TRABAJO FIN DE GRADO

INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

Controlador Neuronal

Esquemas de aprendizaje de modelos internos de brazo robótico con múltiples articulaciones

Autor
Julio Jesús Vizcaíno Molina

Directores
Jesús Garrido Alcázar
Eva Martínez Ortigosa

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, hoy de 2016

Controlador Neuronal

Esquemas de aprendizaje de modelos internos de brazo robótico con múltiples articulaciones

Autor
Julio Jesús Vizcaíno Molina

Directores
Jesús Garrido Alcázar
Eva Martínez Ortigosa

Controlador Neuronal: Esquemas de aprendizaje de modelos internos de brazo robótico con múltiples articulaciones

Julio Jesús Vizcaíno Molina

Palabras clave: robot_controller, neural_networks, deep_learning, baxter_robot

Resumen

Calcular el modelo dinámico de un brazo robótico es la mejor manera de entender y diseñar un controlador para dicho brazo. Sin embargo, es una tarea complicada que se basa en el conocimiento del funcionamiento del robot, así como de sus características. El controlador que aquí se presenta resuelve estos problemas. Se basa en el auto-aprendizaje del modelo, así como de los parámetros del robot. Para ello hace uso de técnicas de aprendizaje automático (Deep Learning). Estas redes son sistemas de propósito general que parametrizan variables internas en la fase de entrenamiento, para así obtener un modelo concreto al final de esta fase. Este modelo es capaz de desempeñar el papel del controlador empleado para modelarlo sobre datos no visto antes.

Neural Controller: Project Subtitle

Julio Jesús Vizcaíno Molina

Keywords: Keyword1, Keyword2, Keyword3, ...

Abstract

Write here the abstract in English.

Yo, **Julio Jesús Vizcaíno Molina**, alumno de la titulación Grado en Ingeniería de Tecnologías de Telecomunicación de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77151856n, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Julio Jesús Vizcaíno Molina

Granada a hoy de este mes de 2016

D. **Jesús Garrido Alcázar**, Profesor del Área de XXXX del Departamento Arquitectura y Tecnología de Computadores de la Universidad de Granada.

D.ª **Eva Martínez Ortigosa**, Profesora del Área de XXXX del Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Controlador Neuronal, Esquemas de aprendizaje de modelos internos de brazo robótico con múltiples articulaciones***, ha sido realizado bajo su supervisión por **Julio Jesús Vizcaíno Molina**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de mes de 201 .

Los directores:

Jesús Garrido Alcázar **Eva Martínez Ortigosa**

Agradecimientos

Poner aquí agradecimientos...

Índice general

Capítulo 1

Introducción

1.1. Motivación

1.2. Objetivos

Los objetivos del trabajo son los siguientes:

1. Estudio de los mecanismos de control implementados en el robot biomórfico Baxter.
2. Caracterización y obtención de una base de datos de movimientos utilizando los controladores incluidos en el robot.
3. Estudio de la viabilidad de implementación de un sistema de control adaptativo basado en técnicas de machine learning.

1.3. Metodología

Capítulo 2

Materiales y métodos

2.1. ROS

ROS(Robot Operative System) [1] es un sistema operativo de código abierto basado en paso de mensajes diseñado para plataformas robóticas. Sus usos van desde el diseño de drivers, a resolver problemas complejos como la navegación autónoma.

La filosofía del sistema operativo es permitir a los desarrolladores abarcar de manera independiente los problemas que surgen al desarrollar una plataforma robótica, de modo que cada módulo pueda comunicarse con el resto, y que no haya que rediseñar un programa cuando se quiere cambiar de plataforma (y que una misma plataforma se beneficie de la implementación de distintos programas).

La versión usada para trabajar con el robot Baxter ha sido Indigo, ya que es la versión soportada por el robot.

2.1.1. Arquitectura

La arquitectura de ROSsigue un modelo distribuido, donde cada nodo realiza una tarea y los nodos pueden comunicarse entre sí. Esta comunicación lo hace mediante paso de mensajes.

Nodos

Los nodos son los programas que realizan las tareas, ejecutables que utilizan ROSpara para comunicarse con otros nodos. Para programarlos se usan las librerías roscpp (para c++) o rospy (para Python). En el presente trabajo se utiliza la implementación en Python (por motivos de compatibilidad con la API de Baxter, así como con las librerías tensorflow y keras explicadas más adelante).

Temas

Los temas son los canales de comunicación que utilizan los nodos para comunicarse con otros nodos. Son los "puertos" que cada tema pone hace público para permitir la comunicación con otros nodos.

Los nodos publican y se suscriben a los temas.

Mensajes

La comunicación entre nodos se hace con el paso de mensajes. Estos son estructuras de datos que permiten a los nodos especificar y conocer la información que envían y reciben.

Servicios

Se trata de otra manera de comunicación, en la que un nodo hace una petición a otro y éste puede responder. Se diferencia de los mensajes en que el suscriptor puede responder al mensaje recibido, y su uso está enfocado a tareas aperiódicas.

Parámetros

Los parámetros son piezas de información contenidas en cada nodo que pueden ser vistas y modificadas por los nodos.

Ejemplo

Como ejemplo, se puede pensar en una cámara de vídeo que retransmite el vídeo en tiempo real. Esta cámara es un nodo (`/camera`) que publica mensajes del tipo imagen (que consiste en una matriz de nxm píxeles) al tema `/camera/image`.

Por otro lado, en un servidor estamos ejecutando un programa de detección de imágenes haciendo uso de `ROS`para la comunicación (nodo `/camera/server`). Este programa se suscribe al tema `/image` para recibir las imágenes que va a analizar.

Adicionalmente, la cámara tiene habilitado un servicio para reiniciar la cámara (`/camera/reset`) así como un conjunto de parámetros para filtrar las imágenes antes de publicarlas (`/color_mode`), que permite cambiar la imagen a blanco y negro, sepia...

2.1.2. `rosbag`

`rosbag` es una herramienta para grabar el paso de mensajes realizados sobre los temas que se le ofrecen como parámetros. Tiene la capacidad de grabar información adicional a los mensajes, como la marca temporal (momento en el que el mensaje es registrado) y de secuencia (número de mensaje

en una secuencia) que cada mensaje tiene asociada. Esto es útil para sincronizar mensajes generados por otro sistema y recibidos por tcp/ip con los mensajes generados en el propio ordenador.

2.2. Baxter

El robot biomórfico Baxter se trata de un robot de bajo coste (28.000 euros) y de baja fuerza, capaz de trabajar en entornos seguros, con otras personas a su alrededor. Esto se debe al uso de motores de bajo torque, así como del uso de actuadores elásticos en serie [2].

Cuenta con un modelo de movimiento automático basado en aprendizaje de ejemplos dados por una persona, lo que permite a personal no cualificado enseñarle a realizar tareas.

2.2.1. Hardware

Estructura física

El robot cuenta con:

2 brazos Con 7 articulaciones (grados de libertad) cada uno y sensores que miden la posición, velocidad y torque aplicado en cada una de las articulaciones.

Pantalla En la cabeza, con la capacidad de moverse hacia los lados.

3 cámaras Una al final de cada brazo y una en la pantalla, con una resolución máxima de 1280 x 800 píxeles y una tasa de refresco de 30 cuadros por segundo.

Pinzas Que se colocan al final de cada articulación. También se puede colocar una bomba de vacío. No interfieren con las cámaras.

Sonar Ubicado en la cabeza.

Botones Para interactuar con el robot, ubicados en los brazos, así como en el torso, a cada lado del robot.

Las articulaciones son:

s0 Rotación del hombro.

s1 Traslación del brazo.

e0 Rotación del brazo.

e1 Traslación del antebrazo.

w0 Rotación del antebrazo.

w1 Traslación de la mano.

w2 Rotación de la muñeca, a fin de satisfacer dicho movimiento cuando la pinza se encuentra insertada.

Dichas articulaciones se pueden observar en la figura 2.1.

Procesamiento

En el interior del Baxter contamos con un procesador Intel Core i7-3770 a 3.4 GHz con tarjeta gráfica integrada, 4 GB de memoria RAM, y 128 GB de disco duro SSD.

Actuadores elásticos en serie

Los motores utilizados para las articulaciones utilizan actuadores elásticos en serie, en contraposición a los actuadores rígidos, usados en tareas de gran precisión.

Los actuadores rígidos basan su correcto funcionamiento en la cantidad de torque que pueden aplicar: a mayor torque, mayor precisión. Suelen ser controlados sin realimentación, y basan su control en un mapeo de voltajes y posiciones. Al tener tanto torque los motores, cualquier fuerza externa se vuelve despreciable, por lo que se puede confiar su control a uno basado en el conocimiento del motor.

Es este gran torque el que hace que los robots diseñados con estos motores no sean seguros para las personas. Los robots industriales en cadenas de montaje usan este tipo de actuadores.

Por otro lado, los actuadores elásticos basan su funcionamiento en la cantidad de torque que aplican, lo que significa que un controlador basado en realimentación es necesario, ya que el motor no conoce la posición en la que está, sino el torque que está aplicando. Esto lo consigue haciendo uso de elementos elásticos, los cuales aplican una fuerza sobre el objeto a mover. La posición de dicho objeto se mide y se transmite al controlador, que aplicará la fuerza correspondiente.

Los robots basados en estos actuadores son más seguros, ya que la cantidad de torque se puede controlar, ya sea por software o por la propia construcción hardware. Baxter utiliza este tipo de actuadores.

2.2.2. Disposición

Para trabajar con el robot se dispone de un laboratorio de dimensiones reducidas, teniendo en cuenta el rango de movimiento del robot y la compartición del espacio con más gente.

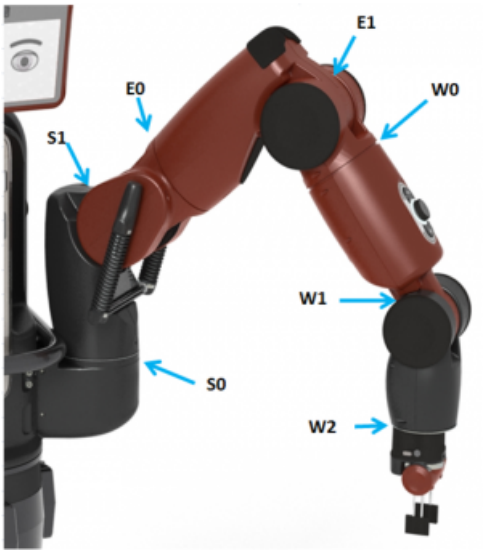


Figura 2.1: Articulaciones del robot Baxter

Además del robot, es necesario un equipo de trabajo, que consistirá en un ordenador de sobremesa con un procesador Intel Core i7-5930k a 3.5 GHz, 32 GB de memoria RAM y 200 GB de disco SSD y 1 TB de disco HDD.

Este equipo se usará para comunicarse con el robot, ejecutando el programa en el ordenador y transmitiendo mensajes entre el robot y éste haciendo uso de `ROS`.

2.2.3. Modos de control

Cada brazo se puede controlar de manera independiente, pudiendo usar cada uno de ellos con un modo de control distinto. Los modos de control son los que se muestran a continuación:

1. Control por posición
2. Control por velocidad
3. Control por torque
4. Control por posición sin procesar (raw mode)

Haciendo uso de la arquitectura de paso de mensajes que nos ofrece `ROS`, somos capaces de enviar en un tema (topic) tanto el modo de control que queremos usar como las características objetivos que queremos en el movimiento. El robot Baxter estará escuchando cada uno de los mensajes que le enviemos a los temas `/robot/limb/left/joint_command` para el brazo

izquierdo y `/robot/limb/right/joint_command` para el brazo derecho. El tipo de mensaje que se envía es `JointCommand`, que tiene como argumentos:

- mode (POSITION_MODE=1, VELOCITY_MODE=2, TORQUE_MODE=3, RAW_POSITION_MODE=4)
- command (lista de flotantes)
- names (lista de nombre de articulaciones)

El modo de control para cada brazo es único, mientras que la posición/-velocidad/torque deseado es único para cada articulación en cada brazo.

Aparte de enviar este tipo de mensajes, podemos definir la velocidad relativa máxima que alcanzará cada articulación en el modo de control por posición publicando al tema `/robot/limb/<right/left>/set_speed_ratio` un valor entre 0 y 1.

El robot Baxter cuenta con una serie de limitaciones en cuanto a posiciones, velocidades y torques se refiere para cada articulación. Estas limitaciones se muestran en el cuadro 2.1.

2.2.4. API de ROS

Baxter hace uso de `ROS`para comunicarse internamente, así como para recibir y enviar información con el exterior.

La interfaz de programación de aplicaciones (API) de Baxter con respecto a `ROS`pone a disposición del programador de los tipos de mensajes y servicios usados por el mismo, así como de los temas publicados para su control.

En este trabajo se hace uso de los siguientes temas:

- `/robot/joint_states` Publica información tanto de la posición como de la velocidad y torque de las articulaciones de todas las articulaciones.

Cuadro 2.1: Límites articulaciones

Art.	(rad) Mín	(rad) Máx	(rad) Rango	(rad/s) Vel máx	(Nm/rad)
S0	-1.7016	+1.7016	3.4033	2.0	843
S1	-2.147	+1.047	3.194	2.0	843
E0	-3.0541	+3.0541	6.1083	2.0	843
E1	-0.05	+2.618	2.67	2.0	843
W0	-3.059	+3.059	6.117	4.0	250
W1	-1.5707	+2.094	3.6647	4.0	250
W2	-3.059	+3.059	6.117	4.0	250

- `/robot/limb/left/set_speed_ratio` Tema al que publicar para ajustar la ratio de velocidad usada por cada articulación del brazo izquierdo (única para todas).
- `/robot/limb/left/joint_command` Tema al que publicar para mover el brazo a las posiciones deseadas del brazo izquierdo (una para cada articulación).

2.2.5. API de Python

La API tiene como objetivo disponer una interfaz basada en el lenguaje de programación Python para controlar y monitorizar el robot Baxter, ejecutando en su base las correspondientes instrucciones `ROS`. Para ello, cuenta con una serie de módulos orientados a los distintos componentes del robot (brazos, pinzas, cámara...).

Se hará uso del módulo `brazo` para realizar el movimiento del mismo. Por motivos de disposición del robot en el laboratorio, se extraerá la base de datos haciendo uso del brazo izquierdo.

Dentro de este módulo, se hará uso de las funciones `set_joint_position_speed(speed)` y `move_to_joint_positions(positions)`. La primera controlará la velocidad máxima relativa para cada articulación, mientras que la segunda moverá el brazo a la posición deseada. Esta función cuenta además con un intervalo de tiempo máximo para realizar el movimiento. Que el tiempo máximo se agote será indicador de que el robot no puede alcanzar la posición deseada (está colisionando consigo mismo).

La función `move_to_joint_positions(positions)` realiza un filtro paso baja de la diferencia de posiciones (actual y deseada) en el tiempo, ofreciendo un movimiento más fluido.

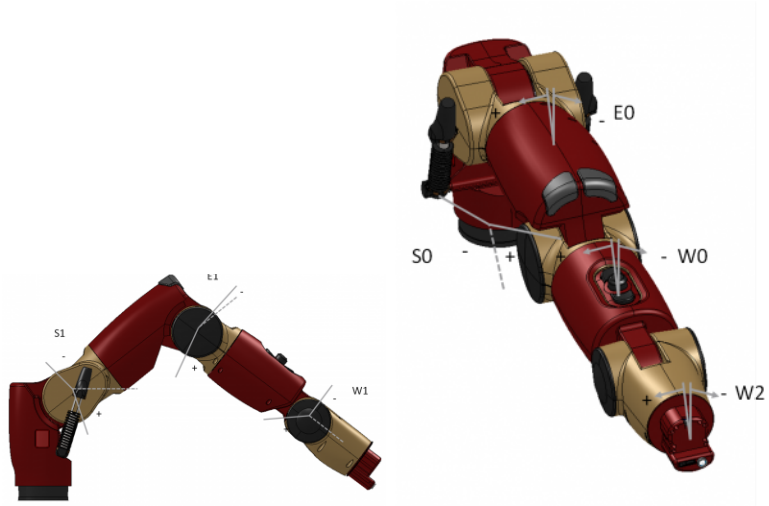
2.2.6. Mecanismos de control

Control por posición

El control por posición consiste en alcanzar las posiciones objetivo para cada una de las articulaciones. El modo de control en el mensaje JointCommand es el 1.

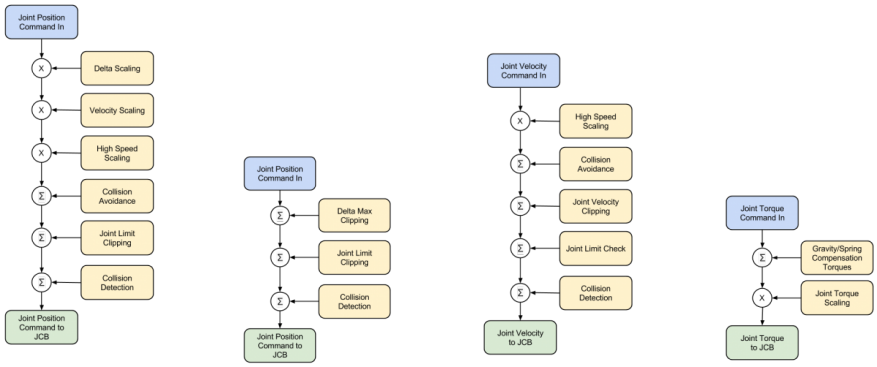
La orden donde todas las articulaciones se ponen en la posición 0 corresponde con el brazo totalmente estirado y con el hombro, codo y muñeca mirando hacia abajo. Las traslaciones y rotaciones se corresponden con los de la figura 2.2.

Dada la naturaleza del robot Baxter, en este modo de operación se aplican unos filtros antes de aplicar la orden de posición, a fin de evitar accidentes y otorgar una experiencia de movimiento más fluida y segura. Los filtros son los que se muestran en la figura 2.4a.



(a) Límites traslación (b) Límites rotación

Figura 2.2: Límites articulaciones



(a) Control por posición (b) Control por posición sin procesar (c) Control por velocidad (d) Control por torque

Figura 2.3: Filtros aplicados

1. Escalado delta. Consiste en escalar las posiciones objetivo en el recorrido para conseguir que todas las articulaciones lleguen al punto deseado a la vez.

2. Escalado de velocidad. Se escala la velocidad de cada articulación en función del 'ratio de velocidad'.

3. Escalado de alta velocidad. Para evitar colisiones entre los dos brazos moviéndose el uno en la dirección del otro, se escala la velocidad y recalcula la posición cuando ésta supera un umbral.

4. Prevención de colisiones. Gracias a un modelo interno del robot, se limitan las posiciones de las articulaciones tales que provocarían un choque con el propio robot.

5. Recorte de posiciones. Si las posiciones superan los límites de la articulación, estas se recortan al límite permitido.

6. Detección de colisión. Si se detecta una colisión (con un objeto externo al robot), mantiene la última posición antes de la colisión.

Como se puede observar, Baxter aplica unos filtros para evitar y detectar colisiones. Esta tarea se realiza de tres maneras:

- Prevención. El robot ejecuta una simulación interna, donde las articulaciones y el cuerpo cuentan con unas regiones de seguridad que, al tocarse, limitan el movimiento del robot.

- Detección de colisión. Esto se realiza de dos maneras

Impacto Cuando el torque de cualquier articulación cambia bruscamente, se considera que ha habido una colisión.

Retención Cuando el torque aplicado aumenta pero la articulación no se mueve, se considera que está colisionando con un objeto inmóvil.

- Escalado de alta velocidad. Cuando la velocidad del brazo supera los 0.2 m/s, las regiones de seguridad que el robot simula internamente aumentan. Cuando se tocan estas regiones, la velocidad se escala y la posición se recalcula.

Control por posición sin procesar

Este modo de control, al igual que el anterior, tiene como objetivo alcanzar la posición deseada para cada articulación. Se diferencia en los filtros que aplica (figura 2.4b).

1. Recorte de delta máximo. Recorta la posición siguiente (en el intervalo de actuación de cada articulación) a la máxima posición alcanzable a la velocidad máxima dada por la 'ratio de velocidad'.

2. Recorte de posiciones. Al igual que en el modo de control anterior, si las posiciones superan los límites de la articulación, estas se recortan al límite permitido.

3. Detección de colisión. Cumple la misma función que en el modo de control por posición.

Como se puede observar, este es un modo de control más avanzado, ya que no tiene en cuenta las colisiones consigo mismo, y ofrece un movimiento más brusco al no llegar todas las articulaciones al punto destino a la vez.

El modo de control en el mensaje JointCommand es el 4.

Control por velocidad

En este modo de control, lo que se busca es adquirir la velocidad objetivo para cada articulación. El modo de control en el mensaje JointCommand es el 2. Al igual que en los modos anteriores, se aplican una serie de filtros (figura 2.4c).

1. Escalado de alta velocidad. Como en el control por posición.

2. Prevención de colisiones. Como en el control por posición.

3. Recorte de velocidades. Se limitan las velocidades para cada articulación para que no excedan el máximo permitido.

4. Comprobación de límites. Se comprueban los límites de las articulaciones. Si se exceden, se detiene el movimiento.

5. Detección de colisión. Igual que en el control por posición.

El dejar de enviar velocidades si se exceden los límites de las articulaciones es una medida de seguridad que exige al programador tener el control sobre las posiciones que puede alcanzar el brazo.

Control por torque

El control por torque consiste en el modo de más bajo nivel que se puede controlar el robot Baxter. Los torques que enviemos serán aplicados por los controladores, haciendo uso solamente de los siguientes filtros (figura 2.4d):

1. Compensación. Los torques se suman a los necesarios para mantener la gravedad 0 y el muelle ubicado en la articulación s1 (traslación del hombro).

2. Escalado de torque. Si un torque excede el torque máximo para esa articulación, escala los torques de todas las articulaciones por el factor de exceso de esa articulación (torque_max/torque).

La compensación de la gravedad se puede desactivar mandando un mensaje vacío al tema `/robot/limb/right/suppress_gravity_compensation` para el brazo derecho, o `/robot/limb/left/suppress_gravity_compensation` para el brazo izquierdo.

De esta manera, un torque aplicado de 0 en todas las articulaciones, mantendrá el brazo en un estado de gravedad 0 si la compensación esta activa. De no estarlo, el brazo caerá en peso muerto.

2.2.7. Simuladores

V-REP

V-REP es un simulador robótico muy potente, con una gran cantidad de modelos predefinidos listos para simular, y una gran cantidad de opciones para configurar.

Gazebo

Al igual que V-REP, es un simulador de plataformas robóticas. Es menos potente, y tiene menos modelos predefinidos, pero está integrado con `ROS`, y el robot Baxter cuenta con un modelo listo para simular para el mismo, por lo que es la opción elegida para realizar las pruebas.

Problemas La versión del simulador en la que está implementada es la 2.2, mientras que la que se puede descargar ahora es la 7.1. Esto hace que no sea lo potente que debiera, y a la hora de realizar simulaciones utilizando el control por torques, el simulador no responde igual que el robot, por lo que su uso quedó limitado a las primeras pruebas que se hicieron en una primera aproximación al control por posición del robot.

2.3. Redes Neuronales

Una red neuronal artificial es un conjunto de algoritmos de aprendizaje automático capaces de extraer modelos a partir de un conjunto de datos de aprendizaje. De esta manera, estos sistemas son capaces de emular la fuente generadora de datos y producir salidas coherentes a partir de entradas no vistas con anterioridad.

2.3.1. Topología

En función de la topología de la red, se contemplan dos tipos fundamentales:

Redes hacia adelante

Las conexiones entre las neuronas es de un solo sentido, de modo que no se forman bucles entre ninguna de las neuronas (figura 2.5).

Propagación hacia atrás de errores

Redes realimentadas

Propagación hacia atrás de errores en el tiempo

Aprendizaje profundo

2.3.2. Regularización

Dropout

L2

2.3.3. Herramientas

Tensorflow

Keras

2.4. Controladores

2.4.1. Error distal

2.4.2. Control realimentado

2.4.3. Control Anticipativo

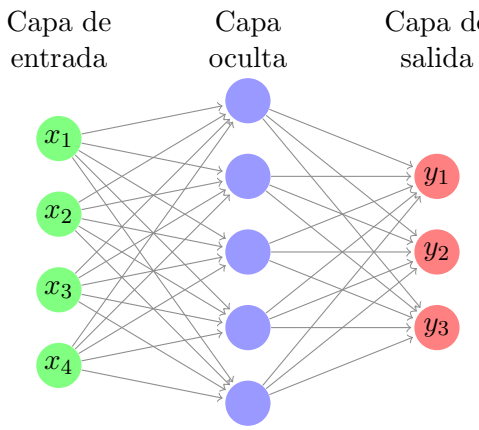


Figura 2.4: Red neuronal de propagación hacia adelante

Capítulo 3

Diseño experimental y resultados

En este capítulo se explicarán las decisiones tomadas en cuanto al diseño del experimento con el fin de alcanzar los objetivos previamente mencionados, así como los resultados obtenidos de dicho experimento.

3.1. Diseño experimental

Una vez estudiados los mecanismos de control del Baxter, se procederá a extraer una base de datos del control por posición, así como diseñar una red neuronal capaz de aprender de este controlador.

3.1.1. Extracción de base de datos

La base de datos consistirá en las posiciones, velocidades y torques registradas por el robot, así como de las órdenes enviadas sobre la posición y la ratio de velocidad deseadas.

Para ello, haremos uso de la interfaz para el lenguaje Python que ofrece Baxter, así como de la herramienta `rosbag`.

Ejecución

El espacio vectorial de las siete dimensiones formadas por cada una de las articulaciones se muestreará de manera uniforme, a fin de obtener una base de datos representativa del controlador que estamos aprendiendo. Esto significa que, para cada movimiento, se elegirán siete posiciones objetivo (una por articulación) de acuerdo a una distribución uniforme sobre el rango de cada articulación. De igual manera se elegirá una ratio de velocidad objetivo con distribución uniforme entre 0 y 1. El tiempo máximo empleado para cada movimiento será de 15 segundos, tiempo suficiente para permitir

el movimiento entre puntos distantes entre sí a una velocidad baja, y por lo tanto, para todos los movimientos.

Tratamiento

Una vez obtenida la base de datos, se prepara para la fase de entrenamiento y evaluación de la red neuronal.

Ficheros `.bag` El primer paso consiste en transformar la base de datos en tipos de datos que entienda el lenguaje de programación Python.

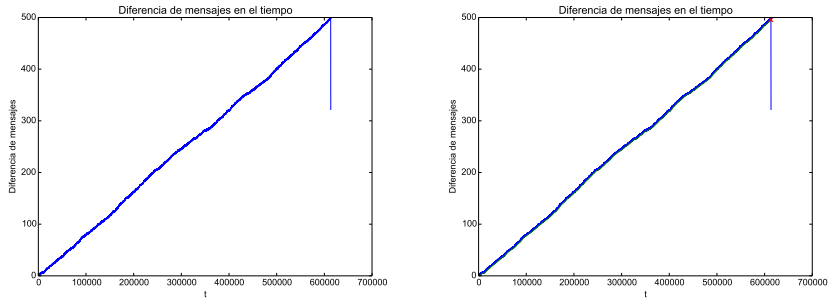
Los ficheros obtenidos con `rosbag` tienen la extensión `.bag`, y consisten en ficheros conteniendo mensajes sobre los que se iteran. De esta manera, iteramos sobre cada mensaje y leemos su contenido, que consiste en el tema del cual proviene el mensaje, una marca temporal de su recepción por `rosbag`, y del mensaje en sí mismo.

La marca temporal corresponde al tiempo del reloj interno del ordenador en el cual se recibe el mensaje. Dado que la conexión entre el robot y el ordenador se realiza por tcp/ip, este tiempo diferirá del de generación en el robot. Para paliar con este problema, `ROS` dispone una marca temporal dentro del cuerpo del mensaje con el instante de creación del mensaje en el robot (de acuerdo a su reloj interno), además de un número de secuencia para ordenar los mensajes en el ordenador.

Error en frecuencia Gracias a esta información, se percibió un error en frecuencia entre el reloj del ordenador y el del robot. Esto significa que, a una frecuencia de generación de mensajes de 100 Hz, el ordenador generó 100 posiciones y velocidades deseadas por segundo (suponiendo que es el que tiene el reloj ajustado correctamente), mientras que el robot generó 99 posiciones, velocidades y torques actuales. Para solventar este problema, en primera instancia se representó gráficamente la diferencia de mensajes obtenidos por parte del ordenador y el robot en función del tiempo (figura 3.1a).

Como se puede observar, esta diferencia crece con el tiempo, lo que significa que el ordenador genera mensajes a mayor velocidad que el robot. Es cuando el ordenador deja de generar mensajes cuando esta diferencia cae en picado, momento en el cual, los mensajes generados por el robot dejarán de corresponderse con las posiciones deseadas y pasarán a corresponderse con las posiciones, velocidades y torques del robot manteniendo la última posición alcanzada.

Por lo tanto, el objetivo será encontrar el momento en el que ocurre dicha bajada, es decir, el pico. El problema surge cuando la señal no es monótona, sino que, como se observa en la figura 3.2a, crece y decrece en intervalos temporales pequeños (debido a la naturaleza a ráfagas de la señal). Para paliar con este inconveniente, se realiza un filtrado paso baja (con un filtro



(a) Señal sin procesar (b) Señal filtrada paso baja

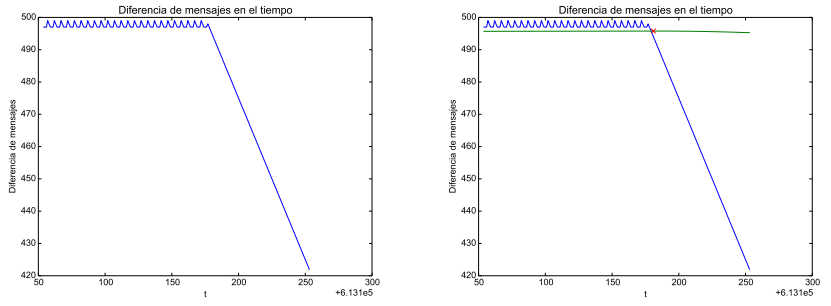
Figura 3.1: Diferencia de mensajes recibidos del ordenador y del robot

de media móvil) que elimine dicha componente frecuencial. El resultado es el que se observa en las figuras 3.1b y 3.2b. A partir de esta señal, solo queda seleccionar el valor máximo de la misma. El valor obtenido es el número de mensaje recibido hasta el cual los mensajes son válidos.

Dado que el control lo vamos a realizar desde el ordenador, adaptamos los mensajes recibidos al mismo. Esto lo hacemos mediante interpolación lineal en las posiciones, velocidades y torques recibidos.

De esta manera, se obtienen vectores de la misma longitud (en el tiempo) tanto enviados como recibidos.

Ráfagas También se percibió la llegada a ráfagas en los mensajes provenientes del robot. Esto no supone un problema para preparar los datos, pero sí lo es a la hora de realizar un controlador realimentado que requiere la información que proporciona el robot sin desfases. El controlador propuesto en este trabajo no es realimentado, pero de serlo, una posible solución



(a) Señal sin procesar (b) Señal filtrada paso baja

Figura 3.2: Detalle de diferencia de mensajes recibidos del ordenador y del robot

sería implementar el controlador en el propio robot (no ejecutarlo en el ordenador), aunque para ello habría que tomar nuevamente la base de datos o ajustarla a la frecuencia de generación de mensajes del robot. Otra posible solución consistiría en un predictor de posiciones y velocidades que ofrezcan al controlador esta información cuando no disponga de la misma. Por ejemplo, supongamos que en un instante concreto, llega una ráfaga de tres posiciones, velocidades y torques provenientes del robot. Esto significa que, dos instantes anteriores, el ordenador no ha obtenido ninguna información, por lo que la última posición, velocidad y torques conocidos son los obtenidos hace dos instantes temporales. En este caso, el predictor generaría 2 posiciones y velocidades (los torques serían la salida del sistema) para los instantes temporales que no disponen de información directa.

3.1.2. Red Neuronal

A continuación se estudia la viabilidad de la implementación de un controlador a partir del propio de Baxter basado en redes neuronales de aprendizaje profundo.

Viabilidad

El primer problema a enfrentarse es el de demostrar que la red neuronal es capaz de aprender algo de los datos extraídos del controlador del Baxter. Por la capacidad de detectar colisiones sabemos que es un sistema realimentado, posiblemente con un controlador PID.

Se podría realizar un controlador donde el modelo dinámico (controlador anticipativo) fuera desempeñado por la red neuronal, y donde la realimentación fuera desempeñada por el controlador PID. También podría implementarse la realimentación como otra red neuronal, pero ambas aproximaciones tienen el problema de enfrentarse a la realimentación y al tiempo real (a una frecuencia de realimentación de 100Hz). Las redes neuronales profundas tienen el problema de ser pesadas en el cómputo (debido a las no linealidades usadas para la activación de las neuronas), poniendo en riesgo la capacidad de afrontar el problema en tiempo real.

Es por esto por lo que se decide hacer un controlador sin realimentación, donde los datos de entrada sean posiciones objetivo, posiciones en el momento de iniciar el movimiento, y velocidad objetivo, siendo los de salida una sucesión de torques a aplicar para alcanzar la posición deseada.

Para ello se elige una arquitectura basada en redes neuronales realimentadas, ya que las no realimentadas tienen la limitación de generar la misma salida ante la misma entrada, y siendo nuestra entrada única (debido a que el sistema no es realimentado), generaríamos siempre la misma salida, en vez de la sucesión de torques deseada.

Nos encontramos con el problema de, si el controlador implementado

por Baxter es realimentado, ¿puede un modelo no realimentado aprender de este? El controlador del Baxter tiene una estructura como la mostrada en la figura 3.3. El funcionamiento es el siguiente:

1. El controlador realimentado genera una señal torque a partir de la realimentación de posición del sistema y la posición deseada.
2. El controlador anticipativo tiene en cuenta el modelo interno del robot y ajusta la señal de control generada por el controlador realimentado.
3. Se transmite la señal y el robot genera el torque enviado.
4. La aplicación genera una nueva posición que es recibido por los sensores y usado para la realimentación.

El controlador realimentado no es más que un sistema (lineal o no) que a ciertas entradas genera ciertas salidas (en función de su historia o no). El controlador anticipativo igualmente es un sistema que tiene en cuenta el modelo dinámico del robot, y por lo tanto los dos sistemas son reproducibles por una red neuronal. Por último, tenemos la realimentación, y es en este aspecto, donde la red neuronal tendrá que aprender a producir estimaciones de la salida provocada (posiciones de las articulaciones) al aplicar los torques.

Por lo tanto, la viabilidad de nuestro controlador depende de la capacidad de la red para generar dicha estimación de la realimentación.

Es por ello por lo que se utilizarán redes neuronales recurrentes, ya que tanto el controlador realimentado, como el anticipativo, como la realimentación en sí misma son sistemas que dependen de la historia de las señales de las que dependen, y esto es exactamente lo que hacen este tipo de redes.

3.2. Resultados

3.2.1. Base de datos

Por motivos de tiempo de entrenamiento, así como para aislar potenciales problemas a la hora de realizar el entrenamiento, se han tomado distintas

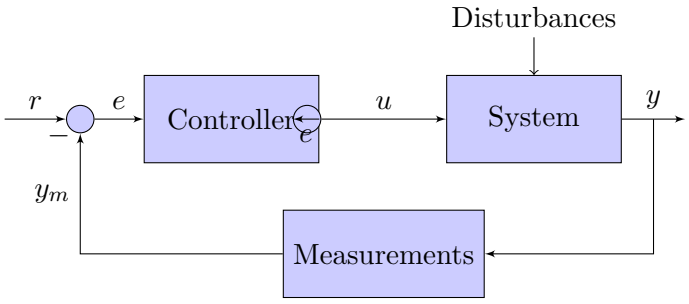


Figura 3.3: Sistema de control de Baxter

bases de datos.

Todas las articulaciones El grueso de nuestra base de datos se ha realizado a partir del movimiento de todas las articulaciones, a distintas velocidades, con un tiempo de espera de 15 segundos.

Una articulación Se ha aislado el problema del tamaño del espacio a muestrear (las siete articulaciones, cada una con su rango de movimiento) reduciéndolo a una de las siete articulaciones en todo su rango de movimiento. El resto de las articulaciones se mantienen quietas en la posición cero.

5 segundos En lugar de limitar el tiempo para cada movimiento a 15 segundos, se limita a 5 segundos, para así aislar el problema de desdoblar la red en un intervalo de tiempo tan alto ($15\,\mathrm{s} * 100\,\mathrm{Hz} = 1500\,\mathrm{muestras}$).

0.5 seg. ratio vel. 1 Es el caso más simple para entrenar la red. Se mueve una sola articulación (el resto se ubican en la posición 0) a máxima velocidad (relación de velocidad 1) durante 0.5 segundos. De esta manera obtenemos una base de datos muy amplia (más de 120 posiciones objetivo por minuto) con el problema del tamaño del espacio y del desdoblamiento temporal muy limitados.

Adicionalmente, se ha obtenido un conjunto de datos con combinaciones de distintas articulaciones, así como del brazo con las pinzas anexas para un tiempo máximo de 15 segundos y velocidad variable.

Capítulo 4

Conclusiones

4.1. Trabajo realizado

4.2. Objetivos alcanzados

4.3. Trabajo futuro

Bibliografía

[1] Open Source Robotics Foundation. *Página principal de ROS*, (último acceso 10 de Diciembre 2016).

[2] Gill A Pratt and Matthew M Williamson. Series elastic actuators. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 1, pages 399–406. IEEE, 1995.