Universidad Zaragoza
1542

Trabajo Fin de Grado

Detección de Cierre de Bucles Deformables en Secuencias de Endoscopia Médica

Deformable Loop_Closure Detection in Medical Endoscope Sequences

Autor

Iñigo Cirauqui Viloria

Director

José María Martínez Montiel

Escuela de Ingeniería y Arquitectura

2021

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe entregarse en la Secretaría de la EINA, dentro del plazo de depósito del TFG/TFM para su evaluación).

D./Dª. Íñigo Cirauqui Viloria ,en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza, Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado en Ingeniería Mecánica, (Título del Trabajo)

Detección de Cierre de Bucles Deformables en Secuencias de Endoscopia Médica

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 17 de Junio de 2021

Fdo: Íñigo Cirauqui Viloria

Agradecimientos

Agradecemos a IRCAD (Instituto de Investigación contra el Cáncer del Aparato Digestivo) el permitirnos utilizar las secuencias privadas del capítulo 4. Estas imágenes se obtuvieron con la pertinente aprobación ética, contando con todas las directrices internacionales, nacionales y/o institucionales para el cuidado de animales.

We thank IRCAD (Research Institute against Digestive Cancer) for allowing us to use their propietary sequences in the experiments of Chapter 4. These images where adquired after an ethical aproval where all applicable international, national, and/or institutional guidelines for the care and use of animals were followed.

Resumen

Detección de Cierre de Bucles Deformables en Secuencias de Endoscopia Médica

Un software de SLAM Visual procesa una secuencia de imágenes tomada por una cámara en movimiento con trayectoria desconocida a través de un entorno también desconocido. El sistema proporciona un mapa de puntos 3D de la escena y la posición de la cámara respecto de este en tiempo real. En funcionamiento normal, el mapa y la posición de la cámara se calculan mediante un modelo de movimiento basado en escenas anteriores. Si se perdiese la posición relativa de cámara respecto al mapa, este cálculo no sería posible con el mismo algoritmo, iniciándose un proceso de relocalización con el que calcular la posición de la cámara sin datos de movimiento.

ORB-SLAM2 está diseñado para funcionar en escenas de robótica móvil que se desarrollan en entornos rígidos, pero también ha sido evaluado con éxito en escenas deformables propias de la endoscopia médica. En el trabajo se desarrolla una evolución de ORB-SLAM2, con algoritmos que dotan al sistema de la capacidad de relocalizar la cámara en entornos no rígidos. Se describe la implementación que logra la relocalización en escenas deformables, incluyendo una evaluación de prestaciones en escenas in-vivo de endoscopia en animales. Se proporciona el código y se proponen líneas de trabajo futuro orientadas a obtener un sistema de SLAM Visual específico para escenas médicas.

Summary

Deformable Loop_Closure Detection in Medical Endoscope Sequences

A Visual SLAM implementation processes a image sequence fetched from a camera that follows an unknown trajectory through and also unknown environment. The software computes a map of the surrounding environment composed of 3D points, and the camera pose with refference to the map in real time. In normal process, both map and camera pose are calculated via a motion model based on information available from previous scenes. If the camera's relative location from the map, the algorithm lacks the necessary information to continue, and to resume normal operation it performs a relocation process, in which a new camera pose is computed without accounting for the motion model.

ORB-SLAM2 is conceived for mobile robotics and rigid environments, but it's also been successfully tested in deformable scenes. This project adds the capability to relocate the camera in deformable environments. We describe the implementation, and present an experimental evaluation of the same using iv-vivo sequences of animal endoscopies. We provide the code and suggest future work, aiming to build a Visual SLAM specifically designed for medical sequences.

Índice

Capítulo 1

Introducción

Se denomina VSLAM, *Visual Simultaneous Localization And Mapping*, a un conjunto de algoritmos empleados en robótica y realidad aumentada que permiten generar un mapa tridimensional de un entorno a partir de imágenes tomadas por una o varias cámaras en movimiento, proporcionando una estimación de la posición de la cámara respecto del mapa. En este proyecto se trabaja con sistemas de SLAM Visual monocular que emplean una única cámara.

El primer VSLAM monocular que funciona en tiempo real en escenas genéricas fue presentado por A. Davison [1], funcionando con un Filtro de Kalman Extendido (Extended Kalman Filter, EKF) [2] que lleva a cabo una actualización probabilística del mapa generado. Cuenta con inicialización robusta, pero solo gestiona en tiempo real un mapa de pocos cientos de puntos. Avances sobre el sistema EKF SLAM han sido presentados por [3, 4], aumentando la robustez y dotando de capacidad para operar en exteriores.

Un salto significativo en el VSLAM monocular llega con PTAM (*Parallel Tracking and Mapping*) [5], el primer método basado en KeyFrames. Su algoritmo permite realizar, en tiempo real, todas las etapas del proceso de reconstrucción fotogramétrica de la escena [6]: emparejamiento de puntos, orientación de la cámara, inicialización y optimización no lineal del mapa (*Bundle Adjustment*, BA), gestionando mapas de miles de puntos y con una localización de cámara muy precisa.

Recientemente, el sistema ORB-SLAM [7] mejora las prestaciones de PTAM al añadir cerrado de bucles y relocalización robusta en tiempo real. Para construir el mapa emplea puntos ORB [8], invariantes a rotación y cambios de escala, con un rendimiento similar a los puntos SIFT [9] (método de referencia en detección de puntos de interés) pero a menor coste computacional.

ORB-SLAM2 [10] es una evolución del primero, que mejora sus prestaciones y añade funciones de realidad aumentada. Al igual que los métodos anteriores, está diseñado para operar en escenas de robótica desarrolladas en entornos rígidos. En el trabajo se actualiza el sistema para permitir el funcionamiento en entornos deformables propios de la endoscopia médica, manteniendo el Tracking de la cámara y un mapa con reducido número de espurios. Además, se incluye un método de relocalización en en-

tornos deformables, mediante una detección mejorada de KeyFrames candidatos para la relocalización y un Bundle Adjustment que minimiza conjuntamente el error de reproyección y la energía de deformación elástica de los puntos del mapa, para lo que se lleva a cabo un análisis por elementos finitos del entorno deformable.

Durante el desarrollo del trabajo se ha presentado ORB-SLAM3 [11], una evolución del actual que permite mantener múltiples mapas simultáneamente. Así mismo en el ámbito deformable, en [12] se comienza a construir modelos de secuencias deformables usando elementos planos y recientemente se ha desarrollado DefSLAM [13], una propuesta que trata la posible deformación en el proceso de SLAM con un método basado en plantillas.

1.1. Motivación y objetivos

La endoscopia es una técnica diagnóstica que consiste en la introducción de una única cámara (endoscopio) a través de un orificio natural o provocado para visualizar una cavidad corporal y recopilar información sobre ella. El ORB-SLAM ha sido empleado satisfactoriamente con imágenes médicas de cirugía abdominal [14, 15] construyendo mapas relativamente precisos de una cavidad con deformación.

Cuando se recorren áreas ya exploradas, o si se pierde la posición de la cámara, el sistema encuentra dificultades para reconocer la escena y proporcionar una posición precisa para la cámara, debido a la deformación sufrida por la escena entre el momento de construcción del mapa y el momento en que se revisita ese entorno. Nuestra contribución es dotar al sistema ORB-SLAM2 de la capacidad para llevar a cabo relocalizaciones precisas en un entorno deformable. Nuestra metodología consiste en hacer la suposición de que la escena es rígida para obtener una aproximación inicial de la posición de la cámara mediante DBoW2 [16], un algoritmo basado en bolsa de palabras. Posteriormente la nube de puntos se emplea para construir un modelo de elementos finitos basado en prismas triangulares o hexaedros y se lleva a cabo una optimización no lineal que deforma el mapa guardado, minimizando conjuntamente el error de reproyección de los puntos 3D y la energía de deformación fruto de alterar el mapa guardado hasta que se aproxime a la escena vista con la pose estimada rígidamente.

El objetivo del trabajo es añadir las estructuras necesarias para permitir este funcionamiento. Las conclusiones están avaladas por una verificación experimental sobre secuencias *in vivo* de endoscopia del abdomen de un cerdo. Se establecen como objetivos intermedios:

1. Adaptar los parámetros del sistema ORB-SLAM2 para procesar entornos deformables.

2. Evaluar la capacidad de relocalización del sistema original empleando secuencias de endoscopia.

3. Mejorar la detección de KeyFrames candidatos a la relocalización.

4. Construir un modelo de elementos finitos de la escena.

5. Verificar los KeyFrames candidatos mediante una optimización no lineal de pose de cámara y posición 3D de los puntos con una función de coste que incluye el cálculo de la energía de deformación elástica de la escena.

Hasta nuestro conocimiento, es la primera vez que se configura un sistema de relocalización de estas características en secuencias de endoscopia médica. En la verificación experimental analizamos secuencias *in vivo* de diferentes *datasets* para dar respaldo experimental a las conclusiones.

1.2. Estructura

La sección 2 presenta la metodología de trabajo y el ajuste realizado sobre el software original para que este funcione en escenas deformables, además de describir los procesos de cierre de bucle y relocalización, profundizando en los puntos sobre los que actúa el trabajo. La sección 3 describe los ajustes que permiten la relocalización en escenas no rígidas de endoscopia, presenta cambios en la detección de KeyFrames candidatos para la relocalización, en el funcionamiento del algoritmo de estimación de pose inicial y en la optimización no lineal. La sección 4 presenta la evaluación experimental y la mejora obtenida. Por último, la sección 5 está dedicada a las conclusiones y al planteamiento de posible trabajo futuro.

El software modificado se encuentra disponible en GitHub [17]. Los anexos proporcionan detalles sobre el software utilizado y las secuencias empleadas en el análisis experimental.

Capítulo 2

ORB-SLAM2 y Puesta en Marcha

ORB-SLAM2 [10] es un sistema de Visual SLAM que basa su funcionamiento en la detección y el emparejamiento de puntos de imagen ORB. El sistema lleva a cabo las siguientes tareas:

Inicialización - Al lanzar el programa no se dispone de información del entorno. La inicialización crea un mapa de partida a partir de los dos primeros KeyFrames, que son seleccionados y empleados para triangular puntos 3D.

Tracking - Cuando se dispone de un mapa, la cámara toma una imagen y detecta en ella los puntos que ya están incluidos en este, a partir de los cuales se estiman la posición y la orientación de la cámara respecto del mapa.

Mapping - Proceso que construye el mapa de la escena a partir de las imágenes. Para ello es necesario identificar puntos correspondientes entre un conjunto de imágenes seleccionadas denominadas KeyFrames. Estas correspondencias se utilizan para triangular en 3D los puntos del mapa. El refinado de la posición de puntos y KeyFrames se realiza con un proceso de optimización no lineal llamado Bundle Adjustment (BA). El mapa crece conforme la cámara explora zonas nuevas y el software decide cuándo es necesario añadir nuevos KeyFrames y puntos que las describan. Además, la gestión del mapa incluye la eliminación de puntos y KeyFrames erróneos o redundantes.

Relocalización - En la operación normal, para calcular la posición de la cámara en cada frame, se dispone de la que tenía en el anterior. Puede darse el caso de que esta no se encuentre disponible y haya que hacer una localización desde cero, este proceso se llama relocalización, también conocido como el caso del robot secuestrado.

Cerrado de bucle - El mapa crece a medida que se exploran nuevas zonas. Si se vuelve a una zona explorada previamente el sistema lo detecta, calculando y compensando el error de deriva acumulado en la trayectoria seguida.

Los puntos de interés en las imágenes, o *KeyPoints*, son la espina dorsal del funcionamiento del sistema. Son localizados mediante el detector oFAST multiescala [8] y, para cada uno de ellos, se dispone de un descriptor binario ORB de 256 bits invariante a la rotación. Esos descriptores se comparan entre sí para detectar puntos correspondientes y se produce el emparejamiento cuando se tiene similitud suficiente de acuerdo con la distancia de Hamming, esto es, número de bits que son diferentes entre los dos descriptores. Un ejemplo se presenta en la Figura 2.1 en la que el descriptor 1 se empareja con el 2, pero no con el 3.

2.1. Sintonía y calibración de los endoscopios

El sistema ORB-SLAM2 ha sido diseñado para procesar escenas rígidas propias de robótica. Para llevarlo a un entorno deformable, se ha implementado una versión ajustada de las modificaciones descritas en [14] para los procesos de Tracking y Mapping de ORB-SLAM.

Se han modificado parámetros adicionales, prestando especial atención al proceso de inicialización, ya que se ha encontrado que, si bien el sistema es capaz de inicializar en toda ejecución, en ocasiones el primer mapa calculado no representa adecuadamente la escena, lo que causa fallos posteriores en Tracking y Mapping. Para corregir esto se han aumentado el paralaje mínimo necesario para triangular los puntos del mapa inicial y la cantidad de ellos que se demanda, esto elimina las inicializaciones erróneas casi en su totalidad. Las modificaciones respecto al software ORB-SLAM2 original se presentan en la Tabla 2.1.

Por último, dado que las lentes de las cámaras causan distorsión radial y tangencial en las imágenes captadas, para operar correctamente es necesario calcular y compensar estas distorsiones. Para ello, se lleva a cabo un proceso de calibración de cada endoscopio empleando imágenes de patrones de ajedrez captadas por él antes o después de la operación. Estas imágenes se alimentan a un código en base C++, que utiliza funciones de la librería OpenCV para obtener la matriz de cámara y los coeficientes de distorsión de la lente. El código se encuentra publicado en [18].

2.2. Cierres de bucle

El proceso de cierre de bucle opera en un thread independiente y paralelo a los de Tracking y Mapping y es capaz de detectar si la cámara está recorriendo áreas previamente mapeadas, en cuyo caso fusiona porciones semejantes del mapa, corrigiendo y distribuyendo el error acumulado durante la trayectoria. Esto permite eliminar la redundancia de puntos 3D al consolidarlos y reducir su número, lo que facilita un funcionamiento prolongado en el tiempo y menor error de deriva.

El proceso parte de un mapa ya construido de un área del abdomen, que la cámara revisita en un momento posterior de la operación, llegando a él por un camino no

Descripción	Source	Línea	Antes	Ahora	Ud.
Tracking					
Área de búsqueda	ORBmatcher.cc	145	2.5/4.0	3.0/4.5	píxel
Invarianza de escala	Frame.cc	195	1/fe*	0.9/fe*	-
Comparación ORB	ORBmatcher.cc	43	100	95	bit
Gestión del mapa, creación de puntos					
Distancia entre keyframes	LocalMapping.cc	314	0.01	0.05	-
Distancia búsqueda g.epipolar	ORBmatcher.cc	167	3.84	3.84	píxel
Paralaje mínimo	LocalMapping.cc	375	1.1459	1.4035	Deg
Error de reproyección	LocalMapping.cc	413	5.991	0.5991	píxel
Comparación ORB	ORBmatcher.cc	44	50	45	bit
Inicialización					
Fundamental/Homografía	Initializer.cc	113	0.40	0.99	-
Paralaje mínimo (kf)	Initializer.cc	118	1.0	0.95	Deg
Puntos adecuados	Initializer.cc	118	50	60	-
Error de reproyección	Initializer.cc	494	4.0	1.0	píxel
Paralaje (kp)	Initializer.cc	857	0.3624	0.8103	Deg
Settings					
Niveles de escala	Settings*.yaml	-	8	6	-
Factor entre niveles de escala	Settings*.yaml	-	1.2	1.1	-
Umbral de oFAST	Settings*.yaml	-	20	24	-
KeyPoints deseados	Settings*.yaml	-	1000	1200	-

Tabla 2.1: Sintonía realizada en ORB-SLAM2: descripción del parámetro modificado, archivo en el que se encuentra, línea en la que se ha alterado el código de acuerdo com los archivos del software original, valor original, valor modificado, unidades de la variable modificada. (**fe: (distancia al punto) / (factor de escala) / (factor de escala de KeyFrame asociado)*).

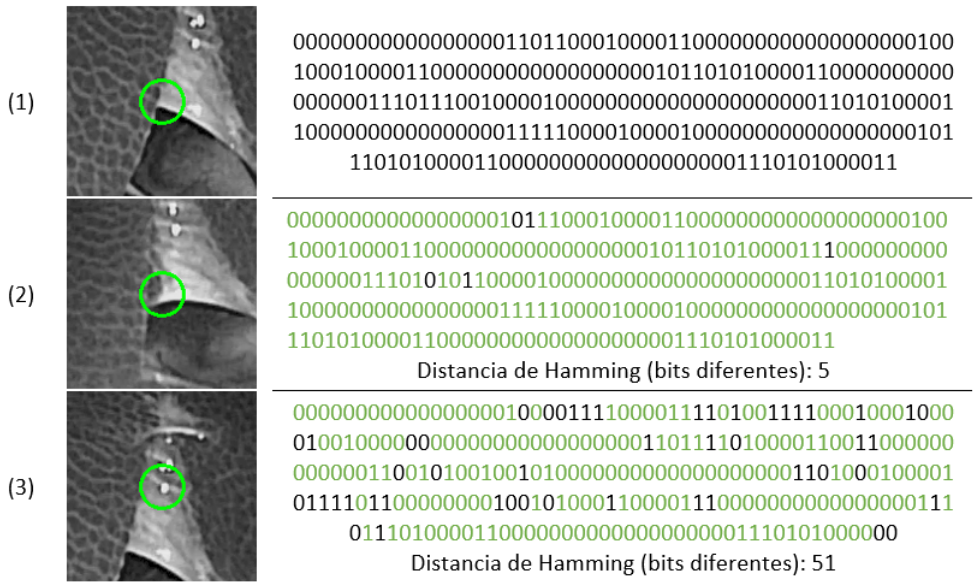


Figura 2.1: Similitud entre descriptores ORB medida mediante la distancia de Hamming. En verde se señalan los bits concordantes en el KeyPoint (1), que presenta similitud con el KeyPoint (2), pero no con el (3).

previamente mapeado. Para la verificación del cierre de bucle el sistema compara el 3D de ambos mapas, el original y el que está siendo generado. Debido a que el endoscopio recorre un entorno deformable, en el que el tejido cambia con el tiempo, ambos mapas pueden no ser lo bastante similares, impidiendo completar el proceso.

La relocalización utiliza la misma lógica y funciones que el cierre de bucle. La diferencia radica en que este último funciona en paralelo con Tracking y Mapping, mientras que la relocalización entra en acción cuando se ha perdido la posición relativa de la cámara respecto del mapa, estando detenidos ambos procesos. Esta pérdida de posición puede darse por oclusiones, movimientos bruscos del endoscopio, extracción de este para su limpieza y otras ocurrencias inherentes a la operación. En ese momento es necesario relocalizar la cámara en un entorno que también puede haber sufrido una deformación no rígida respecto del momento en que se construyó el mapa guardado.

En la endoscopia médica, las imágenes captadas muestran una cavidad generalmente observada siempre desde el mismo extremo, correspondiente al orificio de acceso del endoscopio. La cámara se mueve por la escena pero no completa bucles, por lo que la implementación del proyecto se centra en la relocalización, no obstante, el mismo procedimiento implementado aquí pudiera ser extrapolado a la detección de cierres de bucle.

2.3. Relocalización

El proceso en la relocalización es el siguiente:

1. Empleando el software de Bolsa de Palabras DBoW2 [16] se buscan KeyFrames

candidatos a mostrar la misma escena que capta la cámara, para ello se buscan palabras comunes entre la escena vista y estos KeyFrames.

2. Valiéndose del índice inverso de la bolsa de palabras, se buscan correspondencias de ORB entre el frame actual y estos KeyFrames.

3. En aquellos donde se encuentran suficientes correspondencias, se llevan a cabo iteraciones RANSAC con un algoritmo PnP (Perspective-n-Projection) [19] para obtener una primera aproximación de la pose de cámara.

4. La pose calculada se emplea para proyectar el mapa guardado sobre la imagen y llevar a cabo una búsqueda guiada de correspondencias. Si se obtienen suficientes, se lleva a cabo una optimización no lineal de la pose de cámara.

5. Si esta optimización obtiene suficientes inliers, la relocalización habrá sido satisfactoria y se reanudará el funcionamiento normal desde esa posición. En caso contrario, el proceso actúa con cada nuevo frame captado por la cámara hasta que se logra una pose válida.

La dificultad de este proceso en un entorno médico radica en que, tanto para calcular la pose de la cámara con PnP (P4P) como durante la optimización no lineal, se intenta emparejar puntos ORB detectados en la imagen contra puntos de un mapa creados tiempo atrás. La deformación 3D que han sufrido los puntos guardados causa que la estructura de la escena vista por la cámara no concuerde con la que representa el mapa existente.

Los procesos de relocalización y cierre de bucle son de funcionamiento similar, en el trabajo nos centraremos en mejorar el primero, que puede actuar sobre cualquier entorno ya mapeado, no siendo necesario recorrer circuitos cerrados.

Capítulo 3

Cálculo de pose de cámara en entorno no rígido

Para permitir la relocalización en entornos deformables se modifica el algoritmo para la búsqueda de KeyFrames contra los que relocalizar, llegando a aumentar el número de oportunidades de relocalización. Posteriormente se modifica el modo de validación y refinado de estos candidatos, alterando los procesos de estimación inicial de la pose de cámara y la posterior optimización no lineal de esta.

3.1. Búsqueda de KeyFrames

Cuando se interrumpe el Tracking y no se conoce la pose de la cámara respecto del mapa, esta intenta recalcularse con cada nueva imagen que capta el endoscopio. El método consiste en encontrar un KeyFrame con información similar a la de la imagen que permita obtener una pose inicial a optimizar. El proceso es el siguiente:

1. Para llevar a cabo una búsqueda rápida entre todos los KeyFrames se emplea un algoritmo de Bolsa de Palabras (*Bag of Words, BoW*) que agrupa puntos característicos formando *palabras* y mantiene un índice que conoce en qué KeyFrames está presente cada una. De este modo, calculando las palabras de la imagen actual, puede conocerse rápidamente en qué KeyFrames han aparecido. Todos los KeyFrames encontrados pudieran representar la escena mostrada en el momento de la relocalización, para acotar su número, se buscan emparejamientos entre los puntos de interés que forman las palabras.

2. El algoritmo presente en ORB-SLAM2 requiere al menos 15 correspondencias de puntos de interés, con una distancia de Hamming menor de 50, para considerar que un KeyFrame tiene similitud suficiente con la escena actual.

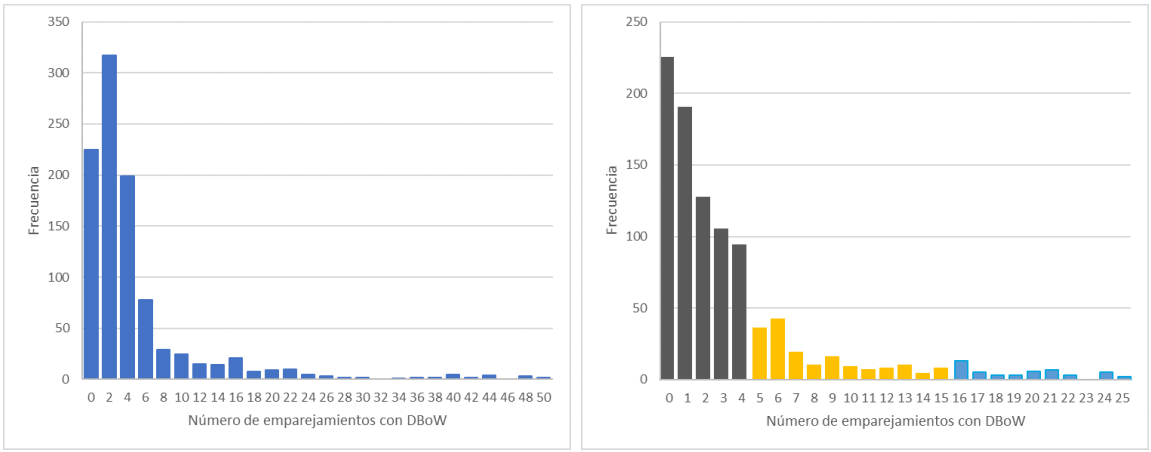
   Se ha analizado el número de KeyFrames en los que se supera este umbral, el histograma de la Figura 3.1a muestra la frecuencia contra la que se obtiene un cierto número de emparejamientos, por ejemplo, solo se obtienen 15 emparejamientos en 20 de cada 1000 KeyFrames. En conjunto se observa que solo en un

10 % de las ejecuciones se supera el umbral establecido, bins azules en la Figura 3.1b).

La deformación de la escena causa que el número de emparejamientos por KeyFrame baje, pues los puntos se han desplazado y deformado, aumentando su distancia de Hamming respecto de los originales.

3. Analizando la secuencia, se observa que la mayor parte de los KeyFrames descartados podrían representar la escena vista. Ya que el objetivo es emplear todos los posibles para aumentar las posibilidades de relocalizar se reduce el umbral a 4 correspondencias, que es el mínimo necesario por el algoritmo de estimación de pose del paso posterior, aumentando el número de KeyFrames utilizables hasta el 35 % del total, añadiendo los bins amarillos de la Figura 3.1b. Además, se aumenta la distancia de Hamming de 50 a 60 bits, lo que acepta aún más correspondencias.

Todo lo anterior permite un mayor aprovechamiento de la escena, aumentando las posibilidades de relocalizar al intentarlo contra un mayor número de KeyFrames.



(a) (b)

Figura 3.1: Histogramas con frecuencia de emparejamientos obtenidos para 1000 KeyFrames candidatos a relocalizar. (a) 0-50 correspondencias. (b) 0-25 correspondencias: menos de 4, insuficientes para el algoritmo P4P (■), 4-15, KeyFrames ganados al relajar el umbral (■), KeyFrames con el umbral original (■).

3.2. PnP para estimación de pose

Con los puntos correspondientes entre los KeyFrames de la Bolsa de Palabras, ORB-SLAM2 lleva a cabo un cálculo iterativo con RANSAC sobre un algoritmo de *Perspective-n-Projection* (PnP). Este método emplea las correspondencias entre puntos 3D y de imagen para estimar una pose y necesita al menos 4 puntos. En escenas deformables, estos puntos han podido sufrir una transformación que impide emparejarlos en las poses que se calculen dificultando encontrar una solución, con lo que de

nuevo se modificarán los umbrales para reducir la cantidad de puntos descartados. La Tabla 3.1 muestra una comparativa entre los valores originales y los modificados y, a continuación, se describe el proceso resultado:

1. Se comienza seleccionando aleatoriamente 4 puntos del subconjunto emparejado con la Bolsa de Palabras. Con ellos se lleva a cabo una estimación inicial de la pose y esta intenta validarse con una búsqueda guiada de correspondencias para todos los puntos obtenidos con la Bolsa de Palabras. En función de la cantidad obtenida, el proceso sigue diferentes caminos:

   a) Si se alcanzan 9 correspondencias se lleva a cabo una segunda búsqueda guiada, se refina la pose y se lleva a cabo una última búsqueda guiada. Si vuelven a alcanzarse 9 correspondencias la pose se considera correcta.

   b) Obteniendo entre 3 y 9 correspondencias, estas se almacenan junto con la pose estimada. Si tras todas las iteraciones no se alcanza un resultado satisfactorio, estas poses se emplean para calcular el mejor posible con una búsqueda guiada de emparejamientos que proyecta todo el mapa sobre el frame. Esto conlleva un alto coste computacional, por lo que solo se aplica con los 3 candidatos con más correspondencias. Si la proyección resulta en al menos 30 correspondencias la pose se valida, con un resultado menor esta se guarda y prosigue el análisis con otro subconjunto de puntos. Si al completar el proceso no se alcanza una pose válida, aquella con más correspondencias será el resultado.

2. Tras obtener la pose resultado, esta se refina con una última búsqueda guiada que en el algoritmo modificado se lleva a cabo mediante proyección del mapa completo, a fin de obtener el mayor número posible de emparejamientos que alimentar a la optimización no lineal del paso posterior. En todas las comparaciones se demanda una distancia de Hamming menor de 60 bits, siendo la original 45. Esto permite contar con la no rigidez, aceptando más puntos de interés.

La Figura 3.2 muestra histogramas correspondientes al número de poses validadas en función del número de correspondencias de partida. El algoritmo original (bins azules) actúa con éxito cuando el número de correspondencias de la Bolsa de Palabras es mayor de 12, mientras que el modificado (bins amarillos) resuelve entre 4 y 12.

Con el nuevo algoritmo el número de correspondencias aumenta hasta 30 veces para valores iniciales mínimos. La Figura 3.2b muestra el factor de incremento en función de las correspondencias iniciales. El mayor inconveniente del nuevo método es un mayor tiempo de ejecución, que depende del número de proyecciones a realizar, siendo de hasta 1 segundo al proyectar el mapa 3 veces.

Experimentalmente se ha obtenido que para secuencias con deformación, el algoritmo original logra relocalizar la cámara en un 27 % de las ejecuciones, mientras que

	Original	Modificado
Correspondencias iniciales BoW	15	4
Correspondencias iniciales RANSAC	12	9
Correspondencias 2da búsqueda BoW	-	9
Correspondencias proyección mapa	-	30
Distancia de Hamming	50	60

Tabla 3.1: Comparación de umbrales aplicados en el algoritmo de PnP para el proceso original y el modificado. Para aumentar las hipótesis evaluadas, se permite comenzar con menor número de correspondencias iniciales y se añade una búsqueda por proyección del mapa completo, aumentando los puntos analizados al no estar limitados por los presentes en la Bolsa de Palabras.

el modificado logra éxito en el 34 %, no completando el proceso en el resto de ocasiones. Las modificaciones aumentan la capacidad de relocalizar en un 26 % respecto a la original, pudiendo procesar escenas de alta dificultad que antes eran descartadas.

3.3. Optimización no lineal y no rígida con Bundle Adjustment Local

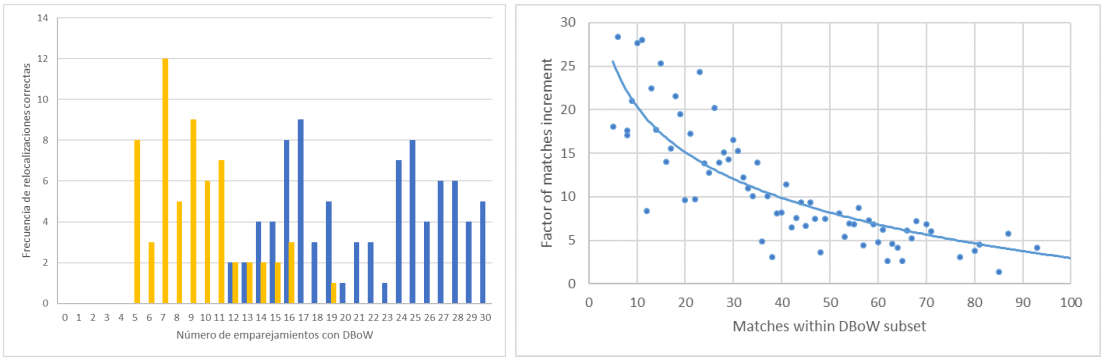
Esta es la última etapa del proceso de relocalización, en la que se ajusta y verifica la pose mediante una optimización no lineal de los resultados anteriores. El objetivo es proporcionar la mejor pose posible de cámara y caracterizar la deformación sufrida por el entorno para retomar el Tracking al completar el proceso. El diagrama de flujo de la Figura 3.3 resume los pasos seguidos por el algoritmo.

Cada secuencia de optimización no lineal sigue el mismo procedimiento:

1. Se parte de una pose de cámara e inliers.
2. Se configuran nodos y arcos que conforman el grafo a optimizar.
3. Se realizan 4 optimizaciones de 10 iteraciones. Solo los inliers se emplean en optimizaciones sucesivas y en cada optimización se reduce el error permitido.

Para procesar la deformación se modifica la definición de nodos y arcos del grafo. Puesto que la posición relativa entre puntos del mapa varía, ahora se permite su movimiento, el cual se acota añadiendo al grafo los KeyFrames en los que estos se observan, obligando a mantener mínimo el error de reproyección también en ellos. La Tabla 3.2 y la Figura A.1 muestran las diferencias entre los métodos.

Consolidando, se ha reemplazado la optimización de pose de cámara del algoritmo original por un Bundle Adjustment local, que ajusta conjuntamente pose de cámara y posición 3D de los puntos. El histograma de la Figura 3.5 muestra la frecuencia con que se obtiene un determinado número de inliers para ambos métodos. Las modificaciones (bins amarillos) resultan en un número de inliers superior al que proporcionaba el algoritmo original (bins azules), aumentando la tasa de éxito del 59 % al 85 %.



(a) (b)

Figura 3.2: Aplicación con éxito de PnP en 1000 KeyFrames candidatos a relocalizar. (a) Histograma 0-30 correspondencias logradas con DBoW. Con algoritmo original (■), con algoritmo modificado (■). (b) Factor en que se ven incrementados los emparejamientos resultado del PnP representado en función del número inicial de emparejamientos del BoW.

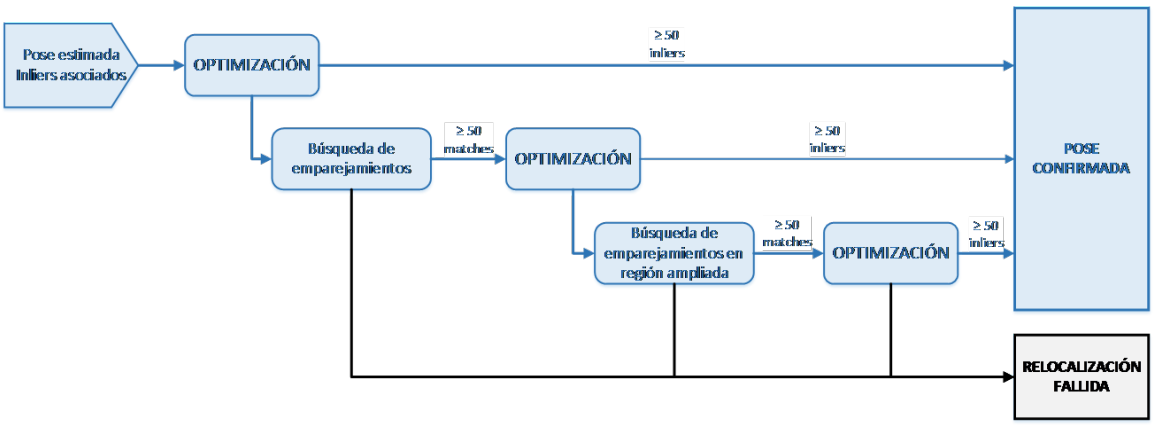


Figura 3.3: Etapas de optimización no lineal para ajustar y confirmar o rechazar la pose de cámara propuesta. En la primera iteración, la pose obtenida por PnP se optimiza y queda confirmada si permite obtener 50 inliers. Con menos de 50, se realiza una búsqueda guiada de emparejamientos proyectando el mapa sobre el frame. Si se obtienen 50 correspondencias se vuelve a optimizar, y con más de 50 inliers la pose se acepta. Con menos de 30 la pose se desecha. Entre 30 y 50 se vuelve a proyectar el mapa con mayores regiones de búsqueda. Si se obtienen 50 correspondencias, se optimiza la pose y si resulta en 50 inliers la pose se verifica, en caso contrario se descarta y el proceso de relocalización volverá a comenzar con el próximo frame captado.

		Entorno rígido	Entorno deformable
Nodo	Cámara	Móvil	Móvil
	MapPoints	Fijos	Móviles
	KeyFrames	-	Fijos
Arcos	MapPoint-Cámara	Si	Si
	MapPoint-KeyFrame	No	Si

Tabla 3.2: Cambios en la construcción del grafo para el algoritmo de Levenberg-Marquardt en función de la rigidez.

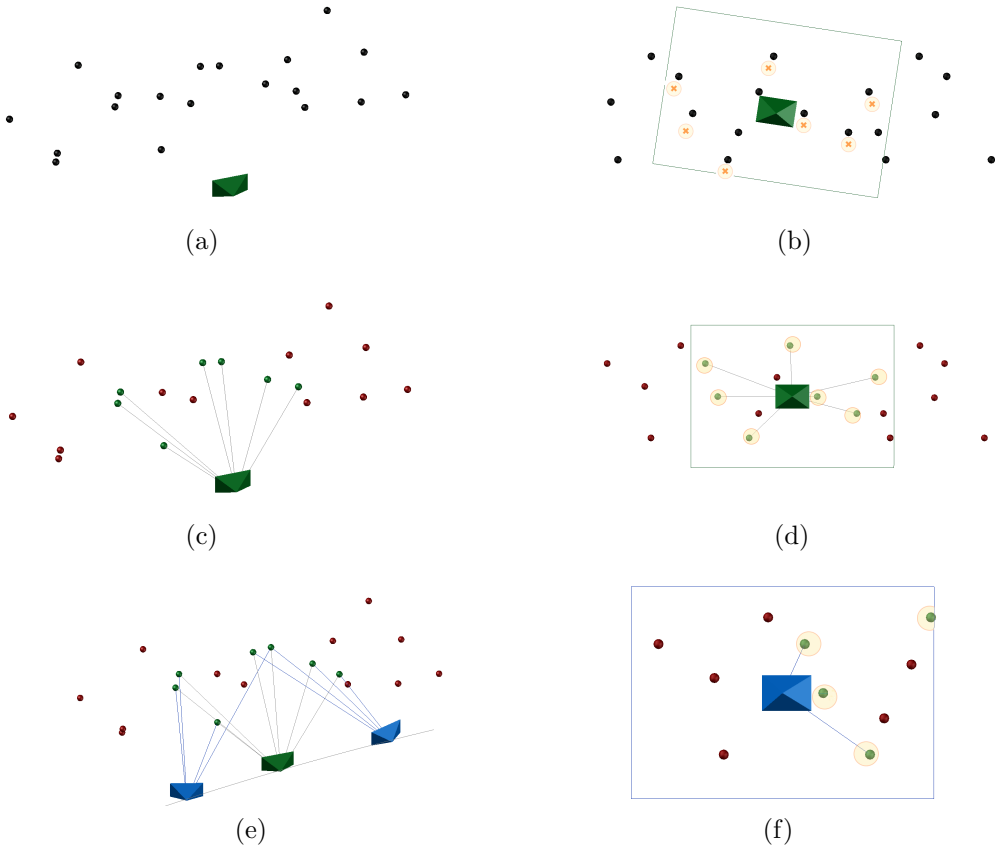


Figura 3.4: Grafos empleados en la optimización. (a) Situación de partida en la que una cámara ▼ observa una nube de puntos • fijos. (b) La posición de la cámara se ajusta hasta que los puntos 3D se proyectan sobre sus correspondientes en la imagen vista ×. (c) Para algunos puntos 3D se encuentra correspondencia en la imagen •. (d) Esto sucede si al proyectar el punto 3D sobre el frame, sus coordenadas 2D se encuentran en las proximidades del punto en la imagen. (e) En la optimización no rígida permitimos el movimiento de cámara y puntos 3D. Para restringir el movimiento se añaden KeyFrames ▼ que observan los puntos. (f) La proyección ha de encontrarse en las regiones de búsqueda para el punto en los KeyFrames, además de en el Frame.

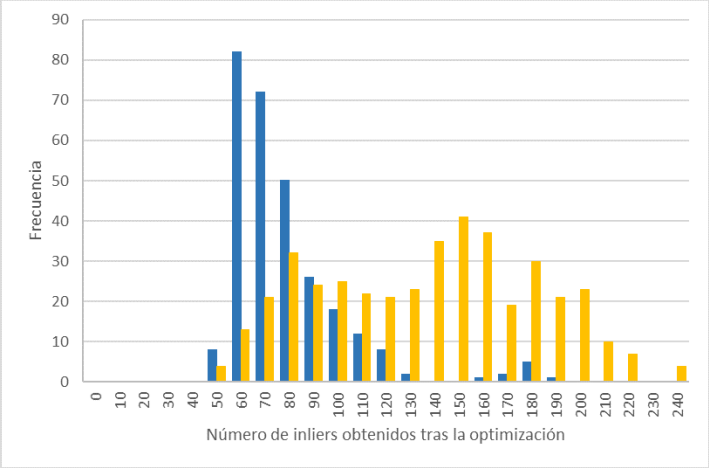


Figura 3.5: Comparativa entre resultados del algoritmo de optimización no lineal original (■) y el modificado (■).

Para controlar el movimiento de los puntos y mantener un comportamiento físicamente realista de los tejidos se modifica la función de coste de la optimización, pasando a minimizar conjuntamente error de reproyección y energía de deformación elástica asociada al movimiento de estos. Para obtener la energía, se integra en el código un análisis por elementos finitos de la escena. Las funciones programadas se encuentran disponibles en GitHub [20] y su descripción en el Anexo A.

Mallado de la nube de puntos del mapa

1. Se parte de un conjunto de puntos del mapa emparejados como el representado por la Figura 3.6a y se implementa una discretización 3D. Esta puede generarse con elementos C3D8 (hexaedros) o elementos C3D6 (prisma triangular).

2. Se le aplica una aproximación por Moving-Least-Squares (MLS), que suaviza la nube de puntos aproximando la superficie a un polinomio y elimina espurios.

3. Una triangulación de Delaunay mediante proyección proporciona una malla plana de elementos triangulares como la de la Figura 3.6b. Los parámetros de cálculo se obtienen dinámicamente en función de características de la nube empleada tales como posiciones relativas de los puntos, curvatura de la superficie, densidad, etc.

    a) Si se ha escogido elemento C3D6, esta malla se replica a una distancia preestablecida, que hará de altura para el elemento. Ambas mallas se conectan para obtener los prismas triangulares como se muestra en las Figuras 3.6c y 3.6d.

    b) Si se trabaja con elementos C3D8, los triángulos se convierten en cuadriláteros, para ello, se añaden nodos adicionales en el punto medio de cada segmento que conforma los triángulos y en el orto centro. Puesto que la conectividad de los triángulos de partida es conocida se evita añadir nodos redundantes,

obteniendo las coordenadas de cada nuevo punto en función de los nodos próximos, como se representa en la Figura 3.6e. Tras la transformación a cuadriláteros, se genera una malla análoga a una distancia pre-establecida y se conectan ambas capas para obtener los hexaedros.

Resolución por elementos finitos de la malla

1. Se definen parámetros de Lamé para el tejido en E=3.5KPa y nu=0.4999, guiándonos por lo presentado en [21] para el hígado. El espesor elemental se calcula experimentalmente para obtener una profundidad comparable al tamaño de los segmentos del elemento.

2. Con la malla definida se construye una matriz de rigidez $\boldsymbol{K}$ que define su comportamiento. Se han definido cálculos de matriz de rigidez elemental $\boldsymbol{Ke}$ para elementos C3D8 y C3D6, cuya forma en coordenadas naturales se muestra en la Figura 3.7. Puesto que la malla se compone de elementos no regulares, la matriz de rigidez elemental es calculada individualmente para cada uno de los componentes del mallado durante el propio proceso de ensamblaje de la matriz de rigidez.

3. Se imponen condiciones de frontera de Dirichlet a la matriz de rigidez, estableciendo un encastre en la cara inferior.

4. Antes de iniciar la optimización no lineal se registra la posición inicial de los nodos que participarán en un vector $\boldsymbol{u0}$.

5. Tras cada iteración se registra la nueva posición $\boldsymbol{uf}$ de los puntos presentes en la cara vista de la malla, manteniendo constante la posición de los puntos de la base, en condiciones de encastre. Con las posiciones finales se calcula el desplazamiento de los puntos respecto de la posición de partida y se imponen condiciones de Dirichlet al vector de desplazamientos correspondientes a las aplicadas sobre la matriz de rigidez.

6. Con desplazamientos y matriz de rigidez definidos puede obtenerse la fuerza sobre los nodos, que permite calcular la energía de deformación inducida en el modelo de acuerdo con la Ecuación 3.3.

$$E = a \cdot F = a' \cdot K \cdot a \tag{3.1}$$

Función de coste

La nueva función de coste a minimizar se muestra en la Ecuación 3.2 y consta de dos componentes:

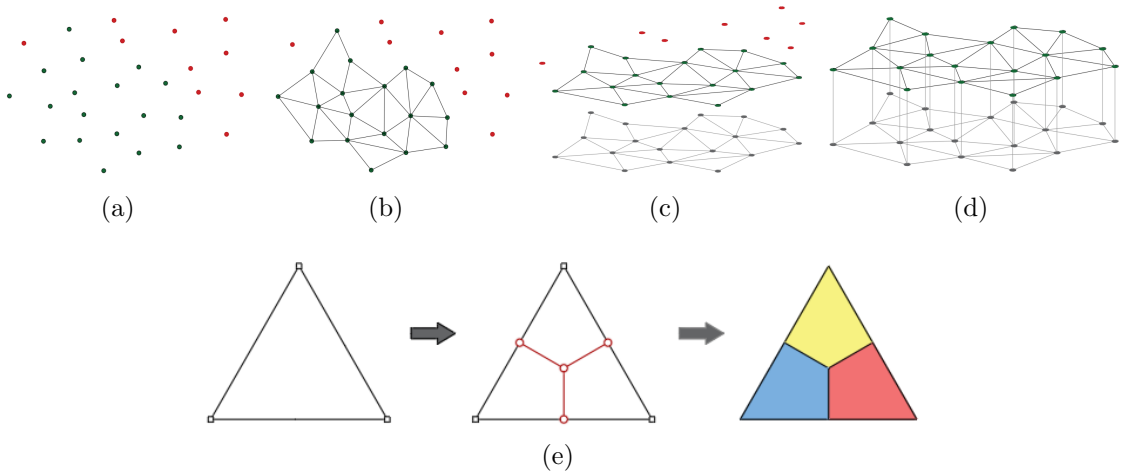


Figura 3.6: Etapas del proceso de mallado para prisma triangular, en (a) se muestra una nube de puntos donde ● representa puntos emparejados y ● puntos sin correspondencia en la imagen, en (b) se muestra la triangulación de la nube, en (c) la malla se replica, y en (d) se unen ambas capas para formar los prismas triangulares. En (e) se representa la transformación de cada triángulo en 3 cuadriláteros.

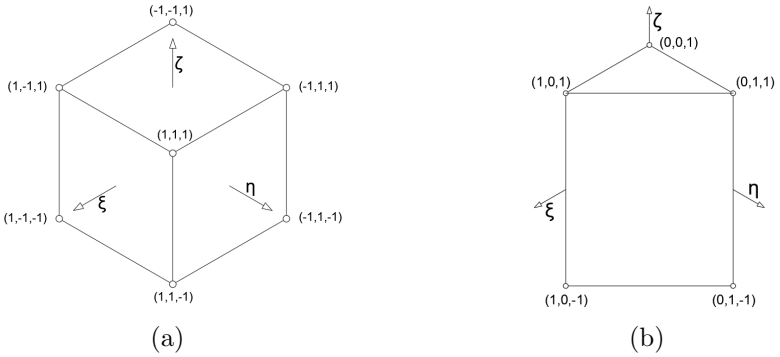


Figura 3.7: Elementos de referencia tridimensionales con aproximación lineal. (a) Hexaedro C3D8. (b) Prisma triangular C3D6.

– Error de reproyección: obtenido a partir de los emparejamientos resultado de proyectar los puntos del mapa sobre Frame y KeyFrames. El valor de error se obtiene según la Ecuación 3.3, que resuelve la diferencia entre la posición u del punto i en la cámara (Frame o KeyFrame) j, y la proyección del punto 3D correspondiente X_i mediante la pose de cámara asociada a la cámara P^j.

  - e_r^F Al proyectar sobre el Frame, tanto la posición 3D de los puntos X_i como los parámetros extrínsecos del Frame θ_{ext}^j son optimizados.
  - e_r^{KFj} Al proyectar sobre los KeyFrames observadores, solo la posición 3D de los puntos X_i varía.

– Energía de deformación mostrada en la Ecuación 3.4 y causada por el movimiento impuesto a los puntos X_i respecto de su posición original. El desplazamiento se

representa en el vector a y obtiene la energía al operarlo con la matriz de rigidez K calculada en el proceso. Esta energía se normaliza dividiéndola por el número de puntos analizados n.

A mayor desplazamiento de los puntos, más se reduce el coste asociado al error de reproyección, pero aumenta el debido a la deformación. El algoritmo busca una posición óptima que minimice el resultado de la Ecuación 3.2 que combina el error de reproyección de los puntos y la energía de deformación. Para ello, se varían posición de puntos del mapa y pose de cámara. Se aplica un peso a cada uno de los componentes de la ecuación para hacer comparables ambos factores, este se ha calculado empíricamente y es diferente para la iteración inicial y las sucesivas.

$$e = \underset{\mathbf{X}_i, \theta^F_{ext}}{\arg\min} \left(w_r \cdot \left(e^F_r + e^{KF}_r \right) + w_{fea} \cdot e_{fea} \right) \tag{3.2}$$

$$e^j_r = \sum_{i,j} \left(u^j_i - P^j \left(X_i, \theta^j_{ext}, \theta^j_{int} \right) \right)^2 \tag{3.3}$$

$$e_{fea} = \frac{a' \cdot K \cdot a}{n} \tag{3.4}$$

Tracking tras la relocalización

Cuando el proceso de relocalización se completa con éxito, la pose de cámara obtenida se emplea para continuar el funcionamiento normal y la última posición deformada de los puntos que han tomado parte en la optimización se inscribe en el mapa existente.

Esto causa que la parte del mapa que ha tomado parte en la optimización se encuentre deformada, mientras que el resto de puntos conservan sus posiciones originales (no deformadas). Cuando se reinicia el Tracking, este lo hace siempre desde la configuración deformada y, en función de la escena y su comportamiento, pueden darse dos situaciones:

- Si la deformación ha sido pequeña o si el movimiento del tejido es propicio, el programa partirá del mapa deformado, pero también hará uso del anterior: Los algoritmos de gestión del mapa y *Bundle Adjustment* globales se encargarán de controlar la posición de los puntos no deformados.

- Si la deformación es grande, el nuevo proceso de Tracking que ha partido de una posición deformada puede no encontrar correspondencias contra el mapa original, con lo que partiendo de la posición deformada se generará un segundo mapa superpuesto al original.

Capítulo 4

Validación experimental

Los algoritmos desarrollados se evalúan en secuencias de laparoscopia abdominal in vivo del dataset público del Centro de Laparoscopias Hamlyn [22, 23], así como con secuencias privadas de mayor deformación propiedad del Instituto de Investigación del Cáncer en el Aparato Digestivo (IRCAD).

Cada secuencia se ha segmentado en varias de menor duración, empleando uno de los cortes para construir el mapa a evaluar y el resto para pruebas de relocalización, forzando la pérdida del Tracking entre las secuencias de mapeo y evaluación. Con el objetivo de evaluar todos los frames posibles de cada secuencia, cuando se alcanzan 5 frames tras la relocalización se fuerza la pérdida del Tracking y se reinicia el proceso. Para todas las secuencias se compara el rendimiento del software original (referenciado como ORB-SLAM2 en los apartados siguientes) respecto del modificado (referenciado como ORB-SLAM2-E (***E**ndoscopia*)). Se presentan resultados para los dos elementos implementados, C3D6 y C3D8.

Se registran verdaderos positivos, negativos y falsos positivos. Con ellos, se construyen indicadores para *Precision* y *Recall*, y todos los contadores se mantienen a 0 hasta que se obtiene la primera relocalización correcta:

- True Positives *TP*: una relocalización se considera TP si tras calcular una pose el sistema mantiene el Tracking durante los 5 frames posteriores.

- False Positives *FP*: se considera FP si tras calcular la pose el Tracking se pierde antes de 5 frames.

- False Negatives *FN*: las secuencias se preparan de modo que la escena vista por todos los frames empleados durante la prueba de relocalización ya haya sido explorada con la secuencia de mapeo, esto permite considerar como FN todos los frames en los que el algoritmo de relocalización no actúe.

- Precision: mide el total de relocalizaciones verdaderamente correctas respecto del total que el sistema considera acertado, expresado en la Ecuación 4.1.

- Recall: mide el total de relocalizaciones que el software considera correctas entre

el total en el que debería haber actuado con éxito, de acuerdo con la Ecuación 4.2.

$$Precision = \frac{TP}{TP + FP} \qquad (4.1) \qquad\qquad Recall = \frac{TP}{TP + FN} \qquad (4.2)$$

4.1. Evaluación de funcionamiento

Los resultados de todas las secuencias se presentan en la Tabla 4.1, una comparativa gráfica en la Figura 4.1 y el análisis en las líneas siguientes. Las secuencias están numeradas de 1 a 5 y este mismo índice se usa para hacer referencia a ellas en las líneas siguientes.

También se presentan capturas del funcionamiento con cada secuencia en la Figura 4.2, para todas ellas aparece una primera imagen con lo visto por la cámara en el momento de la relocalización, en la misma pueden verse diferentes puntos:

- Puntos verdes (•) que coinciden con vértices de la malla: son aquellos puntos del mapa que han tomado parte en el proceso de la relocalización. Ciertos vértices no tienen un punto asociado lo que es debido a que el proceso de Tracking posterior a la relocalización no ha logrado emparejar el punto en ese frame, pero pudiera hacerlo en el futuro.

- Puntos verdes (•) que no forman parte de la malla: son aquellos para los que el proceso de Tracking reactivado tras relocalizar ha logrado encontrar una correspondencia en el mapa.

- Puntos azules (•): tras completar la relocalización con éxito, se reactiva el proceso de Mapeo que genera estos nuevos puntos que son introducidos en el mapa.

Se presentan también una copia de la imagen con la malla superpuesta y la misma malla en el mapa de puntos. En la Figura 4.3 se representa la deformación distribuida a lo largo de la malla, pintando en tonos azules aquellos elementos con menor deformación y en tonos rojizos aquellos donde es mayor. Para mejor representación en todas las imágenes solo se muestra la cara superior de los elementos.

1. Hamlyn con baja deformación: se presenta una vista cercana del hígado, durante la secuencia el endoscopio sufre movimientos ligeros y el tejido se deforma debido a la respiración. Se genera un mapa inicial recorriendo el órgano de derecha a izquierda. Al llegar al extremo, se fuerza la pérdida del Tracking y se inicia la evaluación del proceso de relocalización.

   Se obtiene una mejora considerable en todos los indicadores empleando el software modificado. La precisión aumenta un 30 % al usar prisma triangular y un 22 %

con hexaedros. El Recall aumenta en un 109 % y un 51 % para prisma triangular y hexaedro respectivamente.

Para bajas deformaciones, como aquellas causadas por la respiración, el algoritmo implementado permite una mayor efectividad en la relocalización, el gran incremento del Recall implica un mayor aprovechamiento de la escena, y el aumento de la Precisión indica que no solo se mantiene, sino que se mejora la calidad de las relocalizaciones aún usando los frames que antes eran descartados.

Se ha publicado un vídeo resumen de la escena en [24].

2. Hamlyn con alta deformación: el endoscopio, que se mantiene fijo, muestra un corazón palpitante. Durante la mayor parte de la secuencia solo aparece el tejido y, en los segundos finales, se interactúa con el órgano. Se emplea una secuencia de 10 segundos para construir el mapa, el propio movimiento del órgano sirve para inicializar el sistema. Posteriormente se emplea una secuencia de 38 segundos para pruebas de relocalización. Durante toda la secuencia el órgano sufre deformaciones constantes causadas por el latido y la respiración.

   El ORB-SLAM2 original alcanza una precisión del 21.8 % y un recall del 3.9 %, lo que sugiere que solo actúa en los momentos en que la deformación del órgano es similar a la registrada en la exploración, esto unido a la baja precisión, hace el sistema muy poco fiable en esta secuencia. El algoritmo modificado aumenta la precisión hasta el 57,7 % y 40,1 % para prisma triangular y hexaedro respectivamente, a la par que aumenta el Recall hasta 52,9 % y 34,6 %.

   A pesar de la complejidad de la escena, con altas deformaciones, fluidos, reflejos y oclusiones por herramienta, el nuevo algoritmo permite la relocalización eficaz varias veces por segundo, con una precisión más del doble de la original y un Recall 13 veces superior.

   Un vídeo de la escena se encuentra disponible en [25].

3. IRCAD con baja deformación: en la secuencia se observan pared abdominal, porciones de órganos y gran cantidad de tejido graso. El endoscopio se mueve lentamente recorriendo la escena, que sufre deformación ligera debido a la respiración.

   El algoritmo modificado causa un descenso de la precisión del 3 % para prisma triangular y 9 % para hexaedro, por contraposición el Recall aumenta un 22 % y un 16 % respectivamente.

   Al tratarse de una escena de alta calidad y baja deformación, el algoritmo original tiene un alto rendimiento, a pesar de lo cual el modificado aporta una mejora en cuanto al Recall, de nuevo aumentando el aprovechamiento de la escena.

   Parte de la secuencia se ha subido en [26].

4. IRCAD con deformación media: se muestra una exploración en la que aparecen bazo, hígado y pared abdominal, encontrando deformación importante en los dos últimos. Se emplea el bazo, de alta rigidez, para la inicialización, y se hace un barrido de 20 segundos que explora el hígado. Se evalúa la relocalización con otra secuencia de 20 segundos que muestra bazo e hígado.

   En este caso, el sistema original tiene un buen rendimiento, se observa una ligera pérdida de precisión del 2,9 % para el prisma triangular y una mayor del 9,2 % para el hexaedro. Por contra el Recall aumenta un 15 % y un 11 % para prisma y hexaedro respectivamente.

   Aún con la leve pérdida de precisión, se logra un aumento del Recall, que favorece el aprovechamiento de la escena y reduce el tiempo necesario para relocalizar.

   Porciones de esta escena pueden verse en la primera mitad del vídeo en [27].

5. IRCAD con deformación elevada: se emplea una porción diferente de la secuencia anterior partiendo del mismo mapa inicial. Se tienen 50 segundos que muestran principalmente el hígado y la pared abdominal, ambos con deformación elevada.

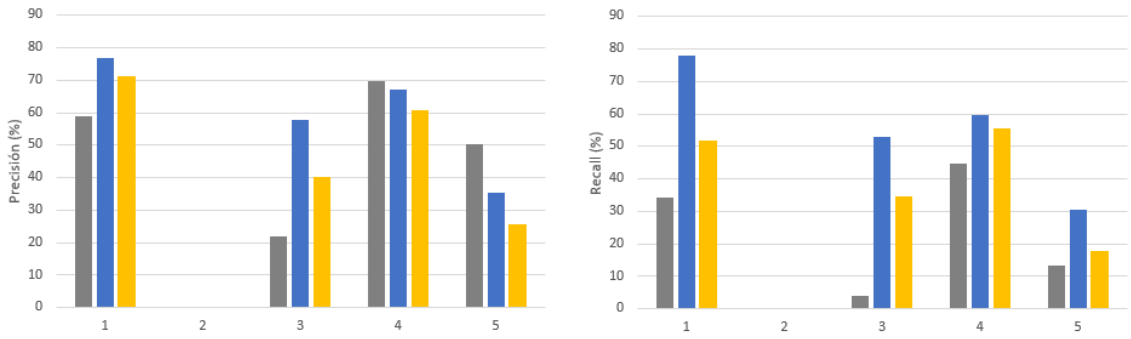
   Al igual que en el caso anterior, el sistema original mantiene una precisión mayor, si bien su Recall se mantiene en el 13 %. El algoritmo modificado causa una leve pérdida de precisión, pero a cambio aumenta el Recall, especialmente con el elemento prisma triangular en el que este es 2,3 veces superior.

   Nuevamente el mayor Recall permite un mejor aprovechamiento de la escena, a costa de la pérdida de precisión.

   En la segunda mitad de [27] puede verse la relocalización en esta secuencia. El mapa inicial se ha construido con la misma secuencia que en el punto anterior.

Id	Dif	Descripción	OS2		OS2E C3D6		OS2E C3D8	
			Prec.	Recall	Prec.	Recall	Prec.	Recall
1	2	Hamlyn baja 05	58.8	34.3	**76.7**	**78.0**	71.7	51.9
2	5	Hamlyn alta 03	21.8	3.9	**57.7**	**52.9**	40.1	34.6
3	1	IRCAD baja	**91.2**	46.0	88.5	**68.7**	82.1	62.2
4	3	IRCAD media	**69.9**	44.7	67.0	**59.5**	60.7	55.6
5	5	IRCAD alta	**50.4**	13.3	35.2	**30.7**	25.7	17.8

Tabla 4.1: Evaluación experimental. La columna *Dif* presenta un valor orientativo, entre 1 y 5, de la dificultad de la escena, a mayor deformación mayor el valor. El par de columnas ***OS2*** hace referencia al software original, El par ***OS2E C3D6*** al software modificado empleando elemento prisma triangular, y el ***OS2E C3D8*** al modificado empleando elemento hexaedro. Todos los valores se presentan como porcentaje.



(a) (b)

Figura 4.1: Comparativa de precisión (a) y Recall (b) para cada secuencia. El bin Gris ■ representa la puntuación del algoritmo original, el azul ■ representa el rendimiento del software modificado empleando elemento prisma triangular y el amarillo ■ representa el rendimiento empleando elemento hexaedro.

4.2. Análisis ablativo

Para concluir la sección, se presenta un análisis identificando la mejora alcanzada individualmente gracias a cada una de las modificaciones realizadas: búsqueda de KeyFrames, cambios en el PnP, y Blundle Adjustment no rígido. La Tabla 4.2 muestra un promedio de KeyFrames e Inliers alcanzados en las diferentes etapas y para todos los intentos de relocalización de cada secuencia. Los valores aumentan en todas las secuencias:

- Número medio de KeyFrames candidatos obtenidos en cada intento de relocalización para umbral de 15 (original) y 4 (modificado) emparejamientos mínimos. El algoritmo modificado al menos duplica el valor del original en todas las ejecuciones lo que concuerda con el aumento general del Recall.

- Promedio de inliers obtenidos con el PnP base del programa frente al avanzado. Con el primero no se alcanza el umbral de 12 inliers en algunas secuencias mientras que el modificado permite obtener una estimación de pose en prácticamente la totalidad de ejecuciones. Además, se observa como en las secuencias con mayor deformación el resultado alcanzado es hasta 5 veces superior al del algoritmo original.

- Promedio de inliers añadidos antes de la optimización empleando la pose resuelta por el PnP con el algoritmo original frente al modificado. Aumenta en todas las ocasiones permitiendo siempre iniciar la optimización no lineal, algo no posible con el algoritmo original en las ejecuciones en que el número de inliers era menor que 50. Nuevamente la ventaja del algoritmo es especialmente importante en las escenas de máxima deformación.

- Media de inliers obtenidos mediante la optimización no lineal de partida y la no

rígida. La cantidad de inliers es siempre mayor y es de especial interés la mejora lograda en las secuencias 2 y 5, de alta deformación, en donde el número de inliers aumenta hasta 5 veces.

Id	Descripción	KFs cand.		Inliers PnP		Inliers Pre		Inliers Post	
		OS2	OS2E	OS2	OS2E	OS2	OS2E	OS2	OS2E
1	Hamlyn baja 05	0.47	**1.24**	24.02	**27.29**	41.76	**81.42**	94.84	**116.13**
2	Hamlyn alta 03	0.39	**1.58**	3.13	**17.38**	7.70	**73.26**	15.23	**77.06**
3	IRCAD baja	0.40	**1.04**	10.51	**16.90**	23.41	**126.09**	70.76	**141.91**
4	IRCAD media	0.36	**1.12**	8.46	**13.68**	15.01	**47.80**	41.82	**65.21**
5	IRCAD alta	0.32	**1.67**	3.88	**12.76**	12.14	**117.10**	27.95	**123.78**

Tabla 4.2: Análisis de mejora lograda, las columnas *OS2* y *OS2E* muestran respectivamente, valores para el algoritmo original y el modificado. ***KFs cand***: promedio de Keyframes candidatos para cada intento de relocalización. ***Inliers PnP***: promedio de Inliers obtenidos en el algoritmo de PnP. ***Inliers Pre***: promedio de puntos obtenidos con la pose del PnP antes de la optimización no lineal. ***Inliers Post***: promedio de inliers tras la optimización no lineal.

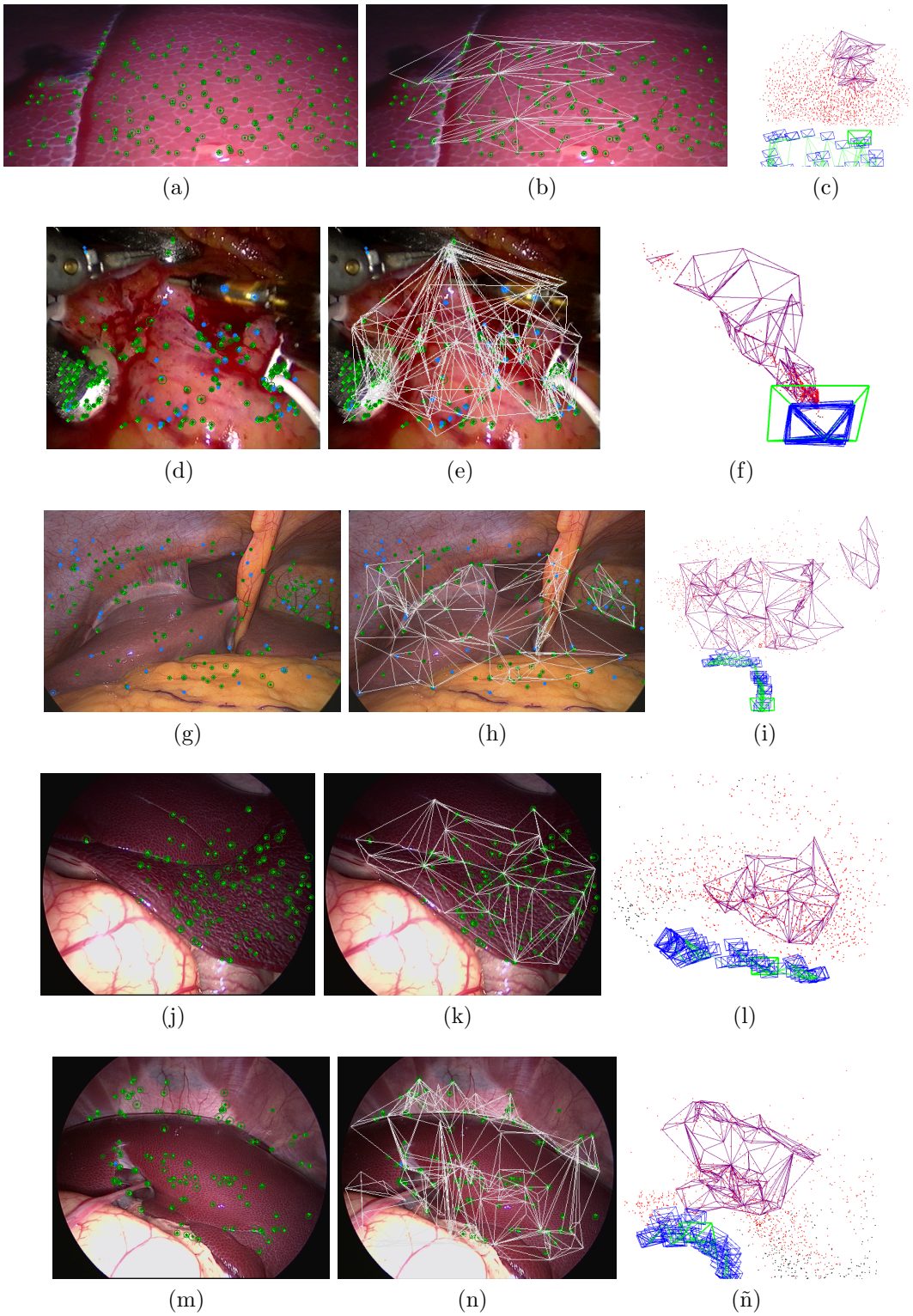


Figura 4.2: Imágenes de las secuencias empleadas en el análisis experimental. Para cada secuencia se presentan 3 imágenes que, de izquierda a derecha, representan el frame en que se relocaliza la cámara y que muestra los puntos que han intervenido en la relocalización (•) y los nuevos puntos calculados tras de completar el proceso (•), la misma imagen con la malla superpuesta y esa misma malla representada sobre el mapa de puntos. Secuencia 1 (a,b,c). Secuencia 2 (d,e,f). Secuencia 3 (g,h,i). Secuencia 4 (j,k,l). Secuencia 5 (m,n,ñ).

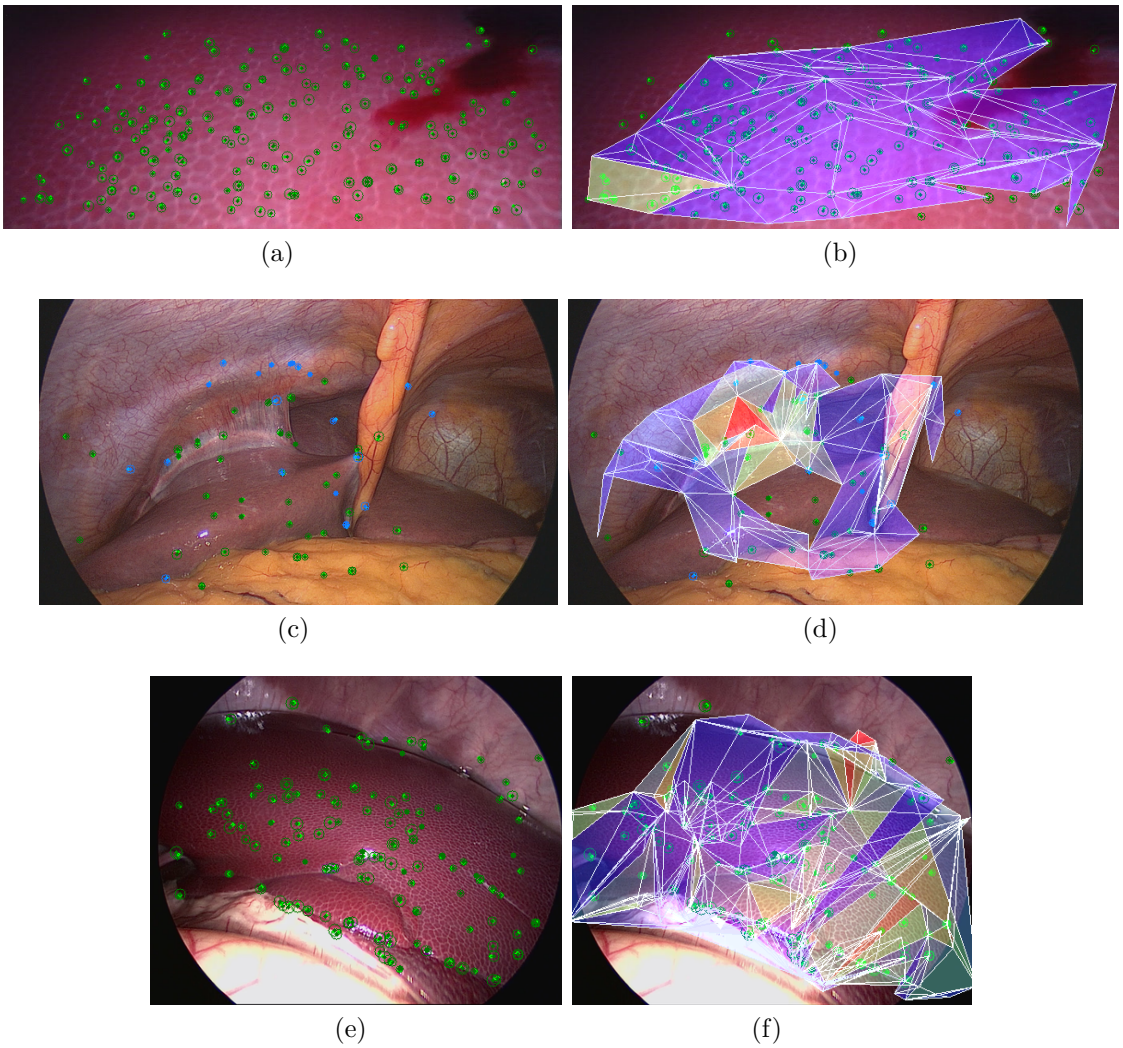


Figura 4.3: Magnitud de la deformación coloreada sobre la malla, rojo y azul corresponden a máxima y mínima deformación respectivamente (■■). Secuencia 1 (a,b), Secuencia 3 (c,d). Secuencia 5 (e,f).

Capítulo 5

Resultados

5.1. Conclusiones y discusión

La optimización implementada, minimizando conjuntamente la energía de deformación y el error de reproyección, dota al sistema de la capacidad de relocalizar la cámara en entornos con deformación, en los que anteriormente no era posible completar el proceso para la mayoría de ejecuciones. En todos los casos analizados, se tiene un incremento en el Recall alcanzado, es decir, en el número de posibilidades de relocalizar la cámara. La precisión generalmente aumenta, salvo en aquellas secuencias de máxima dificultad. No obstante, aún con una relocalización incorrecta, esta es descartada en frames posteriores y el alto Recall permite nuevos intentos de relocalización poco después. Analizando individualmente cada una de las etapas del proceso de relocalización modificado, se ha encontrado que el nuevo algoritmo proporciona una mejora especialmente grande a cada una de ellas y, en particular, en las secuencias de mayor deformación.

Se obtienen mejores resultados con elemento Prisma Triangular que con Hexaedro. Empleando el primero, la nueva implementación es capaz de funcionar a frecuencia de vídeo siempre que el movimiento del endoscopio sea relativamente lento, mientras que con hexaedros el tiempo necesario aumenta considerablemente. Si el tiempo empleado en los cálculos es elevado o si el endoscopio se mueve rápidamente sucede que, si bien la relocalización reporta éxito, el Tracking vuelve a perderse al ser siguiente frame usado distante en tiempo o posición al de la relocalización.

5.2. Trabajo futuro

Se han implementado cálculos para dos elementos, prismas triangulares y hexaedros, ambos tridimensionales. Es de interés profundizar en el empleo de elementos planos, contrastando la respuesta y aprovechando el menor tiempo de cálculo necesario.

Con el proceso actual que emplea puntos del mapa emparejados para la construcción de la malla, el resultado es altamente irregular debido a la diferente densidad de puntos en diferentes regiones de la misma imagen. Pudiera resultar útil evaluar el mallado

mediante reconstrucción de Poisson o similar que proporcione una malla próxima a lo regular, así como evaluar diferentes detectores de puntos que pudieran proporcionar un mayor número de puntos uniformemente distribuidos, nuevamente llevando a una malla más regular. Se ha comprobado offline que otros detectores como SIFT o SURF funcionan de manera ejemplar en escenas en las que el rendimiento de oFAST es bajo.

Por último, se han estudiado varios artículos para acotar los valores de módulos de elasticidad y Poisson, encontrando gran variabilidad entre los valores que proporcionan unos y otros. La nueva clase instalada en el código permitiría, si se introduce una referencia de medida en el campo visual de la cámara, resolver el problema a la inversa y obtener ambos parámetros con precisión. Esto posibilitaría desarrollar aplicaciones de caracterización de tejidos y/o detección de enfermedades, pues para algunas como la cirrosis el comportamiento mecánico del hígado está directamente relacionado con el avance de la enfermedad.

Capítulo 6

Bibliografía

[1] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, 2007.

[2] P Cheeseman, R Smith, and M Self. A stochastic map for uncertain spatial relationships. In *4th International Symposium on Robotic Research*, pages 467–474, 1987.

[3] Javier Civera, Andrew J. Davison, and J. M. M. Montiel. Interacting multiple model monocular SLAM. In *2008 IEEE International Conference on Robotics and Automation*, pages 3704–3709, 2008.

[4] Javier Civera, Oscar Grasa, Andrew Davison, and J. Montiel. 1-point RANSAC for extended kalman filtering: Application to real-time structure from motion and visual odometry. *J. Field Robotics*, 27:609–631, 09 2010.

[5] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, 2007.

[6] Edward M Mikhail, James S Bethel, and J Chris McGlone. *Introduction to modern photogrammetry*, volume 1. John Wiley & Sons Inc, 2001.

[7] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[8] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[9] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–, 11 2004.

[10] 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*, pages 1255–1262, 2017.

[11] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. *IEEE Transactions on Robotics*, pages 1–17, 2021.

[12] Antonio Agudo, B. Calvo, and J. Montiel. 3D reconstruction of non-rigid surfaces in real-time using wedge elements. In *ECCV Workshops*, 2012.

[13] Juan J. Gómez Rodríguez, José Lamarca, Javier Morlana, Juan D. Tardós, and José M. M. Montiel. Sd-defslam: Semi-direct monocular SLAM for deformable and intracorporeal scenes. *IEEE ICRA 2021.*, 2020.

[14] Íñigo Cirauqui. Evaluación de ORB-SLAM en secuencias de endoscopia médica. *Trabajo Fin de Grado, Escuela de Ingeniería y Arquitectura. Universidad de Zaragoza.*, 2016.

[15] Nader Mahmoud, Íñigo Cirauqui, Alexandre Hostettler, Christophe Doignon, Luc Soler, Jacques Marescaux, and JMM Montiel. ORBSLAM-based endoscope tracking and 3d reconstruction. In *International workshop on computer-assisted and robotic endoscopy*, pages 72–83. Springer, 2016.

[16] Dorian Gálvez-López and J. D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012.

[17] Íñigo Cirauqui. ORB-SLAM2 endoscopy modded. `https://github.com/icirauqui/ORB_SLAM2_E`, 2021.

[18] Íñigo Cirauqui. Camera calibration with OpenCV. `https://github.com/icirauqui/OCV_CameraCalibration`, 2021.

[19] Xiao Lu. A review of solutions for Perspective-n-Point problem in camera pose estimation. *Journal of Physics: Conference Series*, 1087:052009, 09 2018.

[20] Íñigo Cirauqui. Finite element analisys. `https://github.com/icirauqui/FEA2`, 2020.

[21] Laurent Sandrin, Jennifer Oudry, Cécile Bastard, Fournier Céline, Veronique Miette, and Sebastian Mueller. Non-invasive assessment of liver fibrosis by vibration-controlled transient elastography (fibroscan). In *Liver Biopsy*, pages 294–301. IntechOpen, 2011.

[22] Danail Stoyanov Peter Mountney and Guang-Zhong Yang. Three-dimensional tissue deformation recovery and tracking: Introducing techniques based on laparoscopic or endoscopic images. In *IEEE Signal Processing Magazine*, pages 14–24. IEEE, 2010.

[23] Danail Stoyanov, George Mylonas, Fani Deligianni, Ara Darzi, and Guang-Zhong Yang. Soft-tissue motion tracking and structure estimation for robotic assisted MIS procedures. In *Medical Image Computing and Computer Assisted Interventions (MICCAI05)*, pages 139–146, 2005.

[24] Íñigo Cirauqui. Loopclosure seq1 laparoscopia abdominal (hígado). https://youtu.be/B7jxEVgPUX8.

[25] Íñigo Cirauqui. Loopclosure seq2 toracoscopia (corazón). https://youtu.be/QcApCBlShxY.

[26] Íñigo Cirauqui. Loopclosure seq3 laparoscopia abdominal (hígado). https://youtu.be/pQmsx9cQ_W0.

[27] Íñigo Cirauqui. Loopclosure seq4 y seq5 laparoscopia abdominal (hígado y bazo). https://youtu.be/QcApCBlShxY.

[28] Óscar G. Grasa, Ernesto Bernal, Santiago Casado, Ismael Gil, and J. M. M. Montiel. Visual slam for handheld monocular endoscope. *IEEE Transactions on Medical Imaging*, 33(1):135–146, 2014.

[29] Oscar Garcıa, Javier Civera, A Gueme, Victor Munoz, and J. M. M. Montiel. Real-time 3D modeling from endoscope image sequences. *International Conference on Robotics and Automation Workshop on Advanced Sensing and Sensor Integration in Medical Robotics*, 2009.

[30] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2005.

Anexo

Anexo A

Clase FEA2 para análisis por elementos finitos

Se ha programado una clase para el análisis por elementos finitos denominada FEA2, *Finite Element Analisys 2*. Los archivos fuentes de la clase se han instalado entre los del optimizador g2o nombrándolos *FEA2.h* y *FEA2.cc*. Los archivos se incluyen en los procesos de compilación de g2o y en el de ORB-SLAM2.

Todo el software desarrollado se encuentra disponible en Github, donde se han creado proyectos para el ORB-SLAM2 con las modificaciones [17] y para el cálculo por elementos finitos de elemento C3D6 y C3D8 [20].

Requerimientos:

- Point Cloud Library (PCL): mínimo v1.8.
- OpenCV: mínimo v3.4.

A.1. Librería

El código cuenta con 3 bloques diferenciados de funciones. A continuación se proporciona una descripción corta de cada una y un detalle de las funciones de cálculo de matriz elemental y ensamblaje, siendo estas las más relevantes para el trabajo.

1. Funciones generales: constructor, destructor, visualización de datos, guardado en disco y operaciones matemáticas.

   a) **Constructor:** la clase se genera con un identificador que la asocia al frame en que va a actuar. Además, se asignan propiedades del material y se calculan parámetros de Lamé y matriz de comportamiento. También se define el tipo de elemento a utilizar.

   b) **Destructor:** destructor vacío.

   c) **InvertMatrixEigen:** calcula la inversa de la matriz proporcionada.

   d) **MultiplyMatricesEigen:** multiplica las matrices proporcionadas.

2. Funciones de mallado: carga y gestión de puntos, configuración de objetos, suavizado por *moving least squares* y mallado.

   a) **GetMapPointCoordinates:** extrae las coordenadas de los puntos del mapa con los que se trabajará. El vector de puntos ha sido llenado desde la función de relocalización.

   b) **LoadMPsIntoCloud:** genera el objeto nube de puntos de la librería PCL y carga la información de posición 3D y normales desde los puntos del mapa.

   c) **MLS:** aplica una aproximación por Moving Least Squares [**?**] que suaviza la malla, aproximándola a un polinomio, y elimina espurios.

   d) **Compute Mesh:** utiliza la nube suavizada para generar una malla triangular empleando la función GreedyProjection de la librería PCL.

   e) **CalculateGP3Parameters:** calcula los parámetros por los que se rige el proceso de mallado. El cálculo se lleva a cabo en función de la estructura de la nube de puntos.

   f) **tri2quad:** genera cuadriláteros a partir de una malla de triángulos.

   g) **SetSecondLayer:** replica la malla de triángulos a cuadriláteros a una distancia predeterminada para formar los elementos tridimensionales.

3. Funciones de análisis: cálculos de matrices de rigidez elementales, ensamblaje, calculo de desplazamientos, fuerzas y energías.

   a) **Set_u0:** carga la posición inicial de los puntos al comienzo de la optimización no lineal, se empleará como base para el cálculo de desplazamientos que darán la energía de deformación.

   b) **ComputeKeiC3D6:** calcula la matriz elemental para un prisma triangular en función de la posición de los nodos proporcionada. Se proporciona información teórica adicional en el Anexo 2.

   c) **ComputeKeiC3D8:** calcula la matriz elemental para un hexaedro en función de la posición de los nodos proporcionada. Se proporciona información teórica adicional en el Anexo 2.

   d) **MatrixAssemblyC3D6:** proceso de ensamblaje de matriz de rigidez de elementos prisma triangular. Se proporciona información teórica adicional en el Anexo 2.

   e) **MatrixAssemblyC3D8:** proceso de ensamblaje de matriz de rigidez de elementos hexaedro. Se proporciona información teórica adicional en el Anexo 2.

   f) **ImposeDirichletEncastre_K:** impone las condiciones de contorno de Dirichlet a la matriz de rigidez previamente calculada. Se encastra la cara duplicada del modelo.

g) **ImposeDirichletEncastre_a:** impone las condiciones de contorno de Dirichlet al vector de desplazamientos previamente calculado.

h) **Set_uf:** carga las posiciones de los puntos actualizadas en cada etapa de la optimización por Levenberg Marquardt.

i) **ComputeDisplacement:** Calcula el desplazamiento de los puntos tras el establecimiento de las posiciones finales.

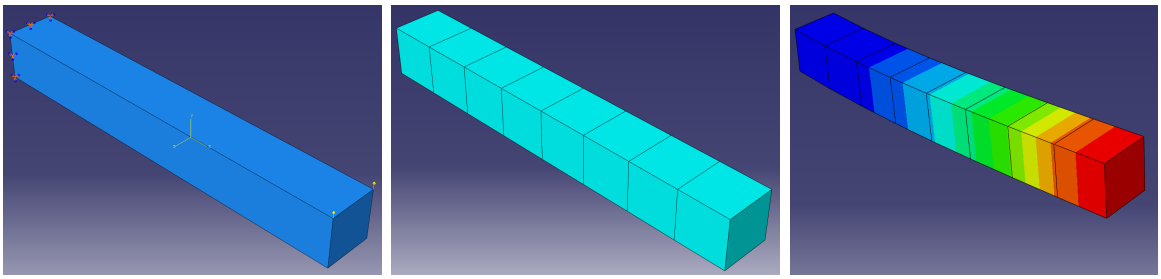
j) **ComputeForces:** Calcula la fuerza necesaria para los desplazamientos generados anteriormente.

k) **ComputeStrainEnergy:** Calcula la energía de deformación elástica causada por esos desplazamientos.

l) **NormalizeStrainEnergy:** Normaliza la energía de deformación calculada.

A.2. Verificación respecto Abaqus

Se presenta una verificación contra el software Abaqus CAE [**?**]. Para ello, se analiza una viga simple de perfil cuadrado como la de la Figura A.1a de dimensiones 8x1x1, esta se encastra en el extremo izquierdo y se aplican dos fuerzas puntuales de 0.5 N en los nodos superiores del extremo derecho. Se genera un mallado en hexaedros C3D8 de lado 0.25, dando lugar a los 64 elementos mostrados en la Figura A.1b. Se resuelve el modelo dando lugar a la deformación mostrada en la Figura A.1c.



(a) (b) (c)

Figura A.1: Modelo en Abaqus con elementos C3D8. (a) Muestra una viga de perfil cuadrado con encastre en un extremo y fuerzas puntuales en el contrario. (b) Mallado aplicado. (c) Deformación generada.

Se obtienen, para los nodos de la cara en la que se aplican las fuerzas, los desplazamientos de la Tabla A.1. La configuración de nodos, encastre y fuerzas se carga a la versión de la librería disponible en [20], los mismos archivos empleados para este análisis se incluyen en la carpeta *input_C3D8* a modo de ejemplo. Se ejecuta el programa dando lugar a los desplazamientos del segundo bloque de columnas de la Tabla A.1.

Nodo	Abaqus			FEA2			Error(%)		
Nodo	u_x	u_y	u_z	u_x	u_y	u_z	e_x	e_y	e_z
1	-0.021	0.198	0	-0.021	0.198	0	0	0	0
2	-0.021	0.198	0	-0.021	0.198	0	0	0	0
3	-0.02	0.156	0.001	-0.02	0.156	0.001	0	0	0
4	-0.02	0.156	-0.001	-0.02	0.156	-0.001	0	0	0
5	-0.018	0.116	0.001	-0.018	0.117	0.001	0	1	0
6	-0.018	0.116	-0.001	-0.018	0.116	-0.001	0	0	0
7	-0.016	0.081	0.001	-0.016	0.081	0.001	0	0	0
8	-0.016	0.081	-0.001	-0.016	0.081	-0.001	0	0	0
9	-0.013	0.051	0.002	-0.013	0.051	0.002	0	0	0
10	-0.013	0.051	-0.002	-0.013	0.051	-0.002	0	0	0
11	-0.01	0.027	0.002	-0.01	0.027	0.002	0	0	0
12	-0.01	0.027	-0.002	-0.01	0.027	-0.002	0	0	0
13	-0.006	0.011	0.002	-0.006	0.011	0.002	0	0	0
14	-0.006	0.011	-0.002	-0.006	0.011	-0.002	0	0	0
15	-0.002	0.003	0.002	-0.002	0.003	0.002	0	0	0
16	-0.002	0.003	-0.002	-0.002	0.003	-0.002	0	0	0
17	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0
19	0.021	0.197	-0.001	0.021	0.197	-0.001	0	0	0
20	0.021	0.197	0.001	0.021	0.197	0.001	0	0	0
21	0.02	0.156	-0.001	0.02	0.156	-0.001	0	0	0
22	0.02	0.156	0.001	0.02	0.156	0.001	0	0	0
23	0.018	0.116	-0.001	0.018	0.117	-0.001	0	1	0
24	0.018	0.116	0.001	0.018	0.116	0.001	0	0	0
25	0.016	0.081	-0.001	0.016	0.081	-0.001	0	0	0
26	0.016	0.081	0.001	0.016	0.081	0.001	0	0	0
27	0.013	0.051	-0.002	0.013	0.051	-0.002	0	0	0
28	0.013	0.051	0.002	0.013	0.051	0.002	0	0	0
29	0.01	0.027	-0.002	0.01	0.027	-0.002	0	0	0
30	0.01	0.027	0.002	0.01	0.027	0.002	0	0	0
31	0.006	0.011	-0.002	0.006	0.011	-0.002	0	0	0
32	0.006	0.011	0.002	0.006	0.011	0.002	0	0	0
33	0.002	0.003	-0.002	0.002	0.003	-0.002	0	0	0
34	0.002	0.003	0.002	0.002	0.003	0.002	0	0	0
35	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0

Tabla A.1: Comparativa de resultados Abaqus-C++ en metros y % de error para una precisión de milímetros.

Anexo B

Datos adicionales análisis experimental

Se proporciona información adicional para identificar las secuencias. La Tabla B.1 proporciona nombres de archivo y segundos de la secuencia empleados para mapeado y evaluación, y la Tabla B.2 muestra los resultados obtenidos con más detalle que el capítulo 4.

Sec	Bag (.bag)	Settings (.yaml)	VideoS	MapS	MapE	RelocS	RelocE
1	Hamlyn05	sHamlyn05	5	20	50	57	79
2	Hamlyn03	sHamlyn03	45	45	60	5	45
3	lrLUCseq2	sEndoscope2	5	5	15	15	28
4	RelocS2	sEndoscope1	NA	NA	NA	0	19
5	RelocS3	sEndoscope1	NA	NA	NA	0	55

Tabla B.1: Detalle de las secuencias analizadas, la primera columna asigna un id a la secuencia, que se respeta en todo el documento. Se proporcionan, de izquierda a derecha, el nombre de la bolsa de imágenes (.bag), el del archivo de settings (.yaml), el segundo de inicio del vídeo, el segundo en que se inicia el mapeo, el segundo en que se completa el mapeo, el segundo en que se inicia la evaluación de la relocalización y el segundo en que se completa la evaluación.

Seq	Model	TP	FP	FN	Pr	Rc
1	Std	80	32	134	0.588	0.343
	C3D6	92	28	26	0.767	0.780
	C3D8	80	32	74	0.717	0.519
2	Std	31	111	758	0.218	0.039
	C3D6	128	94	114	0.577	0.529
	C3D8	91	136	172	0.401	0.346
3	Std	52	5	61	0.912	0.460
	C3D6	46	6	21	0.885	0.687
	C3D8	46	10	28	0.821	0.622
4	Std	72	31	89	0.699	0.447
	C3D6	75	37	51	0.670	0.595
	C3D8	74	48	59	0.607	0.556
5	Std	117	115	764	0.504	0.133
	C3D6	116	214	262	0.352	0.307
	C3D8	91	263	421	0.257	0.178

Tabla B.2: True Positives, False Positives, False Negatives, Precision y Recall para cada modelo y secuencia.

Lista de Figuras

Lista de Tablas