Revista Iberoamericana de Automática e Informática Industrial

Revista Iberoamericana de Automática e Informática Industrial 19 (2022) 144-153

www.revista-riai.org

WGANVO: odometría visual monocular basada en redes adversarias generativas

Javier Cremona *, Lucas Uzal, Taihú Pire

CIFASIS, Centro Internacional Franco-Argentino de Ciencias de la Información y de Sistemas (CONICET-UNR), Bv. 27 de Febrero 210 bis - S2000EZP Rosario, Argentina.

To cite this article: Cremona, J., Uzal, L., Pire, T. 2022. WGANVO: monocular visual odometry based on generative adversarial networks. Revista Iberoamericana de Automática e Informática Industrial 19, 144-153 https://doi.org/10.4995/riai.2022.16113

Resumen

Los sistemas tradicionales de odometría visual (VO), directos o basados en características visuales, son susceptibles de cometer errores de correspondencia entre imágenes. Además, las configuraciones monoculares sólo son capaces de estimar la localización sujeto a un factor de escala, lo que hace imposible su uso inmediato en aplicaciones de robótica o realidad virtual. Recientemente, varios problemas de Visión por Computadora han sido abordados con éxito por algoritmos de Aprendizaje Profundo. En este trabajo presentamos un sistema de odometría visual monocular basado en Aprendizaje Profundo llamado WGANVO. Específicamente, entrenamos una red neuronal basada en GAN para regresionar una estimación de movimiento. El modelo resultante recibe un par de imágenes y estima el movimiento relativo entre ellas. Entrenamos la red neuronal utilizando un enfoque semi-supervisado. A diferencia de los sistemas monoculares tradicionales basados en geometría, nuestro método basado en Deep Learning es capaz de estimar la escala absoluta de la escena sin información extra ni conocimiento previo. Evaluamos WGANVO en el conocido conjunto de datos KITTI. Demostramos que nuestro sistema funciona en tiempo real y la precisión obtenida alienta a seguir desarrollando sistemas de localización basados en Aprendizaje Profundo.

Palabras clave: Localización, Redes neuronales, Robots móviles.

WGANVO: monocular visual odometry based on generative adversarial networks

Abstract

Traditional Visual Odometry (VO) systems, direct or feature-based, are susceptible to matching errors between images. Furthermore, monocular configurations are only capable of estimating localization up to a scale factor, making impossible to use them out-of-the-box in robotics or virtual reality application. Recently, several Computer Vision problems have been successfully tackled by Deep Learning algorithms. In this paper we introduce a Deep Learning-based monocular Visual Odometry system called WGANVO. Specifically, we train a GAN-based neural network to regress a motion estimate. The resulting model receives a pair of images and estimates the relative motion between them. We train the neural network using a semi-supervised approach. In contrast to traditional geometry-based monocular systems, our Deep Learning-based method is able to estimate the absolute scale of the scene without extra information and prior knowledge. We evaluate WGANVO on the well-known KITTI dataset. We show that our system works in real time and the accuracy obtained encourages further development of Deep Learning-based localization systems.

Keywords: Localization, Neural networks, Mobile robots.

*Autor para correspondencia: cremona@cifasis-conicet.gov.ar

1. Introducción

Dentro del campo de la Robótica Móvil, la navegación autónoma es uno de los tópicos más estudiados (Thrun et al., 2005; Siciliano and Khatib, 2016). El concepto de *Odometría* hace referencia a la estimación incremental de la trayectoria que realiza un robot. En dicha estimación, intervienen múltiples mediciones, las cuales son obtenidas a través de los sensores del robot. Una amplia variedad de sensores han sido empleados, entre los que se destacan láseres, encoders, unidades de medición inercial (IMU) y cámaras. La portabilidad, el bajo costo y la información que aportan hacen de las cámaras un tipo de sensor propicio para utilizar en el cálculo de la odometría. En tal caso, las técnicas desarrolladas se clasifican dentro del ámbito de investigación denominado *Visual Odometry* (VO) (Nistér et al., 2004; Scaramuzza and Fraundorfer, 2011). VO ha sido considerado objeto de estudio en los últimos tiempos, logrando importantes avances gracias al desarrollo de nuevas técnicas en el campo de *Computer Vision* (Nistér et al., 2004; Comport et al., 2010; Krombach et al., 2016; Wang et al., 2017; Li et al., 2018). Tradicionalmente los métodos de VO estiman el movimiento de un robot basándose en la geometría (Engel et al., 2018; Forster et al., 2014; Pire et al., 2017).

En la última década, se han aplicado con éxito diferentes tipos de *Convolutional Neural Networks* (CNN) para resolver múltiples problemas de Computer Vision (LeCun et al., 2015), por ejemplo, detectar y clasificar objetos, segmentación semántica y extraer patrones visuales (Salimans et al., 2016; Krizhevsky et al., 2012). En particular, las *Generative Adversarial Networks* (GAN) (Goodfellow et al., 2014) también han recibido una atención considerable recientemente, ya que sacan provecho de las CNNs profundas para realizar aprendizaje no supervisado (Engel et al., 2019; Yi et al., 2019; Tulyakov et al., 2018; Karras et al., 2019). GAN consiste en entrenar en simultáneo dos CNNs. Una de ellas, el Generador, produce imágenes sintéticas similares a las del conjunto de datos, mientras que, al mismo tiempo, otra CNN, denominada Discriminador, distingue muestras reales de las generadas por el Generador. El Discriminador captura patrones de alto nivel sin necesidad de recibir muestras etiquetadas. Esto le permite ser reutilizado de forma semi-supervisada, aplicado a la resolución de diversas tareas, como la clasificación multiclase. (Salimans et al., 2016; Radford et al., 2015).

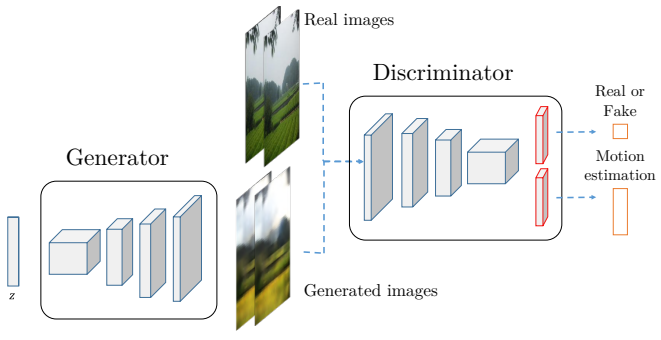

Figura 1: Arquitectura general del método propuesto. El Discriminador recibe imágenes del Generador e imágenes reales como entrada durante la fase de entrenamiento. Además, adaptamos su salida para resolver el problema de VO.

En los últimos años, las CNNs se aplicaron al problema de VO implementando un enfoque conocido como *end-to-end* (Wang et al., 2017; Tateno et al., 2017; Agrawal et al., 2015). Mediante este enfoque, una CNN resuelve una tarea de principio a fin, en lugar de dividirla en diferentes etapas. Tal división ocurre comúnmente en los métodos de VO tradicionales (Mur-Artal et al., 2015; Engel et al., 2018), que suelen estar compuestos por diferentes etapas como la calibración de la cámara, la detección y *matching* de *features* visuales, la estimación del movimiento y la optimización local.

En este trabajo presentamos WGANVO, una CNN basada en GAN que se entrena para resolver el problema de VO de manera integral. En concreto, nuestra red se basa en una variante de GAN conocida como DCGAN (Radford et al., 2015), en la que el entrenamiento adversario minimiza la función de costo de WGAN-GP (Gulrajani et al., 2017). El Discriminador propuesto, luego del entrenamiento adversario, hace una regresión del movimiento relativo entre un par de imágenes que recibe como entrada. Este enfoque, denominado entrenamiento semi-supervisado, combina la técnica no supervisada de GAN con datos etiquetados. El entrenamiento adversario hace que el Discriminador capture toda la información posible sobre las imágenes más allá de la señal supervisada retropropagada del error de predicción. Una visión general de la red propuesta se puede ver en la Fig. 1. Las evaluaciones en el conjunto de datos públicos KITTI (Geiger et al., 2013) muestran que el método propuesto funciona en tiempo real y con resultados prometedores. Comparamos estos resultados con algoritmos de VO existentes en el estado del arte. El código fuente se libera para el beneficio de la comunidad de Robótica y *Deep Learning* (Cremona et al., 2021).

Las restantes secciones de este documento se organizan de la siguiente forma: la Sección 2 revisa los trabajos relacionados. Describimos nuestro método en la Sección 3. Además, se introduce el enfoque semi-supervisado propuesto junto con una función de costo para la parte supervisada. Los resultados experimentales del método propuesto se presentan en la Sección 4. Estos resultados incluyen imágenes generadas por nuestro Generador, una comparación con un método del estado del arte y diferentes tipos de entrenamiento. Cerramos el artículo con las conclusiones y una perspectiva de trabajo futuro en la Sección 5.

2. Trabajo Relacionado

La mayoría de los algoritmos del estado del arte son métodos basados en geometría. Suelen dividirse en métodos basados en *features* visuales (Mur-Artal et al., 2015; Mur-Artal and Tardós, 2017; Pire et al., 2017) y métodos directos (Engel et al., 2018; Forster et al., 2014; Engel et al., 2014). Mientras que los métodos basados en *features* visuales extraen y emparejan puntos salientes de la imagen para estimar el movimiento, los métodos directos utilizan las intensidades de los píxeles. En los últimos años, las técnicas de Deep Learning se han aplicado de diferentes maneras para resolver el problema de VO. Una CNN se puede utilizar para resolver un único módulo del *pipeline* tal como lo hace DVSO (*Deep Virtual Stereo Odometry*) (Yang et al., 2018). En concreto, DVSO está fuertemente basado en DSO (Engel et al., 2018), un método directo que opera sobre

las intensidades de los píxeles minimizando el error fotométrico junto con una optimización conjunta de todos los parámetros del modelo. DVSO implementa una CNN para predecir mapas de profundidad de la escena. Otro ejemplo es VOLDOR (Min et al., 2020), el cual emplea CNNs para estimar el flujo óptico. Particularmente, sus autores implementan una estimación conjunta del movimiento de la cámara y la profundidad en un *pipeline* que no requiere extracción de *features*, ni *bundle adjustment*. Contrariamente al método propuesto en este trabajo, VOLDOR no opera de forma online.

Otro enfoque empleado consiste en abordar el problema de VO mediante técnicas de Deep Learning de manera integral. Este enfoque se conoce como *end-to-end*. En contraste con los métodos tradicionales, no se necesita implementar y ajustar cada módulo en particular (lo cual suele ser menester para garantizar una correcto rendimiento). Además, los métodos monoculares tradicionales no son capaces de estimar la escala absoluta de la escena observada sin utilizar algún conocimiento previo o información extra. Contrariamente, los métodos monoculares que aplican técnicas de Deep Learning son capaces de aprender tamaños y distancias del mundo real durante la etapa de entrenamiento.

DeepVO (Wang et al., 2017) aborda el problema de manera integral. Sus autores proponen entrenar una *Recurrent Neural Network* (RNN). La parte convolucional de esta RNN se ocupa de capturar diferentes abstracciones de alto nivel de los datos de entrenamiento. Al mismo tiempo, el movimiento de la cámara es estimado por la parte recurrente de la RNN a partir de secuencias de imágenes. Las RNNs consiguen un buen rendimiento cuando capturan información de una secuencia ya que disponen de información de movimientos previos en el momento de analizar un par de fotogramas consecutivos. En contraste con DeepVO, que entrena de forma puramente supervisada, nuestro trabajo utiliza GAN, lo que permite un esquema de entrenamiento semi-supervisado.

UnDeepVO (Li et al., 2018) es otro ejemplo de sistema *end-to-end*. El modelo propuesto consta de dos CNNs. Dados dos fotogramas consecutivos, una de ellas estima el movimiento de la cámara. A su vez, la otra CNN, a partir de la entrada, produce mapas de profundidad. La función de costo a minimizar permite entrenar a UnDeepVO en un esquema no supervisado. Este sistema, si bien es monocular, requiere imágenes estéreo durante el entrenamiento para calcular los mapas de profundidad. El método propuesto en este paper requiere únicamente imágenes monoculares, haciendo más versátil y simple su uso; y simplificando la implementación.

GANVO (Almalioglu et al., 2019) es un sistema de VO no supervisado, que resulta de combinar la metodología de GAN con DeepVO y UnDeepVO. La estimación de mapas de profundidad es llevada a cabo por una CNN conocida como Generador, a la vez que una RNN toma un par de imágenes como entrada y estima la pose. Combinando el movimiento estimado con el mapa de profundidad, y dado uno de los fotogramas del par de entrada, se genera un fotograma sintético. Durante el entrenamiento, se busca generar un fotograma sintético lo más similar posible al otro fotograma del par. Por último, el Discriminador recibe pares de fotogramas, tanto imágenes reales como sintéticas. Un valor escalar probabilístico, resultado de correr el Discriminador, indica si el par dado como entrada es real o sintético. Gracias al entrenamiento adversario, la RNN y el Generador se vuelven más eficaces en la realización de sus respectivas tareas. A diferencia de GANVO, nuestro sistema implementa GAN mediante un enfoque directo (Goodfellow et al., 2014; Gulrajani et al., 2017), ya que nuestro Generador no necesita fotogramas como entrada, sino que tiene la capacidad de aprender la distribución del conjunto de datos a partir del feedback del Discriminador. Además, convertimos al mismo Discriminador en un regresor de la pose, mientras que GANVO implementa una CNN adicional para resolver dicha tarea. De forma similar, en (Li et al., 2019) se presenta un sistema de VO basado en GAN y RNNs, cuyo Generador está compuesto por un conjunto de CNNs. Por el contrario, WGANVO presenta una arquitectura simple y minimalista, que junto a la liberación de su código fuente, facilita la aplicación y extensión del sistema.

GeoNet (Yin and Shi, 2018) estima la pose, el flujo óptico y la profundidad, implementando CNNs para resolver cada una de dichas subtareas. El esquema de trabajo es dividido entre la localización de objetos dinámicos de la escena y la parte estática de la misma. Los autores reportan buenos resultados para cada una de las CNNs en las respectivas tareas. PoseNet (Kendall et al., 2015) aborda el problema de *Localización Global*. En este caso, el problema consiste en calcular la localización de la cámara en un mapa conocido, a partir de una imagen. PoseNet implementa una CNN, cuya arquitectura está compuesta por más de 20 capas convolucionales. Este sistema mostró una gran precisión en tiempo real, tanto en ambientes abiertos como cerrados, y en diversas condiciones climáticas.

Respecto a los sistemas mencionados anteriormente en esta sección, se detectó la falta de código públicamente disponible. Esto motivó a liberar el código fuente de WGANVO con el fin de ofrecer a la comunidad un sistema de VO para su uso e investigación a la vez que permite validar los resultados obtenidos en el presente trabajo.

3. Método Propuesto

En esta sección se presenta WGANVO, el método *end-to-end* propuesto para la resolución del problema de VO utilizando CNNs basadas en GAN. A partir de pares de fotogramas consecutivos, WGANVO retorna el movimiento relativo entre ellos.

La arquitectura propuesta se basa en una GAN conocida como DCGAN (Radford et al., 2015) y se muestra en la Fig. 2. La entrada del Generador es un vector aleatorio ($\mathbf{z}$) muestreado de una distribución normal. Dada esta entrada, se aplican cuatro capas convolucionales transpuestas y, finalmente, se devuelven dos imágenes sintéticas. Por su parte, al Discriminador se le pasa un par de imágenes como entrada. Los datos de entrada provienen de dos fuentes: pares de imágenes sintéticas producidos por el Generador y pares de fotogramas de una cámara monocular de un conjunto de datos reales (el proceso de entrenamiento se detalla en la Sección 3.1). Posteriormente, mediante cinco capas convolucionales se extraen características visuales de las imágenes recibidas como entrada. A continuación, el Discriminador se divide en dos bloques, lo que da lugar a dos salidas diferentes. Por un lado, hay una capa lineal que retorna un valor escalar probabilístico que indica si la entrada pertenece al conjunto de datos reales (en realidad, no es estrictamente

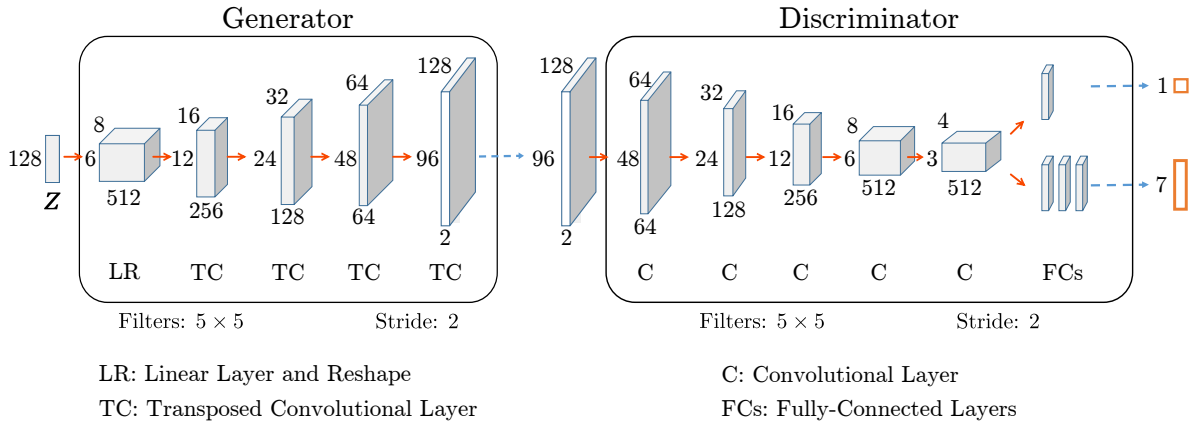

Figura 2: Arquitectura de WGANVO basada en DCGAN (Radford et al., 2015). La entrada del Discriminador es un par de fotogramas y su salida está formada por un valor escalar que se utiliza para distinguir las imágenes sintéticas de las reales y un vector de 7 componentes que representan el movimiento relativo entre los fotogramas de entrada. Este vector está compuesto por un cuaternión unitario y un vector de traslación. El cuaternión se normaliza en la capa de salida. En comparación con la formulación original del DCGAN, se modificaron las dimensiones de las capas, y en el Discriminador se añadió una quinta capa convolucional y capas completamente conectadas para la salida de VO.

una probabilidad, pero puede considerarse un valor que se utiliza para distinguir las imágenes sintéticas de las reales). Por otro lado, hay tres capas completamente conectadas, cuya salida final es el movimiento relativo entre los dos fotogramas de entrada. En lugar de representar este movimiento mediante una matriz SE(3), optamos por utilizar un vector traslacional y un cuaternión unitario para la parte rotacional (Kendall and Cipolla, 2017). Los cuaterniones se usan comúnmente en problemas de optimización, debido a que permiten representar una rotación con 4 valores frente a los 9 valores de una matriz de rotación de SO(3). Adicionalmente, es más sencillo restringir al cuaternión para que tenga norma unitaria que asegurar la ortogonalidad de un elemento de SO(3). La última capa de salida es la encargada de normalizar el cuaternión.

3.1. *Entrenamiento*

Inicialmente, el entrenamiento adversario se realiza utilizando el esquema propuesto por WGAN-GP (Gulrajani et al., 2017). La principal novedad de WGAN-GP es la introducción de un término de penalización de gradiente en la función de costo que hace que el entrenamiento sea estable. Como se ha mencionado anteriormente, la entrada del Generador es un vector compuesto de 128 valores, muestreado a partir de una distribución normal. Luego, la salida de dicho Generador es un par de fotogramas consecutivos sintéticos. El Discriminador se alimenta con pares de imágenes generados por el Generador y pares de imágenes de un conjunto de datos real, tal como ocurre usualmente al entrenar una CNN tradicional basada en GAN. A partir de esta parte del entrenamiento, se busca que el Discriminador reconozca patrones y aprenda abstracciones de alto nivel, al entrenarlo para distinguir los fotogramas reales de los sintéticos. Cuando se realiza el entrenamiento adversario, se ignora la salida de la estimación de movimiento del Discriminador.

Tras el entrenamiento adversario, el Discriminador se entrena únicamente con pares de imágenes del conjunto de datos real de forma supervisada para estimar el movimiento de la cámara. Los valores de los pesos del Discriminador parten desde donde finalizaron en el entrenamiento adversario. Las muestras de entrenamiento etiquetadas son pares de la forma $((I_i, I_{i+1}), (\mathbf{x}_{i(i+1)}, \mathbf{q}_{i(i+1)}))$, donde I_i, I_{i+1} son fotogramas consecutivos de una secuencia de video, $\mathbf{x}_{i(i+1)}$ es un vector de traslación en $\mathbb{R}^3$ entre los tiempos en que I_i y I_{i+1} fueron capturados, y $\mathbf{q}_{i(i+1)}$ es un cuaternión unitario que representa el cambio de orientación correspondiente. Debido a que las poses del *ground-truth* suelen ser almacenadas como matrices de SE (3), durante la carga de los datos etiquetados la parte rotacional de cada transformación $\mathbf{T}_{i(i+1)}$ se convierte al correspondiente cuaternión unitario $\mathbf{q}_{i(i+1)}$, mientras que la parte traslacional de $\mathbf{T}_{i(i+1)}$ resulta en $\mathbf{x}_{i(i+1)}$. Finalmente, la salida del Discriminador durante el entrenamiento de VO se compone de la estimación de la traslación $\hat{\mathbf{x}}_{i(i+1)}$, y la correspondiente rotación $\hat{\mathbf{q}}_{i(i+1)}$. A su vez, se ignora la salida escalar probabilística. La función de costo para el entrenamiento de VO se compone de dos términos: una parte traslacional y otra rotacional. La parte traslacional se define como:

$$L_{\mathbf{x}}(I_m, I_n) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2 , \tag{1}$$

done $\mathbf{x}$ es la traslación asociada a (I_m, I_n) y $\hat{\mathbf{x}}$ es la traslación estimada por el Discriminador cuando la entrada es el par (I_m, I_n). De forma similar, la parte rotacional se define como:

$$L_{\mathbf{q}}(I_m, I_n) = \|\mathbf{q} - \hat{\mathbf{q}}\|_2 , \tag{2}$$

donde $\mathbf{q}$ es la rotación asociada a (I_m, I_n) y $\hat{\mathbf{q}}$ es la rotación estimada por el Discriminador cuando la entrada es (I_m, I_n). Por último, en base al trabajo de Kendall et al. (Kendall and Cipolla, 2017), se define la siguiente función de costo:

$$L_{\beta}(I_m, I_n) = L_{\mathbf{x}}(I_m, I_n) + \beta L_{\mathbf{q}}(I_m, I_n) , \tag{3}$$

donde β es un hiperparámetero que balancea el error rotacional y el error traslacional, debido a que suelen estar en diferentes escalas. Una vez finalizado el proceso de entrenamiento, el Discriminador resulta en un modelo que recibe dos fotogramas como entrada y devuelve la estimación del movimiento relativo entre esos fotogramas, tal como lo hace cualquier sistema tradicional de VO.

4. Experimentos

En esta sección se muestran distintos experimentos y sus resultados. Comenzamos esta sección presentando el conjunto

de datos KITTI (Geiger et al., 2013), que se utiliza para entrenar nuestra GAN. Después, realizamos experimentos con el Generador y el Discriminador para mostrar el rendimiento del método propuesto. Por último, comparamos nuestros resultados con ORB-SLAM2 (Mur-Artal and Tardós, 2017), un conocido método del estado del arte basado en *features* visuales.

4.1. Dataset

El conjunto de datos utilizado en este trabajo es el dataset público KITTI (Geiger et al., 2013). Este conjunto de datos contiene un total de 11 secuencias de imágenes con trayectorias reales (normalmente numeradas de 00 a 10) grabadas con una cámara estéreo a 10 fps, situada en la parte frontal de un coche. El número total de imágenes es de aproximadamente 23 000 y han sido rectificadas y des-distorsionadas. Adicionalmente, se proporcionan las poses reales de las secuencias. Las mismas están expresadas en forma de transformaciones $\mathbf{T}_{wk} \in \mathrm{SE}(3)$.

Las imágenes se preprocesaron de forma *offline*. Aunque las imágenes se obtuvieron con una cámara estéreo, sólo tomamos las correspondientes a la cámara izquierda para trabajar de forma monocular. La resolución de las imágenes, originalmente de 1241×376 px, se redujo a 128×96 px. Esto se logró recortando la región central de la imagen a una resolución de 500×375 px. Posteriormente, esta región recortada se escaló al tamaño final. El principal motivo de esta reducción es acortar los tiempos de cálculo. 128 px y 96 px son tamaños que se suelen utilizar como entrada a las CNNs y permiten acelerar el proceso de entrenamiento, a la vez que mantienen suficiente detalle de la imagen. Tras el preprocesamiento de las imágenes, éstas se almacenaron en pares de fotogramas consecutivos junto con la transformación relativa (desplazamiento traslacional y rotacional) entre ambas imágenes del par.

4.1.1. Dataset aumentado

Disponer de una gran cantidad de datos es una de las claves para entrenar una CNN. Para disponer de más imágenes, generamos nuevos fotogramas a partir del conjunto de datos KITTI. En el presente trabajo espejamos todas las imágenes de las secuencias $\{00, 01, \ldots 10\}$ de KITTI con el objetivo de generar nuevos fotogramas y así contar con más datos etiquetados. Luego, dichos fotogramas se preprocesaron para reducir su resolución a 128×96 px. Por último, fue necesario generar las transformaciones relativas que sirven como etiquetas de los pares de imágenes. Las mismas pueden obtenerse a partir de las etiquetas de las imágenes originales. Para cada nuevo par de imágenes consecutivas $\left(I_i^*, I_{i+1}^*\right)$ espejadas a partir de (I_i, I_{i+1}) con transformación relativa

$$\mathbf{T}_{i(i+1)} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{4}$$

la transformación relativa entre las imágenes espejadas $\mathbf{T}^*_{i(i+1)}$ se define como:

$$\mathbf{T}^*_{i(i+1)} = \begin{bmatrix} \mathbf{R}^* & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{5}$$

donde $\mathbf{R}^*$ se define como:

$$\mathbf{R}^* = \mathbf{MRM}, \tag{6}$$

y la matriz $\mathbf{M}$ se define como:

$$\mathbf{M} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{7}$$

4.2. Imágenes Generadas

En el presente experimento se propone analizar las muestras producidas por el Generador. La salida del Generador refleja si efectivamente esta red está aprendiendo patrones visuales sobre las muestras reales del dataset. Realizamos entonces un entrenamiento adversario entre el Generador y el Discriminador durante 20 000 iteraciones. En este caso, no se entrena al Discriminador para la regresión del movimiento relativo. Las secuencias $\{00, 01, \ldots, 10\}$ se eligieron para el entrenamiento. El *batch* empleado es de 100 pares de fotogramas. Además, se utilizó el optimizador Adam con un valor de 10^{-4} para el *learning rate*, y la función de costo L_{beta} con el parámetro β=100.

Al final del entrenamiento, se observa que la red genera pares de imágenes consecutivas muy similares a las reales. En la Fig. 3 se muestran distintos pares de imágenes generados por la red. En cada par es posible visualizar un ligero movimiento entre ambas imágenes, como si se tratara de un par de fotogramas consecutivos obtenidos del conjunto de datos. Se observan características de alto nivel tales como una ruta o camino en el centro de la imagen y árboles a ambos lados. Además, el cielo se distingue fácilmente y los caminos se dirigen hacia el horizonte.

Figura 3: Pares de frames sintéticos, resultado de la salida de nuestro Generador. Pueden observarse caminos en el centro de la escena, y un ligero desplazamiento entre imágenes del par que se corresponde con el movimiento de la cámara.

4.3. Entrenamiento semi-supervisado

En la presente subsección se estudia la influencia del entrenamiento semi-supervisado propuesto para entrenar nuestro sistema. Para ello, se realizan tres tipos diferentes de entrenamiento. El primero consiste en llevar a cabo el entrenamiento semi-supervisado propuesto en la Sección 3.1. Este tipo de entrenamiento implica realizar el entrenamiento adversario tradicional de WGAN-GP y, una vez finalizado el mismo, llevar a cabo una última etapa de entrenamiento que realice la regresión del movimiento sobre el Discriminador. Referimos al lector a la Sección 3.1 para más detalle. Para tal caso, el Discriminador es alimentado con datos de entrada de dos fuentes de información: imágenes sintéticas del Generador e imágenes reales del

conjunto de datos KITTI. En segundo lugar, se propone entrenar una CNN con una arquitectura exactamente igual a la del Discriminador, salvo que en este caso se elimina la salida probabilística escalar. Esta red se entrena para resolver únicamente la regresión del movimiento de la cámara, sin tener en cuenta al Generador y, por ende, sin llevar a cabo el entrenamiento adversario tradicional de una GAN. Por esto mismo, la entrada del Discriminador sólo consiste en imágenes reales. Por último, el tercer experimento consiste en realizar en simultáneo y en la misma iteración, el entrenamiento adversario de WGAN-GP y la regresión del movimiento sobre el Discriminador.

La métrica de error de KITTI (Geiger et al., 2012) se utiliza para comparar cuánto difiere la trayectoria estimada de la trayectoria real. En base a operar con las transformaciones relativas SE(3), se calculan errores de rotación y traslación por separado para varias sub-secuencias de longitudes de $\{100, 200, \ldots, 800\}$ metros. Luego, cada uno de estos errores de rotación y traslación se promedian por la longitud correspondiente. Finalmente, estos errores normalizados se promedian por el número total de subsecuencias. Referimos al lector al paper original de KITTI (Geiger et al., 2012) para ver en detalle las expresiones para calcular dichos errores. De este modo, se analizan por separado el error de traslación y el de rotación. Los datos etiquetados empleados para entrenar fueron las secuencias $\{00, 01, \ldots, 10\}$ y sus correspondientes secuencias espejadas. Todas las secuencias se utilizan para el entrenamiento, excepto una secuencia que se deja fuera de las muestras de entrenamiento (junto a la secuencia espejada correspondiente) y se usa para testear. Por ejemplo, el modelo se entrena con las secuencias $\{01, 02, \ldots, 10\}$ y sus correspondientes secuencias espejadas, y se testea con la secuencia 00. Este proceso se repite para cada una de las secuencias $\{00, 01, \ldots, 10\}$. Respecto al número de iteraciones, en el primer caso se entrenó la parte supervisada durante 10 000 iteraciones y la regresión durante 40 000 iteraciones. Para el segundo y tercer caso, el modelo se entrenó durante 50 000 iteraciones. El *batch* empleado en cada ejecución es de 100 pares de fotogramas. Nuevamente utilizamos el optimizador Adam con un valor de 10^{-4} para el *learning rate*, y la función de costo L_{beta} con el parámetro β=100.

La Tabla 1 detalla los resultados obtenidos en este experimento según la métrica de error de KITTI. El entrenamiento semi-supervisado propuesto en general mejora los resultados en la estimación del movimiento. Los errores reportados en la secuencia 01 son claramente superiores a los del resto de las secuencias. Esta secuencia se distingue por ser la única en la que la velocidad del automóvil supera los $60\,\mathrm{km\,h^{-1}}$. Por esto mismo, es un reto para nuestro sistema estimar la trayectoria con precisión. Los datos de entrenamiento no incluían movimientos a velocidades similares a las de la secuencia 01, y por esta razón, la precisión de WGANVO se ve afectada. Además, la secuencia tiene lugar en una autopista, en la que los elementos distintivos de la escena están lejos de la cámara. Por último, para que la red captara una mayor variedad de movimientos relativos entre fotogramas, se añadieron a las muestras de entrenamiento secuencias de KITTI en orden inverso, pero esto no mejoró los resultados obtenidos anteriormente.

Tabla 1: Comparación entre diferentes tipos de entrenamiento. WGANVO hace referencia al entrenamiento propuesto en la Sección 3.1. Only-VO representa el entrenamiento que ignora el entrenamiento adversario. Simult. GAN+VO representa la minimización simultánea del error (GAN y VO). Se utiliza la métrica de error de KITTI (Geiger et al., 2012), y t_{rel} y r_{rel} se corresponden con los errores traslacional y rotacional promediados sobre múltiples intervalos de 100 a 800 metros. En la presente tabla, aquellos valores que representan el menor error se indican en **negrita**

	WGANVO		Only-VO		Simult. GAN+VO	
Secuencia	t_{rel} (%)	r_{rel} (°/100m)	t_{rel} (%)	r_{rel} (°/100m)	t_{rel} (%)	r_{rel} (°/100m)
00	**10.54**	3.22	24.04	4.84	12.03	**3.03**
01	**24.94**	3.54	26.34	4.07	45.29	**3.21**
02	18.28	**2.74**	**14.50**	3.89	32.70	4.45
03	8.96	5.70	**7.68**	**3.14**	10.89	3.97
04	**14.14**	3.24	14.81	3.67	17.04	**2.54**
05	**7.01**	3.85	9.18	**2.51**	9.22	2.64
06	**7.87**	**2.19**	11.26	2.97	30.16	6.68
07	7.71	3.79	**6.52**	**3.74**	18.10	4.78
08	**9.04**	3.85	9.94	**3.59**	15.14	3.81
09	**10.49**	4.91	13.65	**2.85**	36.46	4.83
10	**10.89**	**5.03**	12.91	5.54	17.44	6.64

4.4. Comparación con ORB-SLAM2

ORB-SLAM2 (Mur-Artal and Tardós, 2017) es un sistema de SLAM del estado del arte basado en *features* visuales para cámaras monoculares y estéreo. Funciona en tiempo real en una gran variedad de entornos. Para llevar a cabo esta comparación y que sea más justa, modificamos el código fuente de ORB-SLAM2. Tanto el cierre de ciclos como la optimización global fueron desactivados. De este modo, ORB-SLAM2 se comporta prácticamente como un sistema de VO. A continuación, ejecutamos ORB-SLAM2 monocular y estéreo sobre las secuencias de KITTI $\{00, 01, \ldots, 10\}$ con su resolución original.

Tabla 2: Comparación entre WGANVO y ORB-SLAM2 en sus versiones monocular y estéreo. Para ORB-SLAM2 el cierre de ciclos y la optimización global fueron desactivados. Se utiliza la métrica de error de KITTI, y t_{rel} y r_{rel} se corresponden con los errores traslacional y rotacional promediados sobre múltiples intervalos de longitudes de 100 a 800 metros. En la presente tabla, aquellos valores que representan el menor error se indican en **negrita**.

	WGANVO		ORB-SLAM2 (Mono)		ORB-SLAM2 (Stereo)	
Secuencia	t_{rel} (%)	r_{rel} (°/100m)	t_{rel} (%)	r_{rel} (°/100m)	t_{rel} (%)	r_{rel} (°/100m)
00	10.54	3.22	23.01	**0.30**	**0.88**	**0.30**
01	24.94	3.54	-	-	**1.39**	**0.23**
02	18.28	2.74	6.63	**0.24**	**0.81**	0.28
03	8.96	5.70	1.12	0.19	**0.73**	**0.17**
04	14.14	3.24	0.70	0.22	**0.48**	**0.15**
05	7.01	3.85	12.34	**0.22**	**0.61**	0.25
06	7.87	2.19	17.71	0.27	**0.76**	**0.23**
07	7.71	3.79	11.10	**0.36**	**0.88**	0.47
08	9.04	3.85	12.69	**0.30**	**1.05**	0.32
09	10.49	4.91	-	-	**0.83**	**0.26**
10	10.89	5.03	3.90	0.30	**0.57**	**0.26**

La Tabla 2 compara nuestro método con los resultados obtenidos con ORB-SLAM2. WGANVO fue entrenado con el enfoque semi-supervisado presentado en la Sección 3.1, y se testeó sobre cada una de las secuencias del conjunto $\{00, 01, \ldots, 10\}$ como se explica en la Sección 4.3. La trayectoria se alineó y escaló con el *ground-truth* para el caso monocular. Para esto

se utilizó el método Sim(3) Umeyama (Umeyama, 1991). Se puede observar que aunque ORB-SLAM2 estéreo presenta una gran precisión sobre KITTI, la performance disminuye para la versión monocular. Para las secuencias 01 y 09, no se reporta el error, debido a que ORB-SLAM2 presenta problemas durante la estimación. Como anteriormente se mencionó, la secuencia 01 es particularmente desafiante, ya que se grabó en una autopista, a una alta velocidad y los puntos salientes están lejos de la cámara. Fig. 4 muestra las diferentes trayectorias obtenidas en este experimento. Acerca del hardware utilizado en este experimento, se ejecutaron tanto WGANVO como ORB-SLAM2 monocular y estéreo en un ordenador con Intel Core i7-3770, 12 GB RAM y una placa de video GeForce GTX 770 GPU con 2 GB RAM. Adicionalmente, Fig. 5 muestra tres boxplots con los valores (en milisegundos) que les tomó a los sistemas procesar cada fotograma de las secuencias $\{00, 01, \ldots, 10\}$.

Si bien WGANVO tiene un rendimiento inferior al de ORB-SLAM2 estéreo, esto era de esperarse, puesto que los sistemas del estado del arte basados en enfoques geométricos suelen ser robustos y precisos, ya que son el producto de muchos años de investigación. Por contraposición, las técnicas de Deep Learning recientemente han abordado el problema de VO y no han madurado lo suficiente. A pesar de esto, podemos afirmar que la precisión de nuestro sistema y la de ORB-SLAM2 monocular son comparables para varias de las secuencias. Además, WGANVO no es afectado por el problema de no poder recuperar la escala absoluta de la escena observada. A diferencia de los métodos monoculares basados en *features* visuales, WGANVO es capaz de extraer información relacionada con la escala durante el proceso de entrenamiento. En términos de tiempos de ejecución, en promedio WGANVO procesa imágenes a una tasa de 50 fps. Dado que el conjunto de datos de KITTI se grabó a 10 fps, nuestro sistema puede procesar imágenes obtenidas de una cámara similar sin ningún inconveniente.

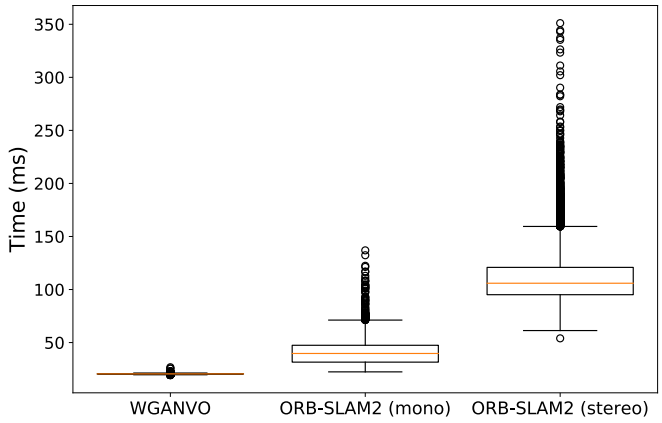

Figura 5: Tiempo de procesamiento de cada fotograma de las secuencias KITTI $\{00, 01, \ldots, 10\}$.

4.5. Comparación con DeepVO

En relación a otros sistemas que apliquen CNNs para resolver el problema de VO, resulta de interés una comparación con otro sistema *end-to-end* monocular que estime la pose de forma online, tal como es el caso de DeepVO (Wang et al., 2017). Este es un sistema que emplea RNNs y un enfoque supervisado en el entrenamiento. DeepVO es evaluado en KITTI empleando la misma métrica utilizada en este trabajo. En la Tabla 3 se presenta dicha comparación. Debido a que el código de DeepVO no se encuentra públicamente disponible, los resultados aquí presentados son aquellos que fueron reportados en (Wang et al., 2017). Como puede verse, las estimaciones resultan comparables, siendo WGANVO superior al estimar la orientación, mientras que DeepVO obtiene una mejor estimación de la traslación.

Tabla 3: Comparación entre WGANVO y DeepVO. DeepVO se entrenó sobre las secuencias 00, 02, 08 y 09 y se testeó sobre el resto de las secuencias. Los resultados de DeepVO aquí presentados fueron reportados en (Wang et al., 2017). Se utiliza la métrica de error de KITTI, y t_{rel} y r_{rel} se corresponden con los errores traslacional y rotacional promediados sobre múltiples intervalos de longitudes de 100 a 800 metros. En la presente tabla, aquellos valores que representan el menor error se indican en **negrita**.

	WGANVO		DeepVO	
Secuencia	t_{rel} (%)	r_{rel} (°/100m)	t_{rel} (%)	r_{rel} (°/100m)
00	10.54	3.22	-	-
01	24.94	3.54	-	-
02	18.28	2.74	-	-
03	8.96	**5.70**	**8.49**	6.89
04	14.14	**3.24**	**7.19**	6.97
05	7.01	3.85	**2.62**	**3.61**
06	7.87	**2.19**	**5.42**	5.82
07	7.71	**3.79**	**3.91**	4.60
08	9.04	3.85	-	-
09	10.49	4.91	-	-
10	10.89	**5.03**	**8.11**	8.83

4.6. Función de costo alternativa

A continuación, consideraremos realizar la regresión de VO con una función de costo alternativa (Kendall and Cipolla, 2017). La misma está basada en el error de reproyección. Este error es una métrica bien conocida que se utiliza habitualmente en los algoritmos de estimación del movimiento (Pire et al., 2017; Mur-Artal and Tardós, 2017), ya que pondera de forma natural las cantidades de traslación y rotación (Kendall and Cipolla, 2017). El error de reproyección se define entonces como el residuo entre puntos 3D proyectados sobre el plano 2D de la imagen utilizando el *ground-truth* y la pose estimada. En particular, la función de costo $L_p\left(I_m, I_n\right)$ se define como:

$$L_p\left(I_m, I_n\right) = \frac{1}{|G_n|} \sum_{\mathbf{x} \in G_n} \left\| f\left(\mathbf{R}, \mathbf{x}, \mathbf{t}\right) - f\left(\hat{\mathbf{R}}, \mathbf{x}, \hat{\mathbf{t}}\right) \right\|_2, \tag{8}$$

donde la función $f\left(\mathbf{A}, \mathbf{x}, \mathbf{b}\right)$ se define como:

$$f\left(\mathbf{A}, \mathbf{x}, \mathbf{b}\right) = \pi\left(\mathbf{K}\left(\mathbf{A}\,\mathbf{x} + \mathbf{b}\right)\right), \tag{9}$$

y π se define como:

$$\pi\left(\left[a_1 \ldots a_n\right]^\top\right) = \frac{1}{a_n} \begin{bmatrix} a_1 \\ \vdots \\ a_{n-1} \end{bmatrix}, \tag{10}$$

y $\mathbf{R}$ y $\mathbf{t}$ representan la parte rotacional y la parte traslacional, respectivamente, de la transformación relativa entre la imagen I_m y la imagen I_n, $\hat{\mathbf{R}}$ y $\hat{\mathbf{t}}$ son las correspondientes estimaciones rotacional y traslacional de nuestro sistema (la salida del Discriminador), la matriz $\mathbf{K}$ contiene los parámetros de calibración

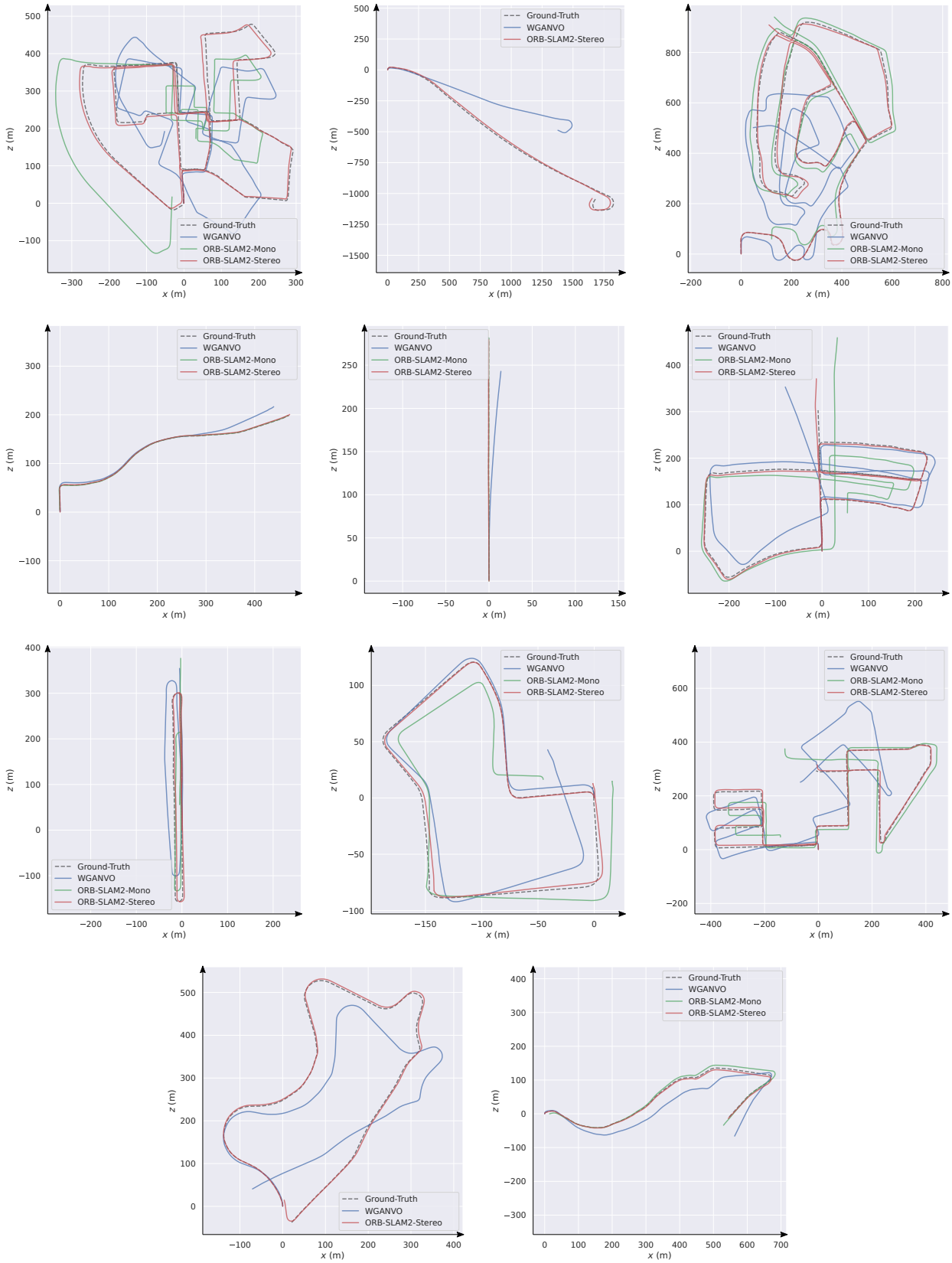

Figura 4: Comparación de las trayectorias estimadas por nuestro sistema (azul) con ORB-SLAM2 Estéreo (rojo), ORB-SLAM2 Monocular (verde) y el *ground-truth*, para las secuencias de KITTI 00 a 10, respectivamente.

de la cámara, y G_n es un conjunto de puntos 3D observables en la imagen I_n que están expresados en el sistema de coordenadas de la cámara.

Los puntos 3D se han obtenido utilizando imágenes estéreo KITTI. Los mismos se extraen en el par de imágenes estéreo empleando el algoritmo SIFT (Lowe, 1999). Luego, se realiza un emparejamiento entre las imágenes del par para encontrar los puntos que corresponden al mismo punto 3D de la escena en cuestión. Por último, usando la calibración, se lleva a cabo una triangulación mediante el algoritmo DLT (Hartley and Zisserman, 2003).

Se realizan dos ejecuciones diferentes. Una de ellas utiliza L_{beta} con $\beta = 100$ y la otra utiliza L_p. En ambas ejecuciones el modelo se entrena tal como se propone en la Sección 3.1, probando sobre cada una de las secuencias del conjunto $\{00, 01, \ldots, 10\}$ como se explica en 4.3. Para la parte no supervisada, el modelo se entrena durante 10 000 iteraciones y para la parte supervisada se entrena durante 40 000 iteraciones. El tamaño de *batch* usado es de 100 pares de fotogramas y el *learning rate* para el optimizador Adam es de 10^{-4} en los dos casos.

La Tabla 4 muestra los resultados obtenidos. Como puede verse, el rendimiento de L_{beta} es ligeramente mejor que el de L_p. A su vez, vale destacar que L_p no requiere validar un hiperparámetro, mientras que L_{beta} requiere de la validación de β. Por su parte, hemos probado L_{beta} donde β tomó valores 1, 100, 100 y 1000 y encontramos que los mejores resultados obtenidos con L_{beta} utilizan un valor $\beta = 100$. Por esto, y por el presente experimento, decidimos aplicar L_{beta} con $\beta = 100$ en todos los experimentos.

Tabla 4: Comparación entre las dos funciones de costo L_β and L_p. Se utiliza la métrica de error de KITTI, y t_{rel} y r_{rel} se corresponden con los errores traslacional y rotacional promediados sobre múltiples intervalos de longitudes de 100 a 800 metros. En la presente tabla, aquellos valores que representan el menor error se indican en **negrita**.

	L_β		L_p	
Secuencia	t_{rel} (%)	r_{rel} (°/100m)	t_{rel} (%)	r_{rel} (°/100m)
00	**10.54**	**3.22**	11.01	4.41
01	24.94	3.54	**23.31**	**3.27**
02	18.28	**2.74**	**16.11**	2.88
03	**8.96**	5.70	9.68	**3.41**
04	**14.14**	**3.24**	14.91	3.71
05	**7.01**	**3.85**	7.18	3.91
06	7.87	**2.19**	**7.56**	2.37
07	**7.71**	3.79	7.82	**3.74**
08	**9.04**	3.85	9.41	**3.19**
09	**10.49**	4.91	10.5	**3.85**
10	**10.89**	**5.03**	10.97	5.41

5. Conclusiones

Este trabajo presenta WGANVO, un método de VO para la estimación del movimiento de la cámara basado en una red GAN. El entrenamiento semi-supervisado de nuestro método se realiza en dos etapas: primero se lleva a cabo el entrenamiento adversario tradicional minimizando la función de costo de WGAN-GP (Gulrajani et al., 2017). Posteriormente, en una segunda etapa se realiza la regresión de VO, minimizando de forma supervisada la función de costo de VO. La función de costo de WGAN-GP se ha empleado en este estudio, ya que es considerablemente estable durante el entrenamiento en comparación con el framework original de GAN (Gulrajani et al., 2017). Como resultado del entrenamiento semi-supervisado, el Discriminador logra capturar patrones visuales de alto nivel de las muestras que recibe como entrada. Durante la fase de entrenamiento, nuestro método es capaz de extraer información de la escala de la escena; un problema frecuente al que se enfrentan la mayoría de los sistemas de VO basados en geometría monocular.

Como trabajo futuro se intentará incorporar los parámetros de calibración de la cámara. Al sumar información de la cámara a la etapa de entrenamiento se busca conseguir un modelo independiente de la cámara utilizada. Esto permitirá el entrenamiento y la evaluación del algoritmo VO propuesto en diferentes conjuntos de datos con parámetros de cámara variables. CAM-Convs (Facil et al., 2019) emplea un enfoque similar, al estimar la profundidad de una cierta escena generando un modelo invariante de la cámara. Esto lo logra concatenando parámetros de calibración de la cámara con mapas convolucionales de la CNN.

Agradecimientos

Este trabajo fue financiado por el CIFASIS, Centro Franco Argentino de Ciencias de la Información y de Sistemas (CONICET-UNR), con el proyecto de Unidad Ejecutora PUE 0015-2016.

Referencias

Agrawal, P., Carreira, J., Malik, J., 2015. Learning to See by Moving. In: Proceedings of the International Conference on Computer Vision. pp. 37–45. DOI: 10.1109/ICCV.2015.13

Almalioglu, Y., Saputra, M. R. U., de Gusmao, P. P. B., Markham, A., Trigoni, N., 2019. GANVO: Unsupervised Deep Monocular Visual Odometry and Depth Estimation with Generative Adversarial Networks. In: Proceedings of the IEEE International Conference on Robotics and Automation. pp. 5474–5480. DOI: 10.1109/ICRA.2019.8793512

Comport, A. I., Malis, E., Rives, P., 2010. Real-time quadrifocal visual odometry. International Journal of Robotics Research, 245–266.

Cremona, J., Uzal, L., Pire, T., 2021. WGANVO Repository. https://github.com/CIFASIS/wganvo, [Online; accessed 19-August-2021].

Engel, J., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C., Roberts, A., 2019. GANSynth: Adversarial Neural Audio Synthesis. In: Proceedings of the International Conference on Learning Representations. URL: https://openreview.net/pdf?id=H1xQVn09FX

Engel, J., Koltun, V., Cremers, D., 2018. Direct Sparse Odometry. IEEE Transactions on Pattern Analysis and Machine Intelligence, 611–625. DOI: 10.1109/TPAMI.2017.2658577

Engel, J., Schöps, T., Cremers, D., 2014. LSD-SLAM: Large-Scale Direct Monocular SLAM. In: Proceedings of the European Conference on Computer Vision. pp. 834–849.

Facil, J. M., Ummenhofer, B., Zhou, H., Montesano, L., Brox, T., Civera, J., 2019. CAM-Convs: Camera-Aware Multi-Scale Convolutions for Single-View Depth. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 11818–11827. DOI: 10.1109/CVPR.2019.01210

Forster, C., Pizzoli, M., Scaramuzza, D., 2014. SVO: Fast semi-direct monocular visual odometry. In: Proceedings of the IEEE International Conference on Robotics and Automation. pp. 15–22. DOI: 10.1109/ICRA.2014.6906584

Geiger, A., Lenz, P., Stiller, C., Urtasun, R., 2013. Vision Meets Robotics: The KITTI Dataset. International Journal of Robotics Research, 1231–1237.
URL: http://dx.doi.org/10.1177/0278364913491297
DOI: 10.1177/0278364913491297

Geiger, A., Lenz, P., Urtasun, R., 2012. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3354–3361.
DOI: 10.1109/CVPR.2012.6248074

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets. In: Proceedings of the Advances in Neural Information Processing Systems. pp. 2672–2680.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A. C., 2017. Improved Training of Wasserstein GANs. In: Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Proceedings of the Advances in Neural Information Processing Systems. Vol. 30. Curran Associates, Inc.
URL: https://proceedings.neurips.cc/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf

Hartley, R., Zisserman, A., 2003. Multiple View Geometry in Computer Vision. Cambridge University Press, New York, USA.

Karras, T., Laine, S., Aila, T., 2019. A Style-Based Generator Architecture for Generative Adversarial Networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4396–4405.
DOI: 10.1109/CVPR.2019.00453

Kendall, A., Cipolla, R., 2017. Geometric Loss Functions for Camera Pose Regression with Deep Learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6555–6564.
DOI: 10.1109/CVPR.2017.694

Kendall, A., Grimes, M., Cipolla, R., 2015. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In: Proceedings of the International Conference on Computer Vision. pp. 2938–2946.
DOI: 10.1109/ICCV.2015.336

Krizhevsky, A., Sutskever, I., Hinton, G. E., 2012. ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira, F., Burges, C. J. C., Bottou, L., Weinberger, K. Q. (Eds.), Proceedings of the Advances in Neural Information Processing Systems. Vol. 25. Curran Associates, Inc.
URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Krombach, N., Droeschel, D., Behnke, S., 2016. Combining Feature-based and Direct Methods for Semi-dense Real-time Stereo Visual Odometry. In: Proceedings of the International Conference on Intelligent Autonomous Systems. pp. 855–868.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436.
URL: https://www.nature.com/articles/nature14539

Li, R., Wang, S., Long, Z., Gu, D., 2018. UnDeepVO: Monocular Visual Odometry through Unsupervised Deep Learning. In: Proceedings of the IEEE International Conference on Robotics and Automation. pp. 7286–7291.
DOI: 10.1109/ICRA.2018.8461251

Li, S., Xue, F., Wang, X., Yan, Z., Zha, H., 2019. Sequential Adversarial Learning for Self-Supervised Deep Visual Odometry. In: Proceedings of the International Conference on Computer Vision.

Lowe, D. G., 1999. Object recognition from local scale-invariant features. In: Proceedings of the International Conference on Computer Vision. pp. 1150–1157.
DOI: 10.1109/ICCV.1999.790410

Min, Z., Yang, Y., Dunn, E., 2020. VOLDOR: Visual Odometry From Log-Logistic Dense Optical Flow Residuals. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4897–4908.
DOI: 10.1109/CVPR42600.2020.00495

Mur-Artal, R., Montiel, J. M. M., Tardós, J. D., 2015. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. IEEE Transactions on Robotics, 1147–1163.
DOI: 10.1109/TRO.2015.2463671

Mur-Artal, R., Tardós, J. D., 2017. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. IEEE Transactions on Robotics, 1255–1262.
DOI: 10.1109/TRO.2017.2705103

Nistér, D., Naroditsky, O., Bergen, J., 2004. Visual odometry. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 652–659.
DOI: 10.1109/CVPR.2004.1315094

Pire, T., Fischer, T., Castro, G., De Cristóforis, P., Civera, J., Jacobo Berlles, J., 2017. S-PTAM: Stereo Parallel Tracking and Mapping. Journal of Robotics and Autonomous Systems, 27–42.
DOI: 10.1016/j.robot.2017.03.019

Radford, A., Metz, L., Chintala, S., 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In: Computing Research Repository (CoRR).
URL: http://arxiv.org/abs/1511.06434

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., 2016. Improved Techniques for Training GANs. In: Proceedings of the International Conference on Neural Information Processing Systems. pp. 2234–2242.

Scaramuzza, D., Fraundorfer, F., 2011. Visual Odometry [Tutorial]. IEEE Robotics and Automation Magazine, 80–92.
DOI: 10.1109/MRA.2011.943233

Siciliano, B., Khatib, O., 2016. Springer Handbook of Robotics. Springer Publishing Company, Incorporated.

Tateno, K., Tombari, F., Laina, I., Navab, N., 2017. CNN-SLAM: Real-Time Dense Monocular SLAM with Learned Depth Prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6565–6574.
DOI: 10.1109/CVPR.2017.695

Thrun, S., Burgard, W., Fox, D., 2005. Probabilistic Robotics. The MIT Press.

Tulyakov, S., Liu, M.-Y., Yang, X., Kautz, J., 2018. MoCoGAN: Decomposing motion and content for video generation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1526–1535.
DOI: 10.1109/CVPR.2018.00165

Umeyama, S., 1991. Least-squares estimation of transformation parameters between two point patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence, 376–380.
DOI: 10.1109/34.88573

Wang, S., Clark, R., Wen, H., Trigoni, N., 2017. DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks. In: Proceedings of the IEEE International Conference on Robotics and Automation. pp. 2043–2050.
DOI: 10.1109/ICRA.2017.7989236

Yang, N., Wang, R., Stückler, J., Cremers, D., 2018. Deep Virtual Stereo Odometry: Leveraging Deep Depth Prediction for Monocular Direct Sparse Odometry. In: Proceedings of the European Conference on Computer Vision. pp. 835–852.

Yi, X., Walia, E., Babyn, P., 2019. Generative adversarial network in medical imaging: A review. Medical Image Analysis 58, 101552.
URL: https://www.sciencedirect.com/science/article/pii/S1361841518308430
DOI: https://doi.org/10.1016/j.media.2019.101552

Yin, Z., Shi, J., 2018. GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1983–1992.
DOI: 10.1109/CVPR.2018.00212