XLIV Jornadas de Automática 2023

https://jautomatica.es

Análisis comparativo de técnicas de segmentación de estructuras reticulares

Soler, F. J.[a,*], Peidró, A.[a], Fabregat-Jaén, M.[a], Payá, L.[a], Reinoso, Ó.[a,b]

[a] *Instituto de Investigación en Ingeniería de Elche (I3E), Universidad Miguel Hernández de Elche, Avda. de la Universidad s/n, Edificio Innova, 03202, Elche, Alicante, España.*

[b] *ValgrAI: Valencian Graduate School and Research Network of Artificial Intelligence, Camí de Vera S/N, Edificio 3Q, 46022 Valencia, España.*

To cite this article: Soler, F. J., Peidró, A., Fabregat-Jaén, M., Payá, L., Reinoso, Ó. 2023. Comparative analysis of segmentation techniques for reticular structures. XLIV Jornadas de Automática, 750-755
https://doi.org/10.17979/spudc.9788497498609.750

Resumen

El presente articulo pretende comparar nuestro trabajo anterior en segmentación de estructuras reticulares con redes neuronales frente a un algoritmo ad hoc con el mismo propósito. Actualmente, las redes neuronales o la inteligencia artificial son conceptos muy usados y sinónimos de avances y mejoras, pero en determinados casos es posible emplear técnicas más clásicas, fuera del paradigma de la inteligencia artificial para desarrollar el mismo tipo de tareas con resultados muy similares. Para corroborar esta última mención, en el presente artículo se realiza un análisis comparativo de forma cuantitativa y cualitativa entre un algoritmo ad hoc y el mejor modelo de red neuronal en nuestro último trabajo para segmentar estructuras reticulares. Para la implementación del algoritmo se emplean métodos clásicos como *Random Sample Consensus* (RANSAC) y crecimiento de regiones. Para realizar la comparación de forma cuantitativa se emplean métricas estandarizadas como *precision*, *recall* y *f1-score*. Estas últimas se calcularán sobre una base de datos propia, compuesta por mil nubes de puntos y generada automáticamente en trabajos anteriores. El algoritmo en cuestión esta diseñado expresamente para tal base de datos.

Palabras clave: Crecimiento de Regiones, RANSAC, Segmentación de Planos, Redes Neuronales, Nubes de Puntos, Robots Trepadores

Comparative analysis of segmentation techniques for reticular structures.

Abstract

This article aims to compare our previous work on segmentation of reticular structures with neural networks against an ad hoc algorithm for the same purpose. Nowadays neural networks or artificial intelligence are widely used concepts synonymous with advances and improvements, but in certain cases it is possible to use more classical techniques, outside the paradigm of artificial intelligence to achieve the same type of tasks with similar results. To corroborate last mention, in this article we perform a quantitative and qualitative comparative analysis between an ad hoc algorithm and the best neural network model in our latest work for segmenting reticular structures. Conventional methods such as Random Sample Consensus (RANSAC) and region growing are used to implement the algorithm. Standardised metrics such as *precision*, *recall* and *f1-score* are used for quantitative comparison. The latter will be calculated on a proprietary dataset, consisting of a thousand point clouds automatically generated in previous work. The algorithm in question is designed specifically for such a database.

Keywords: Region Growing, RANSAC, Plane Segmentation, Neural Networks, Point Clouds, Climbing Robots

1. Introducción

Multitud de construcciones actuales emplean sistemas reticulares como elementos estructurales debido a sus excelentes propiedades mecánicas que le permiten soportar elevadas cargas, tener una buena distribución de fuerzas, ser muy rígidas y muy eficientes en términos de material utilizado.

*Autor para correspondencia: f.soler@umh.es

Debido a estas excelentes propiedades las estructuras reticulares son una de las configuraciones más empleadas en torres de eléctricas, grúas, puentes y demás infraestructuras de gran envergadura. Comúnmente este tipo de estructuras se ensamblan mediante multitud de cuerpos metálicos compuestos por superficies planas como es el caso de las torres eléctricas.

Dichas infraestructuras suelen responder adecuadamente ante las inclemencias del tiempo y en entornos hostiles, lo que requiere tareas de mantenimiento e inspección periódicas. Un área de investigación en constante desarrollo se basa en emplear robots trepadores para llevar a cabo este tipo de tareas. Este tipo de robots son dispositivos cuya característica principal es su capacidad para moverse y operar en multitud de superficies, tanto horizontales como verticales, como paredes, techos o estructuras metálicas. Asimismo, son capaces de realizar diferentes tareas de inspección o mantenimiento en entornos complejos, de difícil acceso y enorme riesgo para los operarios, quienes están expuestos a múltiples peligros como caídas, electrocuciones, etc.

Durante los últimos años, los robots aéreos se han empleado para acometer este tipo de inspecciones y mantenimientos ((Akahori et al., 2016), (Jung et al., 2019)). No obstante, se enfrentan a una limitación significativa al tener dificultades en acceder a zonas interiores de las mismas, lo que en ocasiones imposibilita llevar a cabo de manera efectiva de las tareas mencionadas.

Para poder acometer dichas tareas es preciso un adecuado reconocimiento del entorno por el que desarrollan estas tareas. Actualmente, uno de los sensores más utilizados para percibir el entorno son los denominados LiDAR. Estos sensores proporcionan información de rango y se usan en multitud de aplicaciones a día de hoy, como localización, construcción de mapas, detección y segmentación de objetos, etc.

Identificar correctamente las superficies planas presentes en un determinado entorno es un aspecto clave para llevar a cabo una navegación precisa. Los planos disponibles en la estructura permiten una representación paramétrica haciendo posible un modelado del entorno más ligero.

Equipar a un robot trepador con un sensor LiDAR permite conocer la distribución espacial del entorno que le rodea. Además, si le añadimos a lo anterior la capacidad de identificar superficies planas, parece una solución ideal para abordar tareas de navegación en las estructuras reticulares, donde prácticamente todos sus componentes están formados por planos.

Debido a las propiedades de los sensores LiDAR suelen captar mucha información, en ocasiones demasiada, y es necesario eliminar aquellos datos no deseados. En un primer momento, el objetivo primordial para posibilitar la navegación de la plataforma robótica sobre este tipo de estructuras reticulares consiste en identificar de toda la información proporcionada por los sistemas sensoriales disponibles, aquella información referente únicamente a la estructura, siendo necesario identificar que parte de la información pertenece a esta y cual no. Esta tarea podría definirse como una clasificación de cada punto o una segmentación de la información original en determinadas clases.

Durante los últimos años podemos encontrar trabajos relacionados en los que se emplean sistemas y algoritmos de inteligencia artificial para abordar este problema. Así, encontramos trabajos existentes de segmentación de planos con redes neuronales como (Yang and Kong, 2020) o (Lee and Jung, 2021). En estos trabajos se emplean redes neuronales para identificar planos en entornos de interior y exterior (entornos urbanos) respectivamente. Sin embargo, estos entornos difieren enormemente de los entornos objetivo de nuestro trabajo, razón por la cual en trabajos previos desarrollados en el grupo de investigación se abordó este trabajo con una propuesta de segmentación mediante el empleo de redes neuronales específicas (Soler et al., 2023).

Por otro lado, existen multitud de métodos para identificación de planos con algoritmos fuera del paradigma de la inteligencia artificial como en (Su et al., 2022). En el mismo se propone una segmentación en dos pasos, una primera etapa donde se seleccionan los planos mediante crecimiento de regiones y una segunda fase en la que se clasifican los puntos de frontera entre dos planos, donde el algoritmo de crecimiento de regiones no es capaz de funcionar correctamente. En (Gaspers et al., 2011) se utiliza de forma similar una segmentación en dos etapas pero sobre múltiples resoluciones. Para cada resolución se extraen características denominadas *surfels* basadas en las normales y se intenta asociar esos *surfels* con planos establecidos en resoluciones más bajas. Aquellos *surfels* que no se asocian con ningún plano conocido se intentan agrupar en función de su coplanaridad mediante la transformada de Hough. Por otro lado, se aplica RANSAC a los conjuntos de *surfels* que se han asociado con un mismo plano en resoluciones más bajas para mejorar la precisión. Una vez alcanzada la máxima resolución, se identifican los segmentos coplanares cercanos y se fusionan en un único plano.

El algoritmo propuesto en este estudio se asemeja a los anteriores, en cuanto emplea una estrategia en dos etapas, pero a diferencia de los anteriores (segmentar la nube de puntos proporcionada por el sensor LiDAR en múltiples conjuntos planos) su objetivo es dividir la nube de puntos únicamente en dos conjuntos, estructura y no estructura.

El presente artículo tiene por objetivo comparar el funcionamiento de las redes neuronales frente a algoritmos de aplicación específica para segmentar estructura reticulares en un entorno concreto. Para ello se comparará nuestro trabajo anterior (Soler et al., 2023), con un algoritmo propuesto de aplicación específica para el dataset de que disponemos sobre estructuras reticulares.

El artículo se organiza como sigue. La segunda sección (2) relata brevemente el trabajo anterior para poner al lector en contexto y poder realizar la comparación posterior. En la Sección 3 se profundiza en el método propuesto para segmentar las estructuras reticulares de nuestra base de datos y se comentarán todos sus pasos. Seguidamente, la Sección 4 desarrolla un análisis comparativo entre la segmentación de estructuras reticulares mediante redes neuronales y con algoritmos clásicos. Por último, la Sección 5 trata unas breves conclusiones sobre los resultados obtenidos.

2. Trabajos previos

En esta sección se expone de forma resumida los conceptos fundamentales de nuestro trabajo previo. En trabajos previos (Soler et al., 2023), se llevó a cabo el entrenamiento específico

de redes neuronales con el propósito de identificar estructuras reticulares a partir la información del entorno proporcionada por un sensor LiDAR.

Las estructuras reticulares son sistemas interconectados mediante uniones rígidas formando una configuración reticular tridimensional. Este tipo de sistemas se pueden encontrar en multitud de infraestructuras, como puentes, edificios, torres eléctricas o grúas y generalmente están formados por elementos de carácter metálico con múltiples superficies planas.

El mencionado trabajo se llevó a cabo con el propósito de ser implementado en el robot HyReCRo (Peidro et al., 2015). Un robot trepador de 10 grados de libertad y configuración serie-paralela con la capacidad de navegar a través de estructuras metálicas mediante el uso de un mecanismo de adhesión magnética basados en imanes permanentes conmutados mecánicamente.

Uno de los primeros retos a resolver para cumplir el objetivo del referido trabajo, es la falta de bases de datos para el entrenamiento. Como solución, el trabajo en cuestión expone el desarrollo de un *plugin* sobre el software de simulación de Gazebo para generar de forma automatizada una base de datos etiquetada. A modo de generalizar la base de datos para multitud de estructuras reticulares, esta se forma mediante entornos compuestos por paralelepípedos y elementos como árboles y suelo modelando un entorno real. De igual modo que para los datos de entrenamiento, se generan los datos de evaluación, pero con un modelo de estructura reticular en lugar de paralelepípedos.

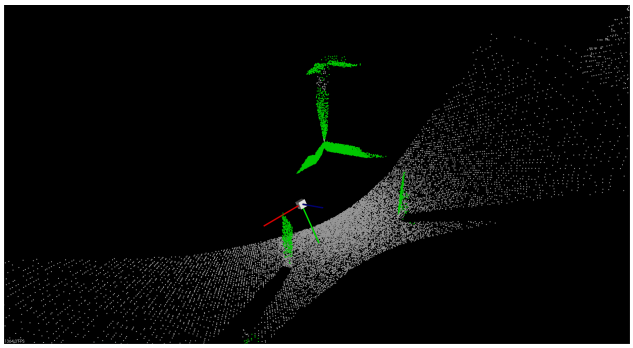


Figura 1: Ejemplo de inferencia del mejor modelo entrenado en (Soler et al., 2023)

Diferentes arquitecturas de redes neuronales fueron analizadas para su resolución, PointNet (Qi et al., 2016), PointNet++ (Qi et al., 2017) y MinkUNet34C (Choy et al., 2019). El mejor de los resultados se obtuvo con la arquitectura MinkUNet34C, que emplea el MinkowskiEngine para realizar convoluciones 3D de forma dispersa únicamente en aquellos puntos que contienen información. Esta última arquitectura ofrece muy buenos resultados, en torno a 96 % de f1-score (5) sobre la base de datos de evaluación. En la Figura 1 se observa un ejemplo de inferencia de esta modelo de red.

En el apartado 4 se exponen en mayor profundidad sus resultados y se comparan con el algoritmo expuesto en el presente artículo.

3. Algoritmo propuesto

El algoritmo planteado en este estudio se compone de dos fases. La primera de ellas realiza una clasificación gruesa, identificando el plano del suelo y elementos cercanos a este. En segundo lugar, se ejecuta un proceso en el que se afina la clasificación. Este método adopta algoritmos conocidos en la literatura para la segmentación e identificación de planos tales como *Random Sample Consensus* (RANSAC) o crecimiento de regiones con el objetivo de discriminar los puntos en dos clases, estructura y no estructura.

Cabe destacar que este algoritmo se ha diseñado expresamente para su funcionamiento en los entornos que se han generado para la base de datos de evaluación de nuestro trabajo previo. Para su desarrollo se toma en consideración que dentro de la lectura del sensor existirá un gran número de puntos pertenecientes al suelo, además de ocupar estos un área elevada del entorno. En el Algoritmo 1 se describe a modo de pseudocódigo el proceso seguido para completar la segmentación. La implementación de este trabajo se apoya en la librería *Point Cloud Library* (PCL) (Rusu and Cousins, 2011) para realizar el tratamiento de las nubes de puntos. En los subapartados siguientes se profundiza en la descripción de cada una de sus fases.

Algorithm 1 Algoritmo propuesto

```
1: Segmentación Gruesa
2:     Voxelize
3:     RANSAC
4: Segmentación Fina
5:     Crecimiento de regiones
6:     Cálculo de autovalores con PCA
7:     Filtrado de conjuntos
```

Dividir en dos etapas el algoritmo tiene por objetivo reducir el coste computacional de este. Partiendo de los experimentos realizados, el mayor coste computacional del algoritmo se debe a la estimación de normales, proceso que consume en torno al 80 % del tiempo total de ejecución del algoritmo. En caso de emplear directamente una clasificación fina, sería necesario estimar las normales de la nube completa, además de tener que calcular y evaluar los autovalores de un mayor número de conjuntos. Aplicando este tipo de enfoque se requeriría de media un 20 % más de tiempo de ejecución por nube para obtener los mismos resultados.

3.1. Clasificación Gruesa

En la etapa inicial del algoritmo se realiza una clasificación gruesa con el fin de obtener los puntos de suelo. En esta etapa se usa un filtrado de Voxel seguido de una extracción del mayor plano con RANSAC.

Dado que RANSAC selecciona el mejor candidato a plano en función del número de *inliers*, si existen zonas con alta densidad de puntos, RANSAC tiende a seleccionar planos en dichas zonas. Para evitar este problema, se aplica en primer lugar un filtrado de Voxel, proceso mediante el cual se homogeneiza la densidad de puntos a lo largo de toda la nube, facilitando así la extracción del mayor plano (plano de suelo). Para abarcar la mayor cantidad de puntos de suelo posibles, se establece un

umbral elevado para la obtención del mayor plano, entorno a los 0.5 metros, salvando así inclinaciones en el terreno.

3.2. *Clasificación Fina*

La clasificación gruesa produce una nube de menor tamaño que contiene los puntos del suelo, así como puntos cercanos a este, dentro de un determinado umbral. Sobre dicha nube reducida se aplica una clasificación más precisa. La idea de esta etapa es dividir la nube en conjuntos planares y discriminar estos últimos por su tamaño.

Para segmentar los diferentes conjuntos planares se hace uso del crecimiento de regiones basado en las normales. El procedimiento para construir los conjuntos se basa en la similitud de las normales de puntos cercanos, por lo que la estimación de dichas características es un aspecto muy determinante en las agrupaciones resultantes.

En los subapartados sucesivos se lleva a cabo un desarrollo más en profundidad sobre la estimación de las normales de cada punto y los criterios de decisión para filtrar los mencionados conjuntos.

3.2.1. *Estimación de normales*

Tal y como se menciona en la sección anterior, la estimación de normales es un componente clave a la hora de establecer correctamente conjuntos planares. La función de estimación de normales implementada en PCL consiste en calcular los autovalores y autovectores sobre la vecindad de cada punto, donde el autovector asociado con el menor autovalor se considera el vector normal referente al punto. Los autovalores y autovectores se obtienen mediante un análisis de las componentes principales (PCA) sobre la matriz de covarianza para cada punto y su entorno de vecindad. Esta matriz (C) utiliza la formulación indicada en (1), donde k es el número de vecinos, $\overline{\boldsymbol{p}}$ es el centroide del conjunto de vecinos, λ_j el autovalor y $\vec{\mathrm{v}}_j$ el autovector para j.

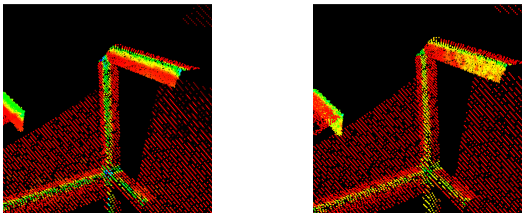
$$C = \frac{1}{k}\sum_{i=1}^{k} \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}}) \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}})^T,\ C \cdot \vec{\mathrm{v}}_j = \lambda_j \cdot \vec{\mathrm{v}}_j,\ j \in \{0, 1, 2\} \quad (1)$$

El método de estimación de normales expuesto en el párrafo anterior requiere seleccionar un conjunto de puntos vecinos para cumplir su cometido. La librería utilizada permite dos opciones excluyentes para tal fin, seleccionar todos los puntos ubicados dentro de una esfera de radio definido, o seleccionar aquellos puntos más cercanos con un límite en número. En función del método de selección de vecinos, los autovectores y autovalores de dicho conjunto pueden variar significativamente. Un ejemplo de esto se puede observar en la Figura 2 donde se representa la nube de puntos en función de su curvatura (2), definida como el cociente entre el menor autovalor y la suma de todos los autovalores. En la parte izquierda se seleccionan los vecinos por radio y a la derecha por vecinos más cercanos.

$$Curvatura = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \quad (2)$$

Tras las pruebas realizadas, se ha llegado a la conclusión de que la estimación del vector normal de cada punto, es más precisa cuando se usa un número determinado de vecinos cercanos. El cálculo de normales mediante radio es altamente erróneo en zonas con baja densidad de puntos, es decir, no existen suficientes puntos en el radio indicado para estimar el vector normal. En cambio, el cálculo de la curvatura, la cual indica en cierto modo la dispersión de los puntos, es más precisa cuando se emplea la selección de vecinos por radio, tal y como se observa en la Figura 2, donde en el método de selección por radio, puntos con elevada curvatura se corresponden con las aristas de corte de los distintos planos de la estructura.

El valor de curvatura es importante ya que como veremos a continuación, se utiliza como criterio de finalización para el crecimiento de regiones. Tal y como se indica en (Pauly et al., 2002), si nos fijamos en (2), el valor máximo de curvatura será $\lambda_{max} = 1/3$ y se dará cuando $\lambda_0 = 1$, puesto que $\lambda_0 \leq \lambda_1 \leq \lambda_2$ y $\lambda_{0-2} \in \{0, 1\}$.



(a) Por radio (b) Por vecinos

Figura 2: Nubes de puntos representadas en función de su curvatura.

Esto quiere decir que valores pequeños de curvatura, indican que los puntos están muy poco dispersos a lo largo del menor autovector (la mayor parte de los puntos caen sobre un plano) y valores cercanos a 1/3 en la curvatura significan que los puntos están uniformemente dispersos por todo el espacio de vecindad seleccionado.

3.2.2. *Crecimiento de regiones*

El algoritmo de crecimiento de regiones es una técnica ampliamente utilizada para identificar y agrupar conjuntos de datos en regiones coherentes. Se basa en el principio de que los puntos adyacentes con características similares deben pertenecer al mismo conjunto.

En el presente trabajo, se emplea la implementación disponible en la librería PCL.En primer lugar, es necesario estimar las normales junto a su curvatura, una vez hecho esto, se selecciona la semilla inicial en el punto con menor curvatura de toda la nube y se va incrementando el tamaño del conjunto con aquellos puntos vecinos que cumplan determinados requisitos. Existen dos condiciones a cumplir para introducir un nuevo punto en el conjunto. La primera de ellas consiste en evaluar la diferencia angular entre las normales de los puntos con la semilla inicial, si esta diferencia es menor a un cierto umbral, el nuevo punto se añade al conjunto. En segundo lugar, se comprueba la curvatura de los nuevos puntos añadidos al conjunto, si esta es inferior a un cierto umbral, dichos puntos se convierten en nuevas semillas para seguir creciendo el conjunto. El conjunto seguirá creciendo hasta que no existan semillas.

3.2.3. *Evaluación de los autovalores*

Para identificar si un conjunto proporcionado por el crecimiento de regiones pertenece o no a la estructura se hace uso

de sus autovalores y autovectores.Se han evaluado tres versiones del algoritmo, cuya diferencia se basa en cómo se usan los autovalores o autovectores para discernir si una agrupación de puntos pertenece a la estructura o no.

3.2.4. Por ratio

Esta variante se basa en el conocimiento previo de la estructura. Con el cual se puede considerar que los planos que pertenecen a la estructura tienen una geometría alargada y estrecha. Partiendo de la consideración anterior, esta variante únicamente utiliza el cociente entre los dos mayores autovalores ($ratio = \frac{\lambda_1}{\lambda_2}$) para obtener una relación entre el largo y el ancho del conjunto candidato y determinar así su clasificación.

3.2.5. Por Módulo

Siguiendo el mismo enfoque que la variante anterior, un conocimiento previo de la geometría de las barras de la estructura, permite filtrar estas a partir del módulo de la proyección del punto más alejado en sus ejes principales. De modo que se establecen umbrales para las dos dimensiones del plano descrito por el conjunto de puntos. Conjuntos cuyos valores sean superiores a los umbrales indicados (máxima longitud y amplitud de los elementos estructurales) serán excluidos de la clasificación de estructura.

3.2.6. Híbrido

La última variante combina las dos anteriores, aplicando en primer lugar un filtrado por módulo y seguidamente por ratio. Este método es el que mejor resultados ofrece tal y como se puede comprobar en la Tabla 1.

3.2.7. Filtro de densidad

Por último, y tomando en consideración que el robot y por lo tanto el sensor, estarán a escasa distancia sobre la estructura, la densidad de los puntos será elevada en zonas pertenecientes a la estructura y al contrario en zonas de suelo y del entorno. Con este último paso se eliminan el resto de puntos espurios del entorno dejando paso únicamente a la estructura.

3.3. Limitaciones

Cabe remarcar que el algoritmo está diseñado expresamente para los datos de que disponemos, en los que se captura gran información del entorno y de suelo debido a que se emplea una simulación realista del sensor comercial Ouster OS1. Los datos generados se ajustan a las especificaciones del sensor real, para ello se ha configurado este para proporcionar una resolución de de 512x128 puntos, un rango máximo de 30 metros, unos campos de visión vertical y horizontal de 45º y 360º respectivamente y un ruido gaussiano de media cero y desviación estándar de 0.008 metros. Sus elevados campos de visión y rango, provocan que se capture mucha información de suelo y del entorno circundante, motivo por el cual la primera etapa del algoritmo es capaz de identificar el plano del suelo.

4. Análisis comparativo

A fin de evaluar y contrastar el rendimiento del algoritmo ad hoc expuesto frente al mejor modelo de red neuronal obtenido en trabajos previos, este apartado desarrolla un análisis comparativo con el fin de destacar las ventajas y limitaciones de cada método.

4.1. Métricas evaluadas

Como métricas de evaluación se van a emplear las mismas que en nuestro trabajo previo y comúnmente extendidas para evaluar el funcionamiento de redes neuronales tales como la *Precision*, el *Recall* y la *F1-Score*. Además, se evalúa el tiempo de inferencia, o tiempo de cómputo medio necesario para obtener la segmentación de una nube de puntos de entrada.

La *precision* (3) indica el grado de certeza del algoritmo o de la red neuronal en su caso. En otras palabras, indica el porcentaje de acierto.

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

El *recall* (4) muestra el volumen de datos que somos capaces de predecir correctamente.

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

Por último, el *f1-score* (5) es una métrica que engloba las dos anteriores, a modo de tener un único indicador del buen funcionamiento del proceso.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{5}$$

4.2. Redes Neuronales

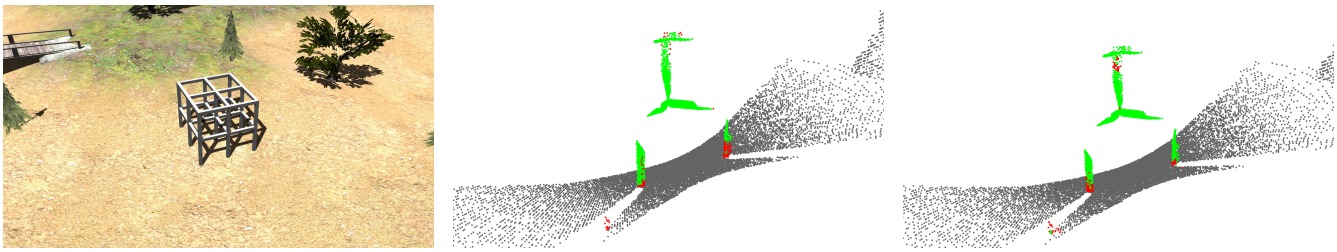
La red neuronal empleada (MinkUNet34C) en esta comparativa se trata de una red convolucional 3D que emplea convoluciones dispersas reduciendo el tiempo de computación considerablemente respecto a las convoluciones tradicionales. Presenta unos buenos resultados para la segmentación de estructuras reticulares así como capacidad de generalización, tanto en cuanto es capaz de identificar diferentes tipos de estructuras reticulares en distintos entornos.

En sus aspectos negativos destacan la necesidad de disponer de una base de datos tanto de entrenamiento como de evaluación lo suficientemente extensas como para obtener un buen rendimiento. Además de este último hecho, se le suma el elevado tiempo de entrenamiento y la necesidad de disponer de un hardware específico para ejecutarse de forma rápida. Los tiempos de entrenamiento registrados rondan las 12 horas en una NVIDIA RTX 3090 de 24GB para una base de datos de 10.000 nubes de puntos y 25.000 puntos por nube.

4.3. Algoritmo ad hoc

El método propuesto en este artículo alcanza unos resultados muy similares a los de la red neuronal. Entre sus principales ventajas respecto a la red neuronal destaca la independencia de bases de datos, hardware específico y tiempo de entrenamiento. En lo referente a esto último, el algoritmo tiene una gran ventaja, ya que se pueden realizar ajustes en los parámetros y obtener resultados inmediatos sin necesidad de completar el proceso de aprendizaje.

Por otro lado, su principal limitación radica en la falta de generalización, siendo necesario adaptar este algoritmo expresamente para cada entorno y geometría de estructura donde se quiera emplear.



(a) Entorno sobre el cual se evalúan los resultados. (b) Resultado del algoritmo propuesto (c) Resultado de MinkUNet34C

Figura 3: Entorno evaluado y ejemplo de la segmentación realizada mediante ambos métodos. El color rojo indica errores de clasificación.

4.4. Resultados

Analizando los resultados mostrados en la Tabla 1 se puede observar como la versión híbrida de nuestro algoritmo es capaz de mejorar la precisión respecto a la red neuronal, lo que se traduce en que sólo un 4,34 % de los puntos identificados como estructura son erróneos. Por otro lado la red neuronal tiene un recall más elevado, valor que indica el porcentaje de puntos de la estructura que es capaz de identificar. La F1-score se utiliza como medida para englobar la precisión y el recall, donde la diferencia entre ambos casos es muy reducida, tan sólo de un 0,36 %. En la Figura 3 se puede observar como con ambos métodos se obtienen resultados bastante similares al igual que refleja la Tabla 1.

Tabla 1: Resultados evaluados en la comparación

	*F*1	*Precision*	*Recall*	*Time*(*ms*)
*MinkUNet*34*C*	**0.9622**	0,9425	**0.9840**	**45**
Ratio	0,8880	0,8503	0,9617	72
Module	0,9427	0,9273	0,9633	72
Hybrid	0,9586	**0.9566**	0,9631	74

5. Conclusiones

Tras el estudio realizado en este trabajo, se evidencia que las redes neuronales no son siempre la mejor opción para realizar determinadas tareas, como se muestra en este artículo, se puede obtener resultados muy similares a las redes neuronales sin necesidad de un proceso de entrenamiento el cual requiere de una base de datos etiquetada para el entrenamiento y su posterior evaluación. Tras analizar los resultados de la comparación queda comprobado que con un algoritmo expresamente diseñado para esta tarea, cuyo tiempo de desarrollo es inferior al de una red neuronal puede resultar más atractivo en determinados casos. Complementando lo anterior, el algoritmo propuesto se puede ejecutar de forma paralela, agilizando el tiempo de ejecución actual y siendo posible su uso en equipos móviles de pequeño tamaño y con capacidad de cómputo limitada.

Agradecimientos

Este trabajo es parte del proyecto PID2020-116418RB-I00 financiado por MCIN/AEI/10.13039/501100011033.

La presente investigación también ha sido posible gracias al proyecto TED2021-130901B-I00, financiado por MCIN/AEI/10.13039501100011033 y por la Unión Europea “NextGenerationEU”/PRTR.

Referencias

Akahori, S., Higashi, Y., Masuda, A., 2016. Development of an aerial inspection robot with epm and camera arm for steel structures. In: 2016 IEEE Region 10 Conference (TENCON). pp. 3542–3545.
DOI: 10.1109/TENCON.2016.7848716

Choy, C., Gwak, J., Savarese, S., 2019. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3075–3084.

Gaspers, B., Stückler, J., Welle, J., Schulz, D., Behnke, S., 12 2011. Efficient multi-resolution plane segmentation of 3d point clouds. In: 4th International Conference on Intelligent Robotics and Applications (ICIRA). pp. 145–156.
DOI: 10.1007/978-3-642-25489-5_15

Jung, S., Song, S., Kim, S., Park, J., Her, J., Roh, K., Myung, H., 2019. Toward autonomous bridge inspection: A framework and experimental results. In: 2019 16th International Conference on Ubiquitous Robots (UR). pp. 208–211.
DOI: 10.1109/URAI.2019.8768677

Lee, H., Jung, J., 12 2021. Clustering-based plane segmentation neural network for urban scene modeling. Sensors 21, 8382.
DOI: 10.3390/s21248382

Pauly, M., Gross, M., Kobbelt, L., 2002. Efficient simplification of point-sampled surfaces. In: IEEE Visualization, 2002. VIS 2002. pp. 163–170.
DOI: 10.1109/VISUAL.2002.1183771

Peidro, A., Gil, A., Marin, J., Reinoso, O., 2015. Inverse kinematic analysis of a redundant hybrid climbing robot. International Journal of Advanced Robotic Systems 12, 1.
DOI: 10.5772/61748

Qi, C. R., Su, H., Mo, K., Guibas, L. J., 2016. Pointnet: Deep learning on point sets for 3d classification and segmentation. CoRR abs/1612.00593.
URL: http://arxiv.org/abs/1612.00593

Qi, C. R., Yi, L., Su, H., Guibas, L. J., 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. CoRR abs/1706.02413.
URL: http://arxiv.org/abs/1706.02413

Rusu, R. B., Cousins, S., 2011. 3d is here: Point cloud library (pcl). In: 2011 IEEE International Conference on Robotics and Automation. pp. 1–4.
DOI: 10.1109/ICRA.2011.5980567

Soler, F. J., Peidró, A., Fabregat-Jaén, M., Payá, L., Reinoso, O., June 2023. Segmentación de planos a partir de nubes de puntos 3d en estructuras reticulares. In: XIII Jornadas Nacionales de Robótica y Bioingeniería. Centro de Automática y Robótica, CAR, Madrid, Spain, pp. 91–98.

Su, Z., Gao, Z., Zhou, G., Li, S., Song, L., Lu, X., Kang, N., 2022. Building plane segmentation based on point clouds. Remote Sensing 14 (1).
URL: https://www.mdpi.com/2072-4292/14/1/95
DOI: 10.3390/rs14010095

Yang, H., Kong, H., 2020. 3dpmnet: Plane segmentation and matching for point cloud registration. In: 2020 3rd International Conference on Unmanned Systems (ICUS). pp. 439–444.
DOI: 10.1109/ICUS50048.2020.9274884