Universidad Zaragoza

Trabajo de Fin de Máster

Seguimiento robusto de características visuales para secuencias de imagen médica en ORB-SLAM2

Robust tracking of visual features for medical image sequences in ORB-SLAM2

Autor

Juan José Gómez Rodríguez

Director

Juan Domingo Tardós Solano

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2019

AGRADECIMIENTOS

Agradecer en primer lugar a Juan D. Tardós, por su orientación e implicación a lo largo de este y otros proyectos junto a el. *¡Muchas gracias Mingo!*.

También me gustaría agradecer a José María Martínez Montiel, quien sin ser mi tutor, siempre ha tenido interés y consejos que darme.

Finalmente, agradecer a mi familia y amigos por el apoyo durante todo este tiempo.

RESUMEN

El problema de *localización y mapeo simultáneos*, más conocido por sus siglas en inglés SLAM, consiste en localizar un agente (por ejemplo, un endoscopio en una intervención quirúrgica) dentro de un mapa generado gracias a información obtenida por una serie de sensores de a bordo. ORB-SLAM2 es un sistema de SLAM desarrollado por el grupo de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza, que utiliza cámaras como sensores. Se trata de un sistema de SLAM indirecto, por lo que basa su funcionamiento en la extracción de características visuales para el seguimiento de la cámara y triangulación de puntos para el mapa. Para ello, utiliza puntos FAST con descriptores ORB para realizar los emparejamientos entre las características visuales extraídas.

En el presente trabajo se ha investigado el uso de un método alternativo para realizar el emparejamiento de características visuales basado en el uso del flujo óptico mediante el método de Lucas-Kanade. Para ello, se han desarrollado una serie de mejoras al método para dotarle de invarianza a la iluminación y rotación, obteniendo un método más preciso y robusto que el método básico, a cambio de ser computacionalmente más costoso. Para reducir el coste, se ha desarrollado el uso de patrones de píxeles, resultando en un método más eficiente a la vez que más preciso y robusto que las implementaciones estándar.

El objetivo final del proyecto es integrar el método de flujo óptico desarrollado en ORB-SLAM2 para poder procesar secuencias endoscópicas obtenidas de cirugías mínimamente invasivas. Éstas suponen un gran desafío debido a condiciones de iluminación altamente cambiantes, baja repetibilidad de las texturas y superficies deformables. Para las pruebas se ha hecho uso secundario de secuencias obtenidas en intervenciones reales de personas, donde el cirujano no sabía que serían utilizadas por un sistema SLAM, ni se modificaron las herramientas ni procedimientos de la intervención. Los resultados obtenidos demuestran que la técnica de flujo óptico desarrollada mejora la calidad de la inicialización del mapa, y los puntos seguidos y triangulados, obteniendo una nueva versión de ORB-SLAM2 que es capaz de procesar de forma efectiva secuencias endoscópicas.

Índice

Capítulo 1

Introducción y objetivos

1.1. Sistemas de localización y mapeo simultáneo

El problema de *localización y mapeo simultáneo*, más conocido por sus siglas en inglés SLAM, surgió en el ámbito de la robótica ante la necesidad de crear un mapa del entorno de un robot y, a la vez, localizarlo dentro del mismo utilizando la información que brindan los diferentes sensores que pueda llevar el robot.

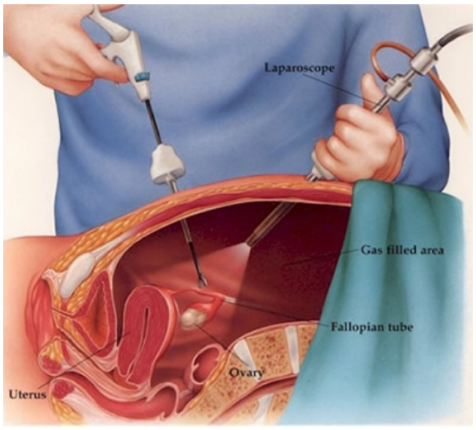
Existen una gran variedad de sensores que se pueden utilizar para resolver el problema de SLAM, sin embargo, los más populares, tanto por su coste como por su versatilidad, son las cámaras. Gracias a ello, se han introducido cámaras como sensor principal en numerosas aplicaciones que requieren de una solución de SLAM, como robots o gafas de realidad aumentada o virtual.

En los últimos años se han generalizado las intervenciones quirúrgicas mínimamente invasivas (Fig. 1.1), consistentes en realizar una o más incisiones pequeñas en el cuerpo para introducir un laparoscopio (instrumento delgado en forma de tubo con una luz y una cámara para observar) a través de una abertura a fin de guiar la cirugía. Este tipo de intervenciones producen un flujo de imágenes que pueden ser procesadas por un sistema SLAM para reconstruir las cavidades internas del paciente, realizar anotaciones en realidad aumentada, etc.

Una solución al problema de SLAM visual, propuesta por el grupo de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza es ORB-SLAM2 [6], el cual esta siendo utilizado ampliamente en numerosos productos y proyectos comerciales.

1.2. Objetivos y alcance del proyecto

ORB-SLAM2 se ha probado ampliamente en secuencias y entornos cotidianos como oficinas o calles de una ciudad. En este tipo de entornos se puede asumir que la escena observada por la cámara es rígida y que dispone de suficiente textura como



(a)

(b)

Figura 1.1: Funcionamiento interno de una laparoscopia (a). Intervención mediante laparoscopia (b).

para poder realizar correctamente extracción y emparejamiento de características visuales. Sin embargo, la escena observada en secuencias tomadas en el interior del cuerpo humano son de naturaleza deformable (latidos, palpitaciones, movimientos del paciente) lo que puede dificultar el correcto seguimiento de la cámara. A ello se le suman cambios de iluminación bruscos junto con escenas con baja textura que pueden dificultar el emparejamiento de características visuales junto con movimientos bruscos del endoscopio. Por ello, se propone un método basado en el uso del flujo óptico observado en las imágenes para el emparejamiento robusto de características visuales.

Por tanto, los objetivos de la presente investigación son los siguientes:

1. Estudio e implementación de un algoritmo de seguimiento de características visuales mediante flujo óptico basado en el algoritmo de Lucas-Kanade [4].

2. Optimización de la precisión y tiempo de cómputo del algoritmo anterior para permitir un funcionamiento preciso en tiempo real.

3. Estudio e integración del algoritmo de flujo óptico en ORB-SLAM2.

4. Análisis y evaluación de prestaciones de la nueva implementación con secuencias médicas.

De esta manera, se pretende poder procesar secuencias obtenidas de intervenciones quirúrgicas mínimamente invasivas sin tener que alterar la manera de proceder de los propios cirujanos.

1.3. Herramientas utilizadas

Para el desarrollo del presente proyecto, se ha partido de la versión de ORB-SLAM2 presente en su repositorio oficial (`https://github.com/raulmur/ORB_SLAM2`). En base a ella, se ha utilizado el siguiente listado de herramientas para la realización del proyecto:

- C++: lenguaje de programación de ORB-SLAM2.
- Valgrind: herramienta para medir tiempos de computo en programas desarrollados en C++.
- Python: lenguaje de programación usado para seleccionar y procesar las imágenes médicas.
- OpenCV: librería de código abierto con estructuras de datos y algoritmos de visión por computador.
- Bash: lenguaje de comandos. Usado para automatizar la ejecución de pruebas y recolección de resultados experimentales.
- Git + GitHub: software y plataforma de desarrollo para el control de versiones.
- LaTeX: sistema de composición de textos para la redacción del presente documento.
- Google scholar: motor de búsqueda enfocado en la búsqueda de contenido y literatura científico-académica usado como apoyo a la investigación.

1.4. Estructura del documento

La investigación desarrollada se expone a lo largo de 4 capítulos en el presente documento. El capítulo 2 esta dedicado a presentar conceptos básicos que serán utilizados a lo largo de todo el documento. En el capítulo 3 se presenta una revisión al algoritmo de Lucas-Kanade para el cálculo del flujo óptico donde se realizan una serie de mejoras al algoritmo. El capítulo 4 esta dedicado a la integración, implementación y resultados del nuevo algoritmo de Lucas-Kanade mejorado en ORB-SLAM2. Los resultados de dicha integración se presentan en el capítulo 5. Finalmente, el capítulo 6 recoge las conclusiones finales de la presente investigación, futuras líneas de trabajo en base a los resultados obtenidos y gestión del proyecto.

Capítulo 2

Conceptos

Para facilitar la comprensión del presente documento, a continuación se presentan una serie de conceptos que serán utilizados a lo largo del documento relacionados con diversos elementos utilizados por un sistema de SLAM como ORB-SLAM2.

2.1. ORB-SLAM2

ORB-SLAM2 [6] es un software desarrollado por la Universidad de Zaragoza propuesto como solución al problema de SLAM. Su principal característica es el ser un sistema de SLAM basado en *KeyFrames* (fotogramas especiales de la secuencia seleccionados por aportar gran cantidad de información sobre el entorno) que utiliza el mismo tipo de características para todas las tareas de SLAM.

La estructura de ORB-SLAM2 (Figura 2.1) se divide en 3 hilos de ejecución paralelos que realizan tareas diferentes: el hilo de *tracking*, encargado de estimar la pose de la cámara en cada *frame* del vídeo; el hilo de *local mapping*, encargado de triangular y añadir nuevos puntos al mapa, y de optimizar tanto las posiciones de las cámaras como del mapa 3D construido; y *loop closing*, encargado de detectar y cerrar bucles en la trayectoria de la cámara.

ORB-SLAM2 requiere de realizar emparejamientos entre las características visuales que utiliza en sus 3 hilos de ejecución. Estos emparejamiento los realiza de manera pasiva mediante descriptores ORB: en cada imagen extrae un conjunto de características visuales e intenta emparejarlas con las características visuales extraídas en imágenes anteriores mediante descriptores ORB calculados para las mismas. Sin embargo, en este trabajo, a partir de la estructura base se ha utilizado el flujo óptico para realizar un seguimiento activo de las características visuales: seguir un conjunto concreto de características visuales extraídas en una imagen de referencia.

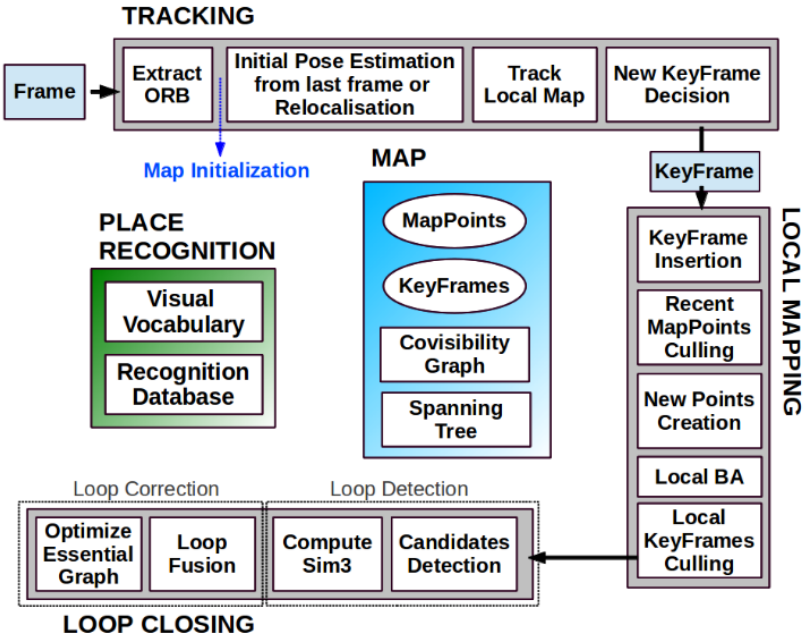


Figura 2.1: Estructura general de ORB-SLAM2. El hilo de *Tracking* preprocesa las imágenes de entrada extrayendo características visuales y describiéndolas de manera que todo el sistema trabaje con las mismas.

2.2. Flujo Óptico

El flujo óptico es un problema ampliamente estudiado en el campo de la visión por computador que consiste en estimar el patrón de movimiento aparente de objetos entre dos imágenes consecutivas, causado por movimientos relativos entre cámara y escena. De esta manera, el flujo óptico puede ser representado como un campo vectorial 2D donde cada vector $d = \begin{pmatrix} d_x & d_y \end{pmatrix}$ representa el desplazamiento de un objeto de la escena entre las dos imágenes consideradas. Para ello, se toman en consideración las siguientes asunciones:

- La intensidad de los píxeles de un objeto en la imagen no varía entre imágenes consecutivas.
- Píxeles cercanos experimentan desplazamientos similares.

Por tanto, el objetivo del problema del Flujo Óptico consiste en obtener los diferentes vectores $d = \begin{pmatrix} d_x & d_y \end{pmatrix}$ de flujo óptico para cada objeto en la imagen que se quiera seguir. Para ello, considérese una secuencia de imágenes $I(x, y, t)$ donde x, y son las coordenadas en píxeles de la imagen de la secuencia tomada en el instante t. De esta manera, un objeto inicialmente observado en $p = (x_p, y_p)$ en el instante t es observado en $I(x_p, y_p, t)$. Pasado un tiempo Δt, dicho objeto sufre un desplazamiento

en píxeles de $(\Delta x, \Delta y)$, siendo observado nuevamente en la secuencia de imágenes en $I(x_p + \Delta x, y_p + \Delta y, t + \Delta t)$. De manera general y bajo la asunción de que la intensidad medida en imagen de un objeto en imágenes consecutivas no varía, se puede obtener la siguiente ecuación:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{2.1}$$

Realizando una aproximación en series de Taylor del lado derecho de la ecuación anterior alrededor del punto (x, y, t), se obtiene:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \tag{2.2}$$

De esta manera, de las ecuaciones anteriores se puede deducir que:

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0 \tag{2.3}$$

o de igual manera:

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} = 0 \tag{2.4}$$

Obteniendo finalmente la *ecuación del Flujo Óptico*:

$$I_x d_x + I_y d_y + I_t = 0 \tag{2.5}$$

Donde $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$ y $I_t = \frac{\partial I}{\partial t}$. Obsérvese que de esta manera se puede obtener el vector de flujo óptico $d = (d_x, d_y)$, sin embargo, se tiene solo una ecuación y dos incógnitas, por lo que se trata un sistema indeterminado de ecuaciones y, por tanto, no se puede resolver.

2.2.1. Método de Lucas-Kanade

Descripción general

El método de Lucas-Kanade [4] es un algoritmo ideado para resolver la *ecuación del Flujo Óptico*. Para ello, aprovecha la asunción de que los píxeles cercanos en una imagen sufren desplazamientos similares para obtener varias ecuaciones de flujo óptico y sobre-determinar el sistema de ecuaciones presentado anteriormente.

Más concretamente, el método de Lucas-Kanade toma una ventana rectangular de píxeles de tamaño $\omega = (2\omega_x + 1, 2\omega_y + 1)$ centrada en el píxel sobre el que se pretende calcular el flujo óptico $p = (x_p, y_p)$. De esta manera, se asume que todos los píxeles en esa misma ventana experimentarán el mismo desplazamiento y por tanto, un mismo vector de flujo óptico $d = (d_x, d_y)$. Por ello, el método toma una ventana de píxeles

en la imagen de referencia y la busca en la nueva imagen. Gracias a ello, se pueden apilar todas las *ecuaciones de Flujo Óptico* de los píxeles dentro de la ventana para sobre-determinar el sistema y poder resolverlo de la siguiente manera:

$$\begin{bmatrix} d_x \\ d_y \end{bmatrix} = G^{-1} \cdot b \tag{2.6}$$

donde:

$$G = \sum_{x=x_p-\omega_x}^{x_p+\omega_x} \sum_{y=y_p-\omega_y}^{y_p+\omega_y} \begin{bmatrix} I_x^2(x,y) & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y^2(x,y) \end{bmatrix} \tag{2.7}$$

$$b = \sum_{x=x_p-\omega_x}^{x_p+\omega_x} \sum_{y=y_p-\omega_y}^{y_p+\omega_y} \begin{bmatrix} -I_x(x,y)I_t(x,y) \\ -I_y(x,y)I_t(x,y) \end{bmatrix} \tag{2.8}$$

Cálculo iterativo del Flujo Óptico

A pesar de que el problema del flujo óptico esta fundamentado sobre la asunción de que la intensidad de los píxeles de un objeto en la imagen son iguales entre imágenes consecutivas, en el mundo real nunca se cumple. Por ello, el problema del Flujo Óptico puede ser visto como un problema de minimización: encontrar el vector de flujo óptico d que minimice la diferencia de intensidades de las ventanas que utiliza Lucas-Kanade entre imágenes consecutivas. Matemáticamente, la función a minimizar es la diferencia de intensidad entre los píxeles de las ventanas:

$$\underset{d}{\operatorname{argmin}}\, \epsilon(d) = \epsilon(d_x, d_y) = \sum_{x=x_p-\omega_x}^{x_p+\omega_x} \sum_{y=y_p-\omega_y}^{y_p+\omega_y} (I(x,y,t) - I(x+d_x, y+d_y, t+\Delta t))^2 \tag{2.9}$$

De esta manera, se resuelve el problema de optimización propuesto siguiendo un esquema de Newton-Raphson: a partir de una estimación inicial, se calcula en cada iteración una estimación flujo óptico mediante la ecuación 2.6 que servirá como semilla en la siguiente iteración. De esta forma, se aumenta la precisión del algoritmo al refinar iterativamente la solución obtenida, permitiendo calcular el flujo óptico para objetos que se han desplazado considerablemente en la imagen. Este esquema iterativo se repite hasta que se realizan un número máximo de iteraciones o hasta que la diferencia de intensidades $\epsilon(d_x, d_y)$ es menor que un umbral determinado.

Método piramidal

La solución propuesta anteriormente solo encuentra soluciones correctas si el vector de flujo óptico es pequeño, es decir, que el movimiento de los objetos a seguir en la

imagen es pequeño. Sin embargo, esto no siempre se cumple y es necesario disponer de un método general capaz de procesar grandes desplazamientos.

Para ello, se propone un procesado multi-escala [1] (Fig. 2.2) típico en visión por computador, construyendo una representación piramidal de las imágenes a procesar por el algoritmo de flujo óptico. En dicha representación, cada nivel de la pirámide esta formada por una versión con menor resolución de la imagen original, siendo el nivel 0 la imagen original y L_m el nivel máximo con la imagen con menor resolución. Con ello, desplazamientos de muchos píxeles en los niveles bajos de la pirámide se convierten en desplazamientos de pocos píxeles en los niveles superiores de la pirámide.

Tradicionalmente, en el método de Lucas-Kanade se utilizan pirámides con un factor de escala 2, es decir, las dimensiones de la imagen se ven reducidas a la mitad por cada nivel en la pirámide en el que se avanza. De esta manera, para un punto que en la imagen se encuentre en el píxel de coordenadas $p = (x_p, y_p)$ se pueden calcular sus coordenadas en los diferentes niveles L de la pirámide de acuerdo a la siguiente fórmula:

$$p_L = \frac{p}{2^L} \tag{2.10}$$

Así pues, se comienza realizando una estimación del flujo óptico de un objeto en el nivel superior de la pirámide, la cual será propagada a los niveles inferiores. De esta manera, para cada nivel de la pirámide, se resuelve de manera iterativa el flujo óptico, el cual será usado como semilla en el nivel inmediatamente inferior de la misma.

Esquema final del algoritmo

Finalmente, el esquema final del algoritmo que calcula el flujo óptico para un conjunto de n puntos $\mathcal{P}$ entre 2 imágenes puede ser resumido en pseudo-código (1) de la siguiente manera:

Otras consideraciones

El método de Lucas-Kanade es un algoritmo parametrizable, por lo que es importante tener en cuenta qué valores se seleccionan para obtener unos resultados correctos del mismo. A continuación se enumeran los mismos:

1. **Tamaño de ventana** w_x, w_y: el tamaño de la ventana seleccionada es crucial. Normalmente un tamaño pequeño implica mayor precisión local al capturar exclusivamente el punto en la imagen a seguir además de cumplir mejor la asunción de que píxeles cercanos experimentan el mismo flujo óptico. Sin embargo, una ventana más grande permite manejar desplazamientos más grandes en la imagen y aumentar la robustez del algoritmo. Además, de manera intuitiva

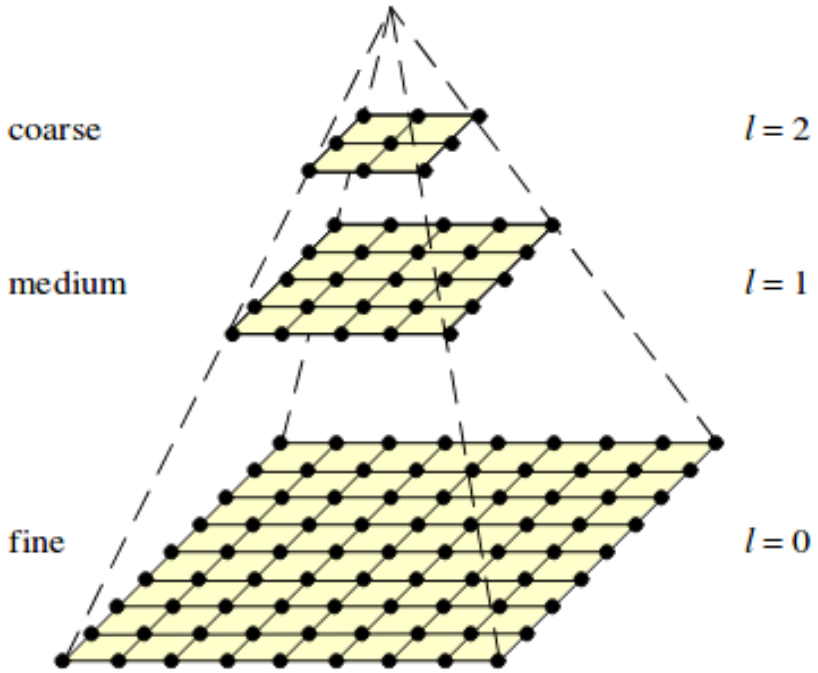


Figura 2.2: Ejemplo de una representación piramidal de una imagen con un factor de escala de 2. Imagen extraída de [10].

Algorithm 1 Pseudo-código del método de Lucas-Kanade iterativo para el cálculo del flujo óptico usando pirámides de imágenes

procedure LUCAS-KANADE

$I \leftarrow I(:,:,t), J \leftarrow I(:,:,t+\Delta t)$

$\{I^L\}_{L=0,\ldots,L_m}, \{J^L\}_{L=0,\ldots,L_m}$ ▷ Construir pirámides de las imágenes

for all punto $p \in \mathcal{P}$ **do**

$d = (d_x, d_y) \leftarrow (0,0)$ ▷ Estimación inicial del Flujo Óptico

for $L = L_m$ hasta 0 **do**

$p_L \leftarrow \frac{p}{2^L} = (x_l, y_l)$ ▷ Calcular coordenadas de p en el nivel L

$$G \leftarrow \sum_{x=x_l-\omega_x}^{x_l+\omega_x} \sum_{y=y_l-\omega_y}^{y_l+\omega_y} \begin{bmatrix} I_x^2(x,y) & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y^2(x,y) \end{bmatrix}$$

for $i = 1$ hasta $maxIter$ o η menor que un umbral **do**

$$b \leftarrow \sum_{x=x_p-\omega_x}^{x_p+\omega_x} \sum_{y=y_p-\omega_y}^{y_p+\omega_y} \begin{bmatrix} -I_x(x,y)I_t(x,y) \\ -I_y(x,y)I_t(x,y) \end{bmatrix}$$

$\eta \leftarrow G^{-1} \cdot b$ ▷ Flujo Óptico en la iteración i

$d \leftarrow d + \eta$ ▷ Semilla para la siguiente iteración

$d \leftarrow 2 \cdot d$ ▷ Flujo Óptico para el siguiente nivel

$p_t \leftarrow p + d$ ▷ Posición final de p en la nueva imagen

se comprueba que a mayor tamaño de ventana, mayor coste computacional del algoritmo. Un tamaño total de ventana óptimo suele ser de $(11, 11)$.

2. **Número de niveles en las pirámides de imágenes** L: un número alto de niveles en la pirámide permite seguir de manera más robusta y precisa objetos que experimentan grandes translaciones en la imagen. Sin embargo, esto viene acompañado con un coste computacional proporcional al número de niveles de la pirámide.

3. **Número de iteraciones máximas** $maxIter$: un número pequeño de iteraciones puede provocar que el algoritmo no tenga suficiente tiempo para converger mientras que un número demasiado elevado puede suponer un gasto computacional innecesario o incluso provocar la divergencia del algoritmo. Normalmente se ha observado que entre 5 y 10 iteraciones son suficientes para que el algoritmo funcione correctamente.

Por otro lado, es importante notar que gran parte de la precisión del método de Lucas-Kanade es debido a que realiza cálculos con precisión subpíxel. Por lo tanto, es necesario poder calcular el valor de intensidad entre píxeles. Para ello, se realiza una interpolación bilineal entre los valores de intensidad de la imagen original de tal manera que:

$$\begin{aligned} x_p &= x_0 + \alpha_x \\ y_p &= y_0 + \alpha_y \end{aligned} \tag{2.11}$$

$$\begin{aligned} I^L(x, y) = (1 - \alpha_x)(1 - \alpha_y) I^L(x_0, y_0) + \alpha_x (1 - \alpha_y) I^L(x_0 + 1, y_0) + \\ (1 - \alpha_x)\alpha_y I^L(x_0, y_0 + 1) + \alpha_x \alpha_y I^L(x_0 + 1, y_0 + 1) \end{aligned} \tag{2.12}$$

2.3. Características visuales y puntos de interés

ORB-SLAM2 es un método de SLAM indirecto: basa su funcionamiento en el emparejamiento de características visuales o puntos de interés (generalmente esquinas) para poder realizar el seguimiento de la cámara y la construcción de un mapa disperso. El propio nombre del algoritmo viene del que usa puntos FAST [7] descritos mediante ORB [8]. Este tipo de descriptores están en el estado del arte, siendo de los más eficientes computacionalmente a la hora de ser computados y comparados, ofreciendo buenas tasas de precisión en los emparejamientos producidos.

Sin embargo, se ha observado en ORB-SLAM2 un fenómeno denominado *feature flickering* o parpadeo de características producido al intentar emparejar los puntos ORB

extraídos entre dos imágenes consecutivas. Este fenómeno se caracteriza por el hecho de que periódicamente conjuntos de puntos dejen de ser emparejados en una imagen (desaparezcan) y vuelvan a ser emparejados en imágenes posteriores (reaparezcan) a pesar de que la cámara no sufra ningún tipo de movimiento (Fig. 2.3) o la escena apenas cambie, dando por tanto, falsos negativos a la hora de emparejar los puntos.

Para entender el origen de este fenómeno, es necesario describir el funcionamiento de los descriptores ORB. Los descriptores ORB son descriptores binarios, es decir, se representan mediante una ristra de bits. Para comparar dos descriptores y poder establecer emparejamientos, se calcula la distancia de *Hamming* entre ambos descriptores, es decir, el número de bits que difieren. De esta manera, para que dos puntos sean emparejados, la distancia de *Hamming* de sus descriptores debe ser menor que cierto umbral para poder establecer dicho emparejamiento.

El fenómeno del parpadeo de características se debe a que ante posibles cambios en la iluminación, orientación, escala del punto en la imagen, etc, la distancia de *Hamming* de los descriptores ORB de un mismo objeto en imágenes consecutivas puede superar el umbral y no ser correctamente emparejado. Esto es debido a limitaciones en la información que puede capturar y representar el descriptor ORB.

En ORB-SLAM2, se ha observado que el fenómeno de parpadeo de características visuales provoca pequeños errores en la estimación de la cámaras. Por ejemplo, en trabajos previos se ha observado que cuando la cámara está estacionaria, la estimación de ORB-SLAM2 sufre pequeñas oscilaciones.

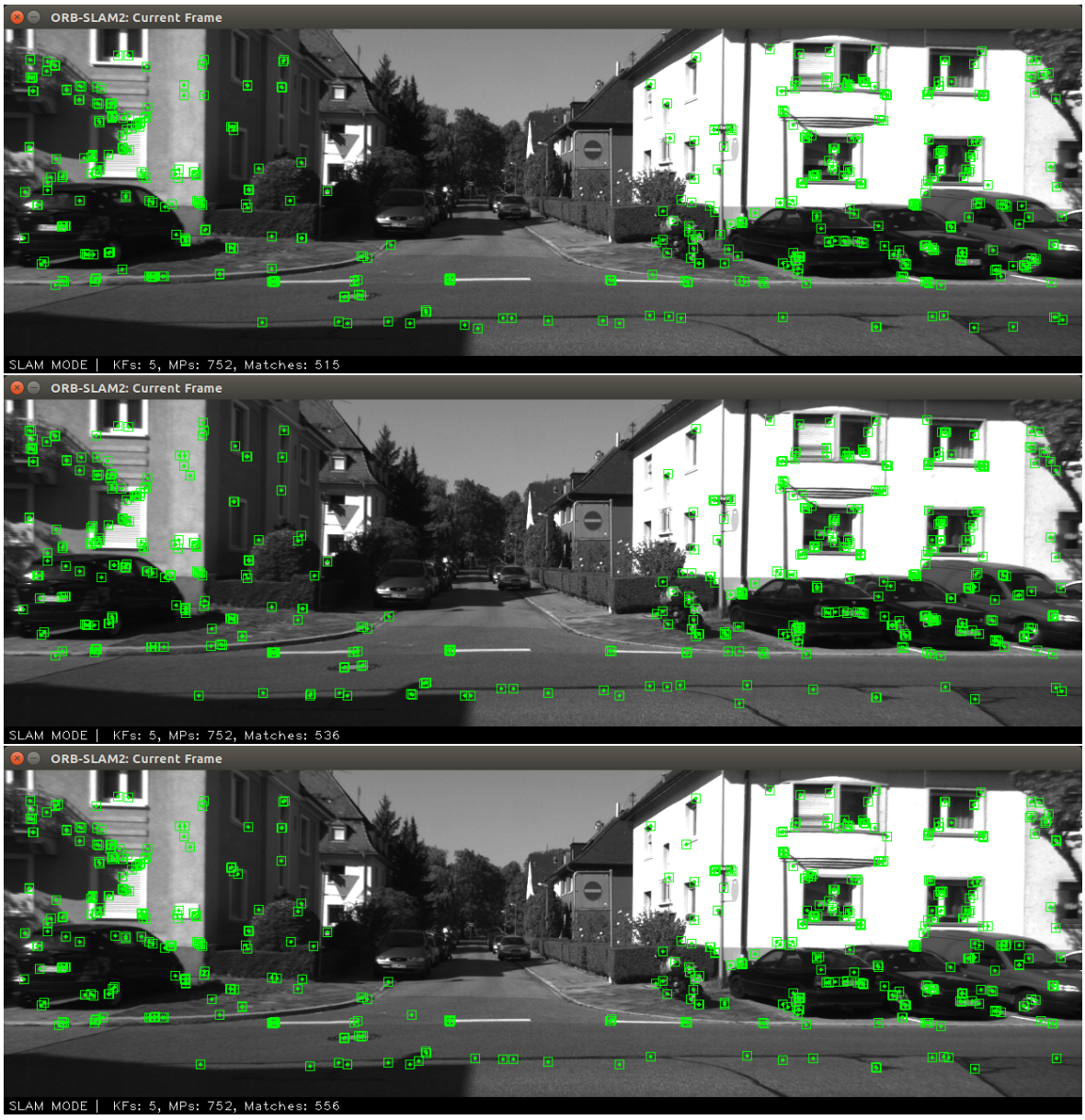


Figura 2.3: Imágenes consecutivas en una secuencia sin movimiento de la cámara ni cambios de iluminación. Obsérvese como algunas características visuales desaparecen y reaparecen.

Capítulo 3

Método de Lucas-Kanade mejorado

El método de Lucas-Kanade presentado anteriormente presenta una serie de inconvenientes que pueden afectar a la precisión y rendimiento computacional del algoritmo, especialmente en secuencias laparoscópicas, y que no han sido considerados en el algoritmo original. A continuación, se presentan dichos inconvenientes y las soluciones adoptadas para solventarlas.

3.1. Invarianza a cambios de iluminación

El método de Lucas-Kanade opera directamente sobre los niveles de gris de la imagen. Como se ha visto en la eq. 2.9, el método busca minimizar la diferencia entre los niveles de gris de las ventanas en la imagen de referencia y en la actual. Sin embargo, ante cambios repentinos de iluminación, los valores de gris en la ventana a buscar en la imagen actual pueden variar, provocando que la convergencia del método de Lucas-Kanade se vea afectada. Esto es especialmente común en escenas con iluminación artificial, como secuencias laparoscópicas, donde el propio endoscopio tiene acoplado la luz que ilumina los alrededores. Esto provoca que la propia luminancia que recibe la cámara depende del propio movimiento de la cámara: al acercarse el endoscopio a tejidos y superficies, la iluminación aumenta con el cuadrado de la distancia.

Para solventar esta situación, muchos métodos deciden hacer una corrección global de la iluminación asumiendo que dicho cambio ha sido igual en toda la imagen. Para ello, se utilizan métodos de procesado de imagen como ecualización del histograma de los niveles de gris de la imagen. Sin embargo, estos cambios de la iluminación rara vez afectan de manera uniforme a lo largo de toda la imagen, habiendo zonas que cambien más que otras.

Por tanto, para conseguir invarianza a la iluminación de manera local, se realiza una corrección de la iluminación[1] sobre los niveles de grises al calcular sobre I_t un valor

[1]Basado en la implementación de `https://cecas.clemson.edu/~stb/klt/`

α de ganancia y β de sesgo entre las ventanas que utiliza el método de Lucas-Kanade. De esta manera, sean I y J dos imágenes consecutivas, se quiere calcular el flujo óptico $d = (d_x, d_y)$ del punto $p = (p_x, p_y)$ observado en I en la imagen consecutiva J. Asumiendo que se toma una ventana de tamaño $\omega = (2\omega_x + 1, 2\omega_y + 1)$ alrededor de p, los parámetros α y β se calculan como sigue:

$$\begin{aligned} \alpha &= \sqrt{\frac{\mu_{I^2}}{\mu_{J^2}}} \\ \beta &= \mu_I - \alpha\mu_J \end{aligned} \tag{3.1}$$

donde

$$\begin{aligned} \mu_I &= \frac{\sum_{x=x_p-\omega_x}^{x_p+\omega_x}\sum_{y=y_p-\omega_y}^{y_p+\omega_y} I(x,y)}{(2\omega_x+1)\cdot(2\omega_y+1)}; \quad \mu_J = \frac{\sum_{x=x_p+d_x-\omega_x}^{x_p+d_x+\omega_x}\sum_{y=y_p+d_y-\omega_y}^{y_p+d_y+\omega_y} J(x,y)}{(2\omega_x+1)\cdot(2\omega_y+1)} \\ \mu_{I^2} &= \frac{\sum_{x=x_p-\omega_x}^{x_p+\omega_x}\sum_{y=y_p-\omega_y}^{y_p+\omega_y} I^2(x,y)}{(2\omega_x+1)\cdot(2\omega_y+1)}; \quad \mu_{J^2} = \frac{\sum_{x=x_p+d_x-\omega_x}^{x_p+d_x+\omega_x}\sum_{y=y_p+d_y-\omega_y}^{y_p+d_y+\omega_y} J^2(x,y)}{(2\omega_x+1)\cdot(2\omega_y+1)} \end{aligned} \tag{3.2}$$

De esta manera, se puede aplicar α y β a la ecuación 2.6 de la siguiente manera para obtener un método robusto a cambios de iluminación (Fig 3.1):

$$\begin{bmatrix} d_x \\ d_y \end{bmatrix} = G'^{-1} \cdot b' \tag{3.3}$$

donde:

$$G' = \sum_{x=x_p-\omega_x}^{x_p+\omega_x}\sum_{y=y_p-\omega_y}^{y_p+\omega_y} \begin{bmatrix} (\alpha I_x(x,y))^2 & \alpha^2 I_x(x,y) I_y(x,y) \\ \alpha^2 I_x(x,y) I_y(x,y) & (\alpha I_y(x,y))^2 \end{bmatrix} \tag{3.4}$$

$$I'_t(x,y) = I(x,y) - \alpha J(x+d_x, y+d_y) - \beta \tag{3.5}$$

$$\begin{aligned} b' &= \sum_{x=x_p-\omega_x}^{x_p+\omega_x}\sum_{y=y_p-\omega_y}^{y_p+\omega_y} \begin{bmatrix} -I_x(x,y) I'_t(x,y) \\ -I_y(x,y) I'_t(x,y) \end{bmatrix} \\ &= \sum_{x=x_p-\omega_x}^{x_p+\omega_x}\sum_{y=y_p-\omega_y}^{y_p+\omega_y} \begin{bmatrix} I_x(x,y)\cdot(\alpha J(x,y) - I(x,y) - \beta) \\ I_y(x,y)\cdot(\alpha J(x,y) - I(x,y) - \beta) \end{bmatrix} \end{aligned} \tag{3.6}$$

3.2. Invarianza a la rotación de la cámara

El método de Lucas-Kanade calcula el flujo óptico de una ventana en una imagen J con respecto de una imagen de referencia I. De acuerdo a la ecuación 2.9, el algoritmo minimiza la diferencia de intensidades entre la ventana en la imagen inicial y la estimada

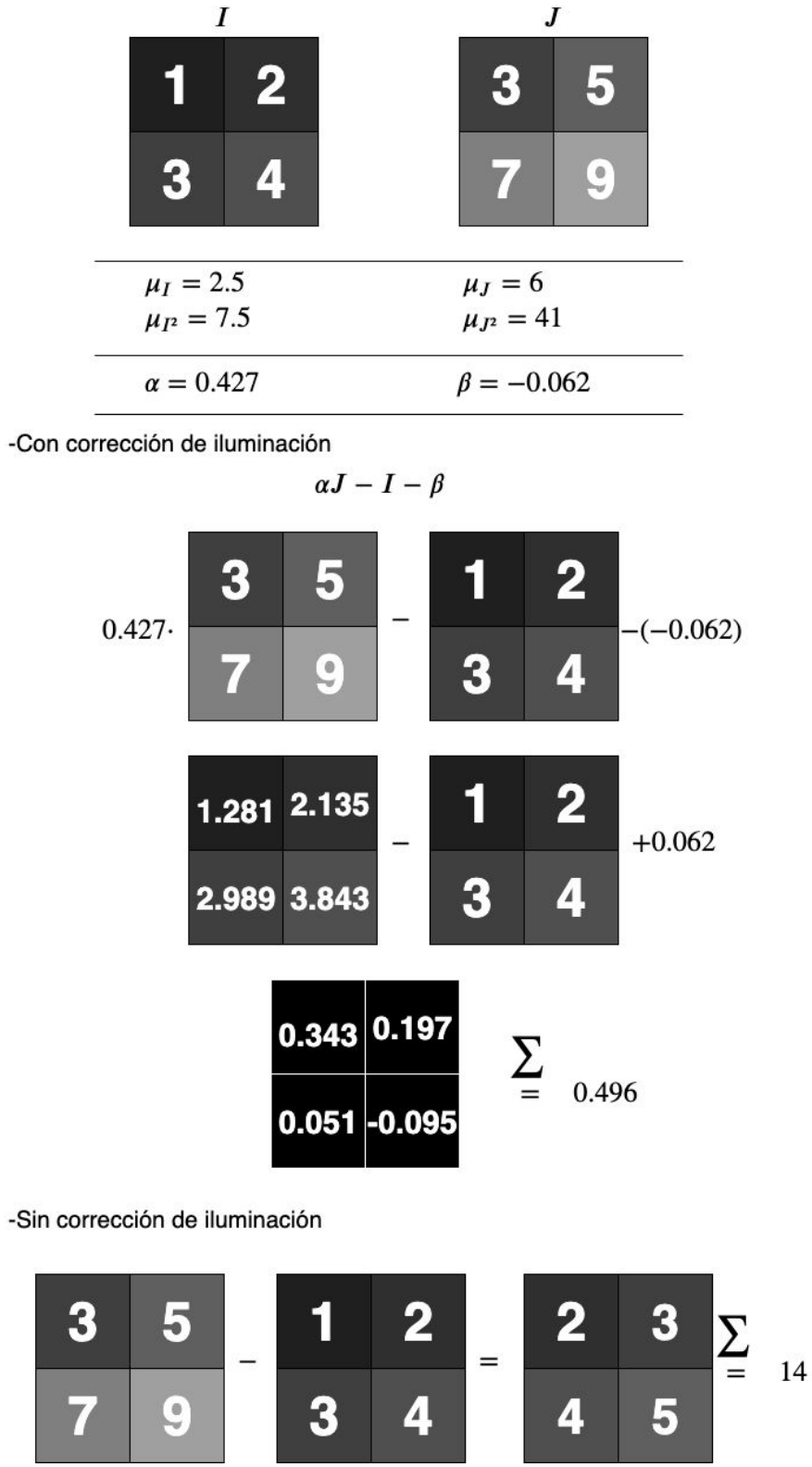


Figura 3.1: Ejemplo del funcionamiento de la corrección visual desarrollada para el calculo de I_t. La imagen J es una versión más iluminada de I de acuerdo a la siguiente expresión: $J = 2I + 1$.

en la nueva imagen. Sin embargo, si la cámara se mueve y sufre una rotación, las ventanas entre la imagen de referencia y la actual pueden no coincidir (Fig. 3.2) y, por tanto, el algoritmo puede no converger o encontrar una solución errónea.

Si se dispone de una estimación del movimiento de la cámara, se puede realizar una corrección de la imagen actual de manera que se compense la rotación. Esto se realiza mediante el cálculo de una homografía que permite mapear los píxeles de la imagen actual con la de referencia. Dicha homografía se puede calcular de la siguiente manera:

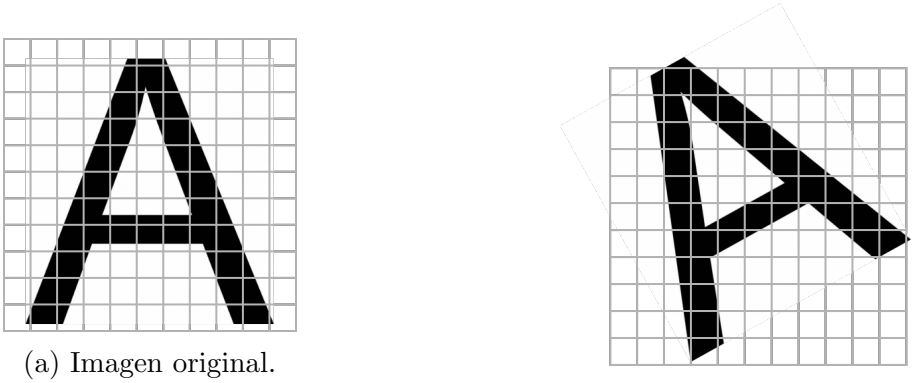
$$^{I}H_{J} = K \cdot {}^{I}R_{j} \cdot K^{-1} \tag{3.7}$$

donde K es la matriz de calibración de I y J y ${}^{I}R_{j}$ la matriz de rotación entre I y J.

3.3. Uso de patrones de píxeles

Cuando se pretende realizar el seguimiento de un número de puntos n entre imágenes consecutivas mediante el método de Lucas-Kanade, se puede observar que el coste computacional del algoritmo puede ser prohibitivo dependiendo de los requisitos de la aplicación, como puede ser la ejecución en tiempo real de un sistema de SLAM. Además del número de puntos, otros factores a tener en cuenta en el coste computacional relacionados con los parámetros del algoritmo son:

- Número de niveles de la pirámide de imágenes L.
- Número máximo de iteraciones $maxIter$
- Tamaño de ventana seleccionado $\omega = (2\omega_x + 1) \cdot (2\omega_y + 1)$.



(a) Imagen original.

(b) Imagen rotada 30º.

Figura 3.2: Ejemplo de cómo la rotación en la imagen afecta a las ventanas. Se observa cómo una rotación de 30º provoca que las ventanas usadas por el algoritmo de Lucas-Kanade sean notablemente diferentes a pesar de ser la misma imagen.

De manera intuitiva, se puede calcular el coste asintótico del algoritmo como:

$$O(n, L, maxIter, \omega) = n \cdot L \cdot maxIter \cdot \omega \tag{3.8}$$

En el apartado 2.2.1 se comentó cómo afectan dichos parámetros a la precisión y eficiencia del algoritmo. Más concretamente se observa que un factor crítico es el tamaño de ventana ω ya que está directamente ligado al número de píxeles que se procesan por iteración de algoritmo. Sin embargo, es importante notar que los píxeles dentro de una ventana cuadrada suelen ser redundantes entre ellos, por lo que en su lugar, se puede procesar un conjunto reducido de los mismos esparcidos en forma de patrón (Figs. 3.3, 3.4) a lo largo de la ventana seleccionada.

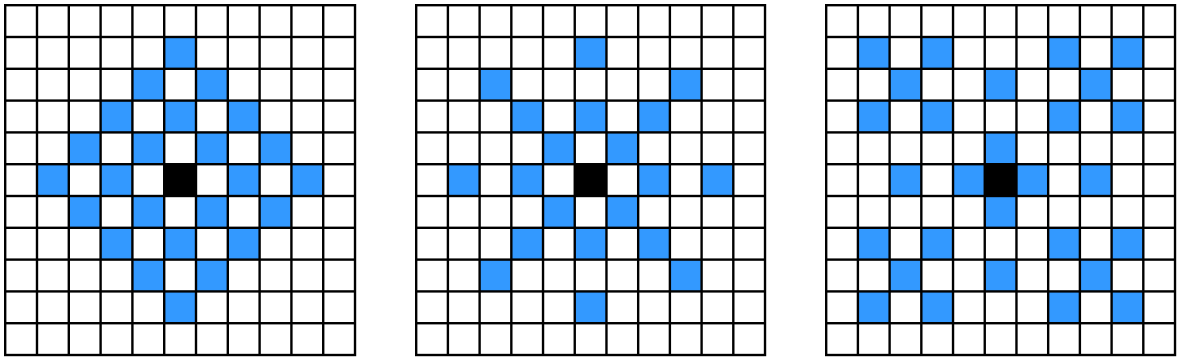


Figura 3.3: Ejemplos de diferentes patrones de píxeles contenidos en una ventana de $(11, 11)$.

3.3.1. Resultados

Para comparar el funcionamiento y precisión de las mejoras aplicadas al método de Lucas-Kanade, se han comparado con la implementación estándar presente en la biblioteca *OpenCV* con los mismos parámetros ($\omega = (11, 11)$, $L = 4$, $maxIter = 10$). Se ha comparado tanto el número de puntos seguidos entre imágenes consecutivas (Tabla 3.1) como el tiempo de cómputo de cada método (Tabla 3.2). En el caso de uso de seguimiento de patrones de pixeles, se ha utilizado el patrón de la izquierda de la Fig. 3.3.

En vista de los resultados extraídos, se comprueba cómo la versión con invarianza a la iluminación consigue seguir, de manera más sostenida, un mayor número de puntos (Figs. 3.5, 3.6, 3.7, 3.8) (incluso ante cambios de iluminación donde el resto de versiones fallan) a costa de duplicar el tiempo de cómputo. Sin embargo, el uso de seguimiento de patrones de píxeles junto con invarianza a la cambios de iluminación consigue seguir robustamente más puntos que la versión de referencia de *OpenCV* con el mismo coste computacional.

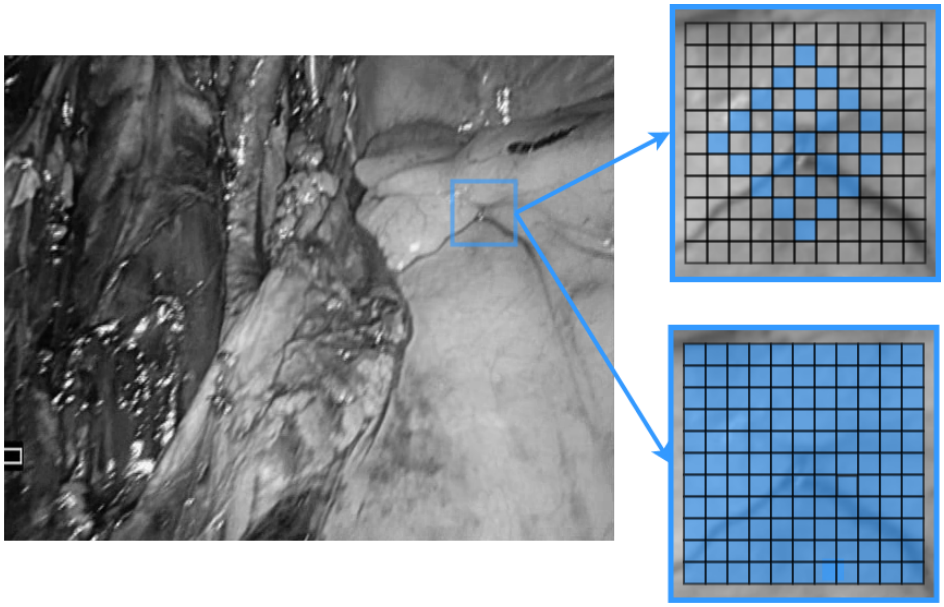


Figura 3.4: Ejemplo entre el uso de una ventana completa (121 píxeles) y un patrón de píxeles (25 píxeles) sobre una imagen.

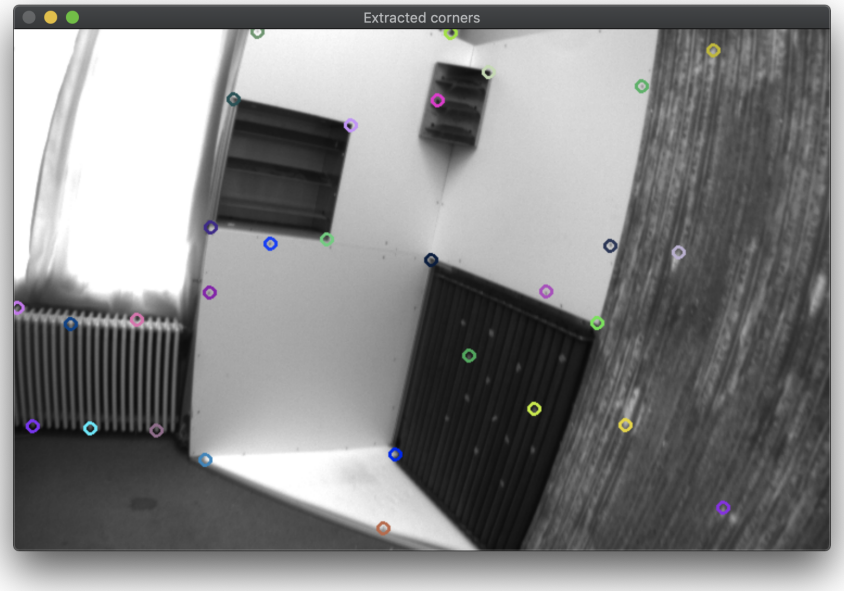
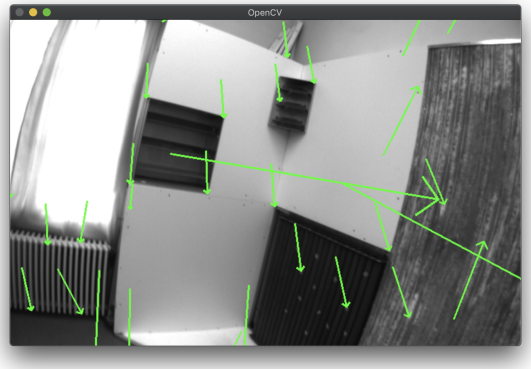
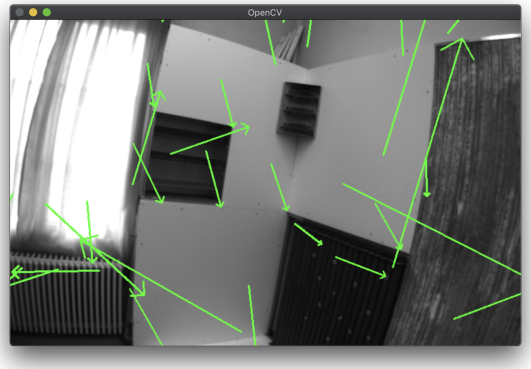


Figura 3.5: Puntos extraídos en la imagen 0 en la prueba de seguimiento de puntos.

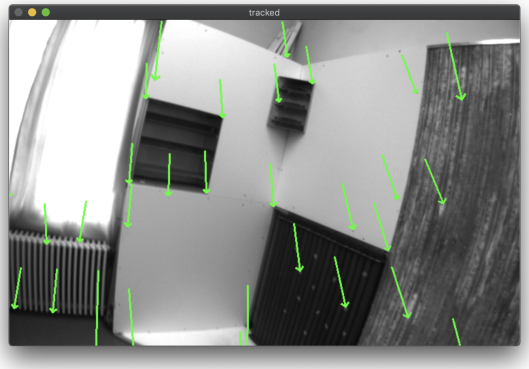


(a) Puntos seguidos en la imagen 3.

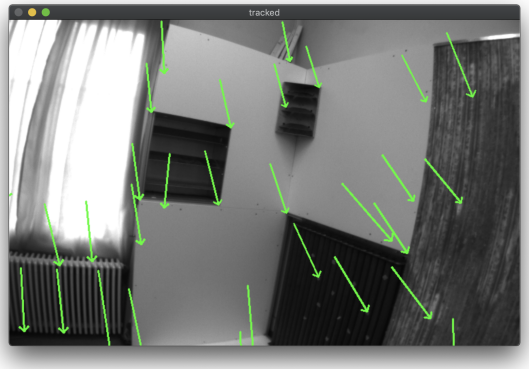


(b) Puntos seguidos en la imagen 5.

Figura 3.6: Seguimiento de puntos con el método de referencia de OpenCV.

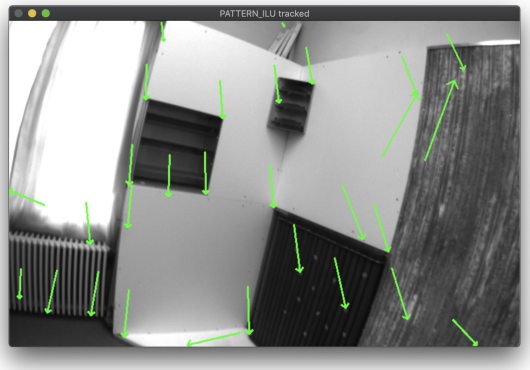


(a) Puntos seguidos en la imagen 3.

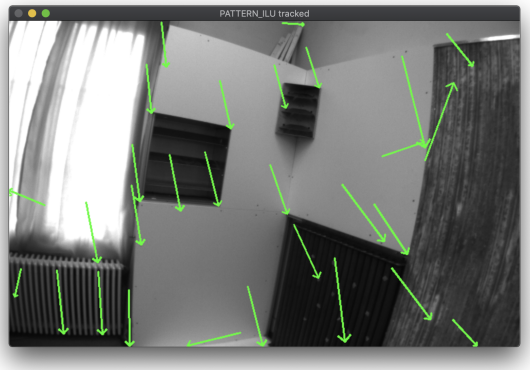


(b) Puntos seguidos en la imagen 5.

Figura 3.7: Seguimiento de puntos con el método invariante a cambios de iluminación.



(a) Puntos seguidos en la imagen 3.



(b) Puntos seguidos en la imagen 5.

Figura 3.8: Seguimiento de puntos con el método invariante a cambios de iluminación y seguimiento de patrones.

Tabla 3.1: Comparación del número de puntos seguidos por cada versión del método de Lucas-Kanade. En la primera imagen (imagen 0) se extraen 30 puntos que se seguirán a lo largo de 5 imágenes consecutivas. En la última imagen se produce un cambio de iluminación.

Imagen	0	1	2	3	4	**5**
OpenCV	30	23	18	17	14	**2**
Invarianza a iluminación	30	25	22	22	20	**17**
Invarianza a iluminación + Uso de patrones de píxeles	30	20	20	20	18	**13**

Tabla 3.2: Comparación entre los tiempos de ejecución de cada versión del método de Lucas-Kanade al seguir 300 puntos. Cada algoritmo se ha ejecutado 100 veces y se ha calculado la media y mediana.

	Mediana (ms)	Media (ms)
OpenCV	1.353	1.474
Invarianza a iluminación	2.230	2.449
Invarianza a iluminación + Uso de patrones de píxeles	1.385	1.484

Capítulo 4

Integración en ORB-SLAM2

ORB-SLAM2, al ser un método indirecto, basa su funcionamiento en la extracción y emparejamiento de puntos de interés para realizar el seguimiento de la pose de la cámara y la construcción del mapa. Más concretamente, usa puntos FAST [7] descritos mediante descriptores ORB [8] extraídos en cada imagen que el sistema recibe, que son emparejados con otros puntos extraídos anteriormente comparando sus descriptores. En otras palabras, se realiza una búsqueda pasiva.

En contraste con escenas de interior o exterior, las secuencias laparoscópicas presentan un entorno desafiante debido a los bruscos cambios de iluminación y a la falta de esquinas naturales que lastran la extracción de puntos y su emparejamiento. Esto provoca que el sistema ORB-SLAM2 tenga dificultades para procesar dicho tipo de secuencias con frecuentes pérdidas de seguimiento de la cámara. El objetivo de este capítulo es integrar en ORB-SLAM2 el método de flujo óptico desarrollado anteriormente, para poder realizar de manera precisa y robusta el emparejamiento y seguimiento de características visuales.

De esta manera, con la inclusión del flujo óptico se pretende realizar una búsqueda activa de características visuales: únicamente en un conjunto determinado de imágenes que aportan gran innovación (*KeyFrames* en ORB-SLAM2) se extraerán puntos FAST, los cuales serán buscados en imágenes posteriores mediante el método de Lucas-Kanade mejorado desarrollado. De esta manera se pretende obtener los siguientes objetivos:

1. Seguimiento más robusto de las características visuales, aumentando su supervivencia a lo largo de imágenes posteriores.

2. Aumento significativo de la precisión al poder trabajar con precisión sub-píxel.

3. Mejora del tiempo de ejecución del hilo de Tracking al no tener que extraer puntos FAST y descriptores ORB en todas las imágenes.

De esta manera, se ha integrado el método de Lucas-Kanade en los hilos de Tracking

y de Local Mapping (Fig. 4.1) para realizar el seguimiento de características visuales sin la necesidad de calcularles un descriptor ORB. Más concretamente, los cambios realizados han sido los siguientes:

- Hilo de Tracking:
  - Inicialización del mapa
  - Seguimiento entre imágenes
  - Reutilización del mapa local
- Hilo de Local Mapping:
  - Extracción de características visuales
  - Triangulación de nuevos puntos del mapa

4.1. Hilo de Tracking

4.1.1. Inicialización del mapa

El objetivo de inicializar el mapa es el de calcular la pose relativa entre dos imágenes o *Frames* de manera que se pueda triangular un conjunto inicial de puntos del mapa. Este proceso debe ser independiente de la geometría de la escena y no debería requerir de la intervención del usuario para seleccionar las dos imágenes iniciales a partir de los cuales se realizará todo el proceso de inicialización.

Los pasos seguidos originalmente en ORB-SLAM2 monocular son los siguientes:

1. Extraer puntos FAST y descriptores ORB en el *Frame* actual F_a y buscar emparejamientos $\mathbf{x}_a \leftrightarrow \mathbf{x}_r$ con los puntos extraídos en el *Frame* de referencia F_r.

2. Computar paralelamente usando *RANSAC* dos modelos geométricos que expliquen la escena: una *homografía* $\mathbf{H}_{ar}$ y una matriz fundamental $\mathbf{F}_{ar}$:

$$\mathbf{x}_a = \mathbf{H}_{ar}\mathbf{x}_r \tag{4.1}$$

$$\mathbf{x}_c^T\mathbf{F}_{ar}\mathbf{x}_r = 0 \tag{4.2}$$

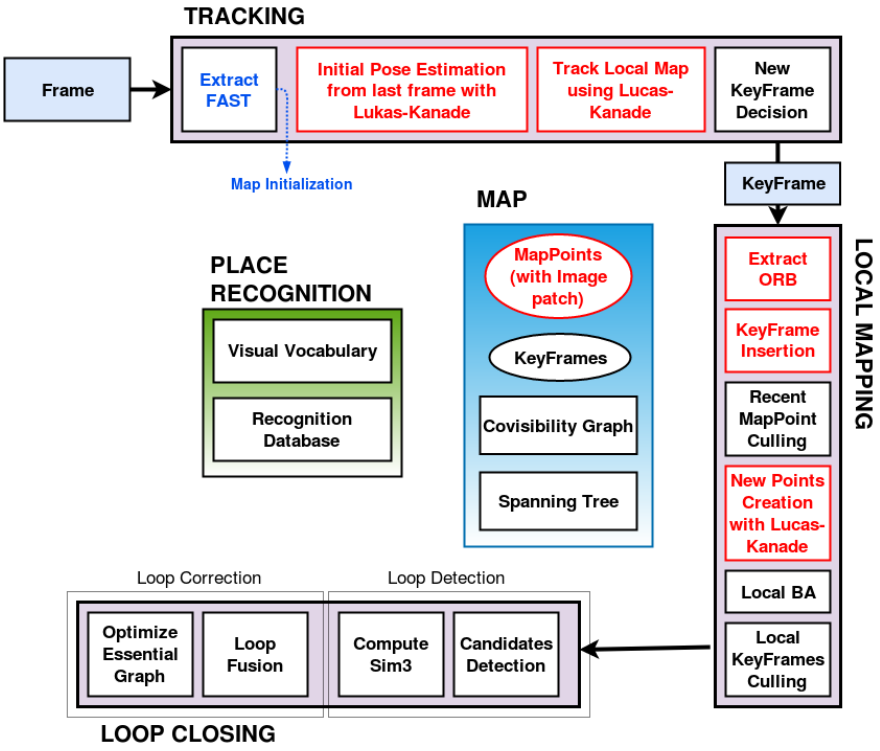


Figura 4.1: Nueva estructura del sistema desarrollado al integrar sobre ORB-SLAM2 el flujo óptico como método para emparejar características visuales.

A cada modelo se le da una puntuación S_H y S_F en función de los errores de reproyección al aplicar el modelo sobre los dos *Frames*. Más detalles en [5].

3. Selección del modelo adecuado: si una escena es planar, la homografía representa correctamente la escena. Sin embargo, si se trata de una escena no planar, la escena esta mejor representada por una matriz fundamental. Para la correcta selección, se usa la siguiente heurística:

$$R_H = \frac{S_H}{S_H + S_F} \tag{4.3}$$

seleccionando la homografía si $R_H > 0{,}45$. En caso contrario, se selecciona la matriz fundamental.

4. Una vez seleccionado el modelo, se reconstruye finalmente el entorno que ven los dos *Frames* y se recupera el movimiento relativo entre ambos, inicializando así el mapa.

Se ha introducido el uso del flujo óptico en el paso 1 de manera que solo se extraen un conjunto de puntos FAST y son seguidos y emparejados en imágenes posteriores mediante el método de Lucas-Kanade modificado desarrollado.

En las pruebas iniciales se observó que el algoritmo de inicialización original de ORB-SLAM2 era demasiado cauto a la hora de inicializar, desechando inicializaciones correctas. Dada la mayor calidad de los emparejamientos producidos por el método de Lucas-Kanade, se decidió disminuir el paralaje necesario a la hora de inicializar puntos del mapa.

Por otro lado, se observó que ante escenas no planares, el algoritmo de inicialización priorizaba la inicialización mediante homografía en muchos casos donde una matriz fundamental habría permitido una rápida y correcta inicialización. De esta manera, para priorizar la inicialización mediante matriz fundamental, se ha modificado la heurística de manera que se intenta inicializar con homografía en caso de que $R_H > 0{,}6$.

4.1.2. Seguimiento entre imágenes

Para estimar la posición de la cámara cuando se recibe una imagen, ORB-SLAM2 extrae y empareja pasivamente características visuales. Como ya se ha comentado, esto provoca el efecto del parpadeo de características visuales y un bajo aprovechamiento de las mismas. Para paliar estos efectos, se ha pasado a realizar una búsqueda activa de características visuales de manera similar a la realizada durante la inicialización del mapa: se siguen en la imagen actual las características visuales extraídas únicamente en el *KeyFrame* más reciente. De esta manera, no es necesario realizar un paso extracción y descripción de características visuales con cada nueva imagen. Por otro lado, y para mejorar la convergencia del método de Lucas-Kanade, se utiliza como semilla inicial la última posición de imagen en la que una característica visual fue observada, teniendo así que calcular flujos ópticos más pequeños.

4.1.3. Reutilización del mapa local

Una de las claves de la mayor precisión de ORBSLAM, en comparación con las técnicas de odometría visual, es su capacidad para emparejar en la imagen actual puntos antiguos del mapa y utilizarlos para mejorar la estimación de la posición y orientación de la cámara. El emparejamiento se consigue proyectando los puntos sobre la imagen actual y realizando emparejamientos guiados mediante descriptores ORB, que proporcionan invarianza a la orientación y la escala.

De nuevo, en este caso se ha utilizado el método de Lucas-Kanade para tal fin. En este caso, la aplicación del mismo no ha sido trivial debido a que no se trata de seguimiento de puntos entre imágenes consecutivas si no de puntos extraídos en cualquier imagen ya procesada por el sistema. Para ello, ha sido necesario guardar la ventana de referencia que utiliza Lucas-Kanade en la que fue observado por primera

vez el punto (apartado 4.2.2). Dado que en este caso los puntos ha seguir pueden haber experimentado grandes translaciones, rotaciones y cambios de escala, se utiliza el método de Lucas-Kanade invariante a rotación mediante el uso de homografías para sintetizar la ventana a buscar en la imagen actual, realizando el seguimiento de manera robusta.

De igual manera, para mejorar la convergencia del método de Lucas-Kanade, se ha obtenido una semilla inicial al reproyectar dichos puntos a buscar sobre la imagen actual. Finalmente, se ejecuta un Bundle Adjustment con objetivo de refinar la pose de la cámara y eliminar posibles seguimientos erróneos de puntos.

4.2. Hilo de Local Mapping

4.2.1. Extracción de características visuales

Dado que no es necesario extraer características visuales en cada imagen, se ha movido esta tarea del hilo de Tracking al hilo de Local Mapping debido a diversos motivos:

- El hilo de Tracking esta sujeto a funcionamiento en tiempo real. Moviendo la fase de extracción y descripción de características visuales al hilo de Local Mapping se consigue reducir el tiempo de ejecución del hilo de Tracking.

- El hilo de Local Mapping procesa exclusivamente *KeyFrames*, por lo que es el lugar idóneo para la extracción de nuevas características visuales. Además, el propio hilo utiliza las nuevas características extraídas para fusionar puntos del mapa duplicados.

Es importante notar que aunque en el hilo de Tracking como en el de Local Mapping ya no se usen descriptores ORB al no ser requeridos por el método de Lucas-Kanade, siguen siendo necesario su cálculo para cerrar bucles.

4.2.2. Triangulación de nuevos puntos del mapa

Para inicializar nuevos en el mapa, ORB-SLAM2 realiza un emparejamiento entre puntos de interés del *keyframe* actual y *keyframes* anteriores, utilizando el descriptor ORB, y triangula los nuevos puntos en 3D. Como ya se ha observado experimentalmente en otros trabajos, la baja repetibilidad del emparejamiento de puntos ORB provoca a que solo un 3 % de los puntos inicializados sean útiles al sistema.

Para mejorar esto, se ha utilizado el método de Lucas-Kanade para realizar el emparejamiento de nuevos puntos de interés para inicializarlos en el mapa. Para ello,

se toman puntos de interés del *KeyFrame* actual que no hayan sido previamente emparejados, y se realiza la búsqueda activa de los mismos en los *KeyFrames* anteriores mediante el método de Lucas-Kanade modificado. Si el emparejamiento se encuentra, se triangula el punto y añade al mapa, guardando junto a él la ventana en la imagen donde fue observado para que posteriormente pueda ser emparejado en el reúso del mapa local.

De esta forma se consigue un mapa formado por puntos que pueden ser seguidos de forma robusta por el método desarrollado, consiguiendo un reaprovechamiento efectivo de la mayoría de puntos del mapa.

Al igual que en el caso de la inicialización del mapa, se observa que la precisión subpíxel obtenida por el método de Lucas-Kanade permite triangular nuevos puntos del mapa con un paralaje menor del usado normalmente por ORB-SLAM2. Por otro lado, y para mejorar la convergencia del método de Lucas-Kanade, se calcula la homografía que relaciona el *KeyFrame* actual con los *KeyFrames* anteriores para obtener mejores semillas iniciales.

Capítulo 5

Resultados experimentales

Para probar los diferentes componentes modificados en ORB-SLAM2 al introducir el método de Lucas-Kanade modificado, se han utilizado diferentes conjuntos de datos o *datasets* de dominio público tradicionalmente usados en la literatura para probar y evaluar las prestaciones de sistemas SLAM. Más concretamente, se han utilizado los siguientes datasets:

1. TUM-D [9]: contiene secuencias de interiores obtenidas con un sensor RGB-D agrupadas en diferentes categorías, para evaluar métodos de SLAM bajo diferentes condiciones de textura, iluminación y estructura.

2. KITTI [3]: contiene secuencias en estéreo grabadas desde un coche en entornos urbanos y de autopista.

3. EuRoC [2]: contiene secuencias tomadas desde un micro-vehículo aéreo (MAV) sobrevolando diferentes habitaciones y entornos industriales.

Finalmente, se ha probado el correcto funcionamiento del sistema en una secuencia laparoscópica (más concretamente, de una linfadenectomía). En ella se observa un entorno desafiante para un algoritmo de SLAM, con superficies deformables y de baja textura, cambios bruscos en la iluminación y zoom de la cámara junto con movimientos agresivos.

Todas las pruebas han sido ejecutadas en un ordenador de sobremesa convencional con una CPU Intel Core i7-3770 @3,4914 GHz x 8 con 8GB de memoria RAM.

5.1. Inicialización del mapa

Se ha probado la calidad de la inicialización del mapa de ORB-SLAM2 usando flujo óptico. Para ello, usando diferentes secuencias de los datasets TUM, KITTI, EuRoC y la secuencia laparoscópica, se ha medido el número de imágenes que necesita

el sistema para inicializar correctamente usando cada imagen de la secuencia como *Frame* de referencia (Fig. 5.1). Además, para comprobar que dichas inicializaciones son correctas, se ha usado el *ground-truth* disponible en la secuencia *Vicon Room 103* del dataset *EuRoC* para medir y comparar los errores de traslación de las inicializaciones (Fig. 5.2). Como se puede observar, nuestro nuevo método de inicialización no solo obtiene resultados más precisos, sino que también es capaz de inicializar con desplazamientos mucho menores de la cámara, gracias a la mayor cantidad y calidad de los emparejamientos obtenidos con el método de Lucas-Kanade modificado.

En el caso de la secuencia laparoscópica, no se presenta ninguna gráfica de comparación ya que la versión de ORB-SLAM2 original era incapaz de inicializar un mapa fiable, mientras que la versión desarrollada es capaz de inicializar de manera correcta a lo largo de toda la secuencia.

5.2. Puntos seguidos

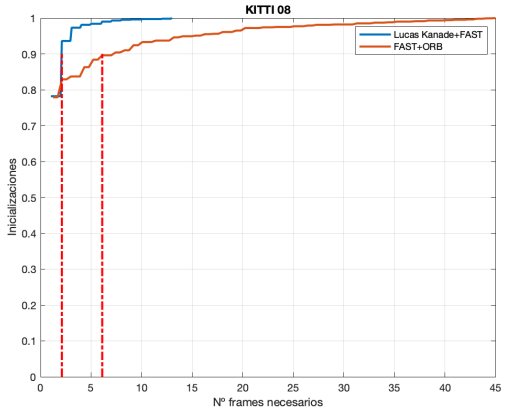
Para evaluar la supervivencia de los puntos seguidos mediante emparejamiento con descriptores ORB o el método de Lucas-Kanade en ORB-SLAM2, se ha medido el número de puntos seguidos en imágenes sucesivas justo después de inicializar el mapa con ambos métodos (Fig. 5.3). En ambos casos, la inicialización del mapa, y por tanto, el conjunto de puntos seguidos, es la misma.

Se observa cómo el algoritmo de SLAM desarrollado basado en el método de Lucas-Kanade es capa de seguir los puntos en la imagen de manera más estable y continuada que mediante el uso de descriptores ORB, consiguiendo una mayor supervivencia de los mismos y, por tanto, un mayor aprovechamiento de los recursos invertidos en extraerlos.

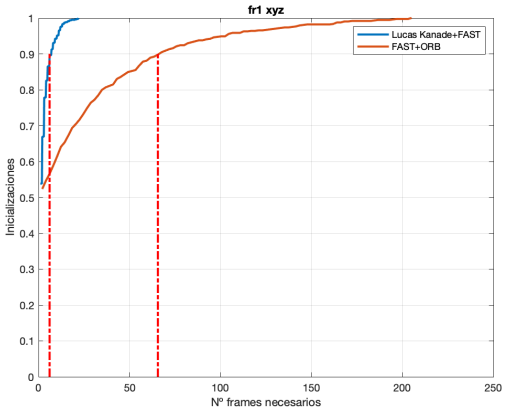
De igual manera, se puede observar el fenómeno del parpadeo de características. Se observa cómo eventualmente, el seguimiento con puntos descritos con ORB aumenta el número de puntos seguidos para, posteriormente, volver a perderlos.

5.3. Triangulación de nuevos puntos del mapa

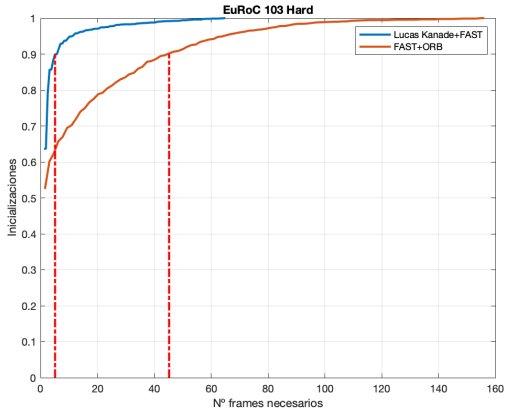
Como se ha comentado ya, el aprovechamiento de los puntos que ORB-SLAM2 extrae en las imágenes tienen una tasa de aprovechamiento muy baja. En este caso, para medir el aprovechamiento entre la nueva versión desarrollada y la implementación original de ORB-SLAM2, se ha medido que porcentaje de puntos disponibles en el *KeyFrame* actual en el momento de triangular nuevos puntos del mapa son efectivamente usados para triangular con éxito nuevos puntos del mapa (Fig. 5.4).



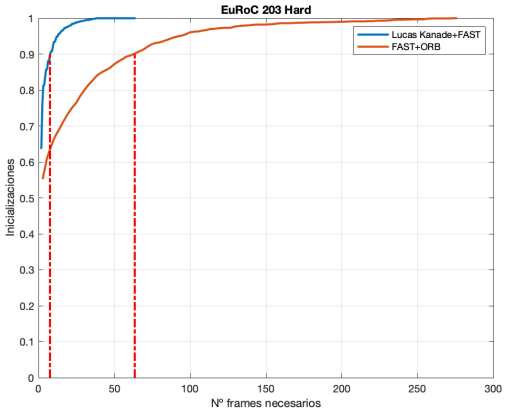
(a) Secuencia *08* del dataset *KITTI*



(b) Secuencia *frg_xyz* del dataset *TUM*.



(c) Secuencia *Vicon Room 103* del dataset *EuRoC*



(d) Secuencia *Vicon Room 203* del dataset *EuRoC*

Figura 5.1: Distribución acumulada de las inicializaciones realizadas a lo largo de cada secuencia, en función del número imágenes que se han necesitado procesar. En azul, nuestro método de inicialización usando emparejamientos obtenidos mediante Lucas-Kanade modificado. En rojo, la implementación original de ORB-SLAM2 con puntos FAST y descriptores ORB.

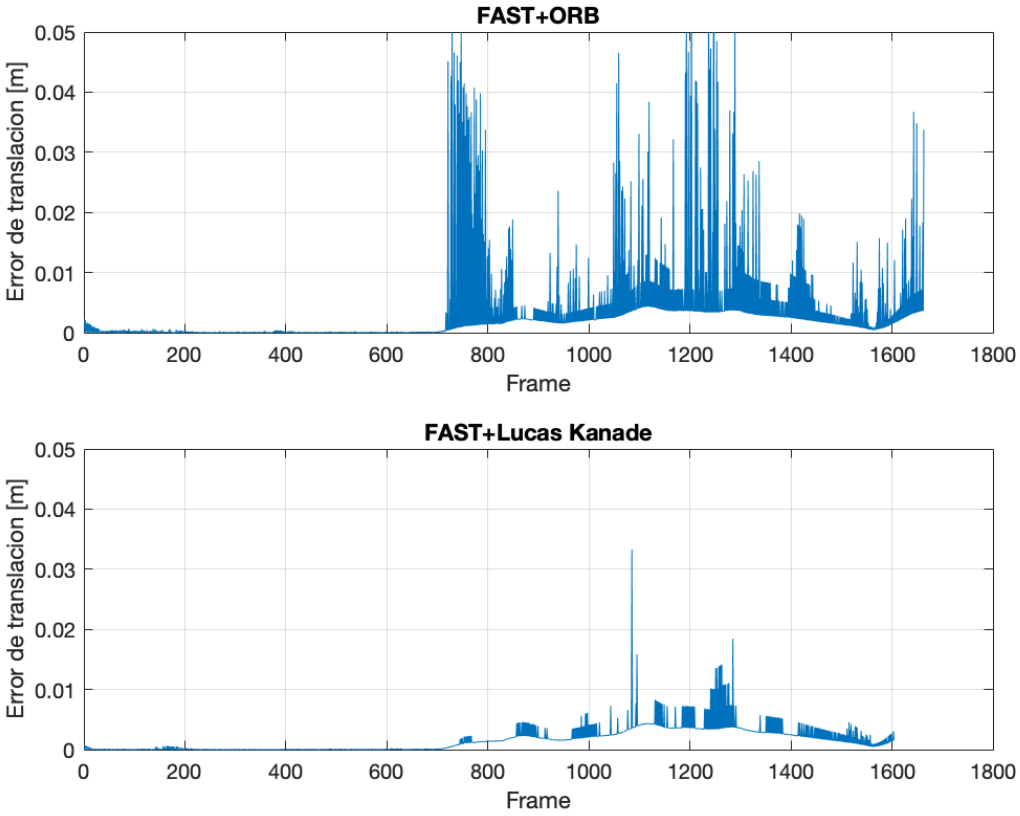
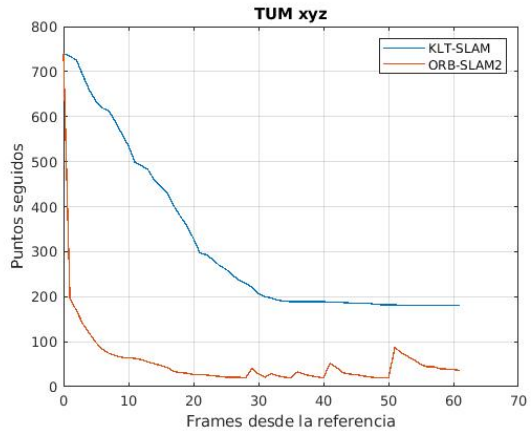
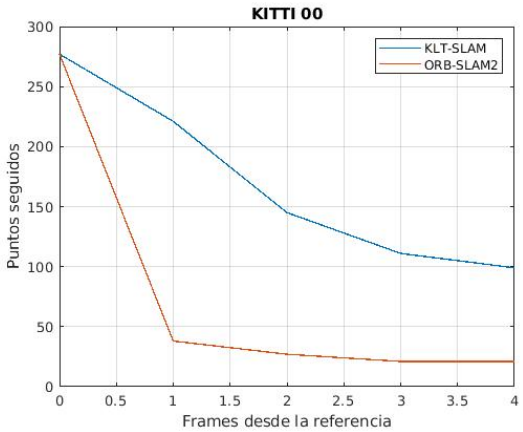


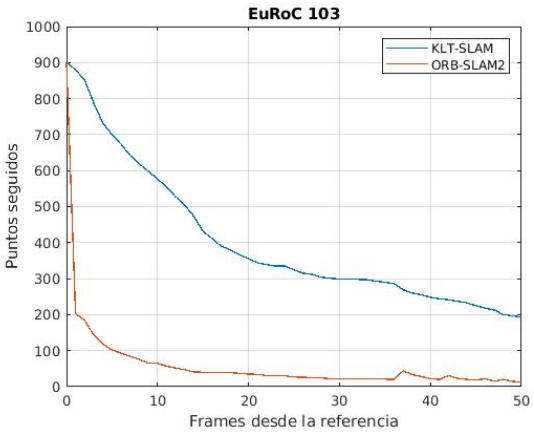
Figura 5.2: Errores de traslación para cada inicialiación a lo largo de la secuencia *Vicon Room 103* del dataset *EuRoC*. Arriba, la implementación original de ORB-SLAM2 con puntos FAST y descriptores ORB. Abajo, nuestro método de inicialización usando emparejamientos obtenidos mediante Lucas-Kanade modificado.



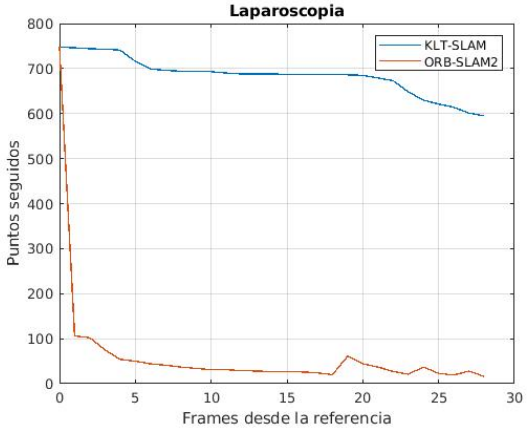
(a) Secuencia *frg_xyz* del dataset *TUM*.



(b) Secuencia *00* del dataset *KITTI*

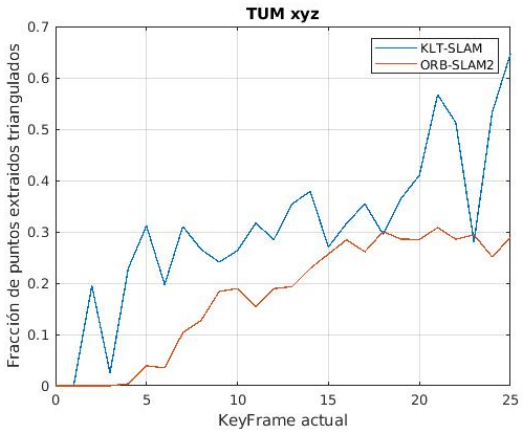


(c) Secuencia *Vicon Room 103* del dataset *EuRoC*

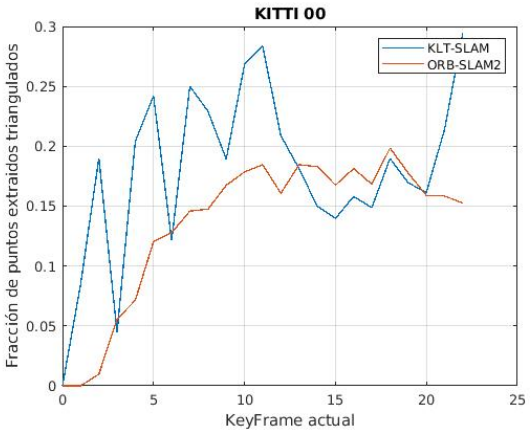


(d) Fragmento de la secuencia laparoscópica.

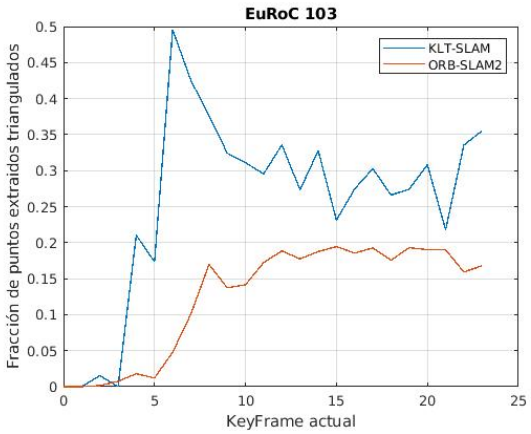
Figura 5.3: Número de puntos seguidos después de la inicialización del mapa. En azul, la version de SLAM desarrollada con el método de Lucas-Kanade. En rojo, ORB-SLAM2.



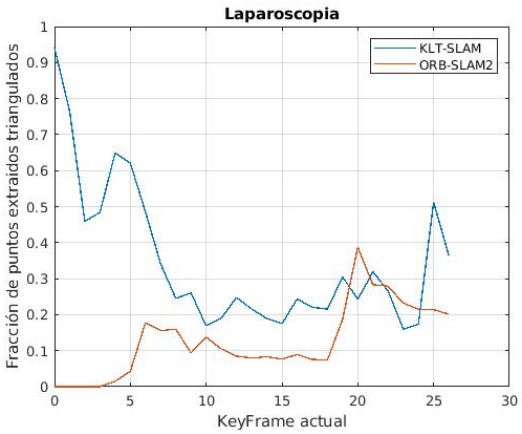
(a) Secuencia *frg_xyz* del dataset *TUM*.



(b) Secuencia *00* del dataset *KITTI*



(c) Secuencia *Vicon Room 103* del dataset *EuRoC*



(d) Fragmento de la secuencia laparoscópica.

Figura 5.4: Fracción de puntos de interés extraído en la imagen que finalmente son correctamente triangulados en puntos del mapa. En azul, la versión de SLAM desarrollada con el método de Lucas-Kanade. En rojo, ORB-SLAM2.

En vista de los resultados, se observa cómo de manera general, el uso del método de Lucas-Kanade para obtener los emparejamientos usados por la triangulación obtiene emparejamientos más precisos, provocando un mayor porcentaje de características visuales correctamente trianguladas y, por tanto, obteniendo un mayor aprovechamiento de las características visuales extraídas.

5.4. Tiempos de ejecución

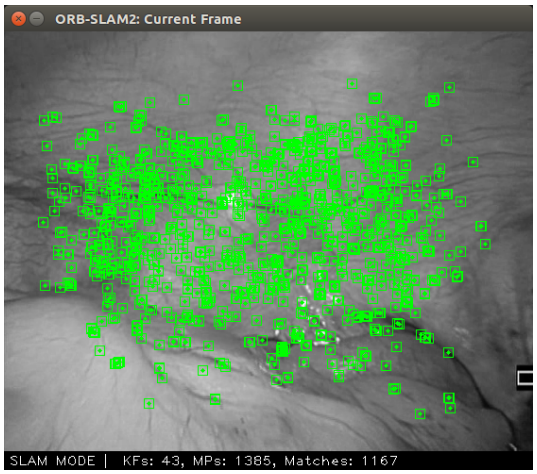
De manera similar a los resultados presentados en [5], se han medido y comparado los tiempos de ejecución de la nueva versión desarrollada con la version original de ORB-SLAM (Tabla 5.1).

Se observa cómo en el hilo de Tracking la estimación de la pose y el reúso del mapa local se vuelven algo más lentos al realizar el emparejamiento de puntos mediante Lucas-Kanade. Por otro lado, en el hilo de Local Mapping, la triangulación de puntos del mapa se vuelve más ligera al obtener de manera general mejores emparejamientos. Sin embargo, el cambio más notable es la reducción del tiempo de ejecución del Bundle Adjustment local, presumiblemente debido a mejores triangulaciones de los puntos y, por tanto, una convergencia más rápida del algoritmo.

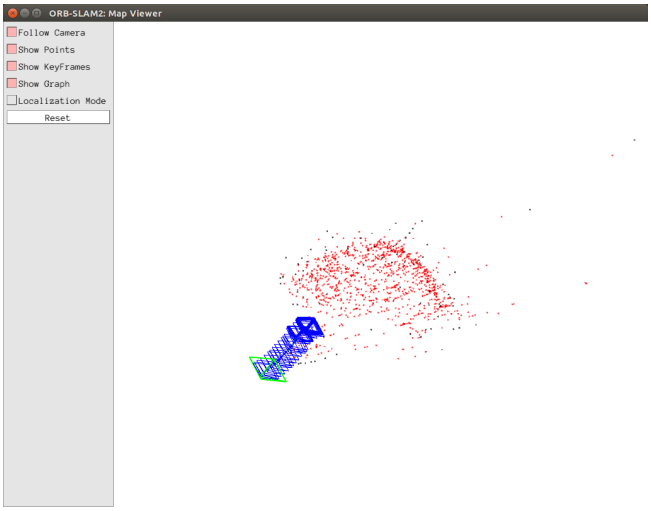
5.5. Secuencia laparoscópica

Finalmente, se ha usado una secuencia laparoscópica realizada con un endoscopio monocular durante una cirugía con un paciente humano. La escena usada se trata de una intervención de linfadenectomía, consistente en extraer los ganglios linfáticos para su posterior estudio y poder determinar si contienen cáncer. La secuencia tiene una duración cercana a las 2 horas, por lo que para probar el sistema desarrollado, se han seleccionado diversos fragmentos de la misma para ser procesados por nuestro sistema. En concreto, se ha seleccionado un fragmento de unos 30 segundos donde se exploran 3 cavidades colindantes (Fig. 5.5).

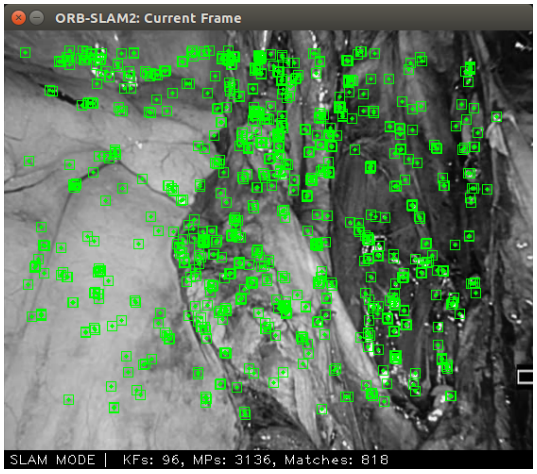
La dificultad de procesar esta escena radica en su naturaleza: en primer lugar, no es una secuencia que estuviese originalmente pensada para ser procesada por un sistema SLAM, por lo que contiene movimientos agresivos y cambios en la iluminación severos, lastrando notablemente el funcionamiento de los sistemas de SLAM tradicionales. Además, se le suma el propio entorno de la cámara al tratarse de superficies deformables en los que no se encuentran de forma natural puntos característicos (como esquinas) que el algoritmo de SLAM necesita para poder funcionar. A pesar de estas dificultades, se observa cómo el sistema desarrollado es capaz de procesar la secuencia, creando un



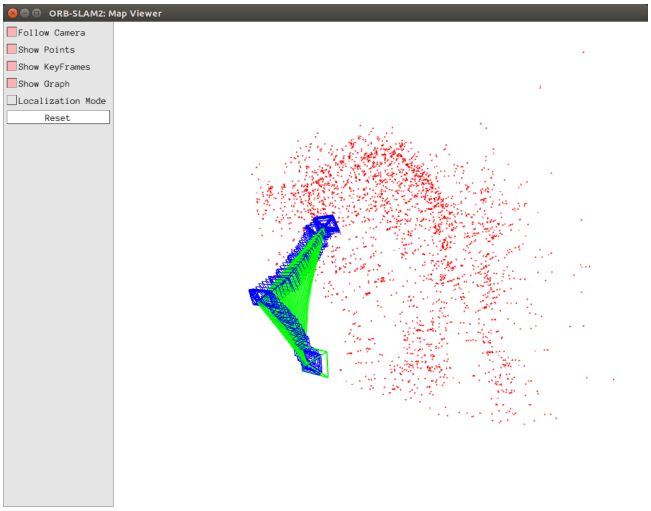
(a) Inicio de la secuencia. Exploración de la cavidad central.



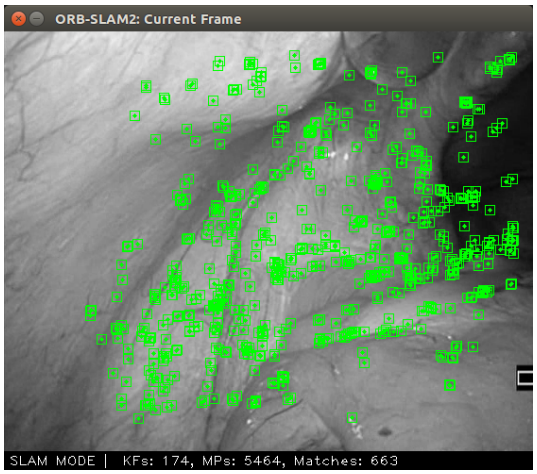
(b) Mapa de la cavidad central.



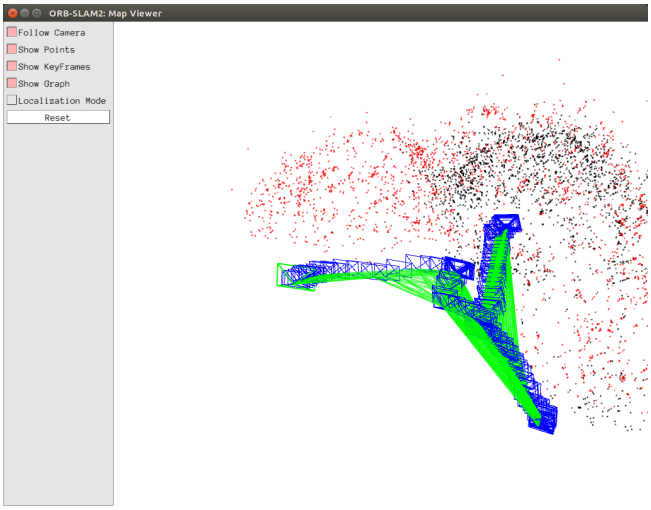
(c) Exploración de la cavidad derecha.



(d) Mapa de la cavidad central y derecha.



(e) Fin de la secuencia. Exploración de la cavidad Izquierda.



(f) Mapa final con las 3 cavidades exploradas.

Figura 5.5: Diferentes fragmentos de la secuencia laparoscópica usada junto con el mapa que el sistema desarrollado construye.

Tabla 5.1: Comparación de los tiempos de ejecución de las diferentes tareas realizadas por ORB-SLAM2 y la versión modificada desarrollada.

Hilo	Operación	ORB-SLAM2 (ms)	KLT-SLAM (MS)
Tracking	Extracción de ORB	11.10	—
	Estimación inicial de la pose	3.38	14.94
	Reúso del mapa local	14.84	23.39
	Total	**30.57**	**38.39**
Local Mapping	Inserción de KeyFrame	10.29	26.34
	Borrado de puntos del mapa redundantes	0.10	0.51
	Creación de nuevos puntos del mapa	66.79	36.60
	Bundle Adjustment local	296.28	86.14
	Eliminación de KeyFrames redundantes	8.07	4.43
	Total	**383.59**	**154.02**

mapa consistente y realizando un seguimiento de la cámara correcto.

El hecho de que el algoritmo de ORB-SLAM2 original no fuera capaz de procesar esta secuencia demuestra que el emparejamiento de puntos basado en el algoritmo de Lucas-Kanade modificado es mucho más eficaz que el emparejamiento basado en descriptores ORB, y permite procesar secuencias laparoscópicas obtenidas de intervenciones reales de manera robusta.

Capítulo 6

Conclusiones

6.1. Conclusiones generales

En el presente proyecto se ha realizado una versión de ORB-SLAM2 a la cual se le ha añadido como método de emparejamiento el uso de flujo óptico con objetivo de obtener una mayor robustez y usabilidad de las características visuales que el propio sistema usa para su funcionamiento. Más concretamente, el resultado final ha sido que el nuevo sistema es capaz de procesar dichas secuencias correctamente mientras que ORB-SLAM2 tiene dificultades para procesarlas dado las situaciones cambiantes de iluminación, entornos deformables y falta de esquinas naturales que el algoritmo de SLAM necesita para extraer características visuales.

Esto es de especial interés dado que la secuencia endoscópica usada en las pruebas se corresponde con una intervención rutinaria en la cual no se tenía en mente obtener una secuencia procesable por un sistema de SLAM, sino que simplemente era la imagen más adecuada para que el cirujano realizase correctamente la cirugía. Ello conlleva que nuestro sistema desarrollado es capaz de procesar dicho tipo de secuencias sin necesidad de alterar el procedimiento o herramientas del equipo médico.

Para poder procesar dichas secuencias tan desafiantes mediante flujo óptico, se ha tomado el conocido método de Lucas-Kanade y se ha mejorado para hacerlo más robusto a cambios de iluminación y rotación. De esta manera, se ha obtenido un algoritmo de flujo óptico que supera en prestaciones a las implementaciones estándar como la de la biblioteca OpenCV, permitiendo de tal modo poder procesar las desafiantes escenas endoscópicas.

Sin embargo, este efecto positivo ha tenido unos efectos colaterales no previstos desde un principio. Por un lado, las reglas usadas originalmente por ORB-SLAM2 para insertar nuevos KeyFrames al mapa se basan en el número de puntos seguidos en el hilo de Tracking. Como ya se ha visto, nuestro nuevo método es capaz de seguir de manera más estable y sostenida dichos puntos, lo que provoca que eventualmente, cuando la

cámara explore una porción nueva de la escena, todos los puntos seguidos se acumulen en una parte de la imagen mientras que la otra queda vacía. Esto ORB-SLAM2 lo entiende como que no se está explorando nada nuevo y, por tanto no hay necesidad de insertar KeyFrames. Sin embargo, la concentración de los puntos del mapa seguidos en una zona de la imagen provoca que la estimación de la pose de la cámara degenere, llegando a provocar que el sistema se pierda.

Por otro lado, se ha observado que el sistema sigue de media una mayor cantidad de puntos lo que incrementa ligeramente los tiempos de ejecución del nuevo sistema, degradando su comportamiento en tiempo real y a largo plazo en secuencias largas, un aumento considerable en el uso de memoria del sistema.

6.2. Trabajo futuro

El trabajo ha mostrado cómo el uso del flujo óptico beneficia en gran medida la obtención de emparejamientos precisos y robustos de características visuales a diferencia que el emparejamiento mediante descriptores ORB. Sin embargo, la inclusión del flujo óptico en ORB-SLAM2 no es sencilla ni trivial debido a que este sistema fue desarrollado teniendo en cuenta la gran cantidad de falsos negativos que el emparejamiento mediante descriptor ORB produce. De esta manera, para poder realizar una completa integración del flujo óptico en ORB-SLAM2 en un trabajo futuro, es necesario modificar en profundidad otras partes del sistema que no se han tenido en cuenta en el presente proyecto:

- La extracción de puntos o características visuales debe ser realizada solo en aquellas zonas de la imagen en las que no se está siguiendo todavía ninguna características visual.

- El mayor ratio de seguimiento y supervivencia de características visuales provoca que ORB-SLAM2 rápidamente se sature, por lo que sería necesario añadir una serie de reglas para limitar el número de puntos seguidos, asegurando que estén igualmente esparcidos sobre la imagen.

- El mayor número de puntos seguidos mediante flujo óptico provoca que ORB-SLAM2 reduzca su tasa de inserción de KeyFrames, provocando que cuando se explore terreno nuevo, sea más propenso a perderse. Es por tanto necesario desarrollar unas reglas de inserción de KeyFrames que tengan en cuenta la distribución de las características visuales en vez de únicamente su número.

– El flujo óptico solo se ha considerado para el seguimiento de características visuales y su posterior triangulación en puntos del mapa. Sin embargo, también podría ser usada para encontrar emparejamientos a la hora de cerrar bucles.

– El uso de patrones de píxeles en el método de Lucas-Kanade es propenso de ser vectorizado para aumentar su rendimiento computacional.

6.3. Gestión del proyecto

El desarrollo del presente proyecto se ha compatibilizado con el curso académico de manera que se han dedicado 15 horas semanales al mismo desde el 1 de Octubre de 2018 hasta el 20 de Septiembre de 2019. De esta manera, se han calculado un total de 765 horas dedicadas al proyecto (Tabla 5.1).

El reparto de tareas (Fig. 6.1) se ha planificado en base a una fase inicial aprendizaje y estudio del flujo óptico y el método de Lucas-Kanade, a partir del cual se ha realizado una implementación base sobre la que se han ido añadiendo las diferentes mejoras desarrolladas. Posteriormente, se realizó la integración en ORB-SLAM2, primero en el hilo de Tracking para, posteriormente, integrarlo en el de Local Mapping.

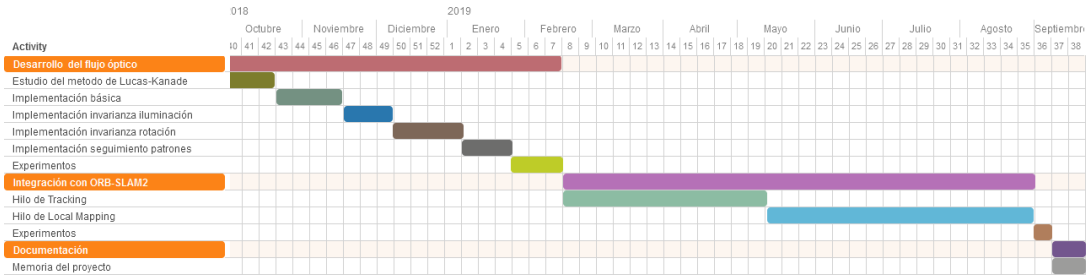


Figura 6.1: Diagrama de Gantt del proyecto.

Tabla 6.1: Desglose de las horas dedicadas al proyecto en las diferentes etapas del mismo.

Etapa	Horas dedicadas
Estudio del método de Lucas-Kanade	45
Implementación básica	60
Implementación invarianza iluminación	45
Implementación invarianza rotación	60
Implementación seguimiento patrones	45
Experimentos	45
Integración hilo de Tracking	240
Integración hilo de Local Mapping	180
Experimentos	15
Memoria del proyecto	30
Total	765

Capítulo 7

Bibliografía

[1] Bouguet, J.-Y., et al. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation 5*, 1-10 (2001), 4.

[2] Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M. W., and Siegwart, R. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research 35*, 10 (2016), 1157–1163.

[3] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research 32*, 11 (2013), 1231–1237.

[4] Lucas, B., and Kanade, T. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop* (1981), pp. 121–130.

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

[6] Mur-Artal, R., and Tardós, J. D. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics 33*, 5 (2017), 1255–1262.

[7] Rosten, E., and Drummond, T. Machine learning for high-speed corner detection. In *European conference on computer vision* (2006), Springer, pp. 430–443.

[8] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. R. Orb: An efficient alternative to sift or surf. In *ICCV* (2011), vol. 11, Citeseer, p. 2.

[9] Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. A benchmark for the evaluation of rgb-d slam systems. 573–580.

[10] Szeliski, R. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

Lista de Figuras

Lista de Tablas