Aplicación de una Red Neuronal para el Seguimiento de la Trayectoria en un Robot Diferencial

Federico Ignacio Nuñez Frau

6 de febrero de 2022

Resumen

En este trabajo se presenta una aplicación de redes neuronales para resolver el problema del seguimiento de una trayectoria para un robot. Esto se logra a partir de indicarle al robot, uno por uno, cuáles son los puntos del espacio que debe visitar. El diseño de un controlador clásico, para obtener una buena precisión en el seguimiento, puede ser una tarea trabajosa debido a errores de modelo, además de requerir mucho esfuerzo en el diseño y ajuste. En este trabajo se utiliza un método que combina un controlador clásico muy sencillo, junto con una red neuronal. Los resultados muestran que la red neuronal mejora la precisión que se obtiene si solamente se utiliza el controlador clásico.

1. Introducción

Este trabajo es una reproducción del trabajo *Deep Neural Networks for Improved, Impromptu Trajectory Tracking of Quadrotors*, desarrollado por la *Dynamic Systems Lab* de la Universidad de Toronto, Canadá [1]. Este conisistió en el entrenamiento de una red neuronal con información de vuelo de drones que luego fue utilizada para mejorar la precisión en el seguimiento de una trayectoria. En el trabajo que aquí se presenta, en lugar de utilizar un drón, se utiliza un simulador de un robot que puede moverse en un plano.

Para un robot que se mueve en un plano xy, la pose de este robot se puede describir como un vector de dimensión 3:

$$\overrightarrow{X(t)} = \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix} \tag{1}$$

donde $(x(t), y(t))$ describe el punto del plano en el que se encuentra el robot y el ángulo $\theta(t)$ describe la orientación del robot respecto de la horizontal. Esto se muestra en la figura 1.

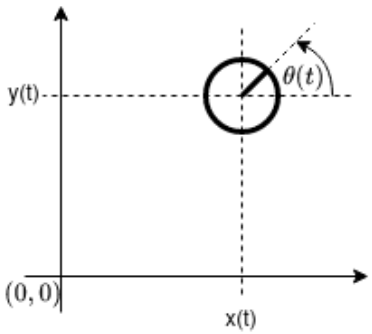


Figura 1: Vista superior del robot. La pose está compuesta por la posición en el plano junto con la orientación.

En este caso se trabaja con el modelo de un robot con accionamiento diferencial [5]. Las ecuaciones (2) describen el movimiento de este robot, donde $v(t)$ y $\omega(t)$ son la velocidad lineal y angular respectivamente del robot.

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} cos[\theta(t)] & 0 \\ sin[\theta(t)] & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{2}$$

De esta forma se tiene un sistema cuya salida es la pose del robot y las entradas son las velocidades lineal y angular.

2. Descripción del Problema

El objetivo es que el robot recorra una trayectoria deseada con una buena precisión. En la figura 2 se muestra un esquema del sistema de control utilizado para recorrer la trayectoria.

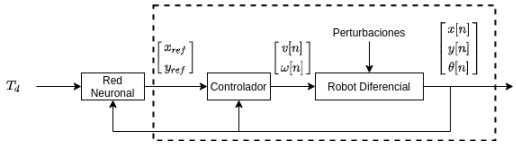


Figura 2: Esquema del sistema seguidor de trayectorias. Lo que se encuentra recuadrado con línea de rayas corresponde al controlador clásico y T_d corresponde a una serie de puntos que debe visitar el robot para recorrer la trayectoria deseada.

El bloque "Controlador" se diseñó siguiendo lo propuesto en [4]. Este controlador clásico se encarga de:

- Hacer que todo lo que se encuentra recuadrado en línea de raya sea un seguidor de referencia para la posición del robot en el plano.
- Que este sistema sea estable.

Adelante se coloca una red neuronal, en este caso un perceptrón multicapa, que se encarga de entregarle al sistema recuadrado en línea de rayas, los puntos (x_{ref}, y_{ref}) necesarios para seguir adecuadamente la trayectoria deseada. Mientras el controlador clásico se encarga de lo ya mencionado, la red neuronal es la encargada de que se obtenga una buena precisión en el seguimiento de la trayectoria.

La trayectoria a seguir, T_d, se define como un conjunto de puntos que debe visitar el robot de forma consecutiva. Esto se muestra en la figura 3.

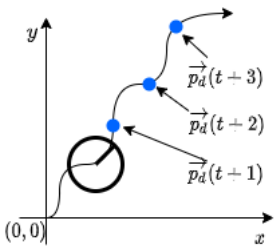


Figura 3: Los puntos azules representan los puntos que debe visitar el robot para seguir la trayectoria deseada, de N puntos.

3. Desarrollo

Se realizaron una serie de simulaciones utilizando un simulador de un robot diferencial. Este simulador, que puede encontrarse en [2], fue desarrollado para probar algoritmos de localización y mapeo simultáneo, para la materia "86.48 Seminario de Electrónica: Robótica Móvil". Para este trabajo, de forma de simplificar las simulaciones, no se utilizaron las funcionalidades de localización ni de mapeo, solamente se le entregan comandos de velocidad lineal y angular al robot y este responde moviéndose en el plano.

De forma de darle mayor realismo a las simulaciones, se utiliza el modelo descrito en la sección 5.3.3 de [3], en el cual se menciona que a los comandos de velocidad lineal y angular entregados por el usuario,

se les suma un ruido blanco gaussiano cuya varianza depende de parámetros del robot. Además al robot se le configuró una velocidad lineal máxima de 2,0 m/s y una velocidad angular máxima de 5,0 rad/s.

En cuanto al perceptrón multicapa, en la figura 2 se muestra que las entradas son la trayectoria T_d y la pose actual. Para disminuir la cantidad de entradas de la red, en lugar de entregarle toda la trayectoria T_d solamente se le entregan L puntos de la trayectoria, con $L << N$. Esto puede justificarse pensando que si el robot se encuentra en el primer punto de la trayectoria, no tiene sentido entregarle como información a la red los últimos puntos de la trayectoria. Además, a cada punto perteneciente a la trayectoria se le agregó una componente de orientación. Esta se calculó considerando la orientación respecto de la horziontal del vector que une cada punto de la trayectoria con el siguiente. En la figura 4 se muestra un esquema con las entradas y salidas.

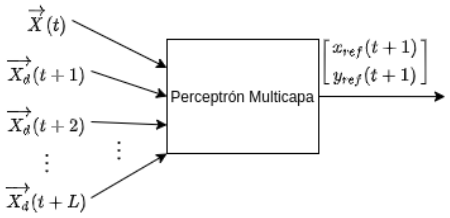


Figura 4: Entradas y salidas de la red neuronal de la figura 2. $\overrightarrow{X}(t)$ es la pose del robot en el instante t, $\overrightarrow{X_d}(t+1)$ y $\overrightarrow{X_d}(t+2)$ son las próximas 2 poses deseadas, $[x_d(t+k)\ y_d(t+k)\ \theta_d(t+k)]^T$, donde los puntos $(x_d(t+k), y_d(t+k))$ son puntos pertenecientes a la trayectoria deseada y $\theta_d(t+k)$ es la componente de orientación para cada punto de la trayectoria.

Se utilizó la biblioteca Tensorflow. Para todas las simulaciones se utilizó una red con 1 capa oculta de 1024 neuronas. Se entrenó a la red utilizando Adam con un ritmo de entrenamiento de 0,0003. También se utilizó *dropout*, con un valor de 0,5 para evitar el *overfitting*. En algunas simulaciones se hicieron modificaciones a estas configuraciones. Las mismas se mencionan según corresponda.

4. Simulaciones Realizadas

Se corrieron simulaciones para 4 trayectorias diferentes. En todos los casos primero se realizó el seguimiento de la trayectoria sin la red neuronal, guardando la pose del robot en cada instante de tiempo. Estos datos se utilizaron para entrenar una red neuronal con las características ya mencionadas. Luego se repitió la simulación pero incorporando la red neuronal entrenada.

En cada simulación se calcularon dos tipos de errores:

$$E_{RMS} = \sqrt{\frac{1}{N}\sum_{t=1}^{N} \|\overrightarrow{p_d}(t) - \overrightarrow{p}(t)\|^2} \tag{3a}$$

$$E_{peak} = max\{\|\overrightarrow{p_d}(t) - \overrightarrow{p}(t)\|\} \tag{3b}$$

donde $\overrightarrow{p}(t) = (x(t), y(t))$ es la posición del robot en cada instante de tiempo y $\overrightarrow{p_d}(t) = (x_d(t), y_d(t))$ es el punto de la trayectoria deseada donde se debería haber encontrado al robot.

Para recolectar los datos de entrenamiento, se simuló el seguimiento de la trayectoria solamente utilizando un controlador clásico, tomando muestras de la pose del robot en cada instante de tiempo. Si se asume que la trayectoria que siguió el robot es la trayectoria deseada T_d, luego los puntos de referencia $\overrightarrow{X}_{ref}(t+1)$ son las referencias que se le deben entregar al controlador clásico para que el robot recorra dicha trayectoria. En la figura 5 se muestra cómo se recolectaron los datos para entrenar a la red.

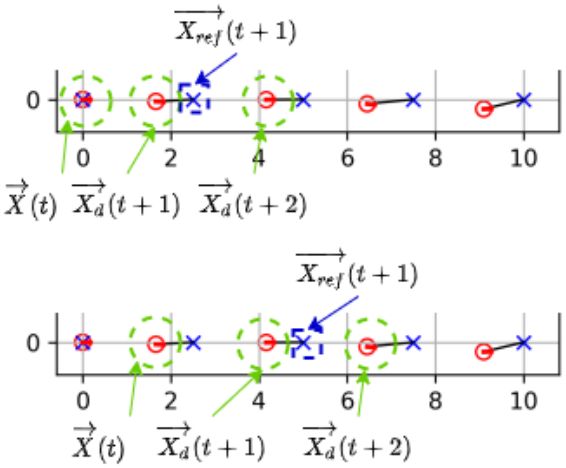


Figura 5: Se muestra una trayectoria en línea recta formada por 5 puntos. Las cruces azules corresponden a los puntos de referencia que se le entregaron al controlador clásico del robot. Los puntos rojos representan el lugar dónde se encontró al robot en cada instante de tiempo. Las líneas negras unen los puntos de referencia vs los puntos donde se encontró al robot en cada instante de tiempo. Se muestran dos imágenes donde se destacan los datos que se utilizaron para entrenar a la red. Si se encontró al robot en $\overrightarrow{X}(t)$, se lo quiere encontrar en $\overrightarrow{X_d}(t+1)$ y luego en $\overrightarrow{X_d}(t+2)$, la red neuronal debería generar la referencia $\overrightarrow{X_{ref}}(t+1)$.

A cada componente de orientación de las entradas de la red de la figura 4 se les sumó el valor 2π. Esto debido a que en algunas simulaciones se obtuvieron resultados que generaron problemas cuando el robot debía recorrer una parte de la trayectoria manteniendo una orientiación de $\pm\pi$. Al hacer esta modificación, el problema fue solucionado.

4.1. Caso 1: Entrenamiento con Datos de la Misma Trayectoria

En este primer caso se muestra el resultado de incorporar la red neuronal al sistema seguidor de trayectorias, para 4 trayectorias diferentes. En cada una se utiliza una red neuronal diferente. La red utilizada para la trayectoria 1 fue entrenada con datos de la trayectoria 1. De la misma manera, cada red utilizada se entrenó con datos de su trayectoria correspondiente.

4.1.1. Trayectoria 1

La trayectoria comienza en el $(0;0)$ y avanza hacia el $(10;0)$. El último punto de la trayectoria es el $(0;2{,}5)$. En la figura 6 se muestra el resultado sin utilizar la red neuronal. En la figura 7a se muestra el resultado incorporando la red neuronal ya entrenada, con $L=1$ y en la figura 7b con $L=2$.

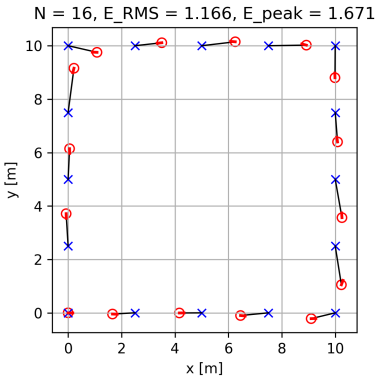
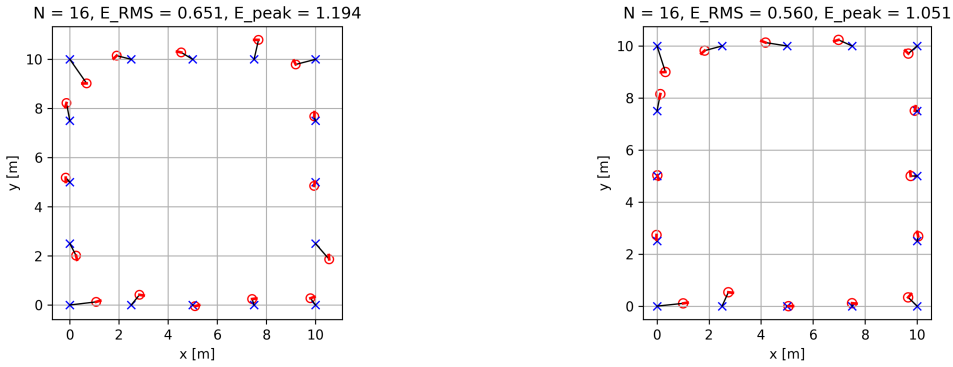


Figura 6: Las cruces azules pertenecen a la trayectoria deseada mientras que los círculos rojos son la pose del robot. Cada pose está unida a una cruz azul a través de una línea negra. Esto representa el lugar donde se encontró al robot vs. el lugar donde se lo quería encontrar.



(a) $L = 1$ (b) $L = 2$

Figura 7: Se muestran los resultados obtenidos incorporando la red neuronal entrenada con datos de la trayectoria de la figura 6.

4.1.2. Trayectoria 2

Muy similar a la trayectoria 1, comienza en el $(0; 0)$ y avanza hacia el $(20; 0)$. El último punto de la trayectoria es el $(0; 2{,}86)$. En la figura 8 se muestra el resultado sin utilizar la red neuronal. En la figura 9a se muestra el resultado incorporando la red neuronal ya entrenada, con $L = 1$ y en la figura 9b con $L = 2$. En esta trayectoria se obsevra una mejora importante en el E_{RMS}, 50 % para $L = 1$ y 60 % para $L = 2$.

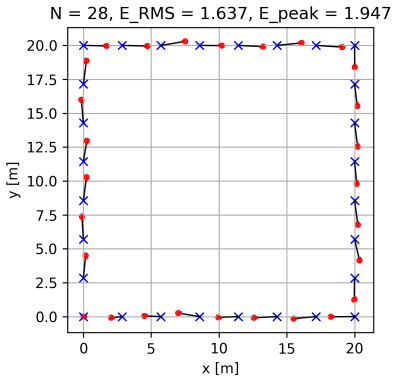
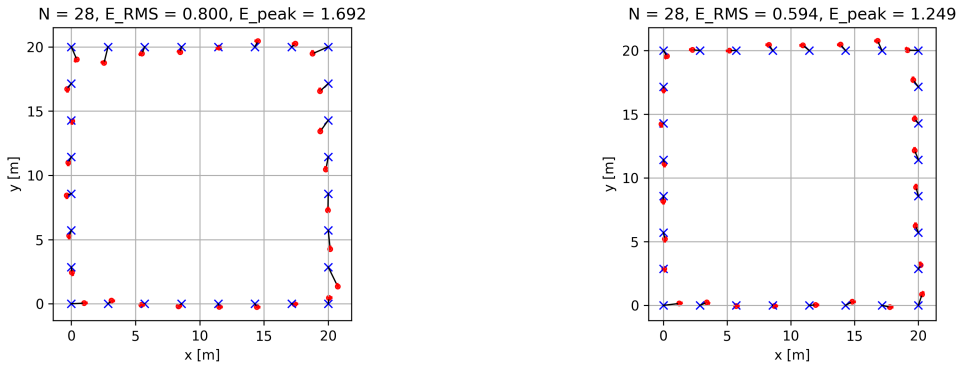


Figura 8: Resultados del caso 2 sin utilizar la red neuronal.



(a) $L = 1$ (b) $L = 2$

Figura 9: Se muestran los resultados obtenidos incorporando la red neuronal entrenada con datos de la trayectoria de la figura 8.

4.1.3. Trayectoria 3

Comienza en el $(0; 0)$ hacia el $(0; 20)$. Luego sigue hacia el $(5; 20)$, baja hasta el $(5; 0)$ y continua hacia el $(10; 0)$. Esta forma se repite un total de 5 veces. En la figura 10 se muestra el resultado sin utilizar la red neuronal. En la figura 11a se muestra el resultado incorporando la red neuronal ya entrenada, con $L = 1$ y en la figura 11b con $L = 2$.

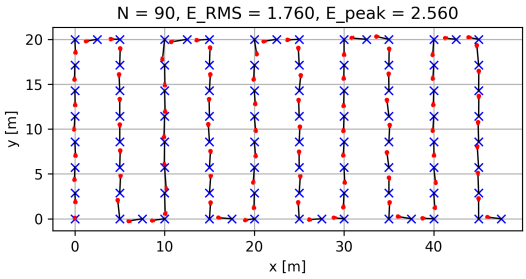
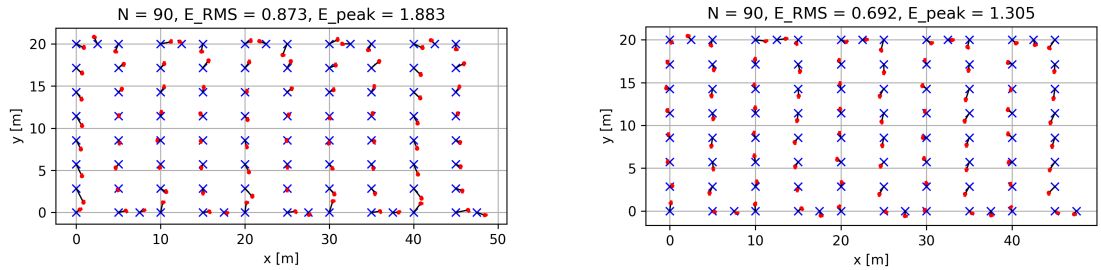


Figura 10: Resultados del caso 3 sin utilizar la red neuronal.



(a) $L = 1$ (b) $L = 2$

Figura 11: Se muestran los resultados obtenidos incorporando la red neuronal entrenada con datos de la trayectoria de la figura 10.

4.1.4. Trayectoria 4

En la figura 12 se muestra el resultado sin utilizar la red neuronal, en la figura 13a con $L = 1$ y en la figura 13b con $L = 2$.

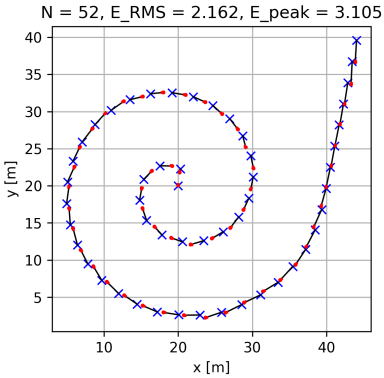
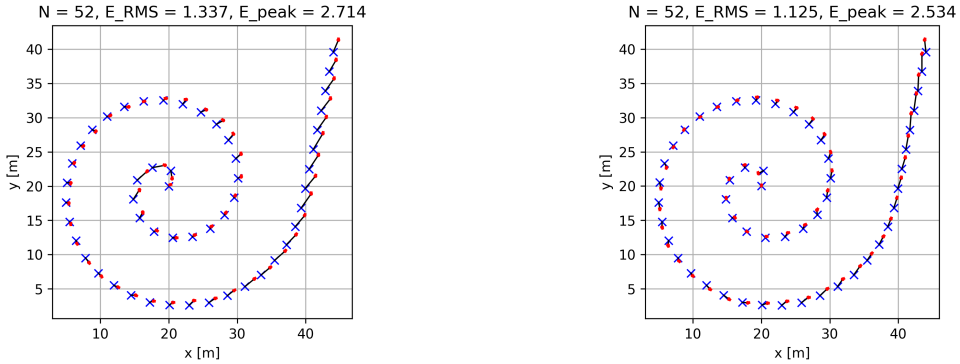


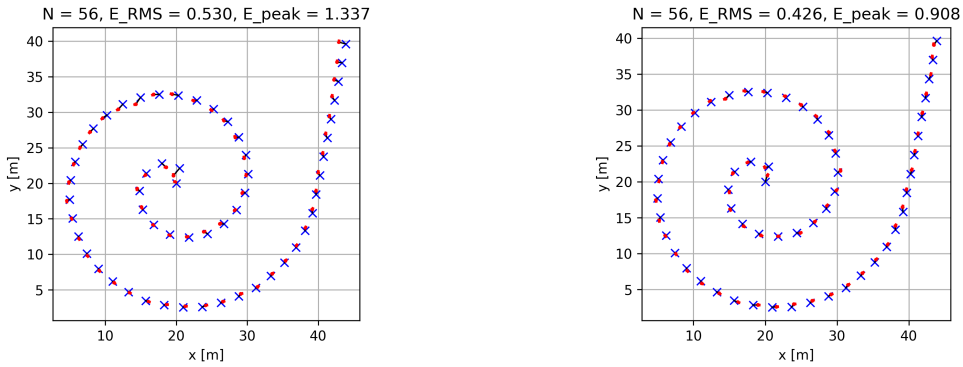
Figura 12: Resultado de la trayectoria 4 sin utilizar la red neuronal.



(a) $L = 1$ (b) $L = 2$

Figura 13: Se muestran los resultados obtenidos incorporando la red neuronal entrenada con datos de la trayectoria de la figura 12.

En ambos casos la con la incorporación de la red entrenada se logró corregir el error en los primeros puntos de la curva, cerca del $(20; 20)$, no así en los últimos. Se repitieron las simulaciones pero utilizando 5000 neuronas en la capa oculta. Los resultados se muestran en las figuras 14a y 14b. Esta vez sí se observa una mejora en toda la trayectoria.

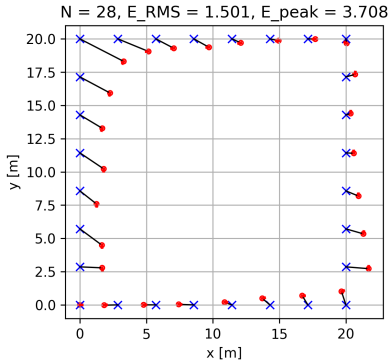


(a) $L = 1$ (b) $L = 2$

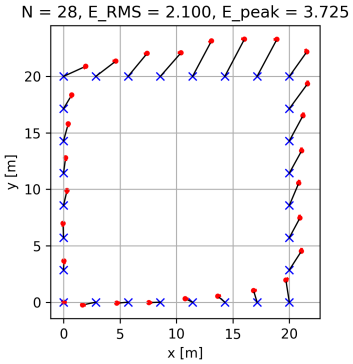
Figura 14: Se muestran los resultados para la trayectoria 4, esta vez con 5000 neuronas en la capa oculta.

4.2. Caso 2: Una Sola Red Para Todas Las Trayectorias

En el caso anterior se utilizó una red diferente para cada una de las trayectorias. En este caso lo que se quiere es entrenar una sola red neuronal que pueda utilizarse para todas las trayectorias. Dado que cada una de las redes del caso anterior fueron entrenadas con datos de sus respectivas trayectorias, es de esperarse que no se obtengan buenos resultados al utilizarlas para seguir una trayectoria diferente. En las figuras 15a y 15b se muestran dos casos en los que esto resulta evidente.



(a) Se utilizó la red entrenada con datos de la trayectoria 4.



(b) Se utilizó la red entrenada con datos de la trayectoria 1.

Figura 15: Para la trayectoria 2, se utilizaron dos redes neuronales diferentes con $L = 2$. En ambos casos se observa que no se obtienen resultados aceptables.

Lo que se propuso fue utilizar una trayectoria de entrenamiento con diferentes características, de $N = 400$ puntos. Esta se puede ver en la figura 16.

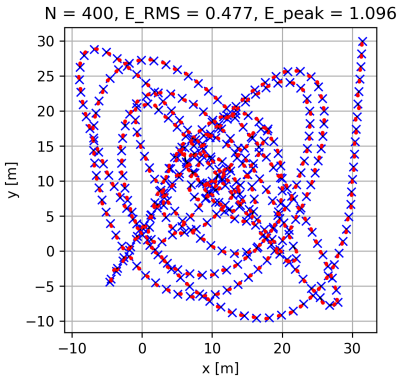
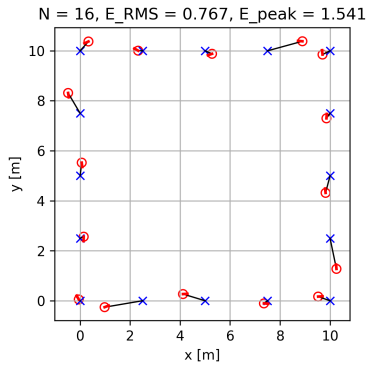


Figura 16: Trayectoria utilizada para entrenar a la red neuronal de este caso. El punto inicial se encuentra en el $(10, 10)$ y el punto final en el $(30, 30)$.

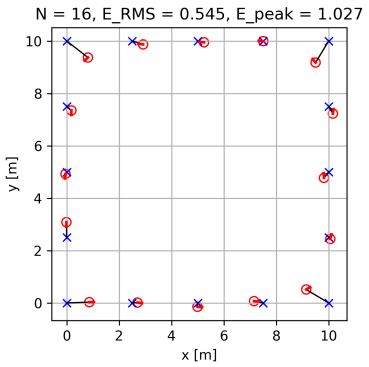
Se repitieron las simulaciones para las trayectorias 1 a 4 utilizando esta red entrenada con la trayectoria de la figura 16. A continuación se muestran los resultados obtenidos para $L = 1$ y $L = 2$.

4.2.1. Trayectoria 1

Se muestra el resultado para $L = 1$ en la figura 17a y $L = 2$ en la figura 17b.



(a) $L = 1$

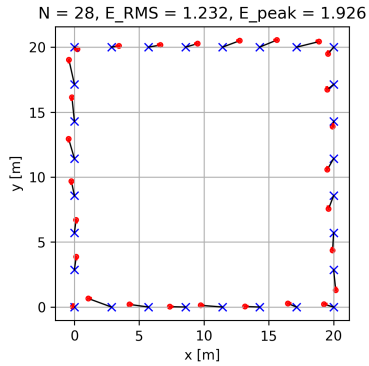


(b) $L = 2$

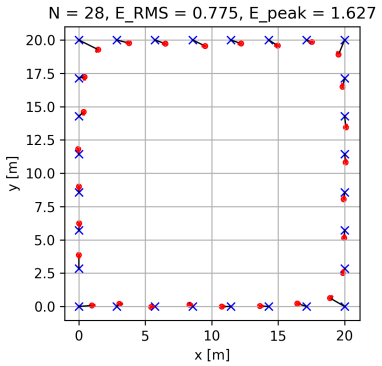
Figura 17: Resultados para la trayectoria 1, utilizando la red neuronal entrenada con la trayectoria de la figura 16.

4.2.2. Trayectoria 2

Se muestra el resultado para $L = 1$ en la figura 18a y $L = 2$ en la figura 18b.



(a) $L = 1$

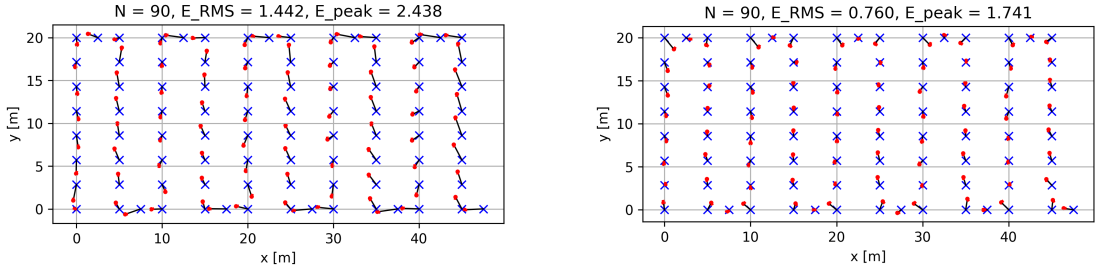


(b) $L = 2$

Figura 18: Resultados para la trayectoria 2, utilizando la red neuronal entrenada con la trayectoria de la figura 16.

4.2.3. Trayectoria 3

Se muestra el resultado para $L = 1$ en la figura 19a y $L = 2$ en la figura 19b.

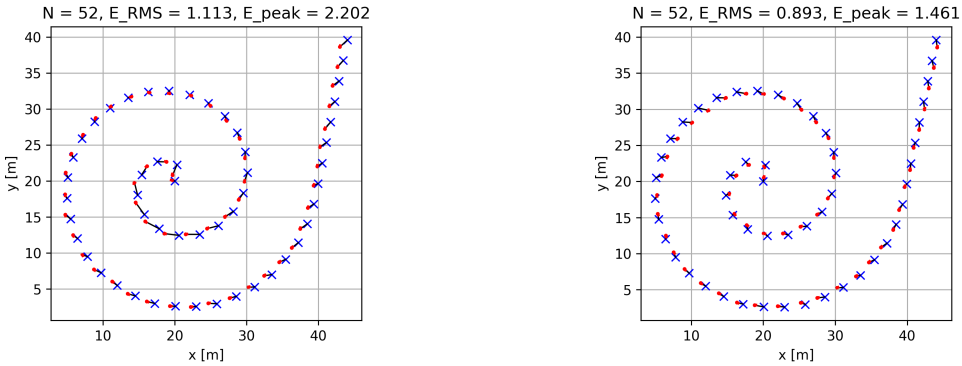


(a) $L = 1$ (b) $L = 2$

Figura 19: Resultados para la trayectoria 3, utilizando la red neuronal entrenada con la trayectoria de la figura 16.

4.2.4. Trayectoria 4

Se muestra el resultado para $L = 1$ en la figura 20a y $L = 2$ en la figura 20b.



(a) $L = 1$ (b) $L = 2$

Figura 20: Resultados para la trayectoria 4, utilizando la red neuronal entrenada con la trayectoria de la figura 16.

5. Resultados

5.1. E_{RMS} obtenidos

A continuación se presenta una tabla que resume los resultados obtenidos para cada simulación. La tabla 1 contiene el E_{RMS} de cada simulación. En cada fila de la tabla se encuentra coloreado en verde el resultado donde se obtuvo el error más bajo. Los valores que se muestran en cada casilla de la tabla se obtuvieron a partir de promediar los E_{RMS} de 5 simulaciones.

Trayectoria	Control clásico	Control clásico + DNN, L = 1	Control clásico + DNN, L = 2	Control clásico + DNN caso 2, L = 1	Control clásico + DNN caso 2, L = 2
1	1.166	0.646	0.589	0.869	0.556
2	1.637	0.799	0.578	1.233	0.771
3	1.760	0.846	0.685	1.433	0.770
4	2.162	1.336	1.245	1.117	0.901

Tabla 1: Comparación del E_{RMS} de las simulaciones realizadas para los distintos casos. Los resultados para la trayectoria 4 corresponden al caso con 1024 neuronas en la capa oculta.

Se observa que para todos los casos el error E_{RMS} es menor en las simulaciones donde se incluyó a la red neuronal respecto del caso donde solo se utilizó el controlador clásico. Esto ya se había mostrado anteriormente para cada caso en particular, lográndose así un mejoramiento de la precisión en el seguimiento de la trayectoria.

En todas las trayectorias se obtuvieron E_{RMS} más bajos para $L = 2$, respecto de los resultados con $L = 1$ (columna 4 vs columna 3 y columna 6 vs columna 5 de la tabla 1). Este comportamiento se asemeja al de un controlador predictivo, donde al incorporar una mayor cantidad de estados deseados a futuro en la ley de control, se obtiene un menor error.

En las trayectorias 2 y 3 las simulaciones utilizando la red del caso 2 presentaron mejoras respecto del controlador clásico, aunque los mejores resultados se obtuvieron con la red entrenada con datos de la trayectoria correspondiente, para $L = 2$. Esto era de esperarse, debido a que las redes neuronales utilizadas en las simulaciones de la columna 4 de la tabla 1 fueron entrenadas únicamente con datos de sus respectivas trayectorias. Como ya se mostró, estas redes no generan buenos resultados al utilizarse con otras trayectorias, a diferencia de la red del caso 2, la cual fue entrenada con una trayectoria de más puntos y de características variadas (trayectoria de la figura 16).

En la trayectoria 1, el E_{RMS} que se obtuvo utilizando la red entrenada con datos de la trayectoria 1, para $L = 2$, resultó ser algo mayor al que se obtuvo con la red del caso 2 (última columna de la tabla 1), aunque hay una muy pequeña diferencia entre ambos, de 0,033.

En cuanto a la trayectoria 4, los mejores resultados se obtuvieron utilizando la red neuronal del caso 2, para $L = 2$. No se cumple lo ocurrido con las trayectorias 2 y 3, donde el mejor resultado se logró entrenando con los datos de la misma red neuronal. Esto principalmente se debe al aporte de los últimos puntos de la trayectoria al error E_{RMS}, como ya se pudo ver en las figuras 13a y 13b. Es probable que la red haya quedado en un mínimo local durante el entrenamiento. Luego al repetir las simulaciones incrementando la cantidad de neuronas de la capa oculta, se pudo evitar este mínimo local y obtener un mejor resultado (figuras 14a y 14b). Si se consideran estos últimos resultados, entonces también para esta trayectoria se cumple el hecho de que el mejor resultado se obtiene para $L = 2$ entrenando con datos de la misma trayectoria, al igual que ocurrió con las trayectorias 2 y 3.

5.2. E_{peak} obtenidos

En la tabla 2 se muestran los E_{peak} de cada simulación. Los valores que se muestran en cada casilla de la tabla se obtuvieron a partir de promediar los E_{peak} de 5 simulaciones.

Trayectoria	Control clásico	Control clásico + DNN, L = 1	Control clásico + DNN, L = 2	Control clásico + DNN caso 2, L = 1	Control clásico + DNN caso 2, L = 2
1	1.671	1.221	1.037	1.718	1.056
2	1.947	1.589	1.230	2.032	1.615
3	2.560	1.846	1.353	2.400	1.623
4	3.105	2.695	2.505	2.183	1.509

Tabla 2: Comparación del E_{peak} de las simulaciones realizadas para los distintos casos. Los resultados para la trayectoria 4 corresponden al caso con 1024 neuronas en la capa oculta.

Al igual que sucede con el E_{RMS}, en todos los casos el error E_{peak} es menor en las simulaciones donde se incluyó a la red neuronal respecto del caso donde solo se utilizó el controlador clásico.

En el caso particular de la columna 5, el E_{peak} en las trayectorias 1, 2 y 3 no presenta una mejora relevante respecto del caso donde solamente se utilizó el controlador clásico. En el caso de la tryaectoria 2 hay un aumento del error E_{peak} respecto del controlador clásico. Esto ocurre debido a que se está utilizando una red neuronal con $L = 1$ por lo que se le está entregando solamente 1 estado próximo deseado como entrada a la red neuronal. Esto genera problemas en situaciones donde el robot debe recorrer una curva pronunciada. En la figura 21 se destaca la esquina inferior derecha de las figuras 18a y 18b.

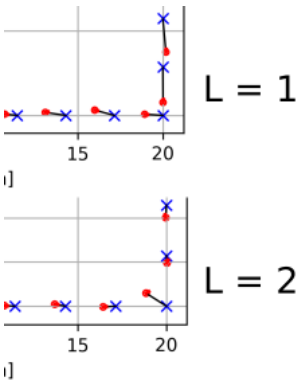
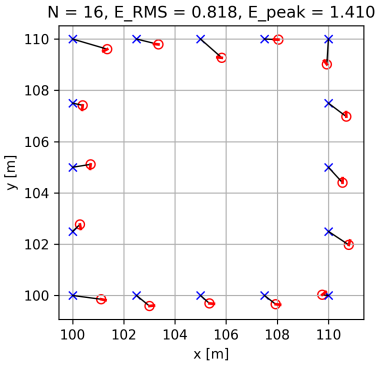


Figura 21: Se muestra parte del recorrido del robot en la trayectoria 2, para los casos con $L = 1$ y $L = 2$.

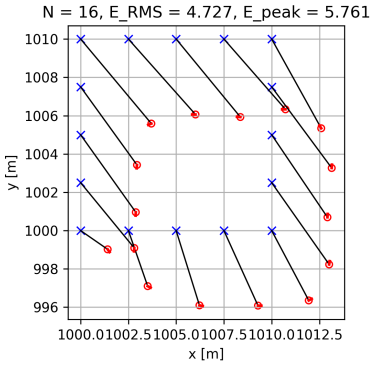
En el caso con $L = 1$, cuando el robot debe dirigirse al punto $(20; 0)$, este lo hace en línea recta. Luego cuando se cambia a la próxima referencia, el robot no puede llegar a tiempo a esta nueva referencia, debido a que debe hacer una rotación de 90°. En cambio, para $L = 2$, el robot no se acerca al $(20; 0)$ en línea recta, ya que la red neuronal recibe como entrada dos puntos próximos deseados, generando que el robot recorra una curva suave y que se obtenga un error más bajo. Esto se vio reflejado en la columna 6 de la tabla 2, donde se ve que los E_{peak} que se obtuvieron para $L = 2$ son más bajos que los del caso con $L = 1$.

6. Limitaciones del Sistema Propuesto: Trayectoria Desplazada en el Espacio

En las figuras 22a y 22b se muestran los resultados de utilizar la red del caso 2, con $L = 2$, para seguir trayectorias iguales a la trayectoria 1, pero desplazadas 100 en x y 100 en y para la figura 22a y 1000 en x y 1000 en y para la figura 22b. Dado que en ambos casos se está queriendo seguir una trayectoria como la 1 pero desplzada en el espacio, el robot debería poder recorrerla sin problemas, como ya hizo en la figura 17b. Sin embargo esto no ocurre.



(a) Desplazamiento en $(100; 100)$.



(b) Desplazamiento en $(1000; 1000)$.

Figura 22: Resultados para la una trayectoria idéntica a la trayectoria 1, pero desplazada 1000 en x y 1000 en y. Se muestran el E_{RMS} y el E_{peak} en cada caso.

Si bien para el resultado con un desplazamiento de 100 se obtuvieron errores superiores a los que se obtuvieron en la figura 17b, la trayectoria recorrida por el robot resulta aceptable. Para el caso con desplazamiento de 1000, los errores obtenidos son aún mayores.

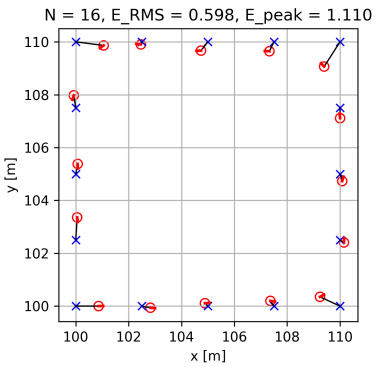
Las entradas de la red neuronal son las poses deseadas y la pose actual del robot. Si estas poses tienen valores muy grandes, es probable que algunas neuronas de la capa oculta estén generando salidas de valores muy grandes, lo que puede estar causando salidas indeseadas. Esto puede entenderse como una limitación del perceptrón multicapa utilizado como bloque para generar referencias en cualquier punto

del plano (x, y), en la figura 2.

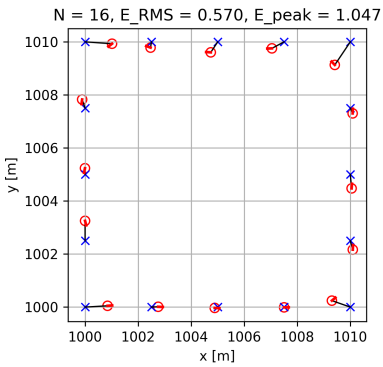
Este problema puede solucionarse fácilmente si se modifican las entradas de la red para trabajar con entradas diferenciales respecto al punto del espacio donde se encuentra el robot en cada instante de tiempo. Si a cada una de las entradas de la red se le resta la parte de posición de $\overrightarrow{X(t)}$, es decir un vector de la forma:

$$\overrightarrow{p(t)} = \begin{bmatrix} x(t) \\ y(t) \\ 0 \end{bmatrix} \tag{4}$$

luego esto equivale a trabajar con un robot que siempre se encuentra en el punto $(0; 0)$ del plano, lo cual asegura que las entradas de la red no tendrán valores grandes. A la salida entregada por la red neuronal se le debe volver a sumar el vector $\overrightarrow{p(t)}$. En las figuras 23a y 23b se muestran los resultados de las mismas simulaciones de las figuras 22a y 22b utilizando este método. En ambos casos los resultados son similares a los que se obtuvieron para la misma trayectoria sin desplazar.



(a) Desplazamiento en $(100; 100)$.



(b) Desplazamiento en $(1000; 1000)$.

Figura 23: Mismos resultados para las trayectorias desplazadas pero utilizando una entrada diferencial en la red.

7. Conclusiones

En este trabajo se mostró, a partir de simulaciones, el uso de una red neuronal para mejorar la presición en el seguimiento de una trayectoria para un robot diferencial. Partiendo de un sistema con un controlador clásico muy sencillo y a partir de los datos recolectados de trayectorias del robot, se pudo entrenar una red neuronal que mejora la precisión, en lugar de utilizar un controlador clásico que requiera un gran trabajo de diseño.

Referencias

[1] Qiyang Li y col. "Deep neural networks for improved, impromptu trajectory tracking of quadrotors". En: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, págs. 5183-5189.

[2] Javier Luiso. *pRoboticsSim*. `https://gitlab.com/javierluiso/proboticssim`. 2020.

[3] Sebastian Thrun, Wolfram Burgard y Dieter Fox. "Intelligent robotics and autonomous agents". En: *Probabilistic robotics, ser.* Mit Press, 2005.

[4] Frederico C Vieira y col. "Position and Orientation Control of a Two-Wheeled Differentially Driven Nonholonomic Mobile Robot." En: *ICINCO (2)*. 2004, págs. 256-262.

[5] Wikipedia contributors. *Differential wheeled robot — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Differential_wheeled_robot&oldid=1000825516`. [Online; accessed 12-October-2021]. 2021.