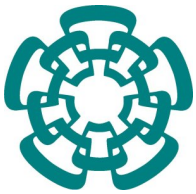


Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional

Unidad Zacatenco
Departamento de Control Automático

Aprendizaje del Comportamiento Humano en el Espacio Articular del Robot Utilizando Redes Neuronales

Tesis que presenta

Jorge Armando Ramírez Díaz

Para obtener el título de

Maestro en Ciencias

En la especialidad de

Control Automático

Director de Tesis

Dr. Wen Yu Liu

Ciudad de México

Agosto, 2019

Dedicado a mi familia que
siempre ha creído en mi y
a Marcela por todo su
amor y apoyo

Agradecimientos

Agradezco inmensamente a mis padres que siempre me ha brindado apoyo incondicional para seguir y cumplir mis sueños.

A mi hermano, quien siempre fue para mi un ejemplo y el mejor amigo.

A Marcela, que con su amor, apoyo y compañía me llena siempre de vida y de energía para seguir adelante

A mi asesor el Dr. Wen, quien me ha contagiado de pasión por la ciencia y que con su guía y ejemplo me ha inspirado a seguir una trayectoria como investigador.

Agradezco a los profesores del departamento de control, que con sus enseñanzas y paciencia me han brindado el conocimiento y aptitudes necesarias para una formación de calidad.

Al CINVESTAV y al CONACYT por todo el apoyo y las facilidades que me han otorgado para continuar mis estudios de posgrado.

Índice general

Índice de figuras

Capítulo 1

Introducción

El uso de robots para asistir y realizar una gran diversidad de tareas se ha visto incrementado en los últimos años, la tecnología existente es capaz de desempeñar tareas con una muy alta precisión y velocidad, superando ampliamente de esta manera a los humanos que intentan desempeñar las mismas tareas. Sin embargo, estos robots son controlados manualmente o programados para seguir una trayectoria especifica la cual es prediseñada. El desarrollo de robots inteligentes, capaces de realizar movimientos autónomos selectivos podría mejorar su desempeño y aumentar en gran medida su margen de aplicaciones.

Figura 1.1: Sistema robótico quirúrgico da Vinci

El uso de robots cirujanos controlados mediante teleoperación se ha vuelto cada vez más común, debido a que este mejora drásticamente las capacidades del cirujano, como lo es el sistema robótico quirúrgico da Vinci (Intuitive Surgical, Sunnyvale, CA) [1]. Por ello en la actualidad el desarrollo de robots inteligentes en el área

médica es muy estudiado, buscando con ello la automatización de robots cirujanos capaces de planear por ellos mismos las trayectorias necesarias para realizar algunas de las tareas más complicadas o más repetitivas en una cirugía, reduciendo así la carga de trabajo de los cirujanos y disminuyendo el tiempo de los procedimientos médicos.

Un aspecto crucial para lograr la automatización en robots cirujanos asistentes se basa en la capacidad del robot de reproducir un comportamiento que le permita desempeñar tareas y habilidades de manera similar a cirujanos expertos.

Los métodos de aprendizaje de comportamiento natural humano se encuentran dados en el espacio de tareas del robot, es decir, un espacio cartesiano con tres dimensiones. En [2], [3], [4] Mayer et al. propone un esquema de aprendizaje por demostración de tareas quirúrgicas en el espacio de trabajo. Otros enfoques más recientes y similares en donde el aprendizaje no se hace por demostración han requerido de la utilización de cinemática inversa, en [5] Murali et al propone un enfoque de aprendizaje por observación. Después de aprender, las trayectorias deseadas deben ser trasformadas al espacio articular del robot mediante cinemática inversa. Sin embargo, para la mayoría de los robots, no se puede obtener soluciones analíticas de la cinemática inversa.

En este trabajo, se aprende el comportamiento humano directamente en el espacio articular del robot siguiendo un enfoque de aprendizaje por demostración. Existen complicaciones en el aprendizaje por demostración en el espacio articular, estas se deben a trayectorias a diferentes velocidades en las demostraciones y los temblores humanos de las demostraciones en las señales articulares que posteriormente se amplifica en el espacio cartesiano. Se utiliza el algoritmo de alineación de trayectorias *Dinamic Time Warping* y redes neuronales para resolver estos problemas, y de esta manera, se logra evitar calcular la cinemática inversa del robot. Los resultados experimentales muestran que este método es efectivo.

Las demostraciones se basan en el comportamiento natural de un humano, ya que los movimientos que este realiza son trayectorias efectivas que el robot puede aprender. El movimiento humano se da por medio de movimientos en las articulas que lo conforman, esto puede ser descrito como una trayectoria en el espacio articular, este comportamiento se traduce como una acción que se encuentra en el espacio de trabajo, como puede ser el dibujar una figura o mover un objeto de un punto a otro. Los métodos clásicos (de cinemática inversa) permiten calcular el movimiento angular necesario de las articulaciones para poder realizar acciones en el espacio de trabajo, pero estos presentan problemas, pues es matemáticamente difícil definir una única trayectoria óptima, además en ciertos casos, dichos métodos no presentan una solución analítica como es en la presencia de redundancia. En el aprendizaje por demostración, al aprender una trayectoria efectiva generada por un humano experto se asegura que la tarea será desempeñada sin impedimentos matemáticos de cinemática.

Van den Berg et al. [6] propuso una técnica iterativa en la cual un robot aprende una trayectoria de referencia y es ejecutada con un aumento significativo en precisión y velocidad. Este trabajo ha sido extendido por Osa et al. en [7] para cambios en el entorno.

Una de las técnicas de aprendizaje puede realizarse mediante redes neuronales, las cuales son un modelo que intenta simular un método de procesamiento básico de información de un cerebro biológico. La red neuronal abstrae las demostraciones que les son proporcionadas generando así un conocimiento que posteriormente podrá ser usado para la generación de trayectorias que reproduzcan un comportamiento natural humano, de la misma manera que se hizo al generar las demostraciones.

1.1. Motivación

En los últimos años con el desarrollo científico y tecnológico avanzando cada vez más en la automatización, se ha logrado sustituir al ser humano por robots en el cumplimiento de un gran numero de tareas, robots que logran realizar tareas de manera mas rápido y precisa. El desarrollo de robots inteligentes, que sean capaces de realizar acciones autónomas en el cumplimiento de una función en las que logre superar al ser humano en habilidades lograría beneficiar muchas áreas donde la robótica es aplicada, un ejemplo de ello es la robótica médica, donde actualmente se intenta desarrollar robots cirujanos capaces de planear por ellos mismos trayectorias necesarias para llevar a cabo tareas complicadas o repetitivas en procedimientos quirúrgicos, dándole de esta manera autonomía al robot.

La aplicación actual de robótica en el aprendizaje por demostración es realizado en el espacio de trabajo del robot, el cual es un espacio cartesiano en tres dimensiones, estas trayectorias son posteriormente trasformadas mediante la cinemática inversa del robot al espacio articular, donde puede ocurrir que dicha cinemática inversa no tenga solución analítica. El presente trabajo presenta un método que aplica el aprendizaje por demostración directamente al espacio articular del robot, teniendo como ventaja el evitar los problemas relacionados al cálculo de la cinemática inversa.

El uso de redes neuronales en la actualidad ha demostrado ser una herramienta del control inteligente para lograr una eficiencia muy alta en el reconocimiento de patrones, clasificación y seguimiento de trayectorias, mediante algoritmos novedosos que agilizan en gran medida el proceso de entrenamiento de redes cada vez más grandes y de arquitecturas más complejas.

1.2. Objetivos

Los objetivos que se plantean en el presente trabajo son:

- Implementar un esquema de aprendizaje basado en demostraciones que imiten un comportamiento natural humano.
- Integrar algoritmos capaces de alinear las demostraciones en distintas velocidades.
- Diseñar y entrenar una arquitectura de red neuronal capaz de mapear del espacio de trabajo al espacio articular de un robot.
- Mejorar en precisión las acciones enseñadas por demostración mediante filtrado.
- Diseñar un algoritmo capaz de cambiar la velocidad de las demostraciones.
- Generar trayectorias eficientes en el espacio articular para cualquier trayectoria deseada en el espacio de trabajo.
- Aplicar experimentalmente el método propuesto y evaluar su desempeño en el espacio de trabajo.

1.3. Estructura de la tesis

En el capítulo 1 se presenta una introducción a los temas desarrollados a lo largo de este trabajo, se plantean las motivaciones y los objetivos buscados.

El capítulo 2 expone los puntos más importantes a ser tomados en cuenta para el diseño e implementación de la metodología de aprendizaje por demostración del comportamiento humano. Desde el robot aprendiz humanoide utilizado para el desarrollo y experimentación, su estructura, magnitudes y los componentes principales que lo conforman, además, se analiza el modelo matemático que describe su cinemática directa. También se analizan aspectos como, quien controla y quien ejecuta la demostración, la coincidencia entre los estados-acciones experimentados, almacenados y ejecutados y las técnicas con las que comúnmente se realizan y se aprenden dichas demostraciones.

En el capítulo 3 se desarrollan los conceptos, ecuaciones y algoritmos matemáticos para el procesamiento de las señales digitales provenientes de las demostraciones. Los algoritmos presentados sirven para generar un alineamiento entre las demostraciones, así como para cambiar la velocidad de estas. Se presenta también los fundamentos necesarios para el análisis de frecuencia y filtrado de las señales articulares.

En el capítulo 4 se presentan los fundamentos de las redes neuronales artificiales y con base en ello se propone una arquitectura de red neuronal de tipo *feed forward* para el aprendizaje de las demostraciones, la cual es capaz de mapear correctamente entre el espacio de trabajo y el espacio articular del robot, logrando así hacer un proceso de cinemática inversa.

En el capítulo 5 se desarrollan los casos de estudio experimental donde se aplica el método del aprendizaje del comportamiento humano y con ello se presentan los resultados obtenidos.

El capítulo 6 comprende las conclusiones y el posible trabajo futuro.

Capítulo 2

Aprendizaje del comportamiento humano

El aprendizaje del comportamiento humano se basa en el aprendizaje por demostración (ApD), la cual es una técnica que permite el mapeo entre estados y acciones de un robot, siendo capaz el robot de generar una acción basada en demostraciones, donde los estados para generar esta u otras acciones son almacenados y aprendidos mediante distintas técnicas que permiten generar las acciones originales o similares.

Las demostraciones se realizan mediante un sujeto denominado maestro, las cuales aprenderá el robot que se le denomina aprendiz con la finalidad que reproduzca una tarea con un comportamiento humano natural.

El aprendizaje del comportamiento humano se divide en dos fases: (i) enseñanza de comportamiento y (ii) generación de comportamiento, la primera se refiere a como se mapean las demostraciones del comportamiento humano hacia el robot (direccionamiento de los estados y su relación con las acciones) y la segunda se refiere a la reproducción de la trayectoria deseada de comportamiento mediante el controlador.

En [8] se presenta una explicación más amplia sobre el ApD.

2.1. Robot aprendiz

El robot utilizado para el desarrollo y experimentación del presente trabajo es el manipulador Cyton Gamma 1500 construido y diseñado por Robai corporation, el cuál es un brazo robótico humanoide redundante, de igual manera a un brazo humano, permitiendo así posicionar y orientar el robot en un gran numero de maneras. Cuenta con siete grados de libertad, siendo movida y controlada cada articulación y el *gripper* por servomotores, se muestran las características de las ar-

ticulaciones en el Cuadro 2.1. En la Figura 2.1 se muestra el robot manipulador Cyton Gamma 1500 con sus dimensiones y articulaciones [9].

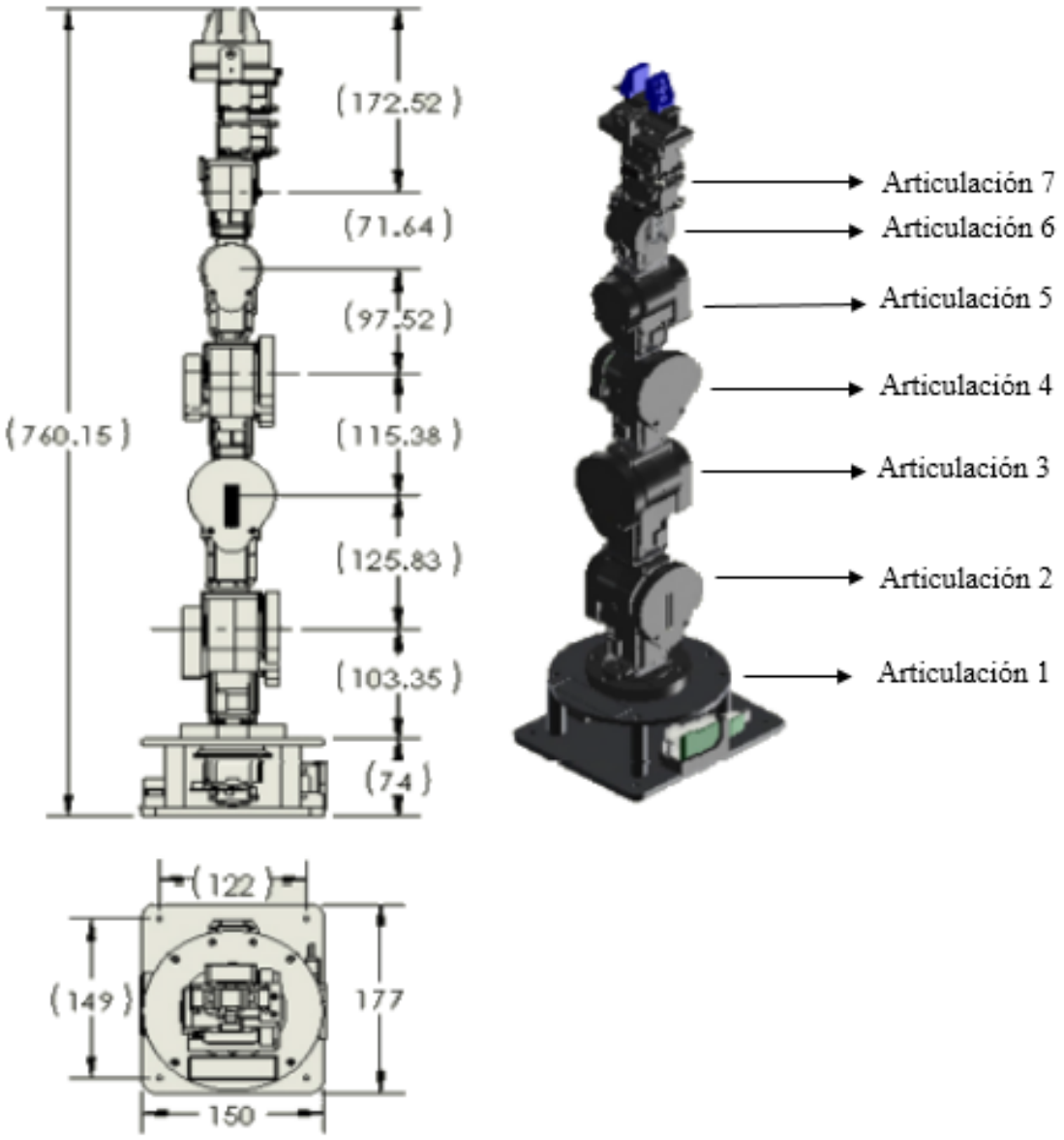


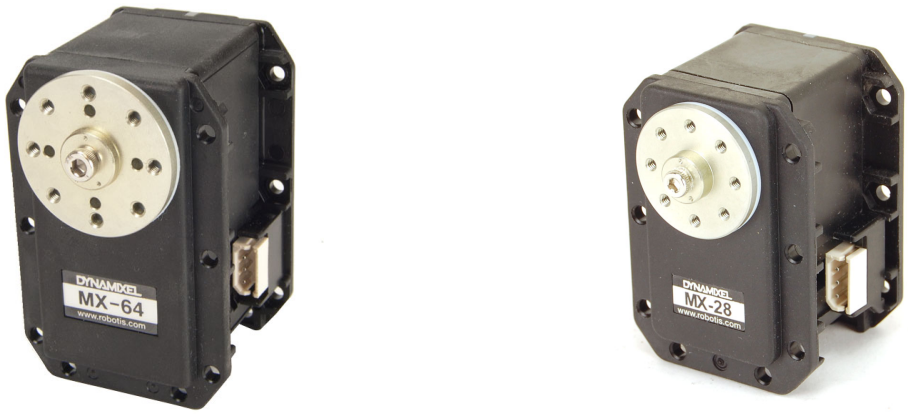
Figura 2.1: Dimensiones del robot manipulador Cyton Gamma 1500

2.1.1. Hardware

El robot manipulador Cyton Gamma 1500 cuenta con dos modelos diferentes de servomotor Dynamixel, MX-64 y MX-28 mostrados en la figura 2.2.

Articulación	Tipo de articulación	Rango de movimiento (grados)	Modelo de servomotor
Shoulder Roll	Spin	300	MX-64
Shoulder Pitch	Articulate	210	MX-64
Shoulder Yaw	Articulate	210	MX-28
Elbow Pitch	Articulate	210	MX-28
Wrist Yaw	Articulate	210	MX-28
Wrist Pitch	Articulate	210	MX-28
Wrist Roll	Articulate	210	MX-28

Cuadro 2.1: Características de las articulaciones del robot manipulador Cyton Gamma 1500



(a) MX-64 (b) MX-28

Figura 2.2: Servomotores utilizados en el manipulador Cyton Gamma 1500

Cada servomotor cuenta con un sensor *encoder absolute contactless* magnético de 12bit (4096 pulsos) que puede operar en 360 grados, teniendo una resolución de 0.0878 grados por pulso [10][11]. Se muestran las especificaciones y el desempeño de los servomotores en las figuras 2.3 y 2.4.

Data

	Unit	Data
Weight	g	135
Dimension	mm	40.2 x 61.1 x 41
Gear Ratio	type/material	200:1 (Spur/Metal)
Network	-	TTL / RS-485
Position Sensor (resolution)	-	Contactless Absolute Encoder (360˚/ 4096)
Motor	-	Maxon Motor
Operation Voltage	V	10~14.8
Stall Torque	N.m	6.0 at 12V
Stall Current	A	4.1 at 12V
No Load Speed	RPM	63 at 12V

Performance Graph

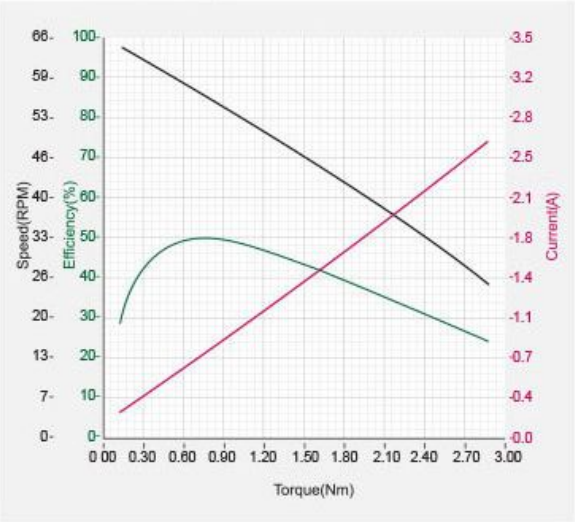


Figura 2.3: Especificaciones del servomotor MX-64

Data

	Unit	Data
Weight	g	77
Dimension	mm	35.6 x 50.6 x 35.5
Gear Ratio	type/material	193:1 (Spur/Metal)
Network	-	TTL / RS-485
Position Sensor (resolution)	-	Contactless Absolute Encoder (360˚/ 4096)
Motor	-	Maxon Motor
Operation Voltage	V	10~14.8
Stall Torque	N.m	2.5 at 12V
Stall Current	A	1.4 at 12V
No Load Speed	RPM	55 at 12V

Performance Graph

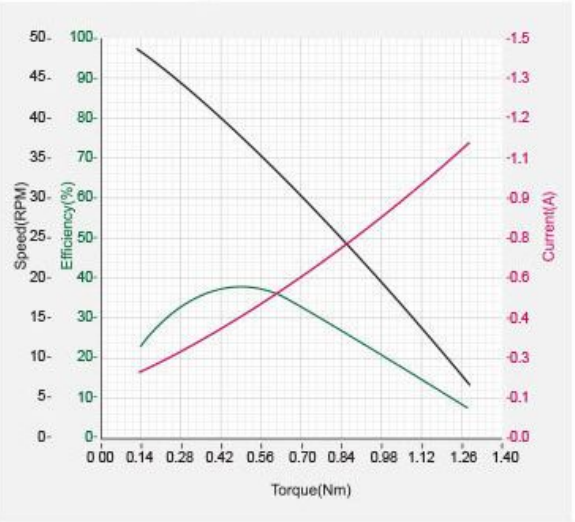


Figura 2.4: Especificaciones del servomotor MX-28

La serie de servomotores MX usa un controlador de tipo PID como principal método de control, mostrado en la figura 2.5, donde las ganancias proporcional,

integral y derivativa están dadas por las siguientes relaciones

$$K_p = \frac{P\ Gain}{8} \quad K_i = \frac{(I\ Gain)(1000)}{2048} \quad K_d = \frac{(D\ Gain)(4)}{1000} \tag{2.1}$$

donde $P\ Gain$,$I\ Gain$ y $D\ Gain$ pueden variar entre 0 y 254.

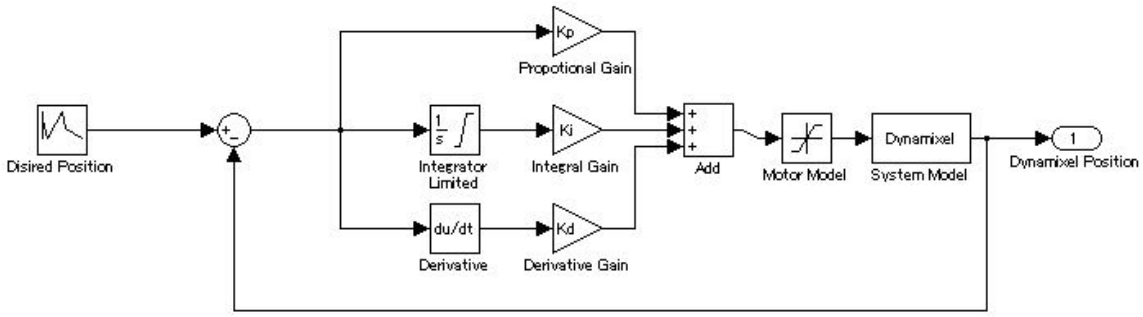


Figura 2.5: Controlador PID del servomotor de la serie MX

Los servomotores se encuentran conectados en serie mediante un adaptador llamado USB2Dynamixel [12] mostrado en la figura 2.6, este se encuentra configurado para comunicación TTL, y así lograr la conexión con una computadora.

Figura 2.6: Adaptador USB2Dynamixel para comunicación con computadora

La compañía ROBOTIS proporciona la biblioteca Dynamixel SDK-master [13], la cual contiene una serie de librerías para el desarrollo de código de programación de control de los servomotores Dynamixel en distintas plataformas y lenguajes de programación, esta biblioteca en conjunto con la librería Dynamixel Simulink [14] permitieron el desarrollo de la interfaz de control del robot en Matlab-Simulink para la fase de aprendizaje del comportamiento humano (figura 5.2) y para la fase de generación de comportamiento en el robot (figura 5.3).

2.1.2. Cinemática del robot

Para el análisis de cinemática directa se sigue el procedimiento basado en la convención de Denavit-Hartenber para cualquier manipulador [15], el cual es:

- Localizar y etiquetar los ejes de articulación $z_0, ..., z_{n-1}$.
- Establecer el marco de referencia base. Establecer el origen en cualquier punto sobre el eje z_0. Los ejes x_0 y y_0 se elijen convenientemente, de manera que formen un marco de referencia de acuerdo con la regla de la mano derecha. Para $i = 1, ..., n-1$ se siguen los pasos del 3 al 5.
- Localizar el origen o_i donde la normal común a z_i y z_{i-1} intersectan a z_i. Si z_i intersecta a z_{i-1} o_i se debe localizar en esta intersección. Si z_i y z_{i-1} son paralelos, localizar o_i en cualquier posición conveniente a lo largo de z_i.
- Establecer x_i a lo largo de la normal común entre z_{i-1} y z_i a través de o_i, o en la dirección normal al plano z_{i-1}-z_i si z_{i-1} y z_i se intersectan.
- Establecer y_i para completar el referencial de acuerdo con la regla de la mano derecha.
- Establecer el marco de referencia del efector final $o_n x_n y_n z_n$. Asumiendo que la n-esima articulación es rotacional, se establece $z_n = a$ en la dirección de z_{n-1}. Establecer el origen o_n convenientemente a lo largo de z_n preferentemente en el centro del *griper* o en la punta de la herramienta que pueda estar portando el manipulador. Establecer $y_n = s$ en la dirección del cierre del *griper* y establecer $x_n = n$ como $s \times a$. Si la herramienta no es un *griper* simple se establece x_n y y_n convenientemente para formar un referencial de acuerdo con la regla de la mano derecha.
- Crear la tabla de parámetros de las articulaciones
  a_i = distancia a lo largo de x_i desde o_i a la intersección de x_i con z_{i-1}.
  d_i = distancia a lo largo de z_{i-1} desde o_{i-1} a la intersección de x_i con z_{i-1}. d_i es variable si la articulación es prismática.
  α_i = ángulo entre z_{i-1} y z_i medido sobre x_i.
  θ_i = ángulo entre x_{i-1} y x_i medido sobre z_{i-1}. θ_i es variable si la articulación es rotacional.
- Formar las matrices de trasformación homogénea A_i a partir de los parámetros de las articulaciones del paso anterior sustituyéndolos en

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

- Formar $T_n^0 = A_1 \cdots A_n$. Esto proporciona la posición y orientación del marco

referencial de la herramienta expresado en coordenadas del marco referencial base.

Se muestra en la figura 2.7 los marcos referenciales posicionados según la convención de Denavit-Hartenberg y en el cuadro 2.2 los parámetros de las articulaciones.

Articulación	a_i	d_i	θ_i	α_i
1	0	d_1	θ_1^*	$-\frac{\pi}{2}$
2	a_2	0	$\theta_2^* - \frac{\pi}{2}$	$-\frac{\pi}{2}$
3	a_3	0	θ_3^*	$\frac{\pi}{2}$
4	a_4	0	θ_4^*	$-\frac{\pi}{2}$
5	a_5	0	θ_5^*	$\frac{\pi}{2}$
6	0	0	$\theta_6^* + \frac{\pi}{2}$	$\frac{\pi}{2}$
7	0	d_7	θ_7^*	0

Cuadro 2.2: Parámetros de las articulaciones según la convención de Denavit-Hartenberg

A partir del cuadro 2.2 y de las dimensiones del robot en la figura 2.1 se consideran las siguientes variables

$$\begin{array}{ll} q_1 = \theta_1^* & d_1 = 177.35 \\ q_2 = \theta_2^* - \frac{\pi}{2} & a_2 = 125.83 \\ q_3 = \theta_3^* & a_3 = 115.38 \\ q_4 = \theta_4^* & a_4 = 97.52 \\ q_5 = \theta_5^* & a_5 = 71.64 \\ q_6 = \theta_6^* + \frac{\pi}{2} & \\ q_7 = \theta_7^* & d_7 = 172.52 \end{array} \tag{2.3}$$

Entonces las matrices de transformación homogénea A_i son

$$A_1 = \begin{bmatrix} \cos(q_1) & 0 & -\sin(q_1) & 0 \\ \sin(q_1) & 0 & \cos(q_1) & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

$$A_2 = \begin{bmatrix} \cos(q_2) & 0 & -\sin(q_2) & a_2\cos(q_2) \\ \sin(q_2) & 0 & \cos(q_2) & a_2\sin(q_2) \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

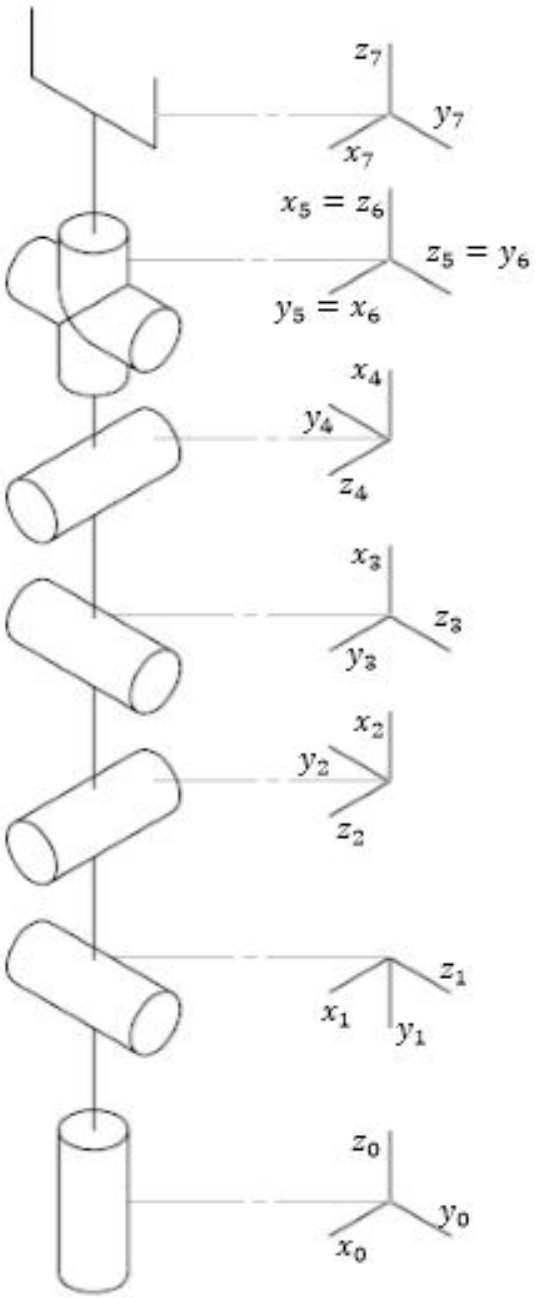


Figura 2.7: Marcos de referencia del manipulador Cyton Gamma 1500

$$A_3 = \begin{bmatrix} \cos(q_3) & 0 & \sin(q_3) & a_3\cos(q_3) \\ \sin(q_3) & 0 & -\cos(q_3) & a_3\sin(q_3) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

$$A_4 = \begin{bmatrix} \cos(q_4) & 0 & -\sin(q_4) & a_4\cos(q_4) \\ \sin(q_4) & 0 & \cos(q_4) & a_4\sin(q_4) \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

$$A_5 = \begin{bmatrix} \cos(q_5) & 0 & \sin(q_5) & a_5\cos(q_5) \\ \sin(q_5) & 0 & -\cos(q_5) & a_5\sin(q_5) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

$$A_6 = \begin{bmatrix} \cos(q_6) & 0 & \sin(q_6) & 0 \\ \sin(q_6) & 0 & -\cos(q_6) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

$$A_7 = \begin{bmatrix} \cos(q_7) & -\sin(q_7) & 0 & 0 \\ \sin(q_7) & \cos(q_7) & 0 & 0 \\ 0 & 0 & 1 & d_7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

T_7^0 esta entonces dado por

$$T_7^0 = A_1A_2A_3A_4A_5A_6A_7 \tag{2.11}$$

$$T_7^0 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

donde

$$\begin{aligned} r_{11} =& s_7(s_5(c_4(s_1s_3 + c_1c_2c_3) - c_1s_2s_4) - c_5(c_3s_1 - c_1c_2s_3)) - \\ & c_7(s_6(s_4(s_1s_3 + c_1c_2c_3) + c_1c_4s_2) - c_6(c_5(c_4(s_1s_3 + c_1c_2c_3) - \\ & c_1s_2s_4) + s_5(c_3s_1 - c_1c_2s_3))) \end{aligned} \tag{2.13}$$

$$\begin{aligned} r_{12} =& s_7(s_6(s_4(s_1s_3 + c_1c_2c_3) + c_1c_4s_2) - c_6(c_5(c_4(s_1s_3 + \\ & c_1c_2c_3) - c_1s_2s_4) + s_5(c_3s_1 - c_1c_2s_3))) + c_7(s_5(c_4(s_1s_3 + \\ & c_1c_2c_3) - c_1s_2s_4) - c_5(c_3s_1 - c_1c_2s_3)) \end{aligned} \tag{2.14}$$

$$r_{13} = c_6(s_4(s_1s_3 + c_1c_2c_3) + c_1c_4s_2) + s_6(c_5(c_4(s_1s_3 + c_1c_2c_3) - c_1s_2s_4) + s_5(c_3s_1 - c_1c_2s_3)) \tag{2.15}$$

$$r_{21} = c_7(s_6(s_4(c_1s_3 - c_2c_3s_1) - c_4s_1s_2) - c_6(c_5(c_4(c_1s_3 - c_2c_3s_1) + s_1s_2s_4) + s_5(c_1c_3 + c_2s_1s_3))) - s_7(s_5(c_4(c_1s_3 - c_2c_3s_1) + s_1s_2s_4) - c_5(c_1c_3 + c_2s_1s_3)) \tag{2.16}$$

$$r_{22} = -s_7(s_6(s_4(c_1s_3 - c_2c_3s_1) - c_4s_1s_2) - c_6(c_5(c_4(c_1s_3 - c_2c_3s_1) + s_1s_2s_4) + s_5(c_1c_3 + c_2s_1s_3))) - c_7(s_5(c_4(c_1s_3 - c_2c_3s_1) + s_1s_2s_4) - c_5(c_1c_3 + c_2s_1s_3)) \tag{2.17}$$

$$r_{23} = -c_6(s_4(c_1s_3 - c_2c_3s_1) - c_4s_1s_2) - s_6(c_5(c_4(c_1s_3 - c_2c_3s_1) + s_1s_2s_4) + s_5(c_1c_3 + c_2s_1s_3)) \tag{2.18}$$

$$r_{31} = s_7(s_5(c_2s_4 + c_3c_4s_2) + c_5s_2s_3) - c_7(c_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) + s_6(c_2c_4 - c_3s_2s_4)) \tag{2.19}$$

$$r_{32} = s_7(c_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) + s_6(c_2c_4 - c_3s_2s_4)) - c_7(s_5(c_2s_4 + c_3c_4s_2) + c_5s_2s_3) \tag{2.20}$$

$$r_{33} = c_6(c_2c_4 - c_3s_2s_4) - s_6(c_5(c_2s_4 + c_3c_4s_2) - s_2s_3s_5) \tag{2.21}$$

$$d_x = d_7(c_6(s_4(s_1s_3 + c_1c_2c_3) + c_1c_4s_2) + s_6(c_5(c_4(s_1s_3 + c_1c_2c_3) - c_1s_2s_4) + s_5(c_3s_1 - c_1c_2s_3))) + a_2c_1c_2 + a_5c_5(c_4(s_1s_3 + c_1c_2c_3) - c_1s_2s_4) + a_3s_1s_3 + a_4c_4(s_1s_3 + c_1c_2c_3) + a_5s_5(c_3s_1 - c_1c_2s_3) + a_3c_1c_2c_3 - a_4c_1s_2s_4 \tag{2.22}$$

$$d_y = a_2c_2s_1 - d_7(c_6(s_4(c_1s_3 - c_2c_3s_1) - c_4s_1s_2) + s_6(c_5(c_4(c_1s_3 - c_2c_3s_1) + s_1s_2s_4) + s_5(c_1c_3 + c_2s_1s_3))) - a_3c_1s_3 - a_5c_5(c_4(c_1s_3 - c_2c_3s_1) + s_1s_2s_4) - a_4c_4(c_1s_3 - c_2c_3s_1) - a_5s_5(c_1c_3 + c_2s_1s_3) + a_3c_2c_3s_1 - a_4s_1s_2s_4 \tag{2.23}$$

$$\begin{aligned} d_z =& d_1 - a_2 s_2 - d_7(s_6(c_5(c_2 s_4 + c_3 c_4 s_2) - s_2 s_3 s_5) - \\ & c_6(c_2 c_4 - c_3 s_2 s_4)) - a_3 c_3 s_2 - a_4 c_2 s_4 - a_5 c_5(c_2 s_4 + \\ & c_3 c_4 s_2) - a_4 c_3 c_4 s_2 + a_5 s_2 s_3 s_5 \end{aligned} \tag{2.24}$$

Se observa que para resolver el problema general de cinemática inversa se requiere resolver las ecuaciones 2.13, 2.14, 2.15, 2.16, 2.17, 2.18, 2.19, 2.20, 2.21, 2.22, 2.23 y 2.24, las cuales forman un sistema de 12 ecuaciones trigonométricas simultaneas no lineales con 7 variables. El sistema de ecuaciones, por su forma no lineal, numero de ecuaciones y variables, se vuelve un problema difícil de resolver, además, se sabe que la redundancia del robot le permite tener una posición y orientación determinada en el efector final con diferentes configuraciones articulares, por lo que la solución no es única.

2.2. Diseño del aprendizaje del comportamiento humano por demostración

2.2.1. Demostrador

En la mayoría de aplicaciones con ApD se toma la decisión de usar un ser humano como maestro para realizar las demostraciones, aunque también puede ser utilizado como maestro un robot o una simulación de planeación.

Ya que el ApD es usado como técnica para el aprendizaje del comportamiento natural de un humano es requisito indispensable que un humano sea quien controla la demostración o quien ejecuta la demostración o ambos.

En la elección del maestro es necesario considerar (i) quien controlara la demostración y (ii) quien ejecutara la demostración. Por ejemplo, considérese un robot que aprende a mover una caja de un punto a otro. Un enfoque de ApD tendría un robot maestro que mueve la caja usando su propio cuerpo para hacerlo, aquí el robot maestro es quien controla la demostración y el mismo cuerpo del robot maestro es quien ejecuta la demostración. Otro enfoque tendría un ser humano que teleopera al robot aprendiz para que mueva la caja, en este caso el ser humano controla la demostración y el cuerpo del aprendiz es quien ejecuta la demostración.

La similitud entre los estados y los espacios de trabajo y acción del maestro y del aprendiz determina los algoritmos que puede requerir para procesar la información.

2.2.2. Técnicas de demostración

La estrategia de como se provee al aprendiz de datos se divide en dos, (i) aprendizaje de tipo *Batch*, donde el aprendizaje se realiza una vez que se han obtenido

todos los datos y (ii) aprendizaje iterativo, donde el aprendizaje se realiza a medida que los datos se encuentran disponibles.

2.3. Obtención de datos

El como se realiza las demostraciones de comportamiento basadas en la relación acción-estado es muy importante para la obtención de datos, pues este depende de la complejidad del robot y de la tarea de comportamiento humano que se pretende enseñar.

A medida que aumenta el número de demostraciones se vuelve más complicado para el aprendiz reproducir el comportamiento original del maestro, pues aumenta la generalidad de las demostraciones.

2.3.1. Correspondencia

La correspondencia es un factor muy importante a tomar en cuenta, pues el aprendiz y el maestro pueden ser morfológicamente diferentes, teniendo el maestro mayor o menor numero de grados de libertad, márgenes de movimiento distinto, eslabones entre articulaciones de diferente tamaño, por mencionar algunos casos, por ello es necesario tomar en cuenta solo algunos estados o hacer un mapeo para que estos coincidan.

La correspondencia puede ser categorizada en:

- Mapeo de almacenamiento: se refiere a la coincidencia entre los estados y acciones experimentados por el maestro y los almacenados.
- Mapeo de personificación: se refiere a la coincidencia entre los estados y acciones almacenadas y los ejecutados por el aprendiz.

Se puede categorizar la obtención de datos en dos categorías basado en la plataforma de ejecución de las demostraciones de comportamiento y en su correspondencia en (i) Demostración y (ii) Imitación.

2.3.2. Demostración

Se considera como demostración cuando el maestro utiliza el cuerpo del aprendiz para ejecutar la tarea. No existe aquí problemas de correspondía por mapeo de personificación, pues al usarse al aprendiz este almacena los estados y acciones experimentados por sus propios sensores. Sin embargo, se puede categorizar en dos técnicas según su mapeo de personificación como teleoperación y sombreo.

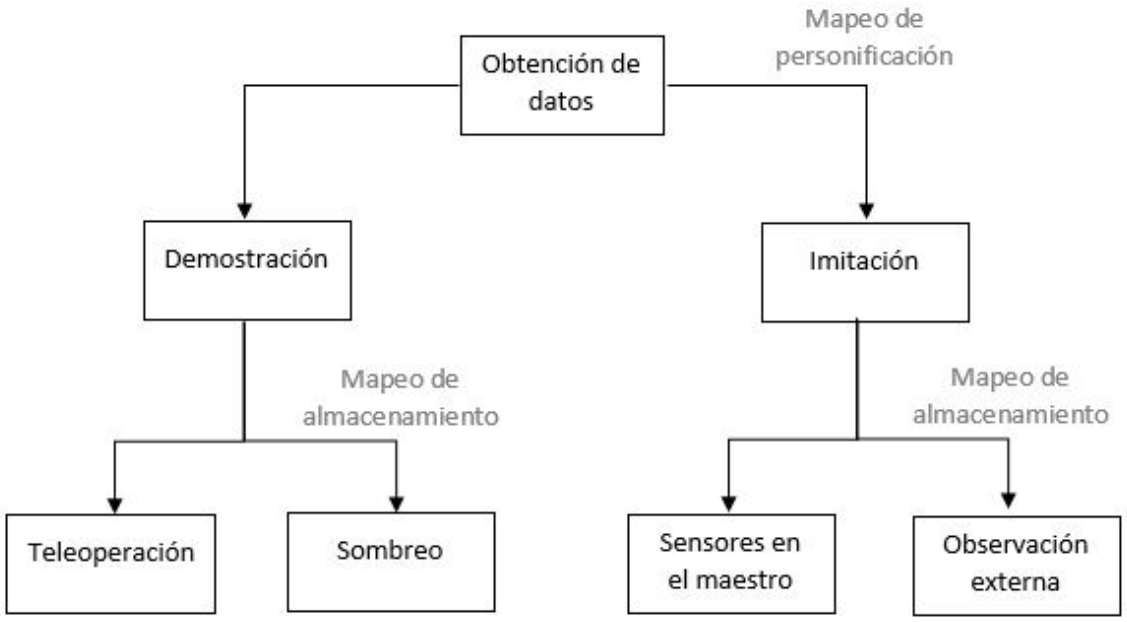


Figura 2.8: Categorización de la obtención de datos según la plataforma de ejecución y su correspondencia.

Operación

Es la técnica de demostración donde el maestro opera directamente al aprendiz quien mediante sus propios sensores almacena los estados y acciones experimentados por el maestro. Al usar el aprendiz sus propios sensores no es necesario un mapeo de personificación ni de almacenamiento, pues los estados y acciones experimentados por el maestro coinciden con los almacenados y de igual manera con los ejecutados por el aprendiz. Por esta razón la operación provee el método más directo de trasferencia en el ApD.

Sombreo

Es la técnica de demostración donde el aprendiz intenta seguir el movimiento de la demostración ejecutada por el maestro, el aprendiz utiliza sus propios sensores para almacenar los estados y acciones. Al intentar seguir los movimientos del maestro implica que los estados y acciones experimentados por el maestro no serán los mismos almacenados, por lo que un mapeo de almacenamiento será necesario.

2.3.3. Imitación

Se considera como imitación cuando los sensores se encuentran montados fuera de la plataforma del aprendiz. Entonces los estados y acciones almacenados serán diferentes a los ejecutados por el aprendiz, por lo que un mapeo de personifica-

ción es necesario. El ApD basado en imitación se puede clasificar en sensores en el maestro y observación externa.

Sensores en el maestro

Es una técnica de imitación en donde los sensores se encuentran montados directamente en la plataforma de ejecución del maestro, de esta manera, los estados y acciones experimentados por el maestro son los mismos que se almacenan. No es necesario un mapeo de almacenamiento. Se suele usar al ser humano como maestro montándole sensores para medir los estados y acciones que experimenta.

Observación externa

Esta técnica de imitación se da cuando los sensores se encuentran fuera de la plataforma de operación del maestro y del aprendiz, por lo que un mapeo de almacenamiento es necesario ya que los estados y acciones experimentados por el maestro serán diferentes a los almacenados. Esta técnica provee el método menos directo y por ello es menos recomendado, pues la exactitud es menor. Se destaca aquí el uso de cámaras para sensar el movimiento de un ser humano quien funciona como maestro.

2.3.4. Excepciones

Existen métodos los cuales pueden ser variaciones de ApD, pero no pueden ser categorizados dentro de las anteriores secciones, pues en la mayoría de los casos solo lo estados son almacenados y no las acciones. Por ejemplo el dibujado de una trayectoria en un plano, donde solo son almacenados los estados y mediante cinemática directa es generada la acción.

2.4. Técnicas de aprendizaje y generación de comportamiento

Una vez se ha almacenado los estados y acciones por alguno de los métodos anteriormente mencionados es necesario enseñar los estados y acciones del comportamiento humano al aprendiz. Las técnicas de aprendizaje se pueden categorizar en mapeo de funciones, modelo de sistema y planeación.

2.4.1. Mapeo de función

El mapeo de funciones intenta aproximar mediante una función, de los estados a la acción, representado por la ecuación

$$f() : E \longrightarrow A \tag{2.25}$$

donde E son los estados almacenados y A la acción almacenada. El objetivo de esta técnica es reproducir el comportamiento fundamental del maestro y generalizar sobre los estados y acciones almacenados y poder encontrar una solución valida para un conjunto de estados similares a los de demostración. En [16] se desarrolla un método de ApD usando el modelo oculto de marcov para un exoesqueleto de 4 grados de libertad. El mapeo de función está dividido en dos enfoques, clasificación y regresión. Ambos enfoques suelen emplear algunas de las herramientas listadas a continuación:

- Modelo de mezclas gausianas
- Arboles de decisiones
- Redes bayesianas
- Clasificadores de vecinos k-cercanos
- Modelo oculto de marcov
- Redes Neuronales
- Maquinas de soporte vectorial

Clasificación

En el enfoque de clasificación se tiene como entrada los estados los cuales son categorizados en el mapeo para producir una salida discreta como acción del robot.

Regresión

En este enfoque los estados almacenados son mapeados para producir una salida continua como acción del robot.

2.4.2. Modelo de sistema

La técnica de modelo de sistema hace uso de un modelo de transición de estado $T(e'|e,a)$ que es determinado por medio de los estados y acciones almacenados, con ayuda de una función de recompensa $R(e)$ que puede ser aprendida por demostraciones o bien, diseñada por el usuario, el valor de la función de recompensa r se asocia al estado e que reproducirá el aprendiz. Usualmente esta técnica es implementada mediante el aprendizaje reforzado.

Diseño de función de recompensa

Esta función de recompensa es definida por el usuario de tal forma que es cero en la mayoría de los casos excepto por obstáculos y cerca del punto objetivo.

Función de recompensa aprendida

Debido a que puede ser complicado el diseño de una función de recompensa se opta por generarla a partir del ApD. Esto puede ser implementado al asociar una recompensa muy alta a los estados encontrados durante la demostración.

2.4.3. Planeación

La planeación como técnica de ApD se basa en generar el comportamiento deseado del aprendiz como un plan siendo así una secuencia de acción que van desde el estado inicial hasta el estado final. Es necesario para esta técnica que los estados se definan antes que la acción sea ejecutada como condiciones previas y los estados resultantes de la ejecución de la acción como condiciones posteriores. La relación entre condiciones previas y posteriores y su similitud, pueden ser controlados por diferentes algoritmos y dependerán de igual manera de la información adicional que proporciona el maestro.

Capítulo 3

Procesamiento de demostraciones en el espacio articular

Los estados del robot almacenados provenientes del ApD del comportamiento humano se encuentran dados por una señal en tiempo discreto, esto se da mediante un proceso de muestreo que remplaza el tiempo continuo con una serie de valores en tiempo discreto dado que las mediciones tomadas por los sensores *enconder* de los servomotores se dan de manera intermitente. Una señal en tiempo continuo se define sobre un intervalo continuo de tiempo y son representadas por una variable independiente continua. Una señal en tiempo continuo también es llamada señal analógica. Una señal en tiempo discreto esta definida solo en valores discretos de tiempo y por tanto la variable independiente que la representa tiene valores discretos, son representadas por una secuencia de números, también llamada serie de tiempo [17].

Las demostraciones del comportamiento humano son conjuntos de ϕ número de señales articulares en tiempo discreto representadas por una secuencia de N números que se denotan como $q_x^y(n)$ donde $n = 1, 2, 3, ..., N$, x y y son el número de la articulación correspondiente del robot y de la demostración, respectivamente. El conjunto de ϕ señales articulares correspondiente a la demostración y se puede expresar como

$$q^y = \begin{bmatrix} q_1^y(n) \\ q_2^y(n) \\ \vdots \\ q_\phi^y(n) \end{bmatrix} \tag{3.1}$$

En el caso donde la señal articular de tiempo discreto es extraída de una señal analógica mediante un muestreo periódico en el cual los instantes de muestreo

están espaciados de manera uniforme cada T_m segundos, T_m se denomina el periodo de muestreo, de manera tal que

$$q_x^y(n) = \mathbf{q}_x^y(nT_m) \tag{3.2}$$

donde $\mathbf{q}_x^y$ es la señal articular analógica correspondiente a la x articulación y a la y demostración. Se define la frecuencia de muestreo como

$$w_m = \frac{1}{T_m} \tag{3.3}$$

3.1. Alineación de demostraciones

Se conoce como *Dynamic Time Warping* (DTW) al algoritmo para analizar y medir similitudes en series de tiempo que pueden ser diferentes en velocidad, de la misma forma se puede encontrar un alineamiento óptimo entre estas series de tiempo al ser deformadas.

Para alinear dos diferentes demostraciones con N y M número de datos, se empieza desde el alineamiento individual de cada articulación x

$$q_x^1(n) = \{q_x^1(1), q_x^1(2), q_x^1(3), ..., q_x^1(i), ..., q_x^1(N)\} \tag{3.4}$$

$$q_x^2(m) = \{q_x^2(1), q_x^2(2), q_x^2(3), ..., q_x^2(j), ..., q_x^2(M)\} \tag{3.5}$$

Para alinear estas series de tiempo mediante DTW se forma una matriz donde cada elemento esta defino por la distancia euclidiana

$$\rho_{ij}^x = |q_x^1(i) - q_x^2(j)| \tag{3.6}$$

Posteriormente se define el camino de deformación denotado por

$$W^x = w_1^x, ..., w_{k-1}^x, w_k^x, ..., w_{l_x}^x \tag{3.7}$$

donde $w_1^x = (1,1)$, $w_{k-1}^x = (a_{k-1}^x, b_{k-1}^x)$, $w_k^x = (a_k^x, b_k^x)$ y $l_x \leq N + M - 1$, a_k^x y b_k^x se encuentran dados por la expresión

$$\begin{aligned} r^x(a_k^x, b_k^x) = \min\{&\rho^x_{(1+a^x_{(k-1)}),(1+b^x_{(k-1)})}, \\ &\rho^x_{(a^x_{(k-1)}),(1+b^x_{(k-1)})}, \rho^x_{(1+a^x_{(k-1)}),(b^x_{(k-1)})}\} \end{aligned} \tag{3.8}$$

de donde se pueden dar los siguientes casos

$$r^x(a^x_{k-1}, b^x_{k-1}) = \rho_{(1+a^x_{(k-1)}),(1+b^x_{(k-1)})} \implies \begin{cases} a^x_k = 1 + a^x_{k-1} \\ b^x_k = 1 + b^x_{k-1} \end{cases} \tag{3.9}$$

$$r^x(a^x_{k-1}, b^x_{k-1}) = \rho_{(a^x_{(k-1)}),(1+b^x_{(k-1)})} \implies \begin{cases} a^x_k = a^x_{k-1} \\ b^x_k = 1 + b^x_{k-1} \end{cases} \tag{3.10}$$

$$r(a^x_{k-1}, b^x_{k-1}) = \rho_{(1+a^x_{(k-1)}),(b^x_{(k-1)})} \implies \begin{cases} a^x_k = 1 + a^x_{k-1} \\ b^x_k = b^x_{k-1} \end{cases} \tag{3.11}$$

W^x es diferente para cada articulación, por tanto, es necesario aplicar el mismo camino de deformación a todo el conjunto de articulaciones de tal forma que no se altere la acción en el espacio de trabajo para cada demostración al alinear cada articulación.

El camino de deformación promedio de alineación para ϕ número de articulaciones se define como

$$l_{min} = \text{mín}\{l_1, l_2, l_3, ..., l_\phi\} \tag{3.12}$$

$$\begin{aligned} w^*_1 &= \frac{1}{\phi}(w^1_1 + w^2_1 + w^3_1 + \ldots + w^\phi_1) \\ w^*_2 &= \frac{1}{\phi}(w^1_2 + w^2_2 + w^3_2 + \ldots + w^\phi_2) \\ &\vdots \\ w^*_{l_{min}} &= \frac{1}{\phi}(w^1_{l_{min}} + w^2_{l_{min}} + w^3_{l_{min}} + \ldots + w^\phi_{l_{min}}) \end{aligned} \tag{3.13}$$

$$W^* = w^*_1, w^*_2, ..., w^*_{l_{min}} \tag{3.14}$$

Por tanto, las nuevas series de tiempo alineadas están dadas por

$$q^{1*}_x(a^*_k) = \{q^1_x(a^*_1), q^1_x(a^*_2), q^1_x(a^*_3), ..., q^1_x(a^*_{l_{min}})\} \tag{3.15}$$

$$q^{2*}_x(b^*_k) = \{q^2_x(b^*_1), q^2_x(b^*_2), q^2_x(b^*_3), ..., q^2_x(b^*_l)\} \tag{3.16}$$

3.1.1. Cambio de velocidad de demostraciones

Si dos series de tiempo cumplen la misma trayectoria a distinta velocidad, estas coincidirán en un patrón con los valores muestreados, pues todos los valores de una de las trayectorias están contenidos en la otra, y separados siempre por un número constante de valores intermedios. Por lo que, si a_k^* y b_k^* son escogidos de tal manera que formen un camino de deformación donde $q_x^1(n)$ coincida en la misma trayectoria que $q_x^2(n)$ a diferente velocidad, el siguiente algoritmo se puede deducir

Dada cualquier señal articular discreta

$$q_x^y(n) = \{q_x^y(1), q_x^y(2), q_x^y(3), ..., q_x^y(i), ..., q_x^y(N)\} \tag{3.17}$$

Se define el paso de datos intermedios de muestreo en base a la velocidad deseada

$$j = redondeo\left(\frac{1}{vel}\right) \tag{3.18}$$

donde la velocidad deseada $vel \in (0, 1]$

Entonces

$$\begin{aligned} k_1 &= j \\ k_2 &= k_1 + j \\ &\vdots \\ k_{\tilde{N}} &= k_{\tilde{N}-1} + j \end{aligned} \tag{3.19}$$

donde $\tilde{N} = round\left(\frac{N}{j}\right)$.

Por lo que

$$q_x^{vel}(k_{\tilde{n}}) = \{q_x^{vel}(k_1), q_x^{vel}(k_2), ..., q_x^{vel}(k_{\tilde{N}})\} \tag{3.20}$$

3.2. Análisis de frecuencia en demostraciones

El análisis de frecuencia en las señales articulares provenientes de demostraciones que son realizadas por maestros humanos es particularmente útil, pues con esto es posible reconocer la frecuencia promedio con la que mueven las articulaciones del robot a lo largo de las demostraciones y con ello filtrar las frecuencias provenientes de temblores musculares producidos por el maestro humano. La trasformada discreta de Fourier es utilizada para hacer este análisis, pasando una señal articular en tiempo discreto al dominio de la frecuencia.

La trasformada discreta de Fourier es una secuencia y no una función de una variable continua como lo es la trasformada de Fourier convencional, esta corresponde a muestras equidistantes en frecuencia de la trasformada de Fourier de tiempo discreto [18].

Considérese una secuencia $q(n)$ proveniente de una señal articular periódica con periodo T, de manera que para cualquier entero n y r

$$q(n) = q(n + rT) \tag{3.21}$$

Al igual que con las señales periódicas en tiempo continuo, las secuencias pueden ser representadas por una serie de Fourier correspondiente a una suma de armónicos relacionados a exponenciales complejos con frecuencias que son enteros múltiples de la frecuencia fundamental $(2\pi/T)$ que se pueden asociar a la secuencia $q(n)$. Estos exponenciales complejos periódicos son

$$e_k(n) = e^{j2\pi/N} = e_k(n + rT) \tag{3.22}$$

Donde k es un entero y la series de Fourier se puede representar de la forma

$$q(n) = \frac{1}{T} \sum_{k=0}^{T-1} Q(k) e^{j2\pi kn/T} \tag{3.23}$$

Y de manera contraria se expresa

$$Q(k) = \sum_{n=0}^{T-1} q(n) e^{-j2\pi kn/T} \tag{3.24}$$

En donde la relación entre ambas secuencias $Q(k)$ y $q(n)$ periódicas se puede escribir como

$$q(n) \overset{DFT}{\longleftrightarrow} Q(k) \tag{3.25}$$

3.2.1. Propiedades de la transformada discreta de Fourier

Se enlistan de manera general las propiedades de la transformada discreta de Fourier [18] a continuación:

Linealidad

Considérese dos secuencias periódicas $x_1(n)$ y $x_2(n)$, ambas con periodo T, en donde

$$x_1(n) \overset{DFT}{\longleftrightarrow} X_1(k) \tag{3.26}$$

$$x_2(n) \overset{DFT}{\longleftrightarrow} X_2(k) \tag{3.27}$$

Entonces se da que

$$ax_1(n) + bx_2(n) \overset{DFT}{\longleftrightarrow} aX_1(k) + bX_2(k) \tag{3.28}$$

Desplazamiento

Considérese una secuencia periódica $x(n)$, entonces $x(n-m)$ es un desplazamiento de $x(n)$

$$x(n-m) \overset{DFT}{\longleftrightarrow} e^{-j2\pi km/N} X(k) \tag{3.29}$$

La ecuación anterior es valida para cualquier desplazamiento $0 < m \leq T-1$. De la misma manera, se cumple que

$$e^{j2\pi nl/T} x(n) \overset{DFT}{\longleftrightarrow} X(k-l) \tag{3.30}$$

Dualidad

Si se cumple que la trasformada discreta de Fourier de una secuencia periódica $x(n)$ es

$$x(n) \overset{DFT}{\longleftrightarrow} X(k) \tag{3.31}$$

entonces se cumple que

$$X(n) \overset{DFT}{\longleftrightarrow} Nx(-k) \tag{3.32}$$

Propiedades de simetría

Se tiene que

$$x(n) = x(((n))_T) \tag{3.33}$$

Se define la trasformada de Fourier Discreta para una secuencia conjugada como

$$x^*(n) \overset{DFT}{\longleftrightarrow} X^*(((-k))_T), \qquad 0 \leq n \leq T-1 \tag{3.34}$$

y

$$x^*(((-n))_T) \overset{DFT}{\longleftrightarrow} X^*(k), \qquad 0 \leq n \leq T-1 \tag{3.35}$$

Una secuencia periódica puede ser descompuesta en la suma de una secuencias conjugada simétrica y una conjugada antisimétrica.

La componente conjugada simétrica es

$$x_e(n) = \frac{1}{2}\{x(n) + x^*(-n)\} \qquad 0 \leq n \leq T-1 \tag{3.36}$$

mientras que la componente antisimétrica conjugada es

$$x_o(n) = \frac{1}{2}\{x(n) - x^*(-n)\} \qquad 0 \leq n \leq T-1 \tag{3.37}$$

Por lo que

$$x(n) = x_e(n) + x_o(n) \tag{3.38}$$

Las anteriores ecuaciones lleva a

$$Re\{x(n)\} \overset{DFT}{\longleftrightarrow} X_e(k) \tag{3.39}$$

$$jIm\{x(n)\} \overset{DFT}{\longleftrightarrow} X_o(k) \tag{3.40}$$

$$x_e(n) \overset{DFT}{\longleftrightarrow} Re\{X(k)\} \tag{3.41}$$

$$x_o(n) \overset{DFT}{\longleftrightarrow} jIm\{X(k)\} \tag{3.42}$$

Convolución

Dado que la multiplicación de los coeficientes de las serie discreta de Fourier provenientes de dos secuencias periódicas corresponden a la convolución de secuencias.

$$X_3(k) = X_1(k)X_2(k) \tag{3.43}$$

Para determinar $x_3(n)$ se sigue que

$$x_3(n) = \sum_{m=0}^{T-1} x_1(m)x_2(n-m) \qquad 0 \leq n \leq T-1 \tag{3.44}$$

Esta es llamada convolución circular punto-T, se identifica el hecho de que ambas secuencias tienen longitud T. Se denota como

$$x_3(n) = x_1(n) \bigcirc x_2(n) \tag{3.45}$$

3.3. Filtrado de señales articulares discretas

Para el diseño de filtros en tiempo discreto es necesario determinar los parámetros de una función de trasferencia o una ecuación diferencial que aproxime una respuesta deseada con tolerancias especificas.

El diseño de filtros se divide en dos categorías

- Respuesta infinita al impulso (IIR), se diseña obteniendo una función de transferencia aproximada
- Respuesta finita al impulso (FIR), se diseña a partir de aproximación polinomial

3.3.1. Filtro FIR

Existen tres métodos comúnmente usados para diseñar filtros FIR

- El método ventana, en donde se genera un filtro IIR ideal con un cierto ancho de banda, posteriormente se elige una función ventana de longitud finita que truncara la respuesta de impulso infinito, atenuando así la banda de parada.
- El método de muestreo de frecuencia, donde se muestrea uniformemente la respuesta en frecuencia deseada, posteriormente se computa la trasformada discreta inversa de Fourier de estas muestras para obtener una respuesta al impulso.
- El método Equiripple, este método diseña el filtro mediante la ecualización de la amplitud de onda en la banda de paso y en la de parada.

3.3.2. Filtro IIR

Las técnicas de diseño de filtros digitales de tipo IIR se basan en la trasformación de filtros analógicos a tiempo discreto, debido a que las técnicas de diseño de filtros analógicos de tipo IIR están en un punto muy avanzado, pues históricamente llevan más tiempo siendo desarrollados.

Trasformación bilineal

La trasformación algebraica que mapea entre el espacio de tiempo continuo y el espacio de tiempo discreto, la cual corresponde a la sustitución de s en la función de trasferencia por

$$s = \frac{2}{T_s}\frac{1 - z^{-1}}{1 + z^{-1}} \tag{3.46}$$

y de manera contraria

$$z = \frac{1 + \frac{T_s}{2}s}{1 - \frac{T_s}{2}s} \tag{3.47}$$

Filtro de Butterworth

La magnitud deseada se define como

$$|H(jw)|^2 = \frac{1}{1 + \left(\frac{w}{w_c}\right)^{2n}} \tag{3.48}$$

Las raíces del polinomio característico están dadas por la expresión

$$p_k = w_c \exp \frac{j\pi(2k+n+1)}{2n} \tag{3.49}$$

donde $k = 0, 1, ..., n-1$, w_c es la frecuencia de corte en *rad/seg* y n el orden del filtro. Por tanto, el polinomio característico es

$$p(s) = \prod_{k=0}^{n-1} (s - p_k) \tag{3.50}$$

w

y la función de transferencia es

$$H(s) = \frac{w_c^2}{p(s)} \tag{3.51}$$

Esta función de trasferencia representa un filtro analógico de orden n con frecuencia de corte en w_c

Filtro Chebyshev

El filtro Chebyshev esta basado en los polinomios de Chebyshev, los cuales se definen como

$$T_N(x) = \cos(N \cos^{-1}(x)) \qquad |x| \leq 1 \tag{3.52}$$

$$T_N(x) = \cosh(N \cosh^{-1}(x)) \qquad |x| > 1 \tag{3.53}$$

donde N es el orden del polinomio. La magnitud deseada se define como

$$|H(w)|^2 = \frac{1}{1 + \epsilon^2 T_N(\frac{w}{w_c})^2} \tag{3.54}$$

El parámetro ϵ determina, para un N dado, la compensación entre la onda de la banda de paso y la de transición. Las raíces del polinomio característico están dadas por la expresión

$$p_k = \gamma_k + j w_k \tag{3.55}$$

donde

$$\gamma_k = A w_c \cos\left(\frac{pi}{2} + \frac{\pi(2k+1)}{2N}\right) \tag{3.56}$$

$$w_k = B w_c \sin\left(\frac{pi}{2} + \frac{\pi(2k+1)}{2N}\right) \tag{3.57}$$

y

$$A = \frac{\left(\sqrt[N]{\alpha} - \sqrt[N]{1/\alpha}\right)}{2} \tag{3.58}$$

$$B = \frac{\left(\sqrt[N]{\alpha} + \sqrt[N]{1/\alpha}\right)}{2} \tag{3.59}$$

$$\alpha = \frac{1}{\epsilon} + \sqrt{1 + \frac{1}{\epsilon^2}} \tag{3.60}$$

Por tanto, la función de trasferencia del filtro Chebyshev esta dada por la expresión

$$H(s) = \frac{K}{p(s)} \tag{3.61}$$

donde $p(s)$ se define de la misma manera que en la ecuación 3.50 y K se elige de manera que la función de magnitud a $w = 0$ es igual a 1 para N impar o igual a $1/\sqrt{1+\epsilon^2}$ para N par.

Capítulo 4

Aprendizaje del comportamiento humano usando redes neuronales

Una red neuronal es un modelo que intenta simular un método de procesamiento básico de información de un cerebro biológico. Ya que los cerebros biológicos pueden realizar tareas complejas se ha encontrado bastante útil encontrar modelos de estos que puedan resolver problemas complejos.

Una red neuronal consta de elementos llamados neuronas que se encuentran interconectadas, realizando un procesamiento local dentro de la red. La complejidad de las estructuras y la fuerza de las conexiones están determinadas por la función que desempeña la red.

Una red neuronal puede desempeñar una gran variedad de tareas, entre las cuales destacan predicción y aproximación de funciones, clasificación de patrones, agrupamiento y pronostico.

Una neurona biológica consta de tres partes principales:

- Dendritas, estas canalizan las señales de entrada, las cuales adquieren un peso mediante la fortaleza en las conexiones hacia el soma.
- Soma, la cual acumula la señal de entrada con su respectivo peso y posteriormente procesar la señal.
- Axón, el cual trasmite la señal de salida hacia otras neuronas a las que se encuentra conectada.

En [19] se define una red neuronal como un procesador masivo distribuido paralelamente el cual tiene una tendencia natural por almacenar conocimiento experimental y poniéndolo disponible para su uso. Se asemeja al cerebro en dos aspectos:

- El conocimiento es adquirido por la red a través de un proceso de aprendizaje.

- La fuerza en las conexiones de las neuronas conocida como pesos sinápticos o pesos son usados para almacenar conocimiento.

El modelo de neurona artificial de la misma manera que la biológica, cuenta con tres elementos principales:

- Conjunto de sinapsis o enlaces de conexión, al igual que en las dendritas, aquí es donde se caracteriza mediante un peso que le asigna la fortaleza en las conexiones, una señal x_j en la entrada sináptica j la cual esta conectada a una neurona κ es multiplicada por un peso sináptico $\alpha_{j\kappa}$
- Punto de suma, en donde se suman todas las señales de entrada multiplicadas por el peso sináptico y que encuentran conectadas a la neurona.
- Función de activación, esta limita la amplitud de la señal de salida de la neurona.

En términos matemáticos se describe una neurona κ por las ecuaciones

$$u_\kappa = \sum_{j=1}^{\psi} \alpha_{j\kappa} x_j + \alpha_{0\kappa} \tag{4.1}$$

y

$$y_\kappa = G(u_\kappa) \tag{4.2}$$

donde $x_1, x_2, x_3, ..., x_\psi$ son las señales de entrada, $\alpha_{0\kappa}$ es el umbral, $\alpha_{1\kappa}, \alpha_{2\kappa}, ..., \alpha_{\psi\kappa}$ son los pesos sinápticos de la neurona κ y $G(\cdot)$ es la función de activación.

Las funciones de activación comúnmente usadas se describen en el cuadro 4.1.

Función escalón unitario	$G(u_k) = \begin{cases} 1 & u_k \leq 0 \\ 0 & u_k > 0 \end{cases}$
Función sigmoide	$G(u_k) = \frac{1}{1+e^{-u_k}}$
Función tangente hiperbólica	$G(u_k) = \frac{1+e^{-u_k}}{1-e^{-u_k}}$
Función gaussiana	$G(u_k) = e^{-u_k^2}$

Cuadro 4.1: Funciones de activación comúnmente usadas

4.1. Red neuronal con conexiones hacia delante

Las redes neuronales con conexión hacia delante permiten a las señales de entrada ir solo en una dirección, de la entrada a la salida. En esta arquitectura no existen ciclos o conexiones hacia atrás. Las redes con conexiones hacia delante son ampliamente conocidas y usadas en un gran numero de campos dada su gran capacidad que poseen. Se diferencian dos estructuras principales

- Red monocapa con conexiones hacia delante, esta arquitectura cuenta con una capa de entrada en donde los nodos fuente no realizan ningún cómputo, estos proyectan las señales de entrada hacia la capa de salida, en donde las neuronas de esta capa realizan todo el cómputo.
- Red multicapa con conexiones hacia delante, esta arquitectura se distingue por tener una o más capas ocultas, a las neuronas de estas capas se les llama neuronas ocultas. Los nodos fuente se encuentran conectados a las neuronas de la primera capa oculta, y de existir más de una capa oculta, estas se encuentran conectadas a la segunda capa oculta y así sucesivamente, de manera que la última capa oculta se conecta con la capa de salida, se dice que la red neuronal esta totalmente conectada cuando cada uno de los nodos en cada capa se encuentra conectado a todos los nodos de la siguiente. Al estar las neuronas conectadas de esta manera permite distribuir la cantidad de cómputo entre un mayor numero de neuronas, dándoles así una mayor capacidad de procesamiento.

4.1.1. Red con conexiones hacia delante de una capa oculta

Se sabe a partir de [20] que una red neuronal con conexiones hacia delante con una única capa oculta y al menos κ número de neuronas ocultas permite casi con cualquier función de activación no lineal aprender exactamente κ distintas observaciones.

Para un número ψ de cualesquiera entradas y κ neuronas sin umbrales, se tiene que las ecuaciones 4.1 y 4.2 en forma matricial se puede expresar como

$$u = A^T x \tag{4.3}$$

y

$$v = G(u) \tag{4.4}$$

donde

$$A = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1\kappa} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2\kappa} \\ \vdots & \vdots & & \vdots \\ \alpha_{\psi 1} & \alpha_{\psi 2} & \dots & \alpha_{\psi\kappa} \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_\psi \end{bmatrix} \tag{4.5}$$

Para la capa de salida con un número ϕ de salidas, de igual manera podemos escribir de forma matricial

$$y = Bv \tag{4.6}$$

donde

$$B = \begin{bmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1\kappa} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2\kappa} \\ \vdots & \vdots & & \vdots \\ \beta_{\phi 1} & \beta_{\phi 2} & \cdots & \beta_{\phi\kappa} \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \hat{q}_\phi \end{bmatrix} \tag{4.7}$$

4.1.2. Paradigmas de aprendizaje

El aprendizaje es el proceso que pasa la red neuronal en la cual modifica sus pesos sinápticos asociados a las neuronas según los datos de la señal de entrada. Los cambios producidos en el proceso de aprendizaje puedes ser visto como el fortalecimiento o debilitación de las conexiones entre las neuronas.

De manera general existen dos paradigmas en el aprendizaje de las redes neuronales:

- Aprendizaje supervisado
- Aprendizaje no supervisado

La diferencia fundamental radica en la presencia o ausencia de un agente externo que controla este proceso, este agente se suele llamar supervisor o maestro.

Aprendizaje supervisado

El aprendizaje supervisado utiliza el conocimiento del entorno que se puede representar como un conjunto de ejemplos que le proveen a la red neuronal una respuesta deseada para un estímulo especifico. Esta respuesta deseada representa la acción óptima a desempeñar por la red neuronal. De esta manera, el conocimiento del entorno es trasferido a la red neuronal por medio de un entrenamiento.

Este aprendizaje a su vez se puede dividir en

- Aprendizaje por corrección de error.
- Aprendizaje por refuerzo.
- Aprendizaje estocástico.

Aprendizaje no supervisado

En el aprendizaje no supervisado no existe un maestro o supervisor que se encargue de trasferir el conocimiento a la red neuronal, en cambio, existe una tarea que mide la calidad de la salida

En este paradigma de aprendizaje destaca el aprendizaje competitivo y cooperativo, en las cuales las neuronas adquieren un comportamiento competitivo o cooperativo con el objetivo de realizar una tarea. Las neuronas tienen una competición en

todas las capas, en las cuales queda una neurona, o una por grupo como ganadora y las demás se mantienen desactivadas. Si el aprendizaje es cooperativo las conexiones con las neuronas vecinas se excitan, reforzando así las conexiones sinápticas.

4.2. Red neuronal para el aprendizaje del comportamiento humano

A partir del capitulo 4.1.1 se propone una red neuronal con conexiones hacia delante de una capa oculta mostrada en la figura 4.1, de manera que sea capaz de mapear entre el espacio de trabajo a el espacio articular del robot.

Ya que la entrada de la red neuronal debe estar dada por coordenadas espaciales que describan una trayectoria en tres diferentes series de tiempo $x(n)$, $y(n)$ y $z(n)$ se propone una estructura de red de tiempo retrasado, la cual incorpora una memoria dinámica de orden p, se sabe que esto ayuda al modelado de series de tiempo [19] capturando la información temporal de la señal de entrada por la neurona e incrustándola en sus propios pesos sinápticos.

Por lo que la relación entre entradas y los pesos sinápticos para un número κ de neuronas de la capa oculta se puede expresar de manera matricial como

$$u = A^T x \tag{4.8}$$

donde la matriz de pesos sinápticos A y el vector de entradas x son

$$A = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1\kappa} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2\kappa} \\ \vdots & \vdots & & \vdots \\ \alpha_{\psi 1} & \alpha_{\psi 2} & \dots & \alpha_{\psi\kappa} \end{bmatrix} \qquad x = \begin{bmatrix} x(n) \\ x(n+1) \\ x(n+2) \\ \vdots \\ x(n+p) \\ y(n) \\ y(n+1) \\ y(n+2) \\ \vdots \\ y(n+p) \\ z(n) \\ z(n+1) \\ z(n+2) \\ \vdots \\ z(n+p) \end{bmatrix} \tag{4.9}$$

donde el número de entradas es $\psi = 3(p+1)$ y p la memoria dinámica.

De la misma forma se expresa el vector

$$v = G(u) \tag{4.10}$$

donde $G(\cdot)$ es la función de activación tangente hiperbólica.

Para la capa de salida con ϕ número de distintas salidas, donde $\hat{q}_\phi$ será la señal articular aproximada correspondiente a la articulación ϕ y se expresa como

$$\hat{q} = Bv \tag{4.11}$$

y de manera equivalente

$$\hat{q} = BG(A^T x) \tag{4.12}$$

donde la matriz de pesos sinápticos y el vector de salidas son

$$B = \begin{bmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1\kappa} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2\kappa} \\ \vdots & \vdots & & \vdots \\ \beta_{\phi 1} & \beta_{\phi 2} & \cdots & \beta_{\phi\kappa} \end{bmatrix} \qquad \hat{q} = \begin{bmatrix} \hat{q}_1(n) \\ \hat{q}_2(n) \\ \vdots \\ \hat{q}_\phi(n) \end{bmatrix} \tag{4.13}$$

4.2.1. Algoritmos de aprendizaje supervisado por corrección de error para una red con conexiones hacia delante de una capa oculta

Para el entrenamiento de la red neuronal propuesta para el aprendizaje del comportamiento humano se presentan dos distintos algoritmos, *Backpropagtion*, el cual es un algoritmo clásico y *Extreme Learning Machine*, el cual es un algoritmo desarrollado por G. B. Huang, Q. Y. Zhu y C. K. Siew en [21].

Backpropagation

El algoritmo de *Backpropagation* es altamente usado para el entrenamiento de redes neuronales multicapa con conexiones hacia delante. El algoritmo se basa en el ajuste de los pesos sinápticos de la red en función del error entre la salida y los valores deseados, formando así un ciclo cerrado realimentado, en el cual el entorno no forma parte. El algoritmo consta de dos pasos:

- Paso hacia delante, en este paso se introduce la señal de entrada (estímulo) a las neuronas de forma tal que se propaga hacia delante desde la capa de entrada hasta la de salida.
- Paso hacia atrás, al finalizar el paso hacia delante se genera una señal de error en la capa de salida, esta se propaga hacia atrás por la red.

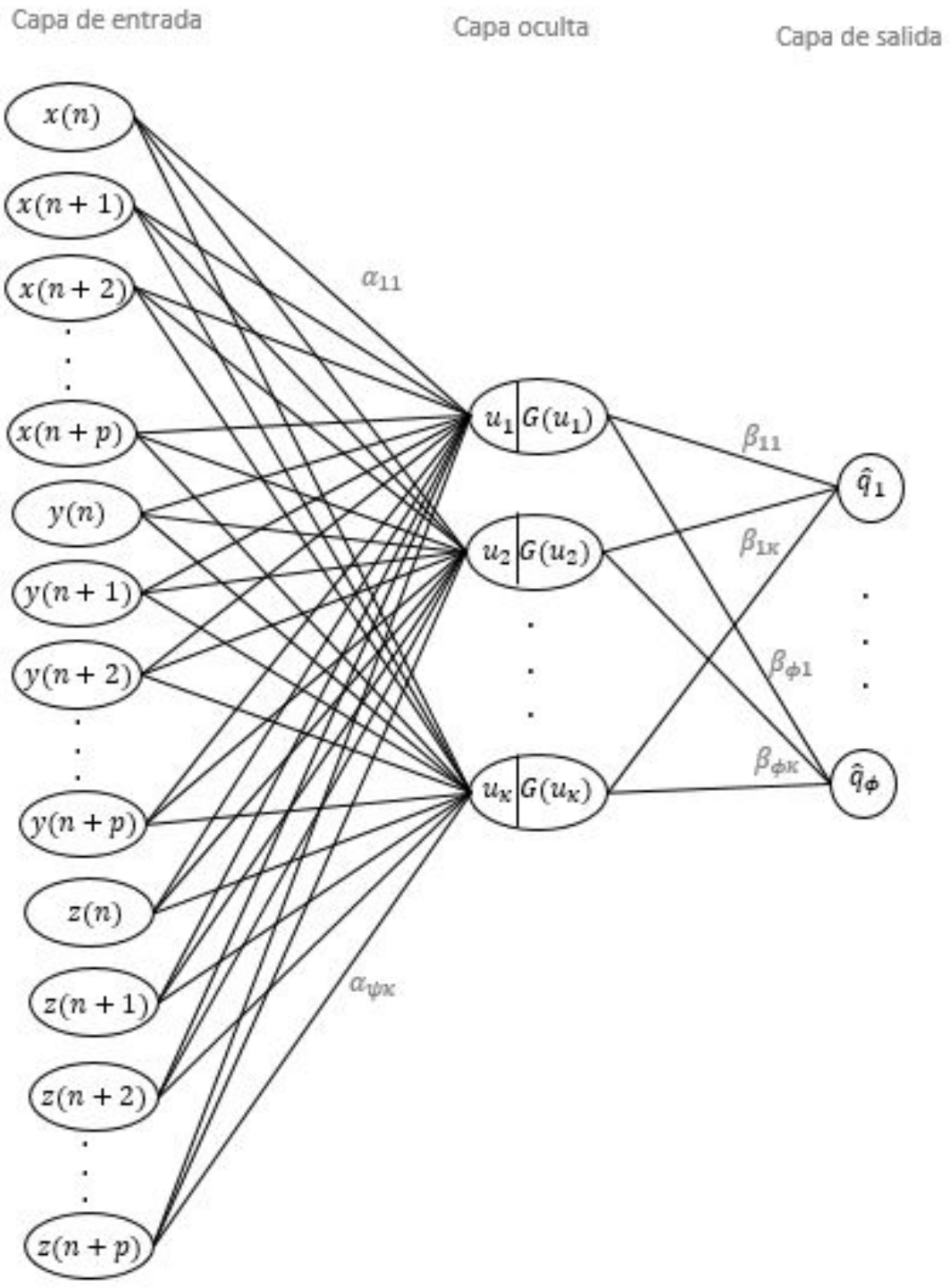


Figura 4.1: Arquitectura de la red neuronal propuesta para el mapeo de cinemática directa

Tomando en cuenta que la red tiene ψ numero de entradas, κ número de neuronas de la capa oculta y ϕ número de neuronas en la capa de salida, se definen las siguientes ecuaciones para ser entrenada por medio del algoritmo de *backpropagation*.

El error entre la respuesta deseada proveniente de las demostraciones alineadas y la salida de la neurona j para la iteración n se define por:

$$e_j(n) = \frac{1}{2}(q_j^{1*}(n) + q_j^{2*}(n)) - \hat{q}_j(n) \tag{4.14}$$

De modo que el error total para todas las neuronas en la capa de salida en la iteración n esta dado por:

$$\mathcal{E}(n) = \frac{1}{2}\sum_{j=1}^{\phi} e_j^2(n) \tag{4.15}$$

Se define el cambio en el error total respecto de los pesos sinápticos β_{ji} de las neuronas de la capa de salida por medio de la regla de la cadena de cálculo, expresando así el siguiente gradiente:

$$\frac{\partial \mathcal{E}(n)}{\partial \beta_{ji}(n)} = \frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial \hat{q}_j(n)} \frac{\partial \hat{q}_j(n)}{\partial \beta_{ji}(n)} \tag{4.16}$$

Para el cambio en el error total respecto de los pesos sinápticos α_{hi} de las neuronas de la capa oculta anterior a la capa de salida el gradiente se expresa como:

$$\frac{\partial \mathcal{E}(n)}{\partial \alpha_{hi}(n)} = \frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial \hat{q}_j(n)} \frac{\partial \hat{q}_j(n)}{\partial v_i(n)} \frac{\partial v_i(n)}{\partial u_i(n)} \frac{\partial u_i(n)}{\partial \alpha_{hi}(n)} \tag{4.17}$$

A partir de la ecuación 4.15 se puede obtener la derivada parcial del error total respecto del error en la entrada j

$$\frac{\partial \mathcal{E}(n)}{\partial e_j(n)} = e_j(n) \tag{4.18}$$

Partiendo de la ecuación 4.14 se tiene

$$\frac{\partial e_j(n)}{\partial \hat{q}_j(n)} = -1 \tag{4.19}$$

Por medio de la ecuación 4.11 se tiene

$$\frac{\partial \hat{q}_j(n)}{\partial \beta_{ji}(n)} = v_i(n) \tag{4.20}$$

y

$$\frac{\partial \hat{q}_j(n)}{\partial v_i(n)} = \beta_{ji} \tag{4.21}$$

A partir de la ecuación 4.10 se tiene

$$\frac{\partial \hat{v}_i(n)}{\partial u_i(n)} = \dot{G}(u_i) \tag{4.22}$$

De la misma forma que 4.20, se tiene

$$\frac{\partial u_i(n)}{\partial \alpha_{hi}(n)} = x_h \tag{4.23}$$

Por tanto, el cambio en el error total respecto a los pesos sinápticos de las neurona de la capa de salida se define por

$$\frac{\partial \mathcal{E}(n)}{\partial \beta_{ji}(n)} = -e_j(n)v_i(n) \tag{4.24}$$

Y el cambio en el error total respecto de los pesos sinápticos de las neuronas de la capa oculta esta definido por

$$\frac{\partial \mathcal{E}(n)}{\partial \alpha_{hi}(n)} = -e_j(n)\beta_{ji}\dot{G}(u_i(n))x_h \tag{4.25}$$

Por lo que la corrección en los pesos sinápticos $\Delta\alpha_{hi}(n)$ y $\Delta\beta_{ji}$ esta definido por:

$$\Delta a\alpha_{hi}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial \alpha_{hi}(n)} \tag{4.26}$$

$$\Delta\beta_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial \beta_{ji}(n)} \tag{4.27}$$

donde η es la tasa de aprendizaje y la regla delta esta dada por

$$\alpha_{hi}(n+1) = \alpha_{hi}(n) + \Delta\alpha_{hi}(n) \tag{4.28}$$

$$\beta_{ji}(n+1) = \beta_{ji}(n) + \Delta\beta_{ji}(n) \tag{4.29}$$

Extreme Learning Machine

En [21] y [22] se propone un método simple y muy eficiente para entrenar una red con conexiones hacia delante de una capa oculta, donde a partir de el planteamiento aleatorio de los pesos α_{hi}, los pesos conectados a la capa oculta β_{ji} son obtenidos minimizando el error de aproximación en el sentido del error cuadrático:

$$min_{B\in\mathbf{R}^{\phi\times\kappa}}|Bv - \frac{1}{2}(q^{1*} + q^{2*})|^2 \tag{4.30}$$

La solución óptima para 4.30 esta dada por

$$B^* = \frac{1}{2}v^{\dagger}(q^{1*} + q^{2*}) \tag{4.31}$$

Donde $v^{\dagger}$ es la inversa generalizada de Moore-Penrose de la matriz $G(u)$. Existe un gran número de métodos que permiten resolver el problema anterior mediante eliminación gaussiana, el método de proyección ortogonal, método iterativo o descomposición de valor singular (SVD)

Por lo que la salida de la capa oculta seria

$$\hat{q} = B^* v \tag{4.32}$$

4.2.2. Error de cinemática inversa

Para evaluar el desempeño del proceso de cinemática inversa que realiza la red neuronal se utiliza una trayectoria ($x(n)$, $y(n)$ y $z(n)$) diferente a la usada para el entrenamiento como prueba y se compara con la cinemática directa de las señales de salida de la red neuronal calculadas en las ecuaciones 2.22,2.23 y 2.24.

El error de cinemática inversa se calcula por medio del error cuadrático medio expresado como:

$$MSE = \frac{1}{3N}\sum_{n=1}^{N}\left[\left(x(n) - \hat{d}_x(n)\right)^2 + \left(y(n) - \hat{d}_y(n)\right)^2 + \left(z(n) - \hat{d}_z(n)\right)^2\right] \tag{4.33}$$

donde $\hat{d}_x(n)$, $\hat{d}_y(n)$ y $\hat{d}_z(n)$ son las coordenadas espaciales de la cinemática directa de las señales de salida de la red neuronal.

Capítulo 5

Resultados experimentales

Para las pruebas y experimentos se utilizó un ser humano para manipular directamente el cuerpo del robot aprendiz del capítulo 2 con la finalidad de trazar trayectorias deseadas en el espacio de trabajo del robot, siendo el ser humano quien controla la demostración y el robot aprendiz quien ejecuta la demostración (sección 2.2.1).

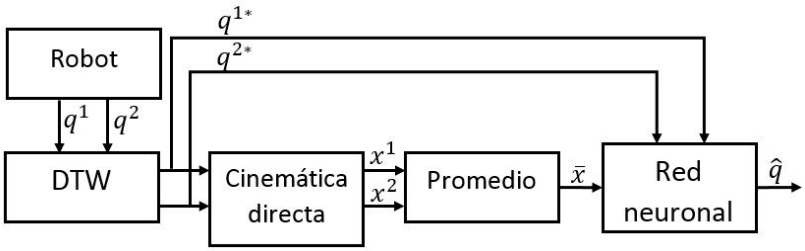
Se utilizó una técnica de demostración iterativa (sección 2.2.2).

Para la obtención de datos se utilizó una técnica de demostración (sección 2.3.2) donde al usar al aprendiz como plataforma de demostración se utilizó los sensores montados en el cuerpo del robot aprendiz por lo que no fue necesario un mapeo de almacenamiento ni de personificación (sección 2.3.1) se considera la técnica de demostración utilizada dentro de las excepciones (sección 2.3.4) pues solo se almacenaron los estados del robot y la acción fue generada mediante la cinemática directa del robot.

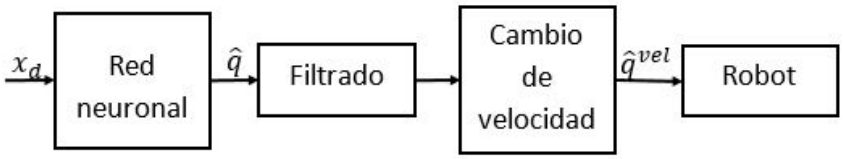
Como técnica de aprendizaje se utilizó un mapeo de función de tipo regresión implementado mediante una red neuronal. Se muestran las dos diferentes fases en el aprendizaje del comportamiento humano en 5.1a y 5.1b.

Como se menciona en la sección 2.1.1, la interfaz de control del robot se desarrolló en Matlab-Simulink. En la fase de aprendizaje de comportamiento se almacenan los estados experimentadas por el robot al momento de trazar las trayectorias deseadas, se usa el diagrama de la figura 5.2 y se utiliza la cinemática directa para generar las acciones, mientras que para la fase de generación de comportamiento aprendido por la red neuronal se utiliza el diagrama de la figura 5.3, en donde se compara los estados que se desean trazar con los que realmente se están trazando, controlados mediante el PID interno de los servomotores (figura 2.5). Ambos diagramas comprenden el bloque Robot de las fases de aprendizaje y generación de comportamiento respectivamente.

Para los caso de estudio se toma la decisión de fijar la articulación inferior q_1



(a) Fase de aprendizaje de comportamiento



(b) Fase de generación de comportamiento

Figura 5.1: Aprendizaje del comportamiento humano usando redes neuronales

en 180 grados para evitar la redundancia. El periodo de muestreo T es de 0.01 segundos

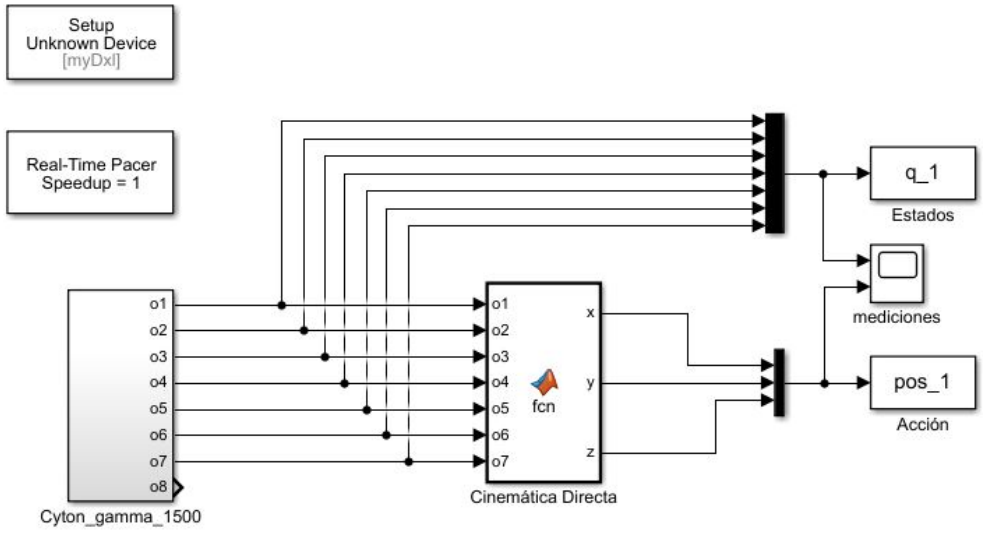


Figura 5.2: Almacenamiento de estados y acciones experimentados por el robot para la fase de aprendizaje del comportamiento humano

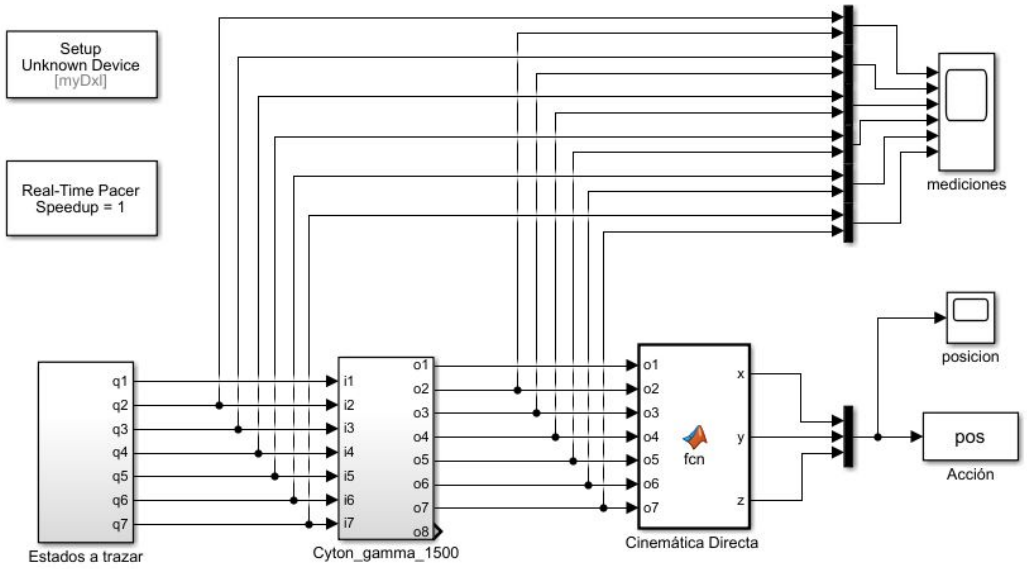


Figura 5.3: Trazado de estados en el robot en la fase de generación de comportamiento humano

5.1. Trayectoria especifica

En el ApD donde se encuentra involucrado el factor humano y se desea aumentar la generalidad del aprendizaje es difícil reproducir la misma trayectoria a la misma velocidad en cada demostración ejecutada, pues para el ser humano es difícil repetir exactamente la misma trayectoria y velocidad en cada demostración.

Para este caso de estudio se plantea entrenar la red neuronal mediante dos demostraciones similares que reproduzcan el comportamiento humano del dibujado de una figura que representa un ocho sobre un plano, realizadas a distinta velocidad. Se muestran los estados de la demostración en la figura 5.4 y la acción que es generada mediante la cinemática directa de los estados en la figura 5.5 en donde la primer demostración dura 5.25 segundos y la segunda 2.97 segundos.

5.1.1. Fase de aprendizaje de comportamiento

Para el aprendizaje de estas demostraciones es necesario hacer un alineamiento general de los estados, de manera que el camino de deformación aplicado para el alineamiento sea el mismo para cada estado, de manera que la acción que representan los estados no se vea alterada.

Se muestra en la figura 5.6 los estados alineados y en la figura 5.7 se muestra la cinemática directa de los estados alineados, donde ambas demostraciones duran 7.15 segundos.

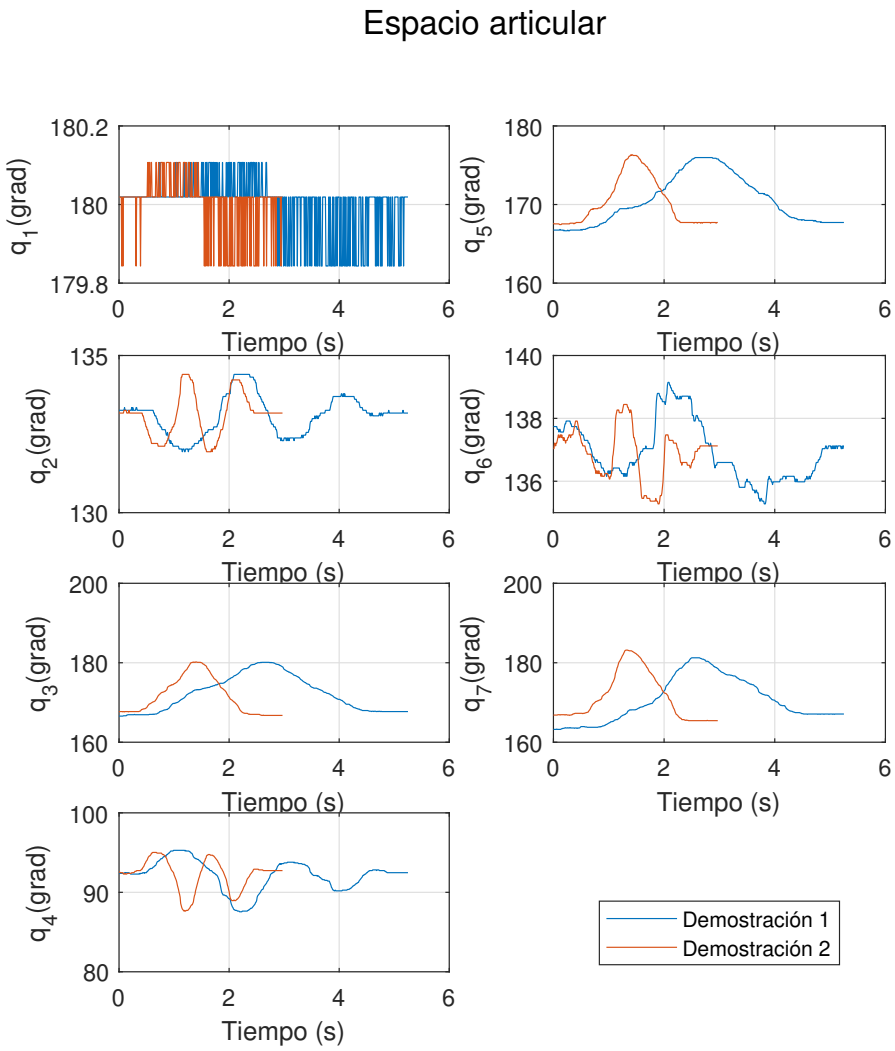


Figura 5.4: Estados de las demostraciones

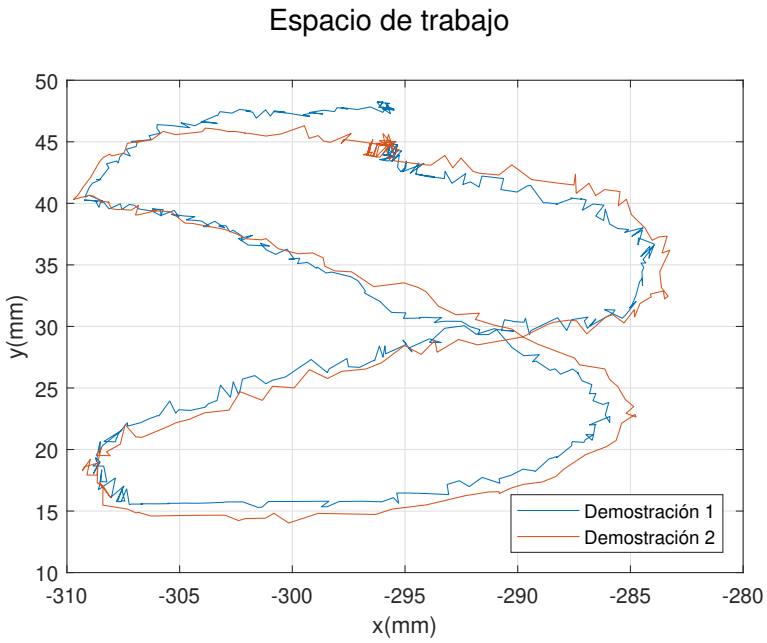


Figura 5.5: Acciones de las demostraciones generadas mediante cinemática directa

Una vez las demostraciones se encuentran alineadas la red neuronal puede aprender la generalidad de estas demostraciones, a mayor numero de demostraciones, mayor es la generalidad y menos posible reproducir el comportamiento original de una de las demostraciones.

Para el aprendizaje de la red neuronal propuesta (figura 4.1) en este caso de estudio se utilizaron un número de neuronas en la capa oculta $\kappa = 700$ y una memoria dinámica $p = 2$. Por tanto, el número de entradas es $psi = 9$. La señal de entrada a la red neuronal es la acción generada por el promedio aritmético de la cinemática directa de las demostración alineadas. Debido a que no existe necesidad de un mapeo de almacenamiento o de personificación, el numero de salidas de la red neuronal se mantiene igual al numero de estado en las demostraciones, siendo $\phi = 7$.

En la figura 5.8 se observa un aprendizaje de la red neuronal donde la respuesta tiene pequeñas variaciones en frecuencia altas debido a la complejidad que representa para la red neuronal aprender algunas trayectorias, esto produce que la acción de cinemática directa de los estados tenga vibraciones como se observa en la figura 5.9. Se propone un filtro pasa bajas para evitar las variaciones de frecuencias altas.

Diseño del filtro

Se toma el aprendizaje de la red neuronal para el estado de la articulación 6 para el diseño del filtro, pues visiblemente se puede ver que es el aprendizaje con mayor variación en frecuencias.

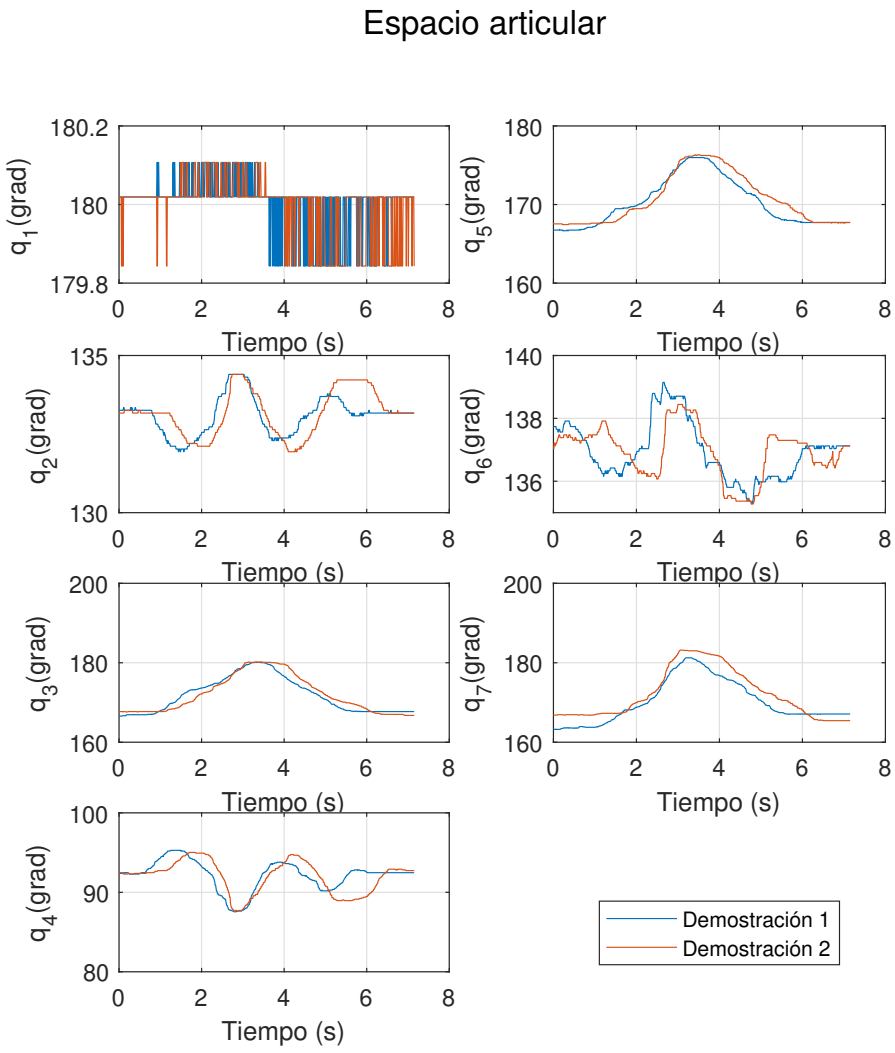


Figura 5.6: Alineación de los estados de las demostraciones mediante DTW

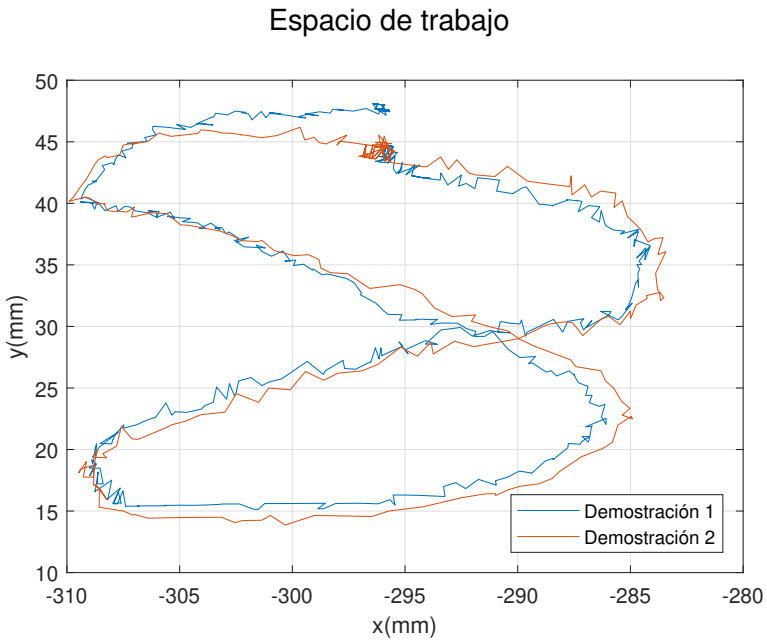


Figura 5.7: Acciones de las demostraciones alineadas generada mediante cinemática directa

A partir del análisis del espectro de frecuencias(sección 3.2) en la imagen 5.11 se desea mantener las frecuencias inferiores a 3.5 Hz y atenuar las superiores, por lo que se diseña un filtro pasa bajas Butterworth de orden 2 según las ecuaciones 3.49 se sigue que las raíces del filtro analógico son

$$p_0 = -15.5501 + 15.5501i \tag{5.1}$$

$$p_1 = -15.5501 - 15.5501i \tag{5.2}$$

por lo que el polinomio característico es

$$p(s) = s^2 + 31.1002s + 483.6112 \tag{5.3}$$

y su función de transferencia es entonces

$$H(s) = \frac{483.6112}{s^2 + 31.1002s + 483.6112} \tag{5.4}$$

mediante 3.46 se llega a la función de transferencia del filtro digital

$$H(z) = \frac{0.0104(z+1)(z+1)}{(s - 0.8455 - 0.1336i)(s - 0.8455 + 0.1336i)} \tag{5.5}$$

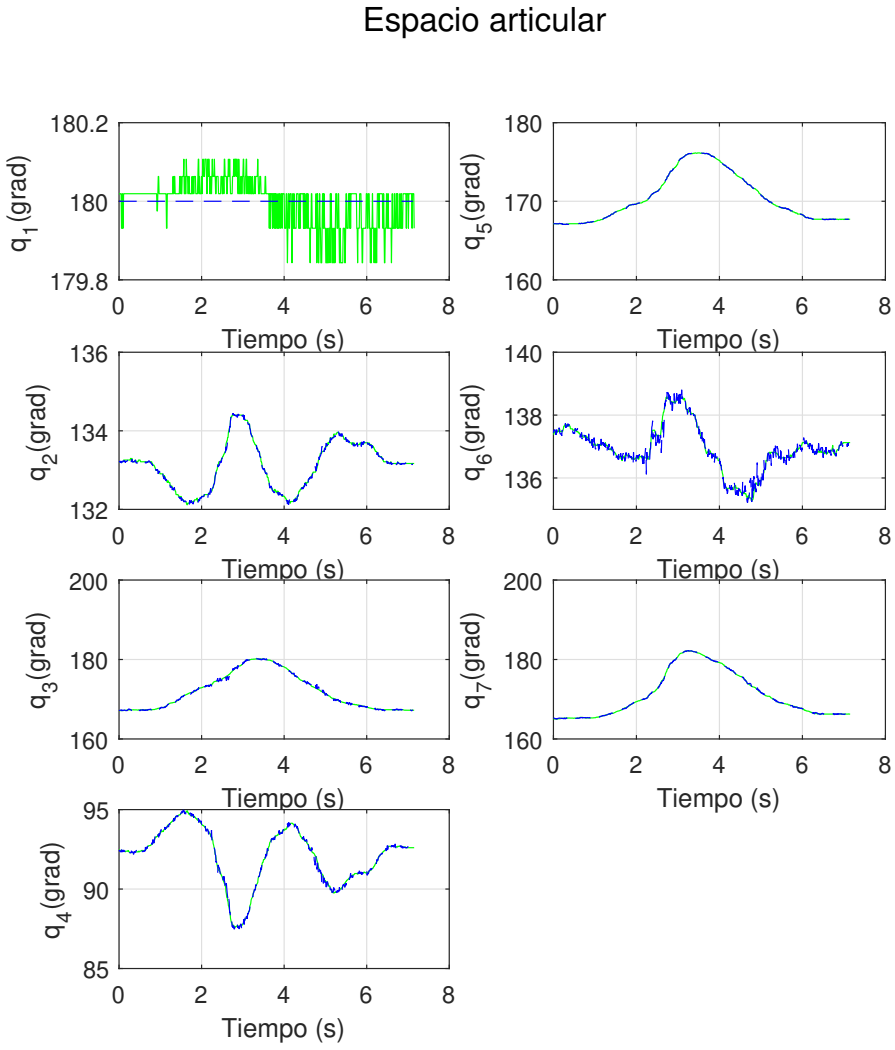


Figura 5.8: Aprendizaje de las demostraciones alineadas (azul) y promedio de las demostraciones alineadas (verde)

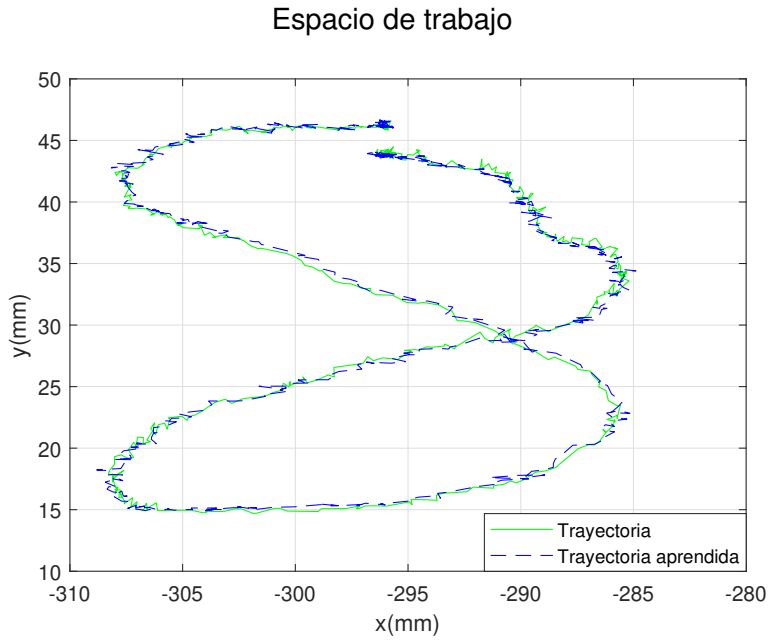


Figura 5.9: Acciones del aprendizaje de las demostraciones alineadas (azul) y del promedio de las demostraciones alineadas (verde) generadas mediante cinemática directa

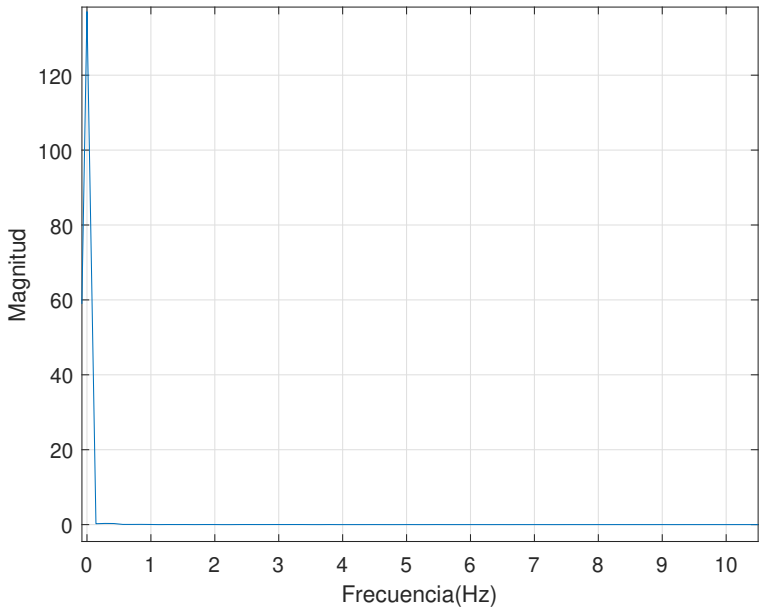


Figura 5.10: Espectro de frecuencias de q_6

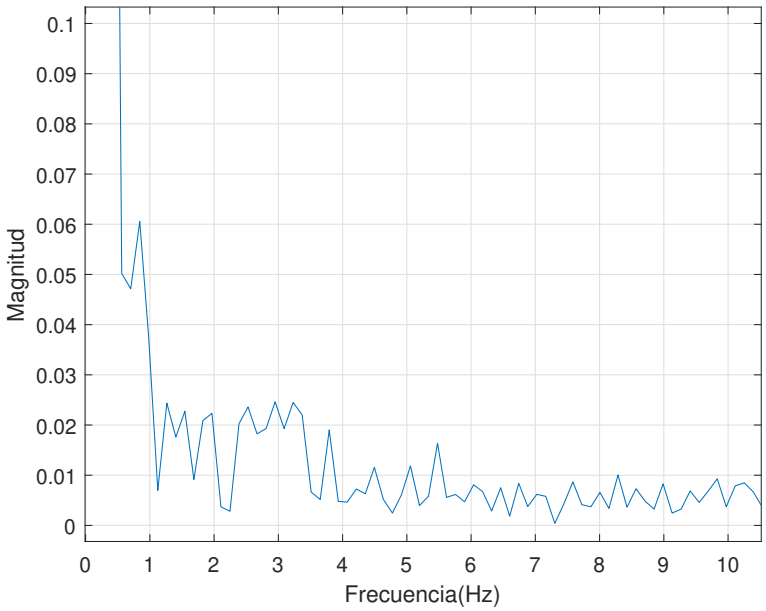


Figura 5.11: Acercamiento del espectro de frecuencias de q_6

La estabilidad de la función de transferencia 5.4 se muestra en la figura 5.12, donde se observa que los polos de la función se encuentran dentro del circulo unitario en el plano z.

El filtro digital aplicado al aprendizaje de las demostraciones de comportamiento se muestra en la figura 5.13 y la cinemática directa de estos estados filtrados se puede observar en la figura 5.14, en donde se aprecia que en la trayectoria resultante se han eliminado las vibraciones producidas por los temblores musculares del maestro al realizar las demostraciones.

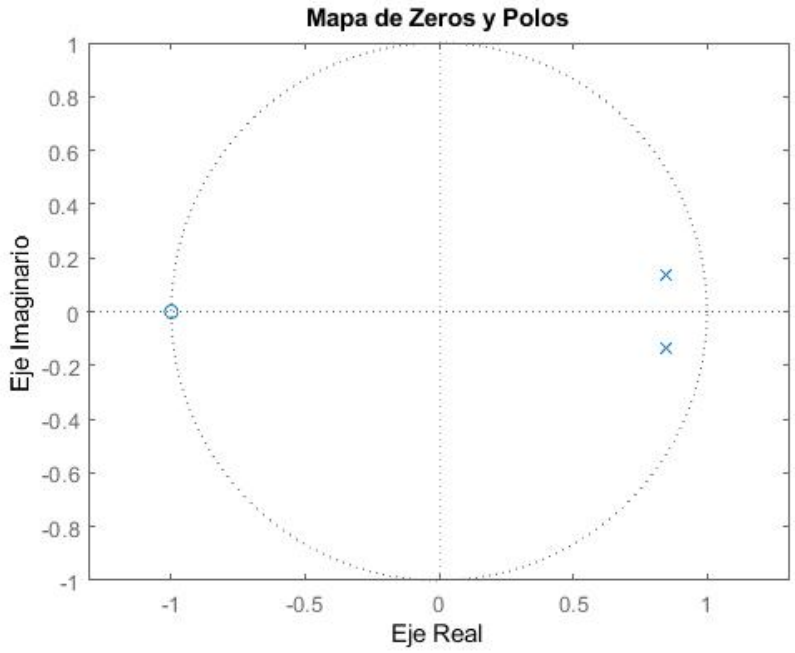


Figura 5.12: Plano z de la función de transferencia del filtro digital

En base a 3.20 se hace un cambio de velocidad en los estados filtrados del aprendizaje de la red neuronal. En la figura 5.15 se observa que al acelerar 10 veces la velocidad de los estados, la trayectoria de la cinemática directa el comportamiento humano se mantiene similar a la original (5.15a), mientras que al acelerar 20 veces la velocidad de los estados (5.15b) se comienzan a presentar pequeñas perdidas de datos que impactan en pequeñas deformaciones en la trayectoria de la cinemática directa, al acelerar 30 y 40 veces la velocidad de los estados la perdida de datos es muy grande impactando en una deformación significativa de la trayectoria de la cinemática directa original.

5.1.2. Fase de generación de comportamiento

Para realizar una prueba física en el robot del trazado de la trayectoria con un comportamiento humano natural se utiliza el trazado de los estados filtrados una vez se han acelerado, mediante el diagrama de simulink mostrado en la figura 5.3. Se aplica una aceleración a la velocidad de los estados de 6.66 mostrada en la figura 5.17.

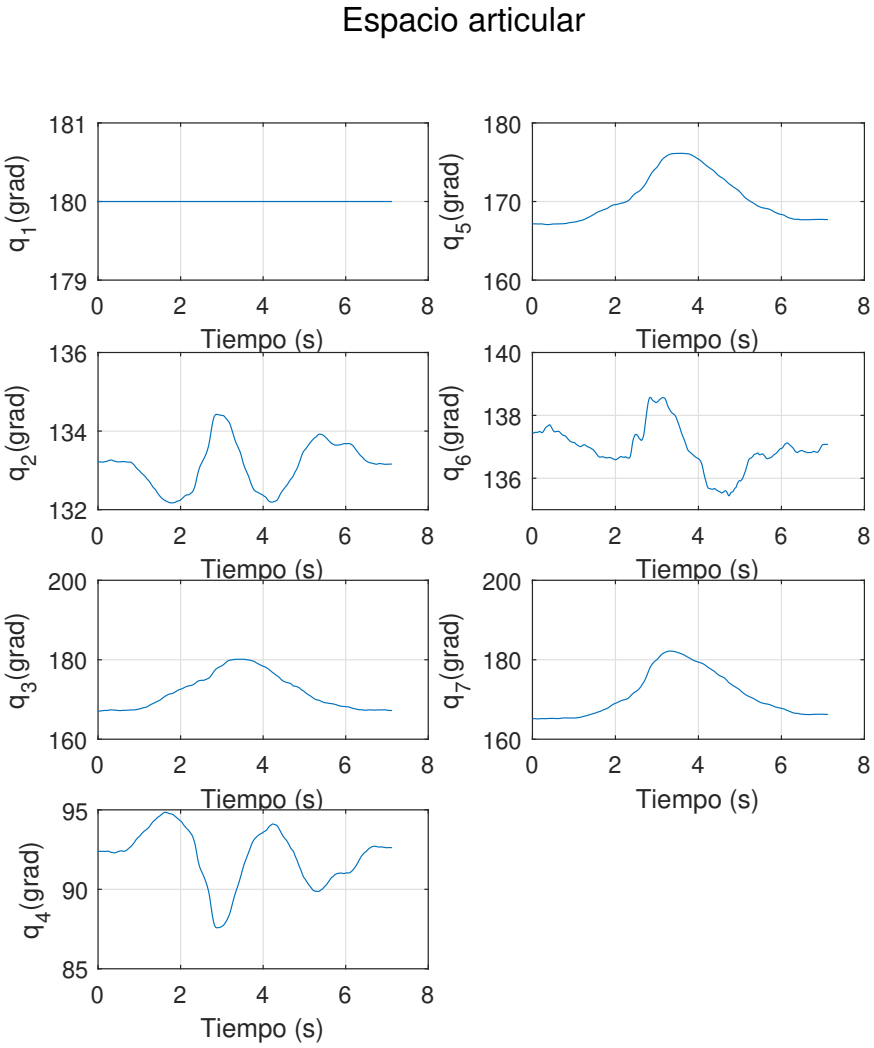


Figura 5.13: Filtrado del aprendizaje de la red neuronal

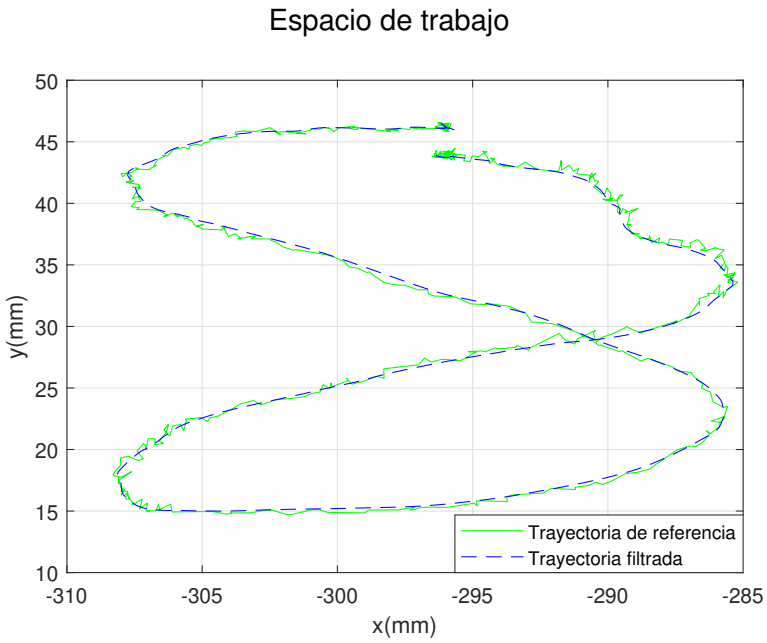


Figura 5.14: Cinemática directa del filtrado del aprendizaje de la red neuronal

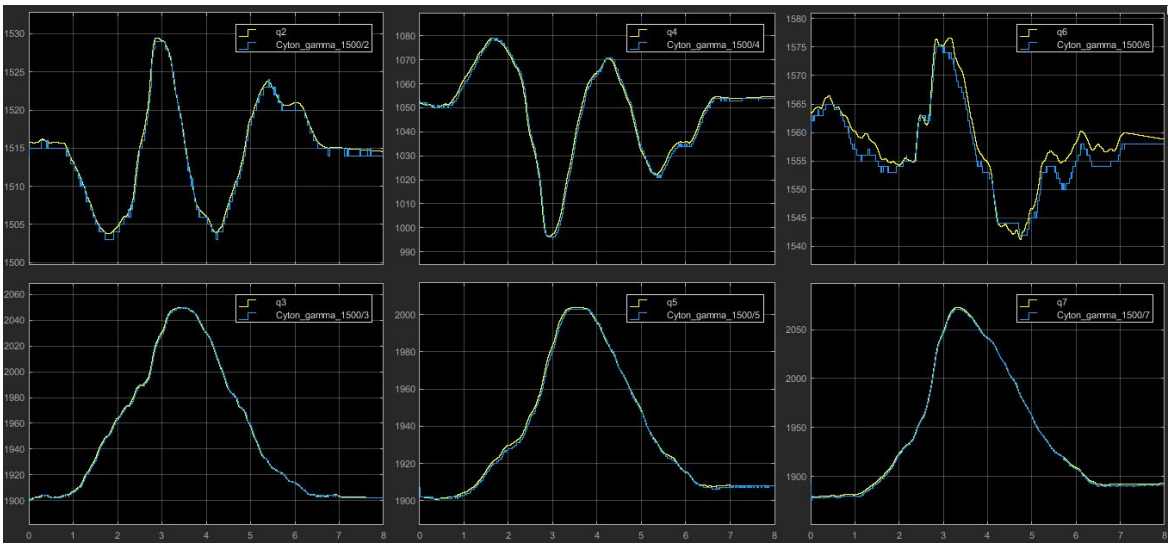
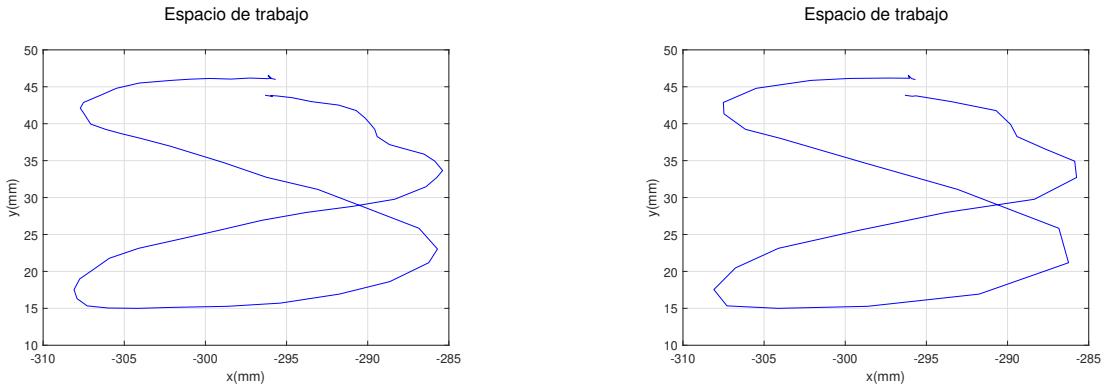
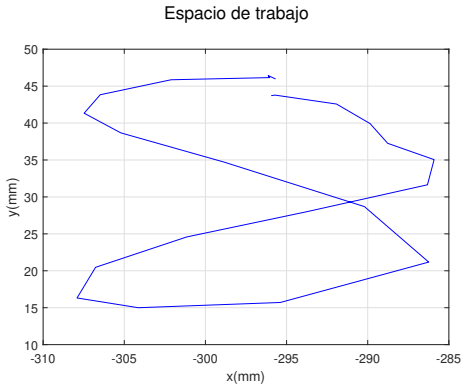


Figura 5.16: Seguimiento de las trayectorias sin antelación aplicada

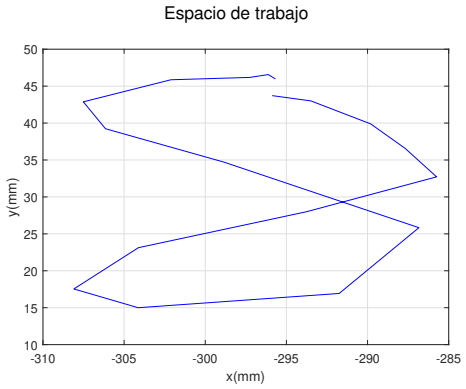


(a) ×10

(b) ×20



(c) ×30



(d) ×40

Figura 5.15: Cambio de velocidad

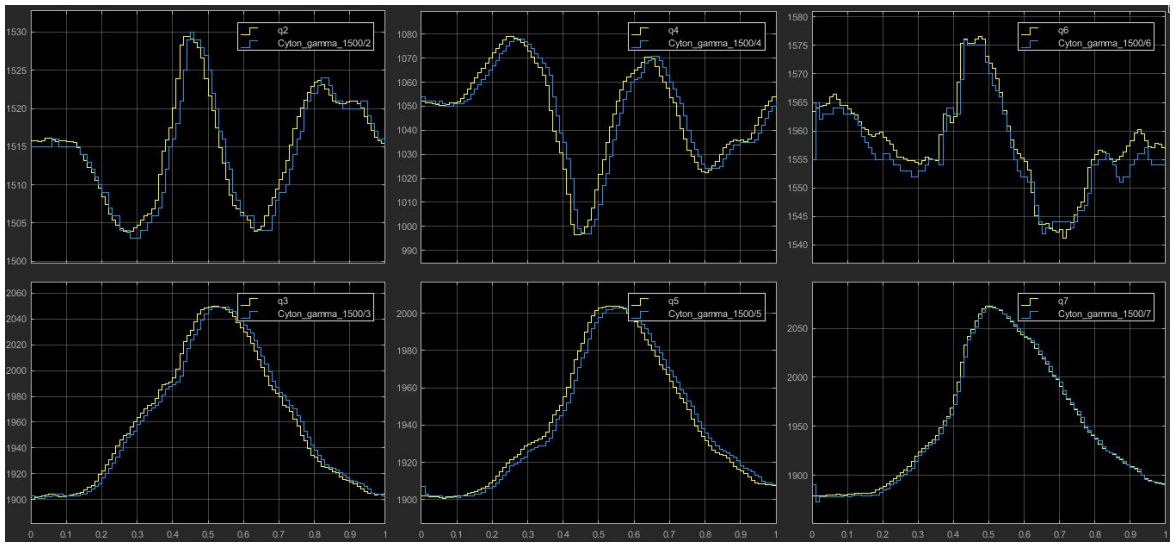


Figura 5.17: Seguimiento de las trayectorias con una aceleración aplicada de 6.66

5.2. Trayectoria libre en el espacio de trabajo muestreado

Para este caso de estudio se pretende entrenar a la red neuronal mediante ApD, de manera que la red neuronal sea capaz de mapear entre cualquier trayectoria deseada en el espacio de trabajo a los estados que reproduzcan el comportamiento natural humano de manera similar a las demostraciones.

Se estudian tres diferentes aprendizajes, donde se utiliza una única trayectoria de demostración del comportamiento humano. Cada trayectoria de demostración abarca un espacio de trabajo sobre un plano de aproximadamente de 6 cm × 8 cm.

5.2.1. Fase de aprendizaje de comportamiento

La primera demostración esta basada en líneas rectas paralelas espaciadas aproximadamente cada 5 mm que abarcan 2,167 puntos en el espacio de trabajo que se muestra en la figura 5.18.

Para la segunda demostración se utilizan círculos concéntricos, cuyas circunferencias están espaciadas 5 mm entre si, las cuales abarcan 3,300 puntos en el espacio de trabajo mostrada en la figura 5.20.

En la tercera demostración se utiliza una trayectoria más compleja, utilizando como base una función de tipo senoidal, esta trayectoria abarca 7,504 puntos en el espacio de trabajo que se muestra en la figura 5.22.

La red neuronal se entrena utilizando como parámetros de entrada la trayectoria de posición, la cual es obtenida mediante la cinemática directa de los estados de la

demostración, se muestra para la primera, segunda y tercera demostración en las figuras 5.19, 5.21 y 5.23 respectivamente.

Se inicia el entrenamiento de la red neuronal con 300,000 neuronas y se aumenta progresivamente hasta llegar a 1,200,000 neuronas para cada demostración.

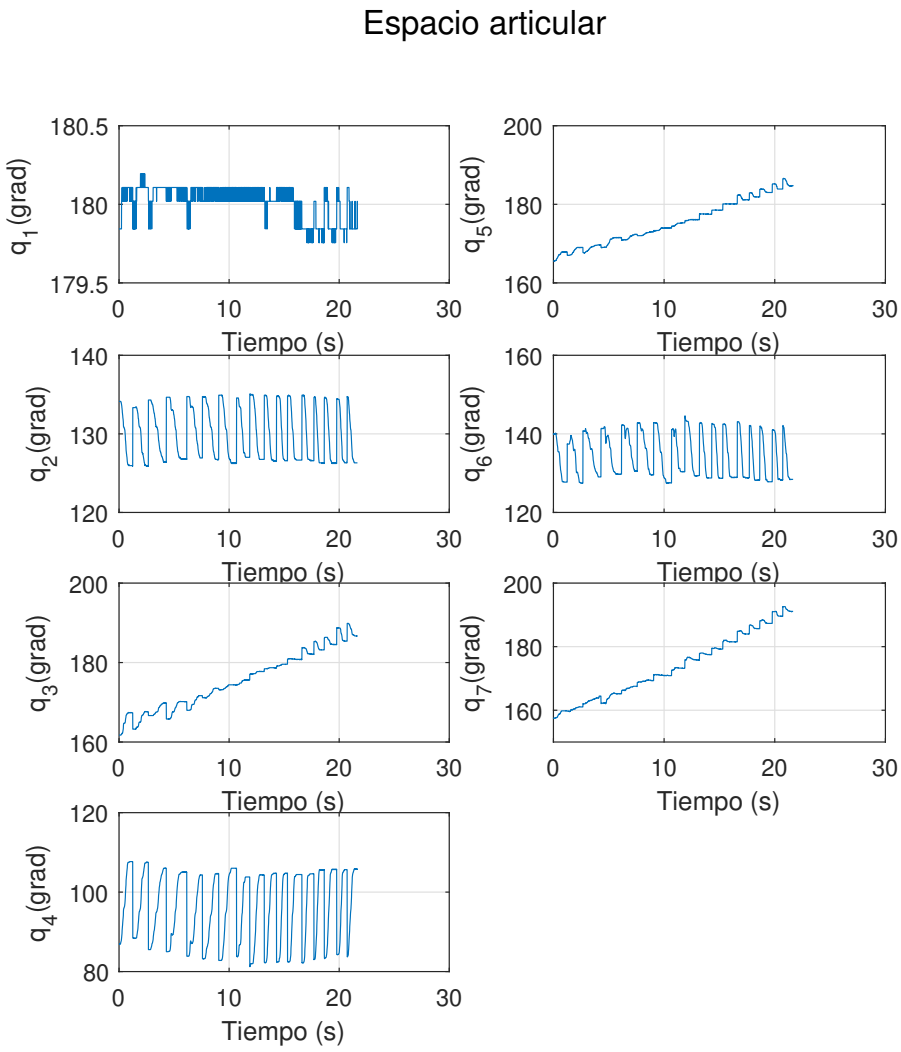


Figura 5.18: Estados de la demostración basada en líneas

5.2.2. Fase de generación de comportamiento

Para la primera demostración basada en líneas paralelas se observa un aprendizaje con un error muy grande en algunas partes de la trayectoria, mostrada en la figura 5.24a, con un error cuadrático medio de 3,705.89, a medida que se aumenta el número de neuronas a 900,000 se obtiene el mejor resultado con un error

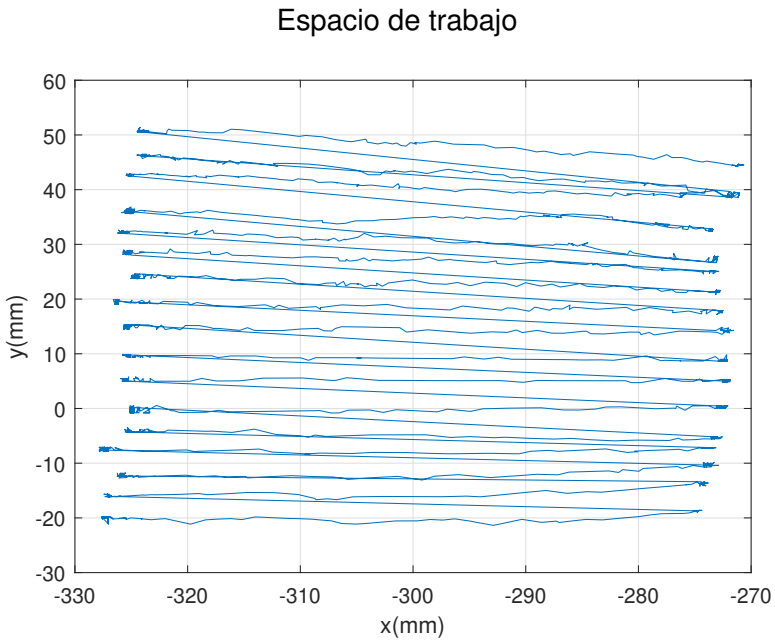


Figura 5.19: Cinemática directa de los estados de la demostración basada en líneas

cuadrático medio de 817.43, el cual tiene un error considerablemente grande en algunas partes de la trayectoria mostrado en la figura 5.24c. El uso excesivo de neuronas, como se ve en la figura 5.24d, donde al usar 1,200,000 causa un ligero aumento en el error de aprendizaje.

El error en el aprendizaje es debido a las articulaciones cuya trayectoria es más difícil de aprender para la red neuronal dada la complejidad de la trayectoria.

En el aprendizaje de la segunda demostración se observan una gran mejoría en los resultados, pues las trayectorias de los estados son mayormente más sencillas de aprender para la red neuronal, además de el número de puntos ayuda a disminuir el error de aprendizaje.

Al usar 300,000 neuronas se obtiene el mejor resultado con un error cuadrático medio de 13.05, al aumentar el número de neuronas para el aprendizaje el error cuadrático medio aumenta ligeramente. Sin embargo, se aprecia en la figura 5.25a existe una notable alteración en la trayectoria de la cinemática directa en donde se acumula la mayor parte del error cuadrático medio.

En esta demostración donde se utilizan un mayor número de puntos se obtiene una mejora significativa en el aprendizaje, pues el error cuadrático medio es muy pequeño, y se observa una trayectoria de la cinemática directa muy similar a la deseada.

Para el caso de aprendizaje con 300,000 neuronas, según la ecuación 4.31 en donde $V^{\dagger} \in \mathbf{R}^{N \times k}$ y N es el número de datos para cada entrada y k el número de

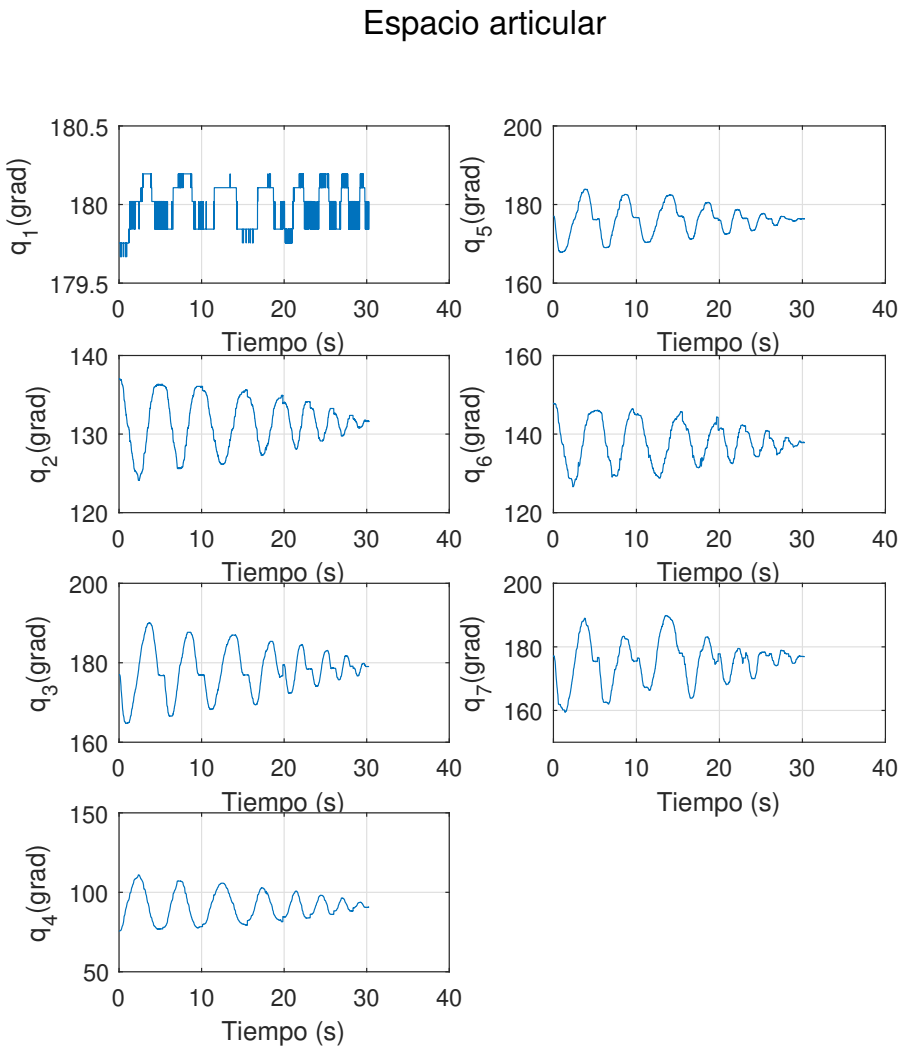


Figura 5.20: Estados de la demostración basada en círculos concéntricos

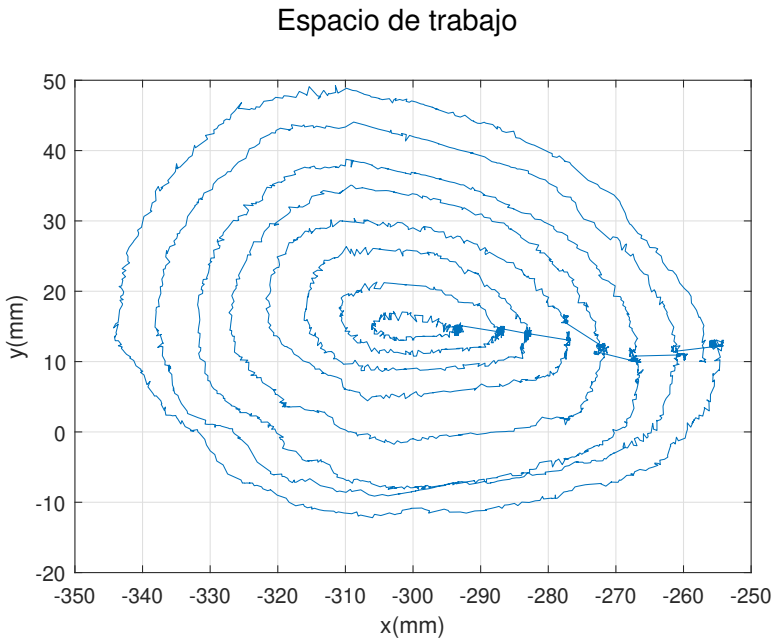


Figura 5.21: Cinemática directa de los estados de la demostración basada en círculos concéntricos

neuronas se tiene que tanto U y V y $V^{\dagger}$ son matrices con 375,200,000 elementos en donde cada elemento ocupa un espacio de 8 bytes en el entorno de matlab, por lo que el almacenamiento de cada una de estas matrices ocupa un espacio total de 3.0016 GB, para el caso de aprendizaje con 600,000 neuronas se requiere un espacio de almacenamiento de 6.0032 GB. Estas matrices son almacenadas usualmente en la memoria RAM de la computadora y dado que al momento de realizar los experimentos para obtener los resultados aquí mostrados no se contó con una la capacidad de 9 y 12 GB respectivamente para los casos de 900,000 y 1,200,000 neuronas.

Numero de neuronas	MSE en lineas	MSE en círculos	MSE func. senoidal
50,000	3,705.89	13.05	2.04
100,000	1,324.03	15.23	1.56
150,000	817.43	14.32	-
200,000	1,019.19	15.23	-

Cuadro 5.1: Error cuadrático medio de los aprendizajes con diferentes demostraciones y numero de neuronas

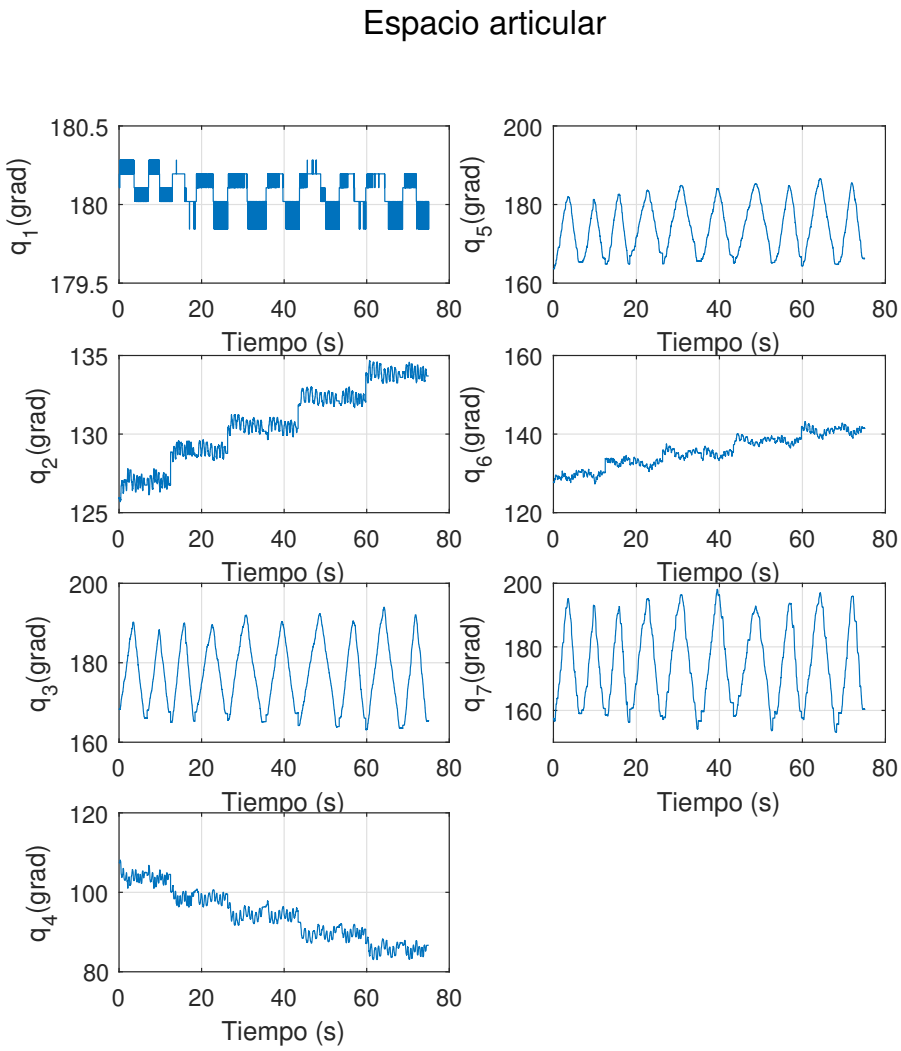


Figura 5.22: Estados de demostración basada en una función senoidal

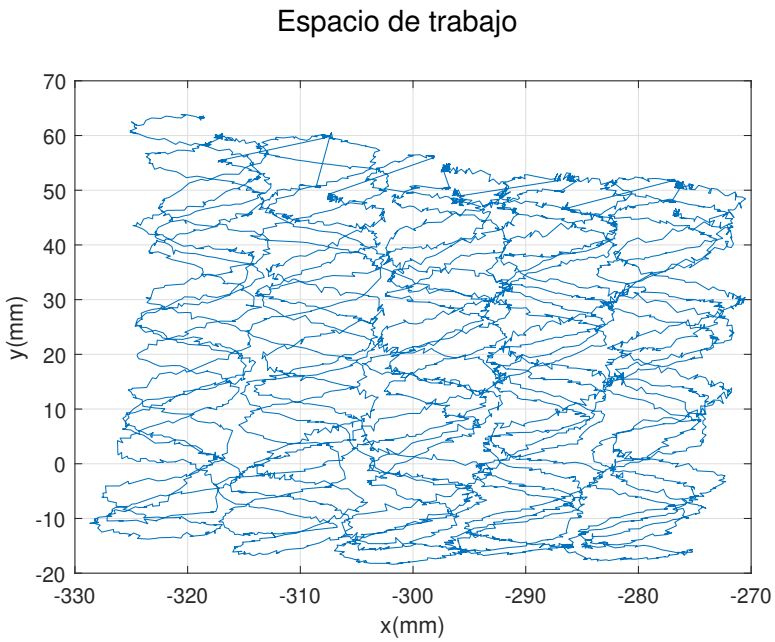
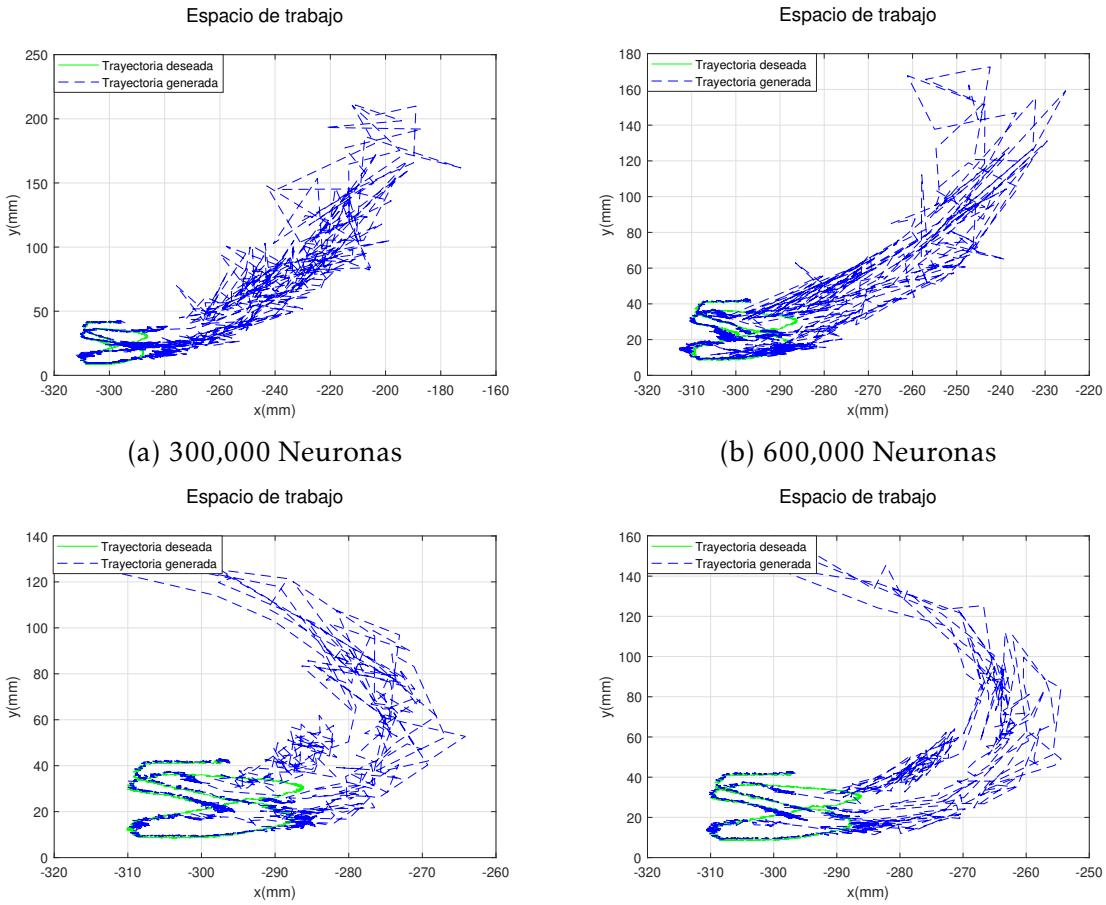


Figura 5.23: Cinemática directa de los estados de la demostración basada en una función senoidal



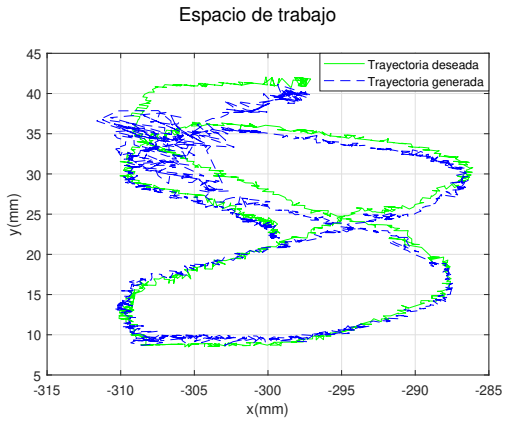
(a) 300,000 Neuronas

(b) 600,000 Neuronas

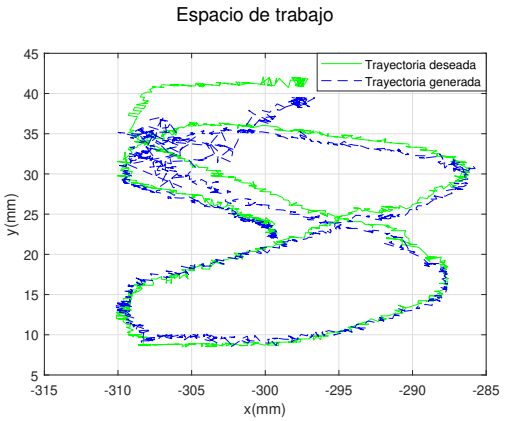
(c) 900,000 Neuronas

(d) 1,200,000 Neuronas

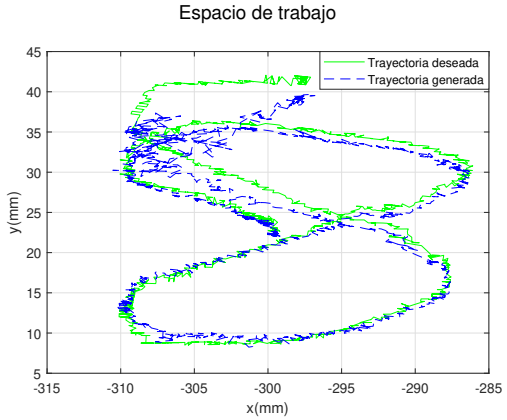
Figura 5.24: Cinemática directa de los estados del aprendizaje de la red neuronal utilizando una demostración basada en lineas paralelas



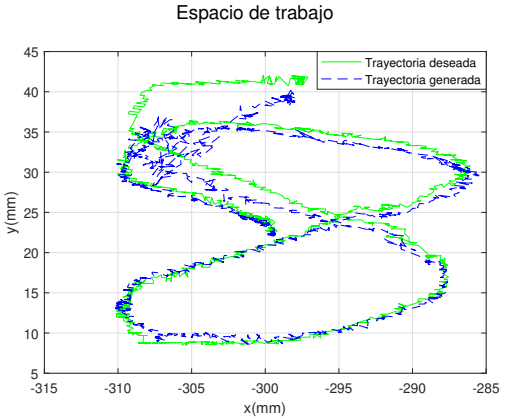
(a) 300,000 Neuronas



(b) 600,000 Neuronas

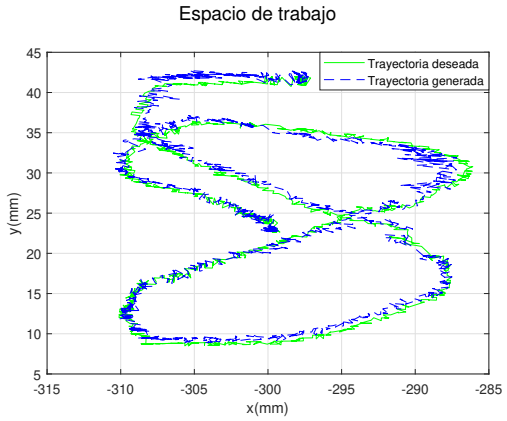


(c) 900,000 Neuronas

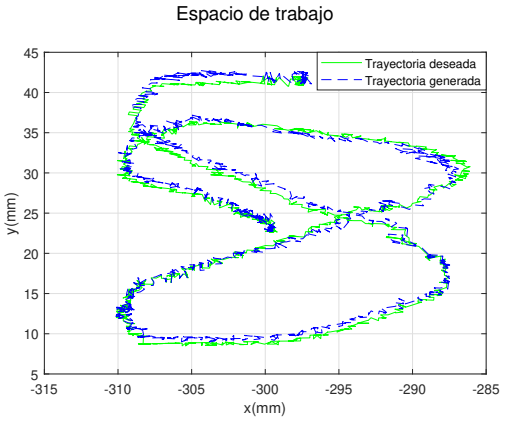


(d) 1,200,000 Neuronas

Figura 5.25: Cinemática directa de los estados del aprendizaje de la red neuronal utilizando una desmotaron basada en círculos concéntricos



(a) 300,000 Neuronas



(b) 600,000 Neuronas

Figura 5.26: Cinemática directa de los estados del aprendizaje de la red neuronal utilizando una demostración basada en una función senoidal

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

En este trabajo se ha presentado un método capaz de enseñar a un robot una tarea que reproduzca el comportamiento natural humano, mostrando de esta manera que el método mejora en velocidad y precisión el desempeño de un humano en una tarea enseñada mediante demostraciones. Dado que el aprendizaje se realiza en el espacio articular no es necesaria la realización de cinemática inversa del robot, siendo una ventaja.

De la misma manera se ha mostrado que el método es capaz de ensañar a un robot una serie de trayectorias que cubren un espacio de trabajo determinado y de esta manera reproducir cualquier tarea no enseñada con anterioridad dentro de este espacio de trabajo con un comportamiento humano, eficiente en velocidad y precisión.

El uso de algoritmos actuales que mejoran la velocidad de aprendizaje en las redes neuronales ha facilitado en gran medida el desarrollo de este método, de cualquier modo, el aprendizaje se puede ver seriamente afectado según la complejidad de la tarea a aprender, a mayor complejidad mayor será el número de neuronas necesarias para aprender la tarea.

6.2. Trabajo futuro

Como se vio en el capítulo anterior el espacio de trabajo muestreado se ve limitado por la longitud y complejidad de la trayectoria de demostración utilizada para el entrenamiento de la red neuronal. Esta limitación de tamaño en el espacio de trabajo podría verse incrementada drásticamente con el uso de técnicas de aprendizaje reforzado profundo.

Además, se plantea explorar las capacidades de operación en ambientes dinámicos y desconocidos, siendo que los robots cirujanos trabajan en circunstancias donde existen cuerpos en movimiento y sucesos inesperados en los que se esperarían que el robot pueda adquirir cierto grado de autonomía en la toma de decisiones.

Apéndice A

Codigos en Matlab

A.1. DTW para alineación de demostraciones

```
    %% Alineamiento de dos trayectorias con DTW
% Graficas
clearvars -except pos_1 pos_2 pos_3 q_1 q_2 q_3 t1 t2 t3
%seleccionar las 2 trayectorias a alinear
posx=pos_1;
posy=pos_2;
qx=q_1;
qy=q_2;
tx=t1;
ty=t2;
qxg=qx*0.0879;%conversion de pulsos a grados
qyg=qy*0.0879;
figure('Name','task_space')
plot(posx(:,1),posx(:,2),posy(:,1),posy(:,2))
ylabel('y(mm)')
xlabel('x(mm)')
grid
suptitle('Espacio de trabajo')
figure('Name','joint_space')
subplot(4,2,1)
plot(tx,qxg(:,1),ty,qyg(:,1))
xlabel('Tiempo (s)')
ylabel('q_1(grad)')
grid
subplot(4,2,3)
plot(tx,qxg(:,2),ty,qyg(:,2))
xlabel('Tiempo (s)')
```

```
ylabel('q_2(grad)')
grid
subplot(4,2,5)
plot(tx,qxg(:,3),ty,qyg(:,3))
xlabel('Tiempo (s)')
ylabel('q_3(grad)')
grid
subplot(4,2,7)
plot(tx,qxg(:,4),ty,qyg(:,4))
xlabel('Tiempo (s)')
ylabel('q_4(grad)')
grid
subplot(4,2,2)
plot(tx,qxg(:,5),ty,qyg(:,5))
xlabel('Tiempo (s)')
ylabel('q_5(grad)')
grid
subplot(4,2,4)
plot(tx,qxg(:,6),ty,qyg(:,6))
xlabel('Tiempo (s)')
ylabel('q_6(grad)')
grid
subplot(4,2,6)
plot(tx,qxg(:,7),ty,qyg(:,7))
xlabel('Tiempo (s)')
ylabel('q_7(grad)')
grid
suptitle('Espacio articular')
%% DTW
for vars=2:1:7
x=qxg(:,vars);
y=qyg(:,vars);
N=numel(x);
k1=1:N;
M=numel(y);
k2=1:M;
%———Trayectory 1 and 2
%%% Euclidean distance
for i=1:N
for j=1:M
d(i,j)=sqrt((x(i)-y(j))*(x(i)-y(j)));
end
end
```

```
% DTW, the new series are xi(k), yj(k)
xi(1)=1;xxj(1)=1;
j=0;
for k=1:N+M
j=j+1;
if xi(j)¡N && xxj(j)¡M
[a,I]=min([d(xi(j)+1,xxj(j)+1),d(xi(j)+1,xxj(j)),d(xi(j),xxj(j)+1)]);
end
if xi(j)==N && xxj(j)¡M
[a,I]=min([Inf,Inf,d(xi(j),xxj(j)+1)]);
end
if xxj(j)==M && xi(j)¡N
[a,I]=min([Inf,d(xi(j)+1,xxj(j)),Inf]);
end
if xi(j)==N && xxj(j)==M
I=0;
end
if I==0
j=j-1;
end
if I==1
xi(j+1)=xi(j)+1;
xxj(j+1)=xxj(j)+1;
end
if I==2
xi(j+1)=xi(j)+1;
xxj(j+1)=xxj(j);
end
if I==3
xi(j+1)=xi(j);
xxj(j+1)=xxj(j)+1;
end
end
wtray(vars)=[xi;xxj];
%wx=x(xi);
%wy=y(xxj);
%q_dtw(vars)=[wx,wy];
end
%% Experimento
w1=cell2mat(wtray(1));
w2=cell2mat(wtray(2));
w3=cell2mat(wtray(3));
w4=cell2mat(wtray(4));
```

```
w5=cell2mat(wtray(5));
w6=cell2mat(wtray(6));
w7=cell2mat(wtray(7));
fin= min([numel(w2(1,:)), numel(w3(1,:)), numel(w4(1,:)), numel( w5( 1, :)), numel(w6(1,:)), nu-
mel(w7(1,:))]);
xprom= round(( w2(1,1:fin) + w3(1,1:fin) + w4(1,1:fin) + w5(1,1:fin) + w6(1,1:fin) + w7(1,1:fin))/6);
xxprom= round(( w2(2,1:fin) + w3(2,1:fin) + w4(2,1:fin) + w5(2,1:fin) + w6(2,1:fin) + w7(2,1:fin))/6);
for vars=1:1:7
x=qxg(:,vars);
y=qyg(:,vars);
wx=x(xprom);
wy=y(xxprom);
q_dtw_1(:,vars)=wx;
q_dtw_2(:,vars)=wy;
end
tz=0:0.01:(numel(wx)-1)*0.01;
%% DTW graphs
figure('Name','DTW')
subplot(4,2,1)
plot(tz,q_dtw_1(:,1),tz,q_dtw_2(:,1))
xlabel('Tiempo (s)')
ylabel('q_1(grad)')
grid
subplot(4,2,3)
plot(tz,q_dtw_1(:,2),tz,q_dtw_2(:,2))
xlabel('Tiempo (s)')
ylabel('q_2(grad)')
grid
subplot(4,2,5)
plot(tz,q_dtw_1(:,3),tz,q_dtw_2(:,3))
xlabel('Tiempo (s)')
ylabel('q_3(grad)')
grid
subplot(4,2,7)
plot(tz,q_dtw_1(:,4),tz,q_dtw_2(:,4))
xlabel('Tiempo (s)')
ylabel('q_4(grad)')
grid
subplot(4,2,2)
plot(tz,q_dtw_1(:,5),tz,q_dtw_2(:,5))
xlabel('Tiempo (s)')
ylabel('q_5(grad)')
grid
```

```
subplot(4,2,4)
plot(tz,q_dtw_1(:,6),tz,q_dtw_2(:,6))
xlabel('Tiempo (s)')
ylabel('q_6(grad)')
grid
subplot(4,2,6)
plot(tz,q_dtw_1(:,7),tz,q_dtw_2(:,7))
xlabel('Tiempo (s)')
ylabel('q_7(grad)')
grid
suptitle('Espacio articular')
%Direct cinematics
for j=1:2
if j==1
q1=q_dtw_1(:,1);
q2=q_dtw_1(:,2);
q3=q_dtw_1(:,3);
q4=q_dtw_1(:,4);
q5=q_dtw_1(:,5);
q6=q_dtw_1(:,6);
q7=q_dtw_1(:,7);
end
if j==2
q1=q_dtw_2(:,1);
q2=q_dtw_2(:,2);
q3=q_dtw_2(:,3);
q4=q_dtw_2(:,4);
q5=q_dtw_2(:,5);
q6=q_dtw_2(:,6);
q7=q_dtw_2(:,7);
end
for n_d=1:1:numel(q1)
radi=pi/180;
o1=(q1(n_d)*radi)+pi;
o2=(q2(n_d)*radi)+pi-pi/2;
o3=(q3(n_d)*radi)+pi;
o4=(q4(n_d)*radi)+pi;
o5=(q5(n_d)*radi)+pi;
o6=(q6(n_d)*radi)+pi+pi/2;
o7=(q7(n_d)*radi)+pi;
a0=74;
a1=103.35;
a6=172.52;
```

```
a7=32;
d1=a0+a1;
a2=125.83;
a3=115.38;
a4=97.52;
a5=71.64;
d7=a6+a7;
A1=[cos(o1) 0 -sin(o1) 0; sin(o1) 0 cos(o1) 0; 0 -1 0 d1; 0 0 0 1];
A2=[cos(o2) 0 -sin(o2) a2*cos(o2); sin(o2) 0 cos(o2) a2*sin(o2); 0 -1 0 0; 0 0 0 1];
A3=[cos(o3) 0 sin(o3) a3*cos(o3); sin(o3) 0 -cos(o3) a3*sin(o3); 0 1 0 0; 0 0 0 1];
A4=[cos(o4) 0 -sin(o4) a4*cos(o4); sin(o4) 0 cos(o4) a4*sin(o4); 0 -1 0 0; 0 0 0 1];
A5=[cos(o5) 0 sin(o5) a5*cos(o5); sin(o5) 0 -cos(o5) a5*sin(o5); 0 1 0 0; 0 0 0 1];
A6=[cos(o6) 0 sin(o6) 0; sin(o6) 0 -cos(o6) 0; 0 1 0 0; 0 0 0 1];
A7=[cos(o7) -sin(o7) 0 0;sin(o7) cos(o7) 0 0; 0 0 1 d7;0 0 0 1];
T6=A1*A2*A3*A4*A5*A6*A7;
if j==1
x_dtw_1(n_d)=T6(1,4);
y_dtw_1(n_d)=T6(2,4);
z_dtw_1(n_d)=T6(3,4);
end
if j==2
x_dtw_2(n_d)=T6(1,4);
y_dtw_2(n_d)=T6(2,4);
z_dtw_2(n_d)=T6(3,4);
end
end
end
figure('Name','FK_DTW')
plot(x_dtw_1,y_dtw_1,x_dtw_2,y_dtw_2)
ylabel('y(mm)')
xlabel('x(mm)')
grid
suptitle('Espacio de trabajo')
%% Promedio
q_dtw_p=(q_dtw_1+q_dtw_2)*(1/2);
for n_d=1:1:numel(q1)
radi=pi/180;
o1=(q_dtw_p(n_d,1)*radi)+pi;
o2=(q_dtw_p(n_d,2)*radi)+pi-pi/2;
o3=(q_dtw_p(n_d,3)*radi)+pi;
o4=(q_dtw_p(n_d,4)*radi)+pi;
o5=(q_dtw_p(n_d,5)*radi)+pi;
o6=(q_dtw_p(n_d,6)*radi)+pi+pi/2;
```

```
o7=(q_dtw_p(n_d,7)*radi)+pi;
a0=74;
a1=103.35;
a6=172.52;
a7=32;
d1=a0+a1;
a2=125.83;
a3=115.38;
a4=97.52;
a5=71.64;
d7=a6+a7;
A1=[cos(o1) 0 -sin(o1) 0; sin(o1) 0 cos(o1) 0; 0 -1 0 d1; 0 0 0 1];
A2=[cos(o2) 0 -sin(o2) a2*cos(o2); sin(o2) 0 cos(o2) a2*sin(o2); 0 -1 0 0; 0 0 0 1];
A3=[cos(o3) 0 sin(o3) a3*cos(o3); sin(o3) 0 -cos(o3) a3*sin(o3); 0 1 0 0; 0 0 0 1];
A4=[cos(o4) 0 -sin(o4) a4*cos(o4); sin(o4) 0 cos(o4) a4*sin(o4); 0 -1 0 0; 0 0 0 1];
A5=[cos(o5) 0 sin(o5) a5*cos(o5); sin(o5) 0 -cos(o5) a5*sin(o5); 0 1 0 0; 0 0 0 1];
A6=[cos(o6) 0 sin(o6) 0; sin(o6) 0 -cos(o6) 0; 0 1 0 0; 0 0 0 1];
A7=[cos(o7) -sin(o7) 0 0;sin(o7) cos(o7) 0 0; 0 0 1 d7;0 0 0 1];
T6=A1*A2*A3*A4*A5*A6*A7;
if j==1
x_dtw_1(n_d)=T6(1,4);
y_dtw_1(n_d)=T6(2,4);
z_dtw_1(n_d)=T6(3,4);
end
end
pos_dtw = [ ( x_dtw_1 + x_dtw_2 ) ', ( y_dtw_1 + y_dtw_2 )', (z_dtw_1 + z_dtw_2) ' ] * (1/2);
```

A.2. Red neuronal para el aprendizaje de comportamiento humano

```
    clearvars -except pos_1 pos_2 pos_3 q_1 q_2 q_3 t1 t2 t3
pos_dtw
q_dtw_p tz
clc
%%% Neuronas
L=9;%neuronas de la capa de entrada
N=700;%neuronas de la capa oculta
%%% Entrada (promedio si es necesario)
%pos_x=pos_1(:,1);
pos_x=pos_dtw(:,1);
%pos_y=pos_1(:,2);
```

```
pos_y=pos_dtw(:,2);
%pos_z=pos_1(:,3);
pos_z=pos_dtw(:,3);
%%% Target trayectories (promedio si es necesario)
%Targ=q_1
%tTarg=t1
Targ=q_dtw_p;
tTarg=tz;
%normalizacion acorde al espacio de trabajo
in_1=(pos_x+435)/(-220+435);
in_2=(pos_y+100)/(130+100);
in_3=(pos_z+2)/(25+2);
for k=2:1:7
clearvars -except pos_1 q_1 t1 pos_dtw q_dtw_p tz tTarg in_1 in_2
in_3 pos_x pos_y pos_z Targ k L N n_train W B q_neu
Num=numel(Targ(:,k));
M1=Num-3;
T_aux=Targ(1:M1,k)';
for t_aux=1:M1
II(1,t_aux)=in_1(t_aux);
II(2,t_aux)=in_1(t_aux+1);
II(3,t_aux)=in_1(t_aux+2);
II(4,t_aux)=in_2(t_aux);
II(5,t_aux)=in_2(t_aux+1);
II(6,t_aux)=in_2(t_aux+2);
II(7,t_aux)=in_3(t_aux);
II(8,t_aux)=in_3(t_aux+1);
II(9,t_aux)=in_3(t_aux+2);
end
%%% Weight matrix for hidden layer
W(k)=rand(L,N);
W_aux=cell2mat(W(k));
for t_aux=1:M1
for j=1:N
Hk(j,t_aux)=0;%%%% input to each hidden node
for jk=1:L
Hk(j,t_aux)=Hk(j,t_aux)+W_aux(jk,j)*II(jk,t_aux);
end
Hk(j,t_aux) = (exp(Hk(j,t_aux)) - exp(-Hk(j,t_aux))) /
(exp(Hk(j,t_aux)) + exp(-Hk(j,t_aux)));%tanh
end
end
%%% Weights in uotput layer, Moore-Penrose pseudoinverse training
```

```
Bk=T_aux*pinv(Hk);
%Bk=T_aux*Hk'*inv((eye(N)/99999)+Hk*Hk');
B(k)=Bk;
%%%%% Output layer
q_neu(:,k)=[Hk'*Bk'];
end
q_neu(:,1)=ones(M1,1)*180;
t_neu=0:0.01:(M1-1)*0.01;
%% Graphs
figure('Name','Neural Network_1')
subplot(4,2,1)
plot(tTarg,Targ(:,1),'g')
hold on
plot(t_neu,q_neu(:,1),'b--')
hold off
xlabel('Tiempo (s)')
ylabel('q_1(grad)')
grid
subplot(4,2,3)
plot(tTarg,Targ(:,2),'g')
hold on
plot(t_neu,q_neu(:,2),'b--')
hold off
xlabel('Tiempo (s)')
ylabel('q_2(grad)')
grid
subplot(4,2,5)
plot(tTarg,Targ(:,3),'g')
hold on
plot(t_neu,q_neu(:,3),'b--')
hold off
xlabel('Tiempo (s)')
ylabel('q_3(grad)')
grid
subplot(4,2,7)
plot(tTarg,Targ(:,4),'g')
hold on
plot(t_neu,q_neu(:,4),'b--')
hold off
xlabel('Tiempo (s)')
ylabel('q_4(grad)')
grid
subplot(4,2,2)
```

```
plot(tTarg,Targ(:,5),'g')
hold on
plot(t_neu,q_neu(:,5),'b–')
hold off
xlabel('Tiempo (s)')
ylabel('q_5(grad)')
grid
subplot(4,2,4)
plot(tTarg,Targ(:,6),'g')
hold on
plot(t_neu,q_neu(:,6),'b–')
hold off
xlabel('Tiempo (s)')
ylabel('q_6(grad)')
grid
subplot(4,2,6)
plot(tTarg,Targ(:,7),'g')
hold on
plot(t_neu,q_neu(:,7),'b–')
hold off
xlabel('Tiempo (s)')
ylabel('q_7(grad)')
grid
suptitle('Espacio articular')
%Direct cinematics
for n_d=1:1:(numel(q_neu(:,2)))
radi=pi/180;
o1=(q_neu(n_d,1)*radi)+pi;
o2=(q_neu(n_d,2)*radi)+pi-pi/2;
o3=(q_neu(n_d,3)*radi)+pi;
o4=(q_neu(n_d,4)*radi)+pi;
o5=(q_neu(n_d,5)*radi)+pi;
o6=(q_neu(n_d,6)*radi)+pi+pi/2;
o7=(q_neu(n_d,7)*radi)+pi;
a0=74;
a1=103.35;
a6=172.52;
a7=32;
d1=a0+a1;
a2=125.83;
a3=115.38;
a4=97.52;
a5=71.64;
```

```
d7=a6+a7;
A1=[cos(o1) 0 -sin(o1) 0; sin(o1) 0 cos(o1) 0; 0 -1 0 d1; 0 0 0 1];
A2=[cos(o2) 0 -sin(o2) a2*cos(o2); sin(o2) 0 cos(o2) a2*sin(o2); 0 -1 0 0; 0 0 0 1];
A3=[cos(o3) 0 sin(o3) a3*cos(o3); sin(o3) 0 -cos(o3) a3*sin(o3); 0 1 0 0; 0 0 0 1];
A4=[cos(o4) 0 -sin(o4) a4*cos(o4); sin(o4) 0 cos(o4) a4*sin(o4); 0 -1 0 0; 0 0 0 1];
A5=[cos(o5) 0 sin(o5) a5*cos(o5); sin(o5) 0 -cos(o5) a5*sin(o5); 0 1 0 0; 0 0 0 1];
A6=[cos(o6) 0 sin(o6) 0; sin(o6) 0 -cos(o6) 0; 0 1 0 0; 0 0 0 1];
A7=[cos(o7) -sin(o7) 0 0;sin(o7) cos(o7) 0 0; 0 0 1 d7;0 0 0 1];
T6=A1*A2*A3*A4*A5*A6*A7;
x_neu(n_d)=T6(1,4);
y_neu(n_d)=T6(2,4);
z_neu(n_d)=T6(3,4);
end
figure('Name','ELM_taskspace')
plot(pos_x,pos_y,'g')
hold on
plot(x_neu,y_neu,'b–')
hold off
ylabel('y(mm)')
xlabel('x(mm)')
grid
legend('Trayectoria','Trayectoria aprendida','Location','southwest')
suptitle('Espacio de trabajo')
```

A.3. Red neuronal para la generación de comportamiento humano

```
    % Testing
% run for deleting all variables except the weights
clearvars -except W B Hd pos_1
%then delte pos_x_1 pos_y_1 pos_z_1 and q1_1
%load new in and run all
%%
% Inputs
pos_x=pos_1(:,1);%pos_9(30:350,1);
pos_y=pos_1(:,2);%pos_9(30:350,2);
pos_z=pos_1(:,3);%pos_9(30:350,3);
L=9;%input number to NN
Num=numel(pos_x);
M1=Num-3;% Testing data number
%normalizacion acorde al espacio de trabajo
in_1=(pos_x+435)/(-220+435);
```

```
in_2=(pos_y+100)/(130+100);
in_3=(pos_z+2)/(25+2);
for t_neu=1:M1
II(1,t_neu)=in_1(t_neu);
II(2,t_neu)=in_1(t_neu+1);
II(3,t_neu)=in_1(t_neu+2);
II(4,t_neu)=in_2(t_neu);
II(5,t_neu)=in_2(t_neu+1);
II(6,t_neu)=in_2(t_neu+2);
II(7,t_neu)=in_3(t_neu);
II(8,t_neu)=in_3(t_neu+1);
II(9,t_neu)=in_3(t_neu+2);
end
for k=2:1:7
W_aux=cell2mat(W(k));
B_aux=cell2mat(B(k));
for t_neu=1:M1
I=W_aux'*II(:,t_neu);
O=(exp(I)-exp(-I))./(exp(I)+exp(-I));%tanh
yy(t_neu)=B_aux*O;
end
q_neu(:,k)=yy';
end
M1=numel(yy);
q_neu = q_neu * (360/4096);%%%%% agregado porpesos
minispace_sin_1 esta en pulsos
q_neu(:,1)=ones(M1,1)*180;
% Graphs
t_neu=0:0.01:(M1-1)*0.01;
figure('Name','Neural Network Testing_1')
subplot(4,2,1)
plot(t_neu,q_neu(:,1))
xlabel('Tiempo (s)')
ylabel('q_1(grad)')
grid
subplot(4,2,3)
plot(t_neu,q_neu(:,2))
xlabel('Tiempo (s)')
ylabel('q_2(grad)')
grid
subplot(4,2,5)
plot(t_neu,q_neu(:,3))
xlabel('Tiempo (s)')
```

```
ylabel('q_3(grad)')
grid
subplot(4,2,7)
plot(t_neu,q_neu(:,4))
xlabel('Tiempo (s)')
ylabel('q_4(grad)')
grid
subplot(4,2,2)
plot(t_neu,q_neu(:,5))
xlabel('Tiempo (s)')
ylabel('q_5(grad)')
grid
subplot(4,2,4)
plot(t_neu,q_neu(:,6))
xlabel('Tiempo (s)')
ylabel('q_6(grad)')
grid
subplot(4,2,6)
plot(t_neu,q_neu(:,7))
xlabel('Tiempo (s)')
ylabel('q_7(grad)')
grid
suptitle('Espacio de tiempo')
% Direct cinematics
for n_d=1:1:(numel(q_neu(:,2)))
radi=pi/180;
o1=(q_neu(n_d,1)*radi)+pi;
o2=(q_neu(n_d,2)*radi)+pi-pi/2;
o3=(q_neu(n_d,3)*radi)+pi;
o4=(q_neu(n_d,4)*radi)+pi;
o5=(q_neu(n_d,5)*radi)+pi;
o6=(q_neu(n_d,6)*radi)+pi+pi/2;
o7=(q_neu(n_d,7)*radi)+pi;
a0=74;
a1=103.35;
a6=172.52;
a7=32;
d1=a0+a1;
a2=125.83;
a3=115.38;
a4=97.52;
a5=71.64;
d7=a6+a7;
```

```
A1=[cos(o1) 0 -sin(o1) 0; sin(o1) 0 cos(o1) 0; 0 -1 0 d1; 0 0 0 1];
A2=[cos(o2) 0 -sin(o2) a2*cos(o2); sin(o2) 0 cos(o2) a2*sin(o2); 0 -1 0 0; 0 0 0 1];
A3=[cos(o3) 0 sin(o3) a3*cos(o3); sin(o3) 0 -cos(o3) a3*sin(o3); 0 1 0 0; 0 0 0 1];
A4=[cos(o4) 0 -sin(o4) a4*cos(o4); sin(o4) 0 cos(o4) a4*sin(o4); 0 -1 0 0; 0 0 0 1];
A5=[cos(o5) 0 sin(o5) a5*cos(o5); sin(o5) 0 -cos(o5) a5*sin(o5); 0 1 0 0; 0 0 0 1];
A6=[cos(o6) 0 sin(o6) 0; sin(o6) 0 -cos(o6) 0; 0 1 0 0; 0 0 0 1];
A7=[cos(o7) -sin(o7) 0 0;sin(o7) cos(o7) 0 0; 0 0 1 d7;0 0 0 1];
T6=A1*A2*A3*A4*A5*A6*A7;
x_neu(n_d)=T6(1,4);
y_neu(n_d)=T6(2,4);
z_neu(n_d)=T6(3,4);
end
%%Error cuadratico medio
Num_neu=numel(x_neu);
Error=0;
for er=1:1:Num_neu
Error = Error + (x_neu(er) - pos_x(er))^2 + (y_neu(er) - pos_y(er))^2 + (z_neu(er) - pos_z(er))^2;
end
MSE=(1/Num_neu*3)*Error;
figure('Name','ELMtesting_taskspace')
plot(pos_x,pos_y,'g')
hold on
plot(x_neu,y_neu,'b–')
hold off
ylabel('y(mm)')
xlabel('x(mm)')
grid
legend( 'Trayectoria deseada' , 'Trayectoria generada'  , 'Location' , 'southwest' )
suptitle('Espacio de trabajo')
```

A.4. Filtrado de señales articulares

```
    %% Filtra la salida de la red neuronal q_neu y la compara con trayectoria tar
%%cargar el filtro Hd
%trayectoriade comparacion
%pos_x=pos_1(:,1);
pos_x=pos_dtw(:,1);
%pos_y=pos_1(:,2);
pos_y=pos_dtw(:,2);
for k=1:1:7
q_neu_aux=q_neu(:,k);
q_neu_aux2=(q_neu_aux/q_neu_aux(1))-1;
```

```
q_neu_aux3=filter(Hd,q_neu_aux2);
q_filt(:,k)=(q_neu_aux3+1)*(q_neu_aux(1));
end
figure('Name','Neural Filter_1')
subplot(4,2,1)
plot(t_neu,q_filt(:,1))
xlabel('Tiempo (s)')
ylabel('q_1(grad)')
grid
subplot(4,2,3)
plot(t_neu,q_filt(:,2))
xlabel('Tiempo (s)')
ylabel('q_2(grad)')
grid
subplot(4,2,5)
plot(t_neu,q_filt(:,3))
xlabel('Tiempo (s)')
ylabel('q_3(grad)')
grid
subplot(4,2,7)
plot(t_neu,q_filt(:,4))
xlabel('Tiempo (s)')
ylabel('q_4(grad)')
grid
subplot(4,2,2)
plot(t_neu,q_filt(:,5))
xlabel('Tiempo (s)')
ylabel('q_5(grad)')
grid
subplot(4,2,4)
plot(t_neu,q_filt(:,6))
xlabel('Tiempo (s)')
ylabel('q_6(grad)')
grid
subplot(4,2,6)
plot(t_neu,q_filt(:,7))
xlabel('Tiempo (s)')
ylabel('q_7(grad)')
grid
suptitle('Espacio articular')
%Direct cinematics
for n_d=1:1:(numel(q_filt(:,2)))
radi=pi/180;
```

```
o1=(q_filt(n_d,1)*radi)+pi;
o2=(q_filt(n_d,2)*radi)+pi-pi/2;
o3=(q_filt(n_d,3)*radi)+pi;
o4=(q_filt(n_d,4)*radi)+pi;
o5=(q_filt(n_d,5)*radi)+pi;
o6=(q_filt(n_d,6)*radi)+pi+pi/2;
o7=(q_filt(n_d,7)*radi)+pi;
a0=74;
a1=103.35;
a6=172.52;
a7=32;
d1=a0+a1;
a2=125.83;
a3=115.38;
a4=97.52;
a5=71.64;
d7=a6+a7;
A1=[cos(o1) 0 -sin(o1) 0; sin(o1) 0 cos(o1) 0; 0 -1 0 d1; 0 0 0 1];
A2=[cos(o2) 0 -sin(o2) a2*cos(o2); sin(o2) 0 cos(o2) a2*sin(o2); 0 -1 0 0; 0 0 0 1];
A3=[cos(o3) 0 sin(o3) a3*cos(o3); sin(o3) 0 -cos(o3) a3*sin(o3); 0 1 0 0; 0 0 0 1];
A4=[cos(o4) 0 -sin(o4) a4*cos(o4); sin(o4) 0 cos(o4) a4*sin(o4); 0 -1 0 0; 0 0 0 1];
A5=[cos(o5) 0 sin(o5) a5*cos(o5); sin(o5) 0 -cos(o5) a5*sin(o5); 0 1 0 0; 0 0 0 1];
A6=[cos(o6) 0 sin(o6) 0; sin(o6) 0 -cos(o6) 0; 0 1 0 0; 0 0 0 1];
A7=[cos(o7) -sin(o7) 0 0;sin(o7) cos(o7) 0 0; 0 0 1 d7;0 0 0 1];
T0=A1;
T1=A1*A2;
T2=A1*A2*A3;
T3=A1*A2*A3*A4;
T4=A1*A2*A3*A4*A5;
T5=A1*A2*A3*A4*A5*A6;
T6=A1*A2*A3*A4*A5*A6*A7;
x_filt(n_d)=T6(1,4);
y_filt(n_d)=T6(2,4);
z_filt(n_d)=T6(3,4);
end
figure('Name','FILT_taskspace')
plot(pos_x,pos_y,'g')
hold on
plot(x_filt,y_filt,'b–')
hold off
ylabel('y(mm)')
xlabel('x(mm)')
grid
```

```
legend('Trayectoria de referencia','Trayectoria filtrada')
suptitle('Espacio de trabajo')
clearvars -except W B Hd q_dtw_p pos_dtw tz q_filt q_neu t_neu pos_1 pos_2 pos_3 q_1 q_2 q_3 t1 t2 t3
```

A.5. Cambio de velocidad de demostraciones

```
    clearvars -except myDxl Ts W B Hd q_dtw_p pos_dtw tz q_filt q_neu t_neu pos_1 pos_2 pos_3 q_1
q_2 q_3 t1 t2 t3
%it seems to be good enoght to have at least 26 (0.26 seconds) data points to have a good
%recovered figure
%velocity
vel=0.025;
T_s=0.01;%time sample
%trayectoriade comparacion
%pos_x=pos_1(:,1);
pos_x=pos_dtw(:,1);
%pos_y=pos_1(:,2);
pos_y=pos_dtw(:,2);
N=numel(q_filt(:,2));
M=N*vel;
pasyj=N/M;
%DTW with zeros as needed
k=1;
q_vel=ones(round(M),7);
q_vel(1,:)=q_filt(1,:);
for j=2:1:round(M)
k=k+pasyj;
kj=round(k);
q_vel(j,:)=q_filt(kj,:);
end
t_vel=0:T_s:T_s*(numel(q_vel(:,2))-1);
%% Graphs
figure('Name','velocity change')
subplot(4,2,1)
plot(t_vel,q_vel(:,1))
xlabel('Tiempo (s)')
ylabel('q_1(grad)')
grid
subplot(4,2,3)
plot(t_vel,q_vel(:,2))
xlabel('Tiempo (s)')
ylabel('q_2(grad)')
```

```
grid
subplot(4,2,5)
plot(t_vel,q_vel(:,3))
xlabel('Tiempo (s)')
ylabel('q_3(grad)')
grid
subplot(4,2,7)
plot(t_vel,q_vel(:,4))
xlabel('Tiempo (s)')
ylabel('q_4(grad)')
grid
subplot(4,2,2)
plot(t_vel,q_vel(:,5))
xlabel('Tiempo (s)')
ylabel('q_5(grad)')
grid
subplot(4,2,4)
plot(t_vel,q_vel(:,6))
xlabel('Tiempo (s)')
ylabel('q_6(grad)')
grid
subplot(4,2,6)
plot(t_vel,q_vel(:,7))
xlabel('Tiempo (s)')
ylabel('q_7(grad)')
grid
suptitle('Espacio articular')
%Direct cinematics
for n_d=1:1:(numel(q_vel(:,2)))
radi=pi/180;
o1=(q_vel(n_d,1)*radi)+pi;
o2=(q_vel(n_d,2)*radi)+pi-pi/2;
o3=(q_vel(n_d,3)*radi)+pi;
o4=(q_vel(n_d,4)*radi)+pi;
o5=(q_vel(n_d,5)*radi)+pi;
o6=(q_vel(n_d,6)*radi)+pi+pi/2;
o7=(q_vel(n_d,7)*radi)+pi;
a0=74;
a1=103.35;
a6=172.52;
a7=32;
d1=a0+a1;
a2=125.83;
```

```
a3=115.38;
a4=97.52;
a5=71.64;
d7=a6+a7;
A1=[cos(o1) 0 -sin(o1) 0; sin(o1) 0 cos(o1) 0; 0 -1 0 d1; 0 0 0 1];
A2=[cos(o2) 0 -sin(o2) a2*cos(o2); sin(o2) 0 cos(o2) a2*sin(o2); 0 -1 0 0; 0 0 0 1];
A3=[cos(o3) 0 sin(o3) a3*cos(o3); sin(o3) 0 -cos(o3) a3*sin(o3); 0 1 0 0; 0 0 0 1];
A4=[cos(o4) 0 -sin(o4) a4*cos(o4); sin(o4) 0 cos(o4) a4*sin(o4); 0 -1 0 0; 0 0 0 1];
A5=[cos(o5) 0 sin(o5) a5*cos(o5); sin(o5) 0 -cos(o5) a5*sin(o5); 0 1 0 0; 0 0 0 1];
A6=[cos(o6) 0 sin(o6) 0; sin(o6) 0 -cos(o6) 0; 0 1 0 0; 0 0 0 1];
A7=[cos(o7) -sin(o7) 0 0;sin(o7) cos(o7) 0 0; 0 0 1 d7;0 0 0 1];
T6=A1*A2*A3*A4*A5*A6*A7;
x_vel(n_d)=T6(1,4);
y_vel(n_d)=T6(2,4);
z_vel(n_d)=T6(3,4);
end
figure('Name','velocity change_tray')
%plot(pos_x,pos_y,'g')
%hold on
plot(x_vel,y_vel,'b')
%hold off
ylabel('y(mm)')
xlabel('x(mm)')
grid
%legend('Trayectoria','Trayectoria a diferente velocidad')
suptitle('Espacio de trabajo')
```

Bibliografía

[1] A. Murali, S. Sen, B. Kehoe, A. Garg, S. McFarland, S. Patil, W. D. Boyd, S. Lim, P. Abbeel, and K. Goldberg, "Learning by observation for surgical subtasks: multilateral cutting of 3d viscoelastic and 2d orthotropic tissue phantoms," *IEEE International Conference on Robotics and Automation, pp. 618-621*, 2000.

[2] H. Mayer, F. Gomez, D. Wierstra, I. Nagy, A. Knoll, and J. Schmidhuber, "A system for robotic heart surgery thatlearns to tie knots using recurrent neural networks," *InIntelligent Robots and Systems (IROS), 2006 IEEE/RSJInternational Conference on, pages 543–548*, 2006.

[3] H. Mayer, I. Nagy, A. Knoll, E. Braun, R. Lange, and R. Bauernschmitt, "Adptative control for human-robot skilltransfer: trajectory planning based on fluid dynamics," *InRobotics and Automation (ICRA), 2007IEEE International Conference on, pages 1800–1807*, 2007.

[4] H. Mayer, I. Nagy, D. Burschka, A. Knoll, E. Braun, R. Lange, and R. Bauernschmitt, "Automation of manual tasks for minimally invasive surgery," *Fourth International Conference on, pages 260–265*, 2008.

[5] G. Guthart and J. Salisbury, "The intuitive/sup tm/ telesurgery system: overview and application," *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[6] J. van den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X.-Y. Fu, K. Goldberg, and P. Abbeel, "Superhuman performance of surgical tasks by robots using iterative learning fromhuman-guided demonstration," *IEEE International Conference on Robotics and Automation Anchorage Convention District May 3-8, 2010, Anchorage, Alaska, USA pages 2074-208*, 2010.

[7] T. Osa, N. Sugita, and M. Mamoru, "Online trajectory planning in dynamic environments for surgical task automation," *Robotics: Science and Systems (RSS)*, 2014.

[8] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demostration," *Robotics and autonomous systems, 469-483*, 2009.

[9] *ROBAI. Cyton Gamma 1500 Arm Specifications*.

[10] *ROBOTIS e-Manual v1.31.30 URL: http://support.robotis.com/en/product/actuator/dynamixel/mx_mx-64/at_ar.htm*.

[11] *ROBOTIS e-Manual v1.31.30 URL: http://support.robotis.com/en/product/actuator/dynamixel/mx_mx-28/at_ar.htm*.

[12] *ROBOTIS e-Manual v1.29.00 URL: http://support.robotis.com/en/product/auxdevice/interface/usb2dxl_manual.htm*.

[13] *ROBOTIS Dynamixel SDK e-Manual URL: http://emanual.robotis.com/docs/en/software/dynamixel overview/*.

[14] *Dynamixel Simulink Library version 1.0.0.0 URL: https://www.mathworks.com/matlabcentral/fileexchange/61944-dynamixel-simulink-library*.

[15] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Dynamics and Control*. John Wiley and Sons Inc., 1989.

[16] J. G. Melendez, "Aprendizaje por demostración en el espació articular para el seguimiento de trayectorias aplicado en un exoesqueleto de 4 grados de libertad," Ph.D. dissertation, Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional, 2015.

[17] K. Ogata, *Discrete-Time Control Systems*. Pearson, 1995.

[18] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Pearson, 2010.

[19] S. Haykin, *Neural Networks a Comprehensive Foundation*. Prentice-Hall, 1999.

[20] G.-B. Huang and H. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," *IEEE Trans. Neural Networks*, 9(1)(1998)224-229.

[21] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *ELSEVIER*, 2006.

[22] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: A review," *ELSEVIER*, 2015.

[23] F. W. Isen, *DSP for MATLAB and LabView Volume III: DigitalFilter Design*. Morgan and Claypool, 2009.

[24] J. M. Giron-Sierra, *Digital Signal Processing with Matlab Examples*. Springer, 2017.

[25] Y. Xu and K. K. C. Lee, *Human Behavior Learning and Transfer*. CRC Press, 2005.

[26] A. Billard, S. Calinon, R. Dillmann, and S. Schaa, *Robot Programming by Demonstration*. Springer, 2008.

[27] H. Yu and I. Henríquez, "Music classification based on melodic similarity with dynamic time warping," *IEEE International Conference on Computational Intelligence and Computing Research*, 2013.

[28] J. Garrido, W. Yu, and A. Soria, "Human behavior learning for robot in joint space," *Elsevier*, 2015.

[29] H. Asada and S. Liu., "Transfer of human skills to neural net robotcontrollers," *Proc. IEEE Int. Conf. on Robotics and Automation*, 1991.

[30] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," *AAAI Technical Report WS-94-03*, 1994.

[31] J. Peters and S. Schaal, "Learning to control inoperational space," *The International Journal of Robotics Research*, 2008.