Trabajo Fin de Grado

Estimación de la escala en ORB-SLAM2 monocular mediante redes convolucionales profundas

Scale estimation in monocular ORB-SLAM2 using deep convolutional networks

Autor

Víctor Martínez Batlle

Director

Juan Domingo Tardós Solano

Grado en Ingeniería Informática

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2020

AGRADECIMIENTOS

Agradezco especialmente a Juan Domingo Tardós su gran predisposición para comentar el progreso de mi trabajo y sus consejos a lo largo de todo el desarrollo del proyecto.

También me gustaría reconocer la ayuda recibida del Ministerio de Educación y Formación Profesional por concederme la Beca de Colaboración que me ha permitido llevar a cabo parte de este proyecto dentro del Departamento de Informática e Ingeniería de Sistemas de la Universidad de Zaragoza.

RESUMEN

La localización y mapeo simultáneos o SLAM consiste en construir un mapa del entorno recorrido por un agente móvil a la vez que el agente es capaz de localizarse a sí mismo dentro del mapa. El sistema ORB-SLAM2 monocular de la Universidad de Zaragoza utiliza la información de una única cámara para cumplir este objetivo. Sin embargo, las técnicas de SLAM visual monocular basadas puramente en geometría presentan limitaciones ya que la escala del entorno no es observable. En consecuencia, ORB-SLAM2 monocular obtiene mapas con una escala desconocida, y sufre de deriva de la escala a lo largo de la trayectoria, lo que da lugar a mapas deformados e inconsistentes.

Por otro lado, resultados recientes demuestran que las redes neuronales convolucionales son capaces de estimar profundidad a partir de una única imagen. La red Monodepth2, al ser entrenada juntamente con secuencias monoculares y estéreo, es capaz de estimar para una imagen monocular, cuál sería la disparidad que obtendría un sistema estéreo virtual. Esa información sintética puede ser usada para obtener la profundidad real de la escena.

El objetivo de este trabajo es integrar en ORB-SLAM2 las predicciones de profundidad proporcionadas en tiempo real por la red Monodepth2, acercándose a las prestaciones de un sistema de SLAM estéreo, a pesar de utilizar una única cámara monocular. Para ello, se ha partido de la red Monodepth2 pre-entrenada en secuencias urbanas del *dataset* KITTI, y se ha llevado a cabo un análisis en profundidad de su precisión. Esto ha permitido utilizar adecuadamente la disparidad estimada por la red neuronal, seleccionando las predicciones de profundidad de menor incertidumbre, que se integran en ORB-SLAM2 como observaciones estéreo virtuales, mientras que el resto se siguen utilizando como observaciones monoculares.

El sistema se ha evaluado en secuencias de KITTI distintas de las utilizadas en el entrenamiento. Los resultados obtenidos demuestran que se estima la escala real del entorno con un error promedio del 3 %, y se obtienen mapas más precisos que los de ORB-SLAM2 monocular, habiendo disminuido la deriva de la escala del 43 % al 3 %. Esto ha permitido reducir de 17,24 m a 6,70 m el error promedio de las trayectorias construidas por el sistema.

Índice

Capítulo 1

Introducción y objetivos

1.1. Motivación y contexto

La localización y mapeo simultáneos o SLAM (*Simultaneous Localisation and Mapping*) es el problema computacional que plantea si un agente móvil es capaz de construir incrementalmente un mapa consistente de un entorno desconocido al mismo tiempo que determina su posición dentro de él [1]. Este campo de estudio encuentra su aplicación en el desarrollo de sistemas de navegación autónoma, implementados en proyectos de robótica general o, más recientemente, en los nuevos vehículos auto conducidos o en herramientas de realidad virtual y realidad aumentada.

Este Trabajo Fin de Grado centra sus objetivos en el caso del SLAM visual; cuyos sensores, denominados cámaras, captan información del entorno en forma de secuencias de imágenes. Si el sistema cuenta con una única cámara, se habla de SLAM monocular. Si, por el contrario, se dispone de dos cámaras que captan información simultáneamente, el sistema se conoce como SLAM estéreo. Más allá de esta clasificación, algunas cámaras son configuradas para capturar radiación fuera del espectro visible por el ojo humano. Este es el caso, por ejemplo, de los sensores de profundidad utilizados para obtener imágenes RGB-D. Estas imágenes almacenan datos de tres canales para codificar la radiación visible y un cuarto canal para almacenar información de la distancia desde el dispositivo de captura a cada punto de la escena.

En el sistema ORB-SLAM2 monocular [2][3], desarrollado dentro de la Universidad de Zaragoza, las secuencias de imágenes son capturadas por una única cámara 2D y no proporcionan información de escala al algoritmo. La falta de observabilidad de las distancias reales provoca que el sistema tenga que trabajar con una escala relativa. Los mapas obtenidos por este método pueden llegar a ser razonablemente precisos si se les proporciona *a posteriori* un factor de escala que ajuste las dimensiones estimadas a las del mundo real. Aun así, el resultado puede verse deteriorado si la estimación de la escala no es consistente a lo largo de la trayectoria del agente móvil. En este caso, no es posible determinar un único factor de escala que adecúe la estimación a la realidad. Este problema es conocido como deriva de la escala o *scale drift*.

Teniendo en cuenta lo anterior, se plantea la posibilidad de estimar la profundidad de los objetos de la escena al ser vistos por la cámara que los captura. Se pretende aproximar una función no lineal que permita obtener mapas de profundidad a partir de las imágenes monoculares que recibe ORB-SLAM2. En este sentido, resultados recientes [4] muestran que las redes convolucionales (CNN) son adecuadas para predecir profundidad. El entrenamiento de estas redes neuronales se lleva a cabo utilizando secuencias de imágenes como señal de supervisión. Si las secuencias son monoculares, la red solamente podrá

aprender a estimar la profundidad a escala (*up-to-scale*). Para poder obtener distancias métricas reales el entrenamiento deberá hacerse con pares de imágenes estéreo. Las opciones híbridas, que aprovechan ambos tipos de secuencias, son las que presentan una mejor precisión en las soluciones del estado del arte.

Integrando ambos sistemas, se espera mitigar el problema de la deriva de la escala hasta alcanzar resultados superiores a los proporcionados por ORB-SLAM2 monocular hasta el momento.

1.2. Objetivos y alcance del proyecto

El objetivo del proyecto es integrar en ORB-SLAM2 monocular la profundidad estimada por una red convolucional, para construir mapas a tamaño real y eliminar la deriva de la escala. Los puntos que tratar en el presente trabajo son.

1. Estudio de técnicas de SLAM y soluciones existentes para la estimación de profundidad con redes convolucionales profundas.
2. Análisis detallado de la predicción de profundidad en escenas urbanas del conjunto de datos KITTI [5] para establecer la mejor manera de adaptar la red al dominio de aplicación.
3. Integración de la profundidad predicha por la red neuronal en ORB-SLAM2 monocular.
4. Evaluación del resultado obtenido en cuanto a la estimación de la escala real del entorno y a la reducción de la deriva de la escala. Para ello se utilizarán secuencias distintas de las usadas en el entrenamiento.

1.3. Metodología y herramientas

Todas las secuencias de imágenes utilizadas en el trabajo se han extraído del conjunto de datos KITTI [5]. Los datos disponibles se han dividido en los tres grupos mostrados en la tabla 1.1. Cada subconjunto tendrá una función específica. Las secuencias con *ground truth* para la trayectoria del vehículo se reservan para evaluación.

El proyecto tiene como punto de partida el código en C++ de ORB-SLAM2 [3] propuesto por Mur-Artal *et al.* y la red neuronal Monodepth2 [4] de Godard *et al.* En cuanto al trabajo con el sistema SLAM (sección 2.1), se ha profundizado en el manejo de la biblioteca OpenCV [6], que ofrece herramientas para aplicaciones de visión por computador. En el caso de las redes neuronales (sección 2.2), se ha mantenido la implementación basada en PyTorch [7] sobre CUDA, adaptando la interfaz de Python al lenguaje C++ mediante la biblioteca LibTorch.

Los cálculos necesarios para el diseño de la integración (capítulo 4) se han llevado a cabo con la herramienta Octave [8], alternativa libre a MATLAB. Para la visualización de resultados del análisis (capítulo 3) y la evaluación final del rendimiento (sección 5) se ha elegido la biblioteca Matplotlib [9].

Tabla 1.1: **Subconjuntos KITTI** de imágenes utilizadas.

Subconjunto	*¿Tiene ground truth?*	*Función*
Raw Data	No	Entrenamiento del modelo neuronal
Odometry 11 a 21	No	Sintonización de hiperparámetros
Odometry 00 a 10	Sí	Evaluación del sistema

Todos los cambios realizados al código base, así como las herramientas auxiliares desarrolladas para este proyecto se encuentran bajo control de versiones, haciendo uso de la herramienta Git. El presente informe del proyecto, también bajo control de versiones, se ha elaborado con el sistema de composición de textos LaTeX.

El resumen de principales tecnologías y herramientas incluye, por tanto, las siguientes.

- C++ (`https://isocpp.org`)
- ORB-SLAM2 (`https://github.com/raulmur/ORB_SLAM2`)
- Monodepth2 (`https://github.com/nianticlabs/monodepth2`)
- KITTI (`http://www.cvlibs.net/datasets/kitti/`)
- OpenCV (`https://opencv.org/`)
- PyTorch y LibTorch (`https://pytorch.org/`)
- CUDA (`https://developer.nvidia.com/cuda-zone`)
- Python (`https://www.python.org/`)
- Octave (`https://www.gnu.org/software/octave/`)
- Git (`https://git-scm.com/`)
- LaTeX (`https://www.latex-project.org/`)

Capítulo 2

Fundamentos previos

2.1. SLAM visual

El algoritmo de ORB-SLAM2 recibe como entrada la secuencia de imágenes tomadas por una cámara, en el caso monocular, o por un par de cámaras, en el caso estéreo. Estos dispositivos se modelan como cámaras estenopeicas, también conocidas con el nombre de cámara oscura o *pinhole*. Son aquellas en las que la imagen es capturada a través de un pequeño orificio por el que entra la luz.

El problema del SLAM visual consiste en minimizar los errores de reproyección de los puntos de la escena, determinando las coordenadas de los puntos, las orientaciones y las posiciones de la cámara en cada momento:

$$\{\boldsymbol{R}_{iw}, \boldsymbol{p}_{iw}, \boldsymbol{x}_{wj} \mid \forall i, \forall j\}^* = \underset{\boldsymbol{R},\boldsymbol{p},\boldsymbol{x}}{\operatorname{argmin}} \sum_{i,j} \rho\left(\|\boldsymbol{u}_{ij} - \pi\left(\boldsymbol{R}_{iw}\boldsymbol{x}_{wj} + \boldsymbol{p}_{iw}\right)\|^2_{\boldsymbol{\Sigma}_{ij}}\right) \tag{2.1}$$

- $\boldsymbol{x}_{wj} = [x_{wj}, y_{wj}, z_{wj}]^T \in \mathbb{R}^3$ son las coordenadas del punto j en el mundo.
- $\boldsymbol{R}_{iw} \in \mathrm{SO}(3)$ es la orientación de la cámara i.
- $\boldsymbol{p}_{iw} \in \mathbb{R}^3$ es la posición de la cámara i.
- $\boldsymbol{x}_{ij} = \boldsymbol{R}_{iw}\boldsymbol{x}_{wj} + \boldsymbol{p}_{iw}$ son las coordenadas del punto j respecto de la cámara i.
- $\boldsymbol{u}_{ij}$ es la observación del punto j desde la cámara i.
- $\pi(\,)$ es la función de proyección de un punto.
- $\rho(\,)$ es una función de coste robusto (por ejemplo, coste de Hubber).
- $\boldsymbol{\Sigma}_{ij} = \sigma^2_{ij}\boldsymbol{I}_{2\times 2}$ es la incertidumbre del punto i en la imagen j. Habitualmente la desviación típica se considera igual a 1 píxel.

En el caso monocular, con distancia focal f_x y f_y en píxeles de cada eje, y el centro óptico en las coordenadas (c_x, c_y) del plano de imagen, se tiene que:

$$\pi_m\left(\boldsymbol{x}_{ij}\right) = \begin{bmatrix} f_x \frac{x_{ij}}{z_{ij}} + c_x \\ f_y \frac{y_{ij}}{z_{ij}} + c_y \end{bmatrix}, \qquad \boldsymbol{u}_{ij} = \begin{bmatrix} u_L \\ v_L \end{bmatrix} \tag{2.2}$$

Sin embargo, en el caso de un sistema estéreo con separación entre cámaras b (*baseline*), la observación de un punto comprende sus coordenadas en la cámara izquierda y su coordenada horizontal en la cámara derecha:

$$\pi_s\left(\boldsymbol{x}_{ij}\right) = \begin{bmatrix} f_x \frac{x_{ij}}{z_{ij}} + c_x \\ f_y \frac{y_{ij}}{z_{ij}} + c_y \\ f_x \frac{x_{ij}-b}{z_{ij}} + c_x \end{bmatrix}, \qquad \boldsymbol{u}_{ij} = \begin{bmatrix} u_L \\ v_L \\ u_R \end{bmatrix} \tag{2.3}$$

La diferencia entre las coordenadas horizontales izquierda y derecha, se conoce como disparidad:

$$D = u_L - u_R = \frac{f_x b}{z} \tag{2.4}$$

2.1.1. Falta de observabilidad de la escala

Si aplicamos un factor de escala s a los puntos del mundo y a la trayectoria de la cámara, tal que:

$$s \cdot \boldsymbol{x}_{wj}, \quad s \cdot \boldsymbol{p}_{iw} \implies s \cdot \boldsymbol{x}_{ij} = \begin{bmatrix} s \cdot x_{ij} \\ s \cdot y_{ij} \\ s \cdot z_{ij} \end{bmatrix} \tag{2.5}$$

Dicho factor no sería observable por la función de proyección monocular:

$$\pi_m\left(s \cdot \boldsymbol{x}_{ij}\right) = \begin{bmatrix} f_x \frac{s \cdot x_{ij}}{s \cdot z_{ij}} + c_x \\ f_y \frac{s \cdot y_{ij}}{s \cdot z_{ij}} + c_y \end{bmatrix} = \begin{bmatrix} f_x \frac{\cancel{s} \cdot x_{ij}}{\cancel{s} \cdot z_{ij}} + c_x \\ f_y \frac{\cancel{s} \cdot y_{ij}}{\cancel{s} \cdot z_{ij}} + c_y \end{bmatrix} = \pi_m\left(\boldsymbol{x}_{ij}\right) \tag{2.6}$$

No obstante, la escala sí es observable por la proyección estéreo, ya que:

$$\pi_s\left(s \cdot \boldsymbol{x}_{ij}\right) = \cdots = \begin{bmatrix} f_x \frac{x_{ij}}{z_{ij}} + c_x \\ f_y \frac{y_{ij}}{z_{ij}} + c_y \\ f_x \frac{x_{ij}-(b/s)}{z_{ij}} + c_x \end{bmatrix} \neq \pi_s\left(\boldsymbol{x}_{ij}\right) \tag{2.7}$$

Por lo tanto, utilizando geometría multivista [10] con un sistema estéreo se pueden obtener mapas 3D a escala real, pero con un sistema monocular, la escala queda indefinida. En ORB-SLAM2 monocular, si se desea solucionar este problema, se ha de proporcionar al sistema información acerca de la profundidad de la escena. De este modo, se pretende encontrar un método fiable para predecir la profundidad a partir de las imágenes monoculares que recibe el algoritmo.

2.1.2. Sistema ORB-SLAM2

El sistema ORB-SLAM2 [3] es publicado en el año 2017 como una ampliación del sistema previo ORB-SLAM [2] de 2015, que únicamente incluía soporte para sensores de tipo monocular. Ambas versiones han sido desarrolladas en la Universidad de Zaragoza y liberadas bajo licencia GNU GPLv3.

La estructura interna de la aplicación permite un alto grado de paralelización para lograr el procesamiento de secuencias de imágenes en tiempo real. Además, en la actualidad, es capaz de procesar información tanto monocular, como estéreo y RGB-D.

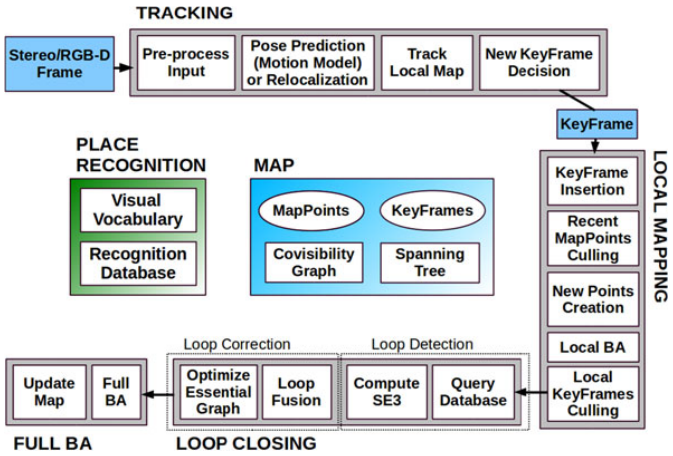


Figura 2.1: **Principales componentes de ORB-SLAM2 [3]**. Tres hilos de ejecución: *tracking*, *local mapping* y *loop closing*. Eventualmente, un cuarto hilo (*full BA*) es creado para ejecutar el proceso de optimización tras un cierre de bucle.

2.1.2.1. Estructura

En la figura 2.1 se muestran la estructura principal de ORB-SLAM2. A continuación, se enumeran y explican brevemente algunas de sus partes.

- Módulo de *Tracking*. Es el punto de entrada al sistema. Recibe secuencialmente las imágenes capturadas por los sensores. Aplica a cada imagen un algoritmo de extracción de puntos ORB. A partir de los descriptores, el algoritmo es capaz de emparejar puntos (*KeyPoints*) entre dos imágenes y/o de identificar correspondencias de estos con puntos del mapa (*MapPoints*) observados previamente. Es en esta fase donde se necesita obtener información relativa a la escala para ser propagada posteriormente a las siguientes etapas. Con los puntos observados en cada momento, ha de ser posible determinar la posición de la cámara respecto al mundo. En caso contrario, el sistema entra en estado de «*relocalización*».

- Módulo de *Local Mapping*. Esta fase es la encargada de crear nuevos puntos del mapa y eliminar aquellos incorrectos o de escasa utilidad. Durante este proceso, se mantiene actualizado el mapa local formado por un subconjunto de poses de cámara covisibles entre sí y sus puntos del mapa asociados.

- Módulo de *Loop Closing*. Se preocupa de detectar cuándo la cámara vuelve a visitar una determinada zona del mapa. En ese caso, se ejecutan acciones correctoras para alinear la información antigua con la incorporada más recientemente al sistema. Este proceso consiste en una optimización con siete grados de libertad, gracias a la cual es posible corregir parcial o totalmente la deriva de la escala.

- *Full Bundle Adjustment*. Como consecuencia del cierre de bucles, da comienzo un proceso de optimización global del mapa. Esta tarea se asigna a un nuevo hilo de ejecución para mantener activos el resto de los componentes del sistema.

2.1.2.2. Diferencias entre Monocular, Estéreo y RGB-D

El módulo de *Tracking* es el encargado de preprocesar la información recibida como entrada y traducirla a una estructura homogénea, independiente del tipo de sensor utilizado para observar el entorno.

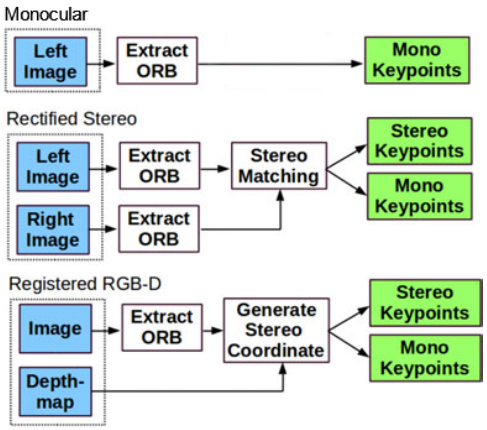


Figura 2.2: **Modos de funcionamiento de ORB-SLAM2 [3]**. Entradas y salidas del módulo de preprocesamiento encargado de extraer *KeyPoints* de las imágenes tratadas por el hilo de *Tracking*.

Basándose en esto, ORB-SLAM2 implementa tres modos diferentes de funcionamiento, los cuales se ilustran en la figura 2.2.

- Monocular. El sensor proporciona en cada instante una única imagen de la escena. El sistema es capaz de extraer puntos ORB, pero no tiene información de profundidad para cada uno de los puntos elegidos. Por ello, el módulo arrojará observaciones monoculares del tipo $[u_L, v_L]^T$.
- Estéreo. Al recibir dos imágenes simultáneas del entorno, el sistema extraerá puntos ORB en ambos fotogramas. Tomando la imagen izquierda como referencia, se buscarán correspondencias de cada punto en la imagen derecha. Los puntos que sean emparejados proporcionarán una observación estéreo del tipo $[u_L, v_L, u_R]^T$. Mientras que, si no se encuentra emparejamiento, se mantendrá la observación monocular.
- Imágenes anotadas con profundidad o *RGB-D*. Cada imagen de entrada se recibe acompañada de un mapa de profundidad. Tras la extracción de puntos en el fotograma izquierdo, se calcula para cada punto $[u_L, v_L]^T$ la correspondiente coordenada u_R (ecuación 2.8) de una observación derecha. Para ello, es necesario conocer la distancia focal f_x en píxeles, la distancia entre cámaras b y la profundidad z del punto. Aquellos puntos para los que no se disponga de información de profundidad se transmitirán como observaciones monoculares.

$$u_R = u_L - \frac{f_x b}{z} \tag{2.8}$$

2.2. Red convolucional Monodepth2

La red neuronal Monodepth2 [4] es publicada en el año 2019 como sistema de estado del arte para la predicción de profundidad a partir de imágenes monoculares. Su estructura (figura 2.3a) se basa en una arquitectura codificador-decodificador. El codificador es una red *ResNet* de 18 capas y el decodificador construye un mapa de disparidad, utilizando una capa de salida sigmoidal. Los autores convierten dicha salida σ en profundidad mediante $D = 1/(a\sigma + b)$, donde a y b se eligen para restringir D entre 0.1 y 100 unidades.

El entrenamiento de la red se lleva a cabo de manera auto-supervisada, utilizando para ello pares de imágenes estéreo (I_L, I_R) y/o parejas de fotogramas monoculares consecutivos (I_t, I_{t+1}). En el caso

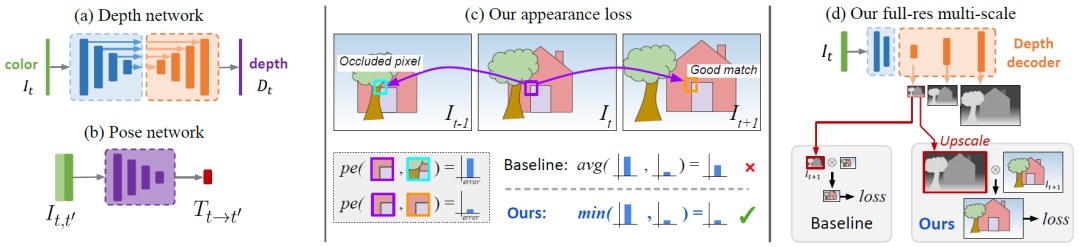


Figura 2.3: **Red convolucional Monodepth2 [4]**. (a) Red convolucional codificador-decodificador para estimar profundidad. (b) Red de pose entre pares de imágenes monoculares. (c) Mejora en la función de coste, usando el mínimo de los errores fotométricos. (d) El coste se calcula comparando las imágenes a escala original.

monocular, se desconocen los movimientos de la cámara entre los dos fotogramas I_t e $I_{t'}$. Por ello, se añade una segunda red neuronal capaz de predecir las poses relativas $T_{t\to t'}$ entre dos imágenes. Esta red es usada únicamente durante el entrenamiento.

La señal de supervisión del entrenamiento es el error fotométrico de reproyección [11]. Para minimizar la influencia de píxeles ocluidos en el coste, se aplica la función de mínimo. Es decir, dada la disparidad predicha para $I_{L,t}$ se sintetizan pseudoimágenes I'_R, I'_{t-1} e I'_{t+1}. Con ellas se calcula la menor diferencia entre cada píxel de la imagen real y su correspondiente imagen sintetizada.

Por último, el error de reproyección se calcula siempre sobre imágenes con la misma resolución que la original. Esto reduce los artefactos de textura.

El entrenamiento de la red asume que la escena permanece inalterada conforme la cámara recorre el entorno. Cuando esta premisa es incumplida, surgen efectos indeseados, como la aparición de zonas de profundidad infinita en el mapa predicho. Con su aportación, Godard *et al.* consiguen que la red ignore objetos que se desplazan a la misma velocidad que la cámara o incluso parejas de fotogramas en los que la cámara ha permanecido estática. Así se eliminan las zonas de valor infinito.

En la figura 2.4, se muestran cuatro ejemplos de los resultados ofrecidos por Monodepth2 sobre los fotogramas de KITTI elegidos por sus autores.

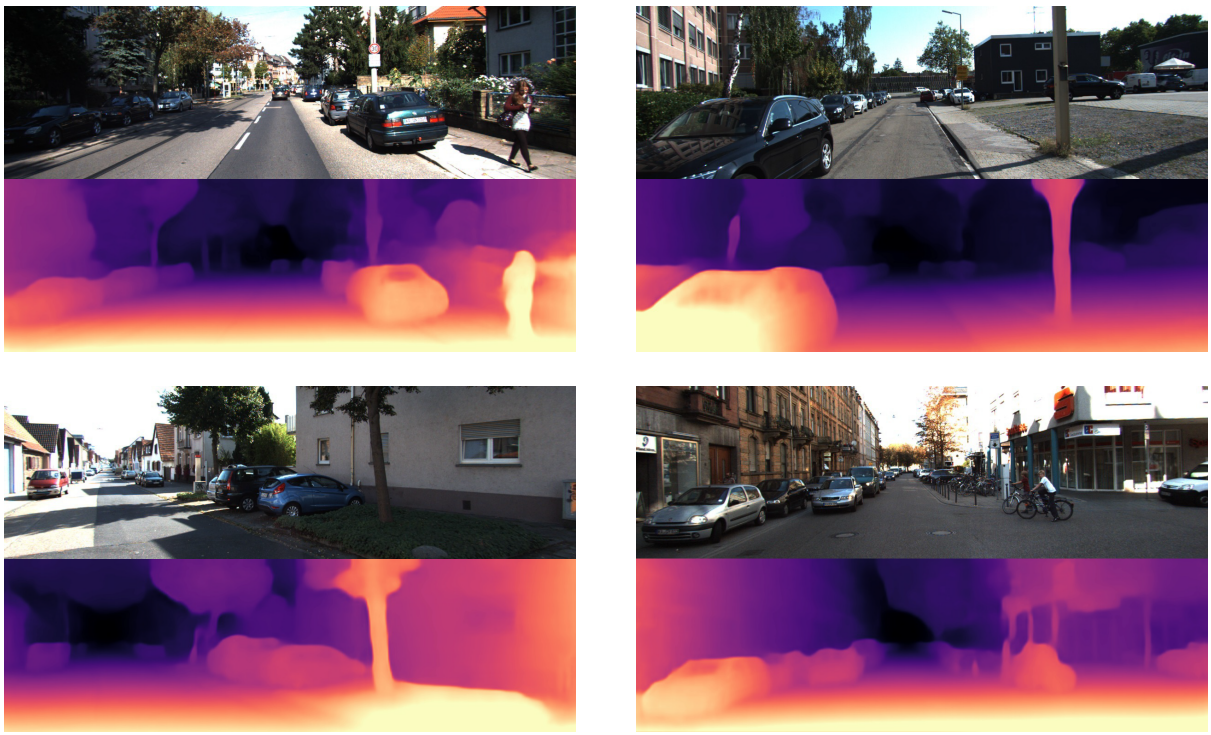


Figura 2.4: **Resultados cualitativos de Monodepth2 sobre el *dataset* de KITTI [4].** Mapas de disparidad obtenidos utilizando el modelo entrenado con secuencias monoculares más estéreo.

Capítulo 3

Análisis de la precisión en profundidad

3.1. Planteamiento

En este capítulo se analiza la precisión de la red neuronal Monodepth2 al estimar profundidad real. Para ello, se ha ejecutado, por un lado, Monodepth2; y por otro, ORB-SLAM2 estéreo. La profundidad predicha ($\hat{z}$) se ha obtenido haciendo uso del modelo «*mono+stereo_1024x320*» [4], entrenado por los autores con pares estéreo y monoculares. La salida de ORB-SLAM2 se toma como profundidad real (z), dado el bajo error cometido por el sistema estéreo para puntos cercanos.

En el análisis se compara la profundidad calculada por ORB-SLAM2 estéreo y la profundidad predicha por Monodepth2, para los puntos ORB observador por ORB-SLAM2 a lo largo de la secuencia KITTI 13 [12]. Para poder comparar las profundidades en las mismas unidades, se convierte la disparidad estimada por Monodepth2 ($\hat{D}$) a profundidad en metros. La red Monodepth2 ha sido entrenada con una distancia entre cámaras b igual a 0,1 m [4]. Sin embargo, la separación real en las secuencias de entrenamiento es de 0,54 m [5]. Para corregir el valor, se divide entre 0,1 m y se multiplica por la distancia real. El factor resultante es 5,4.

$$\hat{z} = \frac{5{,}4}{\hat{D}} \tag{3.1}$$

A partir de ambas profundidades, se calculan:

- *Error:* $\epsilon = \hat{z} - z$
- *Error absoluto:* $\epsilon_a = |\hat{z} - z|$
- *Error relativo:* $\epsilon_r = \frac{|\hat{z}-z|}{z}$
- *Ratio de profundidad:* $R = \hat{z}/z$

Estos valores se han analizado de tres modos: a lo largo de la trayectoria, en función de la profundidad real y subdividiendo el plano de imagen en sectores.

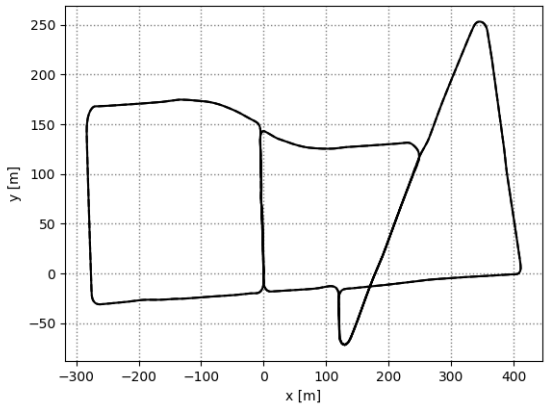


Figura 3.1: **Trayectoria KITTI 13 [12]**. Esta secuencia es una de las más largas del *dataset* y no dispone de *ground truth*. El resultado mostrado corresponde a la trayectoria estimada por ORB-SLAM2 estéreo.

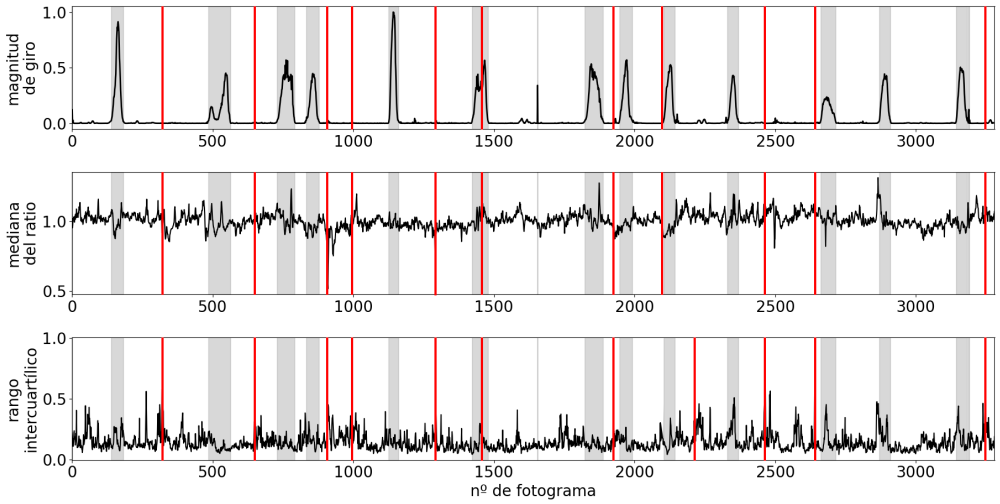
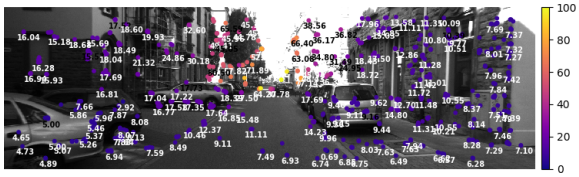


Figura 3.2: **Perfil del error a lo largo de KITTI 13**. **Arriba:** Magnitud del giro (α_t) normalizada. **Abajo:** Mediana y rango intercuartílico del ratio (R) para los puntos de cada fotograma. Se han marcado en rojo algunas imágenes en las que se observaba un máximo local del rango intercuartílico.

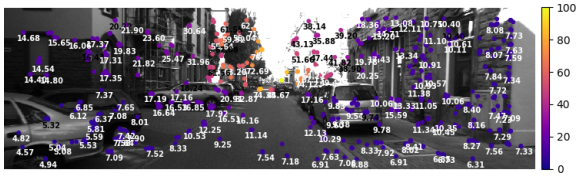
3.2. Distribución del error a lo largo de la trayectoria

Las zonas de giro, como las de KITTI 13 (figura 3.1), son los puntos más desafiantes para el sistema ORB-SLAM2 monocular. Cuando la cámara realiza una rotación entorno a su eje vertical, la escala suele volverse inconsistente con el valor estimado. Por ello, se desea analizar especialmente si la precisión de Monodepth2 es adecuada en las curvas.

Por un lado, en la figura 3.2, se dibuja la evolución del error para los puntos ORB de cada zona de la trayectoria. En esta gráfica, un elevado rango intercuartílico evidencia un mayor número de puntos erróneos en la predicción para ese fotograma. Los instantes marcados con rojo son algunos máximos locales del error, que corresponden con zonas adversas para el sistema de predicción de profundidad.



(a) Profundidad ORB-SLAM2, z.



(b) Profundidad Monodepth2, $\hat{z}$.

(c) Error relativo, ϵ_r.

Figura 3.3: **Fotograma de ejemplo de KITTI 13.** En (a) y (b) se muestran las profundidades en metros para cada *KeyPoint* de la imagen. En (c) el error relativo entre el 0 % y el 10 %. A los errores que superan el 10 % se les asigna igualmente el color amarillo.

Tabla 3.1: **Resumen global** de errores absolutos y relativos para dos secuencias de KITTI.

	KITTI 13	KITTI 00
Media	4,16 m	5,98 m
Mediana	1,34 m	2,44 m
RMSE	381,8 m	505,6 m

	KITTI 13	KITTI 00
Media	20,69 %	14,71 %
Q_1	3,99 %	3,49 %
Mediana	9,34 %	7,81 %
Q_3	19,94 %	16,01 %

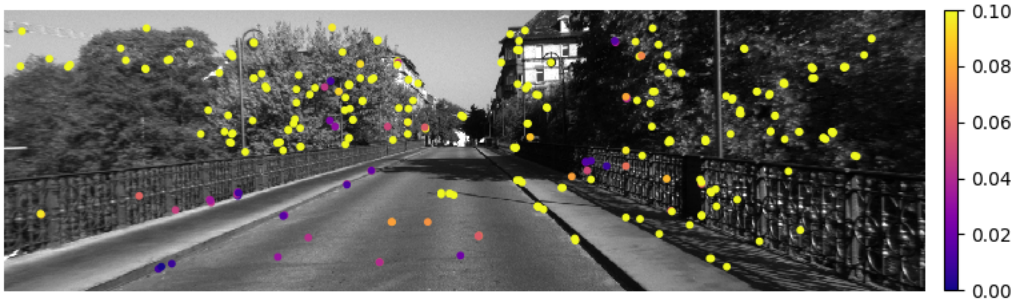
Para detectar automáticamente zonas en curva, se define una variable que representa la *magnitud del giro* efectuado por la cámara en cada instante del recorrido. Sea $\boldsymbol{z_t}$ el vector unitario (hacia delante) del eje z de la cámara en el instante t, la ecuación 3.2 calcula la magnitud del giro.

$$\alpha_t = 1 - \boldsymbol{z_t} \cdot \boldsymbol{z_{t-1}} = 1 - |z_t||z_{t-1}| \cos\theta = 1 - \cos\theta \tag{3.2}$$

En la figura 3.2, se muestra también la evolución de α_t a lo largo de KITTI 13. Nótese que no se aprecia una tendencia clara que sugiera que el sistema de predicción funcione peor en zonas de giro, sombreadas en gris, que en trayectos de línea recta.

Por otro lado, en la figura 3.3, se muestra un ejemplo de los puntos ORB en un fotograma y sus profundidades z y $\hat{z}$. Además, al representar el error relativo de cada punto, se aprecia la existencia de datos atípicos en varias zonas de la imagen. La tabla 3.1 recoge los valores de error absoluto y relativo para dos trayectorias completas de KITTI. El elevado valor de RMSE, frente a la mediana del error, confirma la existencia de predicciones anómalas en la salida de Monodepth2.

Finalmente, en la figura 3.4, se recogen tres ejemplos de zonas en las que se ha detectado un mayor número de datos erróneos. Se trata de zonas con una textura muy marcada (árboles, arbustos u otra vegetación) y áreas de poco contraste (subexpuestas, sobreexpuestas o con sombra).



(a) Fotograma nº 000914.

(b) Fotograma nº 001274.

(c) Fotograma nº 001457.

Figura 3.4: **Problemas detectados en KITTI 13.** En (a) las zonas de vegetación suponen un desafío para la red neuronal, dada la alta frecuencia de cambios en la luminosidad. En (b) y (c) las áreas sobreexpuestas o con sombra parecen dar lugar a errores en la predicción de la profundidad.

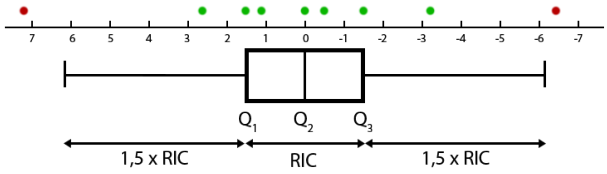
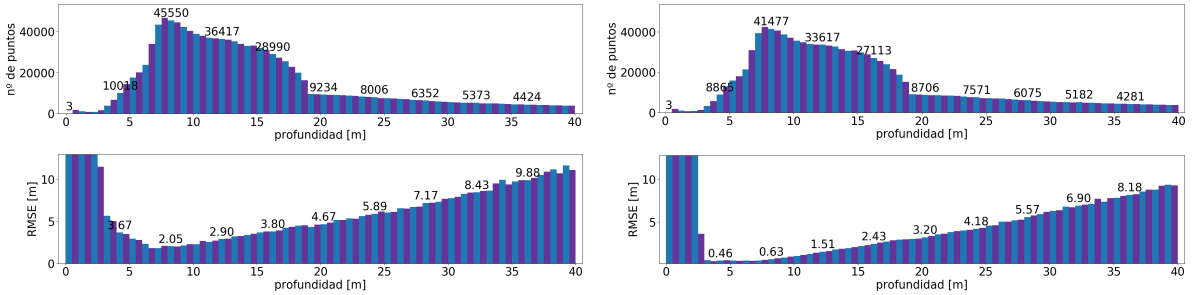


Figura 3.5: **Regla de Tukey [13].** Ejemplo de diagrama de caja y sus correspondientes valores atípicos representados en rojo.



(a) Con valores atípicos.

(b) Sin valores atípicos.

Figura 3.6: **Eliminación de atípicos para calcular RMSE. Arriba:** Número de puntos pertenecientes a cada clúster. **Abajo:** RMSE para cada grupos de puntos.

3.3. Distribución del error en profundidad

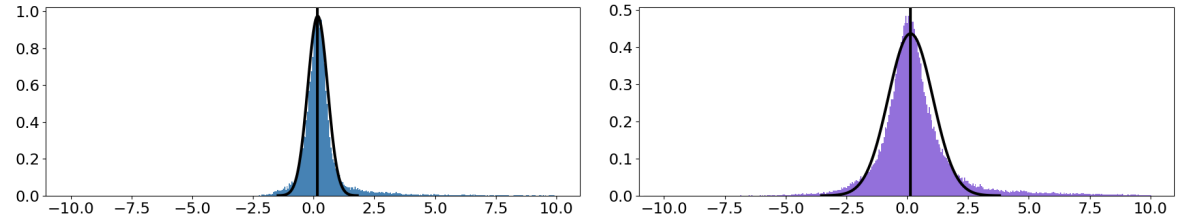
Si bien todas las predicciones de profundidad de Monodepth2 tienen asociado un cierto error, en la sección anterior se ha detectado la existencia de datos atípicos, que podrían lastrar el comportamiento del sistema. ORB-SLAM2 se basa en estimación por mínimos cuadrados, que es adecuada para errores Gaussianos, y utiliza la función de coste robusto de Huber (ecuación 2.1), que permite reducir la influencia de los datos espurios, salvo que sean demasiados y dominen la estimación. El objetivo de esta sección es determinar en qué rango de profundidades hay pocos errores atípicos, y obtener un modelo Gaussiano para el resto de los errores.

Para modelar el error del sistema, los puntos ORB analizados en la sección anterior se dividen en clústeres según su profundidad. Para cada uno de estos grupos se calcula el RMSE una vez eliminados los datos atípicos. La eliminación de anómalos durante el análisis se ha realizado con la regla de Tukey [13], que permite eliminar valores atípicos leves. Con este método, se consideran atípicos aquellos errores fuera del intervalo:

$$[Q_1 - 1{,}5 \times RIC,\ Q_3 + 1{,}5 \times RIC] \tag{3.3}$$

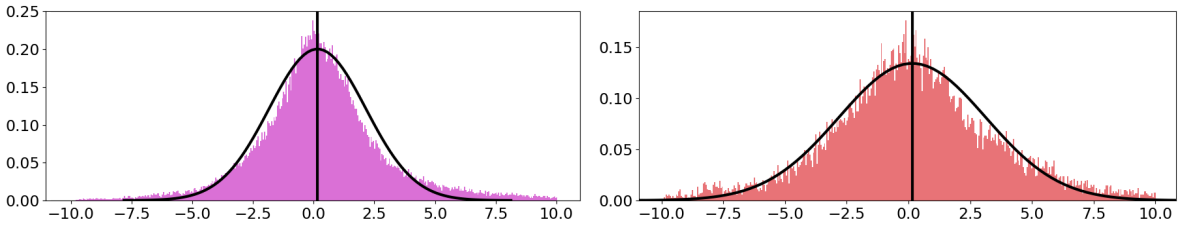
Siendo Q_1 y Q_3 los cuartiles del conjunto o subconjunto de errores analizado; y $RIC = Q_3 - Q_1$, el rango intercuartílico. La figura 3.5 ilustra este método, que ha permitido descartar un 8 % de datos atípicos.

En la figura 3.6, se puede observar la diferencia entre calcular el RMSE con valores atípicos y sin ellos. La mayoría de los puntos ORB se encuentra por debajo de los 20 metros de profundidad. Además, los clústeres de menos de 7 metros presentan un gran número de datos atípicos, por lo que se ha decidido no confiar en la función de Hubber para su eliminación y retirarlos mediante un recorte manual.



(a) De 4,5 a 5,5 metros.

(b) De 9,5 a 10,5 metros.



(c) De 14,5 a 15,5 metros.

(d) De 19,5 a 20,5 metros.

Figura 3.7: **Distribución de los errores en KITTI 13.** Todas las medias (en metros) están desplazadas hacia valores positivos. Monodepth2 tiene tendencia a sobrestimar la profundidad.
Medias del error: (a) 0,2543 m, (b) 0,2574 m, (c) 0,2961 m, (d) 0,4451 m.

Eliminando los datos atípicos, el error de Monodepth2 se puede representar mediante distribuciones Gaussianas. Esto se puede observar en la figura 3.7, donde se ha estudiado la distribución de los errores en cuatro rangos de profundidad. Sobre el histograma de los errores, se ha dibujado la campana de Gauss correspondiente a la media y desviación muestral una vez quitados los atípicos. Nótese que los extremos de los histogramas son mayores que las colas de una distribución normal, esto es consecuencia de la existencia de errores anómalos.

Entonces, se propone modelar la parte gaussiana del sistema Monodepth2, confiando que la detección de datos espurios implementada en ORB-SLAM2 se encargue de los valores anómalos. Así pues, cabe esperar que la función de coste robusto de Hubber sea eficaz ante las muestras atípicas para una distribución normal y que esos datos anómalos serán descartados por el sistema SLAM.

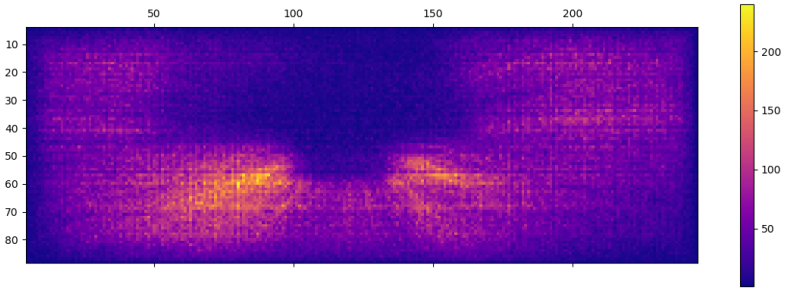
Por último, atendiendo de nuevo a la figura 3.7, se ha detectado la tendencia de la red a sobrestimar la profundidad. En el extremo derecho de cada histograma están los puntos sobrestimados por Monodepth2 (i.e. $\hat{z} > z$). Mientras que, los puntos subestimados (i.e. $\hat{z} < z$) se sitúan en la mitad izquierda. Así, se aprecia un ligero sesgo positivo de los errores.

3.4. Distribución del error en plano de imagen

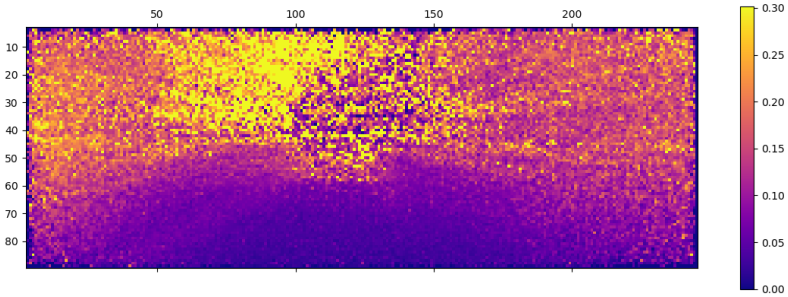
Si se conoce *a priori* la apariencia de las imágenes que llegan al sistema ORB-SLAM2, se pueden delimitar zonas del fotograma donde los valores de profundidad suelen ser erróneos. En el caso de las secuencias del *dataset* KITTI, el entorno se observa desde un turismo que circula por zonas urbanas e interurbanas.

A partir del conjunto KITTI 13, se han obtenido dos mapas de calor, visibles en la figura 3.8. Uno de ellos muestra la distribución 2D de los puntos utilizados en los apartados anteriores, mientras el segundo recoge las zonas con mayor error relativo medio.

Del mapa de la figura 3.8b, se puede extraer que los puntos ORB situados en la calzada son predichos con gran precisión, siendo su error cercano al 5 %. A la izquierda y la derecha de la calzada se encuentra



(a) Número de puntos.



(b) Media del error relativo de los puntos.

Figura 3.8: **Mapas de calor para KITTI 13,** que muestra la distribución, en plano de imagen, de los puntos analizados y sus fallos de predicción. Se ha limitado la profundidad de los puntos utilizados de 5 a 20 metros. A causa del límite, en (a) la zona central (violeta) carece de puntos. Nótese que el error de (b) no es representativo para zonas con pocos puntos.

un cúmulo de puntos, que posiblemente pertenezcan a otros vehículos aparcados en los laterales del asfalto. La profundidad de estos puntos presenta un error algo más elevado, alrededor del 10 %. Por último, el resto de los puntos, a ambos lados de la imagen, suelen corresponder a fachadas de edificios. En ellas, el error de predicción alcanza el 20-30 %.

Si se transfiere esta información al sistema ORB-SLAM2, el algoritmo puede ser capaz, con ligeras modificaciones, de seleccionar puntos situados en las primeras dos zonas descritas. Sin embargo, este método no ha sido implementado finalmente, para no perder generalidad, ya que estos parámetros son dependientes del aspecto visual de las escenas tratadas.

3.5. Conclusiones del análisis

Del análisis realizado se extraen las siguientes conclusiones:

- Los giros de la trayectoria no parecen entorpecer la labor de Monodepth2.
- La red arroja resultados erróneos en zonas con textura e iluminación adversas. Estos errores dependen de factores complejos, relacionados con las características visuales de la escena.
- Por debajo de 7 metros, la profundidad predicha por Monodepth2 presenta numerosos valores atípicos. Los puntos de más de 20 metros son escasos y su incertidumbre es sistemáticamente alta.
- El error de Monodepth2, eliminados los atípicos, se puede modelar con una distribución Gaussiana.

- Monodepth2 tiende a sobreestimar ligeramente la profundidad.
- Sin embargo, predominan los puntos para los que la red funciona adecuadamente. Los valores centrales del error relativo (tabla 3.1) se encuentran en torno al 9 %.

Capítulo 4

Integración de Monodepth2 con el sistema ORB-SLAM2

Se va a integrar la profundidad predicha por Monodepth2 en ORB-SLAM2 modelando el nuevo sistema como un estéreo virtual.

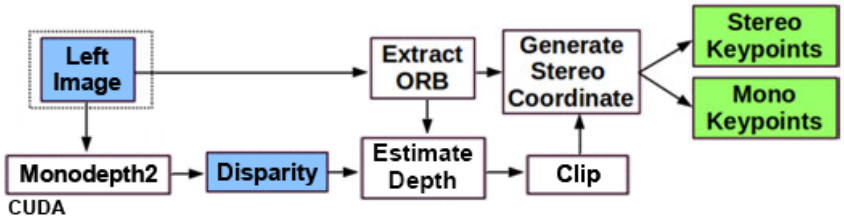


Figura 4.1: **Nuevo modo de funcionamiento de ORB-SLAM2**. Entradas y salidas del módulo de preprocesamiento encargado de extraer puntos característicos de las imágenes tratadas por el hilo de *Tracking*. Este modo se añade a los existentes en la figura 2.2. La extracción de puntos ORB y la estimación de la disparidad se realizan en paralelo.

Para la integración de ambos sistemas se han de tener en cuenta una serie de consideraciones, tanto teóricas como técnicas. Cada una de ellas se resume, a continuación, en la sección correspondiente.

4.1. Modelado como un sistema estéreo virtual

La estimación de la red Monodepth2 lleva asociada cierta incertidumbre (capítulo 3). En esta sección se va a estimar una distancia entre cámaras (*baseline*) que aproxime el error de precisión de Monodepth2 al de un sistema estéreo.

La incertidumbre de una observación estéreo procede del error al medir la disparidad (ΔD) entre las coordenadas en píxeles de un punto característico. A partir del error de disparidad, es posible calcular la incertidumbre de la profundidad del punto (Δz) según la ecuación 4.1. Esta expresión se obtiene derivando la ecuación 2.4.

$$\Delta z = \frac{z^2}{b \cdot f} \cdot \Delta D \tag{4.1}$$

Como se ve, el error en profundidad en un sistema estéreo crece cuadráticamente con la profundidad del punto a la cámara. La distancia focal de la cámara (f) es conocida para todas las secuencias de KITTI.

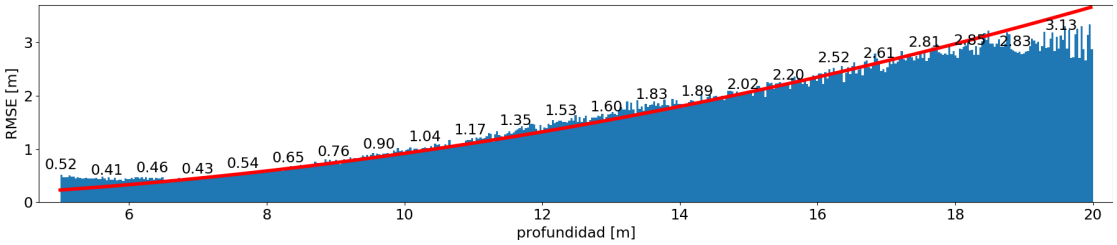


Figura 4.2: **Ajuste del error en profundidad a una parábola.** Se muestra el RMSE para cada profundidad de 5 a 20 metros. Los datos experimentales (azul) equivalen a los de la figura 3.6b. La parábola (rojo) corresponde con la ecuación 4.1 si todas las variables son constantes, exceptuando la profundidad z. Los valores son: $b = 0{,}15146$ m, $f = 718{,}856$ px y $\Delta D = 1$ px.

Por tanto, la única variable desconocida de nuestro sistema estéreo virtual es la separación entre cámaras (b). El presente trabajo utiliza sensores monoculares, de modo que no existe un valor real de b, sino que se elige un valor de manera que la incertidumbre de la profundidad del estéreo virtual corresponda a la del sistema Monodepth2.

Para obtener un valor aproximado, se propone estimar b a partir de los datos experimentales de RMSE (capítulo 3). Se han recalculado los valores de la figura 3.6b, con un tamaño inferior de clúster y eliminando igualmente los valores atípicos. Esto permite obtener un conjunto S de pares $(z, \Delta z)$.

Sea la función de coste:

$$L(b, z, \Delta z) = \Delta z - \frac{z^2}{b \cdot f} \cdot \Delta D \tag{4.2}$$

Se debe resolver el problema de minimización:

$$b = \underset{b}{\operatorname{argmin}} \sum_{(z,\Delta z) \in S} L(b, z, \Delta z)^2 \tag{4.3}$$

A partir de los datos experimentales de KITTI 13, siendo $f = 718{,}856$ píxeles y $\Delta D = 1$ píxel, se ha calculado el valor de b que minimiza el coste cuadrático. La herramienta elegida para ello es Octave y su función `fminunc()`. El resultado, mostrado en la figura 4.2, indica que la incertidumbre de la estimación de profundidad de Monodepth2 se puede aproximar por la de un sistema estéreo virtual con baseline $b = 0{,}15146$ metros.

4.2. Eliminación de puntos con elevada incertidumbre

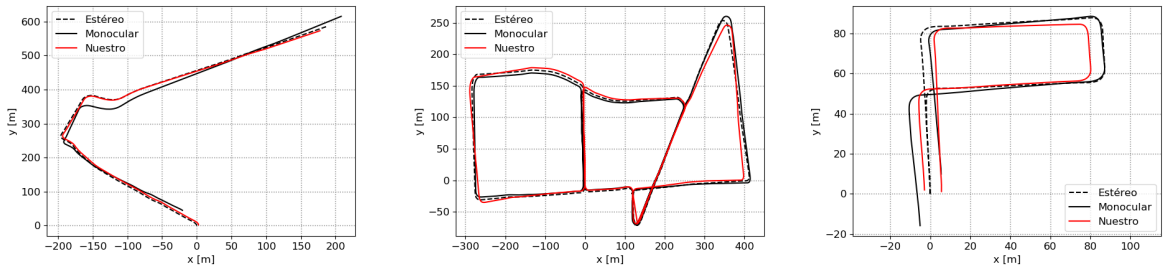
Durante el análisis se ha detectado la necesidad de filtrar valores predichos erróneamente por la red convolucional. El método más efectivo parece ser limitar los puntos utilizados por ORB-SLAM2 en base a su profundidad. Así pues, el filtro diseñado recorta la profundidad, eliminando los valores:

- Inferiores a 7 metros, por contener un elevado número de datos atípicos.
- Superiores a 20 metros, por ser escasos y no aportar información útil para reducir el error de escala.

Cuando un punto se encuentra de 7 a 20 metros de profundidad, se calcula su coordenada derecha u_R (ecuación 2.8). En caso contrario, el punto (u_L, v_L) será tratado como observación monocular y necesitará ser triangulado desde diferentes poses de la cámara para conocer su posición real. Por tanto, en la nueva salida del preprocesamiento (figura 4.1) se tendrán *KeyPoints* estéreo y monoculares.

Tabla 4.1: **Errores de escala para las trayectorias de validación.** Se muestra para las secuencias en las que ORB-SLAM2 funciona. Las secuencias KITTI 12, 17, 20 y 21 se omiten porque el sistema no es capaz de completar la trayectoria al tratarse de zonas interurbanas.

	Ejecución		
Secuencia	*#1*	*#2*	*Media*
KITTI 11	2,27 %	2,11 %	2,19 %
KITTI 13	3,19 %	3,01 %	3,10 %
KITTI 15	6,96 %	6,75 %	6,85 %
KITTI 16	2,07 %	1,90 %	1,99 %
KITTI 18	6,75 %	5,64 %	6,19 %
KITTI 19	4,38 %	5,03 %	4,71 %
		Mediana	**3,90 %**



(a) KITTI 11 (b) KITTI 13 (c) KITTI 14

Figura 4.3: **Resultados de validación con factor de escala.** Las trayectorias calculadas por el sistema final mejoran al aplicar el factor de escala. Todas ellas han sido alineadas a Estéreo con tan solo 6 DoF, es decir, sin aplicar escalado *a posteriori*. En (a) las trayectorias se alinean casi a la perfección. En (b), a pesar de la existencia de varios cierres de bucles, el nuevo sistema estima con cierto error los giros. En (c) la escala del *eje y* es casi perfecta, pero existe un error de unos 8 metros en el *eje x*.

4.3. Corrección de la sobrestimación

Según los datos obtenidos en el análisis previo, la red Monodepth2 tiende a sobrestimar la profundidad de los puntos entre 5 y 20 metros. Si dicho factor es más o menos constante para todas las trayectorias, es posible corregir la sobrestimación, multiplicando los mapas de profundidad antes de ser usados en el sistema SLAM.

A partir de dos ejecuciones de las trayectorias reservadas para validación, se ha calculado el factor a aplicar. En la tabla 4.1, se muestran los errores de escala de cada secuencia, el promedio entre ambas ejecuciones y la mediana de los promedios. Finalmente, el error obtenido es del 3,90 %; por lo que el factor que se aplicará a la escala será igual a $1 - 0{,}039 \approx 0{,}961$.

El sistema final, una vez ajustados todos los hiperparámetros ya comentados, se ha validado con las secuencias de KITTI 11 a KITTI 21. En la figura 4.3, se comentan tres resultados representativos del banco de pruebas ejecutado.

4.4. Adaptación de Monodepth2 al lenguaje C++

La red neuronal elegida para la predicción de profundidad se encontraba implementada en Python, haciendo uso del módulo PyTorch. Sin embargo, existen dos razones por las que esta configuración ha

de modificarse. Por un lado, las prestaciones del lenguaje Python no son siempre adecuadas para un entorno de producción, donde la ejecución debe funcionar en tiempo real. Por otro lado, el sistema base, ORB-SLAM2, está implementado en C++ por lo que la comunicación entre procesos supondría una sobrecarga innecesaria.

Los autores de PyTorch ofrecen una API para adaptar estructuras programadas en Python. Esta biblioteca, denominada *LibTorch*, permite invocar el motor de PyTorch desde C++ e incluye soporte para aceleración por GPU mediante CUDA. La especificación de la red neuronal se almacena en un fichero de intercambio entre ambos lenguajes. Ese documento recibe el nombre de TorchScript.

Sin embargo, la expresividad de TorchScript no permite representar todas las estructuras de datos y control de Python. Por ello, se ha necesitado adaptar la propagación entre capas de Monodepth2. La estructura conceptual de la red permanece inalterada.

El resultado de este procedimiento es un fichero TorchScript que puede ser leído durante la inicialización de ORB-SLAM2. El modelo neuronal, ejecutado sobre CUDA, recibe una imagen y devuelve como salida un mapa de disparidad. La disparidad ha de convertirse en profundidad como se explica en la sección 2.2.

Puesto que la estimación de la disparidad se lleva a cabo sobre la GPU, la CPU puede realizar en paralelo la extracción de puntos ORB en la imagen. Los detalles algorítmicos de esta adaptación pueden consultarse en el apéndice B.

Capítulo 5

Evaluación del rendimiento del sistema

La integración de Monodepth2 y ORB-SLAM2 ha sido evaluada sobre un conjunto de secuencias de KITTI distinto al elegido para el entrenamiento de la red y del usado durante la validación y el análisis de la precisión. En concreto, los datos presentados en esta sección corresponden con las trayectorias KITTI 00 a KITTI 10. Todas ellas disponen de *ground truth*, por lo que se podrán calcular errores reales sobre la salida del sistema de SLAM.

5.1. Estimación de la escala

Nuestro sistema es capaz de estimar la escala de manera consistente a lo largo de la trayectoria. Se ha eliminado la deriva de la escala para las once trayectorias de evaluación. En la tabla 5.1, se comparan los errores de deriva para ambos sistemas. La deriva promedio del sistema ORB-SLAM2 monocular + Monodepth2 es del 3,04 % frente al 43,41 % de ORB-SLAM2 monocular. Estos datos se han calculado a partir de la escala al comienzo y al final de cada secuencia de KITTI, con las siguientes ecuaciones:

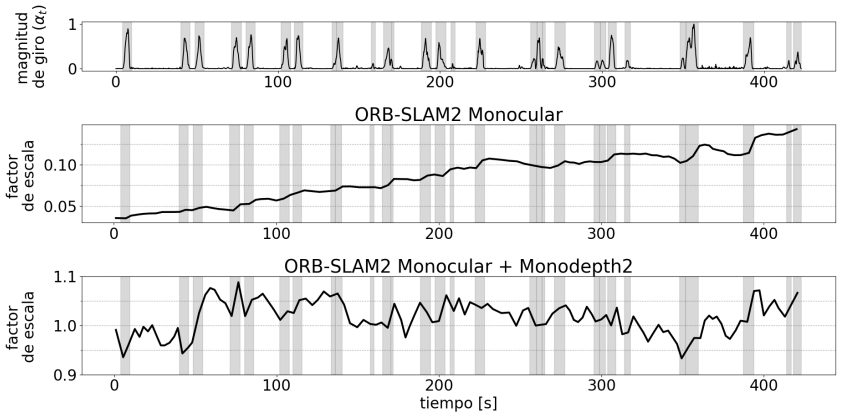
- *Escala estimada:* $\hat{s} = \frac{\text{distancia estimada}}{\text{distancia del } \textit{ground truth}}$
- *Deriva de la escala* $\Delta\hat{s} = \frac{\hat{s}_{\text{final}} - \hat{s}_{\text{inicial}}}{\hat{s}_{\text{inicial}}}$

En la figura 5.1, se muestra la evolución de la escala para ambos sistemas. Nótese que la escala estimada por ORB-SLAM2 monocular aumenta especialmente en las zonas de giro, sombreadas en la gráfica. Nuestra solución, además de predecir la escala real, mantiene más o menos consistente el valor estimado durante toda la trayectoria. Por eso, se aprecia que la escala para nuestro sistema permanece en torno a la escala real, es decir, a escala 1:1.

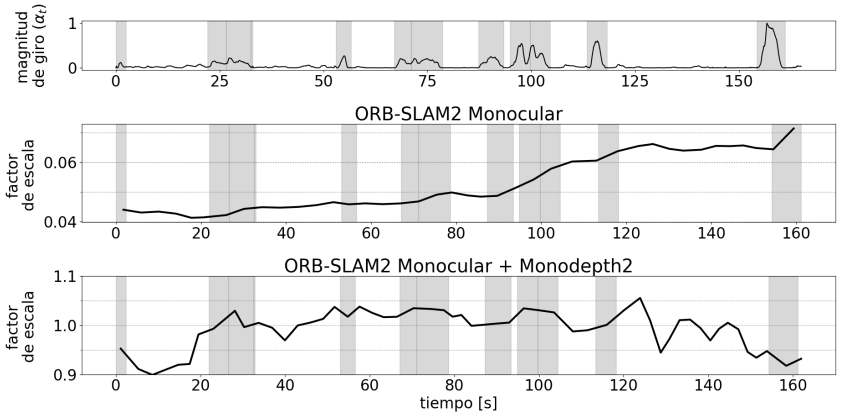
La reducción de la deriva permite obtener mapas menos deformados que los construidos por ORB-SLAM monocular. En la figura 5.2, se puede observar la mejora en la consistencia de la escala, especialmente en los casos KITTI 02, 08 y 09.

Tabla 5.1: **Resultados finales de deriva ($\Delta\hat{s}$).** La secuencia KITTI 01 se omite por no ser posible mapear completamente el entorno, al tratarse de una zona interurbana. La media se ha calculado a partir de los valores absolutos.

Secuencia	*Deriva en Monocular* (%)	*Deriva en Mono + MD2* (%)
KITTI 00	-0,27	-0,07
KITTI 02	51,68	-1,91
KITTI 03	8,57	-5,94
KITTI 04	-5,67	4,78
KITTI 05	-5,27	1,25
KITTI 06	15,81	0,44
KITTI 07	-4,27	-1,66
KITTI 08	280,46	4,61
KITTI 09	49,73	1,86
KITTI 10	12,35	7,91
Media	**43,41**	**3,04**

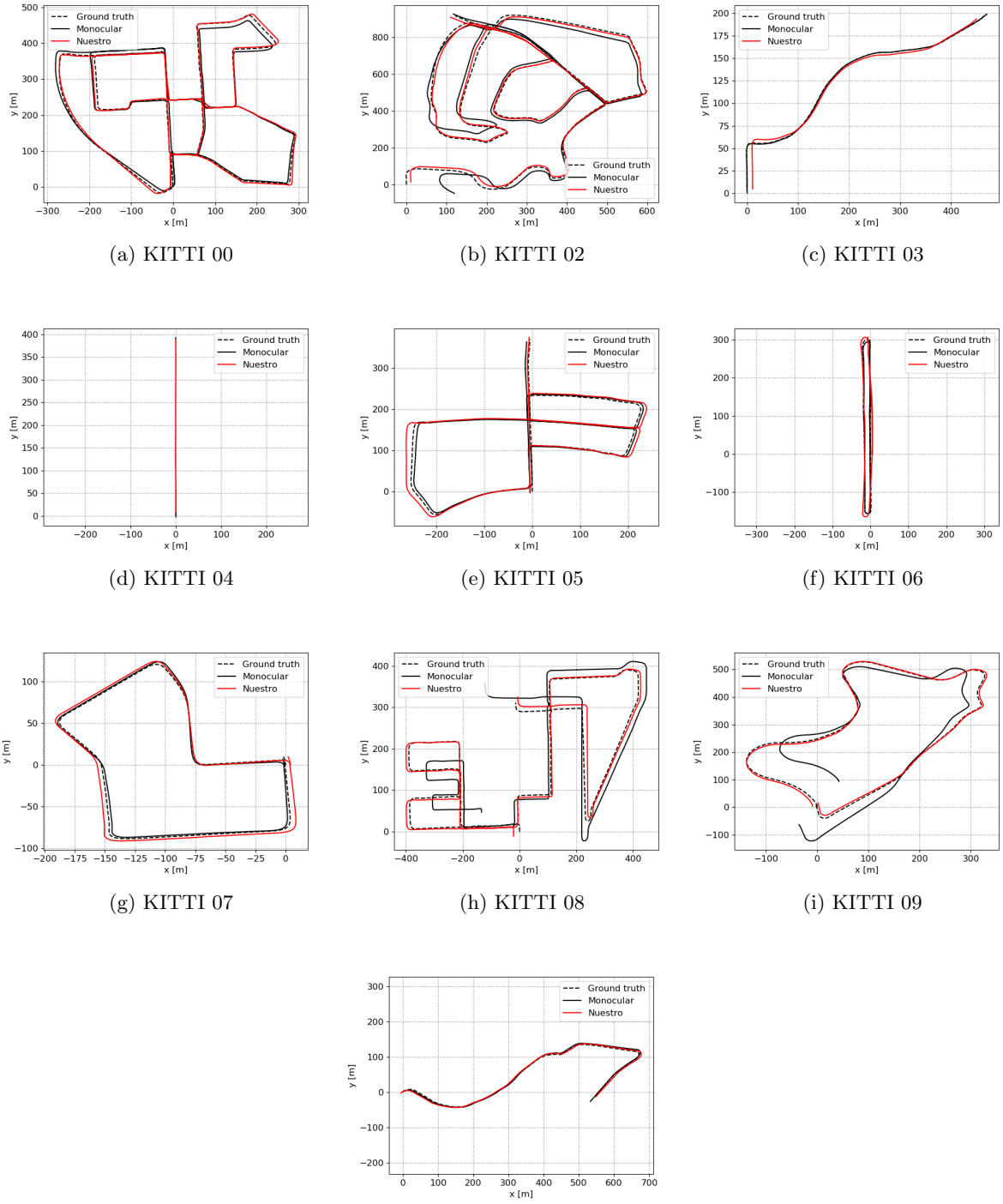


(a) KITTI 08



(b) KITTI 09

Figura 5.1: **Evolución de la escala en ORB-SLAM2 monocular frente a nuestro sistema.** La escala estimada por ORB-SLAM2 monocular no es consistente a lo largo de la trayectoria y aumenta especialmente durante los giros. Nuestro sistema predice la escala real, es decir, cercana a 1:1.



(a) KITTI 00 (b) KITTI 02 (c) KITTI 03

(d) KITTI 04 (e) KITTI 05 (f) KITTI 06

(g) KITTI 07 (h) KITTI 08 (i) KITTI 09

(j) KITTI 10

Figura 5.2: **Resultados finales de ORB-SLAM2 + Monodepth2.** Se compara la trayectoria real (*ground truth*, discontinua), la versión previa de ORB-SLAM2 monocular tras aplicar el factor de escala necesario (negra) y nuestra versión (roja).

5.2. Precisión de la trayectoria

La integración de ORB-SLAM2 y Monodepth2 permite obtener mapas con una escala muy próxima a la real. La tabla 5.2 muestra que el error promedio de escala es tan sólo del 2,91 %. Gracias a esta mejora, los resultados de nuestro sistema son notablemente mejores que ORB-SLAM2 monocular en cuanto a la

Tabla 5.2: **Resultados finales de odometría y escala.** A la trayectoria de ORB-SLAM2 monocular se le ha aplicado un factor de escala calculado *a posteriori*. La secuencia KITTI 01 también ha sido omitida. La media se ha calculado a partir de los valores absolutos.

Secuencia	*RMSE Estéreo* (m)	*RMSE Monocular* (m)	*RMSE Mono + MD2* (m)	*Error de escala* (%)
KITTI 00	1,34	7,96	**4,37**	0,99
KITTI 02	5,81	26,65	**9,19**	-1,67
KITTI 03	0,59	**0,96**	10,14	-6,69
KITTI 04	0,17	**1,52**	4,40	-3,69
KITTI 05	0,79	**5,15**	8,12	4,25
KITTI 06	0,92	16,66	**5,39**	2,77
KITTI 07	0,53	**3,36**	3,64	3,71
KITTI 08	4,03	47,49	**11,00**	2,82
KITTI 09	1,68	56,12	**5,20**	0,1
KITTI 10	1,16	6,55	**5,60**	2,41
Media	**1,70**	**17,24**	**6,70**	**2,91**

precisión de la trayectoria. El error promedio se ha reducido a 6,70 m, frente a los 17,24 m del sistema monocular.

El rendimiento final de la integración entre ORB-SLAM2 y Monodepth2, con todas las mejoras introducidas por el presente trabajo, se recoge en la figura 5.2, para que pueda ser visualizado por el lector. Cada secuencia de evaluación se compara con los errores previos de ORB-SLAM2 al utilizar sensores estéreo y monoculares.

La tercera columna de la tabla 5.2 corresponde con el RMSE de nuestro sistema en las trayectorias reservadas para la evaluación. Si comparamos esta columna con la precisión previa de ORB-SLAM2 monocular, podemos observar tres grupos:

- **Grupo 1:** Nuestra aportación consigue resultados claramente mejores en cinco de las secuencias. Son KITTI 00, 02, 06, 08 y 09. Las mayores mejoras se concentran en las trayectorias sin cierres de bucle, KITTI 08 y 09. Las secuencias KITTI 00 y 02 tienen numerosas zonas de giro, que son especialmente adversas para el sistema monocular original. Así pues, en este conjunto de secuencias se han cumplido el objetivo del trabajo: reducir la deriva de la escala.

- **Grupo 2:** En otros dos casos, ambos sistemas cometen errores equiparables. Se trata de las secuencias KITTI 07 y 10, donde los resultados de ORB-SLAM2 monocular eran aceptablemente buenos. Por ello, nuestro sistema no consigue mejoras notables ampliando la información monocular. Sin embargo, cabe destacar que ORB-SLAM2 monocular requiere un factor de escalado *a posteriori*, mientras que nuestra alternativa predice la escala real sin necesidad de un ajuste manual.

- **Grupo 3:** En tres de las pruebas, el nuevo sistema empeora el rendimiento preexistente. Sucede en KITTI 03, 04 y 05. Los resultados de nuestra solución son erróneos a causa de la naturaleza de las secuencias. Las trayectorias KITTI 03 y 04 carecen de giros pronunciados, por lo que no se producía deriva de la escala originalmente. Además, en ellas abunda la vegetación, por lo que Monodepth2 tiene dificultades para predecir la escala real de los puntos.

En la secuencia KITTI 01, el vehículo circula por una zona interurbana con escasos elementos verticales. El extractor de puntos ORB no es capaz de identificar un número de *features* suficiente para completar el mapeo de la zona.

El error anterior, en ORB-SLAM2 monocular, era 10 veces superior al obtenido con sensores estéreo. Nuestro sistema consigue acercar su precisión a la de ORB-SLAM2 estéreo, siendo el nuevo error 4 veces el cometido por el sistema estéreo, a pesar de que nuestra propuesta trabaja con una única cámara.

5.3. Influencia del recorte de profundidad

El método de filtrado de puntos, expuesto en el capítulo 4, provoca que solo los puntos situados de 7 a 20 metros sean introducidos al sistema como observaciones estéreo. En cambio, los puntos más cercanos o lejanos son tratados como monoculares. La figura 5.3 muestra dos capturas del visor de ORB-SLAM2. Este componente se ha modificado para dibujar de color rojo los puntos monoculares y en verde las observaciones estéreo.

La mejora introducida por este filtro es claramente beneficiosa. En la figura 5.4, se puede observar la diferencia entre una ejecución con predicciones de elevada incertidumbre, y el resultado al aplicar la corrección.

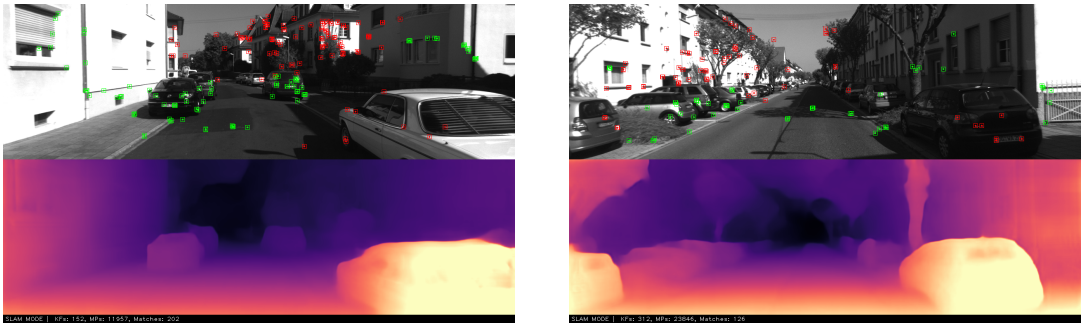
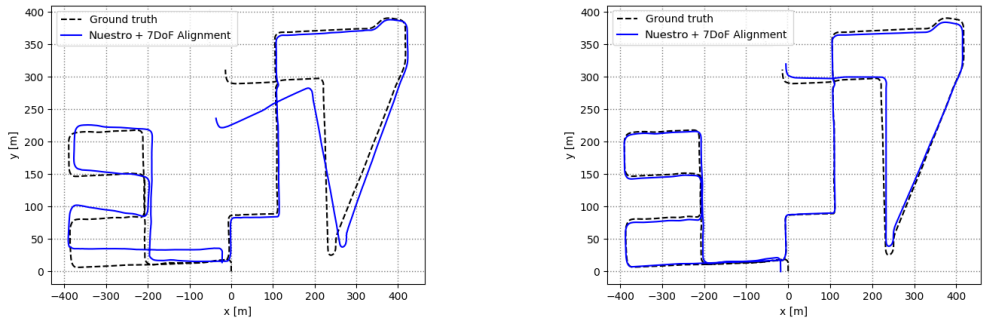


Figura 5.3: **Ejemplos del *Viewer* de ORB-SLAM2. Arriba:** Imagen monocular de entrada. Los puntos ORB detectados por el sistema se dividen entre monoculares (rojo) y estéreo virtual (verde). **Abajo:** Disparidad predicha por la red neuronal. Valores más oscuros corresponden con una menor disparidad y, por tanto, con zonas de más profundidad.



(a) Sin filtrado de puntos.

(b) Con filtrado de puntos

Figura 5.4: **Mejora al eliminar puntos con gran incertidumbre.** Los valores de profundidad erróneamente estimados provocan fallos en los giros de la trayectoria. Descartar la estimación para estos puntos permite obtener resultados mucho más precisos.

5.4. Ejecución en tiempo real

Tras la integración de ORB-SLAM2 y Monodepth2, el sistema es todavía capaz de funcionar en tiempo real. En la tabla 5.3, se recogen las mediciones del tiempo de ejecución para los distintos modos de funcionamiento de ORB-SLAM2 y para las dos principales tareas del nuevo sistema.

El conjunto de datos KITTI trabaja a 10 fotogramas por segundo, es decir, existe un límite temporal de 100 ms para procesar cada imagen. Actualmente, el nuevo sistema solo requiere aproximadamente un 60 % de dicho periodo.

El tiempo necesitado por el sistema ORB-SLAM monocular es inferior al del resto de alternativas analizadas, ya que ejecuta un menor número de tareas. En cambio, nuestro sistema requiere ejecutar la red neuronal Monodepth2 para cada fotograma de entrada.

No obstante, gracias a la paralelización implementada, se ha conseguido reducir el tiempo de ejecución de 85,52 ms a 59,02 ms. Nótese que el nuevo tiempo total es inferior a la suma de las dos principales tareas si fuesen ejecutadas secuencialmente:

$$16{,}72 \text{ ms} + 47{,}47 \text{ ms} = 64{,}19 \text{ ms} < 59{,}02 \text{ ms}$$

Finalmente, el tiempo medio de ejecución para procesar cada fotograma es de 59,02 ms, ligeramente superior al tiempo requerido por ORB-SLAM2 estéreo, que se encuentra en 56,49 ms.

El estudio temporal se ha realizado sobre un procesador Intel(R) Core(TM) i7-7700K con cuatro núcleos a 4,20 GHz y una tarjeta gráfica NVIDIA GeForce(R) GTX 1050 Ti.

Tabla 5.3: **Tiempo de ejecución del sistema.** Coste de procesar cada fotograma de la secuencia por el hilo de *Tracking*. Se resumen los tiempos para los modos previos de funcionamiento y se detalla el rendimiento temporal del nuevo sistema y de dos de sus partes principales.

Sistema	*Mediana (ms)*	*Media (ms)*
ORB-SLAM2 Monocular	24,37	28,52
ORB-SLAM2 Estéreo	54,84	56,49
ORB-SLAM2 + Monodepth2 sin paralelización	73,42	85,52
ORB-SLAM2 + Monodepth2 con paralelización CPU-GPU	57,60	59,02

Tarea	*Mediana (ms)*	*Media (ms)*
Extracción de puntos ORB	16,58	16,72
Obtención de disparidad	45,80	47,47

Capítulo 6

Conclusiones

6.1. Resultados obtenidos

Al integrar la profundidad predicha por la red Monodepth2 en el sistema ORB-SLAM2 monocular, somos capaces de determinar el tamaño real del entorno, sobrepasando las limitaciones introducidas por la falta de observabilidad de la escala en los sistemas de SLAM monocular convencionales. La principal ventaja del nuevo sistema es su capacidad de construir mapas consistentes del entorno y recuperar con buena precisión la distancia real recorrida por el agente.

El sistema estima la escala métrica real del entorno, con un error promedio inferior al 3 %. También se elimina la deriva de la escala que afectaba a la técnica puramente geométrica, pasando del 43 % a únicamente un 3 %. La nueva solución demuestra ser notablemente mejor que su predecesora en las cinco secuencias de evaluación con mayor deriva de la escala, especialmente en las secuencias KITTI 08 y 09.

El error promedio en las trayectorias de evaluación se ha reducido de 17,24 m a 6,70 m, acortando las distancias con el error de 1,70 m obtenido por ORB-SLAM2 estéreo, a pesar de utilizar una única cámara monocular.

Estos resultados abren la puerta a sistemas de SLAM más precisos en aplicaciones donde no se pueden utilizar cámaras estéreo por su mayor tamaño, como drones pequeños o endoscopios médicos.

La principal limitación del sistema es que la red neuronal debe ser entrenada en el mismo tipo de entornos en los que va a funcionar el sistema, ya que de momento su capacidad de generalización es muy limitada.

6.2. Trabajo futuro

Investigaciones futuras en redes neuronales podrían mejorar la generalización de los modelos, ya que actualmente sería necesario re-entrenar el sistema para que este sea capaz de funcionar, por ejemplo, en escenas de interior.

Por otro lado, existe un prometedor campo de trabajo aumentando el uso de redes neuronales profundas en problemas de SLAM. El análisis de precisión efectuado en el presente trabajo, si bien ayuda a mejorar el resultado, podría ser extendido para limitar más la influencia de predicciones con elevada incertidumbre. Por ejemplo, diseñando una red neuronal capaz de predecir no solo la profundidad de los puntos de una escena, sino también su incertidumbre [14].

Un interesante campo de investigación se encuentra en la adaptación de los avances propuestos por este trabajo a aplicaciones biomédicas, como el proyecto EndoMapper del grupo de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza, cuyo objetivo es el mapeo 3D en tiempo real de endoscopias a partir de secuencias de vídeo del interior del cuerpo humano, donde los métodos puramente geométricos sufren deriva de escala [15].

Bibliografía

[1] H. Durrant-Whyte y T. Bailey, «Simultaneous localization and mapping: part I,» *IEEE robotics & automation magazine*, vol. 13, n.º 2, págs. 99-110, 2006.

[2] R. Mur-Artal, J. M. M. Montiel y J. D. Tardos, «ORB-SLAM: a versatile and accurate monocular SLAM system,» *IEEE Transactions on Robotics*, vol. 31, n.º 5, págs. 1147-1163, 2015.

[3] R. Mur-Artal y J. D. Tardós, «ORB-SLAM2: An open-source slam system for monocular, stereo, and rgb-d cameras,» *IEEE Transactions on Robotics*, vol. 33, n.º 5, págs. 1255-1262, 2017.

[4] C. Godard, O. Mac Aodha, M. Firman y G. J. Brostow, «Digging into self-supervised monocular depth estimation,» en *Proceedings of the IEEE International Conference on Computer Vision*, 2019, págs. 3828-3838.

[5] A. Geiger, P. Lenz, C. Stiller y R. Urtasun, «Vision meets Robotics: The KITTI Dataset,» *International Journal of Robotics Research (IJRR)*, 2013.

[6] G. Bradski, «The OpenCV Library,» *Dr. Dobb's Journal of Software Tools*, 2000.

[7] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai y S. Chintala, «PyTorch: An Imperative Style, High-Performance Deep Learning Library,» en *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, págs. 8024-8035.

[8] J. W. Eaton, D. Bateman, S. Hauberg y R. Wehbring, *GNU Octave version 3.8.1 manual: a high-level interactive language for numerical computations*, ISBN 1441413006, 2014.

[9] J. D. Hunter, «Matplotlib: A 2D graphics environment,» *Computing in science & engineering*, vol. 9, n.º 3, págs. 90-95, 2007.

[10] R. Hartley y A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[11] R. Garg, V. K. BG, G. Carneiro e I. Reid, «Unsupervised cnn for single view depth estimation: Geometry to the rescue,» en *European Conference on Computer Vision*, Springer, 2016, págs. 740-756.

[12] A. Geiger, P. Lenz y R. Urtasun, «Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite,» en *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[13] J. W. Tukey, *Exploratory data analysis*. Reading, MA, 1977, vol. 2, ISBN 9780201076165.

[14] N. Yang, L. v. Stumberg, R. Wang y D. Cremers, «D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry,» en *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, págs. 1281-1292.

[15] J. Lamarca, S. Parashar, A. Bartoli y J. M. M. Montiel, *DefSLAM: Tracking and Mapping of Deforming Scenes from Monocular Sequences*, 2019. arXiv: `1908.08918 [cs.CV]`.

Apéndice A

Gestión del proyecto

Este proyecto se ha realizado en base de la investigación llevada a cabo con una Beca de Colaboración de Ministerio de Educación y Formación Profesional dentro del Departamento de Informática e Ingeniería de sistemas de la Universidad de Zaragoza. Su dedicación aproximada ha sido de 3 horas diarias (15 horas semanales). La tabla A.1 expone cuántas horas ha requerido cada una de las tareas de este proyecto. La figura A.1 muestra cómo se han distribuido esas tareas a lo largo del curso académico.

Tarea	*Dedicación*
Configuración del entorno de trabajo	40 h
Investigación en modelos de predicción de profundidad	55 h
Obtención de datos para el análisis de la precisión	35 h
Análisis de precisión a lo largo de la trayectoria	45 h
Análisis de precisión en profundidad	30 h
Análisis de precisión en plano de imagen	15 h
Integración de Monodepth2 en ORB-SLAM2	65 h
Evaluación del rendimiento del sistema	80 h
Redacción de la memoria del proyecto	75 h
Total	440 h

Tabla A.1: Horas dedicadas a cada tarea del proyecto.

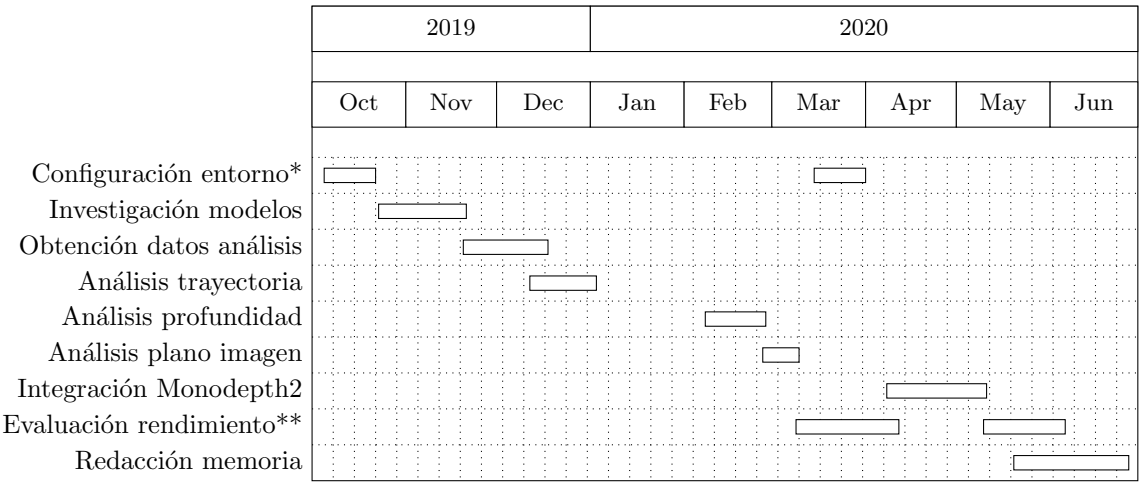


Figura A.1: Diagrama de Gantt del proyecto.
(*) Debido a la situación de confinamiento, se tuvo que habilitar un nuevo entorno de trabajo.
(**) Se realizó una pre-evaluación antes de la integración definitiva del sistema.

Apéndice B

Fragmentos de código

A continuación, se presentan una colección de extractos de código relevantes para entender la implementación algorítmica realizada en C++, sin detallar la declaración de algunas variables.

Algoritmo B.1: Inicialización de Monodepth2 en ORB-SLAM2.

```
void Tracking::loadCNN(cv::FileStorage fSettings) {
  /* cargar modelo neuronal en la GPU */
  encoder = torch::jit::load(fSettings["Monodepth.EncoderModel"]);
  encoder.to(at::kCUDA);
  decoder = torch::jit::load(fSettings["Monodepth.DecoderModel"]);
  decoder.to(at::kCUDA);
  modelSize = cv::Size((float)fSettings["Monodepth.ModelWidth"]
                       (float)fSettings["Monodepth.ModelHeight"]);
}
```

Algoritmo B.2: Procesamiento de un nuevo fotograma.

```
cv::Mat Tracking::GrabImageMonodepth(const cv::Mat &imRGB) {
  /* ejecutar la red neuronal en paralelo */
  std::future<void> futDisp = std::async(&ORB_SLAM2::Tracking::forwardCNN,
                                         this, std::ref(imBGR), std::ref(imDisp));
  /* crear un fotograma a partir de la imagen y su mapa de disparidad */
  mCurrentFrame = Frame(mImGray, imDisp, futDisp);
}

Frame::Frame(cv::Mat &imGray, cv::Mat &imDisp, std::future<void> &futDisp) {
  /* extraer los puntos ORB en paralelo */
  ExtractORB(0, imGray);
  /* esperar a la salida de la red nueronal */
  futDisp.wait();
  /* generar coordenadas estereo a partir de la disparidad */
  ComputeStereoFromMonodepth(imDisp);
}
```

Algoritmo B.3: Invocación a Monodepth2.

```
void Tracking::forwardCNN(cv::Mat& imRGB, cv::Mat& disparity) {
    /* reducir imagen a la dimension de entrada de la red */
    cv::resize(imRGB, modelSize);

    /* convertir la imagen a un tensor */
    torch::Tensor tensor_image = torch::from_blob(input_mat.data);
    /* transferir el tensor a la GPU */
    tensor_image = tensor_image.to(at::kCUDA);

    /* propagar la imagen a traves de las capas neuronales */
    auto result_encoder = encoder.forward(tensor_image);
    /* la salida del codificador se transfiere al decodificador */
    /* el tensor se encuentra en todo momento en la GPU */
    auto result_decoder = decoder.forward(result_encoder);
    /* recuperar los resultados finales desde la GPU */
    auto tensor_result = result_decoder.toTensor().to(at::kCPU);
    cv::Mat disparity = cv::Mat(tensor_result);

    /* convertir salida sigmoidal a valores de disparidad */
    float min_disp = 1 / 100;
    float max_disp = 1 / 0.1;
    disparity = min_disp + (max_disp - min_disp) * disparity;

    /* escalar la salida de la red a la resolucion original de la imagen */
    cv::resize(disparity, cv::Size(imRGB.cols, imRGB.rows));

}
```

Algoritmo B.4: Estimación de profundidad real y filtrado de puntos.

```
void Frame::ComputeStereoFromMonodepth(const cv::Mat &imDisp) {
    /* para cada uno de los puntos ORB extraidos */
    for (int i = 0; i < NUM_POINTS; i++) {
        /* calcular la profundad real a partir de su disparidad */
        float scale_factor = 5.4;
        depth = scale_factor / imDisp[i];
        /* si el punto se encuentra dentro de la profundidad recortada */
        if (minThreshold < depth && depth < maxThreshold) {
            /* calcular la coordenada horizontal derecha */
            /* es decir, se introduce como observacion estereo */
            uRight[i] = uLeft[i] - bf / depth;
        } else {
            /* no se calcula coordenada derecha */
            /* es decir, se introduce como observacion monocular */
            uRight[i] = -1;
        }
    }
}
```