INSTITUTO TECNOLÓGICO DE LA PAZ
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
MAESTRÍA EN SISTEMAS COMPUTACIONALES

ANÁLISIS E IMPLEMENTACIÓN DE UNA RED NEURONAL CONVOLUCIONAL PARA LA ESTIMACIÓN DE POSE UTILIZANDO UN SISTEMA DE VISIÓN MONOCULAR

QUE PARA OBTENER EL GRADO DE
MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA:
JOSÉ ADRIÁN PÉREZ CHONG

DIRECTORES DE TESIS:
DR. SAÚL MARTÍNEZ DÍAZ

LA PAZ, BAJA CALIFORNIA SUR, MÉXICO, AGOSTO 2022.

Instituto Tecnológico de La Paz
División de Estudios de Posgrado e Investigación

La Paz, B.C.S., 16/ AGOSTO /2022

DEPI_MSC/045/2022

ASUNTO: Autorización de impresión

C. JOSÉ ADRIÁN PÉREZ CHONG,
ESTUDIANTE DE LA MAESTRÍA EN SISTEMAS COMPUTACIONALES,
P R E S E N T E .

Con base en el dictamen de aprobación emitido por el Comité Tutorial de la Tesis denominada: **"ANÁLISIS E IMPLEMENTACIÓN DE UNA RED NEURONAL CONVOLUCIONAL PARA LA ESTIMACIÓN DE POSE UTILIZANDO UN SISTEMA DE VISIÓN MONOCULAR"**, mediante la opción de tesis (Proyectos de Investigación), entregado por usted para su análisis, le informamos que se **AUTORIZA** la impresión.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®

DAMARIS NATHANAEL FLORES ZAMORANO,
SUBDIRECTORA ACADÉMICA.

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE LA PAZ
SUBDIRECCIÓN ACADÉMICA

c.c.p. Depto. de Servicios Escolares
c.c.p. Archivo.

DNFZ/icl*

Instituto Tecnológico de La Paz
División de Estudios de Posgrado e Investigación

DICTAMEN DEL COMITÉ TUTORIAL

La Paz, B.C.S., **15/AGOSTO/ 2022**

DAMARIS NATHANAEL FLORES ZAMORANO,
SUBDIRECTORA ACADÉMICA
P R E S E N T E.

Por medio del presente, enviamos a usted dictamen del Comité Tutorial de tesis para la obtención del grado de Maestro, con los siguientes datos generales:

No. de Control M18310006	Nombre JOSÉ ADRIÁN PÉREZ CHONG
Maestría en:	SISTEMAS COMPUTACIONALES
Título de la tesis: ANÁLISIS E IMPLEMENTACIÓN DE UNA RED NEURONAL CONVOLUCIONAL PARA LA ESTIMACIÓN DE POSE UTILIZANDO UN SISTEMA DE VISIÓN MONOCULAR	
DICTAMEN: Se autoriza el trabajo de investigación, en virtud de que realizó las correcciones correspondientes conforme a las observaciones planteadas por este Comité Tutorial.	

A t e n t a m e n t e .
El Comité Tutorial

DR. ISRAEL MARCOS SANTILLÁN MÉNDEZ

MSC. ILIANA CASTRO LIERA

DR. SAÚL MARTÍNEZ DÍAZ

c.c.p. Coordinador de la Maestría.
c.c.p. Departamento de Servicios Escolares.
c.c.p. Estudiante.

ITLP-DEPI-RTT-08 ***Rev.1***

Dedicatoria

A mis padres y abuelos.

Agradecimientos

Agradezco a todos mis profesores del posgrado del Instituto Tecnológico de La Paz por su siempre atento apoyo.

Al Instituto Tecnológico de La Paz por brindarme un espacio de estudio, aprendizaje y el equipo necesario para desarrollarme.

A mi familia por inculcarme la costumbre de superarme y su apoyo incondicional.

A mi amada Saraí por siempre escucharme y estar a mi lado.

Gracias a mi comité de tesis, Dr. Saúl Martínez Díaz, MSC. Iliana Castro Liera, Dr. Israel Marcos Santillán Méndez que me guiaron para lograr el objetivo.

Resumen

En este trabajo de tesis se presenta el análisis e implementación de una red neuronal convolucional para la estimación de pose utilizando un sistema de visión monocular como alternativa a los sistemas tradicionales. Se realiza la comparación de un sistema clásico de estimación de pose con múltiples detectores de puntos característicos tales como *Features from Accelerated Segment Test* (FAST), *Scale-invariant Feature Transform* (SIFT) y *Oriented Fast and Rotated BRIEF* (ORB). Se implementa un algoritmo de localización y detección de objetos mediante redes neuronales de convolución (CNN, *Convolutional Neural Network*) y se utiliza para obtener la pose de la cámara con factor escala mediante puntos conocidos de los objetos detectados.

Abstract

In this work we present an analysis and implementation of a Convolutional Neural Network for camera pose estimation using a monocular visión system as an alternative to traditional systems. Results from a comparison between multiple feature detector algorithms such as Features from Accelerated Segment Test (FAST), Scale-invariant Feature Transform (SIFT) and Oriented Fast and Rotated BRIEF (ORB) are analized. Lastly we implement a object detection and localization algorithm through a convolutional neural network (CNN) and use detected object measurements to obtain camera pose with scale factor.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

La visión por computadora ha tenido avances significativos en diversas áreas a lo largo de los últimos años como consecuencia de las mejoras en las técnicas de análisis de imágenes, el incremento en poder computacional y la popularidad de los sensores fotográficos en los dispositivos móviles. En las áreas dentro de la visión por computadora, la navegación auxiliada con técnicas como la odometría visual se basa fuertemente en la obtención de la pose de la cámara.

La estimación de la pose de la cámara no solo encuentra aplicaciones en áreas como la navegación auxiliada por visión de computadora, sino que también tiene aplicaciones en realidad aumentada y reconstrucción 3D.

En robótica y visión por computadora, la Odometría Visual (OV) es el proceso de determinar la posición y orientación de un robot mediante el análisis de las imágenes asociadas, es decir, estimar la pose de la cámara en una secuencia de imágenes.

1.1. Antecedentes

La Odometría Visual es el estudio de la estimación de la posición de vehículos durante la navegación a través del análisis de una secuencia de imágenes. Las ventajas de la OV están apegadas al bajo costo de los dispositivos fotográficos y a la gran cantidad de información que brindan las imágenes permitiendo buenos resultados. Uno de los primeros trabajos para estimar el movimiento a través de una cámara fue realizado por Moravec en [1], donde una computadora utiliza el análisis de imágenes para ubicar objetos en un ambiente 3D y así controlar un vehículo evitando obstáculos.

La inspiración para la odometría visual es puramente biológica, nuestros cerebros utilizan varias pistas para percibir profundidad y el flujo óptico, tales como variaciones en textura, gradientes, oclusión, tamaño de objetos conocidos, sombras, desenfoque, etc. Por lo tanto es normal el querer integrar alguno de estos métodos a los algoritmos conocidos.

La odometría visual se puede implementar monocular, estéreo o con cámaras RGB-D (cámaras con sensor de profundidad), dependiendo del diseño del sistema. La OV estéreo puede estimar la escala inmediatamente a diferencia de la OV monocular, sin embargo, la OV estéreo requiere un proceso de calibración mayor y una sincronización rígida, sin la cual, el error se propaga a lo largo del tiempo.

La implementación monocular es preferida debido al costo reducido que presenta y la posibilidad de aplicarse en estructuras de menor tamaño donde establecer una separación adecuada entre ambas lentes es complicado.

Existen algoritmos aceptados para la estimación del movimiento basado en una secuencia de imágenes como [2] en el que Scaramuzza & Fraundorfer describen diferentes alternativas como localización y mapeo visual simultáneo(V-SLAM, *Visual Simultaneous Localization and Mapping*), odometría visual estéreo y odometría visual monocular. La OV basada en geometría como se observa en la fig. 1.1 es capaz de calcular el flujo óptico en secuencias de imágenes y su precisión depende del correcto emparejamiento de puntos singulares o característicos como se demuestra en [3]. Contamos con otro tipo de OV basado en algoritmos de aprendizaje que son entrenados con datos etiquetados, aunque este último no esta completamente definido.

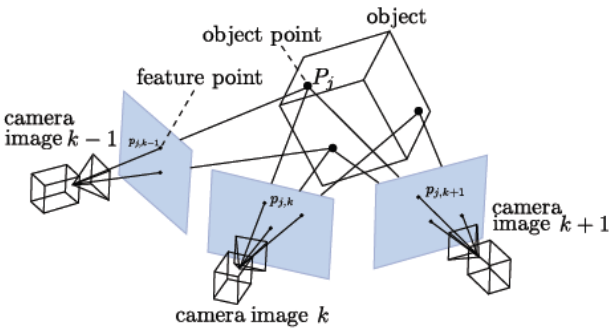

Figura 1.1: Conjunto de puntos 3D en una secuencia de imágenes y pose relativa de la cámara.

Los trabajos actuales de reconocimiento de objetos buscan usar técnicas de aprendizaje automatizado como las redes neuronales convolucionales (CNN, *convolutional neural network*).

Uno de los puntos más interesantes de las CNNs es la habilidad de aprender características de un conjunto de datos y usarlos eficientemente para la tarea asignada.

Muchos trabajos en reconocimiento se han enfocado en recolectar muestras de datos de gran tamaño para generar modelos más poderosos y con mejores técnicas con el objetivo de reducir el sobreentrenamiento. Algunos de las modelos que han mostrado gran efectividad y eficiencia en los últimos años son AlexNet[4] y GoogleNet[5].

Las CNNs pueden aprender a detectar cientos de objetos a través de millones de imágenes, por lo tanto requieren de modelos con grandes capacidades de aprendizaje. Esta capacidad se puede controlar variando la profundidad y amplitud de la red. A pesar de lo atractivo de las CNNs estas suelen ser relativamente costosas de aplicar en imágenes de alta resolución. Afortunadamente las únidades de procesamiento gráfico (GPU, *Graphics Processing Unit*) actuales son lo suficiente potentes como para facilitar el entrenamiento de redes de escalas interesantes.

You Only Look Once (YOLO, por sus siglas en inglés) es una red neuronal convolucional capaz de localizar y clasificar objetos dentro de una imagen con buena precisión y velocidad. Este sistema busca igualar la velocidad y precisión del sistema visual humano en identificar objetos mediante imágenes para facilitar tareas complejas como la navegación automatizada. Un algoritmo rapido y preciso permite la posibilidad para que los equipos de computo puedan manejar vehiculos sin sensores especializados, tambien permite dispositivos de asistencia para obtener información de un entorno y brindarla a usuarios humanos y en general crea la posibilidad para sistemas roboticos responsivos.

1.2. Descripción del problema

Al ser la OV un sistema de localización incremental, el error producido no se corrige en ningún momento sino que es acumulado y propagado a futuras estimaciones.

Además del error acumulado siempre se cuenta con valores anómalos (*outliers*) y la asunción de que el escenario es estático, para combatir estos problemas se utilizan algoritmos robustos como *Random Sample Consensus* (RANSAC)[6] o técnicas de filtrado incrementando la complejidad computacional del cálculo. Debido a esto, se utilizan diferentes técnicas como la fusión de sensores o algoritmos de aprendizaje con datos designados para la disminución de errores.

Aunque se utilizan secuencias de imágenes para estimar la trayectoria de la cámara, existe

mucha información dentro de las imágenes que no se aprovecha, por ejemplo Mo & Sattar[7] utilizan un sistema estéreo donde las nubes de puntos 3D sirven para reconocer ubicaciones conocidas y mejorar la estimación de ruta. En [8] se incorpora una base de datos de dimensiones de estructuras conocidas como paredes, edificios, etc.

El propósito de este trabajo es analizar otra alternativa basada en utilizar la información brindada por objetos conocidos dentro de las secuencias de imágenes analizadas y de esta forma preservar y/o recuperar el factor escala a lo largo del trayecto.

1.3. Objetivos

1.3.1. Objetivo general

Analizar e implementar un sistema de visión monocular que permita estimar la pose de la cámara y calcular la escala del movimiento mediante la detección de objetos con puntos conocidos.

1.3.2. Objetivos específicos

- Implementar un sistema de odometría visual monocular.
- Implementar y entrenar un sistema de red neuronal convolucional para detección y localización de objetos.
- Integrar el sistema de visión monocular con la red neuronal convolucional entrenada.
- Evaluar el desempeño del sistema integrado.

1.4. Limitaciones y alcance

Se desarrollará un sistema de visión por computadora capaz de estimar la pose de la cámara y permita obtener un factor de escala a través de objetos conocidos detectados en la escena. El sistema se probará en un ambiente controlado donde la iluminación permita capturar imágenes con buena visibilidad. Las escenas captadas dentro del ambiente deberán ser estáticas.

Capítulo 2

Marco teórico

En este capítulo se presenta el marco teórico para el análisis y desarrollo del sistema que permita estimar la pose de la cámara apoyándose de una red neuronal convolucional. Primero se describe un sistema de visión por computadora y sus componentes, continuamos con la descripción del funcionamiento de un algoritmo clásico de odometría visual que utiliza la estimación de pose de la cámara. De misma forma se detallan los algoritmos más utilizados para la extracción de características en análisis de imágenes. Por último se determina la estructura de una red neuronal convolucional que nos permita detectar y localizar objetos dentro de imágenes.

2.1. Sistema de visión por computadora

La visión artificial o visión por computadora consiste en la toma de imágenes y extracción de información de las mismas. Para lograrlo son necesarios componentes tales como iluminación, lentes fotográficos (también conocidos como objetivo), sensores fotográficos y equipo de procesamiento computacional. En el caso de los sistemas de visión que solo utilizan un sensor fotográfico o cámara se les conoce como sistemas de visión monoculares.

2.2. Estimar la pose de la cámara

Estimar la pose de la cámara consiste en determinar la transformación 3D necesaria para mover un punto en la escena del sistema global de coordenadas al sistema de coordenadas de la cámara. Como se mencionó anteriormente ésta técnica es un paso fundamental en aplicaciones

de navegación para determinar la posición del robot en el entorno mediante la correspondencia en una serie de imágenes.

2.2.1. Calibración de la cámara

Calibrar una cámara significa encontrar sus parámetros internos, externos y posibles distorsiones que ésta tenga. Una cámara tiene dos tipos de parámetros, los internos o intrínsecos que son aquellos que describen el funcionamiento de una cámara como la distancia focal, el punto principal y el centro óptico. El otro tipo de parámetros son los parámetros extrínsecos que son aquellos externos al funcionamiento de la cámara como la posición y la orientación.

La calibración es un paso necesario para obtener información precisa de estructuras 3D del mundo usando una imagen capturada por una o más cámaras.

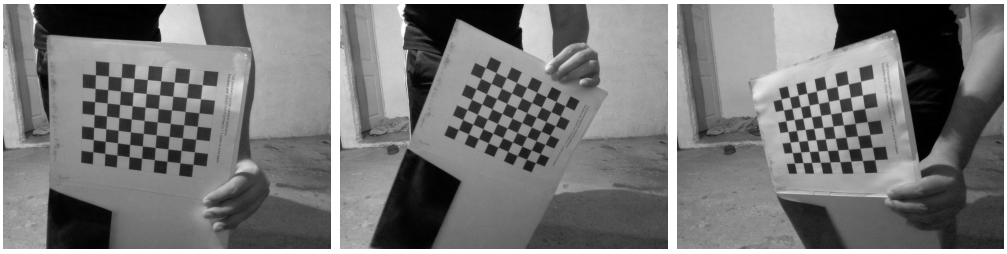

Figura 2.1: Patrón de tablero de ajedrez

En esencia el proceso de calibración de cámara consiste en determinar la geometría y características internas de la cámara calculando mediante un patrón de calibración que contiene rasgos fácilmente detectables de manera precisa en la imagen capturada.

Existen dos tipos de patrónes que son altamente utilizados, el patrón de tablero de ajedrez mostrado en la figura 2.1 y el patrón de círculos mostrado en la figura 2.2.

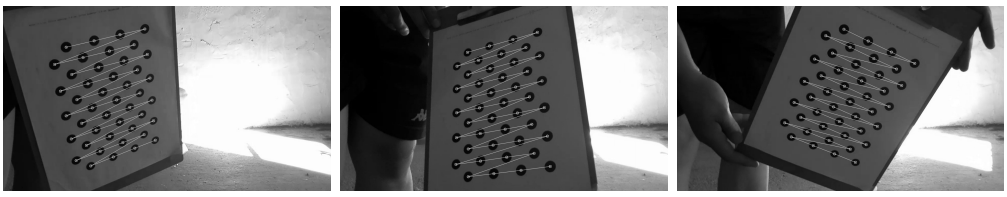

Figura 2.2: Patrón de círculos

2.2.2. Obtener pose de la cámara

El modelo básico de cámara es el modelo de cámara estenopeica (*pinhole camera model*), consiste en una caja con un pequeño orificio (el estenopo) que recibe luz y entra por el orificio, la luz es recibida por un material fotosensible dentro de la caja. Con esto en mente podemos describir los elementos de la cámara estenopeica como el punto focal o centro de proyección siendo el estenopo, y el plano imagen siendo el material fotosensible encargado de captar la luz. Además de los elementos anteriores también contamos con la distancia focal que es la distancia entre el estenopo y nuestro plano imagen como se observa en la figura 2.3.

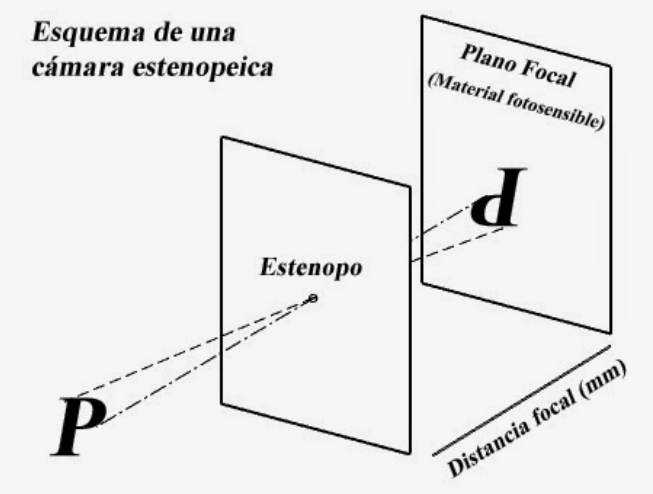

Figura 2.3: Esquema de una cámara estenopeica.

Considerando lo anterior podemos describir la relación matemática entre la proyección de puntos en el espacio 3D hacia el plano imagen. Con la relación $\mathbf{p} \in \mathbb{R}^3$ y su proyección $\mathbf{x}$ definida por:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} T \begin{bmatrix} p \\ 1 \end{bmatrix}$$

donde K es la matriz de los valores intrínsecos de la cámara y T es la matriz de los valores extrínsecos de la cámara.

La matriz K de los valores intrínsecos contiene los parámetros proyectivos de la cámara

como distancia focal, f_x y f_y, y el centro de la cámara, c_x y c_y:

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

por otra parte la matriz T de los valores extrínsecos de la cámara define una matriz 4×4 que representa la pose de la cámara como una transformación 3D euclidiana, por ejemplo una concatenación de una rotación R y una translación t.

$$T = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$$

La estimación de la pose de la cámara se realiza determinando los valores de R y t de un conjunto de correspondencias entre puntos del entorno y sus proyecciones en las imágenes de la cámara. Por medio de dos pasos, primero buscamos las correspondencias de las imágenes por medio de técnicas de procesamiento de imágenes. Después estas correspondencias se utilizan para estimar la pose a través de minimizar el valor de la función del error de la proyección basada en el modelo del agujero.

2.3. Detección de características

Para poder relacionar un par de imágenes en una secuencia de imágenes es necesario encontrar una correspondencia entre ambas, esto lo realizamos mediante el emparejamiento de ciertos puntos característicos. Los puntos característicos son puntos que se pueden encontrar en ambas imágenes por medio de algoritmos llamados detectores de características. Además de extraer dichos puntos, los algoritmos detectores de características también cuentan con técnicas que nos permiten describirlos y de esta forma emparejarlos en otra imagen.

Existen dos tipos de detectores de características, los detectores de esquinas y los que detectan áreas o puntos de interés. Los detectores de esquinas usados comúnmente son Moravec [1], Forstner [9], Harris [10], Shi-Tomasi [11], y FAST[12]. Estos detectores suelen no ser robustos ante cambios de escala y de ángulo de visión. Los detectores de puntos de interés son más tolerantes a cambios por lo que pueden ser más eficaces al momento de extraer características

y emparejarlas en una secuencia de imágenes. Entre los detectores de puntos de interés más populares tenemos SIFT[13], SURF[14] y ORB[15]. Las características obtenidas en un instante deben ser emparejadas con las características obtenidas en el instante anterior.

2.3.1. Features from Accelerated Segment Test

Features from Accelerated Segment Test (FAST) es un algoritmo que se encarga de comparar la intensidad del punto de interés p con la intensidad de su vecindad de píxeles como podemos observar en la fig. 2.4. Para detectar una esquina es necesario que una secuencia de un cierto número de píxeles de la vecindad superen un umbral de intensidad. Este algoritmo se destaca por su velocidad de ejecución que es menor a otros detectores de esquinas, pero no es efectivo ante altos niveles de ruido o cambios en escala. Su desempeño correcto depende de una correcta configuración del umbral de intensidad.

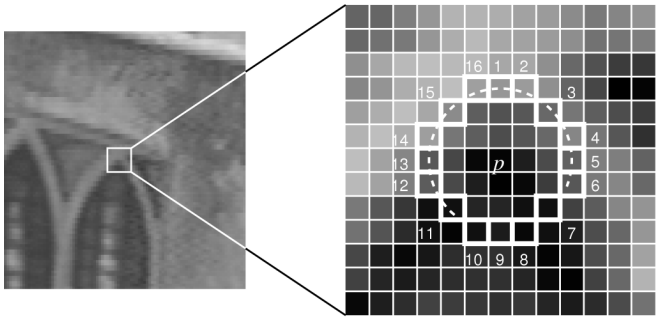

Figura 2.4: FAST, punto de interés cuya vecindad tiene una secuencia de píxeles que supera el umbral definido.

2.3.2. Speed Up Robust Features

El algoritmo *Scale-Invariant Feature Transform* (SIFT) se encarga de detectar y describir puntos de interés de una manera eficiente. SIFT se basa en operaciones que utilizan las diferencias de gaussianas e imágenes piramidales. La robustez del algoritmo se basa en identificar la orientación del punto de interés y su vecindad de píxeles. La orientación se obtiene mediante operaciones gaussianas en la vecindad circular de píxeles para obtener un vector que describa al punto de interés de forma única.

2.3.3. Oriented FAST and Rotated BRIEF

Oriented FAST and Rotated BRIEF (ORB) es otra alternativa que permite la extracción y descripción de puntos de interés. Es una combinación de la detección FAST con la técnica de descripción de puntos de interés *Binary Robust Independent Elementary Features* (BRIEF). Para lograr la efectividad ante variaciones en escala el algoritmo FAST se aplica en una pirámide con diferentes escalas de la imagen. ORB también utiliza una variante de BRIEF capaz de trabajar ante rotaciones. Por lo tanto ORB es robusto ante cambios de escala y rotación. Un punto importante de ORB es el reducido costo computacional que presenta, por lo tanto, es una opción importante al trabajar con sistemas de recursos limitados.

2.4. Eliminación de valores anómalos

Debido a que la extracción y el emparejamiento de características en imágenes secuenciales no es perfecta, es necesario una forma de eliminar valores anómalos. RANSAC es un algoritmo iterativo que toma un muestreo aleatorio de nuestros emparejamientos para obtener la matriz esencial, posteriormente pondera esta matriz con el resto de correspondencias. Después de un número de iteraciones el algoritmo termina dando como resultado la matriz con mayor ponderación respecto a los puntos evaluados. Al tener un carácter aleatorio este algoritmo no obtiene siempre el mismo resultado, pero en la gran mayoría de los casos es el cercano al óptimo.

2.5. Red neuronal convolucional

Las redes neuronales artificiales son un modelo computacional que buscan simular el comportamiento de un cerebro biológico mediante neuronas artificiales que se integran en filas llamadas capas y se encuentran conectadas con neuronas de otras capas. Estos modelos permiten tareas de aprendizaje automatizado donde mediante un entrenamiento basado en conjuntos de datos extensos se obtienen resultados con alta precisión para tareas específicas.

Por otra parte, las redes neuronales convolucionales son redes neuronales artificiales que buscan simular los campos receptivos de las neuronas en la corteza visual. Esto se logra mediantes capas de filtros convolucionales aplicadas en matrices bidimensionales, lo que permite mejores resultados en tareas de visión artificial.

YOLO es una red neuronal convolucional para la detección de objetos. El resultado de la red es una imágen dividida en una cuadrícula de celdas, cada celda es responsable de predecir un objeto en la imágen. El objeto debe ser predecido por la celda en la que se encuentra el centro del objeto. Cada celda predice una cantidad B de cajas delimitantes y una cantidad C de predicciones de clase de objeto. La caja delimitante tiene un formato de $x, y, w, h, certeza$ donde x y y son las coordenadas del centro de la caja, w y h son el ancho y alto respectivamente de la caja. La certeza simboliza la presencia o ausencia de un objeto dentro de la caja. El sistema también predice las probabilidades de que el objeto detectado pertenezca a una clase en especifico, por lo que aparte de los valores anteriores también devuelve una cantidad C de valores que nos indican a que clase pertece el objeto dentro de esta celda.

Capítulo 3

Metodología

En este capítulo se describe el sistema de visión monocular para estimar la pose de la cámara utilizando los algoritmos detectores de caracteristicas conocidos, el cual nos permite estimar la matriz con rotación y traslación de la cámara según se percibe en una secuencia de pares de imágenes.

3.1. Captura de secuencia de imágenes

La captura de imágenes se realizó con dos modelos de cámaras diferentes que se muestran en la figura 3.1, logitech c270 y logitech c920s. El modelo c270 permite una resolución máxima de 1280x720 mientras que el modelo c920s permite una resolución máxima de 1920x1080. Se desactivaron funciones como el enfoque automático y balance automático de blancos para evitar variaciones en las imágenes.

Figura 3.1: Cámaras logitech c270 y c920s.

El entorno donde se realizó la captura de imágenes necesita tener una buena iluminación para captar adecuadamente los puntos característicos por lo que se optó por realizar la secuencia en una habitación cerrada donde se controlaba la iluminación mediante luz LED.

3.2. Calibración de la cámara

Para poder obtener los valores intrínsecos y de distorsión de la cámara es necesario capturar una serie de imágenes donde podamos observar un conjunto de puntos cuyas coordenadas 3D del mundo real sean conocidas. Se calibraron las cámaras con un patrón de tablero de ajedrez con seis esquinas internas en cada fila y nueve esquinas internas en cada columna basandose en el algoritmo descrito por Zhang en [16].

3.3. El algoritmo

El algoritmo de odometría visual recibe una secuencia de imágenes provenientes de una cámara. Asumiendo los cuadros de la secuencia de imagenes sean capturados en un tiempo t y $t+1$, nos referimos a estas como I^t, I^{t+1}. Para cada par de imágenes necesitamos encontrar la matriz de rotación R y el vector de traslación t, estas matrices describen el movimiento de la cámara en un par de imágenes. Es necesario conocer los parametros intrínsecos de la cámara obteniendolos previamente por el proceso de calibración realizado con algún patrón de calibración.

3.3.1. Pseudocódigo del algoritmo

1. Capturar imágenes: I^t, I^{t+1}
2. Corregir distorsión de ambas imágenes.
3. Aplicar algoritmo de detección de puntos característicos en I^t.
4. Calcular matriz esencial que describe la correspondencia de puntos en amba
5. Estimar R, t de la matriz esencial calculada anteriormente.

3.3.2. Corrección de distorsión

La distorsión ocurre cuando lineas rectas en el mundo real se convierten en lineas curvas al ser capturadas por el lente de la cámara. Realizamos la corrección de distorción por medio de un algoritmo que mapea las coordenadas de una imagen sin distorsión a una imagen con distorsión. Esto lo logramos distorsionando lo que seria nuestra imagen sin distorsion con los parametros conocidos de distorsion y luego mapeando la intensidad de los puntos con distorción a la imagen origen sin distorsión. La corrección es un paso necesario para obtener información precisa y se realiza con ayuda de los parametros de distorsión obtenidos durante la calibración.

1. Convertimos las coordenadas de los pixeles (u_{dst}, v_{dst}) a coordenadas normalizadas (x', y').
2. Aplicamos el modelo de distorsión de lente para obtener las coordenadas normalizadas (x'', y'').
3. Convertimos (x'', y'') a coordenadas de los píxeles distorsionados (u_{src}, v_{src}) usando la matriz de calibración k.
4. Interpolamos para encontrar los valores de intensidad asociados a (u_{src}, v_{src}) en nuestra imagen origen y la asignamos a la imagen destino en su píxel (u_{dst}, v_{dst}).

3.3.3. Detección de puntos característicos

Aplicamos algún algoritmo de detección de puntos característicos como FAST, SIFT u ORB en ambas imágenes para obtener dos conjuntos de puntos característicos que comparamos para obtener una correspondencia entre ambas imágenes. En los experimentos el algoritmo que demostró ser más fiable a parte de permitir describir los puntos caracteristicos fue SIFT. Se buscaron los mejores puntos característicos para evitar valores anomalos. Cada punto característico detectado por SIFT cuenta con un valor *response* que describe que tan robusto es el punto, tomamos una cantidad de puntos cuyo valor *response* sea el más alto dependiendo de la cantidad de puntos detectados.

3.3.4. Seguimiento de puntos característicos

Por medio del algoritmo de Kanade-Lucas-Tomasi[17] rastreamos un punto característico en puntos adyacentes y buscamos el vector óptimo para dicho punto característico. Repetimos el procedimiento con el resto de puntos, si la correspondencia de puntos supera un valor umbral, consideramos dicho vector o movimiento como correcto y exitoso el rastreo de puntos.

3.3.5. Calcular matriz esencial

Una vez tenemos las correspondencias de puntos característicos aplicamos una optimización para calcular la matriz esencial. La matriz esencial se define como:

$$y_1^T E y_2 = 0$$

En la ecuación y_1, y_2 son las coordenadas homogéneas normalizadas de la imagen. Si todos nuestros puntos que corresponden son perfectos entonces solo necesitamos cinco puntos característicos entre un par de imágenes secuenciales para estimar el movimiento correctamente. Desafortunadamente, los algoritmos de rastreo de puntos no son perfectos y esto genera varias correspondencias incorrectas. RANSAC es un algoritmo iterativo que, en cada iteración, aleatoriamente toma cinco puntos del conjunto de correspondencias y estima la matriz esencial. Si las correspondencias y verifica si el resto de puntos concuerdan con la matriz esencial. El algoritmo termina después de cierto número de iteraciones y la matriz esencial con el mayor número de correspondencias correctas es retornada.

3.3.6. Construcción de la trajectoria

Al definir la pose de la cámara por R_{pos}, t_{pos}. Podemos rastrear la trayectorio con la siguiente ecuación:

$$R_{pos} = RR_{pos}$$

$$t_{pos} = tR_{pos}$$

3.3.7. Red neuronal convolucional

Con la ayuda de la paquetería *Keras* en el lenguaje de programación *Python*, se implementó una red neuronal convolucional del tipo *YOLO* con una estructura de 24 capas de convolución que extraen características y 2 capas *fully connected* para la predicción de los objetos detectados y su clase. Se utilizó como base la red conocida como *Inception* que previamente fue entrenada con el conjunto de datos de *Imagenet* para facilitar el entrenamiento.

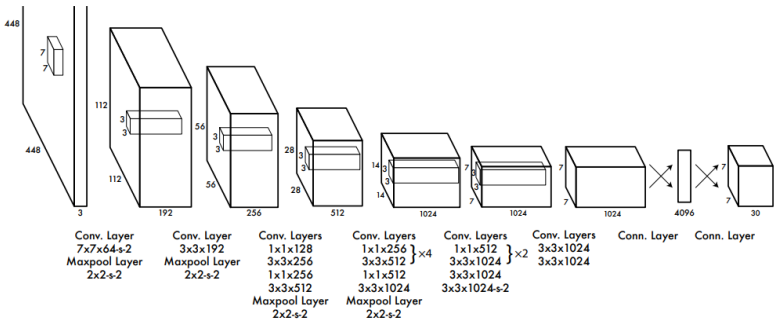

Figura 3.2: La arquitectura de la red cuenta con 24 capas de convolución seguida de 2 capas completamente conectadas.

La arquitectura de la primera versión de YOLO se ejecutó en el dataset de clases de objetos Pascal (*PASCAL VOC*), siguiendo el ejemplo de los autores usamos $S = 7$ y $B = 2$ donde S es el número de celdas en la imágen y B la cantidad de cajas predictivas por celda. Utilizando la misma estructura de celdas y cajas predictoras entrenamos nuestra red con el conjunto de datos con el objetivo de que la red aprenda a extraer características, detectar y predecir objetos en nuestras imágenes.

Debido a que la salida de nuestra red consiste de un vector $x, y, w, h, c, p1, p2, ..., p20$ fue necesario aplicar una función para calcular el error de nuestra red. La función consiste en calcular el error en las coordenadas detectadas de la caja, error en la confianza de que exista un objeto en la caja, error en la clase del objeto y por último un error de que no exista objeto que ocurre cuando se detecta un objeto donde no existe uno.

Al tener varios valores de error es necesario ponderarlos de diferente forma, considerando que el error en las coordenadas de las cajas predictoras tiene mayor importancia al error de la clase de objeto detectado. Para efectos de nuestra red se utilizó un valor de 10 en la ponderación del error de coordenadas, 1 en la confianza de objeto detectado, 1 en la clase de objeto y 0.5 en el error cuando no existe objeto.

Para terminar de entrenar la red se agregó una clase de objeto propia por medio de 50 imágenes etiquetadas manualmente. Se procedio a entrenar la red con 5,000 imágenes del conjunto de datos *PASCAL* durante 135 épocas resultando en una red capaz de predecir objetos y su clase. Después volvimos a entrenar la red resultante con nuestra clase de objeto de nuevo por 135 épocas.

Capítulo 4

Resultados

En este capítulo se presentan las pruebas realizadas a nuestra implementación del algoritmo de odometría visual, los diferentes métodos de extracción y descripción de puntos característicos, las predicciones de la red para detección y clasificación de objetos, la toma de la secuencia de imágenes y la implementación de la red con nuestro algoritmo de odometría visual.

4.1. Algoritmo de odometría visual y extracción de puntos característicos

Los experimentos se realizaron en el conjunto de datos KITTI[18], el conjunto de datos se conforma de una serie de fotogramas que se obtienen de un recorrido en la calle como se observa en la fig. 4.1. En dichas pruebas se obtuvieron resultados donde se muestra el error en la estimación de ruta.

La elección del algoritmo de extracción y descripción de características no es algo trivial y debe ser tomado en cuenta. Los experimentos se realizaron con el fin de encontrar el extractor de características que se desempeñe mejor. Los resultados se comparan con los valores verdaderos obtenidos por medio de sensores láser.

El detector de esquinas FAST a pesar de ser un detector de esquinas incapaz de trabajar con cambios de escala y rotaciones demostró un buen desempeño como se muestra en la tabla 4.1. Cabe mencionar que el movimiento que realiza la cámara entre un instante t y un instante $t+1$ no es de gran tamaño, de igual manera el conjunto de datos no contiene cambios bruscos

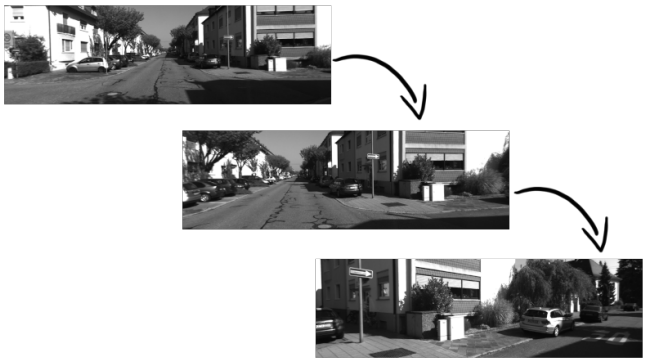

Figura 4.1: Frames 90, 100 y 110 de la secuencia del KITTI dataset.

en el ángulo de visión.

Tabla 4.1: Desempeño del algoritmo de OV con FAST

Frame	***Error absoluto en metros***	***Error acumulado en metros***
100	0.215888	2.333499
400	3.820372	47.092479
700	9.252882	235.379759
1000	13.413948	588.010515

En el caso del algoritmo SIFT los resultados que podemos ver en 4.2 demostraron ser mejores que los demas detectores de características utilizados. Demuestra un desempeño similar a FAST en los primeros frames pero conforme la cámara continua alejándose del origen el resultado se desvía por una cantidad menor que en los otros algoritmos.

Tabla 4.2: Desempeño del algoritmo de OV con SIFT

Frame	***Error absoluto en metros***	***Error acumulado en metros***
100	0.387551	2.587993
400	1.897696	51.734162
700	3.305652	129.757265
1000	3.415369	218.565254

La tabla 4.3 nos muestra que ORB fue el de peor desempeño ante la secuencia analizada, aunque al inicio se mantiene en un desvío menor, conforme la cámara rota y aumenta de velocidad en sus movimientos el error aumenta a valores bastante considerables.

Cabe mencionar que los resultados anteriores se auxilian de la escala absoluta obtenida de

Tabla 4.3: Desempeño del algoritmo de OV con ORB

Frame	***Error absoluto en metros***	***Error acumulado en metros***
100	2.334228	11.414300
400	25.724679	276.235144
700	9.810927	1382.001411
1000	74.890377	2823.852631

los valores de los sensores láser provistos por conjunto de datos. A pesar de que la trayectoria de translación y rotación es estimada se escala con valores reales lo que disminuye el error.

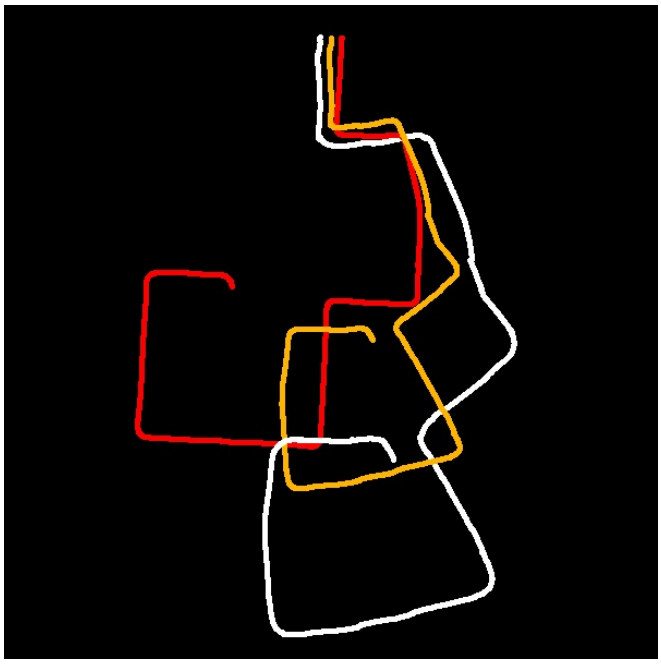

Figura 4.2: Trayectoria real, estimada con escala y sin escala de la primera secuencia del *KITTI dataset*.

Se llevó un pequeño experimento donde se comparan los valores reales contra los estimados con escala y los estimados sin escala demostrando la importancia de la escala. En la figura 4.2 podemos observar la comparación entre dichos valores, siendo la línea roja la trayectoria con los valores reales, la línea naranja la trayectoria estimada apoyándonos de un valor escala y la línea blanca la trayectoria estimada sin utilizar escala.

4.2. Red para detección y clasificación de objetos

Al terminar de entrenar la red con el conjunto de datos de *PASCAL VOC 2007* se evaluó la red con una parte del conjunto seleccionada para evaluación. El resultado nos muestra que a pesar de que la red entrenada cumple con el objetivo tiene problemas como la detección de múltiples objetos en la misma imagen.

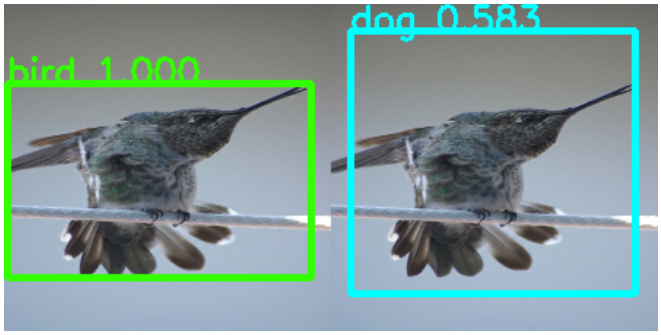

Figura 4.3: Clasificación de los objetos detectados

La red es capaz de detectar objetos en las imágenes, pero se equivoca en la clase de objeto detectado como observamos en 4.3 donde en lugar de predecir un pájaro predecimos un perro. Esto es debido a que nuestra implementación de la ponderación del error de la clase detectada es baja en consideración al error de las coordenadas de la caja predictora.

Figura 4.4: Detección de múltiples objetos próximos

De la misma forma aunque la red es capas de detectar objetos, se le complica detectar múltiples objetos cuando se encuentran muy próximos entre ellos. Esto lo podemos observar en la figura 4.4 donde solo nos predice un objeto de la clase avión en lugar los de cinco que se encuentran en la imagen.

Figura 4.5: Detección de objeto conocido

A pesar de los problemas que tiene nuestro detector nos permite detectar objetos adecuadamente. En la figura 4.5 cumple correctamente la detección de un objeto de la clase persona, aunque en la imagen se encuentran dos objetos de dicho tipo, la caja detecta ambos lo que permite usar la red para nuestro objetivo el cual es detección de zonas de interés.

4.3. Secuencia de imágenes

Se tomaron imágenes de una escena donde se encontraba un objeto conocido, realizando movimientos cortos en la posición de la cámara para poder estimar la trayectoria de esta. Al momento de estimar la trayectoria nos encontramos con un detalle importante, el ruido generado por la inconsistencia en la luz arrojaba valores anómalos en los puntos característicos, este problema no se presentó al momento de la estimación en el conjunto de datos *KITTI*.

Se asumió que el error podía ser generado por la baja calidad del sensor fotográfico de la cámara Logitech c270. Se procedió a realizar la captura de la secuencia de imágenes con el sensor fotográfico encontrado en la cámara Logitech c920s. Al detectar los puntos característicos de la nueva secuencia se encontró una disminución en los valores anómalos pero seguían detectándose y siendo un problema.

Cabe mencionar que dichos valores anómalos eran relevantes al momento de procesar dos imágenes cuando la cámara no realiza ningún movimiento entre dichas imágenes. Al no existir un movimiento, nuestros algoritmos estiman una translación y rotación con los valores que cambiaron, es decir los valores anómalos. Para resolver este problema se pueden utilizar otras técnicas de optimización, pero optamos por ignorar las imágenes cuando no existe un movimiento dominante.

4.4. Implementación de red con algoritmo OV

Teniendo una secuencia de imágenes con un objeto con dimensiones conocidas se procedió a estimar la trayectoria con la pose de la cámara enfocando la detección de puntos característicos en la zona detectada por nuestra red de detección de objetos. El movimiento dominante que se observa en la secuencia es un movimiento frontal que es estimado correctamente.

La estimación de nuestro algoritmo a pesar de ser correcta en el movimiento realizado, al no tener una escala los valores obtenidos son alejados a los reales. Aprovechando que conocemos las dimensiones y las coordenadas de nuestros puntos característicos en el objeto detectado podemos obtener una aproximación de escala. Se realizó la detección de puntos característicos en la caja predicha por la red y se emparejaron los puntos en nuestras imágenes de la secuencia para estimar la trayectoria.

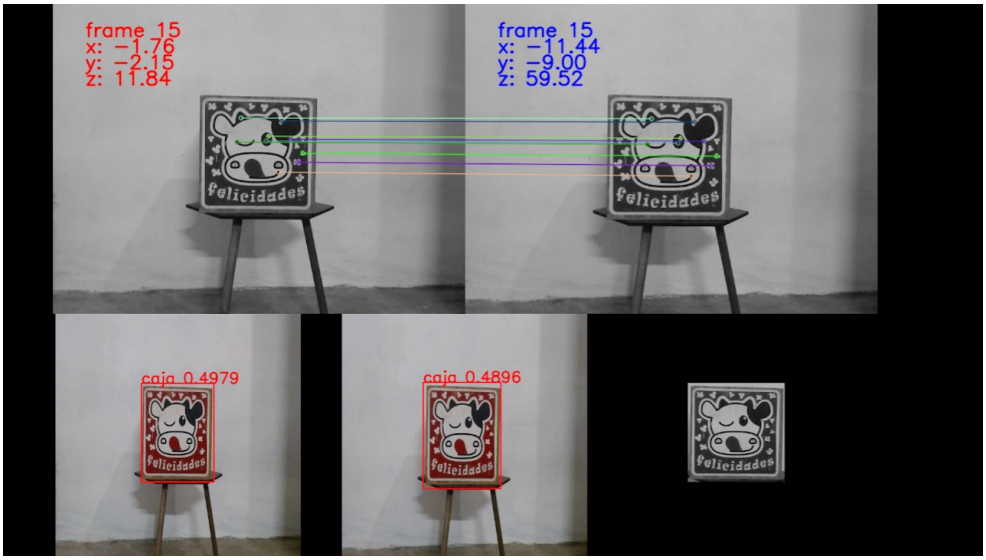

Figura 4.6: Resultado de la integración de la red con algoritmo de OV.

En la figura 4.6 observamos dos imágenes en instantes diferentes de la misma escena con un movimiento de la cámara. La imagen superior izquierda es una imagen en el instante t y la imagen superior derecha es una imagen en el instante $t+1$ de nuestra secuencia. La imagen del instante t cuenta con texto que describe el movimiento estimado sin apoyarse de una escala. La imagen del instante $t+1$ cuenta con texto que describe el movimiento estimado con escala estimada. Las imágenes en la parte inferior muestran la detección de objeto obtenida por nuestra red y por último en la parte inferior derecha vemos el resultado de aplicar una máscara en nuestras imágenes para detectar puntos característicos solo en nuestra zona de interés, es decir, la caja

predictora.

El movimiento realizado por la cámara fue de 90 cm de frente con ligeras variaciones en el ángulo por lo que los resultados no son del todo precisos, pero podemos observar que el resultado sin escala del movimiento dominante al ser de tan solo una decena de unidades no se acerca al valor real, mientras que el valor del movimiento estimado con escala se acerca al real con seis decenas de nueve siendo seis veces mejor que sin aprovechar los valores del objeto conocido.

Capítulo 5

Conclusiones

Al momento de estimar trayectoria, la efectividad de la OV es dependiente de los algoritmos de extracción de características utilizados, aún con la aplicación del algoritmo correcto existe lugar a la mejora de precisión. En los experimentos realizados el algoritmo de extracción de características con mejor desempeño fue *SIFT* debido a su robustez a cambios de escala y rotación. Este algoritmo es ideal para el emparejamiento de puntos gracias ya que no solo extrae puntos si no que también los describe.

En cuanto a generar una secuencia de imágenes, es importante considerar la iluminación y el ruido que pueda captarse. El uso de una buena cámara ayuda a eliminar el ruido. También se pueden usar filtros en las imágenes para eliminar el ruido. Por último, la falta de un movimiento dominante en la secuencia de imágenes debe considerarse debido a que suele generar error.

La estimación de pose y la OV son campos que pueden mejorarse aprovechando información adicional a la obtenida con los métodos tradicionales. Las CNN presentan una alternativa que merece explorarse más a fondo para la integración en los métodos de estimación de pose.

Las redes para la detección y localización de objetos pueden ayudar a estimar la pose de la cámara, pero su aplicación requiere de un tiempo alto en la recolección de datos y en su entrenamiento.

5.1. Trabajo futuro

Se desarrolló un sistema que explora la alternativa de aprovechar información conocida previamente mediante el uso de redes para detección y clasificación de objetos. Los experimentos

realizados se basan en la detección de un objeto, es necesario explorar el desempeño con un conocimiento previo de un mayor número de objetos y en una secuencia con mayor distancia.

Aunque el sistema aprovecha las características de las redes para encontrar información conocida dentro de la imagen, las redes neuronales convolucionales cuentan actualmente con ciertas estructuras que deberían permitir la extracción de características y la estimación de la pose de la cámara completamente dentro de la red.

Las redes neuronales siameses permiten procesar dos imágenes al mismo tiempo y al final combinar ambas en una capa para obtener un resultado deseado. También existen redes neuronales con retroalimentación donde la capa de salida se conecta a la capa entrada, lo que permite a la red tener memoria de entradas anteriores. Ambas estructuras se ajustan a las necesidades de procesar imágenes en una secuencia, permitiendo obtener una estimación de la pose de la cámara. Implementar cualquiera de ambas estructuras nos brindará una perspectiva diferente donde el resultado pueda presentar una mejor precisión.

De la misma forma existen técnicas de optimización utilizadas en la fotogrametría que se enfocan en minimizar el error de n puntos característicos detectados, lo que nos permite minimizar el error incremental en la estimación de la trayectoria. Implementar un sistema aprovechando las técnicas conocidas de fotogrametría permitirá un sistema con mayor desempeño.

Bibliografía

[1] Moravec, H. P. (n.d.). Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover.

[2] Scaramuzza, D., & Fraundorfer, F. (2011). Tutorial: Visual odometry. IEEE Robotics and Automation Magazine, 18(4), 80–92.

[3] Chien, H. J., Chuang, C. C., Chen, C. Y., & Klette, R. (2016). When to use what feature? SIFT, SURF, ORB, or A-KAZE features for monocular visual odometry. International Conference Image and visión Computing New Zealand, 0.

[4] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (n.d.). ImageNet Classification with Deep Convolutional Neural Networks.

[5] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2015). Going Deeper with Convolutions.

[6] Fischler, M. A., & Bolles, R. C. (1980). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography.

[7] Mo, J., & Sattar, J. (n.d.). Place Recognition for Stereo Visual Odometry using LiDAR Descriptors.

[8] Hilsenbeck, S., Moller, A., Huitl, R., Schroth, G., Kranz, M., & Steinbach, E. (2012). Scale-preserving long-term visual odometry for indoor navigation. 2012 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2012 - Conference Proceedings.

[9] Forstner, W. (1987). A feature based correspondence algorithm for image matching. From Analytical to Digital. Proc. Symposium, ISPRS, Commission III, Rovaniemi, 1986, Vol. 3, 150–166.

[10] Harris, C. G., Pike, J. M., Research, P., & Manor, R. (n.d.). 3D POSITIONAL INTEGRATION FROM IMAGE SEQUENCES.

[11] Shi, J., & Tomasi, C. (1993). Good Features to Track Introduction.

[12] Trajković, M., & Hedley, M. (1998). Fast corner detection. Image and visión Computing, 16(2), 75–87.

[13] Lowe, D. G. (1999). Object recognition from local scale-invariant features. Proceedings of the IEEE International Conference on Computer visión, 2, 1150–1157.

[14] Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2007). Speeded-Up Robust Features (SURF).

[15] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (n.d.). ORB: an efficient alternative to SIFT or SURF.

[16] Zhang, Z. (2000). A flexible new technique for camera calibration.

[17] Tomasi, C. & Kanade, T. (1991). Detection and Tracking of Point Features.

[18] Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). visión meets robotics: The KITTI dataset. International Journal of Robotics Research, 32(11), 1231–1237.