PLANIFICACIÓN DE MOVIMIENTO MEDIANTE ALGORITMO DE CAMPOS POTENCIALES CON PARÁMETROS OPTIMIZADOS MEDIANTE REDES NEURONALES APLICADO A UN MANIPULADOR ROBÓTICO DE 6 GDL

PATH PLANNING USING POTENTIAL FIELD ALGORITHMS WITH OPTIMIZED PARAMETERS THROUGH NEURAL NETWORKS APPLIED TO A 6-DOF ROBOTIC MANIPULATOR

Sandro Alcántara Tacora[1*], Erwin López Zapata[1], Jesús Peralta Toribio[1], Ricardo Rodriguez Bustinza[1]

[1]Facultad de Ingeniería Mecánica, Universidad Nacional de Ingeniería, Lima, Perú.

Recibido (Recieved): 19 / 02 / 2020 Aceptado (Accepted): 03 / 10 / 2020

RESUMEN

Este trabajo de investigación presenta un nuevo algoritmo orientado a la evasión de obstáculos en un manipulador robótico de 6 grados de libertad basados en algoritmos de campos potenciales. En primer lugar, este manipulador fue desarrollado usando software CAD con el propósito de que este modelo sería nuestro modelo base para el desarrollo del algoritmo propuesto. Luego se desarrolla el modelo de cinemática inversa usando un proceso de control iterativo multivariable. Después se modificó el modelo matemático mediante la adición de un vector de rotación. Este vector se obtuvo a partir de fuerzas repulsivas entre los obstáculos y las seis articulaciones del manipulador robótico. Como consecuencia, el manipulador pudo encontrar múltiples rutas que llegaban al punto final deseado y que evadía todos los posibles obstáculos en su camino. Con el fin de optimizar dichas trayectorias, se construyó una base de datos de diferentes trayectorias. Esta base de datos contiene las trayectorias que dependen en la posición inicial, final y las coordenadas de los obstáculos, con los hiperparámetros de dichas trayectorias optimizadas. Finalmente, las simulaciones han mostrado que el manipulador pudo completar la tarea de llegar a un punto deseado sin atravesar los obstáculos.

Palabras Clave: Planificación de movimiento, campos potenciales, red neuronal supervisada, cinemática inversa, manipulador robótico

ABSTRACT

This paper presents a new algorithm for obstacle avoidance tasks applied to a robotic manipulator of 6 degrees of freedom based on potential field algorithms. Firstly, this manipulator was designed using CAD software and it was intended to be used as our base model to develop the proposed algorithm. Then, the inverse kinematics model was developed using a multivariate iterative control process. Afterwards, the mathematical model was modified by adding a rotational vector. This vector was obtained by the repulsive forces between the obstacle and the six joints of the robot manipulator. Therefore, the manipulator was able to find possible routes that reached the final desired coordinate and avoided any possible obstacle along its path. To optimize these trajectories, a database was created of different trajectories. This database contains trajectories that depend on the initial, final and the obstacle coordinates, with the trajectories hyperparameters optimized. Finally, the simulations have shown that the manipulator was able to complete the task of reaching a point without falling in the obstacle points.

Keywords: Path planning, Potential field, Supervised Neural Network, Inverse Kinematics, Robotic Manipulator

1. INTRODUCCION

La planificación de movimiento consiste en encontrar una ruta libre de colisión desde una configuración inicial de posición y orientación hasta una configuración final [1].

Muchas de las tareas en robótica hoy en día requieren de una planificación del movimiento que se tiene que realizar antes de ejecutar dichos movimientos con el fin de hacer que estas tareas complejas se ejecuten de una forma segura y autónoma. Por eso, en el estado del arte se encuentra una gran cantidad de artículos relacionados a este tema.

Uno de los primeros trabajos iniciales de investigación acerca de algoritmos de evasión de

* Corresponding author:
E-mail: salcantarat@uni.pe

obstáculos fue desarrollado por Johan Borenstein et al [2] en el cual usa campos potenciales y fusión de sensores para que un robot móvil recorra un trayecto y no se detenga abruptamente ante un obstáculo. Zemin Liu et al [3] desarrolla una variación del algoritmo de árboles aleatorios estrella usando campos potenciales artificiales con el fin de reducir el tiempo de ejecución del algoritmo y obtener un tiempo de ejecución menor.

En el caso de manipuladores antropomórficos que realizan tareas de "pick and place", los objetos a manipular presentan diferentes ubicaciones iniciales y requieren ser colocados en distintas posiciones objetivo, debido a esta variabilidad en los procesos se requieren de algoritmos que puedan generar trayectorias de manera autónoma.

Existen métodos de planificación de movimiento que están basados en el concepto de espacio de configuraciones como son las técnicas de hoja de ruta [4] y los algoritmos de descomposición de celdas [5], que tradicionalmente se usan para resolver problemas de planificación en espacios de configuraciones de baja dimensionalidad siendo computacionalmente ineficiente para casos de alta dimensionalidad, como los manipuladores robóticos [6]. Otros métodos están basados en el espacio de trabajo que consisten en encontrar una ruta en el espacio geométrico euclidiano. Por ejemplo, el método de campos potenciales utiliza el espacio de trabajo donde modela campos potenciales artificiales debido a los polos de repulsión que son los obstáculos y un polo de atracción que es el punto objetivo, de esta manera se logra que el robot pueda moverse de una forma segura hasta su posición final [1].

El método de campos potenciales es ampliamente utilizado en el campo de navegación de robots móviles debido a su fácil implementación [7]. Sin embargo, también han sido implementados en simulación de manipuladores altamente redundantes [8], [9]. Asimismo, Rajendran [10] desarrolla un método de planificación de movimiento aplicado a manipuladores robóticos que se encuentran en espacios desorganizados y con muchos obstáculos. Su técnica se basa en aproximaciones locales de su espacio de configuraciones y también se basa en arboles de búsquedas.

En este trabajo de investigación, se presenta en primer lugar el modelamiento de la cinemática directa del modelo mecánico del manipulador de seis grados de libertad propuesto mediante el método Denavit-Hartenberg. Luego, se realiza la cinemática inversa del modelo mediante un método iterativo que optimiza una función de costo con el fin de tener una solución única; el cual es regulado mediante un hiperparámetro. Luego se realiza la cinemática inversa planificada mediante la adición de 6 fuerzas repulsivas a las seis juntas y se define su función de costo, así como su representación en la arquitectura general del algoritmo. Luego se genera empíricamente una base de datos de trayectorias optimizados en base a tres clases de pérdida. Finalmente se realiza las simulaciones usando la arquitectura construida y se presenta al final los resultados y conclusiones del trabajo de investigación.

2. METODOLOGÍA

2.1 Modelamiento Geométrico Directo

El modelo de Geometría Directa tiene como propósito generar una función capaz de calcular el vector de posición-orientación del efector final del robot en base a los valores de rotación de los 6 motores del robot. Como primer paso, necesitamos definir las relaciones matemáticas entre los diferentes vectores de posición correspondientes a las uniones del robot y la configuración correspondiente del robot mediante las matrices de Denavit-Hartenberg [11].

Podemos definir la función DH como la matriz Denavit-Hartenberg como:

$$DH(\theta,d,a,\alpha)=\begin{bmatrix}\cos(\theta) & -\sin(\theta).\cos(\alpha) & \sin(\theta).\sin(\alpha) & a.\cos(\theta)\\ \sin(\theta) & \cos(\theta).\cos(\alpha) & -\cos(\theta).\sin(\alpha) & a.\sin(\theta)\\ 0 & \sin(\alpha) & \cos(\alpha) & d\\ 0 & 0 & 0 & 1\end{bmatrix} \quad (1)$$

De esta forma, las matrices de transformación se definirán como:

$$\begin{aligned} A_{01}(q_1) &= DH(q_1, L_1, 0, {}^{\pi}/_{2})\\ A_{12}(q_2) &= DH(q_2, 0, L_2, 0)\\ A_{23}(q_3) &= DH(q_3, 0, L_3, 0)\\ A_{34}(q_4) &= DH(0,0, L_4, q_4)\\ A_{45}(q_5) &= DH(q_5, 0, L_5, 0)\\ A_{56}(q_6) &= DH(0,0, L_6, q_6) \end{aligned} \quad (2)$$

Luego, la función que calcula la matriz de transformación del efector final a partir de los valores de rotación de los 6 motores se definirá como:

$$T_{06}(q_1,q_2,q_3,q_4,q_5,q_6) = T_0.A_{01}(q_1).A_{12}(q_2).A_{23}(q_3).A_{34}(q_4).A_{45}(q_5).A_{56}(q_6) \quad (3)$$

De esta matriz de transformación podemos obtener el vector de posición cartesiana y el vector de orientación (considerando las orientaciones de Yaw, Pitch y Roll) de la forma que se define en la ecuación (4).

$$[x; y; z] = T_{06}|_{1:3,4}$$
$$[\varphi; \theta; \phi] = \begin{cases} \left[tan^{-1}\left(\frac{T_{06}|_{3,2}}{T_{06}|_{3,3}}\right); sin^{-1}(T_{06}|_{3,1}); tan^{-1}\left(\frac{T_{06}|_{3,2}}{T_{06}|_{3,3}}\right)\right], T_{06}|_{3,2} \neq 0 \wedge T_{06}|_{3,2} \neq 0 \\ \left[0; \frac{\pi}{2}; tan^{-1}\left(\frac{T_{06}|_{2,3}}{T_{06}|_{1,3}}\right)\right], T_{06}|_{3,2} = 0 \wedge T_{06}|_{3,2} = 0 \end{cases} \quad (4)$$

Finalmente, podemos definir el vector de posición del efector como el vector $x^{\rightarrow} = [x; y; z; \varphi; \theta; \emptyset]$, que podría calcularse a partir de la matriz $T06$ y por lo tanto a partir de los 6 valores de rotación de los motores. La arquitectura de información del procesamiento de señales se muestra en la figura 1.

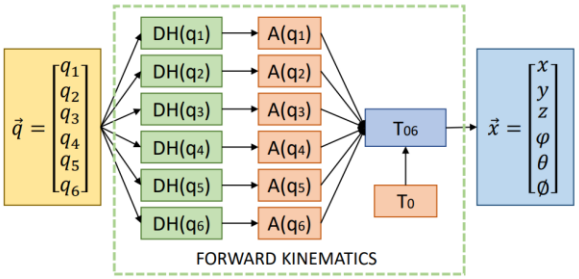

Fig. 1. Arquitectura de procesamiento de señales del modelo de cinemática directa

2.2 Modelamiento Geométrico Inverso

Se propone un método iterativo, que optimiza la ecuación diferencial que define la relación entre los vectores $q^{\rightarrow}$ y $x^{\rightarrow}$. Por esta razón, este método encuentra solo una solución que cambiaría según el valor del hiperparámetro "velocidad de optimización" seleccionado y la configuración inicial del robot. La ecuación diferencial se define en (5).

$$J.d\vec{q} = d\vec{x} \quad (5)$$

Donde J es la matriz jacobiana definida como:

$$J = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{dx}{dq_6} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_6} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \phi}{\partial q_1} & \frac{\partial \phi}{\partial q_2} & \cdots & \frac{\partial \phi}{\partial q_6} \end{bmatrix} \quad (6)$$

Para optimizar esta ecuación diferencial, definimos la variable de pérdida S como:

$$S = (d\vec{x} - J.d\vec{q})^T(d\vec{x} - J.d\vec{q})$$
$$S = (d\vec{x})^T(d\vec{x}) - (d\vec{q})^T(J^T d\vec{x}) - (d\vec{x})^T J d\vec{q} + (d\vec{q})^T(J^T J)(d\vec{q}) \quad (7)$$

Como en el modelo de geometría inversa se calcula iterativamente los valores necesarios de $q^{\rightarrow}$ para alcanzar un $x^{\rightarrow}$ establecido, se minimiza el valor de S modificando los valores de $dq^{\rightarrow}$, por lo tanto, se define la optimización de la variable de pérdida S con respecto a la variable $dq^{\rightarrow}$ de acuerdo a la siguiente ecuación diferencial parcial:

$$\frac{\partial S}{\partial dq_j} = 0 \quad , j = 1,2, \ldots,6 \quad (8)$$

Luego tras reemplazar el valor de S en esta ecuación de optimización:

$$(J^T J)d\vec{q} = J^T d\vec{x}$$
$$d\vec{q} = (J^T J)^{-1} J^T d\vec{x} \quad (9)$$

Donde a $J+ = (JTJ)-1JT$ se le denomina el pseudoinverso de la matriz jacobiana, porque transforma la primera ecuación diferencial en la siguiente ecuación diferencial:

$$d\vec{q} = J^+ d\vec{x} \quad (10)$$

Luego, para calcular los valores necesarios de $q^{\rightarrow}$, se define el vector objetivo como $x^{\rightarrow}SP$ y se realiza un proceso iterativo definido de la siguiente forma:

$$\Delta\vec{q}^n = \alpha J^+(\vec{x}_{SP} - \vec{x}_0)$$
$$\vec{q}^{n+1} = \vec{q}^n + \overrightarrow{\Delta q}^n \quad (11)$$

Donde α es el hiperparámetro del modelo de geometría inversa que establece la velocidad de la optimización.

El único problema con este enfoque es el cálculo de la matriz jacobiana J, ya que requiere soluciones analíticas de todos los elementos de la matriz. Sin embargo, con el modelo de la geometría directa definido en el apartado anterior, se puede calcular indirectamente una aproximación de esta matriz. En la figura 2 se muestra cómo usar este modelo para calcular los valores aproximados de los valores diferenciales parciales del vector $x^{\rightarrow}$ con respecto al valor diferencial del vector $q^{\rightarrow}$.

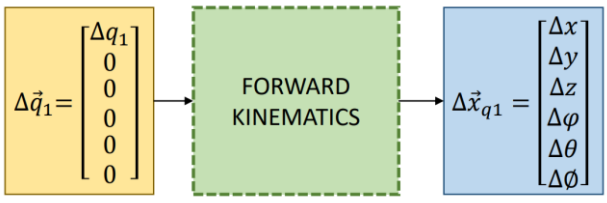

Fig. 2. Aproximación del vector derivada parcial $\partial x^{\rightarrow}/\partial q1$ utilizando el modelo de Geometría Directa.

Por consiguiente, se calcula la matriz jacobiana aproximada repitiendo el proceso que se muestra en la figura 2 con las variaciones de todos los valores de rotación de los motores. Así, se obtiene:

$$J \approx \left[\frac{\Delta\vec{x}_{q1}}{\|\Delta\vec{q}_1\|} \; \frac{\Delta\vec{x}_{q2}}{\|\Delta\vec{q}_2\|} \; \cdots \; \frac{\Delta\vec{x}_{q6}}{\|\Delta\vec{q}_6\|}\right] \quad (12)$$

Finalmente, podemos aplicar las ecuaciones iterativas definidas anteriormente para actualizar los valores del vector $q^{\rightarrow}$ hasta que el sistema alcance el vector deseado $x^{\rightarrow}$. La arquitectura de información del procesamiento de señales se muestra en la figura 3.

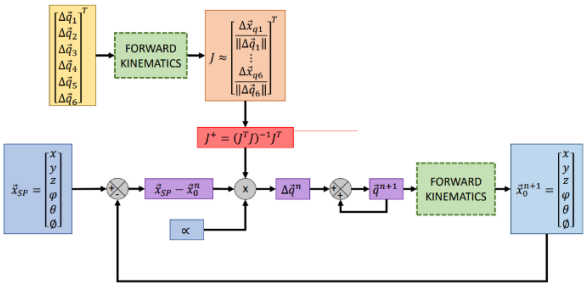

Fig. 3. Arquitectura de procesamiento de señales del modelo de cinemática inversa.

2.3 Modelamiento Geométrico Planificado

Ya que el enfoque elegido para el modelo de geometría inversa se basa en un proceso iterativo y se define como un bucle de control multivariable (figura 3), se puede modificar este modelo para permitir que el sistema encuentre un camino que evite tanto un obstáculo como un alcance el vector requerido $x^{\rightarrow}$. Para este propósito, se utiliza como base el modelo de planificación de la trayectoria de campos potenciales para desarrollar un modelo de Geometría Inversa Planificada.

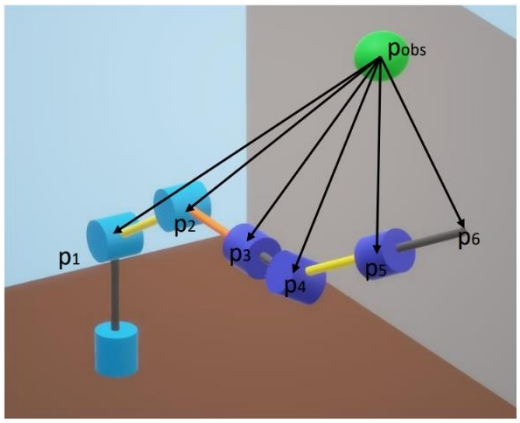

Fig. 4. Representación de las distancias entre las uniones del robot y un obstáculo.

Se definen 6 fuerzas de repulsión de las 6 distancias mostradas en la figura 4. Cada una de estas distancias considera un punto correspondiente al extremo de una de las articulaciones del robot y el punto de la posición del obstáculo (ecuación 13).

$$d_j = \|p_j - p_{obs}\| \quad (13)$$

Luego, se define la fuerza repulsiva de cada distancia como:

$$F_j = \begin{cases} \frac{1}{2} n_j \left(\frac{1}{d_j} - \frac{1}{\rho}\right)^2, & d_j \le \rho \\ 0, & d_j > \rho \end{cases} \quad (14)$$

En consecuencia, la derivada parcial de la fuerza de repulsión respecto de las coordenadas xyz (sin considerar las orientaciones) es:

$$\frac{dF_j}{d\vec{x}} = \begin{cases} n_j \left(\frac{1}{\rho} - \frac{1}{d_j}\right) \frac{(p_j - p_{obs})}{d_j^{\,3}}, & d_j \le \rho \\ \vec{0}, & d_j > \rho \end{cases} \quad (15)$$

Asimismo, se define las matrices parciales Jacobianas para los 6 diferentes puntos del robot como:

$$J_{Fq(j)} = \left(\frac{dF_j}{d\vec{x}}\right)^T \begin{bmatrix} \frac{\partial x_j}{\partial q_1} & \frac{\partial x_j}{\partial q_2} & \cdots & \frac{\partial x_j}{\partial q_6} \\ \frac{\partial y_j}{\partial q_1} & \frac{\partial y_j}{\partial q_2} & \cdots & \frac{\partial y_j}{\partial q_6} \\ \frac{\partial z_j}{\partial q_1} & \frac{\partial z_j}{\partial q_2} & \cdots & \frac{\partial z_j}{\partial q_6} \end{bmatrix} \quad (16)$$

Para calcular estas matrices Jacobianas parciales, se procede a calcular de una manera similar al método para calcular la matriz del Jacobiano para el modelo de geometría inversa, que se describió anteriormente. Entonces se tiene:

$$J_{Fq(j)} \approx \left(\frac{dF_j}{d\vec{x}}\right)^T \left[\frac{\Delta\vec{x}_{q1}}{\|\Delta\vec{q}_1\|} \; \frac{\Delta\vec{x}_{q2}}{\|\Delta\vec{q}_2\|} \; \cdots \; \frac{\Delta\vec{x}_{q6}}{\|\Delta\vec{q}_6\|}\right] \quad (17)$$

De acuerdo con la definición de fuerzas repulsivas se tiene la siguiente ecuación:

$$\Delta\vec{q}_{rep} = -\sum_{j=1}^{6} J_{Fq(j)} \quad (18)$$

Donde el vector $\Delta q^{\rightarrow} rep$ define los valores necesarios para cambiar en los valores de rotación de 6 motores para que la posición actual del robot no choque con el obstáculo.

Finalmente, hemos definido como modelo de atracción el modelo de geometría inversa. Por lo tanto, el proceso iterativo será definido por la regla:

$$\vec{q}^{\,n+1} = \vec{q}^{\,n} + \Delta\vec{q}^{\,n} + \Delta\vec{q}^{\,n}_{rep} \quad (19)$$

Luego, la arquitectura de información del procesamiento de señales se muestra en la figura 5. Se puede observar el sistema de control del robot y una compensación debido a la evasión de obstáculo requerida.

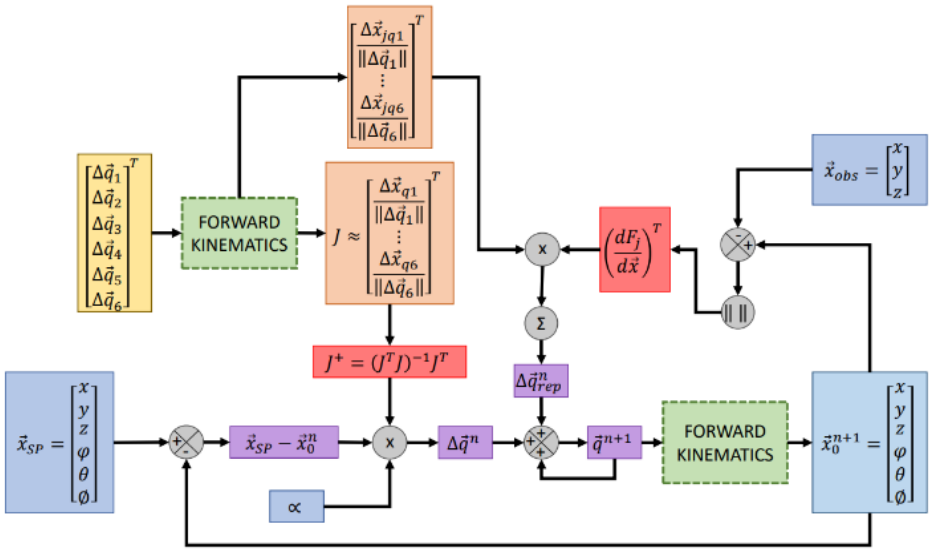

Fig. 5. Arquitectura de procesamiento de señales del modelo de cinemática inversa planificada.

2.4 Generación de la Red Neuronal Supervisada

Para realizar el entrenamiento de la red neuronal supervisada se generó una base de datos con trayectorias optimizadas empíricamente en base a tres clases de pérdidas: pérdidas por energía potencial, pérdidas por precisión de posicionamiento y pérdidas por el campo de repulsión.

Con el modelo de geometría inversa planificada por campos potenciales se generaron 2000 trayectorias con los mismos vectores de posicionamiento (posiciones y orientaciones inicial y final) y posición del obstáculo en los cuales se varió los hiperparámetros de velocidad de movimiento y los coeficientes de campos vectoriales. Luego se realizó la simulación y se obtuvieron sus respectivos valores de pérdidas. Después se seleccionó solo una trayectoria con la menor pérdida relativa respecto a las otras, considerándose esta la trayectoria con parámetros optimizados.

La base de datos se generó repitiendo el proceso anterior, debido a la gran carga computacional para generar dicha base de datos, se ejecutaron las simulaciones en paralelo en 4 computadoras del Centro de Cálculo de UNIFIM (Instituto de investigación de la Universidad Nacional de Ingeniería, Perú).

Con la base de datos generada, se desarrolló la programación del modelo de Red Neuronal Supervisada. Teniendo como datos de entrada los vectores de posicionamiento (X0, Xf y Xobs) y un conjunto de variables de salida (alpha, n1, n2, n3, n4, n5, n6) se prepararon los datos en Python con la librería pandas. Luego, empleando la librería de sckit-learn [12], sklearn se definió la red neuronal supervisada. Los hiperparámetros empleados fueron los que se muestran a continuación en la definición del modelo:

```
MLPRegressor(hidden_layer_sizes=(10),
verbose = True, activation = 'relu', solver =
'adam', alpha = 0.0001, batch_size = 'auto',
learning_rate = 'constant', learning_rate_init =
0.05, max_iter = 10000, tol = 0.0001, momentum
= 0.9, early_stopping = False, epsilon = 1e-08)
```

El proceso de la generación de trayectorias con la aplicación de la red neuronal se visualiza en el siguiente diagrama (figura 6). Donde, para realizar una trayectoria, inicialmente se introducen los vectores de posicionamiento y la posición del obstáculo, luego la red neuronal calcula los parámetros óptimos para dicha trayectoria que son ingresados al modelo de geometría inversa para ejecutar el movimiento en simulación.

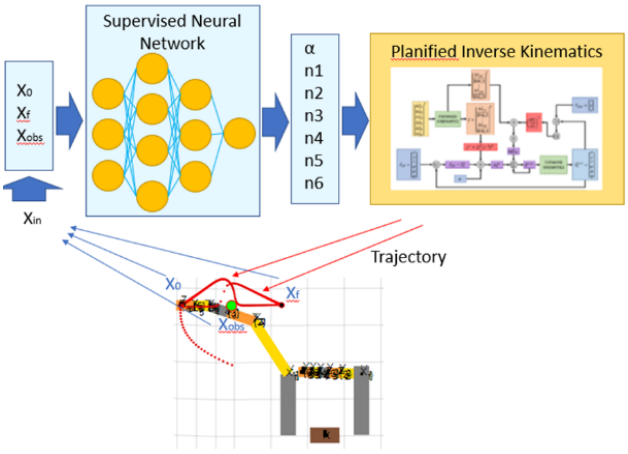

Fig. 6. Flujo de procesamiento de datos de posicionamiento requerido para generación automática de trayectorias.

3. SIMULACION Y RESULTADOS

3.1 Aplicación de la Red Neuronal Supervisada

Se realizó el entrenamiento de la red neuronal. Los resultados se muestran en la tabla I, donde se observa que la variable R2 correspondiente a la correlación entre las variables tiene un valor de 0.8184 aproximadamente.

TABLA I
Resultados tras el entrenamiento de la Red Neuronal Supervisada.

R^2 score	0.8184
Función de costo	21.7614

Por otro lado, la importancia que tiene cada variable en el poder de predicción de este modelo se muestra en la siguiente lista de "features importance" (tabla II) elaborada con un modelo de random forest auxiliar, de forma que se pueda obtener un peso por cada variable de entrada.

TABLA II
"Features importance" del conjunto de datos empleado en el entrenamiento.

Name	Score
Xo(1)	0.229275
Xo(3)	0.228698
Xo(2)	0.14784
Xf(3)	0.0656432
Xf(2)	0.0635456
Xobs(3)	0.0626281
Xobs(2)	0.0583687
Xf(1)	0.038695
Xobs(1)	0.0347584
Xf(4)	0.0244464
Xo(4)	0.0227653
Xf(6)	0.0130426
Xo(6)	0.0102939
Xo(5)	0
Xf(5)	0

En cuanto a la aplicación para la selección de los hiperparámetros optimizados se ingresaron los siguientes vectores de posicionamiento para cuatro trayectorias de prueba:

TABLA III
Vectores de posicionamiento para generar la trayectoria

x_tray_1	np.array([45,-32,-10,80,0,0,45,-32,34,80,0,0,45,-32,22])
x_tray_2	np.array([45,-32,34,80,0,0,35,-5,35,0,0,-30,30,-25,35])
x_tray_3	np.array([35,-5,35,0,0,-30,35,8,10,0,0,-30,35,0,25])
x_tray_4	np.array([35,-32,10,0,0,-30,20,-10,40,80,0,0,28,-20,25])

Siendo los resultados obtenidos por el modelo de red neuronal supervisada entrenada con la base de datos acumulada los siguientes:

TABLA III
Hiperparámetros obtenidos por la red neuronal supervisada

[4.3349e-01 , 1.7708e+00 , 1.9117e+00 , -1.0454e+00 , -6.4818e-01, 4.7675e+00 , 8.9561e+01]
[-3.3954e+00 , 3.9226e-02 , 5.3451e+00 , 1.7323e+01 , 2.6976e+01, -6.5274e+01 , -6.2581e+01]
[-1.1325e+00 , 1.8368e-01 , -3.9822e+00 , 1.6498e+00 , 3.4632e+01 , 7.8781e+01 , 4.6952e+01]
[8.6792e-01 , 6.5504e-01 , 5.6603e+00 , 1.4580e+01 , 4.0459e+01 , 4.1241e+01 , 3.8271e+01]

Con estos valores de parámetros obtenidos se presentan básicamente 2 errores, en la selección de los parámetros en la trayectoria 2 y la trayectoria 3. Esto se puede observar rápidamente en el primer parámetro generado por la red neuronal supervisada para estos movimientos, siendo en ambos casos negativos, lo cual contradice el requerimiento de posicionamiento que se le efectúa al modelo, pues al tener este parámetro un valor negativo, la dirección que originará será en sentido contrario al acercamiento a la posición solicitada. Además, estos parámetros negativos tienen un módulo bastante elevado, pues empíricamente se han observado valores de operación para el módulo de este parámetro del orden de 10-1, es decir valores entre 0.1 y 0.9 aproximadamente.

3.2 Aplicación del modelo de Geometría Inversa Planificada

Para la aplicación del modelo de geometría inversa planificada se implementaron dos manipuladores robóticos desacoplados, llamados robot1 y robot2, en el entorno Simulink-Matlab. El diagrama de bloques desarrollado recibe un vector de entrada de 12 elementos, de forma que contenga un vector de 6 elementos con las posiciones angulares del primer robot y un vector de 6 elementos con las posiciones angulares del segundo robot. En el modelo de bloques, los demultiplexores se encargan de separar los valores de los vectores para controlar el movimiento de las diferentes juntas de los robots.

De la sección anterior (redes neuronales), se pudo apreciar que el modelo de redes neuronales supervisadas no presenta un buen desempeño en predecir los mejores valores de hiperparámetros. Por ello, en esta sección se definen 4 vectores de posicionamiento con los hiperparámetros definidos por el programador. Además, se muestran los resultados obtenidos del modelo de geometría inversa planificada (figura 7) y de la simulación gráfica en Simulink con buenos resultados (figura 8).

Primera trayectoria: Requerimiento a robot 1

- Punto inicial: e_sp_1_n (1,:) = [45;-50;12;80;0;0]
- Obstáculo: e_sp_2_n(1,:) = [45;-42;20]
- Punto final: e_sp_3_n(1,:) = [45;-32;35;80;0;0]

Segunda trayectoria: Requerimiento a robot 1

- Punto inicial: e_sp_3_n(1,:) = [45;-32;35;80;0;0]
- Obstáculo: e_sp_2_n(3,:) = [25;-20;35]
- Punto final: e_sp_3_n(3,:) = [35;-5;35;0;0;0]

Tercera trayectoria: Requerimiento a robot 1

- Punto inicial: e_sp_3_n(3,:) = [35;-5;35;0;0;0]
- Obstáculo: e_sp_2_n(4,:) = [30;-3;33]
- Punto final: e_sp_3_n(4,:) = [40;0;0;0;0;-30]

Cuarta trayectoria: Requerimiento a robot 2

- Punto inicial: e_sp_1_n(4,:) = [40;0;0;0;0;60]
- Obstáculo: e_sp_2_n (5, :) = [40; 25; 0]
- Punto final: e_sp_3_n(5,:) = [40;50;0;0;-30;-30]

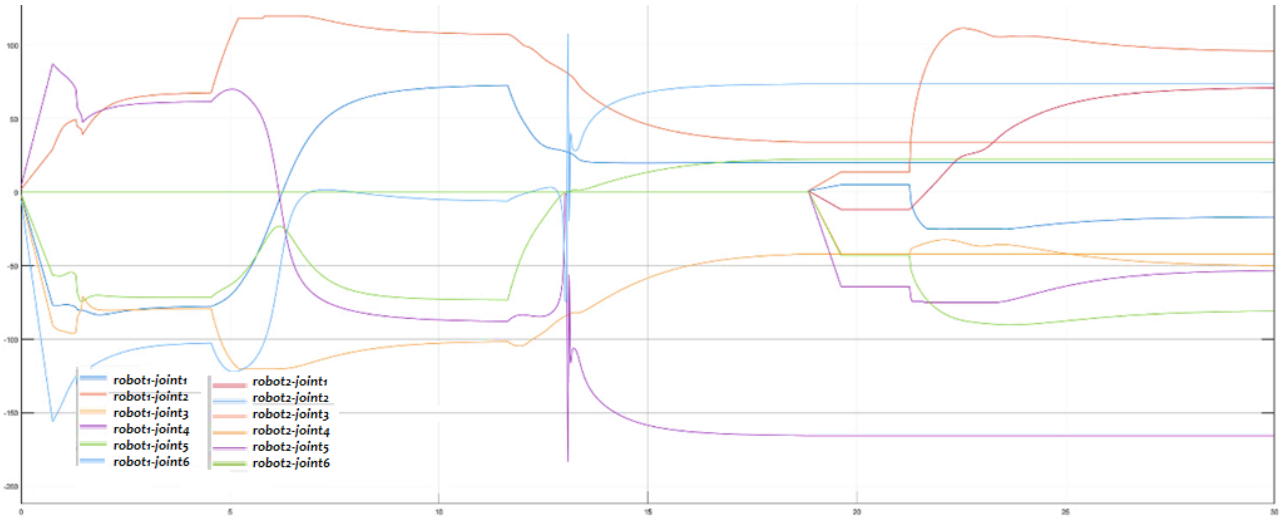

Fig. 7. Serie temporal de las posiciones angulares de ambos robots simulados bajo las trayectorias propuestas.

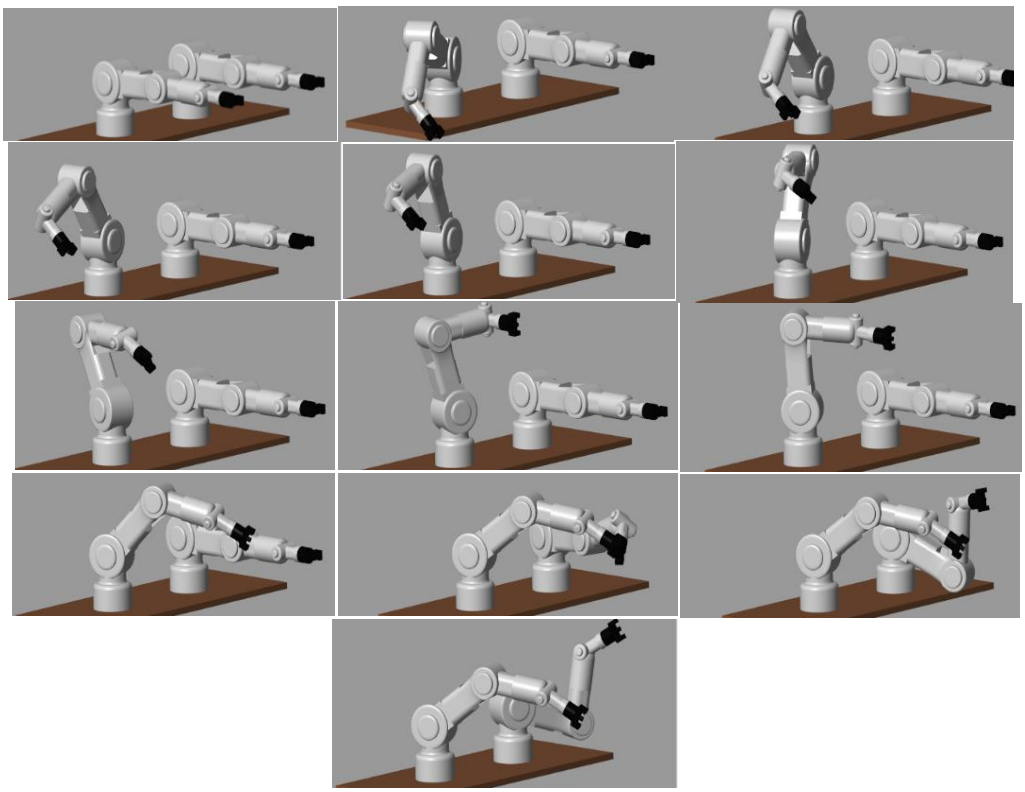

Fig. 8. Secuencia de movimientos de los robots 1 y 2 al simular el modelo de bloques de Simulink bajo las trayectorias propuestas.

Los resultados del script desarrollado en Matlab se muestran en la figura 9, donde la trayectoria morada es ejecutada por el robot 1 y la verde por el robot 2.

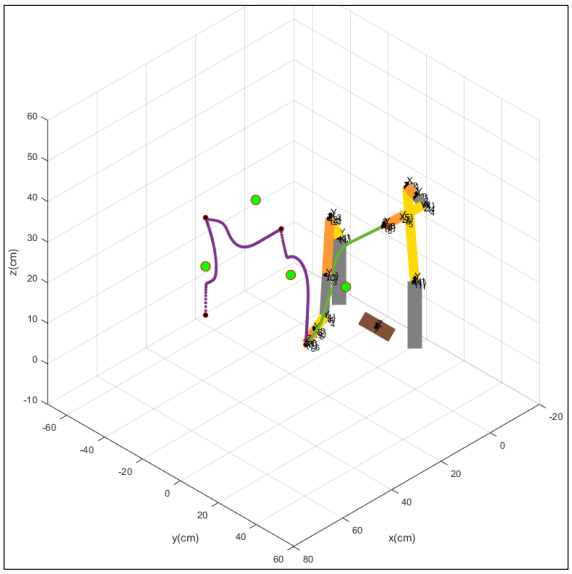

Fig. 9. Trayectoria compuesta generada en el sistema de robots desacoplados.

DISCUSIÓN

De acuerdo con los resultados obtenidos, se puede ver que el manipulador robótico puede realizar la tarea de llegar al punto sin chocar los obstáculos usando el algoritmo propuesto. Sin embrago, el proceso para diseñar el algoritmo, así como la generación de trayectorias para la base de datos es tedioso. Por consiguiente, el método no es favorable cuando se tiene que hacer un prototipado rápido. Por el contrario, el algoritmo propuesto es una solución alternativa a los métodos clásicos y además se puede encontrar algunas mejoras al algoritmo propuesto con el fin de mejorar su rendimiento y hacerlo una opción más favorable a la hora de abordar el problema de planificación de movimiento aplicado en manipuladores robóticos.

CONCLUSIONES

- La inclusión de un modelo de fuerzas vectoriales en un sistema de optimización de geometría inversa para un robot de 6GDL y una correcta selección de los hiperparámetros conducen a un modelo de geometría inversa planificada que permite al robot seguir una trayectoria entre dos posiciones requeridas mientras evita un obstáculo cuya ubicación es informada al robot. Es decir, se concluye que la consideración de los jacobianos de las variaciones de los ángulos de un robot de 6GDL con respecto a las fuerzas repulsivas de un modelo de campos vectoriales en un modelo que considera los jacobianos de las variaciones los ángulos del mismo robot con respecto a la distancia vectorial entre la posición actual y la posición final ha permitido generar un modelo de planificación de trayectorias con una base informada.

- De acuerdo a los resultados empíricos se puede observar que empleando datos de trayectorias óptimas dentro de una región con valores de posiciones similares (tipo de trayectorias homogéneas) para el entrenamiento del modelo de red neuronal empleado se pueden obtener un buen desempeño de predicción, es decir el modelo permite que el robot genere trayectorias óptimas. Esto plantea la existencia de diferentes secciones del espacio que se caracterizan por diferentes predisposiciones en los valores de los modelos. Se concluye que se requiere un análisis más profundo del comportamiento de estos modelos sectorizados, pudiendo plantearse una nueva arquitectura de procesamiento de datos con el uso de clusterización para caracterizar los vectores de posicionamiento en diferentes tipos y realizar diferentes modelos para cada tipo, de forma que se obtenga un mejor desempeño, logrando el objetivo final de generación de trayectorias óptimas para cualquier conjunto de puntos de posicionamiento requerido y obstáculos.

AGRADECIMIENTOS

Los autores agradecemos a la Unidad de Investigación de la Facultad de Ingeniería Mecánica por la financiación del proyecto y al laboratorio de automatización LIIAARC donde se desarrolló el presente trabajo.

REFERENCIAS

[1] S.M. LaValle, Planning algorithms. Cambridge university press, 2006.
[2]Borenstein, J., y Koren, Y, "Real-time obstacle avoidance for fast mobile robots". *IEEE Transactions on systems, Man, and Cybernetics*, 19(5), 1179-1187, 1989.
[3] Liu, Z., Ai, Q., Liu, Y., Zuo, J., Zhang, X., Meng, W., y Xie, S.,"An Optimal Motion Planning Method of 7-DOF Robotic Arm for Upper Limb Movement Assistance". In 2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), pp. 277-282. IEEE., jul. 2019.
[4] A. McLean, I. Mazon, "Incremental roadmaps and global path planning in evolving industrial environments". In Proceedings of IEEE International Conference on Robotics and Automation, 1996, Vol. 1, pp. 101-107.
[5] N. Sleumer, N. Tschichold-Gürmann. "Exact cell decomposition of arrangements used for path planning in robotics". Technical report/ETH Zürich, Department of Computer Science 329, 1999.

[6] T. Mac, C. Copot, D. Tran y R. De Keyser, "Heuristic approaches in robot path planning: A survey", *Robotics and Autonomous Systems*, vol. 86, pp. 13-28, Dec. 2016.
[7] F. Cosío y M. Padilla Castañeda, "Autonomous robot navigation using adaptative potential fields", *Mathematical and computer modelling*, vol. 40, no. 9-10, pp. 1141-1156, Nov. 2004.
[8] H. Shen, H. Wu, B. Chen, Y. Jiang y C. Yan, "Obstacle avoidance algorithm for 7-DOF redundant anthropomorphic arm", *Journal of Control Science and Engineering*, vol. 2015.
[9] E. Conkur, "Path planning using potential fields for highly redundant manipulators", *Robotics and Autonomous Systems*, vol. 52, no. 2-3, pp. 209-228, Aug. 2005.
[10] Rajendran, P., Thakar, S., Bhatt, P. M., Kabir, A. M. y Gupta, S. K, "Strategies for speeding up manipulator path planning to find high quality paths in cluttered environments". *Journal of Computing and Information Science in Engineering*, 1-27, 2020.
[11] Corke, P. I. "A simple and systematic approach to assigning Denavit–Hartenberg parameters". *IEEE transactions on robotics*, 23(3), 590-594, 2007.
[12] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... y Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *The Journal of machine learning research*, 12, pp. 2825-2830.