UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

Instituto de Física y Matemáticas

Detección de texto para la inferencia de escala global en secuencias de video.

Tesis

Que para obtener el título de:

Licenciado en Matemáticas Aplicadas

Presenta:

Raziel Ruiz Vásquez

Director de tesis:
MC. Verónica Rodríguez López
Codirector de tesis:
Dr. Jean-Bernard Hayet

Huajuapan de León, Oaxaca *Octubre 2018*

Dedicatoria

Dedicado a
mi familia,
no solo de sangre.

Agradecimientos

¡Muchas gracias a todos!

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

La reconstrucción 3D es el proceso por el cual, dada una serie de imágenes de un objeto, se puede reconstruir este objeto en una computadora, estimando sus características físicas (dimensiones, volúmen y forma). Esta reconstrucción es una tarea fundamental para diversas aplicaciones de visión por computadora y robótica, como la realidad aumentada. La realidad aumentada consiste en sobreponer en tiempo real objetos o animaciones generadas por computadora sobre la imagen que recoge una cámara (Minguell et al., 2012).

En diversas aplicaciones de robótica o visión por computadora se utiliza una sola cámara de video para reconstruir escenarios. Sin embargo, dadas sus características, son incapaces de determinar de manera no ambigua la escala correcta a la cual debe de reconstruirse un escenario. En este trabajo de tesis, se pretende construir un sistema de inferencia Bayesiana que ayude a determinar este factor de escala, utilizando la información obtenida por el sistema de detección de texto propuesto en (Shi et al., 2017), y por el Sistema de Localización y Mapeo Simultáneos (SLAM, por sus siglas en inglés) propuesto en (Mur-Artal et al., 2015; Mur-Artal and Tardós, 2017).

Una idea que se desarrollará en esta tesis es que, imitando de cierta manera el funcionamiento del sistema visual humano, el factor de escala desconocido se determinará automáticamente a partir de objetos semánticos fáciles de detectar en un escenario, como son los textos. Es decir, se quiere aprovechar la información relativa a la naturaleza misma de los textos que se pueden encontrar en la imagen de un escenario. Por ejemplo, al ver una silla, los humanos podemos inferir naturalmente la profundidad de los pixeles de la silla porque sabemos relativamente bien qué tan grande es una silla en general. A priori, esos elementos semánticos son cualquier tipo de objeto reconocibles por su apariencia, y sobre los cuales se tiene un cierto conocimiento a priori de su tamaño. El sistema de detección de texto permitirá localizar objetos semánticos en la imágen de un escenario, y esta información sobre los objetos semánticos detectados (tipo

de texto y su tamaño), ayudará a corregir adecuadamente el factor de escala de reconstrucción proporcionado por el método visual de localización y mapeo.

1.1. Justificación

La reconstrucción 3D es un proceso fundamental para aplicaciones complejas de la visión por computadora, como la realidad aumentada. La realidad aumentada tiene la capacidad de añadir información sobre los objetos de un entorno, tales como texto, imágenes, objetos 3D, animaciones, entre otros. A fin de añadir correctamente esta información en un flujo de video, es necesario contar con una idea correcta de las dimensiones reales de los objetos de la escena y de la localización de la cámara con respecto a ellos. De ahí la importancia de este trabajo de tesis de construir un sistema de inferencia Bayesiana que ayude a determinar este factor de escala.

Existen diversas áreas importantes de aplicación de SLAM que involucran realidad aumentada, en donde podría ser útil el sistema a desarrollarse en esta tesis. Algunas áreas de aplicación son: militar, por ejemplo para tener controlada la posición del compañero, consultar mapas del terreno o recibir órdenes precisas en aviones de combate; también en medicina, en aplicaciones que ayuden a realizar cirugías.

En este trabajo de tesis, se desea desarrollar un sistema de inferencia de la escala de reconstrucción, que pueda ser utilizado en aplicaciones de realidad aumentada que involucre a escenarios pequeños. Para esto último, se combinará la información de dos algoritmos que han mostrado tener un buen desempeño en la detección de texto, por un lado, en la localización y mapeo simultaneo en un escenario pequeño, por otro lado.

1.2. Estado del Arte

Los sistemas mono SLAM (véase 2.2.2) han mostrado ser una de las mejores herramientas tanto para la reconstrucción de escenarios desconocidos, como para la estimación de la trayectoria de la cámara. Aunque estos sistemas tienen amplias aplicaciones, además de ser baratos, tienen dos limitaciones que alteran su correcto funcionamiento: la primera es que el factor de escala para la reconstrucción es desconocido, y la segunda, es que puede cambiar este factor cuando se estima de forma consecutiva, lo cual puede afectar la consistencia de los mapas globales (Sucar and Hayet, 2017).

Este último problema es conocido en la literatura como *scale drift* ó deriva de la escala. A continuación se realiza una muestra de las propuestas existentes para resolver este problema.

En (Davison, 2003), se presenta un modelo Bayesiano para resolver el problema de seguimiento en los algoritmos mono SLAM mediante el mapeo de un conjunto de características utilizando un modelado estocástico de movimiento y una estrategia de medición activa guiada por información. La inicialización del sistema en tiempo real se realiza mediante objetos conocidos o movimientos conocidos (elementos *ad-hoc*).

En (Salas-Moreno et al., 2013), se presenta un paradigma 3D para un sistema SLAM orientado a objetos suponiendo que la gran mayoría de las escenas contienen estructuras y objetos conocidos. Se utiliza una cámara de profundidad. El conocimiento a priori de los objetos permite su reconocimiento 3D en la escena en tiempo real, para después realizar la creación de un grafo de poses de las ubicaciones relativas de estos objetos. Este grafo se optimiza continuamente a medida que llegan nuevas mediciones, y permite la predicción actualizada, densa y precisa de la siguiente posición de la cámara.

En (Song et al., 2013), se presenta un sistema monocular que se ejecuta en tiempo real y en grandes escenarios, además de ser preciso. El sistema utiliza un mecanismo para la corrección de la escala usando la planaridad local de la carretera con la información proveída por triangular puntos 3D y la homografía plana entre imágenes. Es decir, la escala es corregida con base en el conocimiento de la altura de la cámara sobre el suelo, y en la suposición de planaridad local en la escena observada. Este sistema se basa en las innovaciones multiproceso de *structure-from-motion* (SFM, por sus siglas en inglés) para lograr un mejor rendimiento en términos de tiempo y precisión. La búsqueda epipolar opera en paralelo con otros subprocesos para generar nuevos puntos 3D en cada fotograma (*frame*). El mismo enfoque para evitar deriva de la escala se utiliza en (Gräter et al., 2015a,b).

En (Bourmaud and Megret, 2015), se presenta una solución para la deriva de la escala, la cual consiste en explotar submapas para después estimar el mapa global. En este trabajo se introduce un formalismo que se basa en el concepto conocido como *Known Rotation Problem*, para estimar submapas robustos. También se propone un algoritmo llamado *loopy belief propagation*, para alinear eficientemente una gran cantidad de submapas. Además su propuesta incluye un algoritmo simple y eficiente para eliminar los valores atípicos que se detectan en el grafo de similitudes 3D relativas.

En (Mur-Artal et al., 2015), se presenta un sistema mono SLAM basado en características ORB (véase 2.2.3). Opera en tiempo real, en pequeños y grandes ambientes, así como en interiores y exteriores. Este sistema utiliza las mismas características detectadas para realizar las tareas de: seguimiento, mapeo, relocalización y cierre de ciclos. Este último es la solución que

el autor propone para evitar el problema de deriva de la escala. El proceso de cierre de ciclos busca ciclos cada vez que se agregan nuevos *keyframes*. Cuando se detecta un ciclo, se calcula la transformación de similitud que informa sobre la deriva acumulada en el ciclo y se realiza una optimización del gráfo de poses basándose en las restricciones de similitud para lograr la consistencia global.

En (Gálvez-López et al., 2016), al igual que en el anterior, se utilizan elementos *ad-hoc* tanto para la inicialización como para evitar el problema de deriva de la escala. Para la detección de los objetos, se utiliza un algoritmo de reconocimiento de objetos basado en bolsas de palabras binarias. Estos elementos conocidos pertenecen a una base de datos con más de 500 objetos $3D$. Una vez encontrados estos objetos, la información alimenta a un sistema mono SLAM que explota las restricciones de rigidez del objeto para mejorar el mapa y encontrar la escala real.

En (Frost et al., 2016), se presenta un sistema que propone la detección de objetos para controlar la deriva de la escala. Se utiliza un conocimiento a priori sobre objetos. Estos objetos, son separados por clases de acuerdo a su tamaño. Cuando estos objetos están detectados en la escena, la escala global se determina conjuntamente con la pose de la cámara en ajustes locales. El sistema realiza constantemente este proceso, lo que implica resolver un problema de asociación de datos.

En (Knorr, 2017), el autor propone utilizar la cara del usuario como un objeto conocido, es decir, se utilizan imágenes de la cara para deducir información sobre la iluminación y para estimar la escala real del entorno. La cara se utiliza como sonda de luz. Primero se aprende cómo la cara promedio refleja la luz incidente desde cierta dirección hacia la cámara. Por otro lado, la cara se utiliza como objeto conocido, para resolver la ambigüedad de escala, cuando se reconstruye el mundo real mediante un sistema monocular. Para hacer esto, se calcula la distancia entre los ojos del usuario para después determinar la posición y orientación de la cámara con respecto a la cara en unidades como metros.

Algunos de los trabajos mencionados comparten similitudes con el trabajo presentado en esta tesis (detección de objetos, conocimiento a priori de algunos objetos). Sin embargo la diferencia radica en el desarrollo del sistema de inferencia para estimación de la escala. Además, se combina la información de dos algoritmos: un detector de texto orientado que utiliza Redes Neuronales Convolucionales y un reconstructor mono SLAM que utiliza características ORB.

1.3. Objetivo General

Implementar un sistema de inferencia Bayesiana que permita determinar el factor de escala de reconstrucción de un escenario, a partir de textos detectados en ese escenario y de conocimiento estadístico a priori sobre los tamaños de estos textos.

1.4. Objetivos Particulares

1. Construir una base de conocimientos sobre tamaño de textos a partir de bases de datos de textos encontrados acerca de un tipo de escenario dado (por ejemplo, una oficina).
2. Implementar el detector de texto en imágenes naturales propuesto en (Shi et al., 2017).
3. Implementar el sistema SLAM propuesto en (Mur-Artal and Tardós, 2017).
4. Modelar e implementar un sistema de inferencia Bayesiana, que considere la información a priori sobre el tamaño del texto, para determinar el factor de escala de reconstrucción.

1.5. Metodología

El desarrollo de este trabajo comprenderá las siguientes fases:

1. Revisión e implementación de los sistemas de detección de texto del estado del arte.
2. Revisión e implementación de los sistemas de SLAM visuales monoculares del estado del arte.
3. Construcción de una base de datos con información sobre textos clave que ayuden a identificar el factor de escala.
4. Construcción del sistema Bayesiano que, utilizando la información del detector de texto y del sistema SLAM, determine una distribución sobre el factor de escala.

Se usará inicialmente Matlab como base de implementación; finalmente el sistema de inferencia Bayesiano se implementará en C/C++. Además, los escenarios serán controlados (oficina, aula, sala), es decir, se tendrá un control de los textos así como de su información, además de tener una información estadística sobre las dimensiones reales de los objetos del escenario.

Capítulo 2

Marco Teórico

2.1. Detección de Texto en Imágenes Naturales

La localización automática de texto y el reconocimiento de objetos en imágenes han sido uno de los principales retos en áreas como visión por computadora. La mayoría de los algoritmos usan características de bajo nivel como: color, textura o formas. Estos elementos proporcionan una idea precisa del contenido de una imagen, a diferencia de las características de alto nivel: rostros o texto (Chen et al., 2004; Epshtein et al., 2010). La intervención de métodos que se comportan de manera autónoma para el reconocimiento de características de alto nivel ha permitido el desarrollo de sistemas más capaces para esta tarea.

Al inicio, la mayor parte del trabajo en detección de texto se basó en el Reconocimiento Óptico de Caracteres (OCR, por sus siglas en inglés). El precursor de esta área fue Emmanuel Goldberg en 1914. El objetivo de estos sistemas consiste en la conversión de imágenes de texto (escrito a mano o impreso en los documentos escaneados) a archivos que puedan ser electrónicamente compartidos, almacenados y presentados (González Arroyo, 2013). Estos sistemas han sido entrenados para la detección de texto escaneado, y dependen de la segmentación que separa completamente el texto de los pixeles de fondo. Las imágenes mantienen dos tonos de grises. El problema se complica cuando se trata de detectar texto dentro de imágenes naturales, ya que deben considerarse otros factores como menor cantidad de textos, tamaño de texto, ruido en el color, borrosidad, oclusiones, múltiple variedad de fondo, etc. (Herrera Alonso, 2015).

2.1.1. Aprendizaje Profundo: Redes Neuronales Convolucionales

Aprendizaje Profundo

El Aprendizaje Profundo tiene sus orígenes (no formales) alrededor de los años 80, y se puede definir como una familia de algoritmos y técnicas basados en aprendizaje automático para lograr que una máquina aprenda de la misma forma que lo hace el ser humano. Su meta principal es simular el comportamiento que éste tiene para reconocer imágenes, palabras o sonidos, tratando de descomponer el problema en subproblemas a diferentes niveles de abstracción. Este concepto se dio a conocer por los trabajos de Geoffrey Hinton (García Navarro, 2015).

Algunas de las razones por las cuales surge este concepto, es por las limitaciones que presentan las redes neuronales convencionales con respecto a su arquitectura o algoritmos de entrenamiento. Por ejemplo, el algoritmo de entrenamiento de retropropagación, como su nombre lo indica, propaga el error generado por la red en la capa de salida, hacia las entradas, y cuando la red presenta demasiadas conexiones, este algoritmo puede verse afectado por un mínimo local de la función de error y por tanto no podrá llegar al mínimo global deseado. Otro inconveniente es la limitación que existe en el hardware para la implementación de una red neuronal de múltiples capas (Izaurieta and Saavedra, 2000; López and Fernandez, 2008; Rubio et al., 2006; Villalba et al., 2012; Acevedo Orduña et al., 2011; Bowen et al., 2011; Restrepo Arteaga et al., 2015).

Los modelos utilizados comúnmente para el desarrollo de las técnicas de aprendizaje profundo son (Restrepo Arteaga et al., 2015): Autocodificadores, Máquinas de Boltzmann Restringidas, Redes de Creencia Profunda y Redes Neuronales Convolucionales. Esta última será nuestro objeto de estudio, dado que la detección de texto se hace con este tipo de técnica.

Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales (*Convolutional Neural Network*-CNN) tienen como motivación imitar a la estructura del sistema visual animal. Son redes neuronales multicapa jerárquicas, cuyo nombre se deriva de su funcionamiento y arquitectura.

Algunos de los primeros trabajos en estos campos fueron los de Hubel y Wiesel en la corteza visual del gato, mono rhesus y macaco. Registraron los impulsos generados al estimular determinadas áreas de sus retinas, y encontraron capas de células que son sensibles a pequeñas sub-regiones del campo visual llamadas *campos receptivos*[1]. Estas regiones están colocadas de tal manera que cubren todo el campo visual. Están conformadas por dos tipos de células: simples, las

[1]área ó región de la neurona biológica en la cual la presencia de un estímulo altera la respuesta de dicha neurona, esto es, la tasa de impulsos electroquímicos que ésta genera.

cuales extraen características locales, y complejas, que se encargan de la abstracción e identificación de objetos, además de ser tolerantes a deformaciones o traslaciones. El nombre de las CNN se deriva de su funcionamiento y arquitectura, ya que son redes neuronales multicapa jerárquicas.

Las CNN cuentan con propiedades importantes como su escasa conectividad (como se observa en la Figura 2.1) y la compartición de pesos para ayudar a reducir la cantidad de parámetros a ser aprendidos en el entrenamiento, y así poder centrarse en regiones locales mínimas y ser tolerantes a traslaciones. Las neuronas de una CNN, se disponen en forma tridimensional así que cada capa de neuronas tendrá largo, ancho y profundidad (Varios, 2015), (Serimagmedia, 2015).

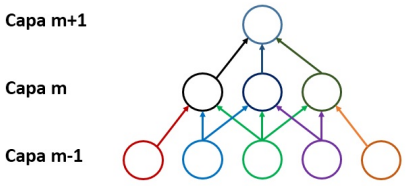


Figura 2.1: Conectividad de CNN (Varios, 2015)

Las CNN, como se muestra en la (Figura 2.1), están formadas por tres tipos de capas (Restrepo Arteaga et al., 2015),(LeCun et al., 1998b):

1.- Capa de Convolución: Esta capa extrae las características de los datos de entrada por medio de la convolución entre dichos datos y filtros. Algunas de las características que se pueden extraer en esta capa son bordes, esquinas y/o cruces. En la Figura 2.2 se aprecia la operación de convolución sobre una porción de la imagen de entrada de un solo canal y de dimensiones 5×5, con un filtro de un solo canal y dimensiones 3×3.

2.- Capa de combinación o submuestreo: Esta capa es usada para obtener una representación de las características que sea invariante contra pequeñas traslaciones y distorsiones. Esto se logra al combinar las respuestas obtenidas en la capa de convolución. La salida de la capa depende del tipo de método que seleccione el usuario: promedio, valor máximo (*Max-pooling*), o submuestreo, como se presenta en la Tabla 2.1.1. En la Figura 2.3, se muestra un ejemplo del cálculo del valor máximo de tamaño 2×2, con una porción de la imagen de un solo canal y tamaño 4×4.

3.- Capa completamente conectada: Esta es la última capa y es la encargada de clasificar y etiquetar a cada clase. Para esto, las capas anteriores han hecho que las características

Figura 2.2: Cálculo de la convolución sobre una imagen de un solo canal y tamaño 5×5 y un filtro de tamaño 3×3 (Chuliá, 2016).

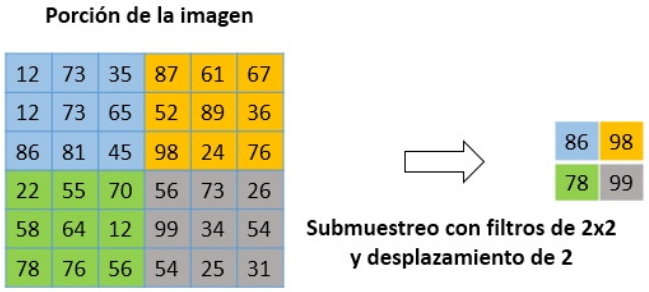


Figura 2.3: Cálculo de *Max-pooling* de tamaño 2×2 y un desplazamiento de tamaño dos, (Chuliá, 2016)

estén en baja resolución para que sea posible alimentar a una red neuronal totalmente conectada. La salida de las capas anteriores a esta capa son mapas de características unidimensionales y las salidas son procesadas por una función de la forma:

$$y_j = f_j\left(\sum_i w_{ij}x_{ij} + b_j\right),$$

donde w_{ij} son los pesos sinápticos, x_{ij} es la salida de la $j-1$-capa, b_j es el sesgo y f_j es una función de activación. En las Tablas 2.1.2 y 2.1.3 se muestra la descripción y la sugerencia de algunas funciones de activación para las capas de una CNN.

Algunos ejemplos exitosos de arquitecturas CNN son:

- LeNet-5

Tipo de submuestreo	Implementación	Descripción
Max-pooling	$\underset{n\times n}{\text{máx}}(x_i)$	Se saca el máximo valor x_i en una región cuadrada de tamaño $n \times n$.
Promedio ó *Downsampling*	$\underset{n\times n}{mean}(x_i)$	Se saca el promedio de los valores x_i de los pixeles en una región cuadrada de tamaño $n \times n$.
Submuestreo ó *Subsampling*	$\tanh\left(\beta \sum_{r,r} \frac{x_i}{r^2} + b\right)$	Se toma el promedio de las entradas de tamaño $r \times r$, se multiplican por un escalar entrenable (β) y se suma un bias entrenable (b). Se calcula el resultado de la función no lineal tanh.

Tabla 2.1.1: Funciones aplicables a la capa de submuestreo en una CNN (Restrepo Arteaga et al., 2015).

Nombre	Función	Descripción
Rectified Linear Units (ReLU)	$f(x) = \text{máx}(x, 0)$	Si la salida x de la red es positiva entonces la salida es x, en caso contrario es 0.
Identidad	$f(x) = x$	La entrada x de la red será la misma salida de la función.
Tangente hiperbólica	$f(x) = \frac{2}{1+e^{-2x}} - 1$	Es una función no lineal. La salida x de la red quedará en un rango de $[-1, 1]$.
Softmax	$f(x_j) = \frac{e^{x_j}}{\sum_K e^{x_j}}$	Utilizada para problemas de clasificación. La salida x de la red quedará en un rango de $[0, 1]$ y la suma total de las salidas es igual a 1.

Tabla 2.1.2: Funciones aplicables a la capa de submuestreo en una CNN (Restrepo Arteaga et al., 2015).

Tipo de capa	Función de activación
Convolucional	Identidad
Max-pooling	ReLU
Complematemente conectada (capa oculta)	ReLU
Complematemente conectada (capa de salida)	Tangente hiperbólica, Softmax

Tabla 2.1.3: Funciones de activación para una CNN (Restrepo Arteaga et al., 2015).

Es la primera CNN desarrollada por Yann LeCun. Consiste de 7 capas, y el tamaño de entrada son porciones de imágenes de tamaño 32×32. En esta arquitectura se aplica el patrón de convolución y submuestreo dos veces, reduciendo la imagen hasta un arreglo de 5×5, y después dos capas completamente conectadas. En la Figura 2.4 podemos ver esta arquitectura.

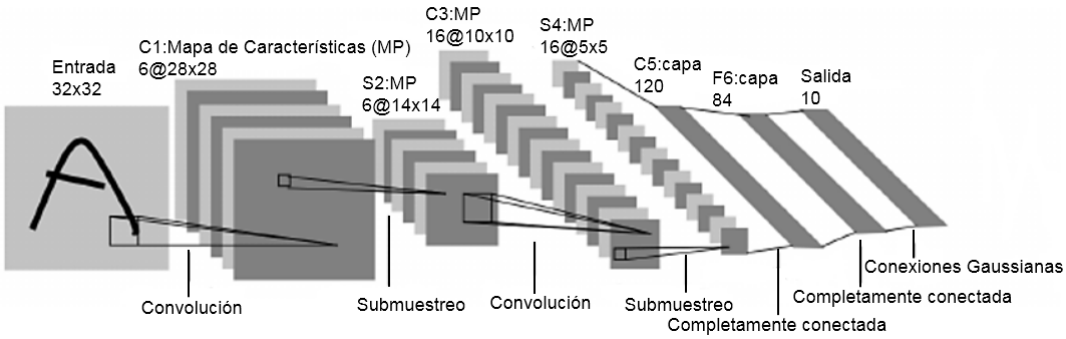


Figura 2.4: Arquitectura de la red LeNet-5, cuyo objetivo es la clasificación de dígitos (LeCun et al., 1998a)

.

- AlexNet
  Esta red fue desarrollada por Alex Krizhevsky, Ilya Sutskever y Geoff Hinton y fue entrenada con la base de datos ImageNet ILSVRC-2012 en 2 GPUs durante una semana. La red consiste de 8 capas: las primeras 5 son convolucionales y el resto son capas completamente conectadas. En la Figura 2.5, podemos apreciar dos caminos, los cuales muestran el proceso de cada GPU. Podemos ver en líneas punteadas las capas en las cuales los GPUs se comunican.

- Zeiler (Clarifai)
  Desarrollada por M.D. Zeiler y R. Fergus. Es una CNN con 5 capas convolucionales, seguidas de 3 capas completamente conectadas. En la Figura 2.6, en la capa 1, se aplican filtros de convolución de tamaño 7×7. Después, se hace *max-pooling* sobre regiones de tamaño 3×3, generando 96 mapas de prominencia de tamaño 55×55. Operaciones similares son repetidas de la capa 2 hasta 5. Las siguientes capas son completamente conectadas. Finalmente, la última capa utiliza la función $softmax$.

- GoogleNet
  Desarrollada por Christian Szegedy y Wei Liu. Es una CNN con 22 capas y utiliza 12

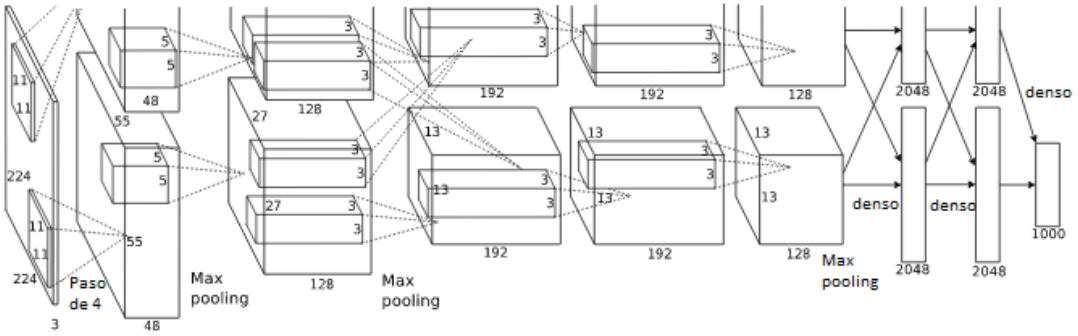


Figura 2.5: La arquitectura AlexNet tiene una arquitectura similar a LeNet, sin embargo, es más profunda y el entrenamiento es más rápido debido a que se utiliza una función ReLu, con más filtros por capa y con capas convolucionales apiladas (Krizhevsky et al., 2012).

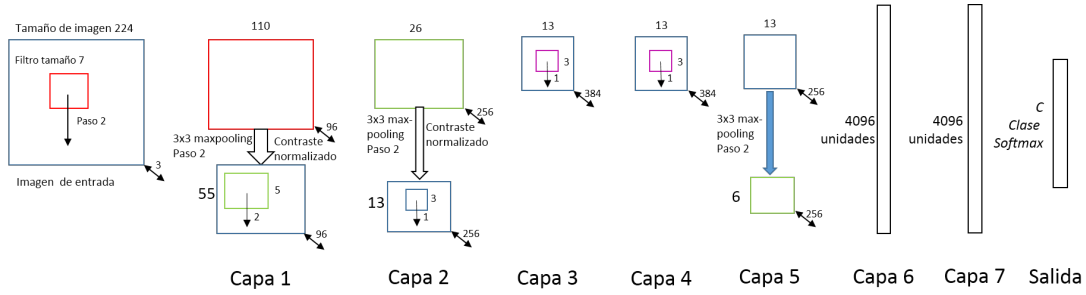


Figura 2.6: La arquitectura de la CNN Zeiler, con 8 capas. Las entradas son imágenes de tamaño $224 \times 224 \times 3$ (Zeiler and Fergus, 2014).

veces menos parámetros que la arquitecura AlexNet. Utiliza varios cálculos en paralelo y la operación de submuestreo sin reducir el tamaño de la imagen. Los autores agregan un concepto nuevo llamado *Inception Layer*, cuya idea es cubrir un área más grande pero manteniendo la resolución de la imágen para obtener información pequeña sobre la imágen (Szegedy et al., 2015).

- VGGNet
  Es una red desarrollada por Karen Simonyan y Andrew Zisserman. Tiene una profundidad de entre 16 y 19 capas. Utiliza filtros de convolución de tamaño 3×3, ya que los autores consideran que la combinación de dos capas de este tamaño es equivalente a un filtro de tamaño 5, además de la disminución en el número de parámetros. También, se utiliza la operación de *maxpool* con un tamaño de 2. En la arquitectura, se puede observar que el número de filtros se duplica después de cada capa de *maxpool*, reduciendo las dimensiones espaciales, pero aumentando la profundidad. En la Tabla 2.1.4 se presentan seis de las con-

Configuración de ConvNet					
A	A-LRN	B	C	D	E
11 capas de peso	11 capas de peso	13 capas de peso	16 capas de peso	16 capas de peso	19 capas de peso
Entrada (imagen RGB 224 x 224)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
softmax					

Tabla 2.1.4: Configuraciones de ConvNet. Los parámetros de las capas de convolución están denotados por "conv (tamaño del campo receptivo)- (número de canales)", "FC (completamente conectada)-(número de canales)", y *Local Response Normalisation* (LRN, por sus siglas en inglés) (Simonyan and Zisserman, 2014).

figuraciones que se han realizado con la arquitectura de ConvNet. Siendo la configuración de 16 capas la que produce el mejor resultado. La configuración de pesos de VGGNet16 (columna C) está a disposición del usuario.

Algunas otras arquitecturas conocidas pero no menos importantes son: ZFNet y ResNet. A continuación presentaremos algunos de las aplicaciones de las CNN en detección de texto.

2.1.2. Sistema de Detección de texto Orientado en Imágenes Naturales

Un sistema de detección de texto es todo aquel que a partir de una imagen cualquiera identifica dónde están localizadas las zonas con texto. Las zonas a localizar son las unidades mínimas del texto, es decir, las palabras. Este tipo de sistema está formado por tres bloques (Herrera Alonso, 2015; Jaderberg et al., 2014):

1. **Clasificador de texto.** Es el proceso que, a partir de una imagen, determina para cada pixel si éste representa a texto o no. Este clasificador se basa en la creación de un mapa de prominencia (ver Figura 2.7) a partir de la imagen original.

Figura 2.7: Mapa de prominencia para hallar el número 7.

2. **Detector de líneas de texto.** Es el componente que permite localizar frases o conjuntos de letras alineadas en una imagen a partir de un mapa de prominencia.

3. **Detector de palabras**: Permite encontrar las palabras que existen en la imagen a partir de las líneas de texto.

A continuación describiremos el sistema de detección de texto utilizado para este trabajo.

Detección de Texto Orientado en Imágenes Naturales por Enlace de Segmentos

Este sistema es conocido como SegLink, y fue desarrollado por Baoguang Shi et al. La diferencia con los sistemas planteados en (Chen et al., 2004; Epshtein et al., 2010; Serrano and Hernández, 2015; González Arroyo, 2013; Herrera Alonso, 2015), es que no sólo detecta texto horizontal sino también texto con cierta orientación, además de ser lo suficientemente rápido para aplicaciones en tiempo real, robusto contra fondos complejos, eficiente y general. Tiene la capacidad de procesar más de 20 imágenes de tamaño 512×512 por segundo, y general ya que puede detectar largas líneas de texto no latino, como el chino.

Dentro de una escena, la detección de texto puede ser vista como la detección de objetos aplicada a textos. La idea principal para detectar el texto dentro de una imagen natural es

descomponer el texto en dos elementos localmente detectables, como son: segmentos y enlaces. Un **segmento** es una caja orientada que cubre parte de una palabra o línea de texto. Un **enlace** conecta a dos segmentos adyacentes, indicando que estos pertenecen a la misma palabra. Entonces, una palabra es localizada por un número de segmentos enlazados. En la Figura 2.9 se puede observar el funcionamiento general de SegLink. Se utilizan imágenes con palabras de diferente tamaño y orientación.

Figura 2.8: Descripción general de SegLink. a) Se detectan segmentos (cuadros amarillos) en la imagen. b) Los enlaces (líneas verdes) se detectan entre pares de segmentos adyacentes. c) Los segmentos conectados por enlaces se combinan en palabras completas. (d-f) Seglink puede detectar largas líneas de texto y además texto de idiomas diversos, como el chino (Shi et al., 2017).

La detección de segmentos y enlaces se realiza mediante una CNN completamente convolucional[2] siguiendo el enfoque de SSD[3], con la diferencia de detectar los elementos que comprenden la palabra de manera ascendente. Estos elementos son combinados para formar palabras completas conforme a los enlaces. Las ventajas de utilizar estos elementos localmente detectables es

[2]Esta red no tiene capas completamente conectadas, en su lugar tiene filtros llamados *deep filters*, que pueden manejar diferentes tamaños de entradas (Long et al., 2015).

[3]Este modelo propone la idea de detectar objetos en múltiples capas de características mediante el uso de predictores convolucionales.

que no importa el tamaño del texto ni la orientación, así que el texto puede ser detectado con gran flexibilidad y eficiencia.

La arquitectura de la CNN está basada principalmente en VGG-16[4]. Las capas totalmente conectadas de VGG-16 se cambian a capas convolucionales (FC6 a conv6; FC7 a conv7). Seguidas por capas convolucionales (conv8-1 a conv11), las cuales extraen características más profundas con campos receptivos más grandes. A seis de las capas de convolución, se le agregan Predictores Convolucionales[5] para detectar segmentos y enlaces a diferentes escalas. Un predictor genera 16 canales para los enlaces a la red de 8 vecinos conectados. Cada dos canales se encuentra una capa de softmax para normalizar y obtener el puntaje de un enlace. En la Figura 2.9 podemos observar la arquitectura de la CNN.

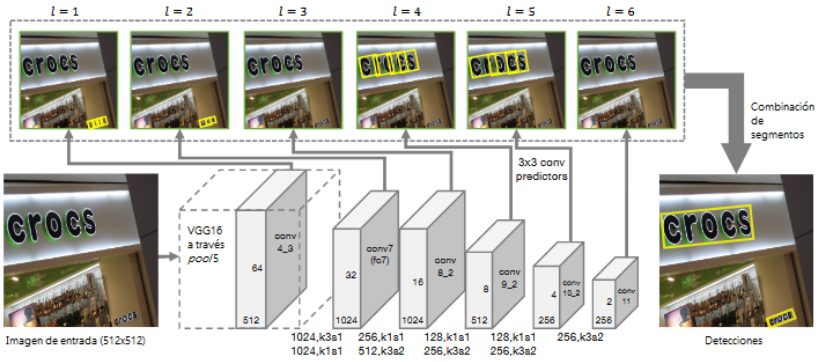


Figura 2.9: Los elementos que conforman la arquitectura de la red son: capas de características (bloques en color gris) y predictores convolucionales (flechas grises). Los filtros de convolución se especifican en el siguiente formato "(#filtros), k(tamaño de kernel), s(pasos)". Los filtros multilínea especifican capas ocultas. Los segmentos (cuadros amarillos) y los enlaces (no se muestran) son detectados por predictores convolucionales en múltiples capas de característica (indexadas con $l = 1, 2, ..., 6$) y combinadas para formar palabras completas mediante un algoritmo de combinación (Shi et al., 2017).

Para tratar con las detecciones redundantes, se introducen dos tipos de enlaces: enlaces dentro de las capas (*within-layers links*), que conectan a segmentos con sus vecinos en la misma capa de la red, enlaces de capas cruzadas (*cross-layer links*), que conectan un segmento a sus vecinos en la capa inferior. La finalidad es conectar segmentos de ubicaciones adyacentes, así como los de diferente escala. En la Figura 2.11 podemos ver un ejemplo de estos conceptos con la notación

[4]Es VGGNet con 16 capas, utilizado por el Grupo de Geometría Visual (VGG) de la Universidad de Oxford en ILSVRC-2014.

[5]Producen un conjunto fijo de predicciones de detección utilizando un conjunto de filtros convolucionales (Liu et al., 2016).

mostrada en la Tabla 2.1.4. Para encontrar segmentos conectados se utiliza un algoritmo de búsqueda en profundidad (DFS) para después combinarlos en palabras completas.

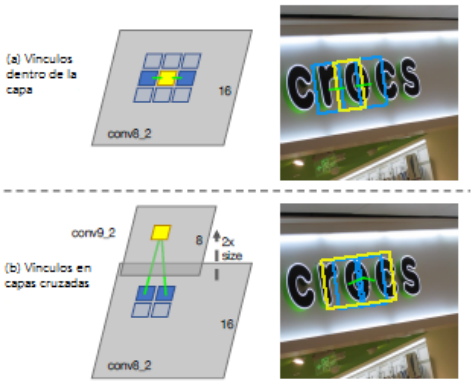


Figura 2.10: En a) Una posición en conv8-2 (bloque amarillo) con conectividad 8 (bloques azules con y sin relleno). Los enlaces detectados dentro de la capa (líneas verdes) conectan un segmento (cuadro amarillo) y sus dos segmentos vecinos (cuadros azules) en la misma capa. Este es un ejemplo de enlace dentro de la capa. En b) los enlaces de la capa cruzada conectan un segmento en conv9-2 (recuadro amarillo) y dos segmentos en conv8-2 (recuadros azules). Este es un ejemplo de enlaces de capas cruzadas (Shi et al., 2017).

El sistema SegLink es novedoso en detección de texto implementado en una CNN simple y altamente eficiente. Las ventajas que ofrece sobre otros sistemas es la detección de texto horizontal, orientado y es multilenguaje. Es decir, detecta texto latino y texto chino. Es por estas razones que se ha elegido este sistema. Al realizar el desplazamiento en el escenario desconocido será de manera paralela a la orientación de una puerta, pared o ventana. En estas áreas es común encontrar zonas de texto. Por lo tanto, estas zonas de texto tendrán cierta inclinación con respecto a la posición de la cámara.

2.2. Localización y Mapeo Simultáneos basados en Visión

2.2.1. El problema de SLAM

SLAM es una técnica utilizada por robots o vehículos autónomos para navegar en un entorno desconocido. La idea es construir el mapa de manera incremental sin perder su posición dentro de éste, utilizando únicamente sus sensores a bordo (Francisco and Antoni, 2013).

La técnica de SLAM consiste en (Pons, 2012):

- Extracción de características. En esta etapa se buscan características (*features*) del entorno que puedan ser fácilmente distinguidas y reobservables, de tal manera que el robot pueda utilizarlos para autolocalizarse.
- Asociación de datos. La idea de este paso es hacer la correspondencia entre características observadas en una iteración anterior y en la actual.
- Estimación del estado. Una vez que se detectaron nuevas características y su asociación, se procede a estimar la posición actual del robot en el mapa.
- Actualización de las características. En este paso se añaden nuevas características al mapa del robot y se reinicia todo el proceso.

2.2.2. SLAM monocular

El SLAM visual monocular o mono SLAM, fue propuesto por A.J. Davison en el 2003. Se define como la técnica con la cual un robot o vehículo autónomo opera una sola cámara para crear un mapa tridimensional y localizarse a sí mismo utilizando objetos comunes del entorno mediante puntos de referencia (Davison, 2003).

Algunas metodologías de SLAM que se han desarrollado son:

- Enfoque basado en filtrado:
  Cada fotograma es procesado por el filtro como una observación para estimar conjuntamente las ubicaciones de las características del mapa y la pose de la cámara. Algunos de los inconvenientes son la acumulación de errores de linealización, así como la pérdida de capacidad de cómputo cuando se procesan fotogramas consecutivos.
- Enfoque basado en fotogramas clave (*keyframe*):
  Calculan el mapa utilizando sólo fotogramas seleccionados (fotogramas clave), lo que permite realizar optimizaciones de Ajuste de Haces (BA, por sus siglas en inglés) más costosas pero más precisas, pues, el mapeo no está vinculado a la velocidad de los fotogramas.

Una desventaja que presenta mono SLAM, es que la profundidad no puede inferirse directamente de una sola cámara, y como consecuencia no se pueden inicializar puntos de referencia a buena escala. Pare resolver este problema se han utilizado Filtro de Kalman Extendido (EKF, por sus siglas en inglés) y parametrización con profundidad inversa (Davison, 2003; Civera et al., 2009). Algunos de los sistemas mono SLAM más conocidos son:

- PTAM:
  Fue desarrollado por Klein y Murray en el 2007. Es un sistema robusto y ligero. Tiene la capacidad tanto de calcular la posición de una cámara en un entorno de tres dimensiones, como de poder estimar una nube de puntos para asignar la posición de los objetos visibles mediante análisis y procesamiento de la información, todo esto a través de secuencias de video. La idea del algoritmo PTAM es llevar a cabo las tareas de localización y mapeo en paralelo (Klein and Murray, 2007).

- LSD-SLAM:
  LSD hace referencia a "*Large-Scale Direct*", y es una técnica de SLAM monocular que permite la construcción a gran escala de mapas consistentes del entorno. En lugar de utilizar puntos de referencia claves, opera directamente sobre las intensidades de la imagen, tanto para el seguimiento como para el mapeo. LSD-SLAM no utiliza esquinas para comparar imágenes, la cámara se rastrea utilizando alineación directa de imágenes, mientras que la geometría se estima en forma de mapas de profundidad semi-densos, obtenidos mediante el filtrado de muchas comparaciones estéreo en píxeles. El algoritmo consta de tres componentes: el seguimiento, la estimación del mapa de profundidad y la optimización del mapa. Una desventaja del sistema, es que al utilizar métodos directos no es muy flexible, y esto dificulta la eliminación de datos atípicos en los datos recopilados (Engel et al., 2014).

- ORB-SLAM:
  Está basado en el sistema PTAM, y mejora algunos inconvenientes del sistema PTAM como es: limitación a operaciones en escalas pequeñas, falta de cierre de ciclos, manejo inadecuado de oclusiones, baja invarianza al punto de vista durante la relocalización y necesidad de la intervención humana para el arranque del mapa. Lo describimos más a detalle a continuación.

2.2.3. ORB-SLAM

Es un sistema SLAM monocular basado en características ORB u *Oriented FAST and Rotated BRIEF*. ORB es un detector binario de características locales desarrollado por Ethan Rueblee en el 2011. Este descriptor es utilizado para reconocimiento de objetos y reconstrucción 3D. Está basado en el detector FAST (*Features from Accelerated Segment Test*) utilizado para

detección de las esquinas y en el descriptor visual BRIEF (*Binary Robust Independent Elementary Features*). Por tanto, las características ORB son binarias e invariantes a rotaciones, ruido y a cambios de escala (Rublee et al., 2011). ORB-SLAM funciona en tiempo real, en ambientes pequeños y grandes, interiores y exteriores. El sistema selecciona los puntos de referencia y los fotogramas claves de la reconstrucción para una excelente robustez y genera un mapa compacto y rastreable que sólo crece si el contenido de la escena cambia (Mur-Artal et al., 2015). Funciona en tiempo real en CPUs estándar en entornos pequeños, para navegación de drones en entornos industriales y coches que circulan por una ciudad. Permite la estimación precisa de la trayectoria con escala métrica. El sistema es robusto a movimiento violento y desordenado (Mur-Artal and Tardós, 2017).

ORB-SLAM, como ya se mencionó, está basado en el sistema PTAM. Este último, aunque es limitado a operaciones a escalas pequeñas, proporciona métodos efectivos para la selección de fotogramas claves, empate (*matching*) de características, puntos de triangulación, localización de la cámara en cada fotograma clave y relocalización después de seguimientos fallidos. Este sistema utiliza las mismas características para todas las tareas de SLAM: seguimiento, mapeo, relocalización y cierre de bucles. A continuación se describen las principales tareas del sistema y en la Figura 2.11 se muestra este funcionamiento (Mur-Artal et al., 2015):

- Seguimiento:
  Se encarga de localizar la cámara con cada fotograma y decide cuándo se inserta un nuevo fotograma clave. Para esto, primero se realiza un empate con las características iniciales del cuadro anterior y se optimiza la pose usando BA. Cuando el seguimiento se pierde, el módulo de reconocimiento de lugar se utiliza para realizar una relocalización global. Una vez que ya se tiene una estimación inicial de la posición de la cámara y las coincidencias de los puntos de referencia, se recupera el mapa visible local utilizando el gráfico de covisibilidad de fotogramas claves. Después, las coincidencias con los puntos del mapa local se buscan por reproyección y la posición de la cámara se optimiza de nuevo con todas las coincidencias. El último paso del proceso de seguimiento es decidir si se inserta un nuevo fotograma clave.

- Mapeo:
  El mapeo local procesa nuevos fotogramas claves y realiza BA (optimización) local para lograr una reconstrucción óptima en el entorno de la pose de la cámara. Los nuevas características ORB sin par en el nuevo fotograma clave, se buscan en fotogramas claves conectadas por el gráfico de covisibilidad para triangular nuevos puntos de referencia. Después, se aplica una política de eliminación de puntos de referencia exigente para tener sólo puntos de alta calidad. Este proceso también se encarga de eliminar fotogramas claves redundantes.

- Cierre de ciclos:
  Busca ciclos (es decir, lugares que ya se visitaron previamente) en cada fotograma clave nuevo. Si se detecta un nuevo ciclo, se calcula una transformación de similitud que informa sobre la deriva de escala acumulada en el ciclo. Entonces, ambos lados del ciclo están alineados y los puntos de referencia duplicados están fusionados. Para finalizar el proceso, se realiza la optimización del grafo de pose sobre las restricciones de similitud para lograr la consistencia global. La optimización se realiza sobre el Grafo Esencial. La información de covisibilidad entre fotogramas claves se representa con un grafo ponderado no dirigido, donde cada nodo es un fotograma clave y si existe una arista entre fotogramas claves es porque comparten observaciones de los mismos puntos de referencia del mapa (al menos 15). El Grafo Esencial es un grafo que conserva todos los nodos pero no las aristas. Para realizar todas las optimizaciones se utiliza el algoritmo de Levenberg-Marquardt.

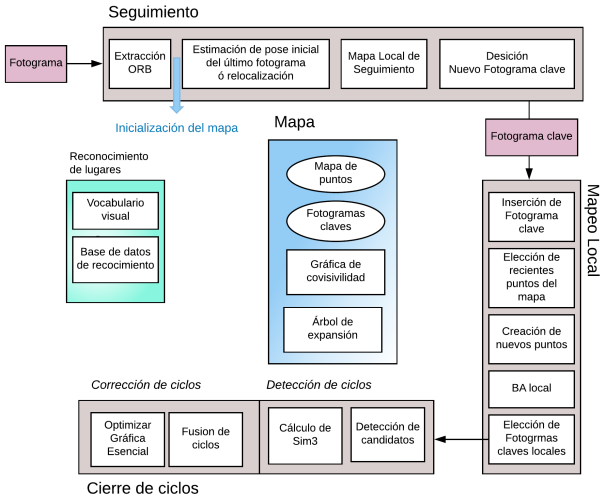


Figura 2.11: Funcionamiento general del sistema ORB-SLAM. Se muestran todos los pasos realizados por los procesos de seguimiento, mapeo y cierre de ciclos. Se muestran también los principales componentes de los módulos de reconocimientos de lugares y el mapa (Mur-Artal et al., 2015).

Algunas características importantes que se deben destacar de este sistema son:

- Bolsa de palabras para reconocimiento de palabras:
  El sistema utiliza una bolsa de palabras en el módulo de reconocimiento de lugares basado en DBoW2[6] para realizar la detección de ciclos y relocalización. Cabe mencionar que las

[6]DBoW2 es una versión mejorada de la biblioteca DBow, una biblioteca C++ de código abierto para indexar y convertir imágenes en una representación de bolsa de palabra.

palabras visuales son sólo una discretización del espacio de descriptores, conocida como vocabulario visual. El sistema crea incrementalmente una base de datos que contiene un índice invertido, que almacena para cada palabra visual en el vocabulario en qué fotograma clave se ha visto. Esta base de datos se actualiza cuando se elimina un fotograma clave mediante el procedimiento de eliminación selectiva.

- Autoinicialización:
  El objetivo es calcular la posición relativa entre dos cuadros para triangular un conjunto inicial de puntos del mapa. El método no debe tener intervención humana y debe de elegir una configuración con paralaje significativo. Se calculan en paralelo dos modelos geométricos, por un lado, una homografía que asume una escena plana, por otro lado, una matriz fundamental que asume una escena no plana. Después se usa una heurística para seleccionar un modelo entre los dos e intentar recuperar la postura relativa con un método específico para el modelo seleccionado. Este método sólo se inicializa cuando es seguro que la configuración de dos vistas es segura, detectando casos de paralaje bajo y la ambigüedad planar doble(Longuet-Higgins, 1986), evitando así inicializar un mapa incorrecto.

En la Figura 2.12, mostramos un ejemplo de la ejecución del sistema ORB-SLAM en un entorno interior.

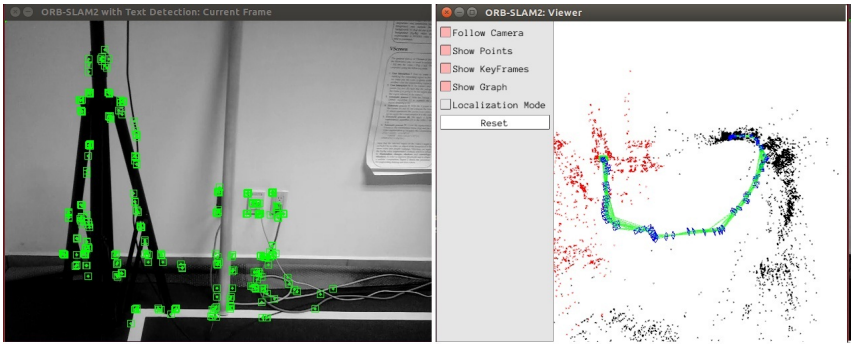


Figura 2.12: Ejemplo del funcionamiento de ORB-SLAM en la construcción de un escenario interior. En la imagen de la izquierda se presentan en verde los puntos de referencia claves en una imagen del escenario. En la imagen de la derecha se presenta la recontrucción del mapa $3D$ en donde se observa: de negro los puntos de referencia claves generados por diversos fotogramas, de rojo los últimos puntos de referencia claves generados que pueden utilizarse en la relocalización, en verde las posiciones de la cámara en el escenario, y en cuadros azules las posiciones de la cámara en cada fotograma clave.

Capítulo 3

Estimación de la Escala Global de Reconstrucción Monocular con Inferencia Bayesiana

3.1. Introducción

El algoritmo SLAM permite reconstruir una escena mediante una nube de puntos 3D, la cual denotaremos por $\mathcal{N} = \{P_i\}_{1 \leq i \leq n}$, donde $P_i = (x_i, y_i, z_i)^T$ es un punto 3D. Es la parte correspondiente al mapeo. Este algoritmo permite rastrear la localización de la cámara χ^k, donde k es un índice del tiempo y χ es una representación de la configuración 3D de la cámara que corresponde a la localización. Esa representación pertenece al grupo Euclidiano Especial $SE(3)$.

Sin embargo, este algoritmo, como muchos otros de visión y como se ha mencionado en el capítulo anterior, reconstruye la escena hasta cierto factor de escala, es decir, los verdaderos puntos 3D, $\mathbf{Q}_i$, están relacionados a P_i por:

$$\mathbf{Q}_i = \lambda P_i + \nu \tag{3.1.1}$$

donde λ es un factor de escala global a toda la escena, a priori **desconocido** y ν es un ruido de reconstrucción. Además, en los algoritmos de SLAM monocular, el factor de escala λ varía con el tiempo. La deriva de la escala, implica que, en la realidad, la relación de la ecuación 3.1.1 se modifica con el tiempo, siendo:

$$\mathbf{Q}_i = \lambda^k \mathbf{P}_i + \nu \tag{3.1.2}$$

donde λ^k es un factor de corrección de escala en el tiempo k.

El objetivo de este trabajo de tesis es precisamente inferir el valor de λ^k. Para realizar esto se propone utilizar un modelo Bayesiano que considere la información de observaciones de texto detectadas en imágenes y que tienen cierta correspondencia con algunos de los puntos detectados por el algoritmo SLAM.

3.2. Planteamiento general

En esta tesis se propone aplicar un enfoque Bayesiano para estimar en cada tiempo k el factor de escala λ^k. Es decir, utilizando la función "argumento máximo", definida como:

$$\arg\max_x(f(x)) := \{x|\forall y : f(y) \leqslant f(x)\}$$

obtenemos la estimación del mejor factor de escala $\hat{\lambda}^k$ que cumple:

$$\hat{\lambda}^k = \arg\max_{\lambda^k}(p(\lambda^k|\mathcal{N}, \mathcal{D}^1, \ldots, \mathcal{D}^k)), \tag{3.2.1}$$

donde:

- $\mathcal{N}$: Es la nube de puntos 3D.
- $\mathcal{D}^k = \{\mathcal{S}_l^k\}_{1 \leq l \leq \mathcal{M}^k}$: Es el conjunto de las $\mathcal{M}^k$ zonas de texto detectadas en el tiempo k. Por simplicidad supondremos que sólo hubo una detección, es decir $\mathcal{M}^k = 1$.

Para aproximar la distribución a posteriori de 3.2.1 se aplica la "regla de la cadena"(Castillo et al., 1997), considerando que las zonas de texto $\mathcal{D}^1, \ldots, \mathcal{D}^k$ están ordenadas de acuerdo a su tiempo de detección:

$$p(\lambda^k \mid \mathcal{N}, \mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^k) \propto p(\mathcal{D}^k \mid \lambda^k, \mathcal{N}, \mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^{k-1}) p(\lambda^k \mid \mathcal{N}, \mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^{k-1}). \tag{3.2.2}$$

Tomando en cuenta la información a priori sobre el tamaño del texto $H_m^k \in \mathcal{H}$ de cada zona de texto $\mathcal{D}^k$ (donde $\mathcal{H}$ es el conjunto discreto de posibles alturas para cada zona de texto $\mathcal{D}^k$), el primer término de 3.2.2 se puede escribir de la siguiente manera:

$$p(\mathcal{D}^k \mid \lambda^k, \mathcal{N}, \mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^{k-1}) \propto \sum_{H_m^k} p(H_m^k, \mathcal{D}^k \mid \lambda^k, \mathcal{N}, \mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^{k-1}). \tag{3.2.3}$$

Como $\mathcal{H}$ y $\mathcal{D}^k$ son conjuntos disjuntos entonces entonces la probabilidad conjunta de 3.2.3 se puede reescribir como (Castillo et al., 1997):

$$p(H_m^k, \mathcal{D}^k \mid \lambda^k, \mathcal{N}, \mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^{k-1}) \propto p(\mathcal{D}^k \mid H_m^k, \lambda^k, \mathcal{N}, \mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^{k-1}) \\ p(H_m^k \mid \lambda^k, \mathcal{N}, \mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^{k-1}). \tag{3.2.4}$$

Además, H_m^k es una variable independiente de $\lambda^k, \mathcal{N}, \mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^{k-1}$; y $\mathcal{D}^k$ es independiente de las $\mathcal{D}^{k-1}$ zonas de texto detectadas previamente, entonces 3.2.4 se puede expresar como:

$$p(H_m^k, \mathcal{D}^k \mid \lambda^k, \mathcal{N}, \mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^{k-1}) \propto p(\mathcal{D}^k \mid H_m^k, \lambda^k, \mathcal{N}) p(H_m^k), \tag{3.2.5}$$

donde:

- $p(H_m^k)$: Es la probabilidad a priori que un texto tenga tamaño H_m^k. Para su aproximación se utiliza una distribución normal $p(H)$ de los posibles tamaños de texto, la cual contempla el tamaño con el que ese texto usualmente aparece en un escenario (por ejemplo, la palabra "SALIDA"tiene a priori más probabilidad de aparecer con grandes caracteres, que la palabra "CONTRATO"que en general aparece con caracteres pequeños).
- $p(\mathcal{D}^k | H_m^k, \lambda^k, \mathcal{N})$: Es el término de verosimilitud. Es la probabilidad de que una detección de texto corresponda a la zona $\mathcal{D}^k$, suponiendo que el texto es de tamaño H_m^k y la escala es λ^k.

Por otra parte, para aproximar $p(\lambda^k \mid \mathcal{N}, \mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^{k-1})$ se contempla la estimación realizada sobre un conjunto discreto de factores λ_i^{k-1} en el tiempo $k-1$ de la siguiente manera:

$$p(\lambda^k \mid \mathcal{N}, \mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^{k-1}) \propto \sum_{\lambda^{k-1}} p(\lambda^k | \lambda^{k-1}) p(\lambda^{k-1} | \mathcal{N}, \mathcal{D}^1, \ldots, \mathcal{D}^{k-1}). \tag{3.2.6}$$

donde:

- $p(\lambda^k | \lambda^{k-1})$: Es el término de propagación, el cual evalúa las variaciones posibles de λ^k en el tiempo.
- $p(\lambda^{k-1} | \mathcal{N}, \mathcal{D}^1, \ldots, \mathcal{D}^{k-1})$: Es la probabilidad del factor λ en el tiempo $k-1$. Cabe mencionar que en el tiempo $k = 0$, los factores de profundidad del conjunto $\mathcal{F}$ tendrán las mismas probabilidades de ser el mejor candidato. Por lo tanto, la probabilidad será uniforme, es decir, $P(\lambda_i^0) = \dfrac{1}{|\mathcal{F}|}$.

Finalmente, sustituyendo 3.2.5 y 3.2.6 en 3.2.2, la probabilidad a posteriori para λ^k está dada por:

$$p(\lambda^k \mid \mathcal{N}, \mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^k) \propto \left(\sum_{\lambda^{k-1}} p(\lambda^k|\lambda^{k-1}) p(\lambda^{k-1}|\mathcal{N}, \mathcal{D}^1, \ldots, \mathcal{D}^{k-1}) \right) \left(\sum_{H^k} p(\mathcal{D}^k \mid H^k, \lambda^k, \mathcal{N}) p(H^k) \right). \tag{3.2.7}$$

Eso significa que en cada paso k, se actualizará la distribución a posteriori para λ^k.

3.3. Término de propagación

Este término está dado por

$$p(\lambda^k|\lambda^{k-1}) \tag{3.3.1}$$

el cual para su aproximación se considerará que sigue una distribución Normal:

$$f(\lambda^k) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{\frac{-(\lambda^k - \lambda^{k-1})^2}{2\sigma_c^2}},$$

donde la desviación estándar σ_c, dependerá de la velocidad angular de la cámara (Sucar and Hayet, 2017) y la media está dada por el posible valor de λ en el tiempo $k-1$.

3.4. Término de verosimilitud

El término de verosimilitud $p(D^k|H_m^k, \lambda^k, \mathcal{N})$, se refiere a la probabilidad de que un punto $P_i \in \mathcal{N}$ esté sobre un plano que contiene a la zona de texto D^k, para una altura posible de texto H_m^k y un factor de escala λ^k. Para la aproximación de este término, se utiliza la distancia algebraica entre el punto $P_i \in \mathcal{N}$ y el plano que contiene a la zona de texto D^k, que llamaremos $\pi(H_m^k)$ (donde H_m^k es la altura potencial del texto). El plano $\pi(H_m^k)$) y su relación con el punto P_i de la nube se muestran en la Figura 3.1.

Debido a que $\pi(H_m^k)$ se encuentra en el marco del texto, para calcular correctamente la distancia $d(\mathbf{p}, \pi(H_m^k))$ es necesario trasladar este plano al marco de la cámara.

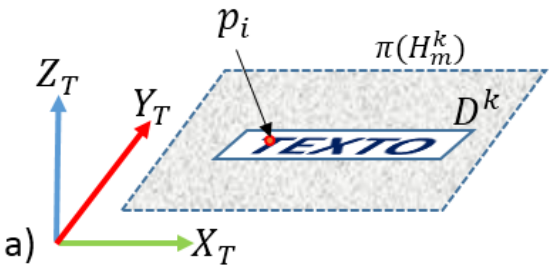


Figura 3.1: El plano $\pi(H_m^k)$ contiene a la zona D^k de texto en el marco del texto y el punto $P_i \in \mathcal{N}$ que está sobre el plano.

Ahora, en el marco de la cámara, eso se traduce por:

$$(\mathbf{R}^T\mathbf{p} - \mathbf{R}^T\mathbf{t})_z = 0 \tag{3.4.1}$$

donde $\mathbf{R} = [\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_3]$ y $\mathbf{t}$ son parámetros de transformación del marco del texto al marco de la cámara. Ya que $z = 0$ la ecuación anterior es equivalente a:

$$\mathbf{r}_3^T(\mathbf{p} - \mathbf{t}) = 0. \tag{3.4.2}$$

Esto se muestra en la Figura 3.2.

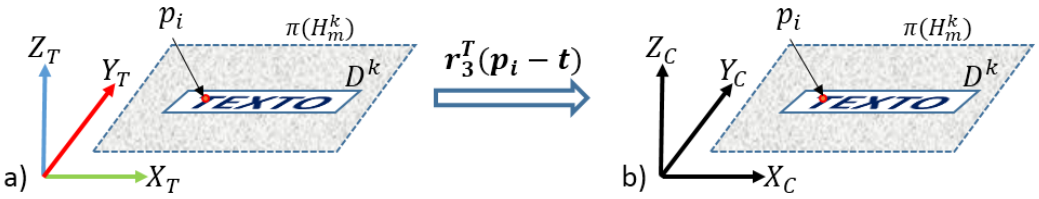


Figura 3.2: *a)* Se observa el plano $\pi(H_m^k)$ que contiene a la zona D^k de texto y el punto $P_i \in \mathcal{N}$ que está sobre este plano en el marco del texto. En *b)* se observa el plano $\pi(H_m^k)$ que contiene a la zona D^k de texto y el punto $P_i \in \mathcal{N}$ que está sobre este plano en el marco de la cámara.

La ecuación 3.4.2 considera que tanto el texto como el plano son regiones no deformadas o sin orientación. Sin embargo, dadas las circunstancias del escenario, el texto puede presentar alguna de las características ya mencionadas.

A fin de contemplar lo anterior, así como las dimensiones reales del texto en la escena, se calcula la homografía $\mathbf{H}$ entre la zona de texto detectada D^k y el "cuadrilátero verdadero"de la

escena que encierra a la zona D^k de texto (ver Apéndice A). La relación anterior se muestra en la Figura 3.3. En esta figura se puede observar que se supone que el "cuadrilátero verdadero"tiene ancho w y altura h (estas dimensiones son desconocidas). Esta homografía 3.3 satisface:

$$\mathbf{H} \propto \mathbf{K}[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}]\mathbf{S}$$

donde $\mathbf{K}$ es la matriz de parámetros intrínsecos de la cámara, $\mathbf{r}_1$ $\mathbf{r}_2$, $\mathbf{t}$ y $\mathbf{S}$ son los parámetros extrínsecos de rotación, traslación y escalamiento, respectivamente, de la transformación rígida cuadrilátero verdadero. $\mathbf{S}$ es una matriz dada por la ecuación:

$$\mathbf{S} = \begin{pmatrix} w & 0 & 0 \\ 0 & h & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Como $\mathbf{K}$ y $\mathbf{H}$ son conocidas, se puede obtener $\tilde{\mathbf{H}}$ como:

$$\tilde{\mathbf{H}} = \mathbf{K}^{-1}\mathbf{H}.$$

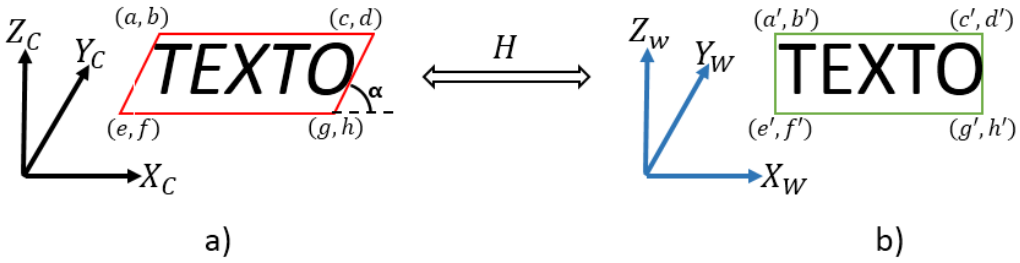


Figura 3.3: Homografía entre el cuadrilátero que encierra a la zona D^k de texto con un ángulo de inclinación α (a) y el cuadrilátero verdadero de la escena que encierra a la zona D^k de texto (b).

Además, si se considera que $\tilde{\mathbf{H}} = [\tilde{\mathbf{h}}_1, \tilde{\mathbf{h}}_2, \tilde{\mathbf{h}}_3]$ y que $\tilde{\mathbf{H}} = [\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}]\mathbf{S}$ con $[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}]\mathbf{S} = [w\mathbf{r}_1\ h\mathbf{r}_2\ \mathbf{t}]$, entonces se puede decir que:

$$\begin{cases} \tilde{\mathbf{h}}_1 = \lambda w \mathbf{r}_1 \\ \tilde{\mathbf{h}}_2 = \lambda h \mathbf{r}_2 \\ \tilde{\mathbf{h}}_3 = \lambda \mathbf{t} \end{cases} \tag{3.4.3}$$

donde w, h y λ son los valores que se desean calcular para encontrar las dimensiones reales del cuadrilátero que encierra el texto. $\lambda \neq 0$, corresponde al factor de escala que permite recuperar

las dimensiones correctas del texto.

Ya que $\tilde{\mathbf{h}}_1$, $\tilde{\mathbf{h}}_2$ y $\tilde{\mathbf{h}}_3$ son conocidos, se puede deducir una estimación para w y h, a partir de la primera y segunda ecuación de 3.4.3 como:

$$\begin{aligned}
\tilde{\mathbf{h}}_1 &= \lambda w \mathbf{r}_1, \\
||\tilde{\mathbf{h}}_1|| &= ||\lambda w \mathbf{r}_1||, \\
||\tilde{\mathbf{h}}_1|| &= \lambda w ||\mathbf{r}_1||, \\
||\tilde{\mathbf{h}}_1|| &= \lambda w, \text{ ya que } ||\mathbf{r}_1|| = 1, \\
w = \frac{||\tilde{\mathbf{h}}_1||}{\lambda}, \mathbf{r}_1 &= \frac{\tilde{\mathbf{h}}_1}{||\tilde{\mathbf{h}}_1||}.
\end{aligned} \tag{3.4.4}$$

$$\begin{aligned}
\tilde{\mathbf{h}}_2 &= \lambda h \mathbf{r}_2, \\
||\tilde{\mathbf{h}}_2|| &= ||\lambda h \mathbf{r}_2||, \\
||\tilde{\mathbf{h}}_2|| &= \lambda h, \text{ ya que } ||\mathbf{r}_2|| = 1, \\
h = \frac{||\tilde{\mathbf{h}}_2||}{\lambda}, \mathbf{r}_2 &= \frac{\tilde{\mathbf{h}}_2}{||\tilde{\mathbf{h}}_2||}.
\end{aligned} \tag{3.4.5}$$

De la ecuación de 3.4.5, se tiene:

$$\lambda = \frac{1}{h}||\tilde{\mathbf{h}}_2||. \tag{3.4.6}$$

De la tercera ecuación de 3.4.3 se puede obtener $\mathbf{t}$:

$$\mathbf{t} = \frac{\tilde{\mathbf{h}}_3}{\lambda}. \tag{3.4.7}$$

Con las ecuaciones 3.4.4, 3.4.5, 3.4.6 y 3.4.7 se puede expresar la Ecuación 3.4.2 en términos de la homografía calculada como:

$$(\tilde{\mathbf{h}}_1 \times \tilde{\mathbf{h}}_2)^T (\mathbf{p_i} - h \frac{1}{||\tilde{\mathbf{h}}_2||} \tilde{\mathbf{h}}_3) = 0. \tag{3.4.8}$$

De esa ecuación, se deduce una distancia algebraica de un punto $\mathbf{p_i}$, expresado en el marco cámara hacia el plano conteniendo el texto detectado $\pi(H_m^k)$:

$$d(\mathbf{p}, \pi(H_m^k)) = (\tilde{\mathbf{h}}_1 \times \tilde{\mathbf{h}}_2)^T (\mathbf{p_i} - H_m^k \frac{1}{||\tilde{\mathbf{h}}_2||} \tilde{\mathbf{h}}_3). \tag{3.4.9}$$

En la ecuación anterior, las coordenadas del punto p_i están expresadas en el marco cámara (hasta un factor de escala). Para expresarlas en función de sus coordenadas en el marco mundo (que nos proporciona el sistema mono SLAM), se utilizan los parámetros de transformación $\mathbf{R}^c, \mathbf{t}^c$ que se obtiene del algoritmo SLAM y el factor de escala λ^k como:

$$\tilde{\mathbf{p}}_i = \lambda^k (\mathbf{R}^c)^T (\mathbf{p}_i - \mathbf{t}^c).$$

Por lo que, finalmente, la verosimilitud queda definida como:

$$p(\mathcal{D}^k | H_m^k, \lambda^k, \mathcal{N}) \propto \exp -\frac{1}{2\sigma_v^2} d^2(\tilde{\mathbf{p}}_i, \pi(H_m^k)),$$

que puede ser expresada explícitamente como:

$$p(\mathcal{D}^k | H_m^k, \lambda^k, \mathcal{N}) \propto \exp -\frac{1}{2\sigma_v^2} \left\{ (\tilde{\mathbf{h}}_1 \times \tilde{\mathbf{h}}_2)^T (\lambda^k (\mathbf{R}^c)^T (\mathbf{p}_i - \mathbf{t}^c) - H_m^k \frac{1}{\|\tilde{\mathbf{h}}_2\|} \tilde{\mathbf{h}}_3) \right\}^2, \qquad (3.4.10)$$

donde σ_v se refiere a la dispersión del punto $\mathbf{p}_i$ del plano de la zona de texto $\mathcal{D}^k$.

Capítulo 4

Resultados

En este capítulo se presentan los resultados obtenidos sobre la estimación del factor de escala aplicando el modelo presentado en el Capítulo 3 y usando la información del texto encontrado en cuatro escenarios distintos.

Primero se describen las características del equipo así como el software utilizado en los experimentos y a continuación el conjuntos de datos. Después, se describe la configuración experimental; y finalmente, se presentan los resultados obtenidos con cuatro experimentos.

4.1. Hardware y Software utilizado

Para la implementación de los sistemas y la ejecución de pruebas se utilizó un cpu como estación de trabajo (*Workstation*) con las siguientes características:

- Procesador intel(R) Core (TM) $i7 - 3770$ de $3.4\ Ghz \times 8$.
- Sistema operativo Ubuntu 14.04 LTS de 64 bits.
- Memoria RAM de 16 GB.
- Tensorflow-gpu 1.1.0.
- Opencv 2.4.9.1, 2.9 y 3.2.
- Libreria *pylib*, *Pangolin*, *Eigen3*, *DBoW2* y *g2o*.

4.2. Datos

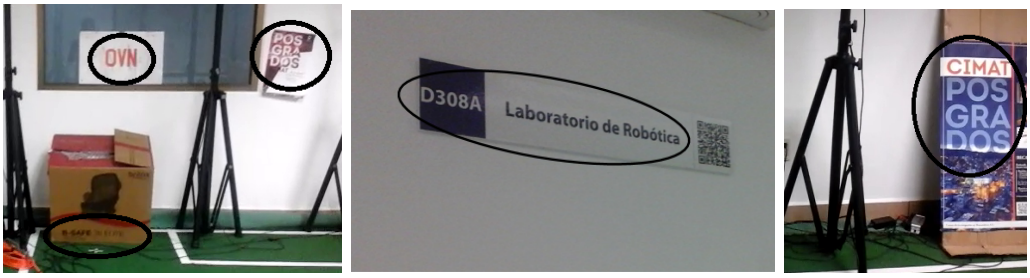
Para los experimentos se utilizaron cuatro videos correspondientes a cuatro escenarios diferentes. Los videos se capturaron en formato $mp4$, con una cámara de 3.15 MP, 2048×1536 pixeles y video de $720p@30fps$. A continuación se describe los escenarios capturados en cada video.

4.2.1. Laboratorio de Robótica del CIMAT, GTO

El primer video corresponde al recorrido de la cancha de futbol para humanoides del Laboratorio de Robótica del CIMAT, GTO. Esta área tiene dimensiones de 8×5 mts y se muestra en la Figura 4.1.

Figura 4.1: Cancha de futbol, Laboratorio de Robótica del CIMAT, GTO.

El video tiene una duración de $2:15$ min. y se utilizan 2159 *frames*. Los textos utilizados para esta primera prueba están contenidos en carteles, letreros en cajas, información del edificio, como se muestra en la Figura 4.2.



(a) Poster y caja (b) Información edificio (c) Poster

Figura 4.2: Tipos de texto encontrados en el Laboratorio de Robótica del CIMAT, GTO.

4.2.2. Laboratorio de Tecnología Avanzada de Manufactura, UTM

El segundo video corresponde al recorrido del Laboratorio de Tecnología Avanzada de Manufactura de la UTM, que tiene dimensiones de 23×17 mts. Como se muestra en la Figura 4.3, esta área cuenta con diferentes pasillos debido al gran tamaño de las máquinas utilizadas en el mismo.

Figura 4.3: Laboratorio de Tecnología Avanzada de Manufactura. Imagen obtenida de http:www.utm.mx/m_tec_avanzmanuf.html

El video tiene una duración de 3 : 42 min y se utilizan 6650 *frames*. Los textos utilizados para esta segunda prueba están contenidos en carteles, información de los equipos, señalización industrial, como se muestra en la Figura 4.4.

(a) Información de equipo 1 (b) Información de equipo 2 (c) Señalización Industrial

Figura 4.4: Tipos de texto encontrados en Laboratorio de Tecnología Avanzada de Manufactura, UTM.

4.2.3. Laboratorio de procesamiento de alimentos, UTM

El tercer video contiene el recorrido del Laboratorio de procesamiento de alimentos de la UTM. Esta área tiene dimensiones de 15×8 mts. Se muestra en la Figura 4.5.

Figura 4.5: Laboratorio de procesamiento de alimentos, UTM

El video tiene una duración de $2:41$ min y se utilizaron 4840 *frames*. Los textos utilizados para esta tercera prueba están contenidos en carteles, nombre y número de serie de los equipos, información del edificio, señalización industrial como se muestra en la Figura 4.6.

(a) Señalización Industrial (b) Información edificio (c) Poster

Figura 4.6: Diferentes textos en el escenario.

4.2.4. Laboratorio de Aplicaciones de Cómputo Distribuido Avanzadas, UTM

El cuarto video corresponde a la captura del recorrido del Laboratorio de Cómputo Distribuido Avanzadas de la UTM. Esta área tiene dimensiones de 3×6 mts y se muestra en la Figura 4.7.

Figura 4.7: Laboratorio de Aplicaciones de Cómputo Distribuido Avanzadas, UTM

El video tiene una duración de $1:56$ min. y se utilizaron 3495 *frames*. Los textos utilizados para esta cuarta prueba están contenidos en carteles y libros en diferentes orientaciones como se muestra en la Figura 4.8.

(a) Libros texto vertical (b) Poster

Figura 4.8: Tipos de texto encontrados en el Laboratorio de Aplicaciones de Cómputo Distribuido Avanzadas.

4.3. Configuración Experimental

4.3.1. Calibración de la cámara

La calibración de la cámara se realizó con la ayuda del paquete de *Toolbox_calib* de Matlab Versión 7. 10 (R2010a), y siguiendo el proceso que se describe en la Figura 4.9.

Los parámetros intrínsecos obtenidos de la calibración de la cámara fueron:

$$K = \begin{pmatrix} 1070.59943 & 0 & 342.61196 \\ 0 & 1071.55058 & 179.50469 \\ 0 & 0 & 0 \end{pmatrix}.$$

Se utilizó la misma cámara para los cuatro experimentos.

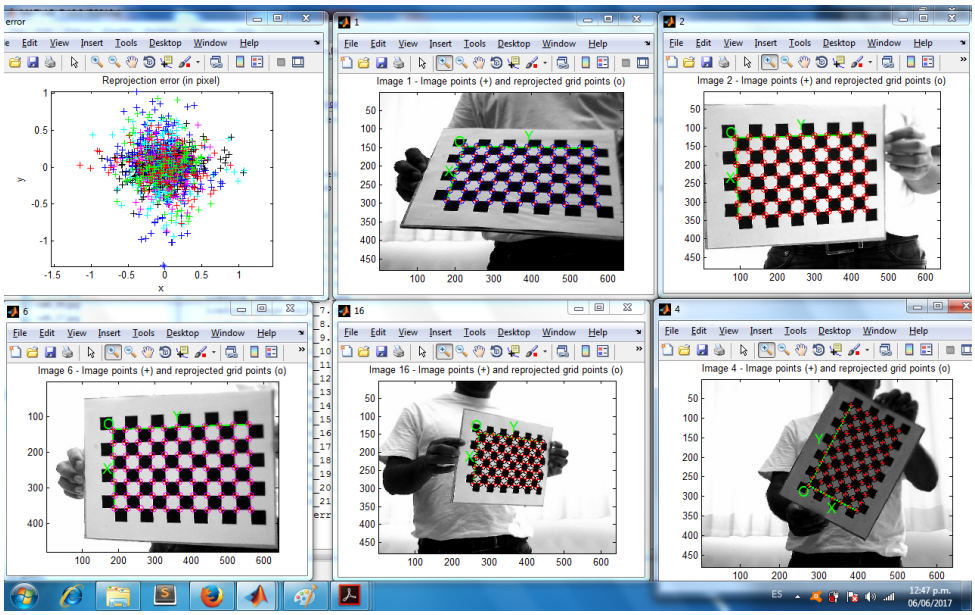


Figura 4.9: Proceso de calibración de la cámara.

4.3.2. Conjunto de posibles alturas para los textos y factores de escala

El conjunto de alturas $\mathcal{H}$ utilizado en las pruebas es el conjunto discreto:

$$\mathcal{H} = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18\}(\text{en cm.}).$$

Además, utilizando un histograma para analizar la ocurrencia de los tamaños de alturas, se puede observar un comportamiento normal. Por lo tanto, se asume que el conjunto de alturas $\mathcal{H}$ está normalmente distribuido con une media muestral de $\mu_h = 10$ y una desviación estándar $\sigma_h = 5.5$. Esta media fue seleccionada debido a que son las alturas más comunes que encontramos en nuestros escenarios de prueba.

La búsqueda de los factores de escala se realiza en el intervalo $[1, 132072]$. Se considera un incremento multiplicativo de $\sqrt{2}$, por lo que el conjunto de posibles factores de escala es:

$$\mathcal{F} = \{x_i | x_i = (\sqrt{2})(x_{i-1})\} \cup \{x_1 = 1\}, i = 2, \ldots, 35.$$

4.3.3. Término de verosimilitud

Para evaluar este término, de acuerdo a la Ecuación (3.4.10), es necesario tener un punto de la nube $\mathbf{p}_i^k \in \mathcal{N}$ que coincida con la zona de texto detectada en el proceso de reconstrucción, tal

y como se muestra en las figuras 4.10 y 4.11. Por esta razón, la construcción de este término se realiza con la primera coincidencia que encuentra el algoritmo, y con una desviación estándar de $\sigma_v = 0.01$.

(a) Imagen original (b) Zonas de texto (polígonos verdes)

Figura 4.10: Detección de las zonas de texto. En (a) se muestra la imagen original antes de ser procesada, y en (b), la detección de las zonas de texto en la imagen.

Figura 4.11: Coincidencia de un punto de la nube con una zona de texto detectada. Se muestran en rojo la zona de texto detectada y el punto de la nube que coincide.

En la Figura 4.12, se presenta el comportamiento de la verosimilitud para $\lambda = 8192$, con cada

posible altura H_m^k. Para este ejemplo, la zona de texto tiene coordenadas: $((288, 120), (288, 92), (351, 92), (351, 120))$ y el punto $\mathbf{p}_i^k$ de la nube es: $(0.203199, -0.011741327, 0.816426)$.

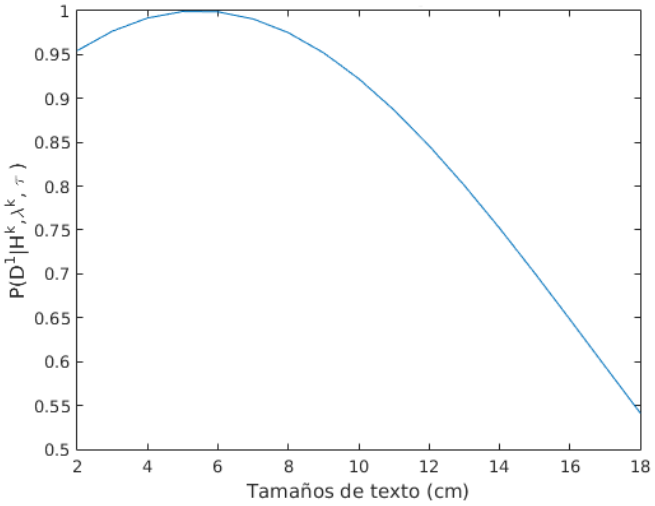


Figura 4.12: Gráfica de verosimilitud para $\lambda = 8192$.

El comportamiendo del término de verosimilitud 3.4.10 se ve afectado por los diferentes factores de escala λ^k. Al analizar el término de la potencia, la parte izquierda en la diferencia, se puede observar que para valores demasiado pequeños (con respecto al lado derecho de la diferencia) las gráficas de las Gaussianas se acumularán en una media alrededor de cero, mientras que cuando la diferencia tiene valores equiparables (entre ambos términos de la diferencia) las Gaussianas tendrán un dezplamiento sobre cada posible valor de altura. Finalmente, cuando estos valores (ambos términos de la diferencia) se superen, las gráficas saldrán del rango de alturas y se observará un tendencia a cero en el rango de alturas.

En la Figura 4.13, se presenta el comportamiento antes mencionado. En el intervalo $[2, 131072]$ el término de verosimilitud tiene la forma de una Gausiana centrada en diferentes valores de alturas. Estas gráficas se van desplazando de izquierda a derecha. Para los valores menores a $\lambda^k = 1800$ aproximadamente, los valores tienden a acumularse en una Gaussiana centrada en $H_m^k = 2$ (las gráficas son las de línea continuas), además, estos valores no muestran cambios. Para valores de λ^k entre $[1800, 46340]$ se puede observar como las gráficas se desplazan, mientras que para los valores más grandes de 50000 se observa una tendencia a cero. Este comportamiento de la verosimilitud permitió establecer el intervalo de búsqueda para los factores de escala. De acuerdo a la gráfica, valores mayores a 50000 no aportan información para la estimación.

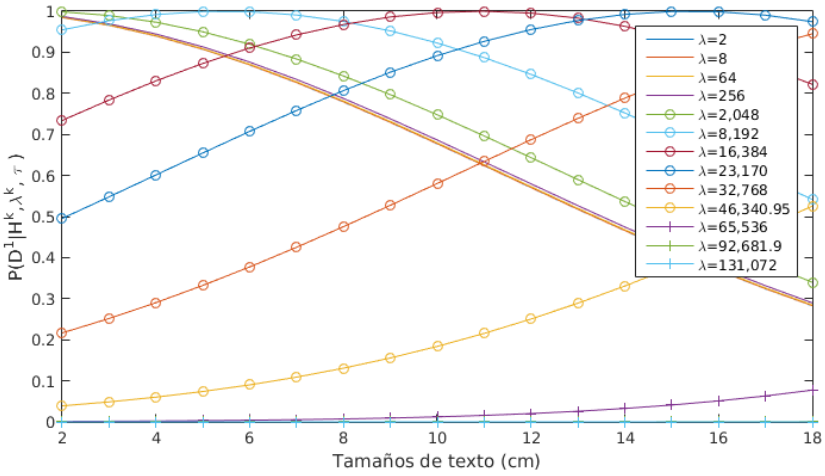


Figura 4.13: Gráfica de verosimilitud para diferentes factores de escala.

4.3.4. Término de propagación

Para el término de propagación (ver Ecuación 3.3.1) los parámetros son una desviación estándar de $\sigma_c = 2000$ y media muestral λ^{k-1}. La gráfica de este término para una iteración k se muestra en la Figura 4.14.

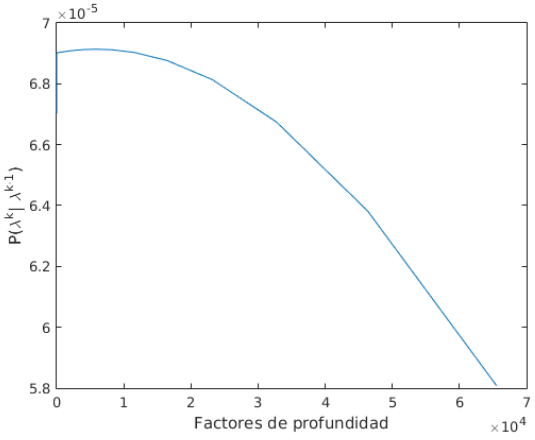


Figura 4.14: Gráfica de la probabilidad condicional $p(\lambda^k|\lambda^{k-1})$ de los factores λ^k dado λ^{k-1}.

4.4. Experimentos

En esta sección se presentan los resultados obtenidos al aplicar el modelo propuesto en el Tercer Capítulo a cuatro escenarios diferentes.

En los cuatro experimentos se utiliza la configuración experimental descrita en la sección anterior. En la estimación del factor de escala se aplica el siguiente proceso:

1. El sistema SegLink tiene como entrada el video del escenario a reconstruir, y una vez que detecta las zonas de texto en cada imágen del video, se guardan las posiciones que acotan a los cuadriláteros.

2. Se ejecuta el sistema ORB-SLAM agregando la información de la posición de las zonas de texto acotadas por un cuadrilátero. La salida son dos ventanas. En la primera ventana se muestra el video y los puntos de referencia ORB que se utilizan para el seguimiento y el mapeo. En la segunda ventana se ve el entorno $3D$ de reconstrucción y seguimiento. Los puntos de la nube cuya proyección corresponda a una zona de texto, son utilizados por el sistema de inferencia para estimar el factor de escala.

3. Para la estimación del factor de escala del escenario se realizan en promedio de 10 a 15 iteraciones por escenario. Las estimaciones se realizan cada 10 ó 15 segundos del video.

4.4.1. Primer experimento

El primer experimento se realizó con el video del Laboratorio de Robótica del CIMAT. El proceso de estimación del factor de escala se muestra en la Figura 4.15.

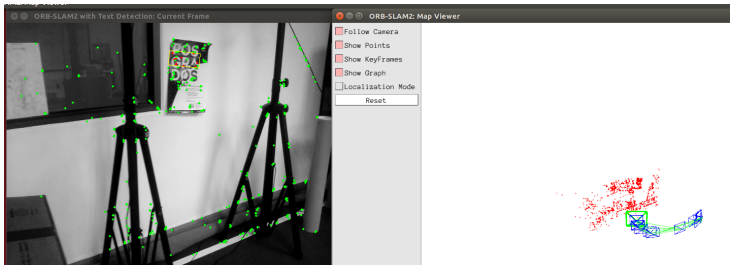


Figura 4.15: Ejemplo del funcionamiento de ORB-SLAM en la construcción de un escenario interior. En la imagen de la izquierda se presentan en verde los puntos claves de una imagen del escenario. En la imagen de la derecha se presenta la reconstrucción del mapa $3D$ en donde se observa: en rojo los últimos puntos claves generados que pueden utilizarse en la relocalización, en verde el recorrido de la cámara en el escenario, y en cuadros azules las posiciones de la cámara.

En las figuras 4.16, 4.17 y 4.18 se presenta el comportamiento del modelo en la estimación de λ^k en cada iteración k. Cabe mencionar que en cada iteración se utiliza una zona de texto diferente. El caso inicial es una recta de color azul (distribución uniforme). De acuerdo a estas gráficas se observa que las probabilidades para los mejores factores se concentran en el rectángulo $[0, 40000] \times [0.03, 0.05]$.

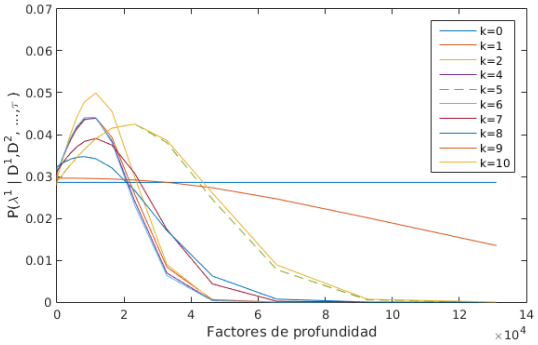


Figura 4.16: Gráfica de $p(\lambda^k \mid \tau, \mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^k)$ para cada factor λ^k en las primeras 10 iteraciones.

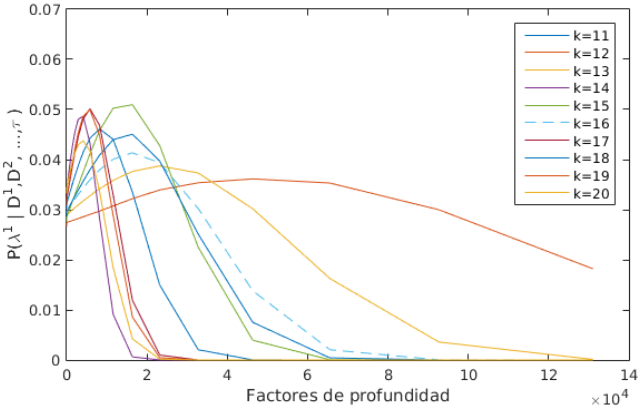


Figura 4.17: Gráfica de $p(\lambda^k \mid \tau, \mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^k)$ para cada factor λ^k de la iteración 11 a 20.

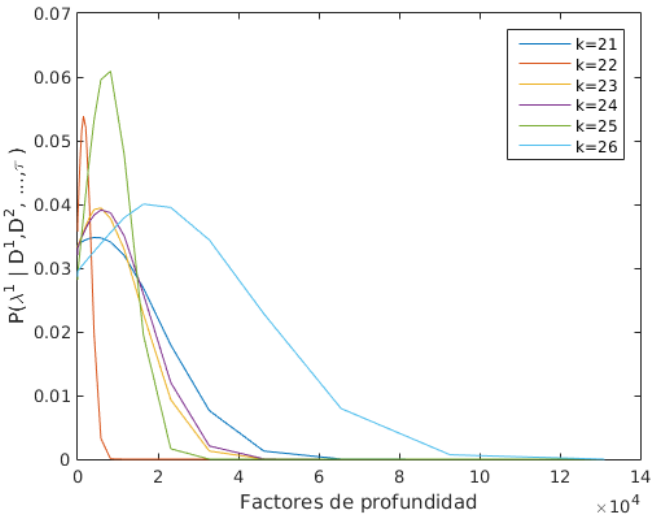


Figura 4.18: Gráfica de $p(\lambda^k \mid \tau, \mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^k)$ para cada factor λ^k de la iteración 21 a 26.

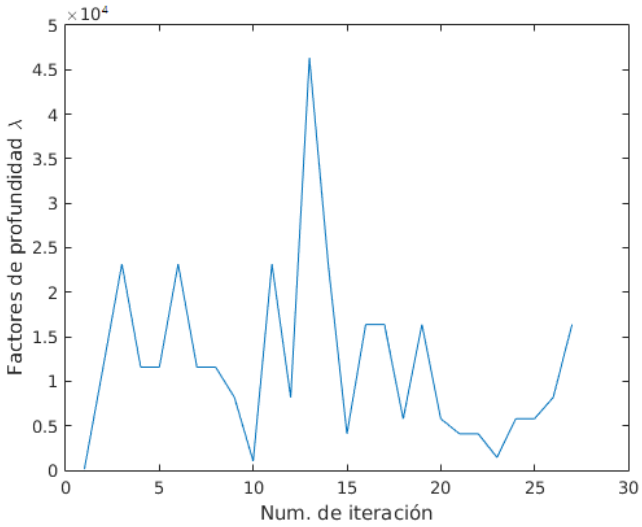


Figura 4.19: Gráfica que muestra el mejor factor de λ^k encontrado en cada iteración.

En la Figura 4.19 se muestra el mejor factor de escala λ^k (el más probable) encontrado en cada iteración. Cabe señalar que este factor varía con el tiempo (*scale drift*) por tanto no no se tendrá un comportamiento constante en el valor del factor en cada iteración. Como se

puede observar existen algunas variaciones o saltos entre los factores de escala, por ejemplo, en la iteración doce y trece. Esto es debido a que existe una diferencia entre los tamaños de las zonas de texto utilizadas por el modelo y los que realmente figuran en la escena, haciendo que el sistema se desestabilice. Para este experimento podemos notar que en promedio las diferencias entre los factores de escala (entre dos iteraciones consecutivas) es 8866.52 con una desviación estándar igual 8756.0445.

4.4.2. Segundo experimento

El segundo experimento se realizó con el video del Laboratorio de Tecnología Avanzada de Manufactura de la UTM. El proceso de estimación del factor de escala se muestra en la Figura 4.20.

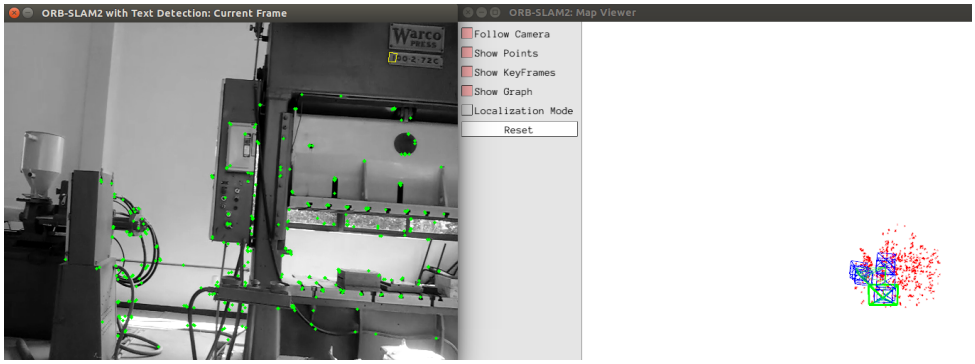


Figura 4.20: Ejemplo del funcionamiento de ORB-SLAM en la construcción de un escenario interior. En la imagen de la izquierda se presentan en verde los puntos claves de una imagen del escenario. En la imagen de la derecha se presenta la reconstrucción del mapa $3D$ en donde se observa: en rojo los últimos puntos claves generados que pueden utilizarse en la relocalización, en verde el recorrido de la cámara en el escenario, y en cuadros azules las posiciones de la cámara.

En la Figura 4.21 se presenta el comportamiento del modelo en la estimación de las distribuciones sobre λ^k en cada iteración k. En cada estimación se utilizan diferentes zonas de texto. El caso inicial ($k = 0$) es una recta de color azul (distribución uniforme). De acuerdo a estas gráficas se observa que los factores con mayor probabilidad se concentran en el rectángulo $[0, 35000] \times [0.035, 0.045]$. Globalmente, se puede observar que la varianza sobre la estimación de λ^k se va reduciendo con cada k. Lo que pareciera indicar que el sistema va recibiendo información coherente para realizar la estimación.

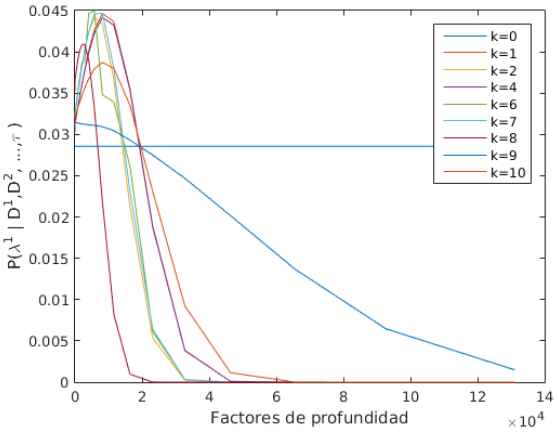


Figura 4.21: Gráfica de $p(\lambda^k \mid \tau, \mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^k)$ para cada factor λ^k en 10 iteraciones.

En la Figura 4.22 se muestra el mejor factor de escala λ^k (el más probable) encontrado en cada iteración. Como se puede observar existen algunas variaciones o saltos entre los factores de escala, por ejemplo, en la iteración tres y cuatro. Esto es debido a que existe una diferencia entre los tamaños de las zonas de texto utilizadas por el modelo, haciendo que el sistema se desestabilice. Para este experimento podemos notar que en promedio las diferencias entre los factores de escala (entre dos iteraciones consecutivas) es 4082.6343 con una desviación estándar igual 2670.15922.

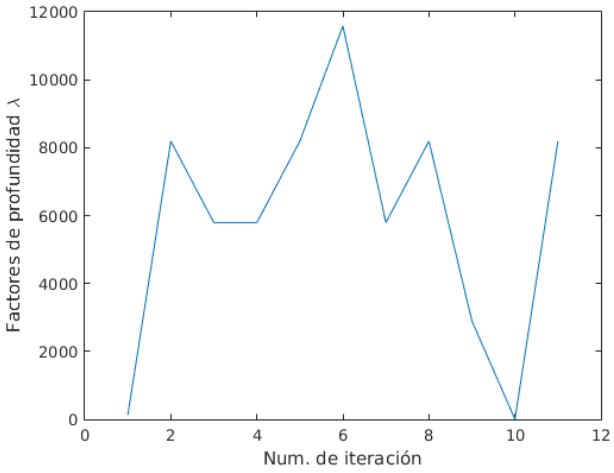


Figura 4.22: Gráfica donde se muestra el mejor factor de λ^k encontrado en cada iteración.

4.4.3. Tercer experimento

El tercer experimento se realizó con el video del Taller de Procesamiento de alimentos de la UTM. El proceso de estimación del factor de escala se muestra en la Figura 4.23.

Figura 4.23: Ejemplo del funcionamiento de ORB-SLAM en la construcción de un escenario interior. En la imagen de la izquierda se presentan en verde los puntos claves de una imagen del escenario. En la imagen de la derecha se presenta la reconstrucción del mapa $3D$ en donde se observa: en rojo los últimos puntos claves generados que pueden utilizarse en la relocalización, en verde el recorrido de la cámara en el escenario, y en cuadros azules las posiciones de la cámara.

En las figuras 4.24, 4.25 se presenta el comportamiento del modelo en la estimación de las distribuciones sobre λ^k en cada iteración k. En cada estimación se utilizan diferentes zonas de texto. El caso inicial ($k = 0$) es una recta de color azul (distribución uniforme). De acuerdo a estas gráficas se observa que los factores con mayor probabilidad se concentran en el rectángulo $[0, 30000] \times [0.035, 0.045]$.

En la Figura 4.26 se muestra el mejor factor de escala λ^k (el más probable) encontrado en cada iteración. Como se puede observar existen algunas variaciones o saltos entre los factores de escala, por ejemplo, en la iteración tres y cuatro. Esto es debido a que existe una diferencia entre los tamaños de las zonas de texto utilizadas por el modelo, haciendo que el sistema se desestabilice. Para este experimento podemos notar que en promedio las diferencias entre los factores de escala (entre dos iteraciones consecutivas) es 588711 con una desviación estándar igual 3612.23.

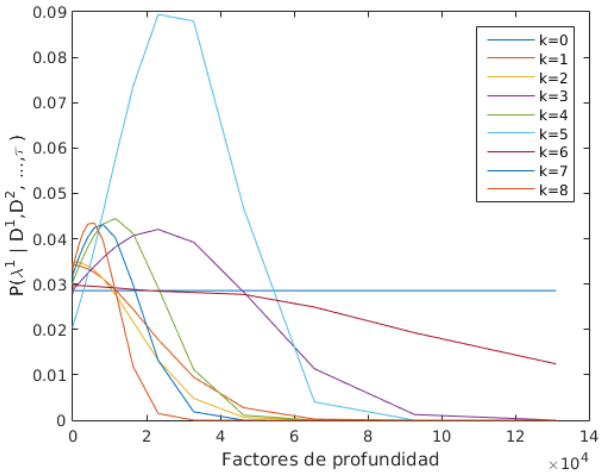


Figura 4.24: Gráfica de $p(\lambda^k \mid \tau, \mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^k)$ para cada factor λ^k en las primeras 7 iteraciones.

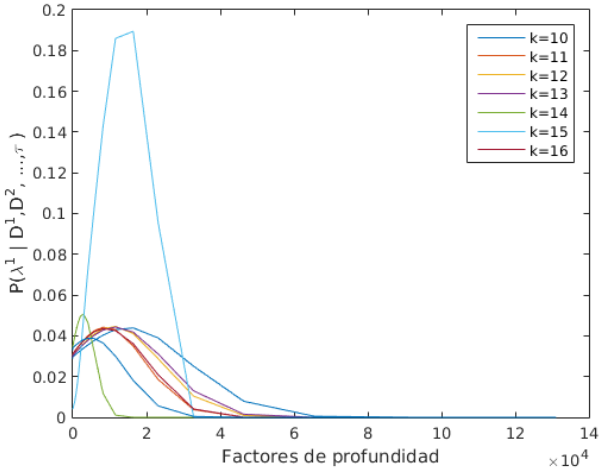


Figura 4.25: Gráfica de $p(\lambda^k \mid \tau, \mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^k)$ para cada factor λ^k de la iteración 8 a 13.

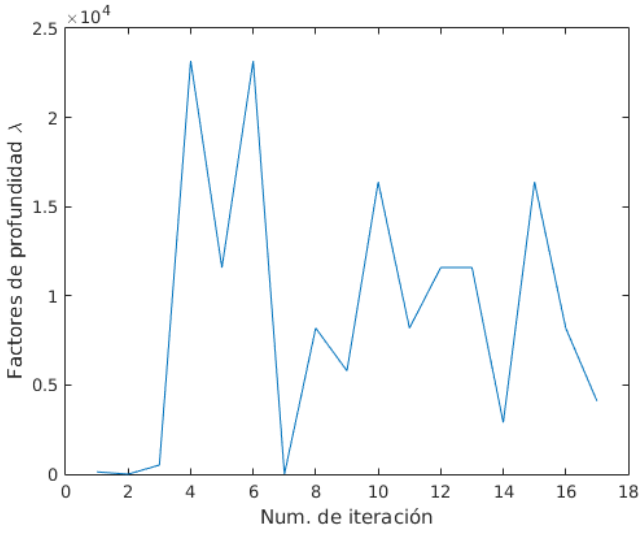


Figura 4.26: Gráfica donde se muestra el mejor factor de λ^k encontrado en cada iteración.

4.4.4. Cuarto experimento

El cuarto experimento se realizó con el video del Laboratorio de Aplicaciones de Cómputo Distribuido Avanzadas de la UTM. El proceso de estimación del factor de escala se muestra en la Figura 4.27.

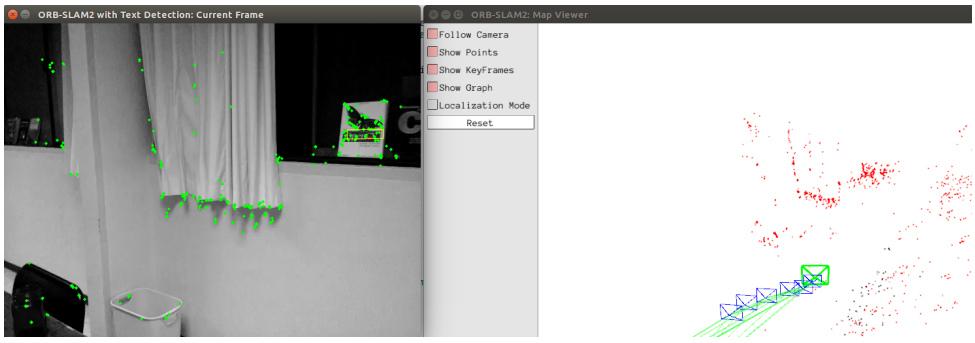


Figura 4.27: Ejemplo del funcionamiento de ORB-SLAM en la construcción de un escenario interior. En la imagen de la izquierda se presentan en verde los puntos claves de una imagen del escenario. En la imagen de la derecha se presenta la reconstrucción del mapa $3D$ en donde se observa: en rojo los últimos puntos claves generados que pueden utilizarse en la relocalización, en verde el recorrido de la cámara en el escenario, y en cuadros azules las posiciones de la cámara.

En las figuras 4.28, 4.29 y 4.30 se presenta el comportamiento del modelo en la estimación de las distribuciones sobre λ^k en cada iteración k. En cada estimación se utilizan diferentes zonas de texto. El caso incial ($k = 0$) es una recta de color azul (distribución uniforme). De acuerdo a estas gráficas se observa que los factores con mayor probabilidad se concentran en el rectángulo $[0, 35000] \times [0.035, 0.055]$.

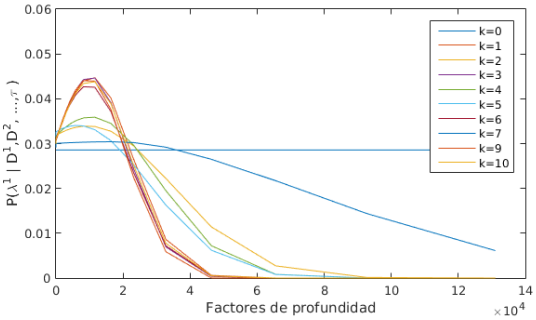


Figura 4.28: Gráfica de $p(\lambda^k \mid \tau, \mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^k)$ para cada factor λ^k de las primeras 10 iteraciones.

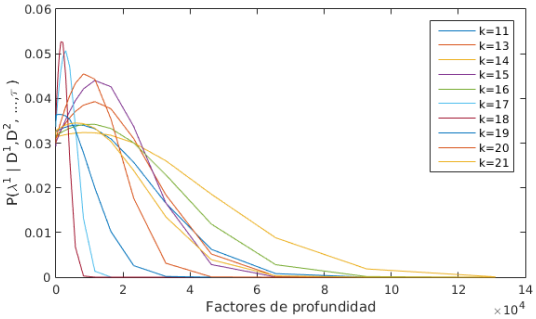


Figura 4.29: Gráfica de $p(\lambda^k \mid \tau, \mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^k)$ para cada factor λ^k de la iteración 11 a 21.

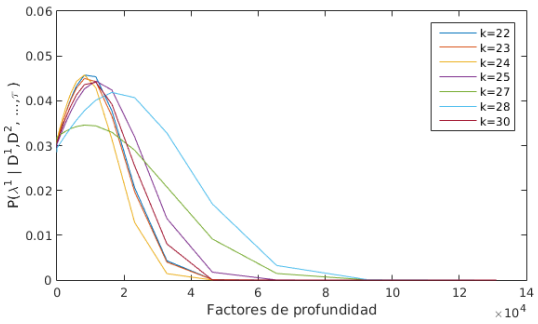


Figura 4.30: Gráfica de $p(\lambda^k \mid \tau, \mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^k)$ para cada factor λ^k de la iteración 22 a 30.

En la Figura 4.31 se muestra el mejor factor de escala λ^k (el más probable) encontrado en cada iteración. Como se puede observar existen algunas variaciones o saltos entre los factores de escala, por ejemplo, en la iteración cinco y nueve. Esto es debido a que existe una diferencia entre los tamaños de las zonas de texto utilizadas por el modelo, haciendo que el sistema se desestabilice. Para este experimento podemos notar que en promedio las diferencias entre los factores de escala (entre dos iteraciones consecutivas) es 9309.3531 con una desviación estándar igual 1126.1124.

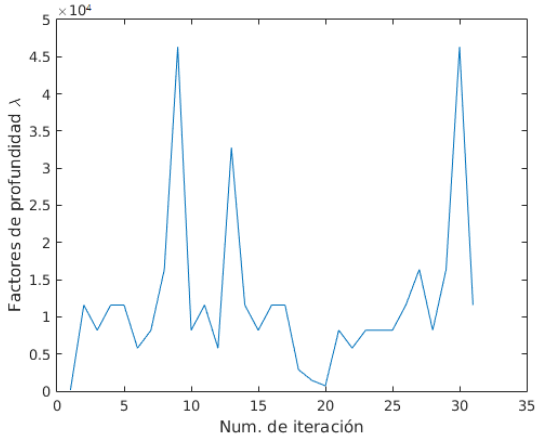


Figura 4.31: Gráfica donde se muestra el mejor factor de λ^k encontrado en cada iteración.

Capítulo 5

Conclusiones y trabajos a futuro

En este trabajo se desarrolló un sistema de inferencia para contrarrestar el efecto de la variación de la escala cuando se reconstruye un escenario con el método de localización y mapeo visual ORB-SLAM. El factor de escala se corrigió utilizando la información relativa de objetos semánticos fáciles de detectar en un escenario, es decir, los textos. Se consideró información a priori, sobre su tamaño, de estos elementos semánticos. La detección de las zonas de texto en las imágenes del escenario se realizó mediante un sistema de detección de texto en imágenes naturales conocido como SegLink.

Con el término de verosimilitud propuesto en el modelo, se determinó el intervalo en donde los factores de profundidad aportan información valiosa, siendo $[1, 132072]$ el intervalo seleccionado con $\sigma_v = 2000$. Mientras que el intervalo para las alturas de los textos fue $[2, 18]$ con una distribución Gaussiana y $\sigma_h = 5.5$.

Como resultado obtuvimos la estimación de factores dentro de un rango estable, las gráficas de la estimación del mejor factor muestran dos comportamientos. El primero es la estabilidad de los factores dentro de un rango a lo largo del video. El segundo, es que se observan brincos entre los factores debido a la variación de los tamaños de texto utilizados en por el modelo en cada estimación. En resumen, se logra evitar la deriva de escala (Sucar and Hayet, 2017) manteniendo el factor en un rango que depende de cada escenario.

Los experimentos se realizaron en cuatro diferentes escenarios se contrarrestó el efecto de la variación de la escala, restringiendola a un intervalo de $[0, 50000]$ lo cual permite concluir que el modelo acota en un intervalo al factor de escala, evitando el desborde.

Algunas de las posibles líneas de investigación que ayudarán a complementar el trabajo

aquí desarrollado, son las siguientes:

- Los experimentos se realizaron en escenarios interiores, por tanto, se tuvo un control del tamaño de los textos que se podían encontrar. Lo cual sugiere desarrollar una herramienta que permita proponer intervalos para las alturas de los textos, según el escenario que se requiere recontruir.
- La CNN utilizada para el reconocimiento de texto inicialmente fue entrenada para el reconocimiento de objetos. Se pretende reentrenar la red con una base de datos de textos en diferentes escenarios para mejorar los resultados de la detección.
- Inferir el factor de escala a partir de la información de la nube de puntos proporcionada por otro sistema mono SLAM.
- Integrar el resultado obtenido por el sistema de inferencia Bayesiano al sistema mono SLAM para mejorar la reconstrucción del escenario.
- Reconstruir objetos de los cuales se conoce su información dimensional a partir del resultado del sistema de inferencia Bayesiano para determinar la exactitud del modelo.
- Hacer más robusto el sistema (rechazo de *outliers*).

Apéndice A

Homografía

Para definir lo que es una homografía se consideran dos imágenes, ambas imágenes son del mismo objeto (la imagen de la portada de un libro, lo podemos considerar como un plano), pero visto de diferentes ángulos, como se muestra en la Figura A.1. El punto rojo representa el mismo punto físico en ambas imágenes (este proceso es conocido como correspondencia de puntos). En las imágenes se pueden observar cuatro puntos de color: rojo, verde, amarillo y naranja.

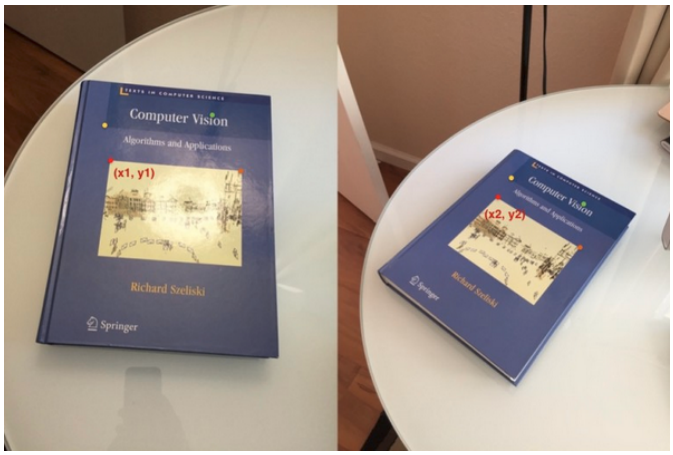


Figura A.1: Imágenes de la portada de un libro vista de dos ángulos diferentes se pueden relacionar por una homografía.

Entonces, se puede decir que una homografía es una trasformación (una matriz de 3×3) que mapea los puntos de una imagen a los puntos correspondientes en otra imagen (hasta cierto factor de escala) (Mallick, 2016).

Esta transformación puede ser expresada en términos de producto matricial si se usan coor-

denas homogeneas para expresar los puntos que correspondientes. Es decir, si se consideran los puntos rojos, Q y q, de la imagen izquierda y derecha, respectivamente, definidos:

$$\tilde{Q} = [X, Y, Z, 1]^T$$

$$\tilde{q} = [x, y, z, 1]^T$$

entonces la acción de homografía se puede expresar como (Bradski and Kaehler, 2008):

$$\tilde{q} = sH\tilde{Q}, \tag{A.0.1}$$

donde s es un factor de escalamiento y $\mathbf{H}$ la homografía que consta de dos partes, la primera es la transformación física, la cual, localiza al plano del objeto que se observa. La segunda, es la transformación de proyección, la cual introduce los parámetros intrínsecos de la cámara.

La trasformación física es la parte que suma los efectos de alguna rotación $\mathbf{R}$ y una traslación $\mathbf{t}$ que relaciona al plano que se observa al plano de la imagen. En forma matricial esto es:

$$W = [R|t].$$

es una matriz de tamaño 3×4, las tres primeras columnas corresponden a las 9 entradas de la matriz de rotación $\mathbf{R}$ y la última columna corresponde al vector de traslación t. Mientras que la trasformación de proyección corresponde a la matriz de los parámetros intrínsecos de la cámara, entonces (A.0.1) se puede expresar como (Bradski and Kaehler, 2008):

$$\tilde{q} = sKW\tilde{Q}, \tag{A.0.2}$$

donde

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

Cuando se desea definir el plano del objeto, $Z = 0$, como la matriz de rotación se divide en 3 columnas ($\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3]$), una de estas columnas no es necesaria y (A.0.2) se reeescribe de la siguiente manera:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = sK[r_1, r_2, r_3, t] \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = sK[r_1, r_2, t] \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

Por lo tanto, la matriz de homografía H que mapea un punto del plano del objeto a una imagen está descrito completamente por:

$$H = K[r_1, r_2, t].$$

Apéndice B

Pseudocódigo de algoritmos utilizados

B.1. Pseudocódigo para calcular el término de verosimilitud

Algorithm 1 Pseudocódigo de Verosimilitud

procedure TERM_VEROSIM ▷ Pseudocódigo para calcular el término de verosimilitud
    Obtenemos la matriz R_{cw} y t_{cw} de rotación y traslación
    Trasladamos el punto de la nube P_w al marco cámara como P_c
    Calculamos la matriz inversa K^{-1} de los parámetros intrínsecos
    Calcumos la homografía H, entre el cuadrilátero que encierra el texto y el cuadro verdadero
    Calculamos el producto entre $\tilde{H} = K^{-1}H$
    Descomponemos la matriz $\tilde{H}$ en sus vectores columna $\tilde{h_1}$,$\tilde{h_2}$ y $\tilde{h_3}$
    for H_m^k del vector de alturas posbibles **do**
        $\exp -\frac{1}{2\sigma_v^2}\{(\tilde{\mathbf{h}}_1 \times \tilde{\mathbf{h}}_2)^T(\lambda^k P_c) - H_m^k \frac{1}{\|\tilde{\mathbf{h}}_2\|}\tilde{\mathbf{h}}_3)\}^2$
    end for
    return Vector que contiene la verosimilitud para cada H_m^k
end procedure

B.2. Algoritmo para calcular el Término de Propagación

Algorithm 2 Pseudocódigo de Propagación

procedure TERM_PROP ▷ Pseudocódigo para calcular el término de propagación
    Proponemos una dispersión de las datos σ_c
    for λ^k del vector de factores de profudidad **do**
        for λ^{k-1} **do**
            Calcular $p(\lambda^k|\lambda^{k-1})$
        end for
    end for
    return Matriz que contiene la probabilidad condicional λ^k y λ^{k-1}
end procedure

B.3. Algoritmo del Modelo de Inferencia

Algorithm 3 Pseudocódigo del Modelo de Inferencia

procedure TERM_PROP ▷ Pseudocódigo para estimar el factor de profundidad
    Proponemos un caso base para los factores de produndidad CB
    for λ^k del vector de factores de profudidad **do**
        Lo envíamos a 2
        if Es la primera ves que estimamos **then**
            Enviamos λ^k a (2)
            Multiplicamos el término de propagación por CB
            Calculamos (1) multiplicado por cada $p(H_m)$.
            Guardamos el producto de (7) y (8)
        end if
        Enviamos λ^k a (2)
        Multiplicamos el término de propagación por el vector de estimación según la iteración correspondiente
        Calculamos (1) multiplicado por cada $p(H_m)$.
        Guardamos el producto de (12) y (13)
    end for
    return Retornamos en el mejor factor λ^k en cada iteración
end procedure

Bibliografía

Acevedo Orduña, G. L., Caicedo Bravo, E. F., and Loaiza Correa, H. (2011). Selección de personal mediante redes neuronales artificiales. *Revista de Matemática: Teoría y Aplicaciones*, 14(1):7–20.

Bourmaud, G. and Megret, R. (2015). Robust large scale monocular visual SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1638–1647.

Bowen, A. M., Telemática, I., and Asensio, H. G. (2011). Inteligencia artificial. redes neuronales y aplicaciones. *Revista de Matemática: Teoría y Aplicaciones*, 14(1):8.

Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. .ºReilly Media, Inc.".

Castillo, E., Gutiérrez, J. M., and Hadi, A. S. (1997). Sistemas expertos y modelos de redes probabilısticas. *Academia de Ingenierıa*.

Chen, D., Odobez, J.-M., and Bourlard, H. (2004). Text detection and recognition in images and video frames. *Pattern recognition*, 37(3):595–608.

Chuliá, L. C. (2015-2016). Reconocimiento de emociones mediante técnicas de aprendizaje profundo. Master's thesis, Universidad Politécnica de Valencia.

Civera, J., Grasa, O. G., Davison, A. J., and Montiel, J. (2009). 1-point ransac for EKF-based structure from motion. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3498–3504. IEEE.

Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1403–1410. IEEE.

Engel, J., Schöps, T., and Cremers, D. (2014). LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision*, pages 834–849. Springer.

Epshtein, B., Ofek, E., and Wexler, Y. (2010). Detecting text in natural scenes with stroke width transform. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2963–2970. IEEE.

Francisco, M.-A. R. and Antoni, G.-S. (2013). SLAM con mediciones angulares: método por triangulación estocástica. *Ingeniería, investigación y tecnología*, 14(2):257–274.

Frost, D. P., Kähler, O., and Murray, D. W. (2016). Object-aware bundle adjustment for correcting monocular scale drift. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 4770–4776. IEEE.

Gálvez-López, D., Salas, M., Tardós, J. D., and Montiel, J. (2016). Real-time monocular object SLAM. *Robotics and Autonomous Systems*, 75:435–449.

García Navarro, B. (2015). Implementación de técnicas de deep learning. Masterthesis, Universidad de la Laguna.

González Arroyo, Á. (2013). *Text detection and recognition in natural images using computer vision techniques*. PhD thesis, Universidad de Alcalá Escuela Politécnica Superior.

Gräter, J., Schwarze, T., and Lauer, M. (2015a). Robust scale estimation for monocular visual odometry using structure from motion and vanishing points. In *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pages 475–480. IEEE.

Gräter, J., Schwarze, T., and Lauer, M. (2015b). Robust scale estimation for monocular visual odometry using structure from motion and vanishing points. In *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pages 475–480. IEEE.

Herrera Alonso, A. (2015). Detección de texto utilizando redes neuronales convolucionales. Masterthesis, Universitat Politècnica de Catalunya.

Izaurieta, F. and Saavedra, C. (2000). Redes neuronales artificiales. *Departamento de Física, Universidad de Concepción Chile*.

Jaderberg, M., Vedaldi, A., and Zisserman, A. (2014). Deep features for text spotting. In *European conference on computer vision*, pages 512–528. Springer.

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

Knorr, S. B. (2017). *Leveraging the User's Face as a Known Object in Handheld Augmented Reality*. PhD thesis, Technische Universität München.

Krizhevsky, A. et al. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998a). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998b). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

Longuet-Higgins, H. (1986). The reconstruction of a plane surface from two perspective projections. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, pages 399–410.

López, R. F. and Fernandez, J. M. F. (2008). *Las redes neuronales artificiales*. Netbiblo.

Mallick, S. (2016). Homography examples using opencv (python / c ++). `https://www.learnopencv.com/homography-examples-using-opencv-python-c/`.

Minguell, M. E., Font, J. F., Cornellas, P., and Regás, D. C. (2012). Realidad aumentada y códigos QR en educación. In *Tendencias emergentes en educación con TIC*, pages 135–157. Espiral.

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

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

Pons, J. V. (2012). *Localización y generación de mapas del entorno (SLAM) de un robot por medio de una Kinect*. PhD thesis, Universitad Politécnica de Valéncia.

Restrepo Arteaga, G. J. P. et al. (2015). Aplicación del aprendizaje profundo (deep learning) al procesamiento de señales digitales. B.S. thesis, Universidad Autónoma de Occidente.

Rubio, N. G., Gámez, M., and Cortés, E. A. (2006). Redes neuronales artificiales: un largo e irregular camino hasta la actualidad. *Historia de la probabilidad y la estadística (III)*, page 233.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE.

Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H., and Davison, A. J. (2013). SLAM++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1352–1359.

Serimagmedia (2015). Redes neuronales convolucionales. la réplica en máquina del aprendizaje humano. `http://www.serimagmedia.com/redes-neuronales-convolucionales-la-replica-en-maquina-del-aprendizaje-humano`.

Serrano, E. R. P. and Hernández, M. G. G. (2015). Diseño de un sistema de reconocimiento automático de palabras contenidas en documentos históricos. *Jóvenes en la ciencia*, 1(2):1607–1612.

Shi, B., Bai, X., and Belongie, S. (2017). Detecting oriented text in natural images by linking segments. In *Proc. CVPR*, volume 3.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *ICLR 2015*.

Song, S., Chandraker, M., and Guest, C. C. (2013). Parallel, real-time monocular visual odometry. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4698–4705. IEEE.

Sucar, E. and Hayet, J.-B. (2017). Bayesian scale estimation for monocular SLAM based on generic object detection for correcting scale drift. *Cornell Uviversity Library*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., and Anguelov, D. (2015). Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*.

Varios (2015). Deep learning tutorial, release 0.1. `http://deeplearning.net/tutorial/lenet.html`.

Villalba, J. D., Gomez, I. D., and Laier, J. E. (2012). Detección de daño en vigas utilizando redes neuronales artificiales y parámetros dinámicos. *Revista Facultad de Ingeniería Universidad de Antioquia*, (63):141–153.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.