la integración de DIFODO en SD-SLAM solo trae ventajas y no conlleva ninguna desventaja pues computacionalmente no solo no implica un mayor tiempo de procesamiento sino que ayuda a evitar el elevado tiempo de procesamiento que conlleva el intentar relocalizarse.

# **Capítulo 6**

## **Conclusiones**

### **6.1. Objetivos conseguidos**

### **6.2. Discusión**

### **6.3. Trabajos futuros**

- Permitir que DIFODO incorpore puntos al mapa (aunque estos no se puedan usar posteriormente en ORB)
- La estimacion de DIFODO tambien puede no ser lo suficientemente buena, establecer una forma de determinar cuando la incertidumbre en la estimacion de la pose de DIFODO es lo suficientemente grande como para que el sistema se vaya a un estado de perdido (por falta de infroamcion de textura y estructura en la escena). Hay una métrica en DIFODO que indica esta incertidumbre ya. Sería encontrar un valor de forma empirica.
- Mejorar la estimacion de DIFODO utilizando técnicas propias de SLAM, por ejemplo como se hace en DDS, de forma que se pueda reducir la deriva del algoritmo utilizando la inforamcion de imagenes anteriores. O bien extrayendo puntos caracteristicos de la imagen de porfundidad con descriptores 3D para buscar poder relocalizarse con la info-ramcion de profundidad tambien. El objetivo tener dos algoritmos de SLAM uno basado en RGB y otro en D que puedan complementarse.

### 6.3. TRABAJOS FUTUROS

---

- Permitir que la primera estimacion sea con DIFODO si se empieza en una zona sin textura donde ORB no puede inicializarse debido a la falta de puntos caracteristicos.

# Bibliografía

- [1] Jean-Yves Bouguet et al. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5(1-10):4, 2001.
- [2] M. Jaimez and J. Gonzalez-Jimenez. Fast visual odometry for 3-d range sensors. *IEEE Transactions on Robotics*, 31(4):809–822, 2015.
- [3] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.
- [4] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I, 2004.
- [5] Eduardo Perdices and José Cañas Plaza. Sdvl: Efficient and accurate semi-direct visual localization. *Sensors*, 19:302, 01 2019.
- [6] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.
- [7] D. Scaramuzza and R. Siegwart. Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *IEEE Transactions on Robotics*, 24(5):1015–1026, 2008.
- [8] Hagen Spies, Bernd Jähne, and John L. Barron. Range flow estimation. *Computer Vision and Image Understanding*, 85(3):209 – 231, 2002.
- [9] J. Sturm, W. Burgard, and D. Cremers. Evaluating egomotion and structure-from-motion approaches using the TUM RGB-D benchmark. In *Proc. of the Workshop on Color-Depth*

## BIBLIOGRAFÍA

---

- Camera Fusion in Robotics at the IEEE/RJS International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.
- [10] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment — a modern synthesis. In Bill Triggs, Andrew Zisserman, and Richard Szeliski, editors, *Vision Algorithms: Theory and Practice*, pages 298–372, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.
  - [11] S. Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 203(1153):405–426, 1979.