Universidad Zaragoza

Trabajo Fin de Grado

Sistemas de localización y construcción de mapas de alta precisión para realidad virtual y aumentada

High-precision localization and mapping systems for virtual and augmented reality

Autor

Santiago Gil Begué

Director

Juan Domingo Tardós Solano

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

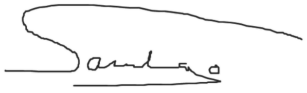
D./Dª. Santiago Gil Begué,

con nº de DNI 72998960L 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, (Título del Trabajo) Sistemas de localización y construcción de mapas de alta precisión para realidad virtual y aumentada,

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

Zaragoza, 28 de agosto de 2017



Fdo: Santiago Gil Begué

AGRADECIMIENTOS

En primer lugar, quisiera dar mis agradecimientos a Juan Domingo Tardós por toda su excelente labor de dirección. No sólo me ha asesorado y resuelto cada una de las tantas dudas que le preguntaba cada semana, sino también Mingo ha sido quien me ha introducido en el mundo de la investigación y en el área de la Inteligencia Artificial y Robótica. *Gracias Mingo.*

Mis agradecimientos también van para Raúl Mur Artal por el desarrollo de ORB-SLAM. Gracias a él he podido tocar en primera persona un *SLAM* y comprender mejor el funcionamiento de estos sistemas. *Gracias Raúl.*

Agradezco también el apoyo de mi familia y de mis amigos. En especial a mis padres Manuel y María Dolores, a mis hermanos María del Mar y Manuel, y a Javier y María. *Gracias familia y amigos.*

Y por último, agradezco también a todos los docentes de la Universidad de Zaragoza de los que me siento afortunado de poder haber aprendido de ellos a lo largo de estos cuatro últimos años. *Gracias.*

Sistemas de localización y construcción de mapas de alta precisión para realidad virtual y aumentada

RESUMEN

La técnica de *localización y mapeado simultáneos*, conocida por sus siglas en inglés *SLAM*, permite la localización libre de deriva de un robot en un entorno desconocido mediante el procesamiento de la información de sensores de a bordo, y sin la utilización de una infraestructura externa como puede ser un sistema de posicionamiento global (*GPS*). ORB-SLAM es el nombre del sistema de localización y construcción de mapas desarrollado por el grupo de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza que es capaz de calcular los movimientos de una cámara con unos 3 a 5 centímetros de error.

Este sistema funciona con sensores de visión a bordo. En sus dos versiones de desarrollo, ORB-SLAM soporta tres tipos de cámara diferentes: cámaras monoculares, estereoscópicas y RGB-D. Durante la presente investigación, se ha añadido el soporte a cámaras con objetivo gran angular, las cuales permiten la visualización de una mayor porción de la escena.

Los puntos ORB con los que trabaja este sistema se extraen con un detector de esquinas tipo FAST. Tal hecho supone que el sistema no proporciona un rendimiento deseable en entornos con ausencia de esquinas, como puede ser el pasillo de un edificio. Para solucionar esta imprecisión del sistema, durante el presente trabajo se ha desarrollado una nueva *feature* denominada *edgelet* que representa a los puntos de contorno de una escena, más comúnmente presentes en todo tipo de escenarios. Se ha demostrado además que la extracción de los *edgelets* de una imagen es un proceso rápido en comparación con la extracción de otros puntos de interés de renombre. Se ha ideado un descriptor que permite un excelente proceso de emparejamiento y seguimiento de estas *features*. Si bien, estas innovaciones requieren un trabajo futuro de profundización para adaptarlas perfectamente en el contexto de *SLAM*.

Se ha instalado y calibrado un sistema de captura de movimiento OptiTrack en el laboratorio 1.07 del edificio Ada Byron de la Universidad de Zaragoza, con el que se ha capturado una secuencia de vídeo, conociendo en todo momento con una precisión sub-milimétrica la pose de la cámara. Se ha desarrollado una demo de realidad aumentada, comparando la experiencia de usuario si se mantiene la localización de la cámara obtenida con un sistema de captura de movimiento, o si se calcula con el sistema ORB-SLAM. Este último consigue una mejor experiencia de usuario y sensación de realismo.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Sistemas de localización y construcción de mapas

Un robot que debe realizar cierta tarea en un entorno desconocido, como puede ser un robot de limpieza en una casa o un dron de vigilancia en una instalación, necesita localizarse dentro de este entorno con el fin de tener éxito en su misión. Puede haber una infraestructura externa para localizar el robot, como un sistema de captura de movimiento o de posicionamiento global (más conocido por sus siglas en inglés, *GPS*). Sin embargo, estos sistemas pueden ser costosos, no estar disponibles, o no proporcionar la precisión requerida.

Es así deseable que la localización se realice mediante el procesamiento de información adquirida por sensores de a bordo. El cómputo del movimiento incremental a partir de la información capturada por estos sensores se denomina odometría. Esta técnica permite recuperar la trayectoria del robot, aunque con un error de acumulación inevitable. Es así una técnica adecuada para la estimación de movimiento a corto plazo, si bien para una operación a largo plazo sobre el entorno se necesita la construcción de un mapa que permita la localización libre de deriva. La construcción del mapa requiere conocer la localización del robot. Surge el problema de *localización y mapeado simultáneos*, conocido por sus siglas en inglés *SLAM*.

El problema de *SLAM* ha sido muy abordado durante las últimas décadas. Se han propuesto múltiples aproximaciones a lo largo de estos años. El grupo de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza ha desarrollado el sistema de localización y construcción de mapas ORB-SLAM. Mediante sensores de visión a bordo, este sistema es capaz de calcular los movimientos de una cámara con unos 3 a 5 centímetros de error.

1.2. Objetivos y alcance del proyecto

Como se ha comentado, el sistema ORB-SLAM funciona con sensores de visión. Actualmente, este sistema soporta tres tipos de cámara diferentes en sus dos versiones de desarrollo. Estos tres tipos corresponden a la cámara monocular, estereoscópica y RGB-D. Como primer objetivo de este proyecto se quiere añadir el soporte a un cuarto tipo de cámara, tratándose éste de la cámara con objetivo gran angular.

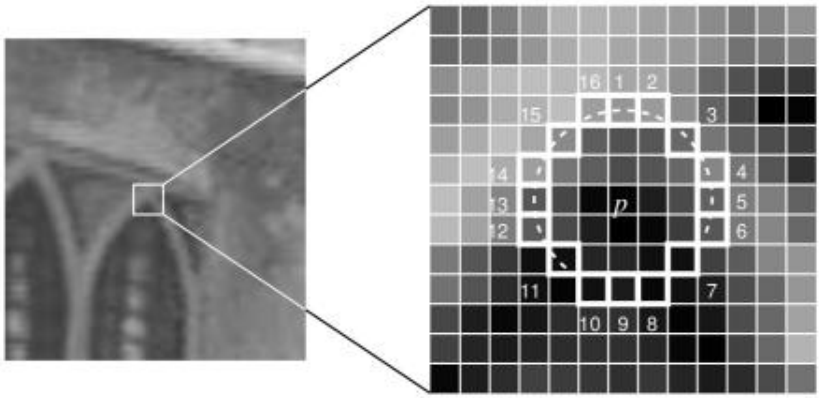


Figura 1.1: El detector FAST verifica si el punto p está rodeado por n puntos seguidos más claros o más oscuros que p, con un umbral de diferencia, Rosten and Drummond (2006).

El mapa generado por el sistema ORB-SLAM se compone de puntos ORB. Estos puntos de interés, conocidos como *features*, se extraen con un detector de esquinas tipo FAST (Figura 1.1). Tal hecho supone que el sistema no proporcione un rendimiento deseable en entornos con ausencia de esquinas, como puede ser el pasillo de un edificio. El segundo objetivo de este proyecto se centra en el estudio e investigación de una nueva *feature* que permita al sistema funcionar en estos entornos. La nueva *feature* corresponde a puntos de contorno, más comúnmente presente en todo tipo de escenarios. Se quiere de esta manera desarrollar la *feature* comentada, así como comprobar su viabilidad en un sistema de *SLAM*. Para ello, se va a integrar sobre el sistema ORB-SLAM de la Universidad de Zaragoza.

Como último objetivo del proyecto se va a instalar y calibrar un sistema de captura de movimiento OptiTrack. Seguidamente, se va a capturar una secuencia de vídeo, de tal manera que se conozca la localización de la cámara en todo momento con una precisión sub-milimétrica con la ayuda de este sistema. Por último, se va a comparar la calidad de la localización de ORB-SLAM respecto a la medida de referencia calculada por un sistema de captura de movimiento en forma de experiencia de usuario en una demo de realidad aumentada.

1.3. Herramientas

Se han utilizado las mismas herramientas que las empleadas en ORB-SLAM para conseguir una integración directa del trabajo realizado. El entorno de desarrollo ha sido CLion mediante una cuenta gratuita de estudiante ofrecida por JetBrains. El listado completo de las herramientas utilizadas para este trabajo se lista a continuación.

- C++: lenguaje de programación base del proyecto.
- OpenCV: librería de estructuras de datos y algoritmos estándar de visión por computador.

- g2o: librería para la optimización de funciones no lineales utilizada en el proceso de *bundle adjustment* en el contexto de *SLAM*.
- OpenGL: librería de gráficos para la demo de realidad aumentada.
- Git + GitHub: software y plataforma de desarrollo para el control de versiones.
- Motive: software para calibrar y utilizar el sistema de captura de movimiento OptiTrack.
- LaTeX: sistema de composición de textos para la documentación del proyecto.
- Harvest: software para la recopilación de esfuerzos y gestión del proyecto.
- Google Scholar: motor de búsqueda para la obtención de toda la literatura académica utilizada como apoyo a la investigación del proyecto.

1.4. Metodología

El proyecto realizado ha estado enfocado en un ámbito de investigación. Ha sido necesaria la lectura de toda la literatura relevante propuesta por el director del proyecto.

Se ha seguido una metodología ágil en todas las fases del proyecto que ha permitido evaluar de forma continua el progreso del mismo. La recopilación de esfuerzos invertidos también ha sido de utilidad en la evaluación del progreso del proyecto.

El código del proyecto se ha gestionado en un repositorio privado dentro del grupo de investigación de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza sobre la plataforma de desarrollo GitHub. Se ha evitado el uso de código con licencia GPL ajeno a la Universidad de Zaragoza.

1.5. Organización de este documento

La investigación desarrollada se expone a lo largo de cuatro capítulos, recogiendo en un quinto las conclusiones del proyecto. El Capítulo 2 se dedica a la presentación de los diferentes tipos de cámara soportados por el sistema ORB-SLAM, y a la introducción del nuevo tipo añadido de cámara con objetivo gran angular. En el Capítulo 3 se detalla la nueva *feature* desarrollada que permite operar con los contornos de una imagen en el contexto de *SLAM*. La integración de los contornos en el sistema ORB-SLAM se recoge en el Capítulo 4. El Capítulo 5 detalla el proceso para capturar una secuencia con OptiTrack, y la experiencia de usuario de la demo de realidad aumentada desarrollada. Finalmente, en el Capítulo 6 se encuentran las conclusiones del proyecto en adición a algunas líneas interesantes de trabajo futuro.

Se recoge en los anexos del presente documento información adicional del proyecto realizado. En el Anexo A se detalla la planificación del proyecto junto a un desglose

cuantitativo de los esfuerzos invertidos en el mismo. En el Anexo B se recoge la información técnica del sistema de captura de movimiento OptiTrack y el proceso seguido para su instalación en el laboratorio de despliegue. También se incluye una breve guía explicativa de los principales casos de uso del sistema. Para terminar, en el Anexo C se recogen las actas de las reuniones celebradas a lo largo del periodo de realización del proyecto.

Capítulo 2

Tipos de cámara

En la primera versión de ORB-SLAM se da soporte al tipo de cámara monocular. En la segunda versión se extiende el soporte a cámaras estereoscópicas y cámaras RGB-D. En el proyecto realizado se ha añadido el soporte para cámaras con objetivo gran angular.

2.1. Cámara monocular

El tipo de cámara monocular es el de menor tamaño, así como el de menor consumo de energía y menor coste económico. Este tipo de cámara requiere una calibración mínima y el tiempo de procesamiento de imágenes es inferior al de otros tipos. Por contra, estas cámaras no pueden observar la escala de la escena capturada. De igual manera, la reconstrucción 3D es más laboriosa.

La cámara monocular puede ser modelada como una cámara *pinhole*, como detallan Hartley and Zisserman (2003), una vez se ha eliminado la distorsión de las lentes. De esta manera, un punto 3D $\mathbf{X_C} \in \mathbb{R}^3$ en coordenadas del sistema de referencia $\mathbf{C}$ de la cámara se proyecta en coordenadas 2D de píxeles $\mathbf{x}$ siguiendo la función de proyección $\pi_m : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ definida en la ecuación 2.1, donde f_x y f_y son las longitudes focales horizontal y vertical, y c_x y c_y las coordenadas horizontal y vertical del punto principal de la cámara.

$$\mathbf{x} = \pi_m(\mathbf{X_C}) = \begin{bmatrix} f_x \frac{X_\mathbf{C}}{Z_\mathbf{C}} + c_x \\ f_y \frac{Y_\mathbf{C}}{Z_\mathbf{C}} + c_y \end{bmatrix} , \qquad \mathbf{X_C} = [X_\mathbf{C}, Y_\mathbf{C}, Z_\mathbf{C}]^T , \qquad \mathbf{x} = [u, v]^T \tag{2.1}$$

2.2. Cámara estereoscópica

Una cámara estereoscópica, más comúnmente conocida por su traducción en inglés *stereo*, se compone de dos cámaras fijas. Idealmente, ambas cámaras están sincronizadas mediante *hardware* para que la captura de imágenes se dispare al mismo tiempo. La profundidad se puede estimar en un solo *frame* encontrando correspondencias entre las imágenes de ambas cámaras. Para facilitar este emparejamiento, las imágenes suelen estar rectificadas de tal

manera que la correspondencia de un píxel en la imagen izquierda se encuentra en la misma fila en la imagen derecha. La calibración de una cámara estereoscópica es más compleja, ya que aparte de la calibración intrínseca de las cámaras hay que calcular la traslación y rotación entre éstas. De esta calibración se obtiene la distancia entre las cámaras, conocida como *baseline* b. La función de proyección de una cámara estereoscópica rectificada $\pi_e : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ con una misma calibración intrínseca de las dos cámaras se define en la ecuación 2.2, donde (u_i, v_i) son las coordenadas en la imagen izquierda, y u_d la coordinada horizontal en la imagen derecha. Dado que las imágenes de ambas cámaras están rectificadas, las coordinadas verticales son iguales, $v_i = v_d$.

$$\mathbf{x} = \pi_e(\mathbf{X_C}) = \begin{bmatrix} f_x \frac{X_\mathbf{C}}{Z_\mathbf{C}} + c_x \\ f_y \frac{Y_\mathbf{C}}{Z_\mathbf{C}} + c_y \\ f_x \frac{X_\mathbf{C} - b}{Z_\mathbf{C}} + c_x \end{bmatrix} , \qquad \mathbf{X_C} = [X_\mathbf{C}, Y_\mathbf{C}, Z_\mathbf{C}]^T , \qquad \mathbf{x} = [u_i, v_i, u_d]^T \quad (2.2)$$

2.3. Cámara RGB-D

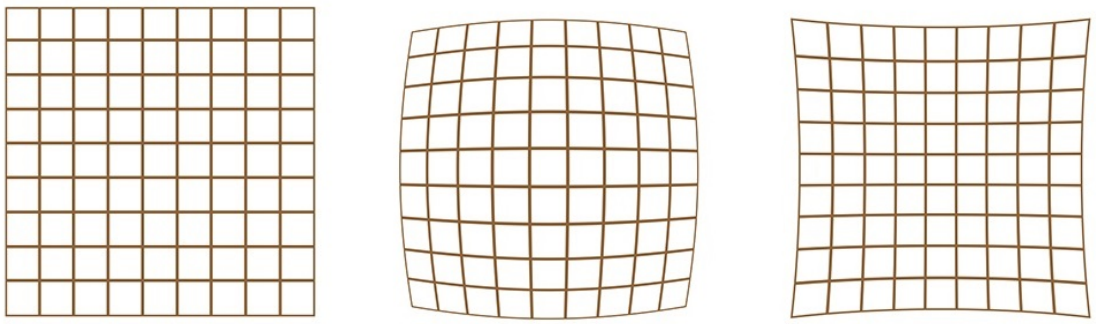
Las cámaras RGB-D son una combinación de una cámara monocular RGB y un sensor de profundidad. Conociendo la calibración intrínseca de la cámara y la calibración extrínseca entre ésta y el sensor de profundidad, se puede registrar una estimación de la profundidad para cada píxel de la imagen capturada por la cámara. Se caracterizan así por su facilidad para reconstruir el mapa 3D. Sin embargo, debido a la naturaleza del sensor de profundidad, su uso está restringido a entornos cerrados y el rango de profundidad está limitado. Además, cuenta con un proceso de calibración más complejo y un mayor consumo energético.

2.4. Cámara con objetivo gran angular

Una cámara con objetivo gran angular es aquella con una distancia focal sustancialmente menor a la de una cámara con objetivo normal, resultando un campo de visión mayor al de la visión humana. Es el caso de las cámaras con objetivo ojo de pez (*fisheye*, en inglés). Este tipo de cámara tiene la ventaja directa de visualizar una mayor porción de la escena que una cámara con un campo de visión estándar.

El modelo de cámara *pinhole* presentado en la sección 2.1 se considera el modelo ideal e intuitivo, mediante el cual las líneas rectas en el mundo real se proyectan también como líneas rectas en la imagen generada por la cámara. El tipo gran angular introduce algunos efectos no deseados que hacen incorrecta la asunción del modelo de cámara *pinhole*. El efecto más evidente es la distorsión radial de barril (Figura 2.1). Esta distorsión radial hace que los

puntos del plano se desplacen de su posición ideal en el modelo de cámara *pinhole*, a lo largo de un eje radial desde el punto principal en el plano de imagen de la cámara gran angular.



(a) Cuadrícula sin distorsión. (b) Distorsión de barril. (c) Distorsión de cojín.

Figura 2.1: Imagen de una cuadrícula sin distorsión (imagen izquierda) comparada con dos conocidos tipos de distorsión radial.

Este efecto de distorsión se puede eliminar por *software*, proceso que se conoce como rectificación de una imagen. Se ha seguido el modelo de distorsión conocido como campo de visión, propuesto por Devernay and Faugeras (2001). Este modelo toma este nombre dado su único parámetro, el campo de visión ω de la cámara *fisheye*. De esta manera, un punto (x_d, y_d) de la imagen distorsionada se rectifica en un punto (x_r, y_r) siguiendo la ecuación 2.3. Para el correcto funcionamiento del modelo de distorsión, las coordenadas de la imagen deben estar normalizadas en el rango $[-1, 1]$. Los valores r_d y r_r se corresponden con la distancia euclídea desde el centro de la imagen distorsionada y rectificada, respectivamente.

$$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \frac{r_r}{r_d} \cdot \begin{bmatrix} x_d \\ y_d \end{bmatrix} , \qquad r_d = \frac{1}{\omega} arctan\left(2 r_r tan\frac{\omega}{2}\right) , \qquad r_r = \frac{tan(r_d \omega)}{2tan\frac{\omega}{2}} \tag{2.3}$$

(a) Imagen distorsionada. (b) Imagen rectificada.

Figura 2.2: Ejemplo de rectificación de una imagen capturada con una cámara *fisheye full-frame* de 16 mm de distancia focal. Tiene así un campo de visión horizontal de aproximadamente $96,73^{o}$. Las líneas rectas distorsionadas en la imagen original son efectivamente rectas en la imagen rectificada, como se puede apreciar claramente en la línea remarcada en color amarillo.

Capítulo 3

Puntos de contorno

En este capítulo se describe todo el proceso para obtener los contornos de una imagen, y los distintos algoritmos que manipulan los puntos de contorno extraídos.

Un contorno se define como un salto en la intensidad de la imagen. De una manera más visual, un contorno puede corresponder a los límites de un objeto, de una textura, incluso de una sombra o un reflejo.

3.1. Cálculo de contornos

En esta sección se explica el algoritmo desarrollado para calcular los contornos de una imagen. Existe ya una implementación del algoritmo de Canny en la librería OpenCV[1] para el cálculo de los contornos de una imagen. Sin embargo, este algoritmo sólo devuelve los contornos en cuestión. En el contexto de *SLAM* no interesa únicamente la obtención de los contornos, sino también de información de los mismos para el cómputo de un descriptor con el que poder realizar posteriormente procesos de emparejamiento. Realizar una segunda pasada a la imagen para calcular los descriptores de los contornos extraídos con Canny no parece una buena idea si éstos se pueden calcular en una única pasada durante la extracción de los contornos. En un contexto de aplicación en tiempo real como es el de *SLAM* donde el tiempo de ejecución es crítico, esto es un motivo suficiente para desarrollar un algoritmo propio que permita en un mismo acto la extracción de contornos y el cómputo de sus descriptores. Además, el algoritmo de Canny incorpora una etapa de segmentación de contornos mediante un umbral con histéresis que no es necesario para el contexto del proyecto.

Para el cálculo de contornos se trabaja con imágenes en escala de grises. Las imágenes en color se pueden convertir de una manera muy sencilla a imágenes en escala de grises aplicando una media aritmética de sus tres canales rojo, verde y azul.

A lo largo de la sección se van a visualizar diferentes figuras ilustrativas del proceso de cálculo de contornos de la imagen expuesta en la Figura 3.1.

[1] `http://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html`

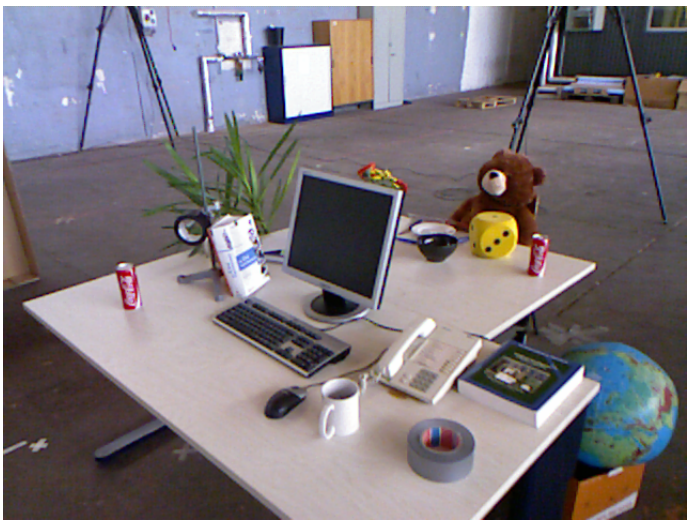


Figura 3.1: Imagen obtenida de la secuencia *freiburg2_desk* de Sturm et al. (2012) sobre la que se va a aplicar el algoritmo de cálculo de contornos.

3.1.1. Eliminación de ruido

El ruido es intrínseco a cualquier imagen digital. Los sensores de una cámara digital capturan el número de fotones incidentes durante el tiempo de exposición de la cámara. Este número de fotones capturados por un sensor se relaciona con la intensidad del píxel correspondiente en la imagen final. El ruido comentado toma origen en este proceso de captura dada la naturaleza aleatoria de la trayectoria de un fotón. Además, los fotones incidentes pueden no golpear a los fotodiodos del sensor, sino a otra parte del chip, aumentando por otra parte la varianza del ruido definido.

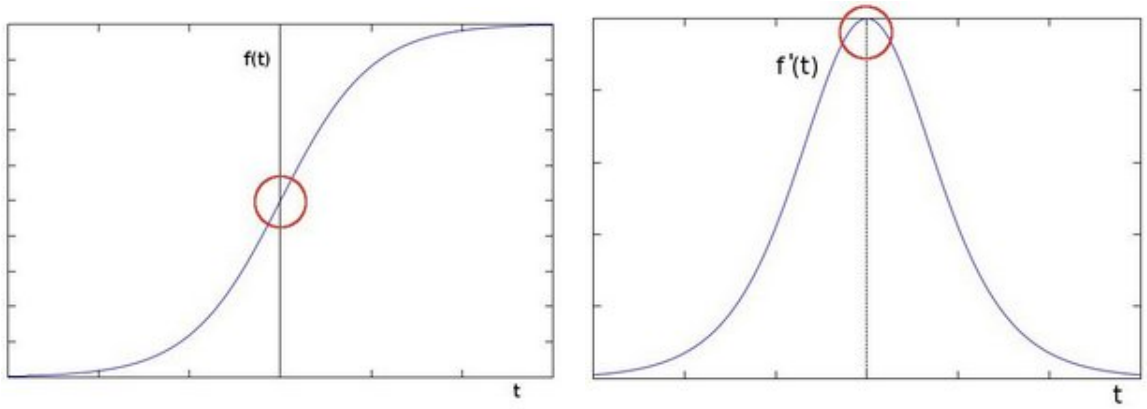
Para la eliminación del ruido se aplica un filtro gaussiano con el objetivo de suavizar la imagen. Este filtro se basa en aplicar sobre la imagen una matriz de convolución cuyos valores se obtienen de una función Gaussiana con desviación estándar σ (ec. 3.1). El nuevo valor de un píxel se obtiene como una función ponderada de sus vecinos. El píxel original recibe la máxima ponderación, mientras que los píxeles vecinos reciben pesos más pequeños conforme su distancia al píxel original se incrementa. Esta ponderación mantiene los contornos de la imagen original a la vez que realiza el efecto de suavizado deseado. En la sección 3.1.5 se visualizan los efectos de aplicar o no este filtro.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.1}$$

3.1.2. Operador de Sobel

Un contorno se define como un salto en la intensidad de la imagen, que es equivalente a un máximo en su primera derivada (Figura 3.2). La primera derivada de la intensidad de la

imagen toma el nombre de gradiente. Para obtener los contornos de una imagen es necesario de esta manera el cálculo de los cambios de intensidad en todos los píxeles de la imagen, lo que es equivalente al cálculo de los gradientes.



(a) Función de la intensidad de la imagen.

(b) Primera derivada de la función de intensidad de la imagen.

Figura 3.2: Un contorno se puede visualizar como un salto en la intensidad de la imagen (izquierda), o como un máximo en su primera derivada (derecha).

El operador de Sobel es un operador discreto de diferenciación que calcula una aproximación del gradiente de una imagen. El funcionamiento de este operador se basa en la convolución de las máscaras de Sobel K_x y K_y (ec. 3.2) a lo largo de toda la imagen I para la obtención de los gradientes horizontales ∇_x y verticales ∇_y (ec. 3.3).

$$K_x = \frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad K_y = \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{3.2}$$

$$\nabla_x = K_x * I\ , \qquad \nabla_y = K_y * I \tag{3.3}$$

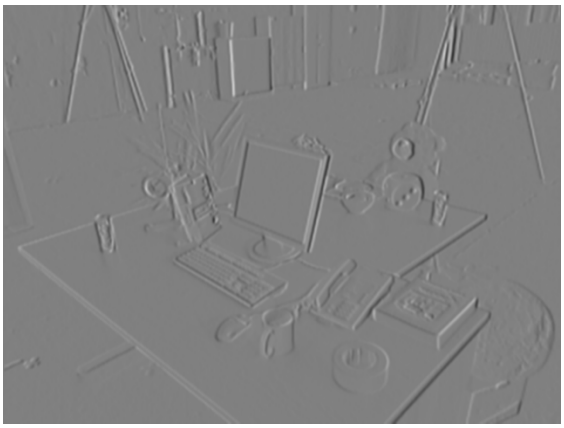
Con el cálculo de los gradientes horizontales y verticales se obtienen los gradientes $\overrightarrow{\nabla f}(x,y)$ (ec. 3.4) en toda la imagen, que se define como la primera derivada de la función de intensidad f. Dado que el gradiente se trata de un vector bidimensional, se puede calcular su magnitud $|\nabla f|$ (ec. 3.5) y su orientación θ (ec. 3.6) de igual forma que con cualquier vector.

$$\overrightarrow{\nabla f}\,(x,y) = (\nabla_x, \nabla_y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) \tag{3.4}$$

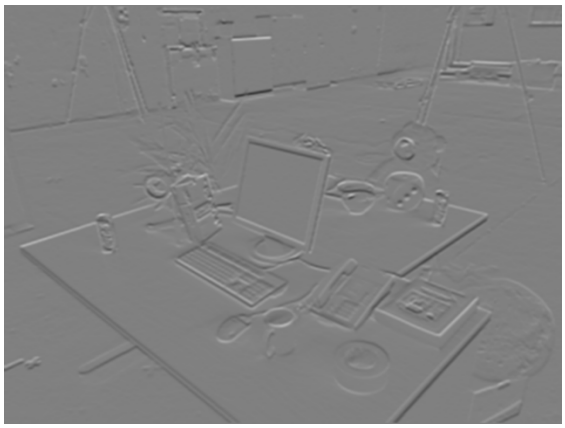
$$|\nabla f| = \sqrt{(\nabla_x)^2 + (\nabla_y)^2} \tag{3.5}$$

$$\theta = atan2\,(\nabla_y, \nabla_x) \tag{3.6}$$

En la Figura 3.3 se presentan los gradientes obtenidos para la imagen de prueba. El cálculo de los gradientes horizontales y verticales mediante la convolución de las máscaras de Sobel tiene en cuenta el sentido del gradiente, por lo que sus valores varían en el rango R [-255, 255]. Para visualizar estos gradientes como se realiza en la Figura 3.3 es necesario un cambio de escala al rango S [0, 255]. Esto se realiza fácilmente mediante la conversión ${}^{S}\nabla_x =^{R} \nabla_x/2 + 128$, y de igual manera para ∇_y. Se consigue así un color grisáceo en zonas de bajo gradiente; un color blanco en píxeles con un alto gradiente positivo, es decir, hacia los semiejes positivos marcados como referencia en las máscaras de Sobel (abajo y derecha); y un color negro en píxeles con un alto gradiente negativo. El cambio de intensidad se dirige desde colores oscuros hacia colores claros. La magnitud del gradiente ya viene escalada en el rango [0, 255]. Su significado es directo, los píxeles con un mayor nivel de blanco se corresponden con contornos más definidos. La orientación del gradiente varía en el rango R [0, 2π], según el ángulo en sentido horario que le diferencia respecto al vector marcado como referencia (hacia la derecha). De nuevo hace falta escalarlo al rango S [0, 255], que se puede realizar mediante la conversión ${}^{S}\theta =^{R} \theta/\pi \cdot 128$.

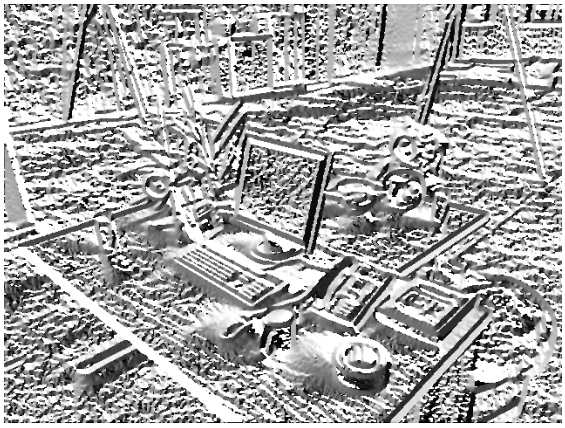


(a) Gradiente horizontal.



(b) Gradiente vertical.

(c) Magnitud del gradiente.



(d) Orientación del gradiente.

Figura 3.3: Gradientes obtenidos de la imagen expuesta en la Figura 3.1 utilizando el operador de Sobel.

3.1.3. Umbralización

Se ha definido un contorno como un salto en la intensidad de la imagen. Sin embargo, es inevitable la existencia de cambios de intensidad, aunque mínimos, a lo largo de una imagen. Por esta razón, conviene añadir el detalle de "considerable" a la definición de salto de intensidad. Se busca, de esta manera, puntos de la imagen que tengan una magnitud de su gradiente considerable. Para cumplir esta condición, se filtran puntos de la imagen con una magnitud del gradiente que no supera un umbral definido.

Se presenta en la Figura 3.4 el histograma de los valores de la magnitud del gradiente de la imagen de prueba. Esta visualización da una idea del umbral a definir a partir del cual un punto es considerado un contorno. En la sección 3.1.5 se exponen los resultados obtenidos en base a diferentes umbrales definidos.

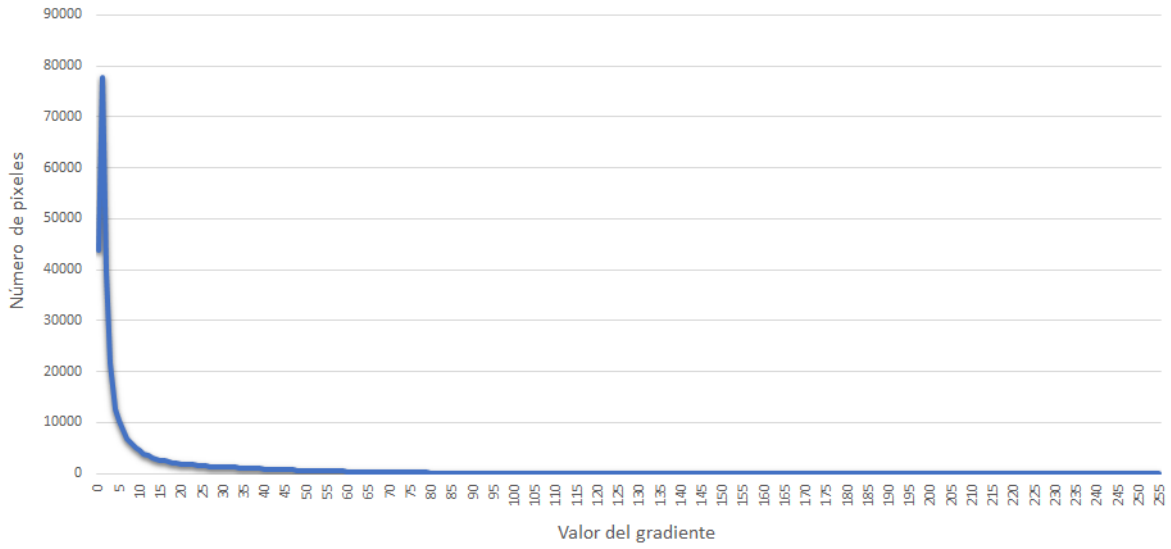


Figura 3.4: Histograma de los valores de la magnitud del gradiente de la imagen expuesta en la Figura 3.1.

3.1.4. Supresión de no máximos locales

Como se aprecia en la Figura 3.3c, todavía no se tienen bien definidos los contornos como una ristra única de píxeles. La última etapa del algoritmo desarrollado consiste en una supresión de los puntos con una magnitud del gradiente no máxima en su dirección del gradiente, obteniendo así los contornos como esta ristra única. Visualizando una ventana 3x3 sobre un píxel A (Figura 3.5), se calcula el módulo del gradiente en B y C interpolando linealmente entre los valores $A3 - A4$ y $A7 - A8$. El píxel A es un punto de contorno si su módulo es mayor que el de B y mayor o igual que el de C. Las parejas de valores sobre las que se interpola dependen de la dirección del gradiente en A.

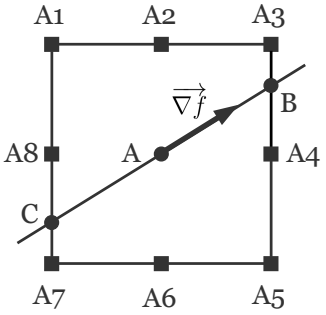
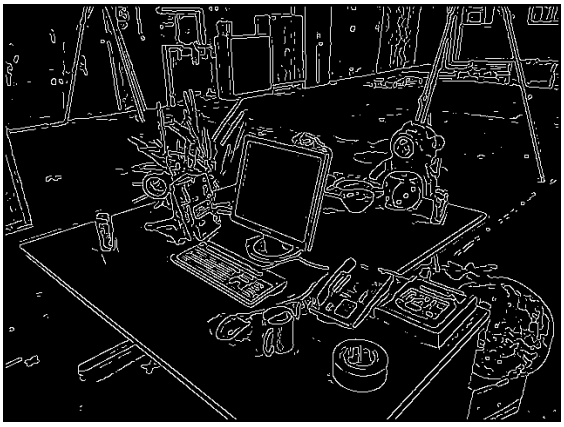
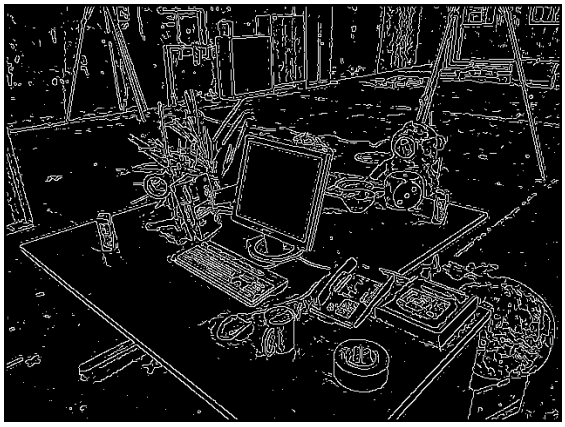


Figura 3.5: Ventana de tamaño 3x3 utilizada para la supresión de puntos no máximos en su dirección del gradiente.

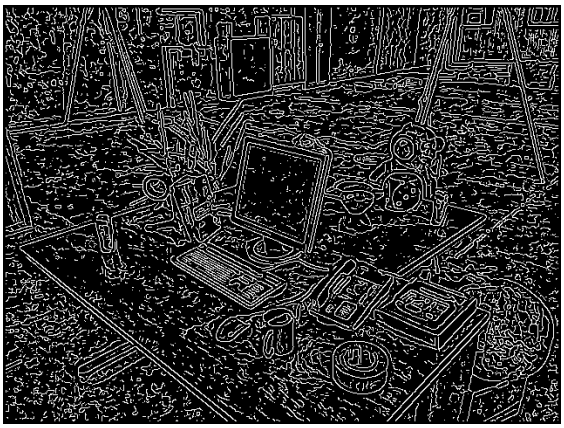
3.1.5. Experimentos



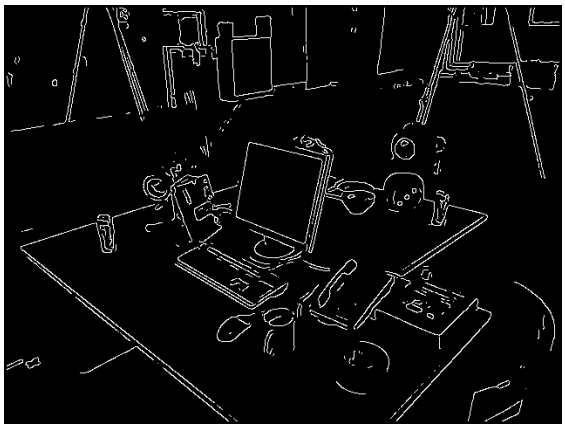
(a) Umbral de 7.5 con filtro gaussiano.



(b) Umbral de 7.5 sin filtro gaussiano.



(c) Umbral de 2 con filtro gaussiano.



(d) Umbral de 32 con filtro gaussiano.

Figura 3.6: Contornos extraídos de la imagen expuesta en la Figura 3.1.

Se han extraído los contornos de la imagen de prueba con diferentes configuraciones del algoritmo presentado (Figura 3.6). En la Figura 3.6b se puede observar el ruido comentado en la sección 3.1.1 si no se aplica el filtro gaussiano. En las otras tres imágenes se ha aplicado un filtro gaussiano con un valor 1,1 de desviación estándar y un tamaño 5x5 de la máscara

de convolución. La Figura 3.6a muestra unos contornos bien definidos al aplicar un umbral de 7.5, siendo éste el pico de la curva presentada en el histograma del gradiente (Figura 3.4). Las Figuras 3.6c y 3.6d añaden y eliminan contornos erróneos, respectivamente, debido a sus umbrales definidos, alejados del pico de la curva del histograma. Se concluye en la elección de la configuración de valores tomada en la Figura 3.6a para la extracción de los contornos.

3.2. Selección de *edgelets*

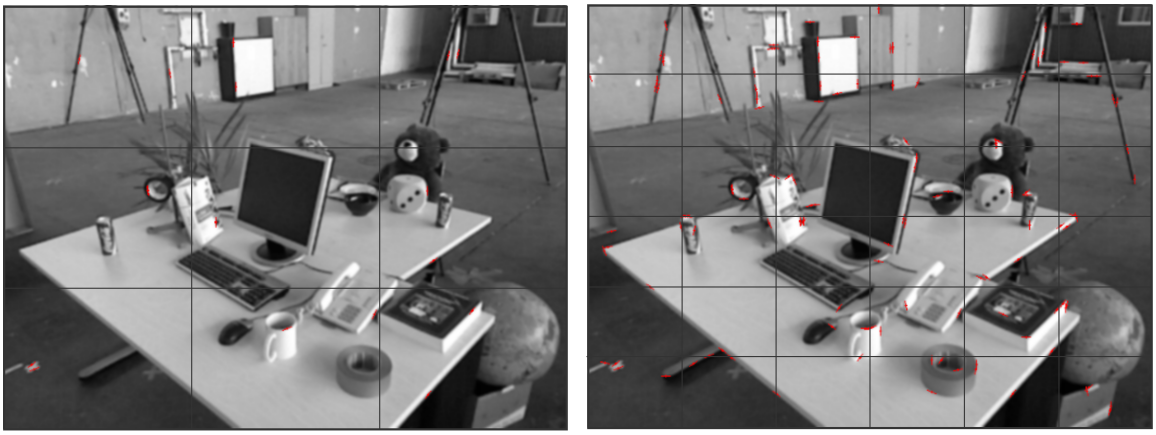
Definido el algoritmo para la obtención de contornos de una imagen, el siguiente paso es utilizarlos en el contexto de *SLAM*. Los sistemas de localización y reconstrucción de mapas que trabajan con métodos indirectos, como Klein and Murray (2007), Mur-Artal et al. (2015) y Mur-Artal and Tardós (2017), estiman la geometría 3D de una escena en base a un conjunto de emparejamientos de puntos de interés desde diferentes posiciones. La idea es aplicar con los contornos el mismo proceso, minimizando un error geométrico en base a sus estimaciones.

Surge un primer problema, y es el coste prohibitivo que implican los procesos de seguimiento y emparejamiento de todos los puntos de contorno extraídos de una imagen. Es necesaria una selección de un conjunto de puntos de contorno más representativos de la imagen. A estos puntos seleccionados se les va a denominar *edgelets*. Se ha adoptado este término de Eade and Drummond (2009). Un *edgelet* es una porción local de un contorno. Esta porción es localmente recta. Se puede visualizar una línea curva como la unión de varios segmentos localmente rectos. Esta representación local de los contornos evita asimismo problemas de seguimiento de contornos en su conjunto, pudiendo presentar, por ejemplo, oclusiones parciales que dificulten la representación de un contorno de extremo a extremo.

Se van a seleccionar los puntos de contorno con más magnitud de gradiente. Éstos, sin embargo, suelen concentrarse en una zona de la imagen. Por esta razón, se van a seleccionar los puntos de más gradiente de cada una de las celdas de un tablero superpuesto a la imagen (Figura 3.7). Se consigue así una uniformidad en la selección de los *edgelets*. Dentro de una celda, se va a dividir la circunferencia en secciones iguales. Para cada sección se va a seleccionar el *edgelet* con más gradiente − si existente − cuya dirección se corresponda con esta porción de ángulo. Se consigue de esta manera la selección de *edgelets* pertenecientes a diferentes contornos. El número de celdas del tablero c y el número de divisiones de la circunferencia d va a acotar el número de *edgelets* seleccionados; como máximo se seleccionarán $c \cdot d$ *edgelets*.

3.2.1. Pseudocódigo del algoritmo

Se presenta el Algoritmo 1 para la extracción de contornos de una imagen y la selección de sus *edgelets*. Como se ha comentado anteriormente, ambas tareas se realizan en la misma pasada de la imagen.



(a) Tablero de 3x3 celdas y un división de la circunferencia en dos secciones.

(b) Tablero de 6x6 celdas y una división de la circunferencia en cuatro secciones.

Figura 3.7: *Edgelets* seleccionados (en color rojo) de la imagen expuesta en la Figura 3.1.

Algoritmo 1 Extracción de contornos y selección de *edgelets*

Entrada: Imagen I.
Salida: Contornos, edgelets y magnitud y orientación de los gradientes de I.

$contornos \leftarrow \{\}$
$edgelets \leftarrow \{\}$
{Suavizado de la imagen: sección 3.1.1}
$I \leftarrow FiltroGaussiano(I)$
{Calcular los gradientes de la imagen (módulo y orientación): sección 3.1.2}
$|\nabla f|, \theta \leftarrow CalcularGradientes(I)$
para todo *píxel* $(i,j) \in I$ **hacer**
  {Umbralización del gradiente: sección 3.1.3}
  si $|\nabla f|(i,j) < UMBRAL$ **entonces**
    continuar
  fin si
  {Supresión de no máximos locales: sección 3.1.4}
  si $|\nabla f|(i,j)$ *no es un máximo local en su dirección* $\theta(i,j)$ **entonces**
    continuar
  fin si
  $contornos \leftarrow (i,j)$
  {Selección uniforme de edgelets: sección 3.2}
  si $|\nabla f|(i,j)$ *es máximo en su sección del grid* **entonces**
    $edgelets \leftarrow edgelets \setminus \{anterior\ máximo\ en\ su\ sección\}$
    $edgelets \leftarrow (i,j)$
  fin si
fin para
devolver $contornos, edgelets, |\nabla f|, \theta$

3.2.2. Evaluación respecto a otras *features*

Se va a evaluar el tiempo de ejecución del cálculo de los puntos de contorno y sus descriptores. Estas mediciones se van a comparar con el algoritmo de Canny, así como con otros detectores de puntos de interés, como son ORB, SURF y SIFT. Para la extracción de

estas *features* se ha trabajado con la versión 3.3 de la librería OpenCV. Se ha utilizado un ordenador Intel(R) Core(TM) i7-3770 CPU @ 3.40 GHz con 8 núcleos, 8192 KB de memoria cache y 8 GiB de memoria RAM. Para reducir la varianza entre mediciones, se ha ejecutado cada algoritmo un número total de cincuenta mil ejecuciones. La extracción de las *features* se ha realizado sobre la imagen de prueba expuesta en la Figura 3.1.

De los resultados presentados se concluye que el tiempo de cómputo es independiente del número de *edgelets* extraídos. Asimismo, el algoritmo desarrollado consigue ejecutarse en la mitad de tiempo que el algoritmo de Canny de OpenCV. Las comparaciones con los detectores de puntos también son positivas, mejorando incluso los tiempos del detector ORB.

Tablero	**Secciones**	**Nº *edgelets***	**Tiempo de ejecución**	
			Alg. desarrollado	**Alg. Canny**
3x3	4	36	5,03 ms	10,23 ms
4x4	5	80	4,98 ms	10,14 ms
5x5	8	181	4,96 ms	10,25 ms
10x10	8	463	5,03 ms	10,08 ms

Tabla 3.1: Tiempos de ejecución para la extracción de *edgelets* utilizando el algoritmo desarrollado y el algoritmo de Canny de OpenCV.

Umbral FAST	**Nº puntos ORB**	**Tiempo de ejecución**
140	36	5,92 ms
120	95	6,30 ms
100	208	7,08 ms
20	500	10,36 ms

Tabla 3.2: Tiempos de ejecución para la extracción de puntos ORB.

Umbral		**Nº puntos SIFT**	**Tiempo de ejecución**
Contraste	**Contornos**		
0.20	1	22	94,42 ms
0.15	2	108	76,90 ms
0.10	3	257	90,27 ms
0.07	4	417	102,35 ms

Tabla 3.3: Tiempos de ejecución para la extracción de puntos SIFT.

Umbral del Hessiano	**Nº puntos SURF**	**Tiempo de ejecución**
13000	34	19,30 ms
8000	94	20,01 ms
5000	192	28,71 ms
1500	499	41,85 ms

Tabla 3.4: Tiempos de ejecución para la extracción de puntos SURF.

3.3. Emparejamiento de *edgelets*

Dados dos *edgelets* e_1 y e_2 vistos en dos imágenes I_1 e I_2, respectivamente, es necesaria la definición de un valor de distancia entre ambos para decidir si corresponden al mismo punto de contorno. Para ello, hay que definir primero el descriptor de un edgelet.

3.3.1. Descriptor de un *edgelet*

Un descriptor de una *feature* codifica información interesante del mismo y actúa como huella digital, de tal manera que permite diferenciar la *feature* de otras. Por ejemplo, ORB utiliza una modificación del descriptor BRIEF invariante a rotación como explican Rublee et al. (2011). Este descriptor almacena una ristra de valores binarios resultado de comparaciones entre pares definidos de píxeles vecinos. Se utiliza para la descripción de puntos de interés, como pueden ser las esquinas de un objeto. El descriptor propuesto de un *edgelet* almacena la siguiente información.

- Magnitud del gradiente: $|\nabla f|$
- Dirección del gradiente: θ
- Valor de intensidad medio: $\bar{f}$

3.3.2. Distancia entre dos *edgelets*

La distancia entre dos descriptores va a medir el grado de diferencia entre ellos. Los valores que almacena el descriptor de un *edgelet* se miden en unidades diferentes, por lo que se va a utilizar la distancia adimensional de Mahalanobis. Aunque las variables del descriptor no son totalmente independientes, se va a hacer la aproximación de que no hay relación de dependencia entre ellas. Así, la distancia d entre dos *edgelets* e_1 y e_2 sigue la ecuación:

$$d(e_1, e_2) = \sqrt{\left(\frac{|\nabla f|_1 - |\nabla f|_2}{\sigma_{|\nabla f|}}\right)^2 + \left(\frac{\theta_1 - \theta_2}{\sigma_\theta}\right)^2 + \left(\frac{\bar{f}_1 - \bar{f}_2}{\sigma_{\bar{f}}}\right)^2} \tag{3.7}$$

Las desviaciones estándar se han ajustado experimentalmente, tomando los valores $\sigma_{|\nabla f|} = 25$, $\sigma_\theta = 10^o$ y $\sigma_{\bar{f}} = 15$. Se considera que los dos *edgelets* e_1 y e_2 corresponden al mismo punto de contorno si $d(e_1, e_2) < \chi^2_{0,95,3} = 7{,}8147$.

3.4. Seguimiento de *edgelets*

Se quiere realizar el seguimiento de *edgelets* a lo largo de una secuencia de imágenes. En esta sección se explica cómo realizar este seguimiento entre dos imágenes con una asunción de proximidad, sin tener ninguna estimación de la profundidad de los *edgelets*.

3.4.1. Búsqueda local

Partiendo de la base de que las dos imágenes tienen una *baseline* pequeña, esto es, han sido tomadas desde posiciones próximas, se puede realizar una búsqueda local de los *edgelets*. Esta búsqueda consiste en replicar la posición de un *edgelet* de la primera imagen en la segunda, y explorar en una zona de píxeles vecinos. Se puede definir una menor área de exploración a mayor asunción de proximidad entre las dos imágenes. Un *edgelet* de la primera imagen se emparejará con aquel punto de contorno en la segunda imagen con el que mantenga la menor distancia entre sus descriptores. Se puede dar el caso de no encontrar un emparejamiento si ningún punto supera el test de χ^2 comentado en la sección 3.3.2.

Esta exploración induce ocasionalmente emparejamientos erróneos [1], y de manera más frecuente, emparejamientos desplazados a lo largo del contorno [2 y 3] (Figura 3.8).

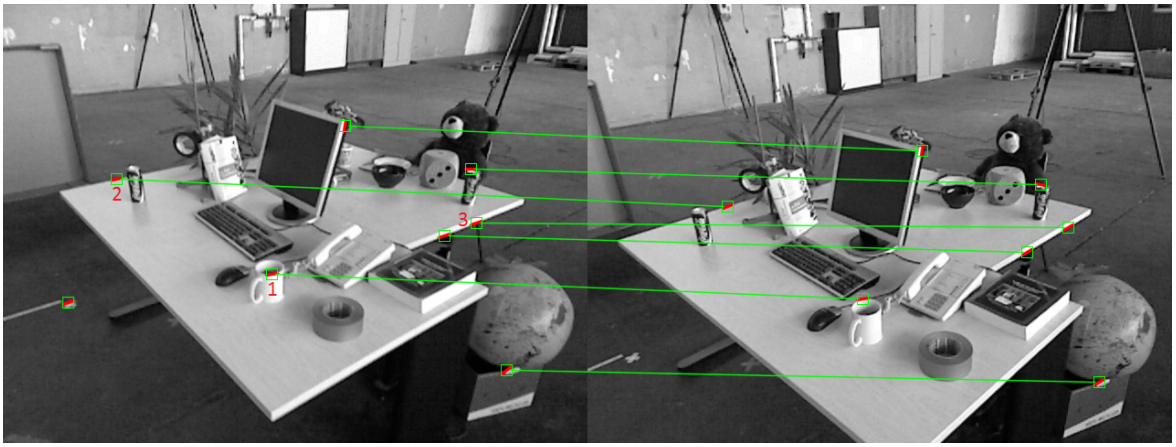


Figura 3.8: Seguimiento de un conjunto de *edgelets* entre dos imágenes, realizando una búsqueda local en una ventana de 61 x 61 píxeles.

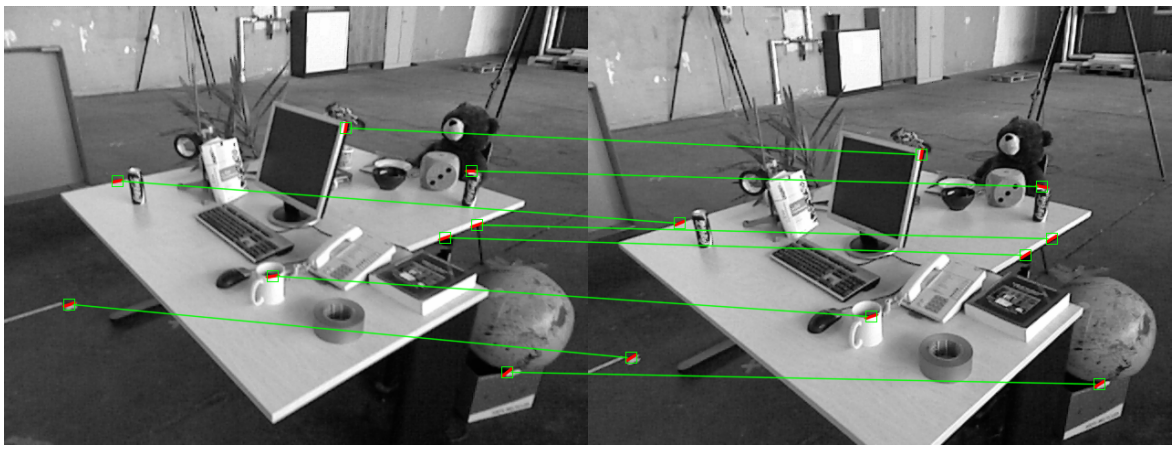


Figura 3.9: Seguimiento de un conjunto de *edgelets* entre dos imágenes, realizando una búsqueda local en un segmento de 61 píxeles sobre la línea epipolar.

3.4.2. Búsqueda local y restricción epipolar

Si se tiene una estimación de las posiciones de las dos cámaras, se puede añadir información a la búsqueda mediante la restricción epipolar (Figura 3.10). La línea unión de la posición $\mathbf{x}_1$ del *edgelet* en la primera imagen y su centro óptico $\mathbf{c}_1$ define el rayo de proyección sobre el que se encuentra el *edgelet* 3D $\mathbf{X}$ a una profundidad desconocida. El plano formado por esta línea y el centro óptico $\mathbf{c}_2$ de la segunda cámara intersecta al plano imagen de esta segunda cámara en una línea denominada epipolar. La proyección del *edgelet* 3D sobre esta imagen, y por tanto el emparejamiento con el *edgelet* 2D de la primera imagen, reside en esta línea epipolar. La búsqueda local sobre la línea epipolar consigue una mejora en los emparejamientos de los *edgelets* (Figura 3.9).

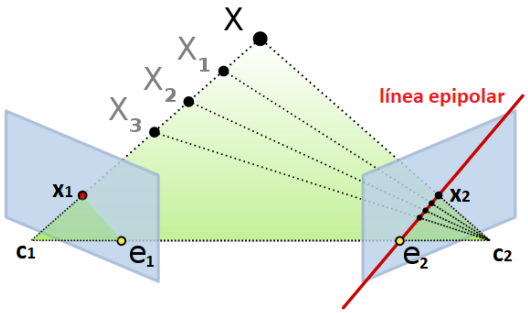


Figura 3.10: Restricción epipolar en la búsqueda de un *edgelet*. En rojo se marca la línea epipolar en la que reside el *edgelet* $\mathbf{x}_2$ buscado.

A continuación, se detalla una breve notación convencional de elementos que va a persistir hasta el final del documento, Szeliski (2010). Los puntos 2D (coordenadas en píxeles en una imagen) se pueden denotar usando un par de valores $\mathbf{x} = (x, y)^T \in \mathbb{R}^2$, o alternativamente mediante coordenadas homogéneas $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{w})^T \in \mathbb{P}^2$. En este espacio proyectivo 2D $\mathbb{P}^2$ se consideran equivalentes los vectores que difieren sólo en escala. Un vector homogéneo $\tilde{\mathbf{x}}$ se puede convertir a un vector no homogéneo $\mathbf{x}$ dividiendo por su último elemento $\tilde{w}$:

$$\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{w})^T = \tilde{w}(x, y, 1)^T \tag{3.8}$$

Los puntos homogéneos cuyo último elemento es $\tilde{w} = 0$ se denominan puntos ideales o puntos en el infinito, y no tienen una representación no homogénea equivalente.

Las rectas 2D también se pueden representar utilizando coordenadas homogéneas $\tilde{\mathbf{l}} = (a, b, c)^T$. Se puede normalizar esta representación, de tal manera que $\mathbf{l} = (n_x, n_y, d)^T = (\mathbf{n}, d)$, con $||\mathbf{n}|| = 1$. En este caso, $\mathbf{n}$ es el vector normal perpendicular a la recta y d es la mínima distancia al origen. La ecuación de una recta con esta representación es:

$$\tilde{\mathbf{x}}^T \cdot \tilde{\mathbf{l}} = ax + by + c = 0 \tag{3.9}$$

Dadas las posiciones de las dos cámaras, se puede calcular la matriz fundamental F_{12} que relaciona los puntos de la primera imagen en la segunda. La línea epipolar $\mathbf{l}_2$ en la segunda imagen de la posición $\mathbf{x}_1$ del *edgelet* en la primera imagen sigue la ecuación 3.10. Nótese la representación en coordenadas homogéneas para el cumplimiento de la ecuación.

$$\tilde{\mathbf{l}}_2^T = \tilde{\mathbf{x}}_1^T \cdot F_{12} \tag{3.10}$$

3.5. Proyección de *edgelets*

Un *edgelet* en 3D $(\mathbf{X}_M, \overrightarrow{V}_M)$ con posición $\mathbf{X}_M$ y orientación $\overrightarrow{V}_M$ en coordenadas del mundo M se quiere proyectar sobre una imagen 2D. Esta imagen ha sido capturada con una cámara de coeficientes intrínsecos K, donde f es la longitud focal y c el centro óptico.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.11}$$

Se denotan $\mathbf{R}$ y $\mathbf{t}$ a los parámetros extrínsecos de la cámara, es decir, a las matrices de rotación y traslación que transforman de un sistema de coordenadas 3D en el mundo a un sistema de coordenadas 3D de la cámara. La proyección π del punto $\mathbf{X}_M$ del *edgelet* en coordenadas del mundo a coordenadas en píxeles de la cámara (u, v) se define como:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K\,[\mathbf{R}|\mathbf{t}] \begin{bmatrix} X_M \\ Y_M \\ Z_M \\ 1 \end{bmatrix}, \qquad \mathbf{x} = \pi(\mathbf{X}_M)\,, \qquad \mathbf{X}_M = [X_M, Y_M, Z_M]^T\,, \qquad \mathbf{x} = [u, v]^T \tag{3.12}$$

El valor s es un factor de escala arbitrario. Nótese de nuevo el manejo de coordenadas homógeneas en esta ecuación, Szeliski (2010).

La proyección de la orientación del *edgelet* se realiza mediante la proyección de dos puntos pertenecientes a la recta (ec. 3.13). Para mantener consistente el sentido de la orientación del *edgelet* proyectado, ambos puntos tienen que pertenecer al mismo lado del plano imagen. Bien $\mathbf{X}_M + \overrightarrow{V}_M$ ó $\mathbf{X}_M - \overrightarrow{V}_M$ cumple esta condición respecto a $\mathbf{X}_M$, si no ambos.

$$\pi(\overrightarrow{V}_M) = \pi(\mathbf{X}_M + \overrightarrow{V}_M) - \pi(\mathbf{X}_M) \quad \vee \quad \pi(\overrightarrow{V}_M) = \pi(\mathbf{X}_M) - \pi(\mathbf{X}_M - \overrightarrow{V}_M) \tag{3.13}$$

3.6. Triangulación de *edgelets*

Se denota P a la matriz 3x4 de proyección de la cámara, equivalente a $K\,[\mathbf{R}|\mathbf{t}]$. Sean dos *edgelets*, en coordenadas homogéneas, $e_1 : \{\tilde{\mathbf{x}}_1 = (u_1, v_1, 1)^T,\ \theta_1\}$ y $e_2 : \{\tilde{\mathbf{x}}_2 = (u_2, v_2, 1)^T,\ \theta_2\}$

pertenecientes a dos imágenes I_1 e I_2 con matrices de proyección P_1 y P_2, respectivamente. Se conoce que ambos *edgelets* corresponden al mismo punto de contorno, por lo que se quiere triangular para obtener su posición y orientación 3D.

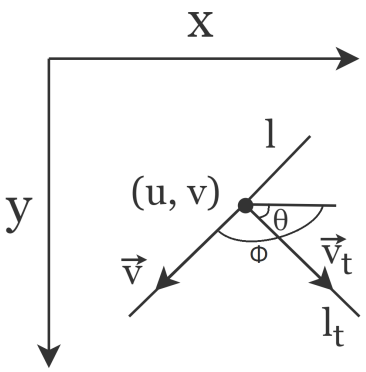


Figura 3.11: Localización y orientación de un *edgelet* 2D.

Se presenta la Figura 3.11 para visualizar una serie de valores que se van a utilizar a continuación. El vector $\overrightarrow{v_t}$ es el vector unitario del gradiente del *edgelet*. En coordenadas homogéneas, éste toma el valor de $\overrightarrow{v_t} = (cos(\theta),\ sin(\theta),\ 0)^T$. El vector $\overrightarrow{v} = (cos(\phi),\ sin(\phi),\ 0)^T$ es el vector director del *edgelet*, con $\phi = \theta + \frac{\pi}{2}$.

Se tiene que el producto vectorial de dos puntos, en coordenadas homogéneas, define la ecuación de la recta que pasa por ambos. De esta manera, tratando un vector como un punto en el infinito, $\tilde{\mathbf{l}} = \tilde{\mathbf{x}} \times \overrightarrow{v}$ y $\tilde{\mathbf{l}}_t = \tilde{\mathbf{x}} \times \overrightarrow{v_t}$.

El *edgelet* 3D $e : (\mathbf{X}_M, \overrightarrow{V}_M)$ se proyecta sobre $\mathbf{l}_1$ y $\mathbf{l}_2$ en las imágenes I_1 e I_2, respectivamente. Por ello, el *edgelet* se encuentra en el plano M_i que conforman el centro óptico de la cámara i y la proyección $\mathbf{l}_i$ sobre ésta, Szeliski (2010).

$$M_i = \tilde{\mathbf{l}}_i^T \cdot P_i\ , \qquad i = \{1, 2\} \tag{3.14}$$

Un plano M se define por cuatro valores, $M : (a, b, c, d)$, siendo $\mathbf{n} : (a, b, c)$ el vector normal del plano y d la mínima distancia al origen. El *edgelet* 3D se encuentra en ambos planos de las dos cámaras, es decir, en la intersección de ellos. Conociendo que el vector director de la línea intersección entre dos planos es el producto vectorial de sus vectores normales, se puede calcular la orientación del *edgelet* mediante:

$$\overrightarrow{V}_M = \mathbf{n}_1 \times \mathbf{n}_2 = (\tilde{\mathbf{l}}_1^T \cdot P_1)_{[a_1,b_1,c_1]} \times (\tilde{\mathbf{l}}_2^T \cdot P_2)_{[a_2,b_2,c_2]} \tag{3.15}$$

La posición $\mathbf{X}_M = (X_M, Y_M, Z_M)$ del *edgelet* se encuentra sobre los dos planos M_1 y M_2. También se encuentra sobre el plano M_{1t}, correspondiente a $\tilde{\mathbf{l}}_{1t}^T \cdot P_1$. De esta manera, se puede obtener la posición resolviendo el sistema de ecuaciones:

$$\begin{bmatrix} a_1 & b_1 & c_1 \\ a_{1t} & b_{1t} & c_{1t} \\ a_2 & b_2 & c_2 \end{bmatrix} \begin{bmatrix} X_M \\ Y_M \\ Z_M \end{bmatrix} = \begin{bmatrix} -d_1 \\ -d_{1t} \\ -d_2 \end{bmatrix} \tag{3.16}$$

Capítulo 4

Integración en ORB-SLAM2

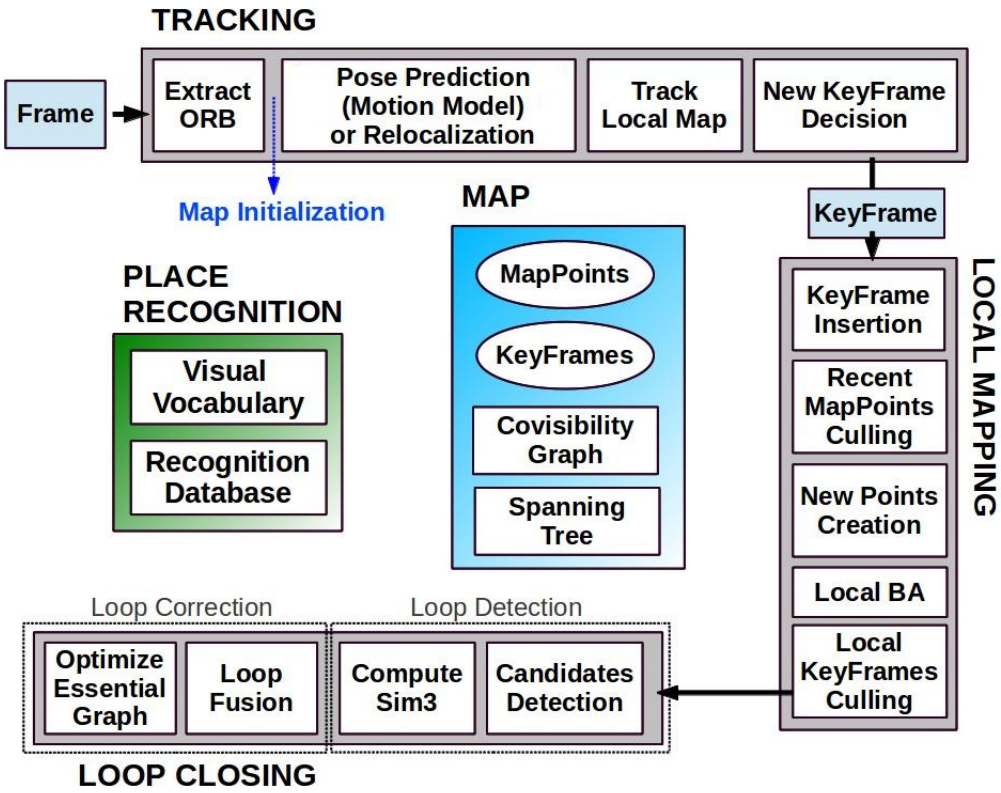
ORB-SLAM tiene tres hilos de ejecución en paralelo (Figura 4.1a): el hilo de *tracking* encargado de la localización de la cámara en cada *frame*; el hilo de *mapping* encargado de la inserción de nuevos puntos 3D al mapa mediante la triangulación de emparejamientos entre *keyframes*, y de la optimización de los puntos del mapa y la trayectoria de la cámara mediante el proceso de *bundle adjustment*; y el hilo de *loop closing* encargado de detectar y cerrar bucles en la trayectoria de la cámara. Un *frame* corresponde con cada imagen capturada por la cámara, mientras que un *keyframe* se trata de un *frame* especial, seleccionado por el proceso de *tracking*, que añade suficiente información nueva al mapa reconstruido. Los *edgelets* se han integrado sobre los procesos de *tracking* y *mapping* (Figura 4.1b).

4.1. Proceso de *tracking*

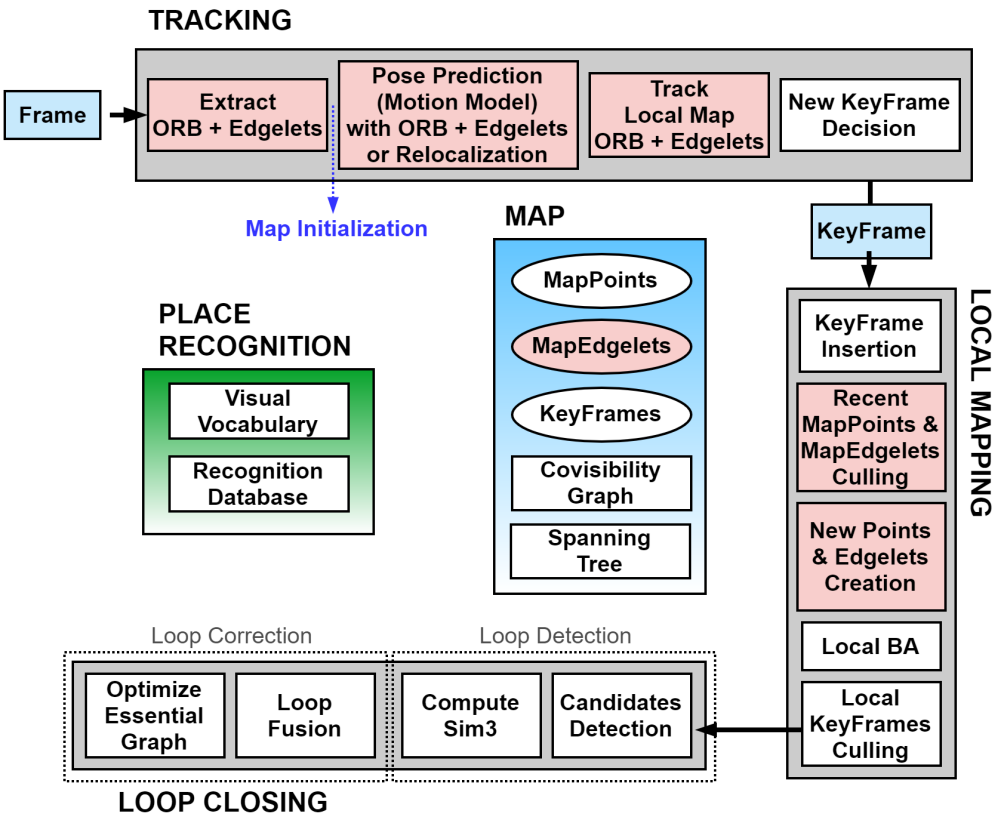
El proceso de *tracking* se encarga de proyectar los puntos 3D del mapa en el *frame* actual y encontrar emparejamientos. Es así un proceso que se ejecuta a frecuencia de *frame* y que necesita calcular las *features* de la imagen y computar sus descriptores. Los emparejamientos se obtienen explorando en una ventana local sobre la proyección predicha del punto 3D.

Dado que el mapa de puntos 3D aumenta con el tiempo de vida de ejecución, el proceso de *tracking* se centra sobre una parte del mapa y así evitar tiempos prohibitivos en mapas grandes. Esta porción se denomina mapa local, y consiste en un conjunto de *keyframes* K_1 que comparten algún punto con el *frame* anterior, y un conjunto K_2 con vecinos de los *keyframes* K_1 en el grafo de covisibilidad. Se proyecta así cada punto del mapa visto en K_1 o K_2.

El primer paso para integrar los *edgelets* en este proceso de *tracking* consiste en extraer los contornos del *frame* actual y calcular sus descriptores. Seguidamente, los *edgelets* 3D contenidos en el mapa local se proyectan sobre la imagen y se busca un emparejamiento mediante una búsqueda local de la misma forma que se ha explicado en la sección 3.4.1. Se tiene de esta manera un conjunto nuevo de emparejamientos de *edgelets* que complementa al conjunto ya existente de correspondencias de puntos ORB (Figura 4.2).



(a) Visión en conjunto del sistema original ORB-SLAM, Mur-Artal et al. (2015).



(b) Visión del sistema tras la integración de los *edgelets*. En color rojo se marca los módulos donde se han incorporado las nuevas *features*.

Figura 4.1: Visión en conjunto original de los tres hilos de ejecución concurrentes que conforman el sistema ORB-SLAM (imagen superior), en comparación con el sistema tras la integración de los *edgelets* (imagen inferior).

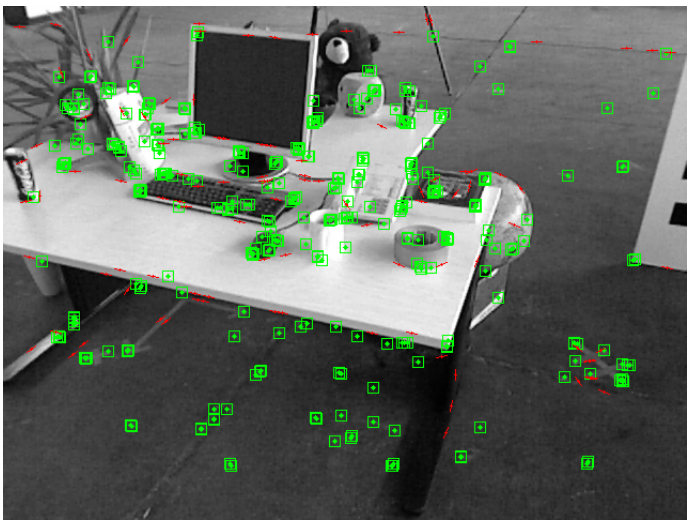


Figura 4.2: Emparejamientos obtenidos durante el proceso de *tracking* del sistema ORB-SLAM2 integrado con *edgelets* sobre un *frame* de la secuencia *freiburg2_desk*. En color verde se visualizan los emparejamientos de puntos ORB ya presentes en la versión original del sistema y en color rojo los emparejamientos de *edgelets* añadidos en la nueva versión.

Este proceso de *tracking* es consciente de un mal funcionamiento cuando no se han encontrado suficientes emparejamientos. Esto significa que el *tracking* se ha perdido, no se conoce la localización de la cámara y es necesaria una relocalización. Esta condición se da actualmente en el sistema cuando el número de emparejamientos de puntos ORB es inferior a veinte. Para completar la integración de los *edgelets* en este proceso de *tracking* se ha actualizado esta condición a que la suma de los emparejamientos de puntos ORB y de *edgelets* sea inferior al mismo umbral. Se consigue de esta manera un correcto funcionamiento del sistema de localización en escenas con ausencia de puntos ORB pero presencia de contornos.

4.2. *Motion-only bundle adjustment*

Dado un conjunto de correspondencias entre puntos 3D del mapa y puntos 2D de la imagen se puede estimar la localización de la cámara en el momento actual mediante un proceso de optimización conocido como *bundle adjustment*.

Nótese que el conjunto de emparejamientos se utiliza para poder estimar la localización de la cámara en el momento actual, pero que para calcular estos emparejamientos se necesita una estimación de la localización para realizar el proceso de proyección de puntos. La estimación inicial de la cámara se realiza mediante un sencillo algoritmo de estimación de movimiento. El proceso de *bundle adjustment* calcula una mejor aproximación de la localización de la cámara con los emparejamientos obtenidos sobre la estimación inicial.

El proceso de *bundle adjustment* optimiza las poses de todas las cámaras y las posiciones de todas las *features*, de tal manera que se minimiza el error geométrico de los emparejamientos existentes entre los puntos 3D del mapa y los puntos 2D de cada cámara.

Existe un caso particular de este proceso conocido como *motion-only bundle adjustment* que optimiza la orientación de una sola cámara $\mathbf{R} \in SO(3)$ y su posición $\mathbf{t} \in \mathbb{R}^3$, minimizando el error de reproyección entre los puntos 3D del mapa $\mathbf{X}_M^i \in \mathbb{R}^3$ en coordenadas del mundo y los puntos 2D de la imagen $\mathbf{x}^i \in \mathbb{R}^2$, con $i \in \mathcal{X}$ el conjunto de los emparejamientos:

$$\{\mathbf{R}, \mathbf{t}\} = \underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \sum_{i \in \mathcal{X}} \rho \left(\left\| \mathbf{x}^i - \pi_{\mathbf{R},\mathbf{t}} \left(\mathbf{X}_M^i \right) \right\|_\Sigma^2 \right) \tag{4.1}$$

La función robusta de Huber ρ penaliza errores grandes de reproyección debidos a posibles emparejamientos espurios. La función de proyección π es la definida en la ecuación 3.12. Solamente se utiliza la posición del *edgelet* en el residuo de optimización. La matriz Σ impone la desviación estándar que se permite en el error geométrico de un emparejamiento. En el caso de los puntos ORB, se permite un error geométrico pequeño hacia ambos ejes de la imagen:

$$\Sigma_{ORB} = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix} \tag{4.2}$$

El error permitido es de un píxel ($\sigma = 1$). Además, este valor se multiplica por la escala en la que se ha obtenido el punto ORB. En los *edgelets* se permite un error mayor en la dirección del contorno ϕ, ya que la posición de un *edgelet* es poco precisa longitudinalmente:

$$\Sigma_{edgelet} = R \cdot \begin{bmatrix} {\sigma_{long}}^2 & 0 \\ 0 & {\sigma_{trans}}^2 \end{bmatrix} \cdot R^T \,, \qquad R = \begin{bmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{bmatrix} \tag{4.3}$$

Se ha definido este error longitudinal como 10 píxeles, mientras que el error transversal se toma 1 píxel como en los puntos ORB. Es necesaria la multiplicación por la matriz de rotación R para adecuar la matriz Σ a la orientación del *edgelet*.

Este problema de optimización de una función no lineal se ha resuelto utilizando el método Levenberg-Marquardt implementado en la librería g2o de Kümmerle et al. (2011).

4.3. Proceso de *mapping*

Los dos procesos anteriores se centran en la tarea de localización de la cámara. Para completar el significado de *SLAM* es necesaria una reconstrucción del mapa 3D, siendo encargado de esta tarea el proceso de *mapping*.

Para calcular la profundidad de un punto con una cámara monocular se necesita conocer su localización en dos imágenes tomadas desde posiciones diferentes. De esta manera, se puede introducir el punto 3D en el mapa mediante el procedimiento de triangulación comentado en la sección 3.6. La primera localización se obtiene mediante la extracción de *features* en la

imagen actual, realizada en el proceso de *tracking*. La segunda se calcula en este proceso de *mapping*, mediante el seguimiento de las *features* extraídas hasta una posición anterior o posterior de la cámara que mantenga una condición de paralaje suficiente con la inicial.

La extracción de *edgelets* se realiza como se ha explicado en la sección 3.2, añadiendo la condición de extraer puntos de contorno nuevos, aún no contenidos en el mapa 3D. De esta manera, no se van a seleccionar como *edgelets* aquellos puntos de contorno cercanos a los emparejamientos con el mapa local obtenidos en el proceso de *tracking*.

El seguimiento de los *edgelets* se modeló en un primer intento mediante una búsqueda local (sec. 3.4.1) en el siguiente *keyframe*. La mayoría de los emparejamientos se desplazaban a lo largo del contorno, por lo que la triangulación era errónea. En un segundo intento, se añadió la restricción epipolar (sec. 3.4.2) a la búsqueda en el siguiente *keyframe*. Añadir esta restricción implica la detección de movimientos de la cámara en la dirección de un contorno; en tal caso, la línea epipolar se solapa con el contorno y no se puede distinguir el *edgelet* buscado. Para esta detección hay que calcular la línea epipolar $\mathbf{l}_1$ en el *keyframe* inicial y comprobar que tiene una diferencia angular suficiente con el contorno. Como ya se ha comentado, el producto vectorial de dos puntos en coordenadas homogéneas define la ecuación de la recta que pasa por ambos. De esta manera, la línea epipolar $\mathbf{l}_1$ se define como el producto vectorial entre la posición $\mathbf{x}_1$ del *edgelet* y el epipolo $\mathbf{e}_1$ (ec. 4.4). El epipolo es la proyección del centro óptico $\mathbf{c}_2$ de la segunda imagen en la primera. Los resultados mejoran respecto a la anterior aproximación, pero sigue habiendo una imprecisión en la posición de los *edgelets* 3D.

$$\tilde{\mathbf{l}}_1 = \tilde{\mathbf{x}}_1 \times \tilde{\mathbf{e}}_1 \ , \qquad \tilde{\mathbf{e}}_1 = P_1 \cdot \tilde{\mathbf{c}}_2 \tag{4.4}$$

Como tercera aproximación, se detecta que la *baseline* entre *keyframes* consecutivos no es la suficiente para realizar un proceso de triangulación preciso. ORB-SLAM tiene una política de inserción de *keyframes* generosa, de tal manera que se facilita la inserción de puntos en el mapa. Posteriormente, en este hilo de ejecución de *mapping*, se realiza un proceso denominado *culling* que elimina *keyframes* redundantes y puntos del mapa erróneos. Teniendo en cuenta esta política, se ha añadido una restricción mayor al *baseline* mínimo b entre *keyframes* para realizar el proceso de triangulación. Esta restricción implica que la cámara se haya movido lo suficiente en base a su distancia a la escena (ec. 4.5). Se calcula así una estimación de la profundidad media $\bar{d}$ de ésta conociendo los puntos del mapa contenidos en la imagen. El seguimiento de los *edgelets* se realiza ahora mediante una búsqueda local con restricción epipolar en *keyframes* cercanos que cumplan la nueva condición impuesta.

$$\frac{b}{\bar{d}} > 0{,}075 \tag{4.5}$$

Los resultados tras esta consideración mejoran respecto a los procesos anteriores. En el mapa de la escena reconstruida se tiene una idea intuitiva de los contornos de la misma (Figura 4.3, `https://youtu.be/gOnnKomMbx8`). Si bien, la reconstrucción del mapa no muestra una fidelidad completa de la escena, presentando incluso un número notable de puntos de contorno erróneos, lo que refleja la todavía inmadurez de la *feature* desarrollada en el contexto de *SLAM*. En términos cuantitativos, un 3,15 % de los puntos ORB utilizados en el proceso de *bundle adjustment* resultan espurios, por contra de un 58,42 % en el caso de los *edgelets*. Aunque se trata de una tasa de error elevada, también se presenta una primera base de esta nueva *feature* que atrae el interés de un trabajo futuro de profundización.

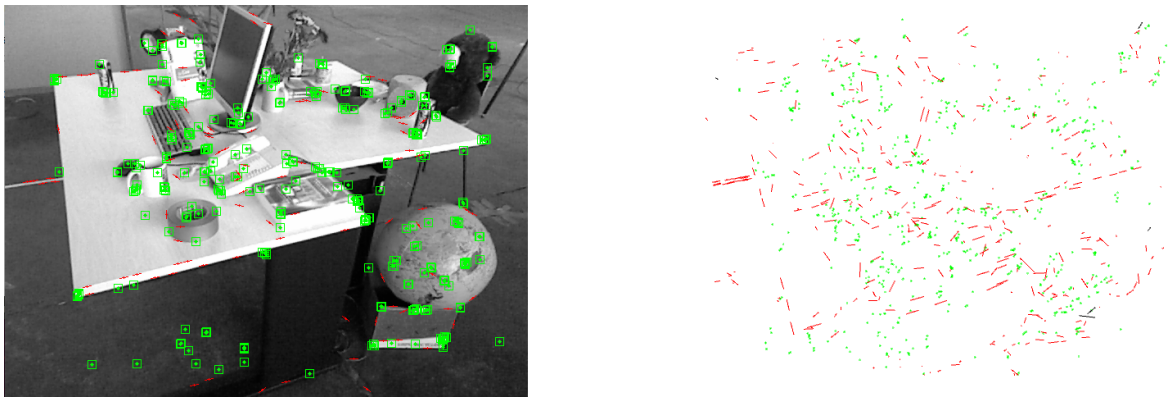


Figura 4.3: En la imagen derecha, el mapa 3D reconstruido de la escena de la imagen izquierda. En color verde se visualizan los puntos ORB y en color rojo los *edgelets* añadidos. El mapa visualizado intuye una reconstrucción de la escena en la que se observan los principales contornos de la misma, como los límites de la mesa.

Capítulo 5

Demo de realidad aumentada

Una visión de realidad aumentada combina elementos virtuales con el entorno físico del mundo real capturado por la cámara. Para poder situar los elementos virtuales se necesita conocer la posición y orientación en tiempo real de la cámara de captura. El sistema ORB-SLAM es capaz de conseguir este aspecto, de igual manera que lo es un sistema de captura de movimiento. En este capítulo se pretende comparar ambos sistemas en forma de experiencia de usuario en una demo de realidad aumentada.

5.1. Seguimiento de la cámara con OptiTrack

He colaborado en la puesta en marcha de un sistema de captura de movimiento OptiTrack (Anexo B). Con la ayuda de este sistema, he capturado una secuencia de vídeo con la pose de la cámara conocida en todo momento con un error sub-milimétrico. Localizar la cámara en todo momento implica la necesidad de que OptiTrack sepa detectarla. Para ello, he construido un armazón estable sobre la cámara de captura en el que he situado cuatro marcadores distanciados en las tres dimensiones (Figura 5.1). Estos marcadores conforman un cuerpo rígido A que permite la localización de la cámara sobre el sistema OptiTrack.

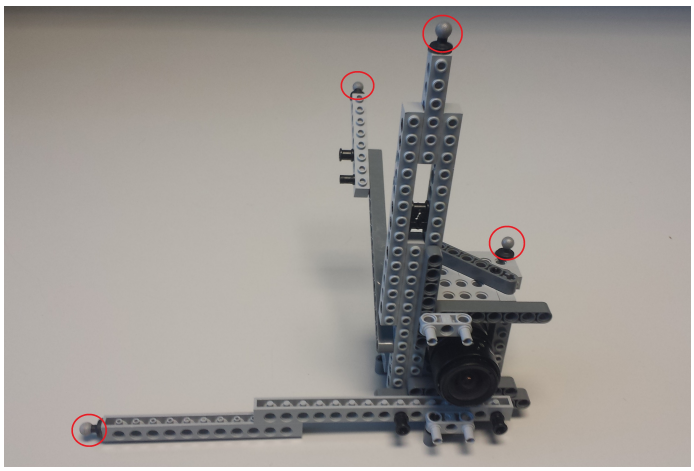


Figura 5.1: Armazón construido sobre la cámara de captura de la secuencia. Se han situado cuatro marcadores sobre éste (en color rojo), de tal manera que el sistema OptiTrack sabe localizar la cámara en todo momento.

Para que un *frame* de la cámara de captura se corresponda con un *frame* del sistema OptiTrack, es necesario igualar la frecuencia de captura de ambos sistemas. La frecuencia de captura de OptiTrack es controlada por el propio sistema, mientras que para la cámara se ha desarrollado un programa de captura propio. Este programa pide a la cámara las imágenes a la frecuencia deseada, y las guarda en disco tal cual se las envía ésta. Se detectó que con este funcionamiento se conseguía una frecuencia de captura ligeramente menor a la ordenada. Se pensó como origen de este problema la incapacidad de la escritura en disco a la misma frecuencia que la captura de imágenes. En una segunda versión del programa desarrollado, las imágenes se guardan en memoria RAM y se vuelcan a disco al terminar el proceso de captura, consiguiendo satisfactoriamente la frecuencia deseada.

Por otra parte, estos sistemas empiezan la captura en momentos diferentes por regla general. La granularidad deseada es de milisegundos, por lo que se considera impensable iniciar los sistemas en el mismo momento de forma manual. La sincronización se ha conseguido situando la cámara totalmente quieta en el volumen de captura. Seguidamente, se ha realizado un movimiento brusco sobre ella. El *frame* en el que se detecta un cambio en la posición de la cámara marca el punto de sincronización de los dos sistemas. A partir de este, se tiene una asociación completa de los *frames* de la cámara con los *frames* de OptiTrack.

Con el proceso anterior, se tiene a la cámara localizada en cada *frame* de la secuencia. En verdad, se tiene localizado el cuerpo rígido A y no el centro óptico C de la cámara. Para conocer la localización del centro óptico de la cámara sobre el sistema de referencia de OptiTrack, T_{OC}, hay que realizar un proceso de calibración para calcular la transformación T_{AC} del centro óptico de la cámara al cuerpo rígido. Siendo T_{OA} la medida conocida de localización del cuerpo rígido respecto al sistema de referencia OptiTrack, la localización del centro óptico de la cámara sigue la ecuación 5.1.

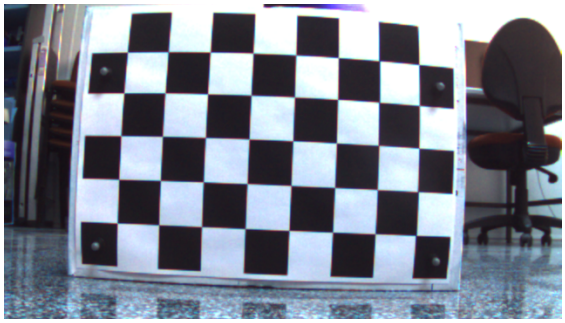
$$T_{OC} = T_{OA} \cdot T_{AC} \tag{5.1}$$

Para la calibración de la transformación entre el centro óptico de la cámara y el cuerpo rígido hay que construir un escenario como se aprecia en la Figura 5.3. La localización del cuerpo rígido T_{OA} es conocida. También se conocen las posiciones de los cuatro marcadores situados en el patrón de calibración. La cámara está enfocando este patrón, y los cuatro marcadores están visibles en la imagen capturada (Figura 5.3b). Se conocen así las posiciones 2D de los centroides de los marcadores en la imagen. Las correspondencias entre las posiciones 3D y 2D de los marcadores son también conocidas, por lo que se puede calcular la localización del centro óptico de la cámara T_{OC} mediante el algoritmo P3P. La transformación T_{AC} entre el armazón de la cámara y el centro óptico de la misma se calcula finalmente despejando estos dos valores en la ecuación 5.1.

Figura 5.2: Umbralización de los marcadores con el método de Otsu.

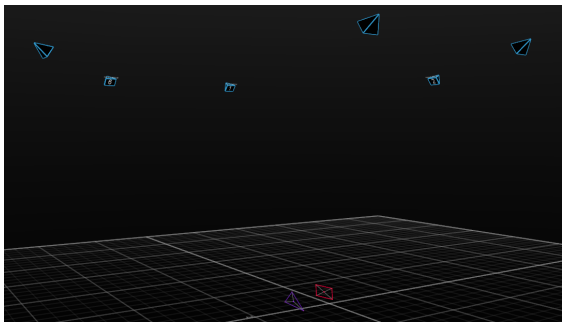
Los marcadores se han situado sobre un fondo negro. De esta manera, mediante una umbralización con el método de Otsu se etiqueta si un píxel pertenece al marcador. El cálculo del centroide de un marcador es directo con esta información.

(a) Cámara enfocando el patrón de calibración.



(b) Imagen capturada por la cámara.

(c) Escenario de calibración dentro del volumen de captura de OptiTrack.



(d) Patrón de calibración (en color rojo) y cámara (en morado) detectados por OptiTrack.

Figura 5.3: Escenario de calibración para calcular la transformación del armazón de la cámara al centro óptico de la misma.

5.2. Realidad aumentada y experiencia de usuario

Se ha desarrollado una demo de realidad aumentada basada en el trabajo de Raúl Mur[1]. En esta demo se compara la experiencia de usuario si la localización de la cámara se realiza mediante ORB-SLAM, o si se mantiene la localización calculada por un sistema de captura de movimiento. Si se observa este vídeo, se percibe cómo la experiencia de usuario ofrecida por el sistema ORB-SLAM es notablemente mejor. Este sistema ofrece una mayor robustez de los objetos virtuales, dando la sensación de que éstos son un elemento más del mundo real. La localización con el sistema de captura de movimiento, en cambio, produce un efecto de oscilación de los objetos que les disminuye esta cualidad de realismo.

Demo de realidad aumentada: `https://youtu.be/zV58K-C8T-I`

[1] `https://github.com/raulmur/ORB_SLAM2/tree/master/Examples/ROS/ORB_SLAM2/src/AR`

Figura 5.4: Frame de la demo de realidad aumentada desarrollada.

La robustez conseguida por ORB-SLAM se explica por la asociación de los objetos virtuales con el mapa reconstruido del entorno. Este aspecto permite además un sencillo posicionamiento de nuevos elementos virtuales en zonas propensas a albergar objetos comunes. Por ejemplo, si se quieren posicionar sobre superficies planas, como puede ser encima de una mesa o del suelo de una habitación, la detección de planos a través de un algoritmo de consenso con los puntos contenidos en el mapa facilita este proceso.

El posicionamiento de nuevos elementos virtuales con el sistema de captura de movimiento requiere conocer las coordenadas sobre las que se quieren posicionar. El efecto de oscilación que afecta de manera negativa a la experiencia de usuario se explica en la Figura 5.5. Un error ε_m en el cálculo de la posición de un marcador va a originar un error angular α en relación a la distancia r del marcador al centroide de todos los marcadores. Por semejanza de triángulos, se cumple la ecuación 5.2. El error en píxeles ε_p de la proyección del objeto en el plano imagen es directamente proporcional al error del marcador y la distancia local f, e inversamente proporcional al radio r del marcador al centroide. De forma parecida sucede con el error ε_o del objeto 3D. Si se considera un error medio del marcador de 1 mm, en un escenario con una cámara de 200 píxeles de distancia focal y una separación r del marcador de 10 cm, se obtiene una oscilación del objeto virtual de dos píxeles.

$$\frac{\varepsilon_m}{r} = \frac{\varepsilon_p}{f} = \frac{\varepsilon_o}{d} \ , \qquad \varepsilon_p = f \cdot \frac{\varepsilon_m}{r} \ , \qquad \varepsilon_o = d \cdot \frac{\varepsilon_m}{r} \tag{5.2}$$

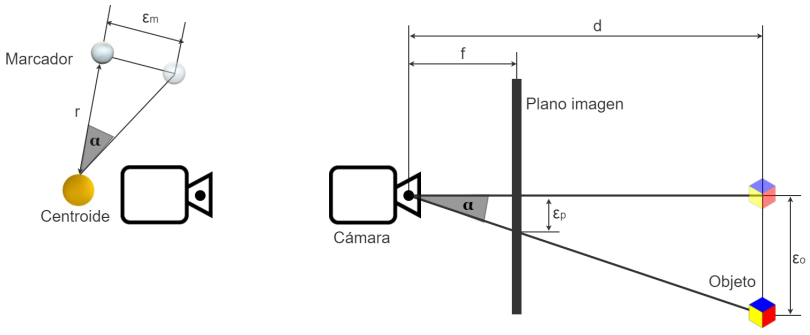


Figura 5.5: Una menor separación r de los marcadores supone un mayor error angular α, con la consiguiente oscilación ε_p del objeto virtual.

Capítulo 6

Conclusiones

En este capítulo se extraen las principales conclusiones del trabajo realizado. Asimismo, se proponen algunas líneas interesantes de trabajo futuro. Finalmente, se proporciona una evaluación personal del autor.

6.1. Conclusiones generales

La realización de este trabajo ha supuesto la adición en el sistema ORB-SLAM del soporte para cámaras con objetivo gran angular, como es el caso de las cámaras *fisheye*.

También, se ha investigado y desarrollado una *feature*, adoptada con el nombre de *edgelet*, que corresponde a los puntos de contorno de una escena. La extracción de estas *features* es un proceso rápido, ejecutándose en la mitad de tiempo que la extracción de contornos de Canny, y mejorando incluso el tiempo de otros extractores de renombre como ORB. Este hecho concluye en la capacidad del uso de los *edgelets* en un sistema de tiempo real. Se ha ideado un descriptor para esta *feature* que permite un buen proceso de emparejamiento y seguimiento. La integración de los *edgelets* en un sistema de SLAM deja intuir una primera reconstrucción, todavía inmadura, de los contornos de una escena.

Por último, se ha instalado y calibrado un sistema de captura de movimiento OptiTrack en el laboratorio 1.07 del edificio Ada Byron de la Universidad de Zaragoza. Se ha preparado una guía de uso básica (secciones B.3 y B.4) para la facilitación de futuros trabajos académicos y de investigación. Se ha desarrollado una demo de realidad aumentada sobre una secuencia con la pose de la cámara calculada por un sistema de captura de movimiento. La conclusión directa de este experimento es que se obtiene una mejor experiencia de usuario y sensación de realismo si se localiza la cámara con ORB-SLAM frente al sistema de captura de movimiento.

6.2. Trabajo futuro

Como trabajo futuro, se propone la adición de más tipos de cámara al sistema ORB-SLAM, como por ejemplo la cámara omnidireccional. Incluir algún patrón de diseño

software que permita la encapsulación del tipo de cámara utilizada facilitaría la futura adición de múltiples modelos de cámara.

Sobre la base establecida en este trabajo de los *edgelets*, se propone profundizar en su desarrollo e intentar conseguir un buen desempeño en el contexto de *SLAM*. Por ejemplo, la orientación de un *edgelet* no se utiliza en el proceso de *bundle adjustment*, más que para adecuar la matriz sigma a la orientación del *edgelet*. De igual forma, se puede integrar completamente los *edgelets* en el proceso de *bundle adjustment* global y no únicamente en el *motion-only*. Ambas resultan interesantes líneas de trabajo futuro.

Asimismo, se pueden explorar otras alternativas para integrar los contornos de una escena en la reconstrucción del mapa y la localización de la cámara. Una vía de exploración puede ser considerar un pequeño conjunto de puntos de contorno unidos como una nueva *feature* independiente. De esta manera, la posible imprecisión en el cálculo de un punto de contorno puede compensarse con la información del resto de puntos de contorno.

6.3. Evaluación personal

Este proyecto ha significado para mí la puerta de entrada al mundo de la investigación. Los problemas abordados han sido diferentes a los que me he encontrado a lo largo de estos cuatro últimos años de estudio en el grado. Por ejemplo, en el desarrollo de cualquier aplicación o página web, el seguimiento de unas directrices solía ser el camino directo a la solución final. En el campo de la investigación, sin embargo, para alcanzar la solución final debes trabajar con hipótesis, comprobando las mismas, sin que ello te garantice que incluso de todas las hipótesis barajadas alguna sea la llave necesaria para resolver el problema. Ante este nuevo contexto de problemas he aprendido a cambiar el *chip* y aumentar mi iniciativa a desarrollar múltiples soluciones, aunque éstas finalmente puedan no tener el resultado esperado.

Este trabajo también me ha supuesto una mejora en la gestión de proyectos al tratarse del primero realizado de estas dimensiones. De mayores dimensiones ha sido el proyecto sobre el que he tenido que integrar mi parte de trabajo: un tamaño de cerca de veinticinco mil líneas de código[1] y una repercusión con más de mil extensiones de trabajo en su repositorio público[2]. El total desconcierto inicial ante este proyecto y ante tecnologías nuevas fue planteado como un reto que ha sido superado exitosamente. También por primera vez he realizado un proceso de instalación y configuración de un sistema de tan elevadas dimensiones y coste económico como ha sido del sistema de captura de movimiento OptiTrack. Como evaluación final, el proyecto ha sido una experiencia perfecta que me gustaría continuar con una tesis doctoral.

[1] Líneas de código calculadas con la herramienta CLOC: `http://cloc.sourceforge.net/`
[2] `https://github.com/raulmur/ORB_SLAM2`

Capítulo 7

Bibliografía

Devernay, Frederic and Olivier Faugeras (2001), "Straight lines have to be straight." *Machine vision and applications*, 13, 14–24.

Eade, Ethan and Tom Drummond (2009), "Edge landmarks in monocular slam." *Image and Vision Computing*, 27, 588–596.

Hartley, Richard and Andrew Zisserman (2003), *Multiple view geometry in computer vision*. Cambridge university press.

Klein, Georg and David Murray (2007), "Parallel tracking and mapping for small ar workspaces." In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, 225–234, IEEE.

Kümmerle, Rainer, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard (2011), "g 2 o: A general framework for graph optimization." In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 3607–3613, IEEE.

Mur-Artal, Raul, Jose Maria Martinez Montiel, and Juan D Tardos (2015), "Orb-slam: a versatile and accurate monocular slam system." *IEEE Transactions on Robotics*, 31, 1147–1163.

Mur-Artal, Raúl and Juan D Tardós (2017), "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras." *IEEE Transactions on Robotics*.

Rosten, Edward and Tom Drummond (2006), "Machine learning for high-speed corner detection." *Computer Vision–ECCV 2006*, 430–443.

Rublee, Ethan, Vincent Rabaud, Kurt Konolige, and Gary Bradski (2011), "Orb: An efficient alternative to sift or surf." In *Computer Vision (ICCV), 2011 IEEE international conference on*, 2564–2571, IEEE.

Sturm, J., N. Engelhard, F. Endres, W. Burgard, and D. Cremers (2012), "A benchmark for the evaluation of rgb-d slam systems." In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*.

Szeliski, Richard (2010), *Computer vision: algorithms and applications*. Springer Science & Business Media.

Glosario

Contorno Un contorno es un salto de intensidad considerable en una imagen. De una manera más visual, un contorno puede corresponder a los límites de un objeto, de una textura, incluso de una sombra o un reflejo.

Edgelet Un *edgelet* es una porción local de un contorno. Esta porción es localmente recta. Se puede visualizar una línea curva como la unión de varios segmentos localmente rectos.

Feature En visión por computador y procesamiento de imágenes, una *feature* es una pieza de información relevante para resolver cierta tarea computacional. En el contexto de *SLAM*, una *feature* es una zona de interés de la imagen como una esquina, un contorno o un objeto.

Objetivo gran angular Una cámara con objetivo gran angular es aquella con una distancia focal sustancialmente menor a la de una cámara con objetivo normal, resultando un campo de visión mayor al de la visión humana.

Realidad aumentada Visión a través de un dispositivo tecnológico de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales, creando así una realidad mixta en tiempo real.

SLAM Localización y Mapeado Simultáneos, del inglés *Simultaneous Localization And Mapping*, es una técnica usada por robots y vehículos autónomos para construir un mapa de un entorno desconocido en el que se encuentra, a la vez que estima su trayectoria al desplazarse dentro de este entorno.

Anexos

Anexo A
Gestión del proyecto

En este Anexo se presenta la planificación del proyecto, la progresión del mismo y un desglose cuantitativo de los esfuerzos invertidos.

A.1. Planificación del proyecto

En el lanzamiento del proyecto se detallaron el alcance y los objetivos del mismo (ver sección 1.2). De igual manera, se realizó una planificación temporal de las principales actividades que conforman los objetivos marcados (Figura A.1). La fecha de finalización del proyecto se marcó para el día 20 de junio de 2017. La necesidad de una nueva distribución del laboratorio de despliegue del sistema de captura de movimiento OptiTrack causó un retraso del proyecto respecto a la planificación establecida inicialmente. De esta manera, con fecha 19 de abril se realizó una re-planificación del proyecto (Figura A.2), definiendo el día 28 de agosto como nueva fecha de finalización de éste.

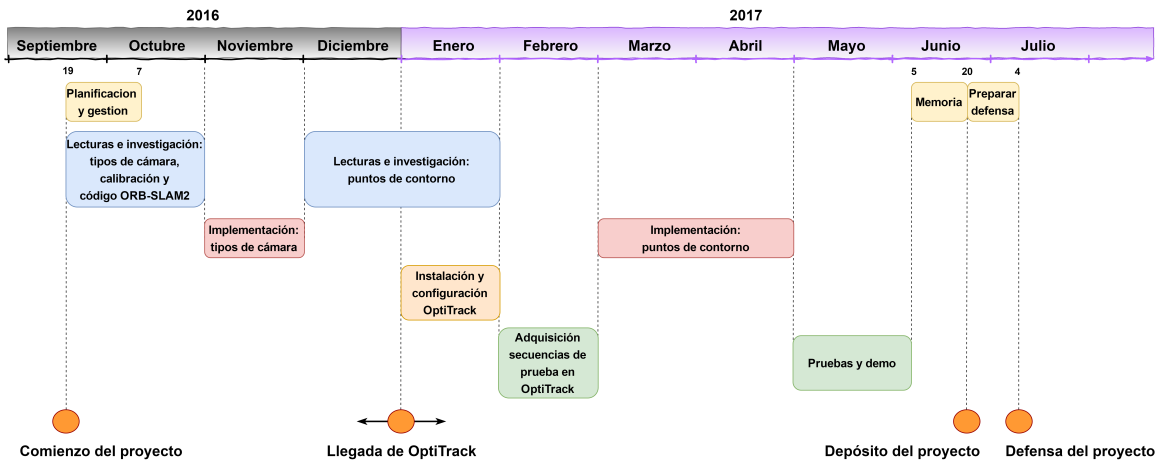


Figura A.1: Planificación temporal inicial del proyecto.

A.2. Esfuerzos invertidos

Se detalla en la Figura A.3 una gráfica de la dedicación real a las distintas actividades del proyecto en forma de diagrama de Gantt.

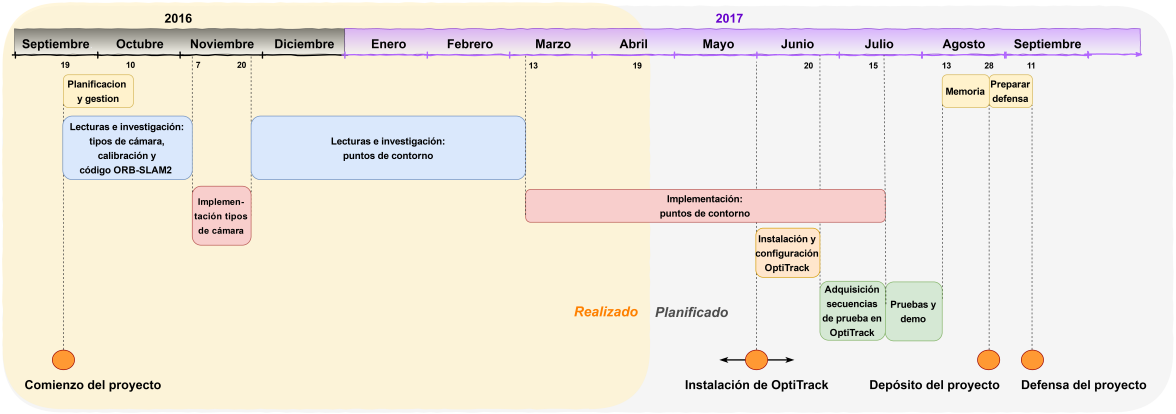


Figura A.2: Re-planificación temporal del proyecto.

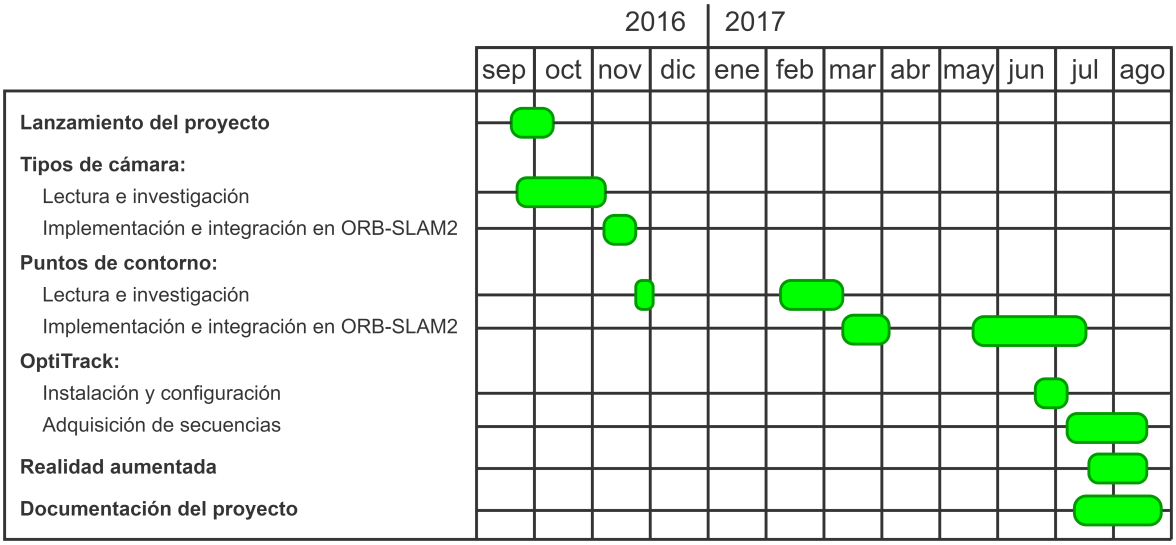
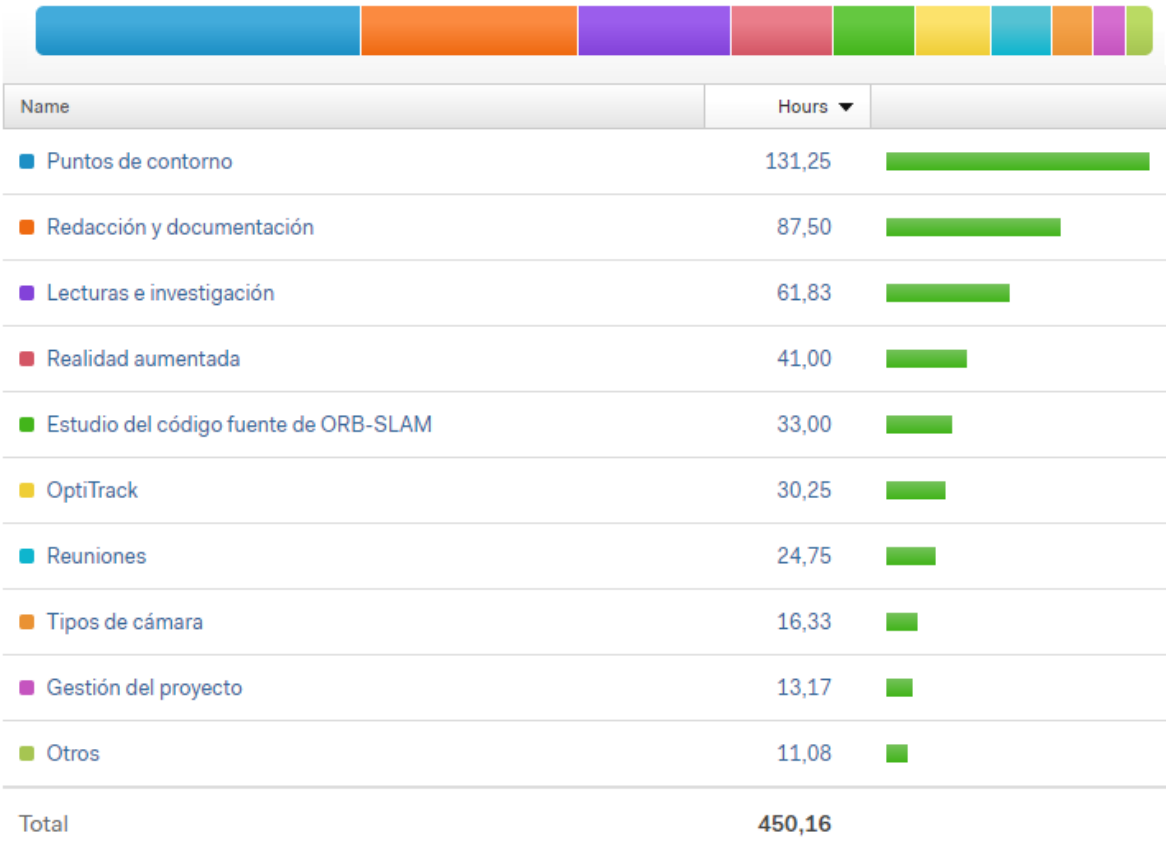


Figura A.3: Diagrama temporal detallando los esfuerzos invertidos en las distintas actividades del proyecto realizado.

En la Figura A.4 se presenta el desglose de los esfuerzos invertidos en las distintas actividades del proyecto. La cantidad total de horas invertidas es de 450,16.

Por último, la Figura A.5 muestra un desglose temporal de los esfuerzos invertidos en el proyecto que permite visualizar la evolución del mismo.

Como medida de esfuerzo adicional, las líneas de código integradas en el sistema ORB-SLAM2 hacen un total de 1140. Originalmente, la versión base del sistema se compone de 24712 líneas de código.



Name	Hours
Puntos de contorno	131,25
Redacción y documentación	87,50
Lecturas e investigación	61,83
Realidad aumentada	41,00
Estudio del código fuente de ORB-SLAM	33,00
OptiTrack	30,25
Reuniones	24,75
Tipos de cámara	16,33
Gestión del proyecto	13,17
Otros	11,08
Total	**450,16**

Figura A.4: Desglose cuantitativo de los esfuerzos invertidos en las distintas actividades del proyecto realizado.

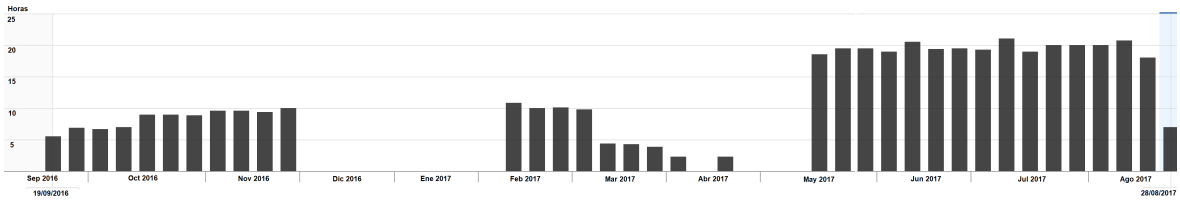


Figura A.5: Desglose temporal de los esfuerzos invertidos en el proyecto realizado.

Anexo B
OptiTrack

A continuación se describen los detalles técnicos y el proceso de instalación seguido para la puesta en marcha del sistema de captura de movimiento OptiTrack. Asimismo, se detallan los principales casos de uso del software Motive para utilizar este sistema.

B.1. Detalles técnicos

El sistema de captura de movimiento OptiTrack instalado consta de seis cámaras Prime 41 (Figura B.1). Esta cámara tiene una resolución de 2048 x 2048 píxeles con un tamaño de píxel de 5,5 μm x 5,5 μm. Tiene un ratio de frame ajustable entre 30 y 180 FPS, con una latencia de 5,5 ms. Mediante barrido *global shutter*, tiene una velocidad de disparo por defecto de 0,5 ms, llegando a un valor mínimo de 0,01 ms y un valor máximo de 8,1 ms a 120 FPS y 5,3 ms a 180 FPS. La cámara tiene una lente de 12 mm F#1,8 y un campo de visión de 51º en ambos ejes horizontal y vertical, con un filtro de paso de banda de 850 nm. La especificación del tamaño de la cámara se detalla en la Figura B.2.

Figura B.1: Cámara Prime 41 integrada en el sistema OptiTrack instalado.

Para la calibración se utiliza la serie Micron ofrecida por OptiTrack (Figura B.3). Esta serie es idónea para calibraciones extremadamente precisas de volúmenes de captura de tamaño pequeño a mediano. Las varillas permanecen calibradas a lo largo de un amplio rango de temperatura dada su constitución de la aleación Invar, destacada por su coeficiente de expansión térmica excepcionalmente bajo. La precisión de calibración tiene una desviación estándar de dos micrones. La varilla de calibración tiene tres marcadores de 12.7 mm espaciados 250 mm. El cuadrado de calibración se especifica en la Figura B.4.

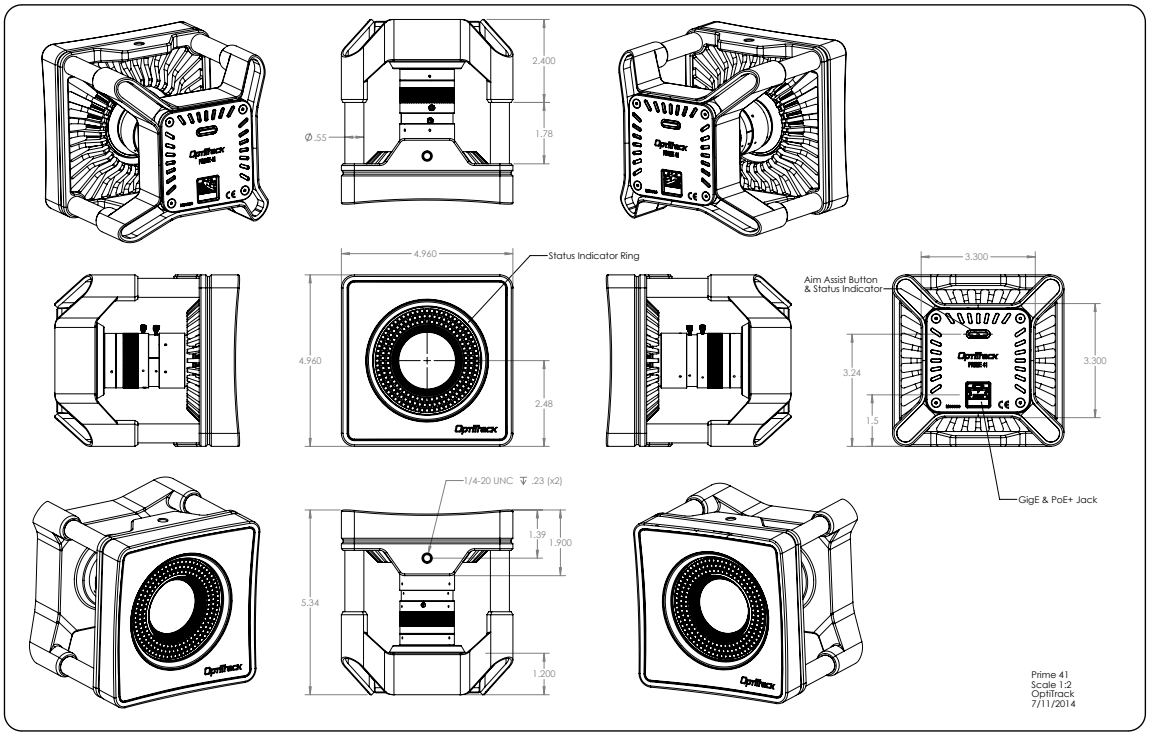


Figura B.2: Dibujo técnico de la cámara Prime 41.

(a) Varilla de calibración CWM-250. (b) Cuadrado de calibración CS-200.

Figura B.3: Serie Micron utilizada para la calibración del sistema OptiTrack.

B.2. Instalación y puesta en marcha

La zona de despliegue del sistema se trata del laboratorio 1.07 del edificio Ada Byron de la Universidad de Zaragoza[1]. El proceso de instalación ha sido dirigido por Luis Riazuelo Latas, ingeniero asociado a proyectos del grupo de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza; y en colaboración con Javier Domínguez Conti, estudiante de Máster de Ingeniería Industrial de la Universidad de Zaragoza.

Se muestra en la Figura B.5 el proceso evolutivo de la instalación del sistema en el laboratorio. En primera instancia, fue necesario despejar la zona donde se iba a desplegar el sistema. Como se observa en la Figura B.5b, esto implicó desalojar las mesas centrales del

[1]Calle María de Luna 3, 50018, Zaragoza

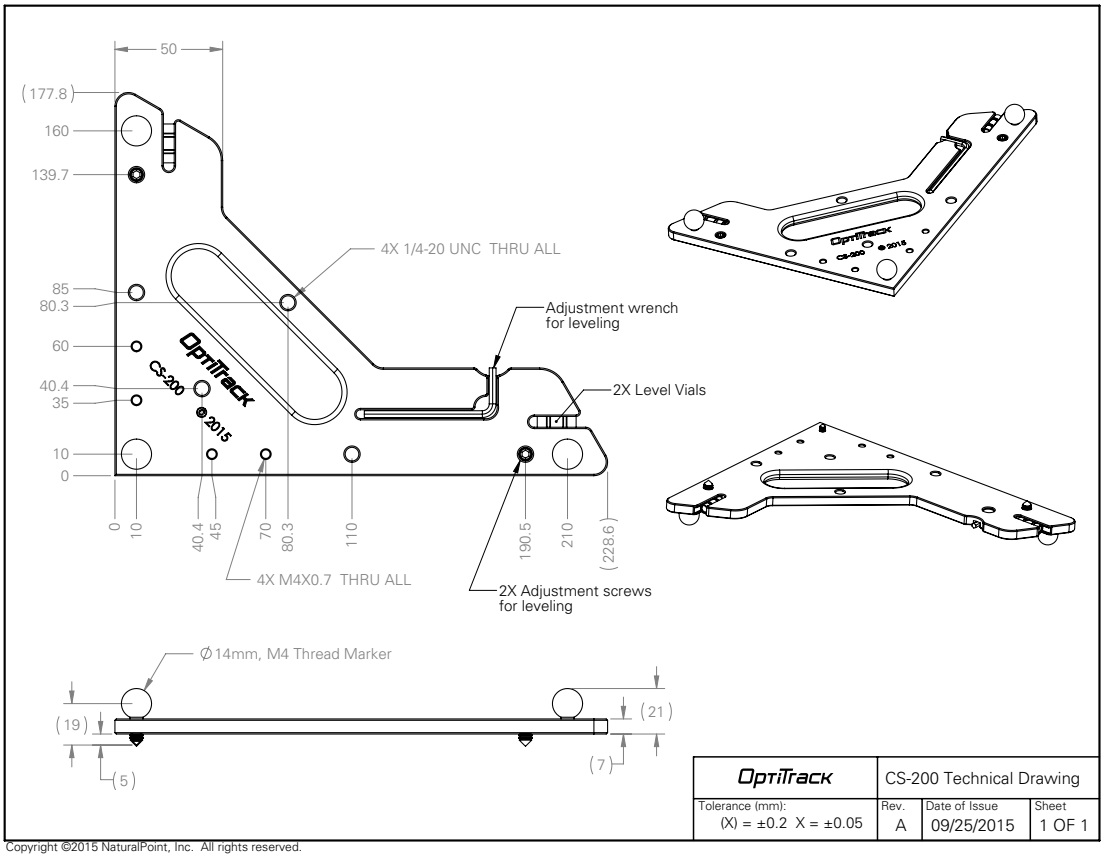


Figura B.4: Dibujo técnico del cuadrado de calibración CS-200.

laboratorio y cortar la corriente eléctrica que alimentaba los equipos correspondientes. La siguiente fase consistió en el montaje de la estructura del sistema sobre la que reposan las cámaras de captura de movimiento. El tamaño de esta estructura es de 5,70 x 5,70 x 3,00 metros, formando una base de aspecto cuadrado. Finalmente, se han colocado las cámaras sobre la estructura y se han conectado mediante *Ethernet* al *switch* central que está conectado al equipo de trabajo del laboratorio. La disposición de las seis cámaras es de una unidad en dos laterales contrarios, y dos unidades en los dos laterales restantes.

B.3. Calibración del sistema

A continuación se recogen las instrucciones para la calibración del sistema OptiTrack. Se va a utilizar el software Motive, disponible en el equipo de trabajo del laboratorio. La calibración total del sistema consta de dos fases.

> ***Nota:*** *Se recomienda calibrar el sistema antes de cada experimento.*

(a) Estado inicial del laboratorio antes de la instalación del sistema.

(b) Estado intermedio del laboratorio una vez desalojada la parte central.

(c) Estado final del laboratorio después de la instalación del sistema.

Figura B.5: Proceso evolutivo de la instalación del sistema de captura de movimiento OptiTrack en el laboratorio 1.07 del edificio Ada Byron de la Universidad de Zaragoza.

B.3.1. Posición y orientación de las cámaras

La calibración del sistema tiene como objetivo conocer las posiciones y orientaciones de las cámaras respecto a un sistema de referencias del mundo. Esto se consigue observando imágenes 2D desde múltiples cámaras sincronizadas y asociando la posición de marcadores de calibración conocidos desde cada cámara mediante triangulación. A continuación se recogen los pasos recomendados para calibrar correctamente el sistema.

1. Abra el software Motive disponible en el equipo de trabajo del laboratorio. Las cámaras deberían mostrar en este momento un color azulado (Figura B.7b). En caso contrario, asegúrese de que las cámaras y el equipo de trabajo están correctamente conectadas al *switch*, y este dispositivo se encuentra perfectamente alimentado y encendido.

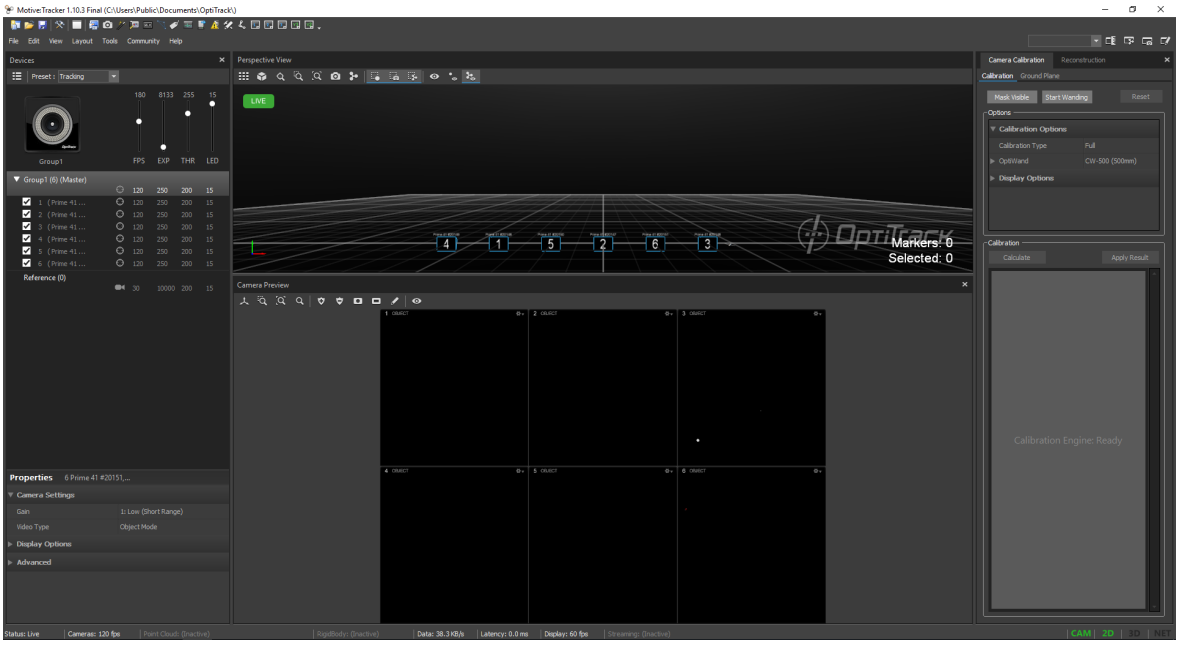


Figura B.6: Pantalla por defecto al iniciar el software Motive.

2. El software le debería abrir por defecto la pantalla presentada en la Figura B.6. No se conoce la posición y orientación de las cámaras, y éstas aparecen visualizadas en una línea recta en la vista 3D (*Perspective View*). Si en caso contrario se ha cargado una configuración del sistema antigua, puede eliminarla pulsando *Edit* → *Reset Application Settings* en la barra de herramientas.

3. En la vista multi-cámara 2D (*Camera Preview*) puede visualizar las imágenes capturadas por las cámaras. Puede cambiar el tipo de las imágenes capturadas. Por defecto está en el modo objeto que detecta marcadores. Puede probar a visualizar las imágenes mediante el modo escala de grises o el modo MJPEG (Figura B.8). En estos modos de captura, la cámara debería mostrar un color naranja (Figura B.7d).

4. En la parte izquierda de la pantalla (*Devices*) se listan las cámaras conectadas y se permite cambiar diferentes parámetros de configuración de las mismas, como el ratio de *frames* por segundo o el tiempo de exposición. En las tres vistas comentadas, las cámaras vienen etiquetadas con un número identificador que corresponde con el número de la cámara física. Si selecciona una cámara, se resaltará en color amarillo dentro del programa, y la cámara en cuestión adoptará el mismo color amarillo (*Figura B.7c*). Puede utilizar esta funcionalidad para localizar de manera rápida la cámara deseada.

5. OptiTrack detecta como marcadores ciertos reflejos en el suelo o algunos objetos reflectantes. Para que no se consideren como marcadores durante la fase de calibración, se debe aplicar una máscara que ignore estos reflejos estáticos. Motive ofrece un

(a) Sin color. (b) Color azul. (c) Color amarillo. (d) Color naranja.

Figura B.7: Las cámaras muestran un color diferente en base al estado en el que se encuentran.

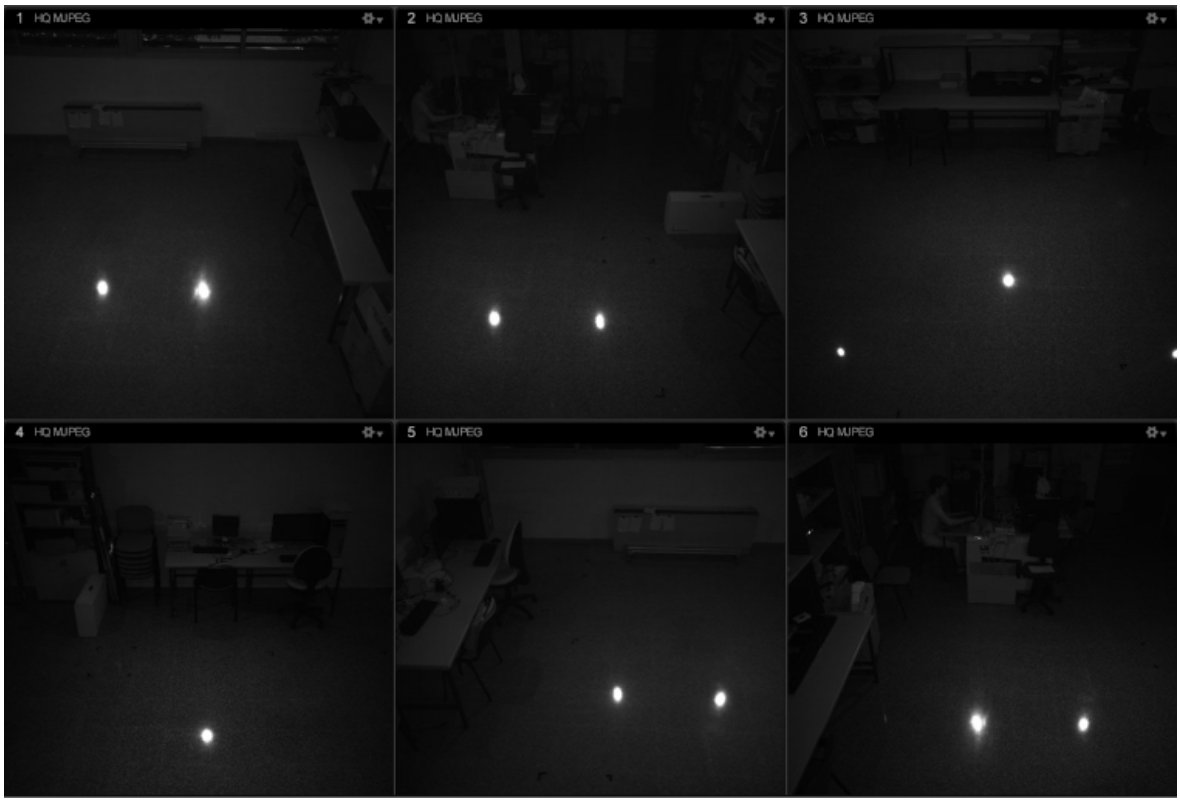


Figura B.8: Imágenes capturadas en tipo MJPEG de alta calidad desde las seis cámaras del sistema OptiTrack instalado.

algoritmo para la detección y generación automática de una máscara sobre estos extraños reflejos. Es posible que esta máscara generada no sea mínima, por ello Motive también ofrece funciones para crear la máscara manualmente desde píxeles concretos, regiones rectangulares o regiones circulares. En cualquier caso, se recomienda pulsar el botón antes de realizar este proceso para eliminar posibles máscaras anteriores.

Nota: *La información en las regiones enmascaradas no se utiliza para la triangulación durante el proceso de wanding. Un uso excesivo de la máscara puede inducir frecuentes oclusiones de los marcadores. Por esta razón, es altamente recomendable apartar del volumen de captura, o simplemente ocultar, todos los objetos reflectantes que se puedan. De igual manera, usualmente se tiene ya preparada la varilla de calibración para la siguiente fase, quedando sus marcadores visibles por las cámaras. Se recomienda ocultar la varilla y otros posibles marcadores durante esta fase de enmascaramiento.*

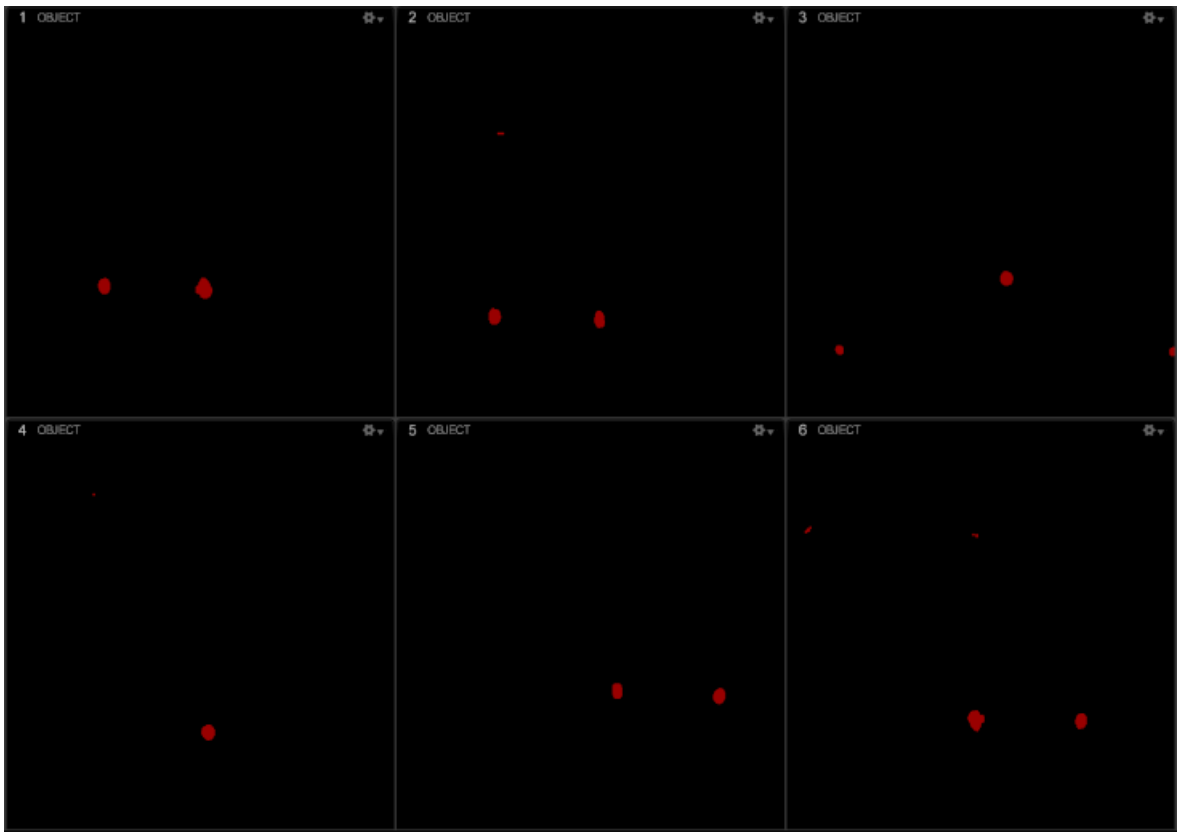


Figura B.9: Máscara aplicada de forma automática a las imágenes capturadas por las cámaras del sistema OptiTrack para ignorar los reflejos estáticos considerados como marcadores durante la fase de calibración del sistema (en color rojo).

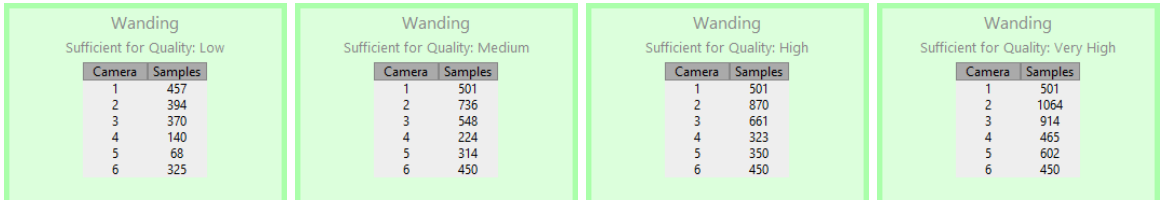
6. El siguiente paso es realizar el proceso de *wanding*. Para ello, primero hay que seleccionar en la parte derecha de la pantalla (*Camera calibration*) la varilla de calibración que se va a utilizar. Seleccione en *OptiWand* el valor *Micron Series* y escriba en *Wand Length (mm)* la longitud exacta de la varilla de calibración que va a utilizar. Esta longitud viene identificada en la parte posterior de la varilla con una precisión sub-milimétrica. La varilla de calibración disponible en el laboratorio tiene una longitud de 249,848 mm (Figura B.10). También puede seleccionar el tipo de calibración que va a realizar (*Calibration Type*). Seleccione el tipo *full* para calibrar las cámaras desde cero, descartando cualquier información anterior acerca de la posición de las cámaras o distorsión de las lentes. Para realizar una calibración más rápida seleccione *refine*, ejecutando solamente un refinamiento de una configuración anterior. Por último, la opción *visual* únicamente visualizará la solución calculada sin aplicar la calibración. Puede utilizar esta opción cuando quiera verificar la calidad de la configuración actual, comparando los resultados obtenidos. Pulse el botón *Start wanding* para empezar el proceso. Las cámaras no deberían mostrar ningún color en este momento (Figura B.7a).

Figura B.10: Longitud con precisión sub-milimétrica de la varilla de calibración CWM-250 disponible en el laboratorio de despliegue del sistema OptiTrack.

7. Desplace la varilla de calibración dentro del volumen de captura: este proceso se denomina *wanding*. Un mayor número de muestras tomadas por las cámaras supondrá una mejor calibración del sistema. Motive va informando en todo momento de la calidad de la calibración y el número de muestras obtenidas en cada cámara (Figura B.11). Se recomienda alcanzar el nivel de calibración *Very High* y una cantidad de muestras equitativa en todas las cámaras, alcanzando las tres mil muestras en cada una. Se expone una lista de consideraciones para obtener una mejor calibración del sistema.

   – Haga movimientos lentos con la varilla de calibración. Realizar movimientos rápidos emborronará los marcadores en las imágenes capturadas, distorsionando los cómputos de sus centroides y reduciendo así la calidad de la calibración.
   – Evite la oclusión de los marcadores de la varilla de calibración.
   – Mueva la varilla de calibración en un volumen tri-dimensional. Puede realizar movimientos en ocho con el brazo para conseguir muestras por todo el volumen y con gran variedad de orientaciones.
   – Incrementar el proceso de *wanding* en el volumen objetivo de captura mejorará la precisión de la calibración en esta zona.
   – Empezar y parar el proceso de *wanding* con la varilla dentro del volumen de captura evita obtener muestras raras fuera del volumen cuando se entra y se sale.

   Se muestra en la Figura B.12 las estadísticas finales de un proceso de *wanding* para una calibración recomendada del sistema de alta calidad.



(a) Calidad baja. (b) Calidad media. (c) Calidad alta. (d) Calidad muy alta.

Figura B.11: Niveles de la calidad de la calibración del sistema OptiTrack.

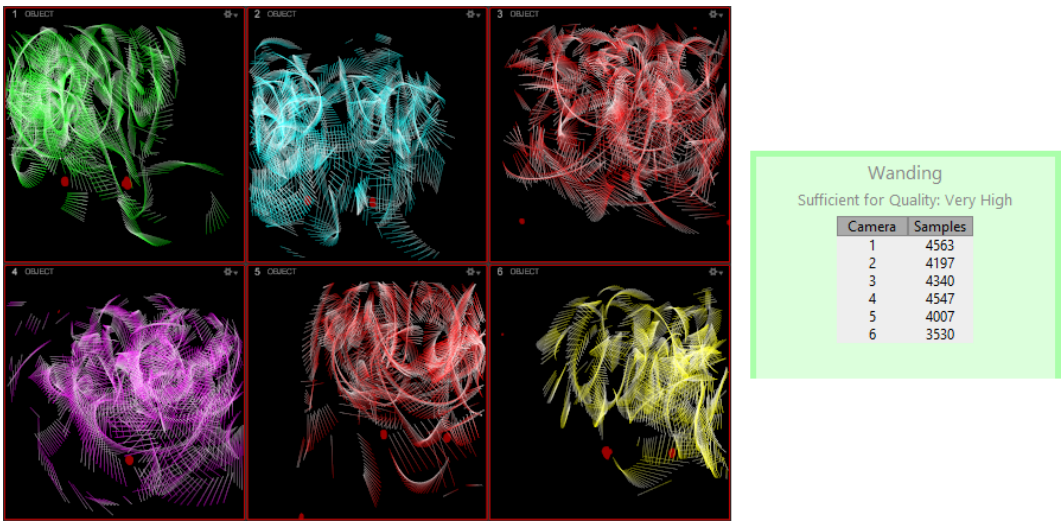


Figura B.12: Estadísticas finales de un proceso de *wanding* recomendado. A la izquierda, se visualiza el *tracking* de la varilla de calibración de manera uniforme por las seis cámaras del sistema. A la derecha, el número elevado de muestras obtenidas en cada cámara que permite conseguir una excelente calidad de calibración.

8. Para terminar el proceso de *wanding* presione el botón *Calculate*. La posición y orientación de las cámaras se calculan mediante un algoritmo de optimización minimizando el error geométrico de las posiciones de los marcadores muestreados. La pantalla debería tener una apariencia como la Figura B.13. Se visualiza el seguimiento de la varilla de calibración tanto en la vista 3D (*Perspective View*) como en cada una de las seis cámaras (*Camera Preview*). En el centro de la pantalla se le debe haber abierto un cuadro informativo de los resultados de la calibración obtenida y varias medidas de error. Pulse *Apply* para aplicar la calibración calculada.

B.3.2. Posición y orientación del suelo

En la calibración anterior se calculan la posición y orientación de las cámaras sobre un sistema de referencia desconocido (Figura B.14). Puede cambiar esta calibración a un sistema de referencia conocido, y conseguir así que el origen se encuentre en el suelo del laboratorio (Figura B.17). Para ello se va a utilizar el cuadrado de calibración CS-200, marcando éste el origen del nuevo sistema de referencia.

Posicione el cuadrado de calibración CS-200 en donde quiera que se encuentre el origen de referencia. Se recomienda situarlo en el centro del volumen de captura, y asegurarse de que está visible por todas las cámaras del sistema. A continuación, calibra el cuadrado ajustando las dos tuercas de las esquinas (Figura B.15). En el software Motive, ponga el valor 19 en el campo *Vertical Offset (mm)* dentro de la sección *Ground Plane Calibration Square*. Este valor indica la distancia entre los marcadores del cuadrado y el suelo, con un signo positivo por su sentido hacia arriba. Pulse *Set Ground Plane* para configurar el nuevo sistema de referencia.

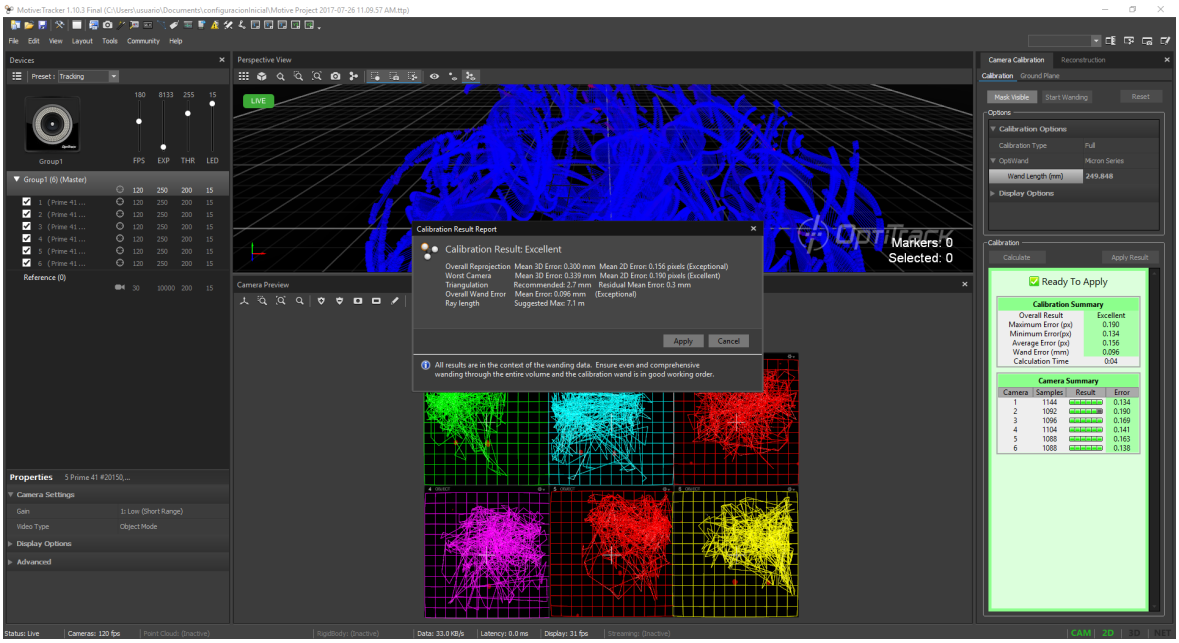


Figura B.13: Pantalla final tras la calibración de la posición y orientación de las cámaras del sistema OptiTrack. En el centro de la misma aparece un cuadro informativo de los resultados de la calibración calculada.

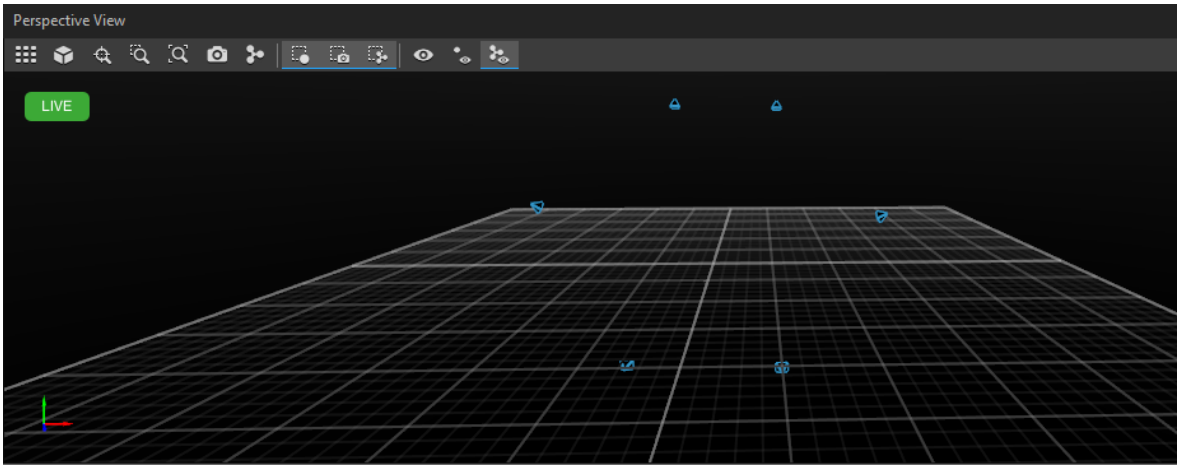


Figura B.14: Posición y orientación de las cámaras del sistema OptiTrack calibradas sobre un sistema de referencia desconocido.

(a) Mala calibración. (b) Buena calibración.

Figura B.15: Calibración del cuadrado CS-200. La burbuja debe estar centrada entre los marcadores para conseguir una buena calibración.

B.4. Grabación de una secuencia

Después de calibrar el sistema puede empezar a grabar una secuencia. Usualmente, se quiere capturar objetos, a los que se le incorporan marcadores y forman un cuerpo rígido. Ejemplos frecuentes de objetos para capturar con este sistema pueden ser un cuerpo humano o un dron (Figura B.16). En Motive, un cuerpo rígido son tres o más marcadores interconectados con cada uno añadiendo la asunción de que los objetos son indeformables, es decir, la relación espacial entre los marcadores permanece constante.

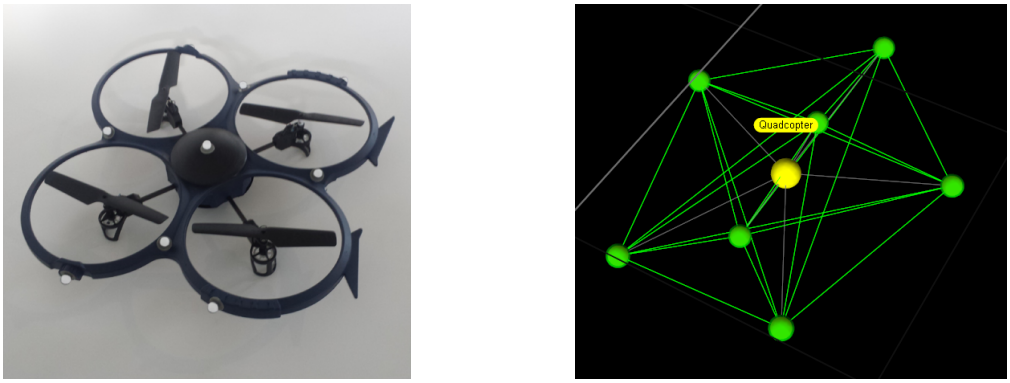


Figura B.16: En la imagen izquierda un dron, mostrando un ejemplo recomendado de posicionamiento de los marcadores en un objeto. En la imagen derecha, el correspondiente cuerpo rígido en Motive. En color verde, los marcadores capturados; y en color amarillo, el pivote del cuerpo rígido.

El mínimo número de marcadores para formar un cuerpo rígido son tres, aunque se recomienda utilizar cuatro siempre que sea posible. Motive deja añadir hasta un máximo de veinte marcadores. Se recomienda posicionar los marcadores lo más esparcidos posibles dentro del cuerpo rígido, consiguiendo de esta manera detectar pequeños movimientos en la orientación del objeto y mejorando así la precisión de la captura. Se recomienda también cubrir las superficies reflectantes del objeto con materiales no reflectantes, y posicionar los marcadores en el exterior del objeto para que las cámaras puedan capturarlos fácilmente.

Para crear un cuerpo rígido en Motive, seleccione los marcadores correspondientes en la vista 3D. Seguidamente, haga clic derecho en esta vista y seleccione *Rigid Body* → *Create From Selected Markers*. También puede realizarlo utilizando la *Hotkey*, por defecto *Ctrl + T*.

Si quiere realizar el seguimiento de múltiples objetos, es recomendable que sean únicos. Esta cualidad de unicidad se basa en la figura del posicionamiento de los marcadores y las distancias individuales entre los mismos. Se recomienda así distinguir los cuerpos rígidos en estos dos aspectos, consiguiendo no solamente una reducción de la computación requerida, si no también una mejora en la estabilidad del seguimiento al no etiquetarlos erróneamente. De todas formas, Motive permite realizar el seguimiento de objetos no únicos fijándose en

la historia de la trayectoria de éstos. Para ello, debe especificar un valor falso de la variable *Unique* en las propiedades del objeto *Properties* → *General Settings*.

Para grabar una secuencia una vez tiene configurados los cuerpos rígidos, seleccione el modo *Live* en la sección *Timeline* (Figura B.17). La captura se almacena en un fichero *take* (extensión *.tak*). En la derecha del botón de captura se listan los tipos de dato que se quiere almacenar. El tipo 2D almacena las imágenes de objetos en cada cámara, mientras que el tipo 3D añade información de los marcadores reconstruidos en 3D. El tipo JT se utiliza para capturar esqueletos, no abordados en esta guía. Un fichero *take* se puede exportar en formato tabular *csv* y *c3d*. Cada fila en estos formatos corresponde a un *frame* de la secuencia capturada. Para cada *frame* de la secuencia se almacenan la posición y orientación de los cuerpos rígidos y de los marcadores capturados. En el modo *Edit* puede visualizar y procesar las secuencias grabadas.

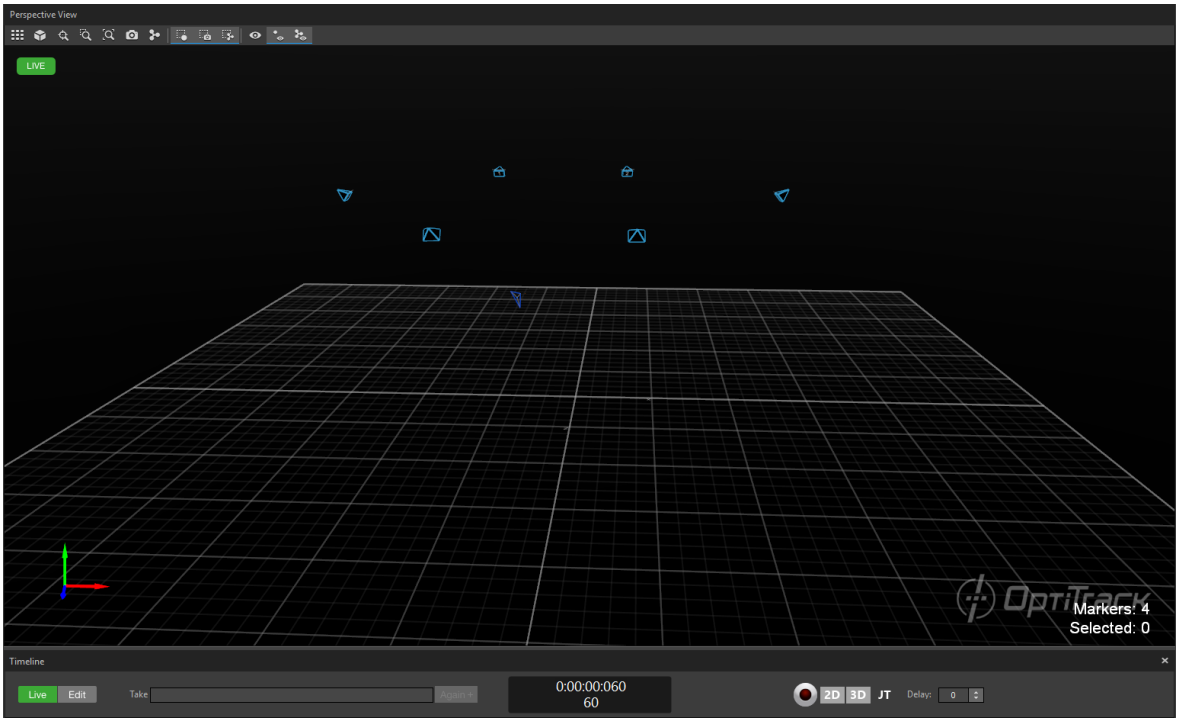


Figura B.17: Pantalla de grabación de una secuencia con el sistema Optitrack. La posición y orientación de las cámaras, en color azul claro, están calibradas respecto a un sistema de referencias en el suelo del laboratorio. En el volumen de captura se está reconociendo y grabando un cuerpo rígido, en color azul oscuro.

Anexo C

Actas de reuniones e informes

En el presente anexo se recogen las actas de las reuniones celebradas a lo largo del proyecto y los informes intermedios realizados con el objetivo de la gestión del mismo.

Se presenta en la Figura C.1 el calendario del periodo de realización del proyecto, marcando las fechas de las reuniones celebradas.

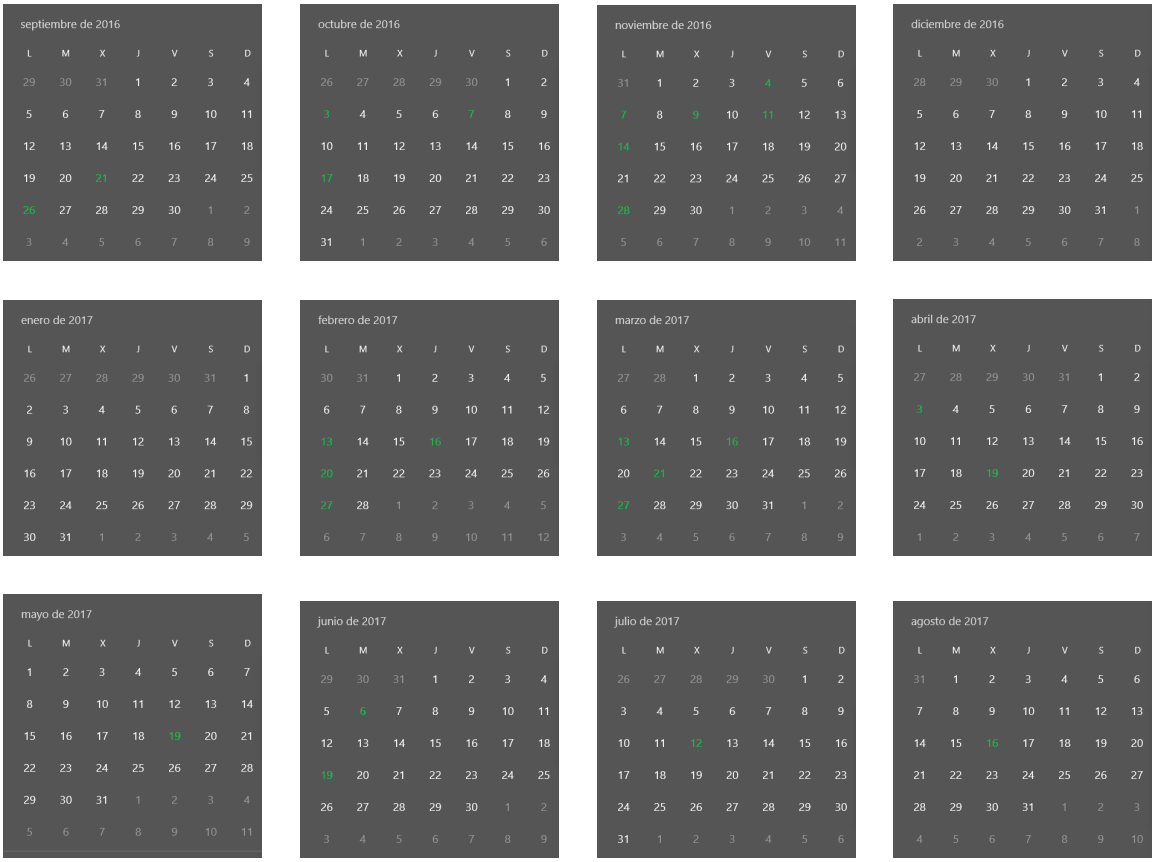


Figura C.1: Calendario abarcando el periodo de realización del proyecto, marcando en color verde las fechas en las que se ha celebrado una reunión.

Acta de la primera reunión del proyecto SLAM visual de alta precisión para AR/VR

En Zaragoza, siendo las 11:10 horas del lunes día 21 de septiembre de 2016, se reúne por primera vez el equipo de investigación del proyecto *SLAM visual de alta precisión para AR/VR*.

Asistentes:

Director y ponente del proyecto: Juan Domingo Tardós Solano.
Estudiante responsable del proyecto: Santiago Gil Begué.

Orden del Día:

1. Planificación de reuniones y gestión del proyecto.
2. *Google Scholar*.

1. Planificación de reuniones y gestión del proyecto.

Se plantea realizar un seguimiento semanal del proyecto con el fin de su supervisión y gestión. Se compatibilizan horarios entre el estudiante y el director, considerando finalmente la mejor propuesta organizar las reuniones semanales los lunes a las 12:00, para de esta manera poder encaminar el trabajo que se llevará a lo largo de la semana. Las reuniones tendrán lugar en el despacho del director, D.1.16 del edificio Ada Byron.

2. *Google Scholar*.

Es comentada la herramienta Google Scholar para poder acceder al contenido de *papers* de investigación. Es detallado asimismo la finalidad académica de esta herramienta, por lo que a veces es preciso estar conectado a la red de la Universidad de Zaragoza para poder acceder a la versión oficial y final de estos *papers* de investigación. Se aclara una vía de conexión a esta red: mediante red privada virtual (*VPN*), https://vpn.unizar.es/.

Tareas asignadas:

1. Empezar lectura del *paper* "ORB-SLAM: A Versatile and Accurate Monocular SLAM System" – *Santiago Gil Begué.*
2. Empezar lectura del *paper* "Large-Scale Direct SLAM for Omnidirectional Cameras" – *Santiago Gil Begué.*
3. Instalar ORB-SLAM2 en la zona de trabajo – *Santiago Gil Begué.*

Finalizados todos los puntos, se levanta la sesión a las 11:40.

Zaragoza, a 21 de septiembre de 2016.

Fdo. Director del proyecto.

Juan Domingo Tardós Solano

Fdo. Responsable del proyecto.

Santiago Gil Begué

Acta de la segunda reunión del proyecto SLAM visual de alta precisión para AR/VR

En Zaragoza, siendo las 10:30 horas del lunes día 26 de septiembre de 2016, se reúne por segunda vez el equipo de investigación del proyecto *SLAM visual de alta precisión para AR/VR*.

Asistentes:

Director y ponente del proyecto: Juan Domingo Tardós Solano.
Estudiante responsable del proyecto: Santiago Gil Begué.

Orden del Día:

1. Instalación y ejecución del proyecto ORB-SLAM2.
2. Vocabulario del proyecto.
3. Optimizaciones de ORB con OpenCV3.

1. Instalación y ejecución del proyecto ORB-SLAM2.

Se enseña la correcta instalación del proyecto ORB-SLAM2 en la zona de trabajo del estudiante, haciendo un seguimiento de todas las dependencias necesarias. El proyecto es ejecutado sobre el conjunto de datos TUM, obteniendo una correcta ejecución. Se asigna como tarea probar la correcta ejecución del proyecto sobre el conjunto de datos KITTI, y otros datos que impliquen cerrados de bucles.

2. Vocabulario del proyecto.

Tras parte de la lectura del *paper* "Large-Scale Direct SLAM for Omnidirectional Cameras", se resuelven y repasan ciertas dudas y aspectos referentes al proyecto:

Image alignment y *bundle adjustment*. Dos aproximaciones enfrentadas de cómo realizar el tracking. La primera de ellas utiliza la totalidad de la imagen capturada, por lo que cuenta con más información (todos los puntos y superficies densas). La segunda aproximación se basa en extraer solamente los puntos de interés (*keypoints*). ORB-SLAM2 utiliza la segunda aproximación.

El acrónimo *FoV* (*Field of View*), campo de visión. Una cámara omnidireccional es aquella que tiene un campo de visión mayor a 180 grados.

El acrónimo *VO* (*Visual Odometry*), odometría visual. Es el proceso de determinar la posición y orientación de un robot. Es así una versión simple de SLAM, reconstruyendo éste además el mapa visualizado (M de *Mapping*).

Distorsiones radiales en cámaras. Son comentados los tipos de distorsión de barril y de cojín. Paliar contra estas distorsiones se denomina rectificar: rectificar una imagen, cámaras rectificadas.

Los puntos homogéneos. Son explicados brevemente, así como el proceso de transformación de coordenadas normales a coordenadas homogéneas. Son utilizados para trabajar con sistemas lineales.

Las funciones de proyección y des-proyección π y π-1. Su utilización es obvia, mapear puntos del mundo real al modelo de cámara y viceversa. Se comenta la necesidad de que la función de des-proyección sea derivable.

La resolución angular. Es explicada brevemente, visualizando gráficamente el efecto de cómo se degrada esta resolución en los bordes de una imagen.

3. Optimizaciones de ORB con OpenCV3.

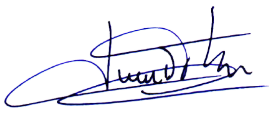
Se comenta el versionado de la librería OpenCV utilizada por ORB-SLAM2. Actualmente se llevan dos versiones en paralelo, la 2.4 y la 3, utilizando ORB-SLAM2 la primera de estas. Se detalla la posibilidad de mejora de adaptar el proyecto ORB-SLAM2 sobre la versión 3 de OpenCV, consiguiendo optimizaciones sobre los descriptores ORB. Queda así pendiente el estudio de esta posibilidad, una vez esté asentado el conocimiento sobre el proyecto por parte del estudiante.

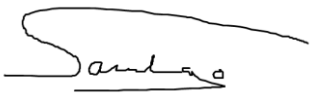
Tareas asignadas:

1. Profundizar lectura del *paper* "ORB-SLAM: A Versatile and Accurate Monocular SLAM System" – *Santiago Gil Begué.*
2. Finalizar lectura del *paper* "Large-Scale Direct SLAM for Omnidirectional Cameras" – *Santiago Gil Begué.*
3. Descargar el conjunto de datos KITTI y otros que impliquen cerrados de bucles, para ejecutar ORB-SLAM2 con estos datos – *Santiago Gil Begué.*

Finalizados todos los puntos, se levanta la sesión a las 11:20.

Zaragoza, a 26 de septiembre de 2016.





Fdo. Director del proyecto.

Juan Domingo Tardós Solano

Fdo. Responsable del proyecto.

Santiago Gil Begué

Acta de la tercera reunión del proyecto SLAM visual de alta precisión para AR/VR

En Zaragoza, siendo las 12:00 horas del lunes día 03 de octubre de 2016, se reúne por tercera vez el equipo de investigación del proyecto *SLAM visual de alta precisión para AR/VR*.

Asistentes:

Director y ponente del proyecto: Juan Domingo Tardós Solano.
Estudiante responsable del proyecto: Santiago Gil Begué.

Orden del Día:

1. Documentos adjuntos al proyecto.
2. Vocabulario del proyecto.
3. Primeras posibles optimizaciones.
4. Tipos de licencias.
5. Próxima reunión.

1. Documentos adjuntos al proyecto.

Se comenta la documentación que ha de ser enviada junto a la finalización del proyecto. Entre ella destaca la memoria final con una extensión de 20 páginas. Se propone también adjuntar como anexo la organización del equipo, incluyéndose las actas de reuniones, los esfuerzos recogidos a lo largo del proyecto, y una comparativa en forma de diagrama de Gantt de las planificaciones inicial y real del proyecto. Esta planificación inicial queda pactada de realizarla en la siguiente reunión de forma conjunta. El seguimiento real del proyecto se realizará mediante la plataforma Harvest.

También es comentada la documentación que ha de enviarse referente a la Beca de Colaboración concedida a este proyecto de investigación. Se visualizan las plantillas de los informes intermedio y final que han de entregarse.

2. Vocabulario del proyecto.

Tras parte de la lectura del *paper* "ORB-SLAM: A Versatile and Accurate Monocular SLAM System", se resuelven y repasan ciertas dudas y aspectos referentes al proyecto:

ORB: Oriented and Rotation BRIEF. Es un descriptor binario, una ristra de 256 bits, que compara la iluminación de unos píxeles definidos. Es así invariante a la iluminación (salvo saturaciones de la cámara, obviamente). FAST en cambio es para la extracción de puntos de interés invariantes a la rotación y orientación.

Dificultad de la inicialización del mapa ya que no se cuenta con información para conocer la profundidad de los puntos (en cámaras monoculares). Es comentado el algoritmo *RANSAC (Random sampling consensus)*. También el mínimo número de puntos necesarios que hagan match para esta inicialización, o localización global; 6 puntos, aunque 5 en la práctica debido al factor escala. De aquí toma el nombre el algoritmo *five-point*, aunque también se suele hacer con 7 u 8 puntos por prestaciones. En ORB-SLAM2 se realiza con 8 puntos.

El grafo de covisibilidad, el *Essential Graph*, y el *spanning tree*. Son también mostrados en un ejemplo visual para su mayor comprensión. A raíz de estos términos surge la optimización del *pose-graph*, que se realiza para compensar la deriva (*drift*) acumulada en un bucle (*loop*). Explicados los 7 grados de libertad, procedientes 3 de la posición, otros 3 de la orientación, y 1 último de la escala.

Image-to-image, *map-to-map* y *image-to-map*. Métodos para localización global una vez perdido el tracking. Se comenta el coste extra del método *map-to-map*, terminando en que el método actual utilizado en ORB-SLAM2 es *image-to-image*, lanzado sobre una bolsa de palabras. Esta bolsa de palabras es offline con palabras visuales generales. Se cuestiona la diferencia de un vocabulario general y uno específico, y la existencia de vocabularios online.

Escenas planas y no planas. Para la inicialización, se utilizan métodos que trabajan con homografías para las escenas planas; mientras que para no planas se utiliza el ya nombrado *five-point*.

Motion-only bundle adjustment (BA). Este término *motion-only* hace referencia a que los puntos están ya fijos, siendo sólo variables las poses (cámaras). Esto es debido a que el programa se compone de threads, suponiendo uno de ellos (*tracking thread*) que el mapa ya está bien formado, por lo que los puntos son fijos. Esto es hecho por eficiencia.

MapPoints y KeyFrames. Se detalla en gran medida estas estructuras de datos. Se cuestiona el por qué se guarda la media de los vectores desde los que se ha visualizado un punto, explicándose que es para descartar visualizar los puntos en un intervalo de +-60º. Se comenta que esto no es realizado en el proyecto anterior PTAM. También es comentada la distancia de *hamming*, utilizado en los descriptores ORB. Un KeyFrame se compone de la matriz de transformación, los parámetros intrínsecos (distancia focal y punto principal), y lista de los descriptores ORB.

3. Primeras posibles optimizaciones.

Se proponen unas posibles mejoras en el código tras una primera lectura del mismo por parte del estudiante:

Guardar los *mappoints* y *keyframes* en memoria estática, en vez de en memoria dinámica. La problemática de esto es que hay que pre-dimensionar el tamaño que se le quiere guardar, con los consiguientes errores de sobrepasar este tamaño.

Cambiar la política de *locks*. Se cuestiona la necesidad de exclusión mutua de todas las operaciones de la clase Map. Las operaciones de sólo lectura no serían necesarias de este bloqueo. Problemática de ser cuidadosos con este cambio, por lo que se espera a tener bien en conocimiento la totalidad del código del proyecto.

4. Tipos de licencias.

Se comenta brevemente los tipos de licencias de software existentes que pueden aparecer a lo largo de la realización del proyecto. Entre ellas son comentadas las licencias *GPL*, *BSD y LGPL*.

Se acuerda evitar el uso de código *GPL* ajeno a la Universidad de Zaragoza en la realización del proyecto. Para ello, el estudiante consultará al director antes de usar código externo.

5. Próxima reunión.

Se acuerda una siguiente reunión el viernes 07 de octubre a las 12:00, debido a que el lunes 10 de octubre correspondiente a la siguiente reunión semanal es no lectivo. Queda de esta manera adelantada la reunión con el fin de perseverar la regularidad semanal de las reuniones.

Tareas asignadas:

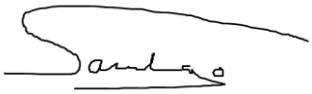
1. Finalizar lectura del *paper* "ORB-SLAM: A Versatile and Accurate Monocular SLAM System" – *Santiago Gil Begué.*
2. Finalizar lectura del *paper* "Large-Scale Direct SLAM for Omnidirectional Cameras" – *Santiago Gil Begué.*

Finalizados todos los puntos, se levanta la sesión a las 14:08.

Zaragoza, a 03 de octubre de 2016.

Fdo. Director del proyecto.

Juan Domingo Tardós Solano



Fdo. Responsable del proyecto.

Santiago Gil Begué

Acta de la cuarta reunión del proyecto SLAM visual de alta precisión para AR/VR

En Zaragoza, siendo las 11:53 horas del lunes día 07 de octubre de 2016, se reúne por cuarta vez el equipo de investigación del proyecto *SLAM visual de alta precisión para AR/VR*.

Asistentes:

Director y ponente del proyecto: Juan Domingo Tardós Solano.
Estudiante responsable del proyecto: Santiago Gil Begué.

Orden del Día:

1. Cronograma de actividades del proyecto.
2. Memoria del proyecto.
3. Citas y referencias: Estilo Chicago.
4. Control de esfuerzos: Harvest.
5. Política de back-ups.

1. Cronograma y actividades del proyecto.

Se realiza un cronograma de las actividades del proyecto [Fig. 1]. Se marca como fecha de finalización del proyecto el 20 de junio de 2017, teniendo que depositar tal día el proyecto en la página web de la Universidad de Zaragoza https://deposita.unizar.es/.

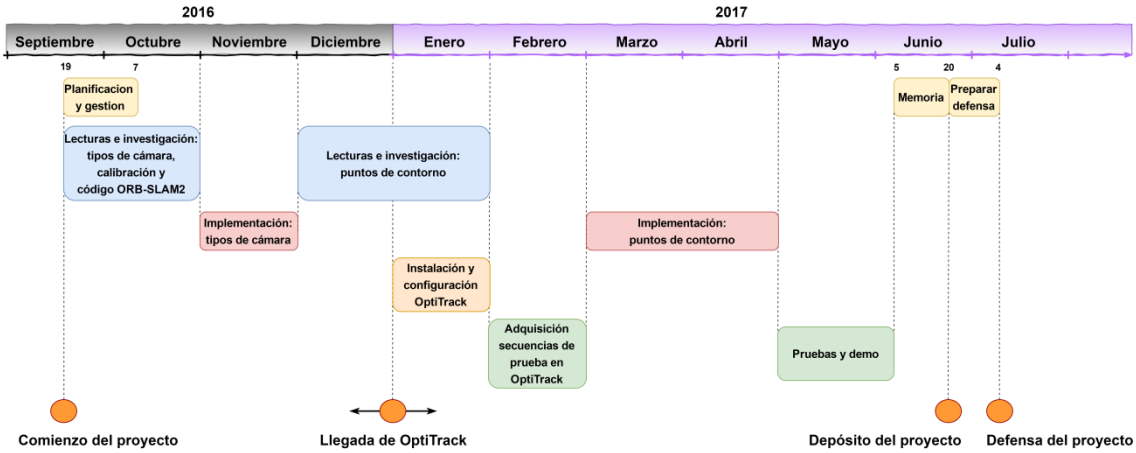


Fig. 1. Diagrama de Gantt del proyecto.

2. Memoria del proyecto.

Se debate qué sistema de composición de textos utilizar para el realizado de la memoria del proyecto. Destacan dos propuestas: Microsoft Word y LaTeX. Se concluye en la utilización de la aplicación Microsoft Word para el procesamiento de textos de este proyecto, con el principal motivo de su dominio por parte del estudiante.

3. Citas y referencias: Estilo Chicago.

Se define el estándar de citas y referencias que se cumplirá en toda la documentación referente al proyecto. Las citas y referencias que se realicen seguirán el estilo Chicago, definido en la guía de estilo *The Chicago Manual of Style [University of Chicago Press, 1906]*. *Una* versión online de la décimo sexta edición de esta guía se encuentra disponible en la página web http://www.chicagomanualofstyle.org/16/contents.html.

4. Control de esfuerzos: Harvest.

Se presenta en profundidad la herramienta Harvest que va a ser utilizada para la gestión de los esfuerzos invertidos en el proyecto.

5. Política de back-ups.

Se detalla una política de back-ups que se seguirá a lo largo del proyecto para prevenir posibles pérdidas de información referente al proyecto que se encuentre alojada en un servidor. Todos los datos alojados en algún servidor seguirán esta política de back-up (como por ejemplo, los datos de esfuerzos que se encuentran almacenados en el servicio Harvest).

La política definida impone una copia de seguridad semanal de los datos. Las copias de seguridad se almacenarán en la zona de trabajo personal del estudiante, así como en un disco duro externo.

Tareas asignadas:

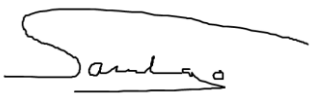
1. Lectura del segundo capítulo del libro *Computer Vision [Richard Szeliski, 2010]*.

Finalizados todos los puntos, se levanta la sesión a las 13:08.

Zaragoza, a 07 de octubre de 2016.

Fdo. Director del proyecto.

Juan Domingo Tardós Solano



Fdo. Responsable del proyecto.

Santiago Gil Begué

ROBOTICS
UNIVERSIDAD DE ZARAGOZA

Acta de la quinta reunión del proyecto SLAM visual de alta precisión para AR/VR

En Zaragoza, siendo las 12:15 horas del lunes día 17 de octubre de 2016, se reúne por quinta vez el equipo de investigación del proyecto *SLAM visual de alta precisión para AR/VR*.

Asistentes:

Director y ponente del proyecto: Juan Domingo Tardós Solano.
Estudiante responsable del proyecto: Santiago Gil Begué.

Orden del Día:

1. Vocabulario del proyecto.
2. Modelos de cámaras.
3. Espacio euclídeo y álgebra de Lie.
4. Revisión de calendario planificado.

1. Vocabulario del proyecto.

Tras la completa lectura del segundo capítulo del libro *Computer Vision [Richard Szeliski, 2010]*, se resuelven y repasan ciertas dudas y aspectos referentes al proyecto:

La distancia inversa *d*, con un principal significado de la inversa de la profundidad *z* de un punto 3D ($d = 1 / z$). Es utilizada ya que la distribución de probabilidad de *z* dada una imagen, con un paralaje muy pequeño entre dos posiciones de la cámara, tiene una función bastante incómoda de trabajar, tendiendo a infinito (puesto que una reducción del ángulo de paralaje entra las dos posiciones de la cámara puede enviar ambos rayos al infinito). Sin embargo, al invertir esta profundidad *z* obteniendo la distancia inversa *d*, se obtiene una distribución semejante a una gaussiana. Por ello, la inicialización de la profundidad de un punto con esta distribución es mucho más directa, cubriendo fácilmente toda la probabilidad de *d* y reajustando posteriormente para obtener el valor exacto. Sin embargo con la distribución de *z* esto es imposible, ya que como se ha remarcado, tiende a infinito. En ORB-SLAM2 se trabaja con la profundidad *z*, pero triangulando puntos 3D entre *KeyFrames*, es decir *Frames* que han experimentado un paralaje suficiente. Se deja una referencia al uso de esta distancia inversa en el campo de la robótica, en el *paper Inverse Depth Parametrization for Monocular SLAM [Javier Civera, Andrew J. Davison and J. M. Martínez Montiel*, 2008*]*.

El resto de conceptos de este segundo capítulo del libro no han ocasionado serias dudas al estudiante.

2. Modelos de cámaras.

Se detallan los tres modelos de cámaras omnidireccionales que se van a estudiar.

El primero es el presentado en el *paper Large-scale direct SLAM for omnidirectional cameras [David Caruso, Jakob Engel and Daniel Cremers*, 2015*]*. El segundo de ellos es el modelo arcotangente o *FoV*, utilizado en la anterior versión *Parallel Tracking and Mapping* (*PTAM)*. El tercero y último es el modelo polinómico de *Scaramuzza* (Tarea 2).

3. Espacio euclídeo y álgebra de Lie.

Se detalla que las operaciones de giro no son euclidianas, es decir, no se establecen dentro de un espacio euclídeo. Las operaciones de composición no son conmutativas: no se obtiene el mismo resultado si primero se gira sobre un eje y luego sobre otro que si se realiza en el orden contrario.

Aquí toma papel el álgebra de Lie. Se detalla la idea base de su funcionamiento y se deja una referencia para su estudio en profundización (Tarea 1). Principalmente se basa en que localmente las operaciones sí son conmutativas (ya que localmente la magnitud de las operaciones es pequeña, y el error cometido es así mismo muy pequeño). En la optimización del grafo esencial de ORB-SLAM2 esto es posible ya que las iteraciones se producen sobre un espacio local dentro del sistema entero.

4. Revisión del calendario planificado.

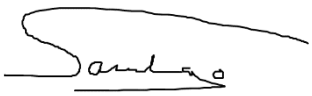
Se revisa el cronograma de actividades del proyecto para verificar el correcto seguimiento del mismo, y asegurar del cumplimiento de los plazos marcados.

Tareas asignadas:

1. Lectura del segundo capítulo *Preliminaries* de la tesis doctoral *Local Accuracy and Global Consistency for Efficient Visual SLAM [Hauke Strasdat,* 2012*] – Santiago Gil Begué.*
2. Lectura del *paper A toolbox for easily calibrating omnidirectional cameras [Davide Scaramuzza, Agostino Martinelli and Roland Siegwart,* 2006*] – Santiago Gil Begué.*
3. Lectura del *paper Accuracy of fish-eye lens models [Ciarán Hughes, Patrick Denny, Edward Jones, et al.,* 2010*] – Santiago Gil Begué.*

Finalizados todos los puntos, se levanta la sesión a las 13:05.

Zaragoza, a 17 de octubre de 2016.



Fdo. Director del proyecto.

Juan Domingo Tardós Solano

Fdo. Responsable del proyecto.

Santiago Gil Begué

Acta de la sexta reunión del proyecto SLAM visual de alta precisión para AR/VR

En Zaragoza, siendo las 11:55 horas del viernes día 04 de noviembre de 2016, se reúne por sexta vez el equipo de investigación del proyecto *SLAM visual de alta precisión para AR/VR*.

Asistentes:

Director y ponente del proyecto: Juan Domingo Tardós Solano.
Estudiante responsable del proyecto: Santiago Gil Begué.

Orden del Día:

1. Consolidación de términos de las lecturas realizadas.
2. Código ORB-SLAM2 y modelos de cámaras.

1. Consolidación de términos de las lecturas realizadas.

Tras las lecturas encargadas en la anterior reunión, se consolidan y repasan ciertos aspectos:

Respecto al segundo capítulo *Preliminaries* de la tesis doctoral *Local Accuraccy and Global Consistency for Efficient Visual SLAM [Hauke Strasdat, 2012]*, se comenta el entendimiento total por parte del estudiante de la teoría subyacente. Respecto a la utilidad práctica, se consolidan los términos de matriz de proyección entre un espacio no euclídeo y el plano tangente en un punto; y la actualización entre iteraciones de un algoritmo de optimización.

Respecto a los dos *papers* de cámaras omnidireccionales se comenta el entendimiento de ambos, pero sin una idea clara que materializar por parte del estudiante. Por ello, queda pendiente el total entendimiento de la *toolbox* propuesta por *Davide Scaramuzza* para la calibración de cámaras omnidireccionales.

2. Código ORB-SLAM2 y modelos de cámaras.

Se revisa por encima el código ORB-SLAM2 y se detectan las zonas de código encargadas de la calibración de los modelos de cámara. Se incide en este hecho, siendo necesario independizar este proceso en una clase aparte *Camera*. De esta manera, la adición de nuevos modelos de cámara será directa, además de tener toda la calibración de modelos de cámara encapsulada.

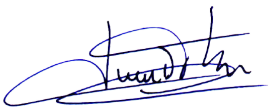
Para ello, se convoca la siguiente reunión el lunes 07 de noviembre a las 13.00 horas en la que estará presente el estudiante de doctorado desarrollador del proyecto ORB-SLAM2, Raúl Mur Artal. Con el previo estudio en profundidad del estudiante Santiago (Tarea 1), se consensuará el diseño de esta abstracción comentada, contando con el importante punto de vista de Raúl.

Tareas asignadas:

1. Estudio del código ORB-SLAM2, en especial el uso de los distintos modelos de cámara y su calibración – *Santiago Gil Begué.*

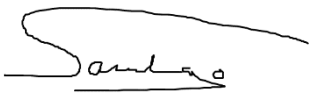
Finalizados todos los puntos, se levanta la sesión a las 12:18.

Zaragoza, a 04 de noviembre de 2016.



Fdo. Director del proyecto.

Juan Domingo Tardós Solano



Fdo. Responsable del proyecto.

Santiago Gil Begué

Acta de la séptima reunión del proyecto SLAM visual de alta precisión para AR/VR

En Zaragoza, siendo las 13:20 horas del lunes día 07 de noviembre de 2016, se reúne por séptima vez el equipo de investigación del proyecto *SLAM visual de alta precisión para AR/VR*.

Asistentes:

Director y ponente del proyecto: Juan Domingo Tardós Solano.
Estudiante de doctorado desarrollador de ORB-SLAM2: Raúl Mur Artal.
Estudiante responsable del proyecto: Santiago Gil Begué.

Orden del Día:

1. Diseño del modelo de cámara *fish-eye* en ORB-SLAM2.
2. Cámara *fish-eye* que se va a utilizar para el proyecto.

1. Diseño del modelo de cámara *fish-eye* en ORB-SLAM2.

Se comenta y debate las partes de ORB-SLAM2 que trabajan con puntos distorsionados y des-distorsionados, y que por lo tanto afectan al diseño de la adición del modelo de cámara *fish-eye*. En la inicialización monocular, tanto mediante homografía como con la matriz fundamental utilizando el algoritmo de 8 puntos, se asume un modelo *pihnole* sin distorsión, siendo necesario de esta manera des-distorsionar. En la relocalización, mediante el algoritmo *Perspective-n-Point (PnP)* utilizando un valor de n = 4 (*), se usan también coordenadas des-distorsionadas. En el cerrado de bucle (*loop closing*, sim3) solamente influye al calcular el error de re-proyección.

Se proponen dos posibles soluciones. Una primera sería la des-distorsión de los puntos obtenidos por la cámara *fish-eye*, siendo así una solución simple sin tener que adaptar el código actual. Otra segunda propuesta, para conseguir más precisión (objetivo del proyecto), es trabajar con puntos des-distorsionados en la inicialización y P4P, pero realizar el *Bundle Adjustment* (BA) con puntos distorsionados, así como el *matching* de puntos sobre la imagen distorsionada. Para ello, en el BA se debe pasar los jacobianos de la distorsión y la re-proyección del error a g2o. Se elige esta segunda propuesta.

(*) Se cuestiona, para trabajo futuro, utilizar P3P en vez del actual utilizado P4P. Se comenta el posible soporte de este algoritmo por las librerías OpenGV y OpenCV.

2. Cámara *fish-eye* que se va a utilizar para el proyecto.

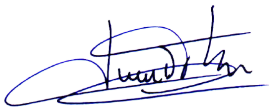
Se debate la cámara *fish-eye* que se va a utilizar para la realización del proyecto y validación de su correcto funcionamiento. Entre las propuestas está la utilización de una cámara *stereo DUO* ya en propiedad por el grupo de investigación; y la adquisición de una cámara nueva. La primera propuesta es descartada por no tener soporte para ROS, eligiendo así la adquisición de una cámara nueva. Finalmente se decide adquirir la cámara **uEye UI-3221LE-M-GL** (*Tarea 5*).

Tareas asignadas:

1. Configuración del proyecto ORB-SLAM2 en un entorno de desarrollo integrado – *Santiago Gil Begué.*
2. Instalación de ROS para procesar secuencias en tiempo real – *Santiago Gil Begué.*
3. Implementación del modelo de cámara *fish-eye* en ORB-SLAM2 – *Santiago Gil Begué.*
4. Implementación de la encapsulación del modelo de cámara en ORB-SLAM2 – *Santiago Gil Begué.*
5. Compra y adquisición de una cámara fish-eye uEye UI-3221LE-M-GL – *Juan Domingo Tardós Solano.*

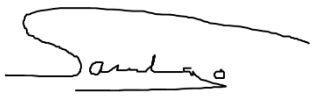
Finalizados todos los puntos, se levanta la sesión a las 14:20.

Zaragoza, a 07 de noviembre de 2016.



Fdo. Director del proyecto.

Juan Domingo Tardós Solano



Fdo. Responsable del proyecto.

Santiago Gil Begué

Acta de la primera reunión de desarrollo del proyecto SLAM visual de alta precisión para AR/VR

En Zaragoza, siendo las 11:00 horas del miércoles día 09 de noviembre de 2016, se reúne por primera vez el equipo de desarrollo del proyecto *SLAM visual de alta precisión para AR/VR*.

Asistentes:

Estudiante de doctorado desarrollador de ORB-SLAM2: Raúl Mur Artal.
Estudiante responsable del proyecto: Santiago Gil Begué.

Orden del Día:

1. Configuración del proyecto ORB-SLAM2 en un IDE.
2. Instalación y funcionamiento de ROS.

1. Configuración del proyecto ORB-SLAM2 en un IDE.

Se detalla el entorno de desarrollo integrado (IDE) que se va a utilizar para el desarrollo del proyecto. Al trabajar con el lenguaje de programación C++, dos son las propuestas de entornos de desarrollo especializados en este lenguaje: *QtCreator* y *CLion* (de *JetBrains*), ambos de uso gratuito. Se decanta por el primero, *QtCreator*, al haber sido utilizado por Raúl durante el desarrollo de ORB-SLAM2 y tener experiencia sobre el mismo.

Tras el intento de configuración del proyecto ORB-SLAM2 previo a la reunión por parte de Santiago, y con el resultado de una mala configuración, se detecta el error existente. La instalación de las dependencias de ORB-SLAM2 se había realizado como *root* (en sistema Unix), por lo que *QtCreator* no sabía encontrar tales dependencias. Se reinstalan las dependencias sin *root*, configurando correctamente esta vez el proyecto en el entorno de desarrollo.

Se traspasa la experiencia de Raúl a Santiago respecto al entorno de desarrollo *QtCreator*. Por ejemplo, la configuración de la compilación del proyecto (indicando con *–j* la compilación con todos los *cores* de la estación de trabajo) o el acceso directo a las cabeceras de los distintos ficheros del proyecto.

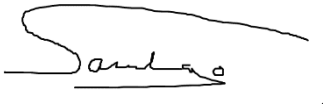
2. Instalación y funcionamiento de ROS.

Se instala correctamente ROS (*Robot Operating System*) y se explica la idea de su funcionamiento. Se utiliza para el procesamiento de secuencias en tiempo real por ORB-SLAM2. ROS permite la comunicación de mensajes entre los nodos que conforman el sistema. En ORB-SLAM2, esto se utiliza para comunicar las imágenes o *frames* que se van a procesar, las cuales están suscritas en un cierto canal.

Se detalla su funcionamiento mediante una secuencia de prueba. Para montar ROS, se realiza mediante el comando *$> roscore*. Para iniciar ORB-SLAM2 como nodo del sistema, se lanza el comando *$> rosrun ORB_SLAM2 Mono <vocabulary.txt> <settings.yaml>* en el caso de utilizar un sensor monocular. Para iniciar la secuencia grabada y almacenada en un fichero .bag, se ejecuta el comando *$> rosbag play [--pause] <secuencia.bag> [-s <segundo>]*, donde el parámetro -s es opcional, indicando el segundo de la secuencia desde el que se quiere empezar (desde el inicio por defecto).

Finalizados todos los puntos, se levanta la sesión a las 11:30.

Zaragoza, a 09 de noviembre de 2016.



Fdo. Responsable del proyecto.

Santiago Gil Begué

Acta de la segunda reunión de desarrollo del proyecto SLAM visual de alta precisión para AR/VR

En Zaragoza, siendo las 12:00 horas del viernes día 11 de noviembre de 2016, se reúne por segunda vez el equipo de desarrollo del proyecto *SLAM visual de alta precisión para AR/VR*.

Asistentes:

Estudiante de doctorado desarrollador de ORB-SLAM2: Raúl Mur Artal.
Estudiante responsable del proyecto: Santiago Gil Begué.

Orden del Día:

1. Encapsulación del modelo de cámara.
2. Automatización de pruebas para comprobar la precisión del sistema.

1. Encapsulación del modelo de cámara.

Se presenta la encapsulación del modelo de cámara diseñada y desarrollada. Se comenta su potencial, pero con algún desajuste con el diseño, como que la cámara encapsula atributos más allá de la geometría y proyección de la cámara (como los extractores de *features*). Se concluye en que una encapsulación total y correcta requiere un conocimiento más profundo del código, por lo que se decide que se realizará primeramente la implementación del modelo de cámara *fisheye*, implementando la encapsulación de los modelos seguidamente, una vez se tenga más experiencia sobre el código. Así, la versión ya implementada se guarda como referencia para el trabajo futuro.

2. Automatización de pruebas para comprobar la precisión del sistema.

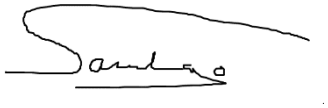
Se pregunta cómo comprobar el correcto funcionamiento del proyecto realizado. Se comenta que ORB-SLAM2 no es determinista debido a las condiciones de carrera de los *threads* del sistema, por lo que dos ejecuciones consecutivas (casi seguro) no devuelven la misma precisión. Sin embargo, las precisiones serán similares. De esta manera, no hay una solución determinista que sepa si el sistema está funcionando correctamente, pero sí una que sepa que el error de precisión no se ha visto alterado (en promedio).

Para comprobar el error de precisión en una secuencia, Raúl tiene desarrollado un script https://github.com/raulmur/evaluate_ate_scale que compara la trayectoria obtenida con el *ground truth* (aplicando a la trayectoria obtenida previamente una transformación de similaridad para ajustar las coordenadas globales a las del *ground truth*). Con este script, la automatización del cálculo de la precisión es directa, aplicando el script a diferentes secuencias y comprobar así el correcto funcionamiento del sistema en los diferentes escenarios.

Con todo esto, se propone realizar un script que llame al script de Raúl para varias secuencias definidas. Este script global se ejecutará a modo de test, tras la implementación de código, comprobando de esta manera que el sistema funciona correctamente con la nueva funcionalidad añadida.

Finalizados todos los puntos, se levanta la sesión a las 12:50.

Zaragoza, a 11 de noviembre de 2016.



Fdo. Responsable del proyecto.

Santiago Gil Begué

Acta de la octava reunión del proyecto SLAM visual de alta precisión para AR/VR

En Zaragoza, siendo las 12:06 horas del lunes día 14 de noviembre de 2016, se reúne por octava vez el equipo de investigación del proyecto *SLAM visual de alta precisión para AR/VR*.

Asistentes:

Director y ponente del proyecto: Juan Domingo Tardós Solano.
Estudiante responsable del proyecto: Santiago Gil Begué.

Orden del Día:

1. Funcionalidad vs refactorización global.
2. Secuencias de prueba para el cálculo de la precisión.
3. Modelo atan, calibración e implementación.

1. Funcionalidad vs refactorización global.

Se antepone la implementación de nueva funcionalidad en el sistema ante la refactorización global del código.

2. Secuencias de prueba para el cálculo de la precisión.

Se debate qué secuencias utilizar para probar que el sistema sigue funcionando correctamente con las nuevas funcionalidades añadidas. Se concluye que las secuencias *fr2* del *dataset* TUM ofrecen la garantía para confirmar la mantenibilidad de la precisión del sistema. Además, la duración de estas secuencias es corta, lo que facilita una rápida ejecución del test de precisión; así como que se necesita únicamente de una calibración. Se escogen las secuencias *xyz*, *desk1*, *desk2* y *office* del total de secuencias de este conjunto.

3. Modelo atan, calibración e implementación.

Se comenta el modelo atan y sus funciones de distorsión y des-distorsión de puntos que serán precisas implementar *(Tarea 2)* para dar soporte a cámaras *fish-eye*. Para comprobar el correcto funcionamiento del sistema con este modelo de cámara, también es necesario obtener la calibración de la cámara utilizada en las secuencias de prueba *(Tarea 1)*. Para ello, se precisa de un calibrador del modelo atan. Se concluye en el uso del calibrador utilizado por el sistema PTAM para el modelo de cámara atan.

Como trabajo futuro, se comenta brevemente el modelo polinómico presente en la *toolbox* de Scaramuzza. Ante la dificultad de una solución analítica del polinomio utilizado en este modelo, se propone la utilización de una *look-up table*.

Tareas asignadas:

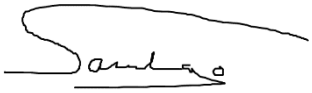
1. Obtención de un calibrador para el modelo atan y calibrar la cámara utilizada en las secuencias *fr2* del *dataset* TUM – *Santiago Gil Begué.*
2. Implementación del modelo atan – *Santiago Gil Begué.*

Finalizados todos los puntos, se levanta la sesión a las 12:46.

Zaragoza, a 14 de noviembre de 2016.

Fdo. Director del proyecto.

Juan Domingo Tardós Solano



Fdo. Responsable del proyecto.

Santiago Gil Begué

Primer informe intermedio del proyecto SLAM visual de alta precisión para AR/VR

En Zaragoza, a día 07 de marzo de 2017, se realiza el primer informe intermedio del proyecto *SLAM visual de alta precisión para AR/VR*, con el objetivo de gestionar el transcurso del proyecto y los objetivos cubiertos en base al calendario planificado.

1. Objetivos cubiertos.

El calendario planificado para el proyecto se presenta en la Figura 1.

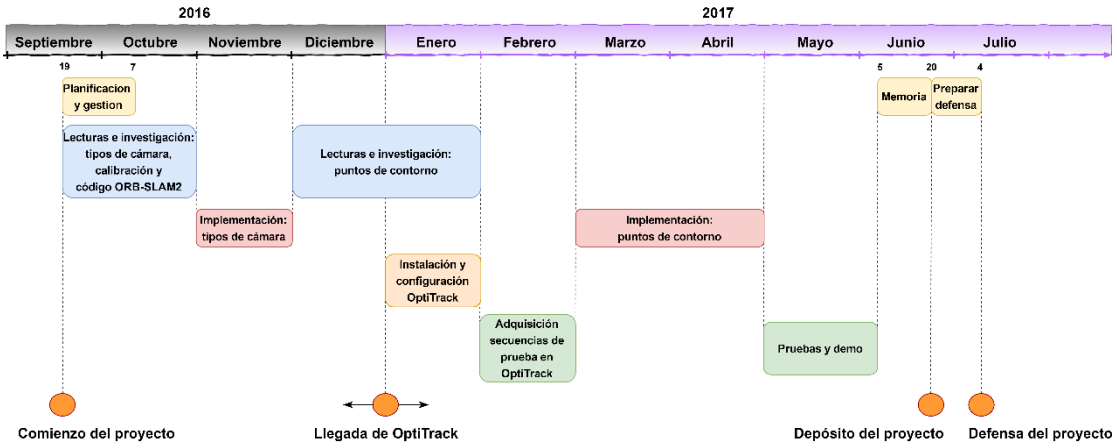


Fig. 1: Cronograma de tareas planificado para el proyecto SLAM visual de alta precisión para AR/VR.

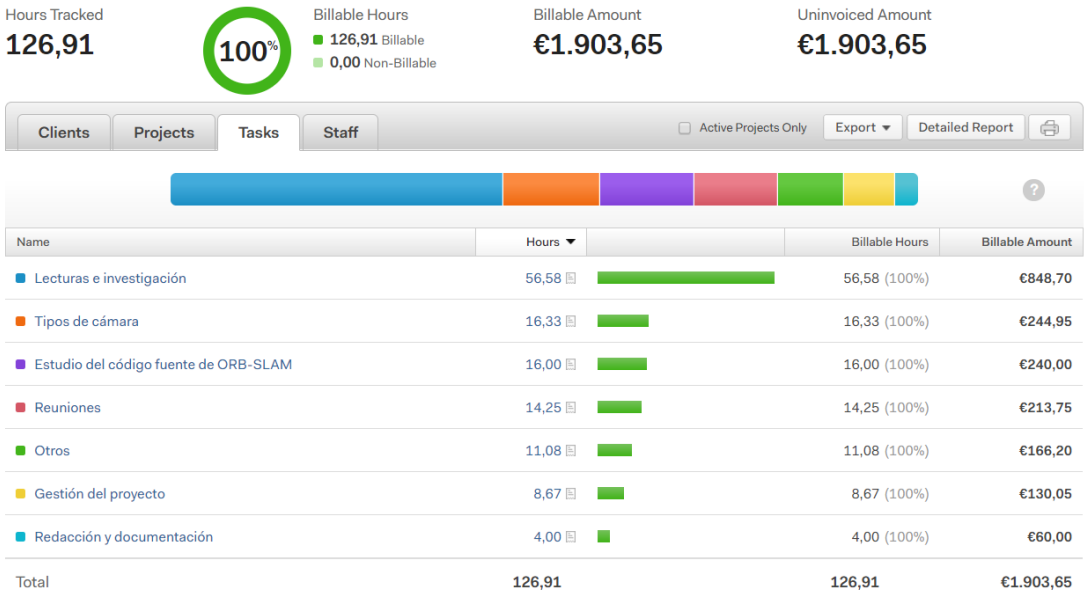
La primera tarea de planificación y gestión se ha realizado correctamente. El estudiante también ha completado la tarea de iniciación a la investigación mediante la lectura de toda la literatura relevante propuesta por el director. Se ha añadido satisfactoriamente el modelo de cámara *fish-eye* al software existente. Las pruebas exhaustivas de la nueva funcionalidad y la puesta en marcha del sistema de captura de movimiento Optitrack han sido retrasadas debido a la necesidad del estudiante de la atención académica a los exámenes de séptimo semestre.

2. Esfuerzo invertido.

Se presenta en la Figura 2 el detalle de todo el esfuerzo invertido en el proyecto hasta el día 07 de marzo de 2017. En total se ha invertido una cifra de 126.91 horas a lo largo del proyecto.

A modo de ejemplo en una situación profesional real, no vinculante al proyecto en curso, se ha asignado un sueldo por hora de trabajo de 15€. Esto permite al estudiante recopilar información a modo de histórico de costes tanto temporales como económicos con los que poder realizar una estimación más acertada de los costes de futuros proyectos.

Se puede observar que aproximadamente la mitad del esfuerzo invertido se ha centrado en la lectura y estudio de la literatura relevante para el proyecto. De esta manera, el estudiante ha podido obtener una formación y una iniciación a la investigación necesaria para la realización del proyecto. La otra mitad del esfuerzo invertido se reparte equitativamente en el resto de tareas.



Name	Hours	Billable Hours	Billable Amount
Lecturas e investigación	56,58	56,58 (100%)	€848,70
Tipos de cámara	16,33	16,33 (100%)	€244,95
Estudio del código fuente de ORB-SLAM	16,00	16,00 (100%)	€240,00
Reuniones	14,25	14,25 (100%)	€213,75
Otros	11,08	11,08 (100%)	€166,20
Gestión del proyecto	8,67	8,67 (100%)	€130,05
Redacción y documentación	4,00	4,00 (100%)	€60,00
Total	126,91	126,91	€1.903,65

Fig. 2: Detalle del reparto del esfuerzo invertido entre las diferentes tareas del proyecto.

3. Reuniones realizadas.

Las ocho primeras reuniones de equipo, así como las dos reuniones del equipo de desarrollo, se han recogido en sus actas correspondientes.

21/09/2016 – Primera reunión de equipo.
26/09/2016 – Segunda reunión de equipo.
03/10/2016 – Tercera reunión de equipo.
07/10/2016 – Cuarta reunión de equipo.
17/10/2016 – Quinta tercera reunión de equipo.
04/11/2016 – Sexta reunión de equipo.
07/11/2016 – Séptima reunión de equipo.
14/11/2016 – Octava reunión de equipo.

09/11/2016 – Primera reunión del equipo de desarrollo.
11/11/2016 – Segunda reunión del equipo de desarrollo.

A partir de la octava reunión, se ha decidido cesar la realización del conjunto de actas de las reuniones con el objetivo de minimizar el impacto burocrático del proyecto, y mejorar la eficiencia en la propia investigación. El contenido más importante de las reunioneses es apuntado por el estudiante, pero no digitalizado en un documento. La granularidad semanal de las reuniones se sigue perseverando, habiéndose reunido las siguientes fechas:

28/11/2016 – Novena reunión de equipo.
13/02/2017 – Décima reunión de equipo.
16/02/2017 – Undécima reunión de equipo.
20/02/2017 – Duodécima reunión de equipo.
27/02/2017 – Décimo tercera reunión de equipo.

4. Recursos utilizados.

El laboratorio 1.07 del edificio Ada Byron en la facultad de Ingeniería y Arquitectura de la Universidad de Zaragoza se ha establecido como estación de trabajo del estudiante y zona de despliegue del sistema OptiTrack. Se ha instalado y hecho uso del software ORB-SLAM2, propiedad del Grupo de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza.

Para la iniciación hacia la investigación se ha utilizado la literatura más relevante:

- ORB-SLAM: A Versatile and Accurate Monocular SLAM System [Raúl Mur-Artal, J. M. M. Montiel and Juan D. Tardós, 2015].
- Large-Scale Direct SLAM for Omnidirectional Cameras [David Caruso, Jakob Engel and Daniel Cremers, 2015].
- Computer Vision [Richard Szeliski, 2010].
- Local Accuracy and Global Consistency for Efficient Visual SLAM [Hauke Strasdat, 2012].
- A toolbox for easily calibrating omnidirectional cameras [Davide Scaramuzza, Agostino Martinelli and Roland Siegwart, 2006].
- Accuracy of fish-eye lens models [Ciarán Hughes, Patrick Denny, Edward Jones, et al., 2010].
- SVO: Semi-Direct Visual Odometry for Monocular and Multi-Camera Systems [Christian Foster, Zichao Zhang, Michael Gassner and Davide Scaramuzza, 2016].
- Direct Sparse Odometry [Jakob Engel, Vladlen Koltun and Daniel Cremers, 2016].
- A Photometrically Calibrated Benchmark For Monocular Visual Odometry [Jakob Engel, Vladyslav Usenko and Daniel Cremers, 2016].
- Computer Vision: A Modern Approach [Jean Ponce and David Forsyth, 2003].
- Edge Landmarks in Monocular SLAM [Ethan Eade and Tom Drummond, 2009].

Segundo informe intermedio del proyecto SLAM visual de alta precisión para AR/VR

En Zaragoza, a día 19 de abril de 2017, se realiza el segundo informe intermedio del proyecto *SLAM visual de alta precisión para AR/VR*, con el objetivo de gestionar el transcurso del proyecto y los objetivos cubiertos en base al calendario planificado.

1. Objetivos cubiertos.

El calendario planificado para el proyecto se presenta en la Figura 1.

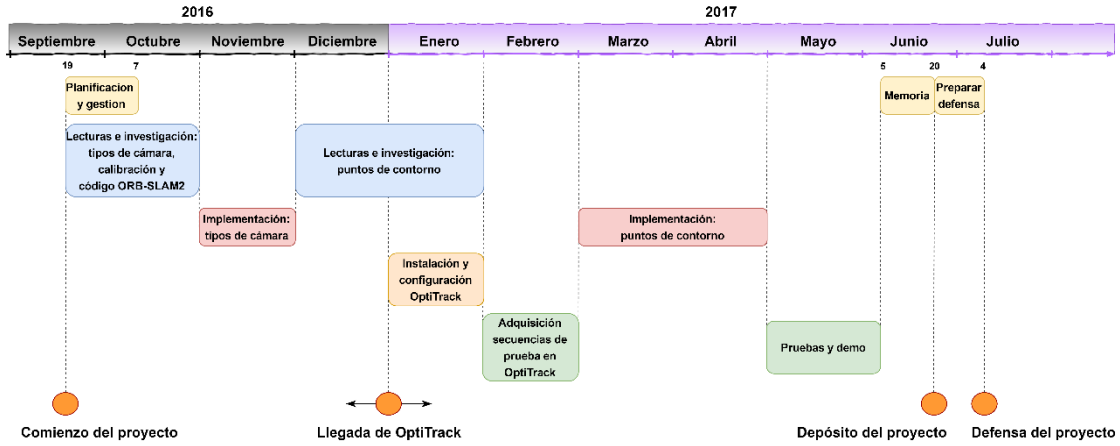


Fig. 1: Cronograma de tareas planificado para el proyecto SLAM visual de alta precisión para AR/VR.

- **Planificación y gestión:** Esta tarea se ha realizado correctamente. Como resultado, se ha obtenido la descripción del proyecto y los objetivos del mismo, planificados en un diagrama de tareas, como se presenta en el apartado anterior.

- **Lecturas e investigación:** El estudiante ha completado las tareas de iniciación a la investigación mediante la lectura de toda la literatura relevante propuesta por el director. Para la investigación de tipos de cámara y calibración se han estudiado las publicaciones [1] a [7]. La investigación de puntos de contornos se ha afianzado mediante la lectura de las publicaciones [8] a [15]. El código del software ORB-SLAM2 se ha estudiado y comprendido correctamente, ayudado de las publicaciones [1] y [2].

- **Implementación:** Se ha añadido correctamente el modelo de cámara *fish-eye* al software ORB-SLAM2. Se ha realizado satisfactoriamente un programa que realiza el seguimiento y emparejamiento de puntos de contorno. Estos puntos se han obtenido con el operador de Canny, aplicándoles dos filtros de umbralización del gradiente y supresión de no máximos locales. Los puntos de contornos elegidos, denominados *edgelets*, se eligen como los contornos de máximo gradiente de las regiones de un tablero superpuesto a la imagen de dimensiones configurables. El seguimiento de *edgelets* asume un pequeño paralaje entre dos imágenes consecutivas de una secuencia de vídeo, lo que permite buscar cada *edgelet* en una zona próxima, elegida experimentalmente de tamaño 31x31. Actualmente se está trabajando en la integración de este código en ORB-SLAM2 que permita mejorar la precisión de este software.

- **Optitrack:** Las tareas relacionadas con OptiTrack, tanto su instalación y configuración como la adquisición de secuencias en este sistema de captura de movimiento, han sido pospuestas como motivo de un retraso administrativo en la reestructuración de la unidad eléctrica del laboratorio de despliegue.

2. Re-planificación del proyecto

No ha sido posible la instalación y configuración del sistema de captura de movimiento Optitrack debido a la necesidad de una nueva distribución de la instalación eléctrica del laboratorio de despliegue. Se ha comentado desde el equipo técnico de la facultad que se le dará una solución en los próximos meses. De esta manera, y al no causar ningún problema en el responsable y director del proyecto, se ha realizado una re-planificación del mismo (Figura 2), marcando como fecha de finalización el 28 de agosto de 2017.

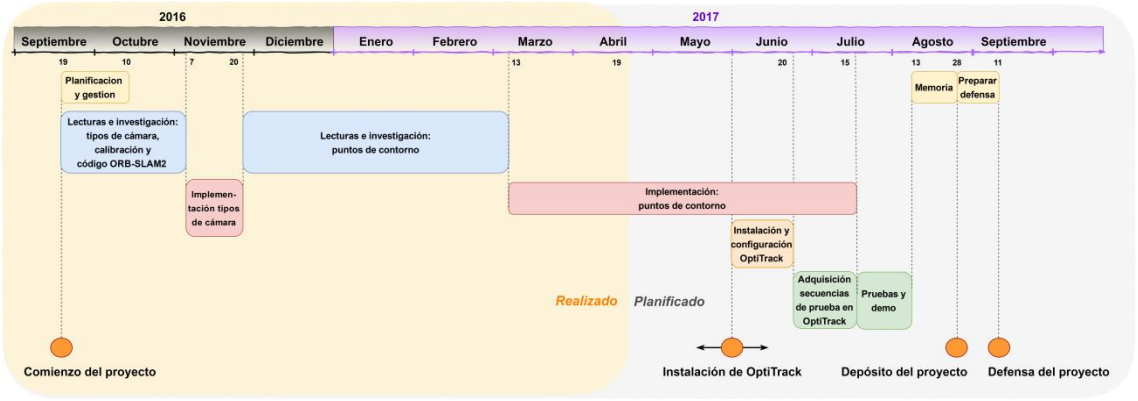


Fig. 2: Cronograma de tareas re-planificado para el proyecto SLAM visual de alta precisión para AR/VR.

2. Esfuerzo invertido.

La Figura 4 corresponde al desglose del esfuerzo invertido desde el comienzo del proyecto. En total se ha invertido una cifra de 149.24 horas a lo largo del mismo. En el cómputo total, el esfuerzo invertido en tareas de implementación es aproximadamente un tercio del esfuerzo invertido en lecturas e investigación, lo que refleja la todavía falta de implementación del grueso del proyecto.

A modo de ejemplo en una situación profesional real, no vinculante al proyecto en curso, se ha asignado un sueldo por hora de trabajo de 15€. Esto permite al estudiante recopilar información a modo de histórico de costes tanto temporales como económicos con los que poder realizar una estimación más acertada de los costes de futuros proyectos.

3. Reuniones realizadas.

Se han celebrado las siguientes seis reuniones.

13/03/2017 – Décimo cuarta reunión de equipo.
16/03/2016 – Décimo quinta reunión de equipo.
21/03/2016 – Décimo sexta reunión de equipo.
27/03/2016 – Décimo séptima reunión de equipo.
03/04/2016 – Décima octava reunión de equipo.
19/04/2016 – Décimo novena reunión de equipo.

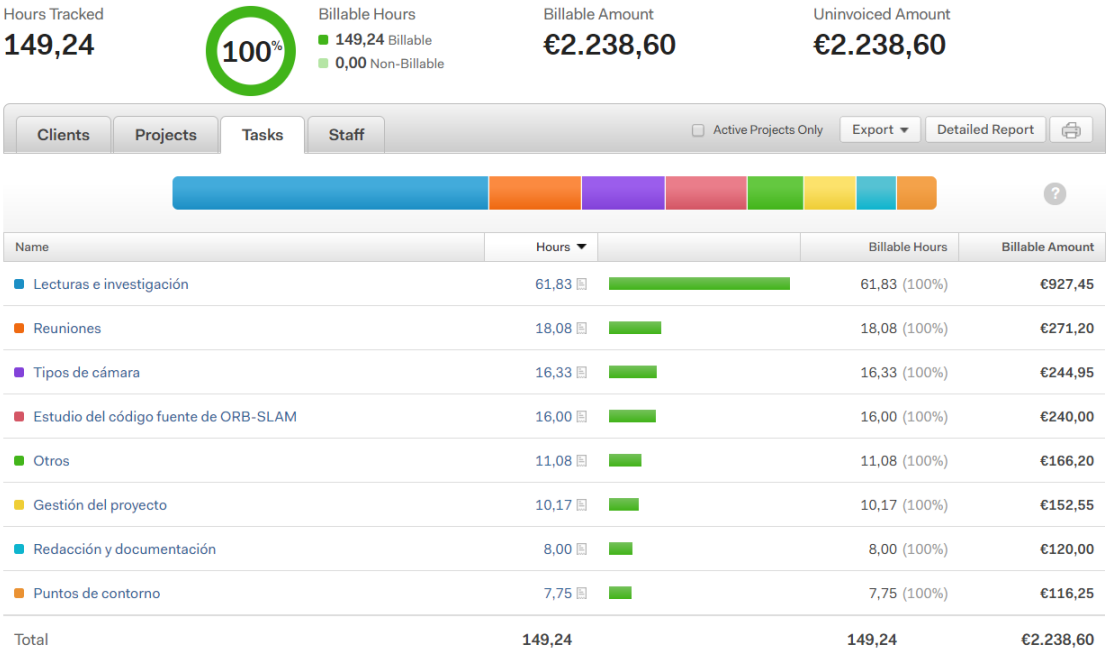


Fig. 3: Detalle del reparto del esfuerzo invertido entre las diferentes tareas del proyecto desde el comienzo del mismo.

4. Recursos utilizados.

Todo el desarrollo de funcionalidad acerca de la extracción de puntos de interés y demás tipos de algoritmos de visión por computador ha sido implementado utilizando la librería gratuita OpenCV.

Para seguir con la iniciación hacia la investigación se ha utilizado la literatura más relevante:

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

[2] Mur-Artal, R., & Tardos, J. D. (2016). ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. *arXiv preprint arXiv:1610.06475*.

[3] Caruso, D., Engel, J., & Cremers, D. (2015, September). Large-scale direct slam for omnidirectional cameras. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on* (pp. 141-148). IEEE.

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

[5] Strasdat, H. (2012). *Local accuracy and global consistency for efficient visual SLAM* (Doctoral dissertation, Ph. D. thesis, Imperial College, London).

[6] Scaramuzza, D., Martinelli, A., & Siegwart, R. (2006, October). A toolbox for easily calibrating omnidirectional cameras. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on* (pp. 5695-5701). IEEE.

[7] Hughes, C., Denny, P., Jones, E., & Glavin, M. (2010). Accuracy of fish-eye lens models. *Applied optics*, *49*(17), 3338-3347.

[8] Forster, C., Zhang, Z., Gassner, M., Werlberger, M., & Scaramuzza, D. (2017). Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, *33*(2), 249-265.

[9] Engel, J., Koltun, V., & Cremers, D. (2017). Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[10] Engel, J., Usenko, V., & Cremers, D. (2016). A photometrically calibrated benchmark for monocular visual odometry. *arXiv preprint arXiv:1607.02555*.

[11] Forsyth, D. A., & Ponce, J. (2003). A modern approach. *Computer vision: a modern approach*, 88-101.

[12] Eade, E., & Drummond, T. (2009). Edge landmarks in monocular SLAM. *Image and Vision Computing*, *27*(5), 588-596.

[13] Riemenschneider, H., Donoser, M., & Bischof, H. (2010, September). Using partial edge contour matches for efficient object category localization. In *European Conference on Computer Vision* (pp. 29-42). Springer Berlin Heidelberg.

[14] Klein, G., & Murray, D. (2008). Improving the agility of keyframe-based SLAM. *Computer Vision–ECCV 2008*, 802-815.

[15] Gomez-Ojeda, R., Moreno, F. A., Scaramuzza, D., & Gonzalez-Jimenez, J. (2017). PL-SLAM: a Stereo SLAM System through the Combination of Points and Line Segments. *arXiv preprint arXiv:1705.09479*.

5. Cambio en el sistema de composición de textos para la memoria del proyecto.

El estudiante ha recibido formación externa al proyecto acerca del sistema de composición de textos LaTeX. De esta manera, y dada la preferencia del estudiante de este sistema por su facilidad en la gestión de referencias a figuras y bibliografía, se ha elegido LaTeX como sistema de composición de textos para la memoria final del proyecto.