Aprendizaje robótico por imitación utilizando imágenes 2D y 3D

Carlos Andrés Peña Solórzano

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Eléctrica y Electrónica.
Bogotá, Colombia
2015

Aprendizaje robótico por imitación utilizando imágenes 2D y 3D

Carlos Andrés Peña Solórzano

Tesis presentada como requisito parcial para optar al título de:
Magister Automatización Industrial

Director:
Flavio Augusto Prieto

Codirector:
José Gabriel Hoyos Gutiérrez

Línea de Investigación:
Automatización industrial

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Eléctrica y Electrónica.
Bogotá, Colombia
2015

A las personas que hicieron importantes contribuciones a mi vida: Mis padres y hermana

Resumen

Cada vez es más común encontrar robots realizando tareas en áreas compartidas con humanos, donde se espera que sean capaces de aprender de las acciones realizadas por otros y de adaptarse a nuevas situaciones.
La aproximación más utilizada es aprendizaje por imitación, donde el robot es capaz de aprender a partir de la observación de la tarea siendo realizada por un operario. Luego de comparar varias de las técnicas de programación por demostración, se seleccionan las primitivas de movimiento dinámico (DMP) con reconstrucción utilizando regresión de procesos gaussianos (GPR). Las DMP codifican cada uno de las trayectorias dadas por los grados de libertad pertinentes a la acción a aprender, en este caso, llevar la mano hacia un objeto ubicado sobre una mesa. Por otro lado, GPR permite generalizar los movimientos del entrenamiento a nuevas trayectorias, cuando cambian tanto la posición inicial de la mano como la ubicación del objeto.
Se realizó una comparación de varias técnicas de aprendizaje, teniendo en cuenta el error al objetivo, la correlación cruzada entre las señales de entrada y salida, y el tiempo de codificación y reconstrucción de la trayectoria. Además, la técnica de generalización se compara contra un algoritmo basado en distancia de Mahalanobis y distribución gaussiana, que utiliza los datos de la trayectoria sin codificar para realizar la estimación. La técnica regresión de procesos gaussianos, presentó un mayor desempeño al probarlo con 30 puntos de consulta para el valor inicial de la mano, y 30 puntos para la posición final o posición del objeto.
La técnica de regresión de procesos gaussianos junto a primitivas de movimiento dinámico, se presenta como una solución para el aprendizaje por imitación de tareas, así como para la generalización a nuevas trayectorias a partir de la base de datos, al presentar bajos tiempos de codificación y errores pequeños con respecto a los valores objetivo.

Palabras clave: aprendizaje por imitación, generalización de trayectorias, primitivas de movimiento dinámico, procesamiento de imágenes..

Abstract

It is becoming increasingly common to find robots performing tasks in shared areas with humans, they are expected to be able to learn from the actions taken by others and adapt to new situations.
The most widely used approach is learning by imitation, where the robot is able to learn from watching the task being performed by an operator. After comparing several programming by demonstration techniques, the dynamic movement primitives (DMP) with reconstruction using Gaussian process regression (GPR) was selected. DMP encodes each of the paths given by the relevant degrees of freedom to bring the hand toward an object placed on a table. Furthermore, GPR allows to generalize the training movements to new paths when changing both the initial hand position and the location of the object.
A comparison of various learning techniques was performed, considering the error to the target, the cross-correlation between the input and output signals, and time of coding and reconstruction of the trajectory. Besides, the technique is compared against a generalization based on the Mahalanobis distance and Gaussian distribution, which uses data from uncoded trajectories for the estimate. The Gaussian process regression technique, presented a better performance when tested with 30 queue points for the initial value of the hand, and 30 points for the final position of the object.
Gaussian process regression along dynamic movement primitives is presented as a solution for learning by imitation of task, as well as generalization to new paths from the database, because of its fast encoding times and small errors regarding the target values.

Keywords: Learning by imitation, trajectory generalization, dynamic movement primitives, image processing.

Contenido

Lista de Figuras

Lista de Tablas

Lista de símbolos

Abreviaturas

Abreviatura	Término
LbI	Learning by Imitation
PbD	Programming by Demonstration
DMP	Dynamic Movement Primitives
GMM	Gaussian Mixture Models
GMR	Gaussian Model Regression
ANN	Artificial Neural Networks
HMM	Hidden Markov Models
LWR	Locally Weighted Regression
GPR	Gaussian Process Regression

1 Introducción

La investigación en áreas como la robótica y los sistemas autónomos ha presentado grandes avances en los últimos años, de acuerdo a varios autores [4, 5, 6]. Dichos avances han llevado al desarrollo de plataformas con capacidades motoras, perceptuales y cognitivas cada vez más complejas. Sumado a esto, el área de la robótica está cambiando rápidamente de ambientes industriales controlados, hacia ambientes dinámicos con interacción humana, debido principalmente al incremento de robots de servicio, los cuales se espera sean parte de muchas actividades humanas en el futuro próximo [7, 8].

El aumento de dicha interacción presenta grandes desafíos relacionados con el *aprendizaje* robótico y las capacidades de *interactividad* [9]. Las metodologías tradicionales de programación e interfaz con los robots no son suficientes dada la dinámica de los nuevos entornos a que están expuestos. Los autómatas requieren de la capacidad de aprender a realizar nuevas tareas de acuerdo con las preferencias de los usuarios, además deben mejorar el desempeño durante su ciclo de vida y permitir que personas no expertas puedan programar nuevas actividades robóticas de forma natural [6]. Como respuesta a estos problemas, se presentan las técnicas de aprendizaje basadas en demostración (LbD por sus siglas en inglés), llamadas también programación por demostración (PbD por sus siglas en inglés), aprendizaje por imitación (LbI por sus siglas en inglés) y aprendizaje por observación, entre otros [10, 7, 8].

En esta forma de programación, el robot, provisto de todos los sensores y hardware físicos necesarios para realizar una tarea, observa como ésta es realizada por un humano u otro robot e intenta imitar los movimientos y/o cumplir el objetivo de la tarea. Por esto, recientemente dichas técnicas de aprendizaje han atraído la atención de los investigadores en robótica, al considerarlo un mecanismo de aprendizaje prometedor para transferir conocimientos de un experto a un agente artificial [5, 11].

Existen tres razones principales para migrar de robots puramente preprogramados a interfaces flexibles basadas en usuario para entrenar robots [12]: Primero, la observación de ejemplos reduce la búsqueda de una posible solución, permitiendo mejorar y acelerar el aprendizaje. Segundo, el aprendizaje por imitación ofrece una manera implícita de entrenar una máquina, minimizando o eliminando la tediosa tarea de programación. Tercero, estudiando y modelando la interacción entre percepción y acción, que es el núcleo del aprendizaje por imitación, permite proponer mejoras en los procesos de percepción con el fin de ejecutar acciones de

manera exitosa.

Uno de los desafíos que se tratan en la actualidad, es cómo enseñar a los robots de manera simple, a realizar tareas comunes para los humanos. En la base de este problema, radica la importancia de que el robot pueda detectar objetos antes de poder manipularlos. Se propone calcular la ubicación y orientación de los objetos en el espacio, usando varias etapas de procesamiento de imágenes. Para enseñarle al robot las trayectorias que debe seguir su brazo para alcanzar un objeto, se emplea una técnica de programación por demostración seleccionada luego de una comparación, obteniendo los datos de entrenamiento de un operario que realiza la acción a distintos puntos en el espacio. Utilizando las imágenes de color y profundidad obtenidas con el sensor Kinect de Microsoft, es posible calcular los ángulos de las articulaciones del operario al realizar las tareas. Para aumentar las habilidades finales del robot, se propone emplear una técnica de regresión para generalizar, a partir de la base de datos, a nuevas trayectorias dadas posiciones iniciales y finales cercanas a las usadas en el entrenamiento.

Para la reproducción de los movimientos aprendidos, se plantea utilizar el simulador del robot iCub [13, 14]. El robot fue diseñado como una plataforma común para investigadores interesados en el estudio de sistemas cognitivos artificiales. El iCub es del tamaño aproximado de un niño de tres años, y está equipado con un gran número de sensores, la mayoría de los cuales pueden utilizarse durante las simulaciones.

El documento se encuentra organizado de la siguiente forma: en el Capítulo 2, se presentan los trabajos relacionados en el área de aprendizaje por imitación. En el Capítulo 3, se presenta la implementación y evaluación de las técnicas de programación por demostración. En el Capítulo 4, se describe la adquisición de datos tanto de la ubicación del objeto como de la tarea realizada por la persona. En el Capítulo 5 se describe la técnica empleada, se describen los experimentos y se presentan los resultados. Se finaliza con las conclusiones, en el Capítulo 6.

2 Técnicas de aprendizaje por imitación - Trabajos relacionados

Las técnicas de aprendizaje por imitación pueden clasificarse de distintas formas, de las cuales algunas se presentan a continuación.

2.1. Clasificación de las técnicas de aprendizaje por imitación

Las técnicas de programación por demostración han sido clasificadas de acuerdo a varios criterios, algunas de las cuales se presentan en la Figura **2-1**.

De acuerdo a la Figura **2-1**, se presenta una descripción más detallada de las categorías.

Según la aproximación utilizada.

Aprendizaje de la política de control de forma directa, también llamada imitación a nivel de tarea, donde se requiere un conocimiento previo de cómo transformar las acciones a nivel de tarea en acciones a nivel de actuador [15]. *Aprendizaje de trayectorias demostradas*, donde se utilizan datos de las coordenadas del objeto manipulado, además del movimiento del actuador en términos de los ángulos entre las articulaciones [16]. *Aprendizaje por imitación basado en modelo*, donde a partir del comportamiento demostrado, la dinámica de la tarea se aproxima en forma de un modelo de adelanto predictivo [17, 18, 19].

Conforme a la complejidad de la tarea a imitar, de menor a mayor grado.

Imitación de un gesto simple [20], aprendizaje de una secuencia de gestos para formar un movimiento más complejo [21], generalización de los movimientos sobre el rango permitido por el cuerpo [22, 23], y aprendizaje de diferentes categorías de movimientos especializados para tareas específicas [24, 25].

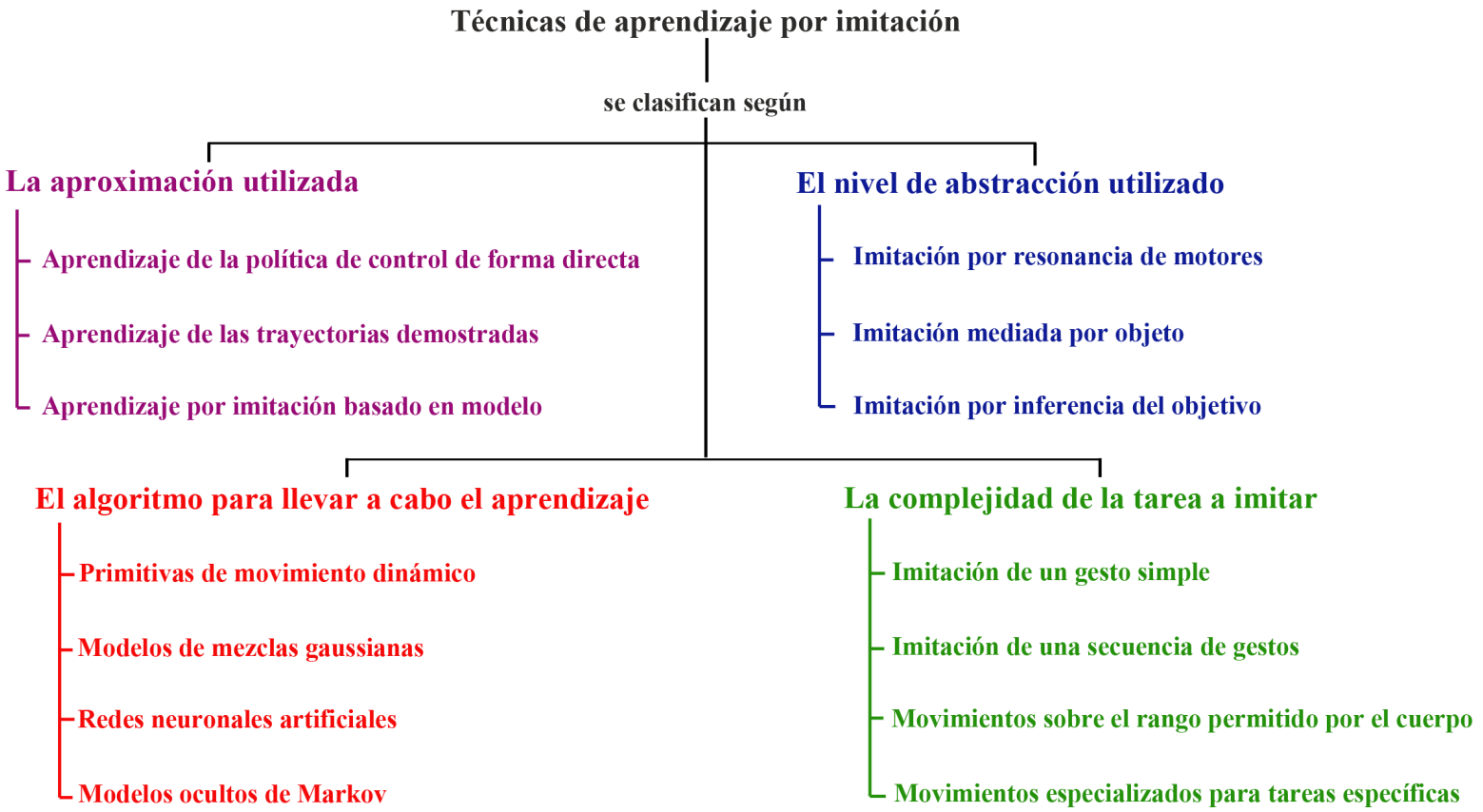

Figura 2-1: Algunas clasificaciones de las técnicas de aprendizaje por imitación.

De acuerdo al nivel de abstracción utilizado, de menor a mayor grado.

Imitación por resonancia de motores, donde se realiza un mapeo de las trayectorias estado-acción observadas al cuerpo del imitador y se generalizan [26]. *Imitación mediada por objeto*, donde se tiene en cuenta la interacción de quien realiza la tarea con los objetos del entorno, replicando los resultados o efectos de las acciones [27, 28]. *Imitación por inferencia del objetivo*, donde se deduce el objetivo que motiva la demostración observando la realización de esta [6, 29].

Según el algoritmo utilizado para llevar a cabo el aprendizaje.

Las primitivas de movimiento dinámico (DMP por sus siglas en inglés), permiten codificar los movimientos del robot, donde una DMP puede codificar una trayectoria específica. En el caso de un movimiento discreto (punto a punto), cada grado de libertad del robot (DoF por sus siglas en inglés) se describe por un sistema de ecuaciones diferenciales no lineales [25], lo que le permite reaccionar a perturbaciones sin introducir discontinuidades en el movimiento resultante. Además, DMP no es directamente dependiente del tiempo, por lo que es fácil detener la ejecución del movimiento sin tener en cuenta la evolución del tiempo de forma extensiva.

Las DMPs pueden ser extendidas para incluir capacidad de evitar obstáculos, cambiando de una primitiva a otra mediante retroalimentación sensorial [30] o creando posiciones objetivo virtuales utilizando campos potenciales [31]. Nemec y Ude [32] proponen realizar acciones secuenciales, asegurando que las primitivas de movimiento consecutivas se unen en forma continua. Además de replicar los movimientos o acciones aprendidas, utilizando una reconstrucción probabilística basada en regresión de procesos gaussianos (GPR por sus siglas en inglés) pueden crearse acciones para situaciones que el robot no ha encontrado anteriormente, lo cual se presenta tanto en simulación como en experimentos reales de alcanzar posiciones, tomar objetos y lanzar esferas [33, 34].

Los modelos de mezclas gaussianas (GMM por sus siglas en inglés), son una estrategia popular para aproximación de señales utilizando densidad de datos binarios o continuos. GMM concede gran flexibilidad al permitir un compromiso entre la complejidad del modelo resultante al evaluar la señal de entrada, y las variaciones en los datos de entrenamiento disponibles. El conjunto de datos a modelar pueden ser ángulos de las articulaciones, trayectorias de las manos, o vectores de distancia entre mano y objeto. El modelado se hace a través de una mezcla de funciones de densidad probabilística gaussianas, mientras que la reconstrucción suele realizarse con regresión de mezclas gaussianas (GMR por sus siglas en inglés) [35].

GMM se ha utilizado principalmente en tareas de manipulación, donde la trayectoria imitada se remodela para satisfacer las restricciones, pudiendo adaptarse a cambios en las condiciones iniciales y a desplazamientos del objetivo ocurridos durante la ejecución del movimiento [1, 36, 37]. Una aplicación más específica es presentada por Reiley *et al.* [38], donde a través de demostraciones de tareas quirúrgicas por parte de expertos, se extraen características importantes de la tarea y se generan trayectorias más suaves para la reproducción.

Las redes neuronales artificiales (ANN por sus siglas en inglés), han sido usadas comúnmente para resolver problemas y en aplicaciones de robótica más allá del aprendizaje por imitación. Éstas redes requieren un entrenamiento utilizando los datos obtenidos de la observación de la tarea. Una vez que se alcanza la suficiente exactitud en el aprendizaje, la red neuronal artificial obtenida tiene la capacidad de producir una respuesta que imite la tarea realizada en la demostración [39]. ANN se ha implementado para enseñar a un robot patrones de movimientos cíclicos, pudiendo recontruir cada patrón de forma síncrona con el movimiento de un humano realizando el movimiento frente a él [40], además de aplicaciones de programación por demostración en robots industriales para clasificación de objetos [41] y sistemas de reconocimiento de gestos y lenguaje de signos [42].

Los modelos ocultos de Markov (HMM por sus siglas en inglés), son un método de modelamiento probabilístico que deduce un sistema dinámico capaz de generar los datos de entrada dados [2]. Éstos modelos se emplean para generar y representar tareas básicas a nivel de motor en arquitecturas de aprendizaje por imitación, como lo indica Azad *et al.* [43]. También son utilizados para extraer regularidades y filtrar las acciones a aprender. El robot observa la realización de la tarea y construye un modelo oculto de markov que la describe [6]. Extensiones de la técnica se han desarrollado, incluyendo parámetros que permiten simultáneamente resolver los problemas de reconocimiento, parametrización, y síntesis de la acción observada [44], además de extraer de un conjunto de demostraciones, aquellas perceptualmente similares [45].

Para el trabajo presentado, se utilizó la categoría basada en el algoritmo utilizado para llevar a cabo el aprendizaje (Figura **2-1**), los cuales se detallan a continuación.

2.2. Primitivas de movimiento dinámico

Las primitivas de movimiento dinámico, permiten la codificación de los movimientos del robot o quien realiza la demostración. Entre sus características principales, se encuentran su capacidad de reaccionar a perturbaciones sin introducir discontinuidades en el movimiento resultante, y su dependencia no directa del tiempo, que permite detener la ejecución del movimiento de forma independiente a la evolución del tiempo [30, 46].

Una primitiva de movimiento dinámico puede codificar una trayectoria específica del robot (Figura **2-2**). La trayectoria de cada grado de libertad y, dado en espacio de articulaciones o de la tarea, está descrito por el siguiente sistema de ecuaciones diferenciales (Ecuaciones 2-1-2-3) [33, 47].

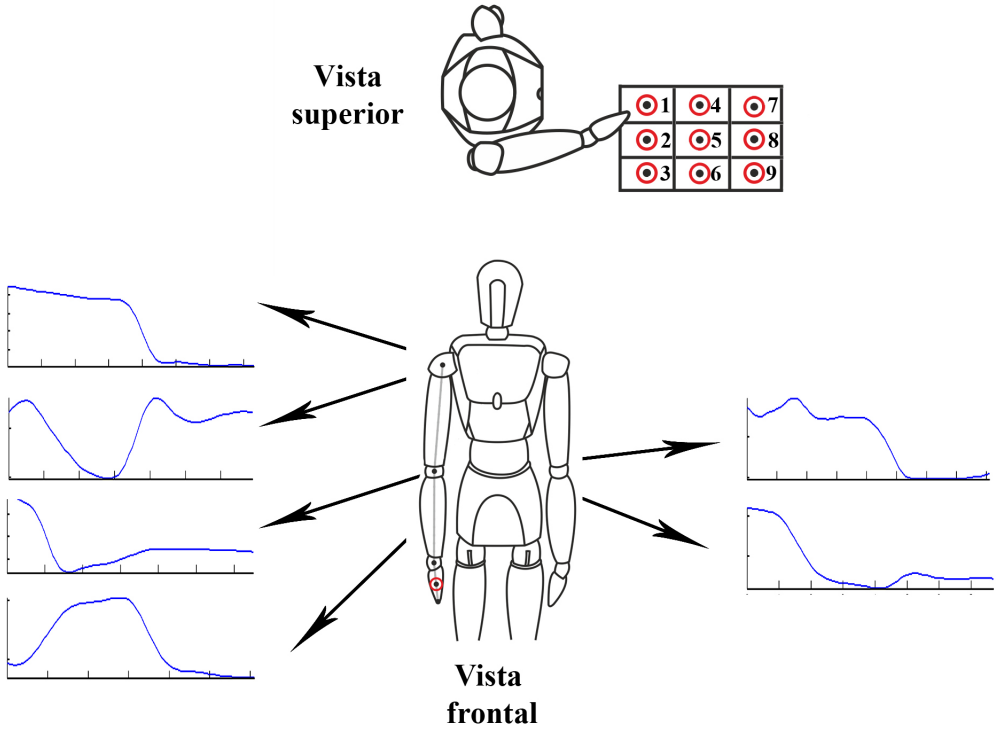

Figura 2-2: Las primitivas de movimiento dinámico permiten codificar los movimientos realizados durante la demostración.

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) + f(x), \tag{2-1}$$

$$\tau \dot{y} = z, \tag{2-2}$$

$$\tau \dot{x} = \alpha_x x, \tag{2-3}$$

donde z es una variable auxiliar y x es la variable de fase utilizada para hacer implícita la dependencia de f del tiempo (con valor inicial $x(0) = 1$). α_x, α_z, β_z y τ son variables que deben definirse de tal forma que el sistema converja al punto de equilibrio $(z, y, x) = (0, g, 0)$.

El término no lineal f contiene parámetros libres que permiten al robot realizar un movimiento suavizado punto a punto de la posición inicial y_0 a la configuración final g.

$$f(x) = \frac{\sum_{k=1}^{N} w_k \psi_k(x)}{\sum_{k=1}^{N} \psi_k(x)} x, \qquad \psi_k(x) = \exp(-h_k(x - c_k)^2), \tag{2-4}$$

donde c_k son los centros de las funciones de base radial distribuidos en la trayectoria y $h_k > 0$. Los pesos w_k se estiman para que el DMP codifique la trayectoria deseada. Cada grado de libertad está representado por su propio sistema de ecuaciones (2-1)-(2-2), con una fase común (2-3).

Conversión de las trayectorias a primitivas de movimiento dinámico

Supongamos que tenemos un conjunto de movimientos del robot $\mathbf{M}_i$, $i = 1, ..., NumEj$, donde $NumEj$ es el número de movimientos que resultan en una ejecución exitosa de una tarea determinada. Cada movimiento $\mathbf{M}_i$ está codificado por una secuencia de puntos de la trayectoria $\{y_{ij}, \dot{y}_{ij}, \ddot{y}_{ij}\}$, medidos en los tiempos t_{ij}, $j = 1, ..., n_i$, $t_{i1} = 0$, donde n_i denota el número de muestras en la trayectoria.

Para reducir la cantidad de datos que deben procesarse para generalizar la acción, primero se codifican cada uno de los movimientos demostrados $\mathbf{M}_i$, como primitivas de movimiento dinámico. Se denota

$$f_{ijl} = {\tau_i}^2 \ddot{y}_{ijl} - \alpha_z(\beta_z(g_{il} - y_{ijl}) - \tau_i \dot{y}_{ijl}), \tag{2-5}$$

donde $i = 1, ..., NumEj$, $j = 1, ..., n_i$, $l = 1, ..., dof$.

La Ecuación (2-5) se obtiene al reescribir el sistema de dos ecuaciones diferenciales de primer orden (2-1)-(2-2), como una de segundo orden (reemplazando $z = \tau\dot{y}, \dot{z} = \tau\ddot{y}$ en la Ecuación (2-1)). Los parámetros w_{ikl}, $k = 1, ..., N$, donde N es el número de funciones kernel de DMP (Ecuación (2-4)), pueden ser estimados resolviendo el problema de regresión lineal (2-6).

$$\mathbf{Xw} = \mathbf{f}, \tag{2-6}$$

donde

$$\mathbf{X} = \begin{bmatrix} \frac{\psi_1(x_1)}{\sum_{i=1}^{N}\psi_i(x_1)}x_1 & \dots & \frac{\psi_N(x_1)}{\sum_{i=1}^{N}\psi_i(x_1)}x_1 \\ \dots & \dots & \dots \\ \frac{\psi_1(x_T)}{\sum_{i=1}^{N}\psi_i(x_T)}x_T & \dots & \frac{\psi_N(x_T)}{\sum_{i=1}^{N}\psi_i(x_T)}x_T \end{bmatrix}, \tag{2-7}$$

y $\mathbf{w} = [w_1, ..., w_N]$, $\mathbf{f} = [f_1, ..., f_T]$. Los parámetros de fase $x_{ij} = x(t_{ij})$ se calculan integrando la Ecuación (2-3) con la condición límite $x_{i1} = x(0) = 1$.

El proceso anterior permite convertir los datos iniciales de la trayectoria $\mathbf{M}_i$ en una DMP. $\mathbf{M}_i \mapsto (w_i, g_i, \tau_i)$, donde $\mathbf{w}_i \in \Re^{N \times dof}$ son los pesos de DMP para todos los grados de libertad, $\mathbf{g}_i \in \Re^{dof}$ son los valores finales de las trayectorias demostradas, y $\tau_i \in \Re$ es el tiempo de duración de las trayectorias demostradas.

DMP se ha utilizado con aprendizaje por refuerzo, donde los objetivos y los parámetros de tiempo se fijan, y sólo se aprenden los pesos para ajustar la forma de la trayectoria. Cuando la mejor posición objetivo no es conocida, es necesario predecirla de forma simultánea evitando que el aprendizaje de ambos parámetros interfiera de forma destructiva uno con otro. Para solucionar este problema, Tamosiunaite *et al.* [48] utilizan técnicas de aproximación al valor de la función para el aprendizaje del objetivo, y métodos de búsqueda de política directa para predecir los pesos. Los resultados fueron comprobados en una tarea de aprender a verter un líquido, tanto en simulación como en un brazo robótico. El método demostró ser estable y robusto. Por su parte, Stulp *et al.* [49], utiliza aprendizaje por refuerzo para enseñarle al robot primitivas de movimiento robustas ante incertidumbres en la estimación del estado. Específicamente, durante maniobras de alcance de objetos, el robot aprende estrategias de manipulación fina para llevar al objeto a una orientación en que el agarre tenga mayores posibilidades de éxito. Los resultados demuestran que las primitivas de movimiento se vuelven más robustas al agarrar objetos cilíndricos, además de objetos no convexos.

En [50], DMP se utiliza para enseñarle al robot habilidades de manipulación fina, un golpe preciso en billar y el girar una caja utilizando palillos como herramientas. Se agrega al robot realimentación sensorial para aprender un modelo predictivo del resultado de la tarea, lo que le permite abortar ejecuciones de la tarea donde se predice una falla.

Para generar patrones de movimiento más complejos, Kulvicius *et al.* [51] modifican la formulación original de DMP. El método que proponen, puede reproducir la trayectoria al blanco con gran exactitud, produciendo transiciones suaves y naturales, tanto en posición como en velocidad. Los resultados fueron comprobados mediante generación de escritura a mano simulada, la cual se llevó también a la práctica en un robot.

Para interacción humano-robot, el trabajo de Prada *et al.* [52] presenta la implementación y validación experimental de un sistema para entrega de objetos entre un robot y un operario. La ley de control define una trayectoria para el brazo robótico hacia una posición de entrega desconocida previamente. Para la evaluación, se realizaron 1000 pruebas de intercambio de objetos entre robot y humano, validando la implementación y creando una guía hacia interacciones más eficientes.

2.3. Modelos de mezclas gaussianas

Una representación probabilística de los datos $\xi_j = \{\xi_{t,j}, \xi_{s,j}\}$ con $\xi_{t,j} = X_{t,j}$ se usa para estimar las variaciones y correlaciones entre las variables, permitiendo una caracterización localizada de las diferentes partes del movimiento. El modelado de mezclas permite la aproximación en densidad de datos continuos o binarios [36] y presenta flexibilidad al buscar un compromiso entre la complejidad del modelo y las variaciones en los datos de entrenamiento. Un modelo de mezclas con K componentes está definido por la función de densidad de probabilidad (Ecuación 2-8).

$$p(\xi_j) = \sum_{k=1}^{K} p(k)p(\xi_j \mid k), \tag{2-8}$$

donde ξ_j es el dato de entrada, $p(k)$ es la probabilidad previa, y $p(\xi_j \mid k)$ es la función de densidad de probabilidad condicional.

Consideremos un conjunto de datos $\{\xi_{t,j}, \xi_{s,j}\}_{j=1}^{N}$. Los datos consisten de N puntos de dimensionalidad D, incluyendo la información temporal, y pueden representar información en el espacio de articulación o de la tarea. El conjunto de datos es modelado por una mezcla de K gaussianas de dimensionalidad D, como se presenta en la Ecuación 2-9.

$$p(k) = \pi_k,$$

$$p(\xi_j \mid k) = N(\xi_j; \mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^D |\Sigma_k|}} \exp(-\frac{1}{2}((\xi_j - \mu_k)^T \Sigma_k^{-1} (\xi_j - \mu_k))), \tag{2-9}$$

donde $\{\pi_k, \mu_k, \Sigma_k\}$ son los parámetros de la componente gaussiana k, que definen respectivamente la probabilidad previa, la media y la matriz de covarianza. La estimación de la máxima probabilidad de los parámetros de la mezcla se lleva a cabo de manera iterativa, utilizando el algoritmo Esperanza - Maximización (EM). EM es una técnica simple de búsqueda local que garantiza un incremento monotónico de la probabilidad del conjunto de entrenamiento

durante la optimización. El algoritmo requiere un estimado inicial , para lo cual se utiliza la técnica de agrupación k-medias, que evita caer en mínimos locales. Los parámetros gaussianos se derivan de los clústers encontrados por k-medias.

Regresión de mezclas gaussianas

Para reconstruir la forma general de la señal, se aplica regresión de mezclas gaussianas. Valores de tiempo consecutivos ξ_t se utilizan como datos de prueba, y los valores espaciales correspondientes $\hat{\xi}_s$ se estiman a través de la regresión. Para cada GMM, los componentes espaciales y temporales (parámetros de entrada y salida) son separados, es decir, la media y la matriz de covarianza de la componente gaussiana k están definidos por (2-10) [35].

$$\mu_k = \{\mu_{t,k}, \mu_{s,k}\}, \qquad \Sigma_k = \begin{pmatrix} \Sigma_{t,k} & \Sigma_{ts,k} \\ \Sigma_{st,k} & \Sigma_{s,k} \end{pmatrix}. \tag{2-10}$$

Para cada componente gaussiana k, la esperanza condicional de $\xi_{s,k}$ dado ξ_t, y la covarianza condicional estimada de $\xi_{s,k}$ dado ξ_t, son los observados en la Ecuación 2-11.

$$\hat{\xi}_{s,k} = \mu_{s,k} + \Sigma_{st,k}(\Sigma_{t,k})^{-1}(\xi_t - \mu_{t,k}),$$

$$\hat{\Sigma}_{s,k} = \Sigma_{s,k} - \Sigma_{st,k}(\Sigma_{t,k})^{-1}\Sigma_{ts,k}. \tag{2-11}$$

$\hat{\xi}_{s,k}$ y $\hat{\Sigma}_{s,k}$ se mezclan de acuerdo a la probabilidad de que el componente gaussiano $k\epsilon\{1, ..., K\}$ sea responsable por ξ_t.

$$\beta_k = \frac{p(\xi_t \mid k)}{\sum_{i=1}^{K} p(\xi_t \mid i)}. \tag{2-12}$$

Utilizando las ecuaciones (2-11) y (2-12), para una mezcla de K componentes, la esperanza condicional de ξ_s, dado ξ_t, y la covarianza condicional de ξ_s, dado ξ_t, están dados por la Ecuación 2-13.

$$\hat{\xi}_s = \sum_{k=1}^{K} \beta_k \hat{\xi}_{s,k}, \qquad \hat{\Sigma}_s = \sum_{k=1}^{K} \beta_k^2 \hat{\Sigma}_{s,k}. \tag{2-13}$$

Por lo tanto, evaluando $\{\hat{\xi}_s, \hat{\Sigma}_s\}$ en diferentes tiempos ξ_t, se obtiene una forma generalizada de los movimientos $\hat{\xi} = \{\xi_t, \hat{\xi}_s\}$ y se producen las matrices de covarianza asociadas (Figura **2-3**). Con este modelo probabilístico, solo las medias y las matrices de covarianza de las

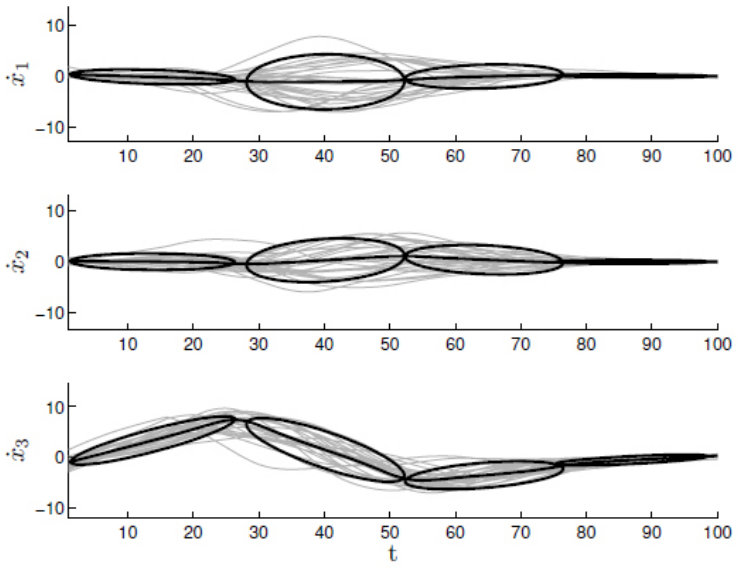

Figura 2-3: Modelo de mezclas gaussianas entrenado con 26 demostraciones. Las líneas delgadas representan la generalización aprendida utilizando GMR y los elipses representan los componentes gaussianos de la distribución de probabilidad mixta. Tomado de [1].

gaussianas deben mantenerse en memoria.

GMM se ha utilizado para alcance de objetos por parte de un robot [37], donde la trayectoria imitada puede cambiar de forma para satisfacer restricciones y adaptarse a cambios en las condiciones iniciales y ha desplazamientos del objetivo durante la ejecución del movimiento. Los resultados de dicho artículo se obtuvieron de experimentos en el robot Nao, el robot humanoide de Aldebaran.

Se han hecho desarrollos que permiten mejorar el desempeño y la eficiencia de GMM, mediante el uso de algoritmos difusos (modelos de mezclas gaussianas difusos o FGMM por sus siglas en inglés) [53]. La comparación se realizó contra GMM convencional en términos del grado de ajuste y velocidad de convergencia. La técnica que proponen no sólo puede ajustarse de forma no lineal, sino que también el tiempo de convergencia es mucho menor.

El desempeño de GMM está ligado a la elección del número apropiado de mezclas, para lo cual Kruger *et al.* [54] proponen un modelo de mezclas gaussianas infinitas (IGMM por sus siglas en inglés). En esta aproximación, IGMM encuentra automáticamente el número de mezclas necesarias para reflejar la complejidad de los datos. Los resultados se obtuvieron de una base de datos de entrenamiento usada en un proyecto anterior, además de trayectorias adquiridas con el iCub cuando un operario mueve el cuerpo del robot.

2.4. Redes neuronales artificiales

Las redes neuronales artificiales han jugado un rol importante en la programación de robots por demostración, aunque presentan una gran desventaja: la red debe ser entrenada con una gran cantidad de datos para obtener buenos resultados, lo que hace a esta técnica lenta en comparación con las demás estudiadas [41, 40]. Entre las principales ventajas se encuentra una reproducción fiel de la señal original, asi mismo como un error del valor final pequeño. Suelen utilizarse para ajuste de datos, agrupamiento, y reconocimiento de patrones; además, la arquitectura de las redes puede ser supervisada o no supervisada.

La red implementada en el capítulo siguiente presenta una arquitectura supervisada, la cual se entrena para producir una salida deseada en respuesta a muestras de entrada, por lo que se utilizan para modelar y controlar sistemas dinámicos, clasificar datos con ruido, y predecir eventos futuros. Además, es una red feedforward (o prealimentada), la cual presenta conexiones en un sentido desde la capa de entrada hasta las capas de salida. Se suelen utilizar para predicción, reconocimiento de patrones, y ajuste de funciones no lineales.

La red de dos capas creada, presenta neuronas ocultas con función de disparo sigmoidal y neuronas de salida con función de disparo lineal (Figura **2-4**). Este tipo de redes pueden ajustar problemas de mapeado multidimensional, dados datos consistentes y suficientes neuronas en la capa oculta. Para ajustar de forma automática los pesos y bias de la red, se utilizan funciones de aprendizaje como el algoritmo Levenberg-Marquardt (LM).

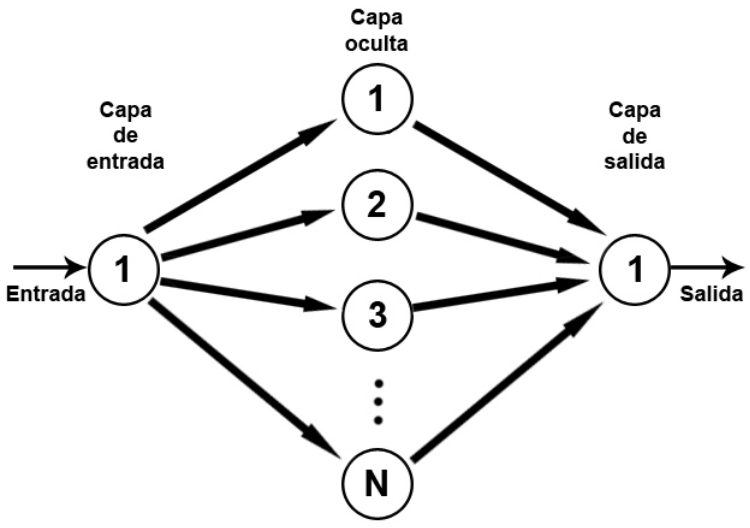

Figura 2-4: Arquitectura de la red neuronal artificial implementada.

ANN se ha utilizado en programación por demostración en trabajos relacionados con la navegación de los robots [55], donde se compara con la técnica GMM, y cada una se relaciona

con una estructura de memoria en el cerebro. En [56], la aproximación de PbD generaliza trayectorias demostradas del robot móvil, a un mapeo general entre las características visuales extraídas de una imagen omnidireccional y un movimiento apropiado del autómata. En ese caso, se realiza una comparación entre adaptación al contexto y las redes neuronales artificiales. Por su parte, Neto *et al.* [57] propone la programación por demostración de robots industriales utilizando un alto nivel de abstracción. Gestos y posturas de las manos son reconocidos utilizando una aproximación estadística y redes neuronales artificiales.

Además, ANN se ha utilizado como base para otros algoritmos. En [58], se propone una red reguladora de genes (GRN por sus siglas en inglés) basada en redes neuronales recurrentes (RNN pos sus siglas en inglés), como mecanismo de control robusto y confiable de robots. Se utiliza aprendizaje por imitación para adquirir secuencias de datos de comportamiento del robot, y la técnica propuesta para inferir controladores automáticamente.

2.5. Modelos ocultos de Markov

Los modelos ocultos de Markov son procesos estocásticos que toman una serie de datos en el tiempo como entrada, y como salida presenta la probabilidad de que los datos sean generados por el modelo [2, 59, 60]. HMM se divide en dos tipos: modelos ocultos de Markov discretos (DHMM por sus siglas en inglés) que se utiliza para la codificación de vectores discretos, y modelos ocultos de Markov continuos (CHMM por sus siglas en inglés), para tratar secuencias de vectores multidimensionales continuos.

Modelos ocultos de Markov discretos

Los DHMMs consisten de un conjunto finito de estados $\mathbf{Q} = \{q_1, ..., q_N\}$, un conjunto finito de etiquetas de salida $\mathbf{O} = \{o_1, ..., o_M\}$, una matriz de probabilidad de transición entre estados $\mathbf{A} = \{a_{ij}\}$, una matriz de probabilidad de salida $\mathbf{B} = \{b_{ij}\}$, y un vector de distribución inicial $\pi = \{\pi_i\}$. Las transiciones entre estados se desarrollan de forma probabilística y algunas etiquetas $\mathbf{o}_i$ se generan como salida durante la transición, como se observa en la Figura **2-5**.

Modelos ocultos de Markov continuos

A diferencia de DHMM donde las salidas de transición del proceso son etiquetas discretas, CHMM presenta como salidas vectores multidimensionales continuos (Figura **2-6**). En CHMMs, la matriz de probabilidad de salida $\mathbf{B}$ se convierte en una función de densidad de probabilidad. Comúnmente, la función de densidad se aproxima mediante combinación lineal de funciones gaussianas, como se observa en la Ecuación 2-14.

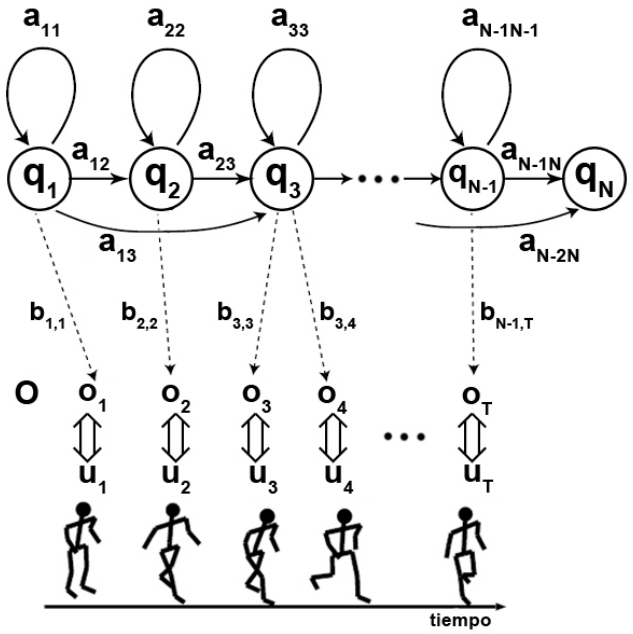

Figura 2-5: Codificación discreta con modelos ocultos de Markov tipo izquierda a derecha. Tomado de [2]

$$P_i(\mathbf{o}) = \sum_{j=1}^{m} c_{ij} N_{ij}\left(\mathbf{o}; \mathbf{\Sigma}, \mu\right), \tag{2-14}$$

donde $P_i(\mathbf{o})$ es la función de densidad de probabilidad para el vector continuo de salida $\mathbf{o}$ en el nodo de estado i, m es el número de funciones de mezclas gaussianas, y c_{ij} es el coeficiente de mezcla. $N_{ij}\left(\mathbf{o}; \mathbf{\Sigma}, \mu\right)$ es la función gaussiana, como se muestra en la Ecuación 2-15.

$$N_{ij}\left(\mathbf{o}; \mathbf{\Sigma}, \mu\right) = \frac{1}{\sqrt{(2\pi)^D |\Sigma_{ij}|}} \exp(-\frac{1}{2}(\mathbf{o} - \mu_{ij})^T \Sigma_i^{-1}(\mathbf{o} - \mu_{ij})), \tag{2-15}$$

donde $\mathbf{\Sigma}$ es la matriz de covarianza, μ es el vector de medias y D es el número de dimensiones del vector continuo $\mathbf{o}$. Los parámetros se calculan utilizando el algoritmo Baum-Welch.

Cada vector de medias de la función gaussiana se considera una representación importante del movimiento observado. Por lo tanto, se redefine el elemento de movimiento u como se muestra en la Ecuación 2-16. El número de elementos de movimiento es igual al número de componentes de mezclas gaussianas.

$$u_i = \{\Sigma_i, \mu_i\}. \tag{2-16}$$

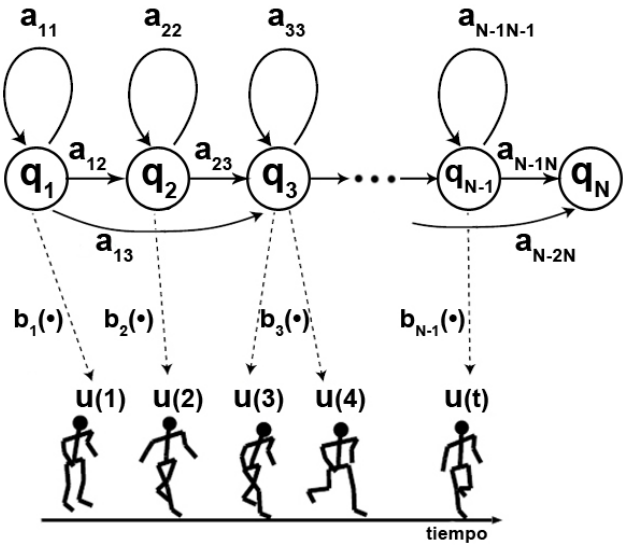

Figura 2-6: Codificación continua con modelos ocultos de Markov tipo izquierda a derecha. Tomado de [2]

Modelos más robustos pueden conseguirse utilizando HMM para la codificación, mientras que la reproducción se realiza mediante GMR, permitiendo extraer redundancias a partir de múltiples demostraciones y reproducción de la dinámica de los movimientos observados. En [61] se prueba este concepto con el robot humanoide iCub, adquiriendo los movimientos de baile de ambas manos, realizando movimientos cíclicos y cruzados. Además, se utiliza un brazo robótico, al cual se enseña a golpear una esfera utilizando una raqueta de tenis de mesa. Finalmente, se utiliza otro robot humanoide sosteniendo una cuchara y aprendiendo a alimentar a otro robot. Los resultados muestran la capacidad de la técnica que proponen, para manejar varias restricciones simultáneamente.

Otra forma de generalización luego de la codificación con HMM, utiliza puntos clave que en las trayectorias de entrenamiento de posición y velocidad. Esta aproximación tiene la ventaja de que puntos clave de todas las demostraciones se utilizan para la generalización a una nueva trayectoria [62].

En aprendizaje condicionado, [63] presenta la tarea de servir líquido de una botella en un vaso. No sólo el perfil de fuerza debe aprenderse, sino que el movimiento está condicionado a la cantidad de líquido en la botella. El robot es tele-operado utilizando un dispositivo háptico, y las demostraciones son codificadas utilizando modelos ocultos de Markov paramétricos (PHMM por sus siglas en inglés). La codificación encapsula la relación entre los parámetros de la tarea y el perfil de fuerza/torque. Los resultados se obtuvieron tanto en simulación como de forma experimental, donde el desempeño fue mayor al conseguido utilizando HMM convencional, ya que requiere menos entrenamiento, presenta una codificación más compacta,

y presenta mejores capacidades de generalización.

2.6. Discusión

En este capítulo se presentó una taxonomía de las técnicas de aprendizaje por imitación, además de la teoría detrás de las técnicas que se comparan en el Capítulo 3. Se habló de algunas ventajas y desventajas de cada una de ellas, y se relacionan trabajos en que se han aumentado sus capacidades al modificar el algoritmo clásico o cambiando la técnica de regresión utilizada para la reconstrucción de la señal.

3 Análisis de las técnicas de aprendizaje por imitación

Para la comparación de las técnicas, se utilizó siempre una trayectoria compuesta por 6 grados de libertad y obtenida como se indica en la sección 4.2 (Figura **3-1**). La implementación se realizó en MatLab, utilizando las librerías diseñadas de forma específica para cada técnica. En este capítulo se presentan los resultados obtenidos mediante tres métricas, luego de codificar las trayectorias descritas con las técnicas de aprendizaje por imitación seleccionadas.

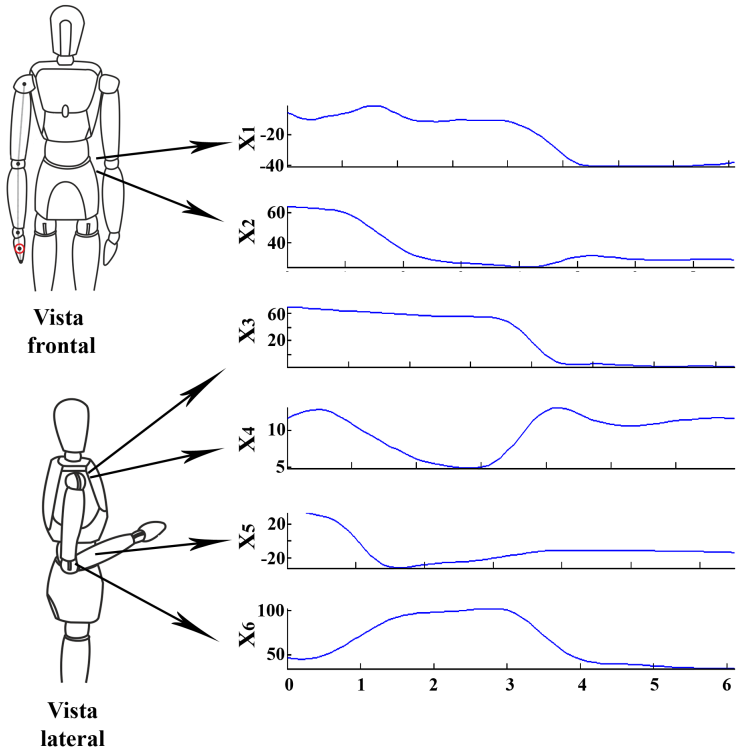

Figura 3-1: Trayectorias de los seis ángulos utilizadas para comparar las técnicas.

Como se observa en la Figura **3-1**, los dos primeros ángulos se relacionan con balanceo del torso, mientras que los cuatro siguientes dependen de las articulaciones del hombro derecho, el codo derecho y la mano derecha.

3.1. Implementación de DMP

La librería de DMP compartida por Ude [64], provee los algoritmos básicos que pueden utilizarse para estimar las primitivas de movimiento dinámico a partir de una o múltiples demostraciones. La codificación se realiza como se describe en el Algoritmo 1.

Algoritmo 1 Algoritmo implementado para Primitivas de Movimiento Dinámico

1: **procedimiento** DMP
2: Ajuste de los datos de entrenamiento.
3: Definición de los parámetros de la técnica: $\alpha_x = 2, \alpha_z = 20, \beta_z = 5$.
4: Definición del número de funciones de base radial: $N = \{10, 15, 20\}$.
5: Conversión de las trayectorias de entrenamiento a DMP resolviendo el problema de regresión lineal planteado y hallando los valores de los pesos w.
6: Reproducción a partir de la DMP reemplazando los valores obtenidos en el sistema de ecuaciones
7: **end procedimiento**

De acuerdo al Algoritmo 1, los parámetros α_x, α_z y β_z definen el desempeño final de la técnica. Valores apropiados se encuentran en la literatura ($\alpha_z = 4\beta_z$), los cuales se ajustan para obtener una reconstrucción de la señal fiel a la trayectoria de entrada.

Se comprobó el funcionamiento de la técnica al cambiar el número de funciones de base radial N, como se presenta en la Figura **3-2**. Se seleccionaron los valores $N = \{10, 15, 20\}$, ya que números más bajos presentan una reconstrucción alejada de la señal original, mientras que valores más altos no presentan variaciones considerables. Se proponen estos tres valores para determinar el cambio en el desempeño, un rango donde la fidelidad de la señal reproducida es alto.

En la Figura **3-2**, se observa en negro la señal original y en rojo la trayectoria reconstruida. Los errores entre ambas disminuye al aumentar el número de funciones de base radial, lo cual se nota particularmente en los puntos donde las curvas son pronunciadas, por ejemplo, a los dos segundos en los ángulos X_5 y X_6.

3.1.1. Tiempos de codificación y reconstrucción

Se define como el tiempo que toma a cada técnica codificar la señal de entrada en términos de sus parámetros específicos, además del tiempo que toma reconstruir la señal a partir de dichos parámetros.

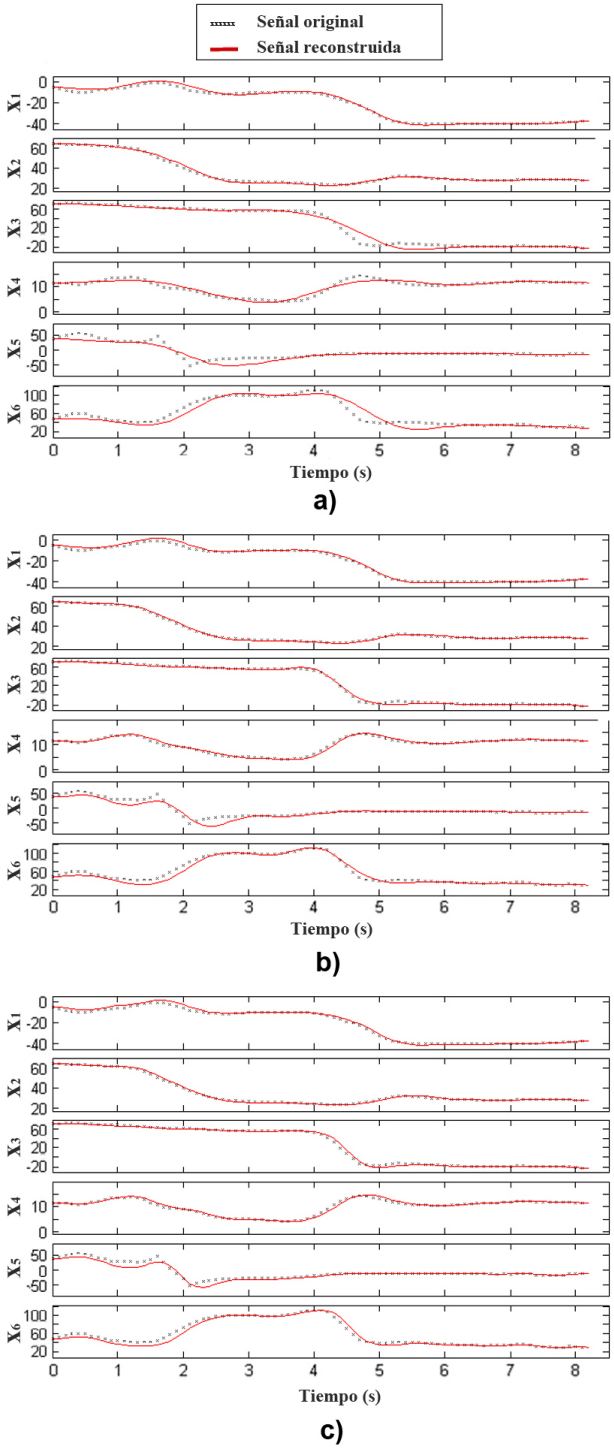

Figura 3-2: Reconstrucción de las señales con DMP. En puntos la señal original, en rojo la señal reconstruida. a) N=10. b) N=15. c) N=20.

Tabla 3-1: Tiempos de codificación y reconstrucción para DMP.

N	**Codificación** $t_{promedio}$ **(ms)**	**Reconstrucción** $t_{promedio}$ **(ms)**
10	$6,39$	$2,11$
15	$6,51$	$2,17$
20	$6,65$	$2,23$

Como se observa en la Tabla **3-1**, tanto el tiempo de codificación como de reconstrucción son directamente proporcionales al número de estados elegidos para codificar la señal. Aún así, al aumentar en 5 cada vez el número de estados, el cambio en los tiempos es menor a $0,15ms$, lo cual es un cambio pequeño al compararlo con los resultados de las tablas **3-3**-**3-7**, presentadas en las secciones siguientes.

3.1.2. Correlación cruzada y error relativo del valor final

La correlación cruzada es un método estándar para estimar el grado de correlación entre dos secuencias. Dadas dos series $x(i)$ y $y(i)$ donde $i = 0, 1, 2, ...N-1$, la correlación cruzada r en el retardo d está definido como se muestra en la Ecuación 3-1.

$$r_d = \frac{\sum_i [(x(i) - \bar{x})(y(i-d) - \bar{y})]}{\sqrt{\sum_i (x(i) - \bar{x})^2} \sqrt{\sum_i (y(i-d) - \bar{y})^2}}, \tag{3-1}$$

donde $\bar{x}$ y $\bar{y}$ son las medias de las series correspondientes. El coeficiente r, es una medida del tamaño y la dirección de la relación lineal entre las variables x y y. Utilizando la opción normalizar en Matlab, obtenemos una serie de salida conteniendo los valores r en el intervalo $[0, 1]$. Obteniendo el valor máximo de esta secuencia, podemos comparar las técnicas de aprendizaje, teniendo en cuenta que un valor cercano a 1 indica una correlación más alta entre las señales de entrenamiento y de reproducción (Tabla **3-2**).

El error relativo indica qué tan bien se ajusta una medida con respecto al valor que se considera real (Ecuación 3-2). En este caso, el valor real se define como el valor final de la señal de entrada, o valor objetivo. La comparación se realiza con respecto al valor final de la señal reconstruida (Tabla **3-2**). Usualmente, el error relativo se presenta como un porcentaje.

$$error\_relativo(\,\%) = \left(\frac{Valor_{real} - Valor_{medido}}{Valor_{real}} \right) * 100 \tag{3-2}$$

Tabla 3-2: Correlación cruzada y error relativo para DMP.

N	Correlación cruzada promedio	Error relativo promedio (%)
10	$0,9916$	$2,45$
15	$0,9974$	$3,28$
20	$0,9983$	$1,44$

En la Tabla **3-2**, se presenta un aumento de la correlación cruzada al aumentar el número de estados N. Al haber un mayor número de funciones de base radial, la codificación guarda más información que resulta importante al momento de reconstruir las curvas de la señal, aumentando así dicho índice. Por otra parte, el valor final de la reproducción depende de la ecuación diferencial, la cual converge al objetivo, pero tiene en cuenta la velocidad y aceleración del movimiento entrenado. Para no crear discontinuidades, el cambio se hace de forma suave, y puede no haberse alcanzado el valor de convergencia al finalizar la reconstrucción, lo que produce errores sin un patrón específico.

3.2. Implementación de GMM y GMR

La implementación se realizó utilizando la librería de Calinon [35, 65], con el Algoritmo 2.

Algoritmo 2 Algoritmo implementado para Mezcla de Modelos Gaussianos

1: **procedimiento** GMM
2: Ajuste de los datos de entrenamiento.
3: Definición del número de componentes Gaussianos: $N = \{5, 7, 9\}$.
4: Inicialización de la ubicación de las funciones gaussianas utilizando la técnica k-medias.
5: Entrenamiento de la GMM utilizando el algoritmo de Esperanza-Maximización (EM) mediante múltiples iteraciones.
6: Reproducción de la trayectoria utilizando GMR.
7: **end procedimiento**

Como muestra el Algoritmo 2, una de las técnicas más utilizadas para inicializar la posición de las distribuciones gaussianas, es el algoritmo k-medias. Dado que se basa en cálculo de distancias, es computacionalmente más rápida que otras técnicas. Además de esto, el entrenamiento se basa en el algoritmo Esperanza-Maximización, conocido por su velocidad, simpleza, robustez y facilidad de implementación.

Se comprobó el funcionamiento de la técnica cambiando el número de mezclas gaussianas N, como se presenta en la Figura **3-3**. Se seleccionaron los valores $N = \{5, 7, 9\}$, debido

a que valores más bajos alejan la reconstrucción de la señal original, mientras que valores más altos producen matrices grandes que se indeterminan fácilmente al ser invertidas. Se proponen estos tres valores para determinar el cambio en el desempeño, un rango donde la fidelidad de la señal reproducida es alto.

En la Figura **3-3**, la primera columna corresponde a la señal de entrada, la segunda a la ubicación final de las distribuciones gaussianas, y la última a la trayectoria reconstruida (línea sólida roja) junto al nivel de confianza (sombra roja). La ubicación de las gaussianas nos permite inferir la ubicación de los datos, donde la ejecución con nueve funciones presenta unas ubicaciones más organizadas que permiten entrever la secuencia que tendrá la señal reproducida. En esta técnica, el aumento del número de funciones mejora el desempeño, pero puede llevar a singularidades, donde estimar las matrices de covarianza se vuelve difícil y el algoritmo empieza a divergir.

3.2.1. Tiempos de codificación y reconstrucción

Tabla 3-3: Tiempos de codificación y reconstrucción para GMM.

N	**Codificación** $t_{promedio}$ **(ms)**	**Reconstrucción** $t_{promedio}$ **(ms)**
5	$6,62$	$0,16$
7	$11,54$	$0,36$
9	$16,78$	$0,38$

Como se observa en la Tabla **3-3**, el tiempo de codificación es directamente proporcional al número de estados elegidos para codificar la señal. Un aumento de tan solo 2 gaussianas cada vez, produce una alza en el tiempo de alrededor de $5ms$. Por otro lado, aunque el tiempo de reconstrucción también aumenta, este valor no afectaría significativamente el tiempo total de ejecución del algoritmo.

3.2.2. Correlación cruzada y error relativo del valor final

Tabla 3-4: Correlación cruzada y error relativo para GMM.

N	**Correlación cruzada promedio**	**Error relativo promedio (%)**
5	$0,9966$	$3,16$
7	$0,9983$	$2,29$
9	$0,9985$	$1,47$

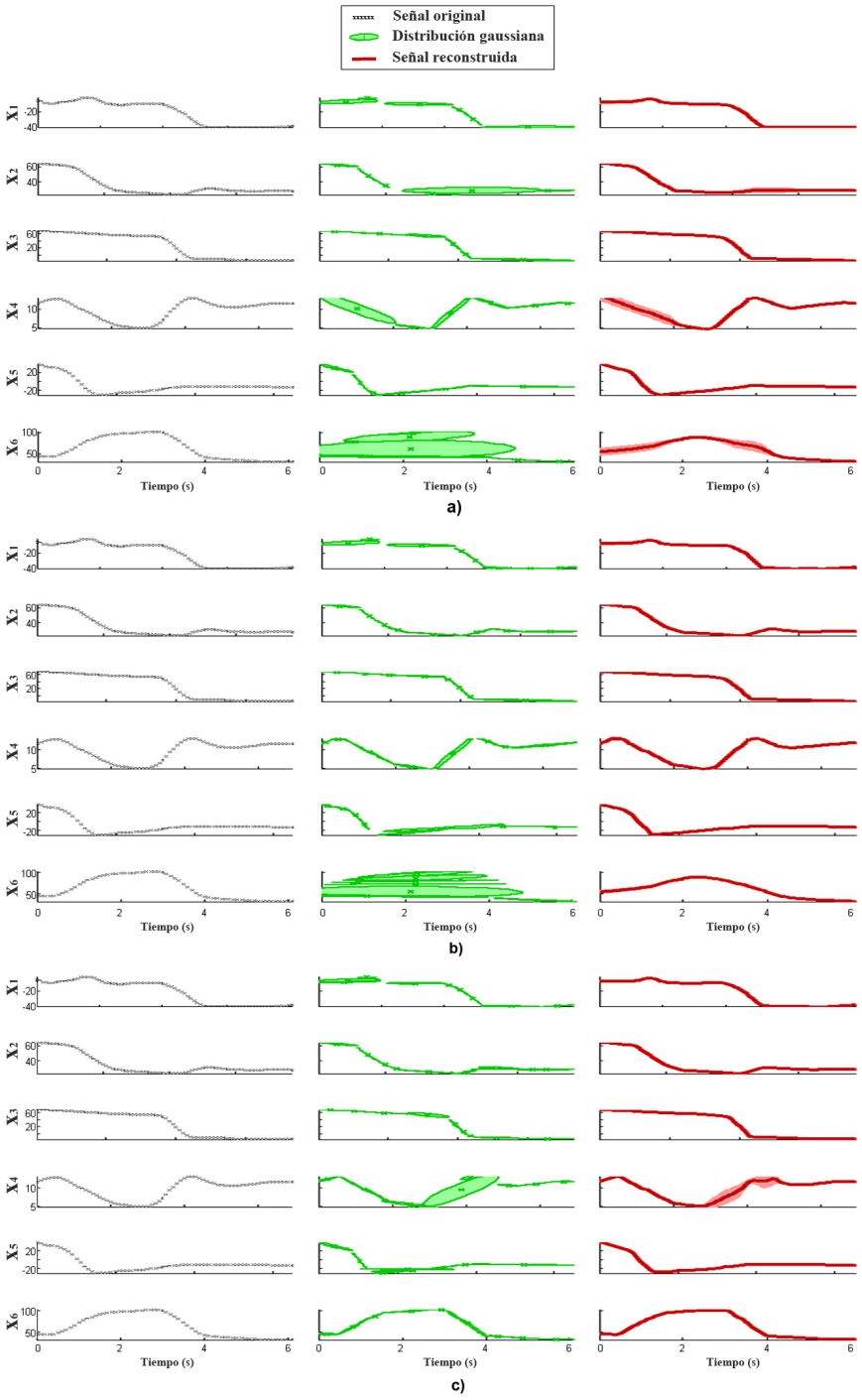

Figura 3-3: Codificación con GMM y reconstrucción de las señales con GMR. En puntos la señal original, en verde las gaussianas generadas y en rojo la señal reconstruida. a) N=6. b) N=8. c) N=9.

En la Tabla **3-4** se observa una relación directamente proporcional entre la correlación cruzada y el número de gaussianas, y una relación inversamente proporcional entre el error del valor final y el número de estados. El aumento en el número de gaussianas N, mejora el desempeño global de esta técnica.

3.3. Implementación de ANN

La implementación se realizó utilizando la librería de redes neuronales de Matlab [66], con el Algoritmo 3.

Algoritmo 3 Algoritmo implementado para Redes Neuronales Artificiales

1: **procedimiento** ANN
2: Ajuste de los datos de entrenamiento.
3: Definición del número de neuronas en la capa oculta: $N = \{5, 10, 15, 20\}$.
4: Entrenamiento de los pesos de la red mediante múltiples iteraciones (algoritmo Levenberg-Marquardt).
5: Reproducción de la señal utilizando la red entrenada.
6: **end procedimiento**

Como se observa en el Algoritmo 3, el parámetro fundamental para la técnica ANN es el número de neuronas en la capa oculta. El entrenamiento se realiza con Levenberg-Marquardt, uno de los algoritmos de optimización más utilizados, que hereda la velocidad del algoritmo Gauss-Newton y la estabilidad del método del descenso más pronunciado [67].

Se varió la cantidad de neuronas N de la capa oculta, como se presenta en la Figura **3-4**. Se seleccionaron los valores $N = \{5, 10, 15, 20\}$, ya que valores mayores no producen cambios perceptibles. Se proponen estos cuatro valores para determinar el cambio en los tiempos de codificación y reconstrucción, al utilizar un rango amplio de valores utilizados en trabajos relacionados.

En la Figura **3-4**, se observa la principal ventaja de utilizar ANN, la fidelidad entre la señal de entrada y la trayectoria reconstruida. Aún con sólo cinco neuronas en la capa oculta, la correlación entre ambas señales es superior a la obtenida con las demás técnicas.

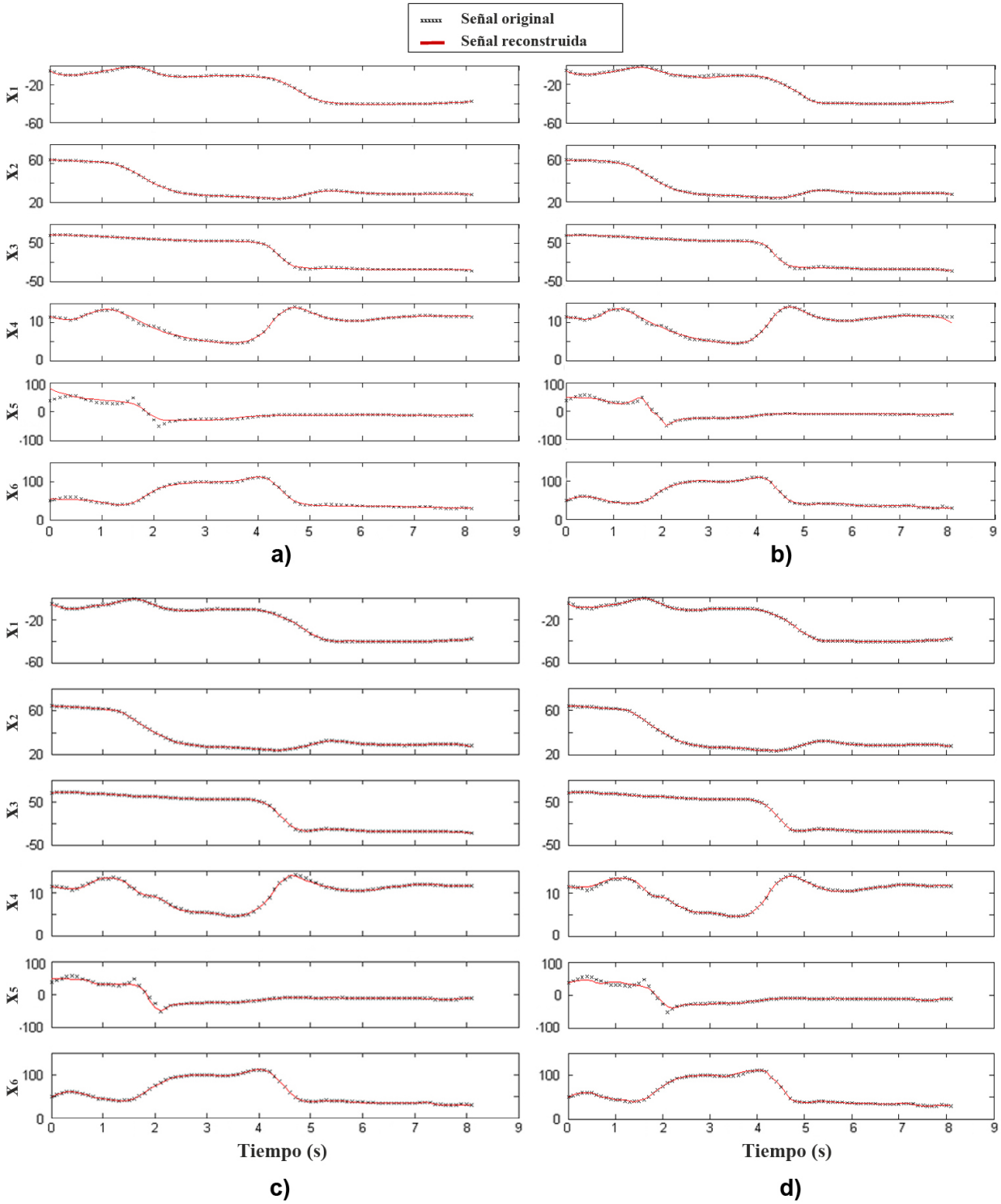

Figura 3-4: Codificación y reconstrucción de las señales con ANN. En puntos la señal original y en rojo la señal reconstruida. a) N=5. b) N=10. c) N=15. d) N=20.

3.3.1. Tiempos de codificación y reconstrucción

Tabla 3-5: Tiempos de codificación y reconstrucción para ANN.

N	Codificación $t_{promedio}$ (ms)	Reconstrucción $t_{promedio}$ (ms)
5	$633,8$	$35,1$
10	$528,0$	$34,3$
15	$619,4$	$35,3$
20	$641,6$	$35,1$

Como se observa en la Tabla **3-5**, aunque el tiempo no aumenta proporcionalmente con el número de neuronas en la capa oculta, la codificación de la señal toma más de medio segundo, mucho más que lo que presentan las técnicas vistas en las secciones anteriores. Por su parte, el tiempo de reconstrucción permanece constante.

3.3.2. Correlación cruzada y error relativo del valor final

Tabla 3-6: Correlación cruzada y error relativo para ANN.

N	Correlación cruzada promedio	Error relativo promedio (%)
5	$0,9979$	$2,91$
10	$0,9994$	$2,64$
15	$0,9987$	$1,34$
20	$0,9976$	$1,44$

En la Tabla **3-6**, no se observa un patrón en los índices de desempeño al aumentar el número de neuronas en la capa oculta. Esto se debe a que los valores iniciales de los pesos varían en cada ejecución, así como el número de iteraciones durante el entrenamiento. Esto da lugar a errores aletorios dentro de cierto rango, aún cuando la correlación cruzada se mantiene por encima de $0,997$.

3.4. Implementación de HMM

La implementación se realizó utilizando la librería escrita por Kevin Murphy en 1998, basado en el algoritmo descrito por Rabiner [68] (Algoritmo 4). El Algoritmo 4 presenta la implementación de HMM utilizada.

Algoritmo 4 Algoritmo implementado para Modelos Ocultos de Markov

1: **procedimiento** HMM
2: Ajuste de los datos de entrenamiento.
3: Definición de los parámetros: Número de mezclas Gaussianas $(M) = 3$, coeficiente de mezcla $(O) = 1$
4: Definición del número de estados: $Q = \{5, 10, 15, 20\}$.
5: Inicialización de la ubicación de las funciones Gaussianas utilizando la técnica k-medias.
6: Entrenamiento del HMM utilizando el algoritmo de Esperanza-Maximización (EM) mediante múltiples iteraciones.
7: Reproducción de la trayectoria utilizando splines a partir de los datos de salida del HMM.
8: **end procedimiento**

Se comprobó el funcionamiento de la técnica al variar el número de estados Q manteniendo fijo el número de mezclas gaussianas de salida $M = 3$, como se presenta en la Figura **3-5**. Se seleccionaron los valores $Q = \{5, 10, 15, 20\}$, ya que valores más bajos no presentan resultados aceptables y valores mayores no producen cambios perceptibles. Se proponen estos tres valores para determinar el cambio en el desempeño, un rango donde la fidelidad de la señal reproducida es alto.

En la Figura **3-5**, se observa que pocos estados producen una reconstrucción muy pobre de la señal original, perdiendo gran cantidad de información de las curvas. Se produce una mejora en la fidelidad de la reconstrucción a medida que se aumenta el número de estados Q, sin embargo, el desempeño general al compararla con las demás técnicas es bajo.

3.4.1. Tiempos de codificación y reconstrucción

Tabla 3-7: Tiempos de codificación y reconstrucción para HMM.

N	**Codificación** $t_{promedio}$ **(ms)**	**Reconstrucción** $t_{promedio}$ **(ms)**
5	83, 6	15, 0
10	163, 3	15, 6
15	491, 0	15, 2
20	1869, 9	15, 4

En la Tabla **3-7**, se observa un crecimiento exponencial en el tiempo de codificación, a la vez que el tiempo de reconstrucción se mantiene casi constante, mientras el número de estados

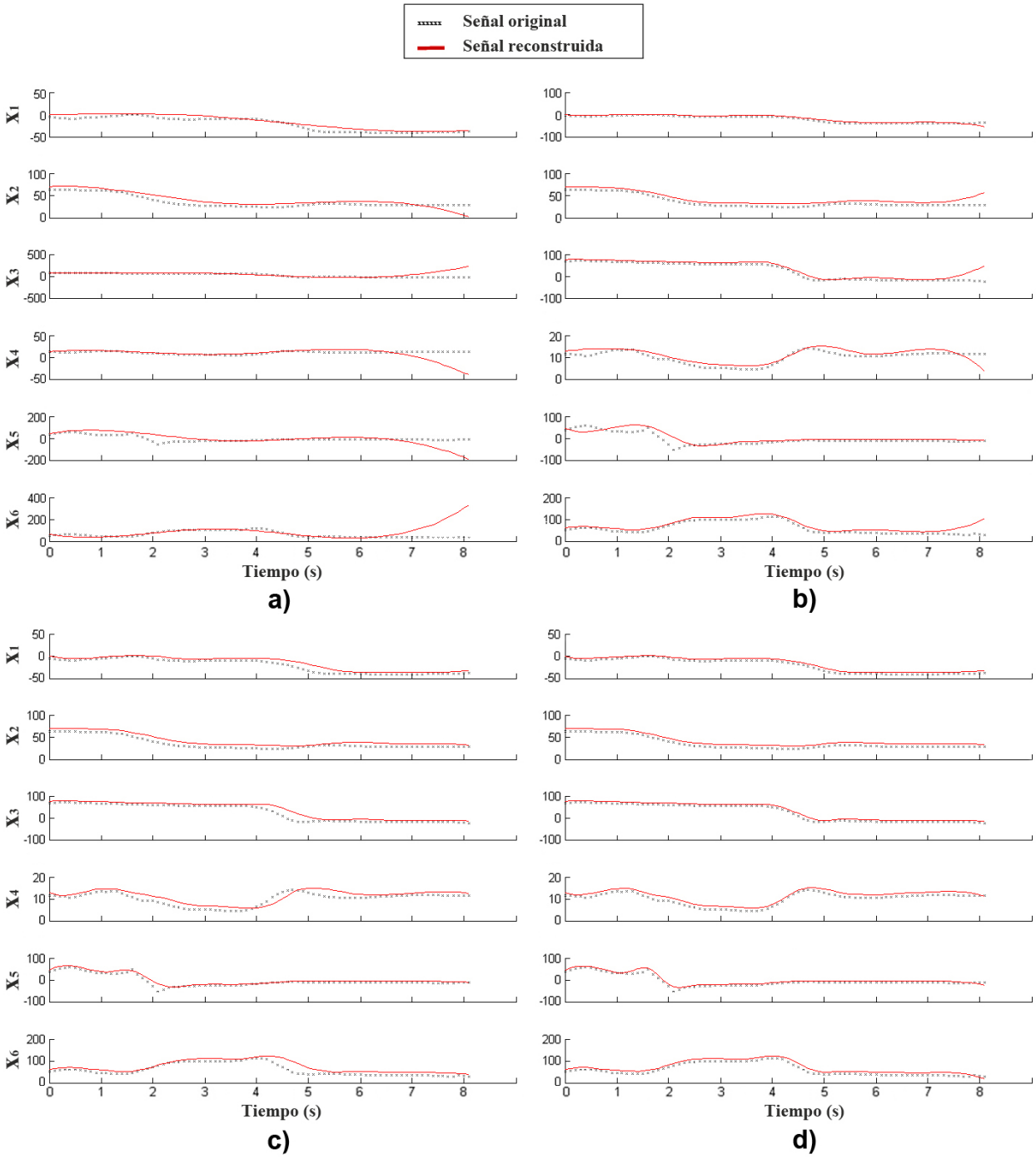

Figura 3-5: Codificación y reconstrucción de las señales con HMM. En puntos la señal original y en rojo la señal reconstruida. a) N=5. b) N=10. c) N=15. d) N=20.

N se incrementa. La tasa de aumento es tan alta, que con 15 estados, el tiempo total es menor al tiempo con la técnica ANN, pero al llegar a 20 estados, se ha superado su valor máximo de ANN en más de un segundo.

3.4.2. Correlación cruzada y error relativo del valor final

Tabla 3-8: Correlación cruzada y error relativo para HMM.

N	Correlación cruzada promedio	Error relativo promedio (%)
5	0,7212	>100
10	0,9536	>100
15	0,9793	20,86
20	0,9895	35,99

En la Tabla **3-8** se observa una relación directamente proporcional entre la correlación cruzada y el número de estados. Esta técnica presenta errores por encima de 20 %, los errores más grandes si se compara con las técnicas vistas en las secciones anteriores.

3.5. Discusión

Como se observa en las tablas **3-2**, **3-4**, **3-6** y **3-8**, en general, el aumento del número de estados incrementa la correlación cruzada y disminuye el error relativo del valor final. De las tablas **3-1**, **3-3**, **3-5** y **3-7**, con GMM y GMR con cinco estados, el tiempo de reconstrucción es menor al de las demás técnicas, pero al costo de un error superior al 3 %.

Teniendo en cuenta la información de las tablas de tiempo de codificación y reconstrucción, y las tablas de error, se decide utilizar el algoritmo DMP con $N = 20$, el cual presenta un tiempo menor a $9ms$, sumando el tiempo de codificación y reconstrucción, a la vez que mantiene un error menor al 1,5 %. En cuanto a los datos de correlación cruzada promedio, la mayoría de las técnicas presentan valores mayores a $0,99$, obteniendo un $0,9983$ con la técnica seleccionada.

4 Desarrollo del sistema de adquisición de datos

Tanto para la comparación de las técnicas (Capítulo 3), como para la creación de la base de datos para generalizar a nuevas trayectorias (Capítulo 5), es necesario determinar la ubicación de las articulaciones durante el entrenamiento. En este capítulo se presentan los cálculos necesarios para obtener los valores de los ángulos, a partir de la información de posición de las articulaciones. Además, se propone calcular la posición y orientación de los objetos que sirven como objetivo para posicionar la mano del robot.

4.1. Trabajos relacionados

Un factor importante para las técnicas de aprendizaje por imitación es el método de adquisición de los datos de la demostración. Como lo menciona Billard *et al.* [12], inicialmente se utilizaba teleoperación, donde el robot es manipulado vía joystick por un experto mientras obtiene la información de sus propios sensores. De esta forma, el mapeo estado/acción se adquiere de forma directa de la experiencia. La adquisición de los datos de entrenamiento fue reemplazada progresivamente por la enseñanza cinestésica, donde el robot no es controlado de forma activa pero sus articulaciones pasivas se estimulan realizando un movimiento deseado. Además de esto, guantes de datos así como otros sensores ubicados en el cuerpo del instructor (exoesqueletos), adquieren los datos de forma directa, realizando captura de movimientos, aceleraciones o fuerzas [10]. Interfaces como visión y sensores de distancia basados en láser, son ejemplos de observación externa. Dichos sensores pueden o no estar localizados en el cuerpo del robot, por lo que no existe un mapeo directo entre los estados/acciones observadas y los estados/acciones en el cuerpo del autómata [6].

Se ha difundido de manera amplia el aprendizaje robótico por imitación utilizando imágenes como datos de entrada [43, 7, 8, 5]. Esto debido al hecho de que muchos seres vivos, entre ellos el ser humano, se apoyan en la visión para resolver un extenso conjunto de tareas. Al referirse a imágenes como datos de entrada, podemos encontrar imágenes a color, las cuales proveen información plana de la escena, e imágenes 3D, o de profundidad. Desde el punto de vista de la ingeniería, las cámaras de video 2D presentan bajos costos, no son invasivas y proveen gran cantidad de información. Son especialmente útiles cuando se combinan con información del entorno o datos de profundidad (imágenes 3D) [4]. Un módulo de

visión provee datos relacionados a forma y posición de los objetos. Además, permite realizar seguimiento de éstos y de las partes del cuerpo del instructor relevantes para la tarea [11].

En el pasado los sensores que proveían imágenes 3D eran de difícil adquisición dada su poca disponibilidad y altos precios. En Noviembre de 2010, Microsoft lanza el sensor Kinect con la capacidad de producir flujos de imágenes RGB y de profundidad a un precio mucho menor a los sensores de rango tradicionales. Las imágenes de profundidad se obtienen de una cámara de tiempo de vuelo, que mide la distancia de cualquier punto desde el sensor. Una matriz de luz cercana a la infrarroja se proyecta a través de la escena y el sensor calcula el tiempo de vuelo que le toma a cada punto de la matriz de luz generada reflejarse desde el objeto [69].

Se pueden encontrar desarrollos en diversas áreas utilizando el sensor Kinect, entre éstos seguimiento en 3D de las articulaciones del esqueleto humano, o partes de éste [69, 70, 71]. Se ha utilizado para capturar movimientos humanos con los cuales determinar sus comportamientos e intenciones [72, 73]. También se ha usado para reconocer gestos con la mano [74, 75], como sensor de contacto [76] y como sensor de navegación y prevención de obstáculos [77]. Además ha permitido la creación de interfaces de interacción humano-máquina más naturales [78], a la vez que se presenta como una potencial tecnología para la educación interactiva [79].

En el área de aprendizaje robótico por imitación utilizando el sensor Kinect de Microsoft, se pueden citar los trabajos de León *et al.* [7, 8]. En éstos desarrollos, se enseña a un brazo robótico de 6 grados de libertad a tomar objetos y colocarlos en una nueva ubicación. Para esto se obtiene la posición 3D de la mano del instructor, así como de los objetos del entorno. Previamente se ha calibrado el área donde se realiza la demostración para incluir todos los movimientos del brazo. La secuencia de movimientos de la realización de la tarea se procesa para generar un mapa estado/acción del robot. Finalmente, se utiliza un reforzamiento del aprendizaje mediante realimentación del instructor al robot mientras éste imita la tarea.

Varios proyectos con librerías de uso libre se encuentran disponibles actualmente, como lo reporta [78]. Se decidió utilizar OpenNI ya que fue establecido como un estándar de la industria. Junto a la infraestructura lanzada por OpenNI se encuentra un software intermedio denominado NITE cuyas librerías están disponibles para aplicaciones escritas en C, C++ y C#. NITE provee algunas funcionalidades entre las que se encuentran:

- Detección del esqueleto y seguimiento individual de las posiciones de las articulaciones.
- Acceso a datos de video y profundidad.
- Ejemplos de aplicaciones de interfaces controladas por gestos, segmentación de usuarios, seguimiento de puntos, seguimiento del esqueleto, entre otros.

Al utilizar la infraestructura OpenNI como driver del Kinect, se obtienen imágenes RGB de 640×480 píxeles y 2^8 bits de resolución, e imágenes de profundidad de 640×480 píxeles con resolución de 2^{11} bits a 30 cuadros por segundo.

4.2. Obtención de los datos de las articulaciones

Utilizando la librería NITE se creó un programa que reconoce el esqueleto del usuario y accede a los datos (x, y, z) de cada articulación en coordenadas del mundo; cada dato correspondiente a un eje tiene como unidad los milímetros. El valor x presenta la ubicación en el eje horizontal, perpendicular al eje de la cámara; y corresponde a la elevación del punto en forma vertical; z es la profundidad medida desde el sensor.

Las articulaciones usadas para calcular los ángulos requeridos fueron (Figura **4-1**): cadera derecha e izquierda, cuello, hombro derecho, codo derecho y mano derecha. Las articulaciones del brazo izquierdo no se utilizan, ya que la demostración de la tarea consiste en llevar la mano derecha hacia el objetivo.

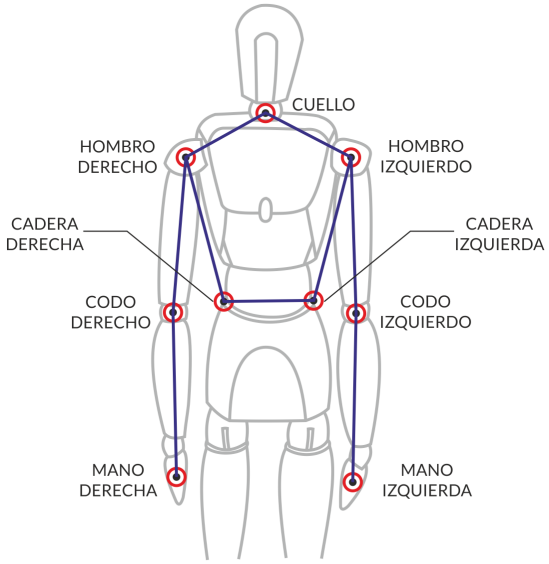

Figura 4-1: Articulaciones del cuerpo utilizadas para el cálculo de los ángulos.

Los ángulos calculados fueron:

- Guiñada del torso (yaw)
- Cabeceo del torso (pitch).
- Cabeceo del hombro.

- Balanceo del hombro.
- Guiñada del hombro.
- Codo.

Estos ángulos se seleccionaron teniendo en cuenta que exista un ángulo análogo en el robot, y que tenga un papel relevante en la tarea de llevar la mano derecha a la posición objetivo. En el caso específico de los ángulos del torso, se tienen en cuenta dado que modifican el alcance final de la mano. A continuación se describen los ángulos y cálculos requeridos.

Guiñada del torso: Ángulo de giro con respecto al eje y, dado por el vector de gravedad (Figura **4-2**). Se utilizan los valores x y z de las articulaciones de cadera izquierda y derecha (Ecuación 4-1).

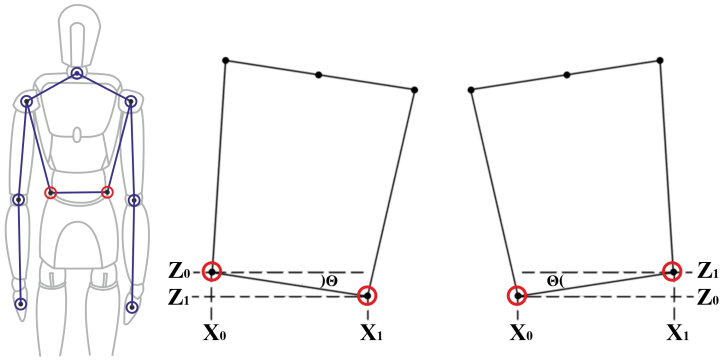

Figura 4-2: Guiñada del torso.

$$\tan\theta = \frac{z_1 - z_0}{x_1 - x_0}. \tag{4-1}$$

El ángulo obtenido es cero cuando el torso se encuentra completamente de frente a la cámara. Valores positivos o negativos indican un giro de la cadera en un sentido o en el otro. El rango está comprendido entre -50 y 50 grados, según las restricciones del robot.

Cabeceo del torso: Movimiento adelante-atrás (Figura **4-3**). Se utilizan los datos de profundidad z y de altura y del cuello y una de las articulaciones de la cadera (Ecuación 4-2).

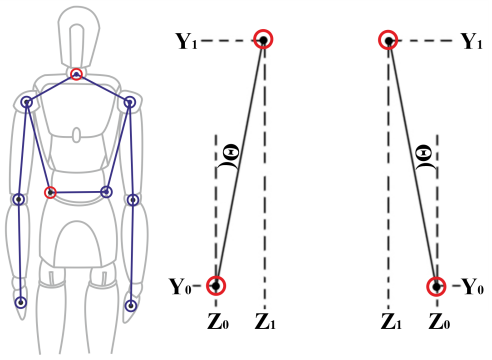

Figura 4-3: Cabeceo del torso.

$$\tan\theta = \frac{z_1 - z_0}{y_1 - y_0}. \tag{4-2}$$

El ángulo obtenido es cero cuando el torso no presenta inclinación adelante o atrás. Valores positivos o negativos indican una inclinación del torso en un sentido o en el otro. El rango está comprendido entre -10 grados hacia atrás y 70 grados hacia adelante, según las restricciones del robot.

Cabeceo del hombro: Movimiento adelante-atrás del brazo (Figura **4-4**). Se calcula con los valores y y z de las articulaciones de hombro y codo 4-3.

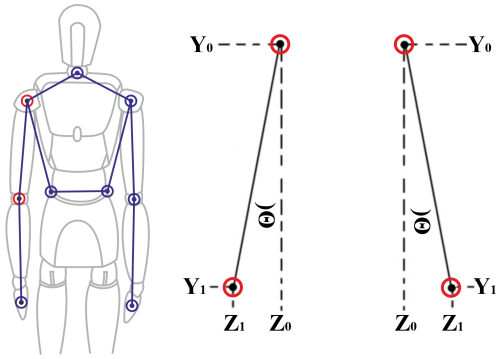

Figura 4-4: Cabeceo del hombro.

$$\tan\theta = \frac{z_1 - z_0}{y_1 - y_0}. \tag{4-3}$$

El ángulo obtenido es cero cuando el codo se encuentra paralelo al torso. Valores positivos o negativos indican un movimiento adelante o atrás del codo con respecto al hombro. El

rango está comprendido entre -95 grados hacia adelante y 10 grados hacia atrás, según las restricciones del robot.

Balanceo del hombro: Movimiento de aducción-abducción del brazo (Figura **4-5**). Se calcula con los datos x y y de las articulaciones de hombro y codo (Ecuación 4-4).

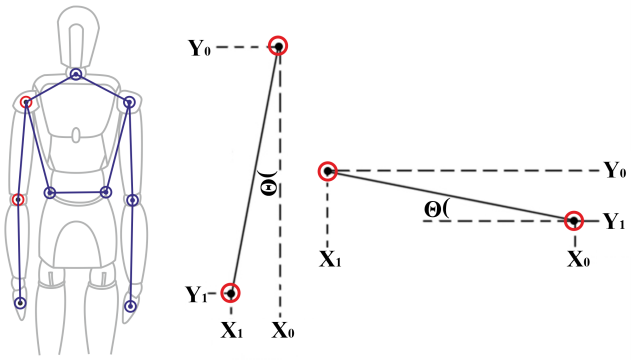

Figura 4-5: Balanceo del hombro.

$$\tan\theta = \frac{x_1 - x_0}{y_1 - y_0}. \tag{4-4}$$

El ángulo obtenido es cero cuando el codo se encuentra pegado al torso. Los valores obtenidos indican un movimiento del codo separándose de forma lateral al cuerpo. El rango para la aplicación definida está comprendido entre 0 y 90 grados, según las restricciones del robot.

Guiñada del hombro: Giro del brazo sobre el eje definido entre la articulación del hombro y el codo (Figura **4-6**). Se calcula con los valores x y z de las articulaciones de codo y mano (Ecuación 4-5).

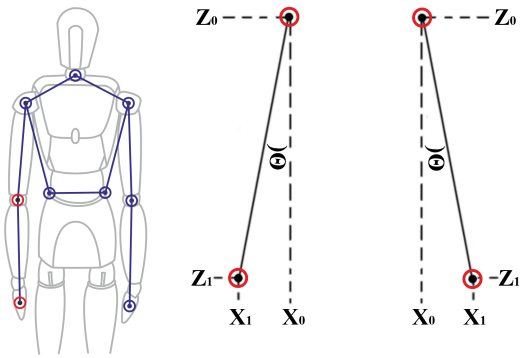

Figura 4-6: Guiñada del hombro.

$$\tan\theta = \frac{x_1 - x_0}{z_1 - z_0}. \tag{4-5}$$

El ángulo obtenido es cero cuando la articulación de la mano se encuentra alineada en el eje x con respecto a la articulación del codo. Valores positivos o negativos indican un giro a la izquierda o la derecha de la mano con respecto al codo. El rango está comprendido entre -37 grados alejándose la mano del torso y 80 grados acercándose, según las restricciones del robot.

Codo: Se calcula utilizando la ley de cosenos con vértices (y, z) en las articulaciones de hombro, codo y mano (Figura **4-7**).

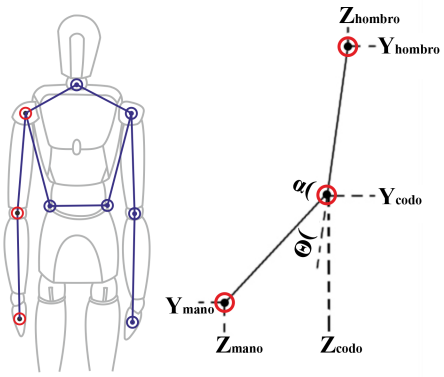

Figura 4-7: Ángulo del codo.

$$\theta = 180° - \cos^{-1}\left(\frac{a^2 + b^2 - c^2}{2ab}\right), \tag{4-6}$$

donde

$$a = \sqrt{(z_{hombro} - z_{codo})^2 + (y_{hombro} - y_{codo})^2},$$

$$b = \sqrt{(z_{codo} - z_{mano})^2 + (y_{codo} - y_{mano})^2} \qquad y$$

$$c = \sqrt{(z_{hombro} - z_{mano})^2 + (y_{hombro} - y_{mano})^2}.$$

El ángulo obtenido es cero cuando el codo se encuentra extendido completamente. El rango está comprendido entre 15.5 y 106 grados, según las restricciones del robot.

4.3. Segmentación en color (posición del objeto)

Como núcleo de la etapa de reconocimiento se utilizó el sensor Kinect de Microsoft, el cual consta de una cámara RGB con resolución 640 × 480 y una cámara de profundidad basada en infrarrojo, produciendo imágenes como las mostradas en las Figuras **4-8**a y **4-8**b, respectivamente.

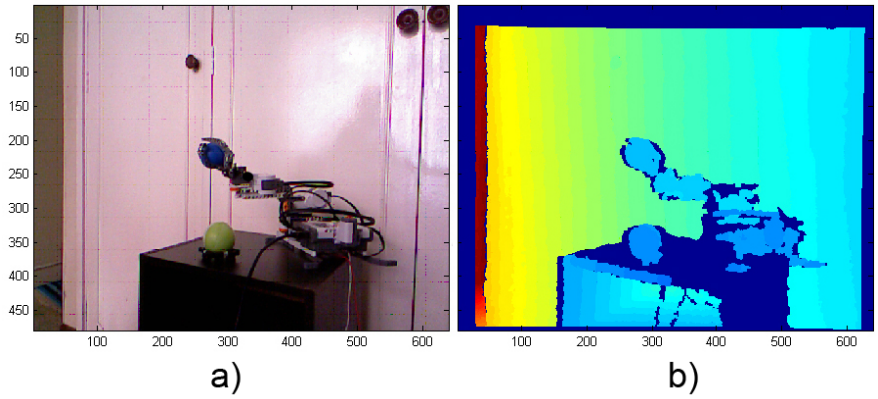

Figura 4-8: Imágenes capturas con el sensor Kinect. a) Imagen RGB. b) Imagen de profundidad.

Para realizar la segmentación, se convirtió la imagen RGB obtenida a un mapa de colores HSV (Figura **4-9**), donde se continuó trabajando con el canal de tono (*hue*), debido a que es menos sensible a los cambios en iluminación. Para esto, se utilizan las ecuaciones 4-7 a 4-11 en cada píxel R, G y B de la imagen [80].

$$\begin{aligned} R' &= R/255 \\ G' &= G/255 \\ B' &= B/255 \end{aligned} \tag{4-7}$$

$$\begin{aligned} C_{max} &= max(R', G', B') \\ C_{min} &= min(R', G', B') \\ \Delta &= C_{max} - C_{min} \end{aligned} \tag{4-8}$$

$$H = \begin{cases} 60^{o} \times \left(\frac{G'-B'}{\Delta} mod 6\right), & C_{max} = R' \\ 60^{o} \times \left(\frac{B'-R'}{\Delta} + 2\right), & C_{max} = G' \\ 60^{o} \times \left(\frac{R'-G'}{\Delta} + 4\right), & C_{max} = B' \end{cases} \tag{4-9}$$

$$S = \begin{cases} 0, & C_{max} = 0 \\ \frac{\Delta}{C_{max}}, & C_{max} \neq 0 \end{cases} \tag{4-10}$$

$$V = C_{max} \tag{4-11}$$

Identificando en la Figura **4-9** el área donde se ubican las esferas azul y verde, se toma el valor de los píxeles en dichas áreas para encontrar el rango para cada color en el canal H.
Rango azul: $158 \leq H \leq 166$
Rango verde: $40 \leq H \leq 66$
Donde el valor H se refiere al valor del píxel en el canal de tono, que se ha ajustado para estar en el rango $[0, 255]$. Con los rangos obtenidos, se realiza una umbralización de la imagen. Valores dentro del rango se hacen iguales a cero, mientras que valores por fuera de este se hacen iguales a 255, creando una imagen a blanco y negro, donde el área oscura representa la ubicación de los objetos azul y verde, Figuras **4-10** y **4-11**, respectivamente.

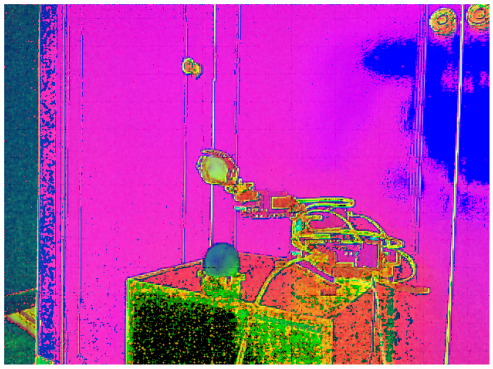

Figura 4-9: Imagen HSV.

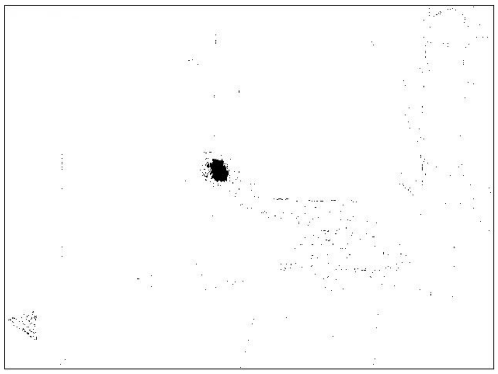

Figura 4-10: Segmentación por umbralización del color azul en el canal H.

Figura 4-11: Segmentación por umbralización del color verde en el canal H.

En las Figuras **4-10** y **4-11**, las imágenes obtenidas de la segmentación presentan ruido. Para disminuirlo, se utiliza la operación apertura morfológica, la cual expande las áreas con píxeles blancos, eliminando las áreas de ruido pequeñas. Para eliminar el ruido por completo, se utiliza la función momento estadístico m_{00} para determinar el área de los contornos resultantes. Con esta información, se eliminan las áreas más pequeñas, hasta que sólo se tiene el contorno con mayor superficie, en este caso, la esfera.

Dado que la imagen de color y profundidad se encuentran alineadas en X y Y, se utilizan las coordenadas del contorno resultante para eliminar de la imagen de profundidad lo que no pertenece al área del objeto, obteniendo la imagen de la Figura **4-12**.

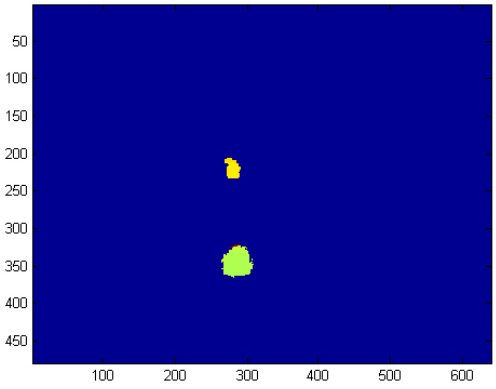

Figura 4-12: Segmentación de la imagen de profundidad.

Para determinar la posición del objeto en el espacio, se calculan los centroides de las áreas que designan los dos objetos en la imagen. Se calcula el centroide en x como el primer *momento estadístico* de x sobre el área (m_{10}/m_{00}) y el centroide en y como el primer *momento estadístico* de y sobre el área (m_{01}/m_{00}). Para las imágenes de ejemplo que se presenta, dichos valores corresponden a las coordenadas en píxeles: $(280, 221)$ y $(286, 346)$ (Figura **4-13**).

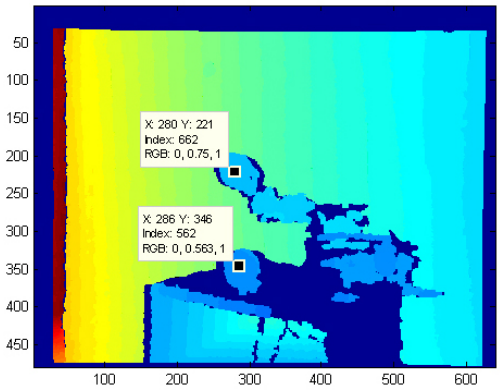

Figura 4-13: Centroides para la imagen de ejemplo.

Una de las funciones más importantes encontrada en la librería de manejo del sensor Kinect, permite convertir los valores de píxeles a milímetros, es decir, de sistema de coordenadas 2D a 3D ó sistema de coordenadas del mundo. Dicha transformación se realiza sobre la imagen de profundidad previamente segmentada, obteniendo el resultado que se muestra en la Figura **4-14**.

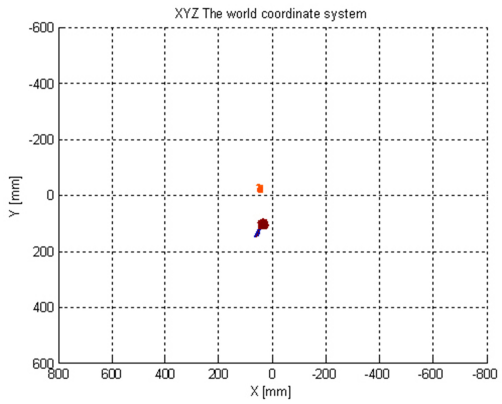

Figura 4-14: Imagen segmentada en coordenadas del mundo.

La matriz con las coordenadas del mundo consta de tres canales, cada uno de dimensión 640×480. Para obtener los datos (X, Y, Z) en milímetros, se lee la matriz en la posición (u, v, Z), donde u y v corresponden a los centroides x y y obtenidos anteriormente, y Z corresponde al valor de profundidad. Para el ejemplo, dichos valores corresponden a las coordenadas en milímetros: $(46, -26, -662)$ para el objeto azul y $(32, 102, -563)$ para el objetivo verde (Figura **4-15**).

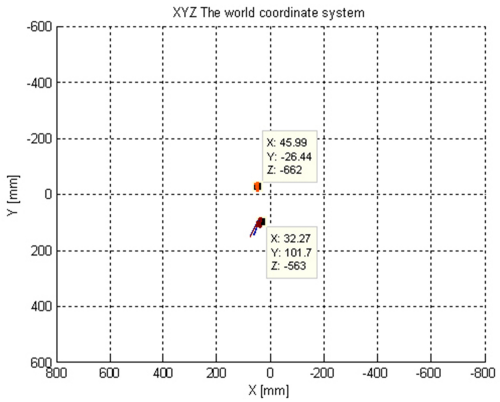

Figura 4-15: Valor de los centroides en coordenadas del mundo.

Las operaciones anteriores permiten hallar la posición de la esfera con respecto a la ubicación de la cámara, la cual corresponde al valor cero de los ejes de coordenadas (X, Y, Z). Si se quisiera agarrar el objeto, sólo la información de posición sería insuficiente. Para obtener más datos del objeto en el espacio, se propone calcular su orientación con respecto a un eje de coordenadas conocido.

Para facilitar el cálculo de la orientación del cubo y el cilindro, se utilizaron marcadores en los objetos. En el caso del cubo, se marcó una línea en cada una de las caras de un color diferente al cuerpo del objeto. Mediante la segmentación de la línea (proceso análogo al procesamiento de la imagen con las esferas) y hallando el punto más alto (x_0, y_0, z_0) y más bajo de esta (x_1, y_1, z_1), se tiene suficiente información para calcular la orientación del objeto en el espacio. En el caso del cilindro, se marcaron las circunferencias superior e inferior con los colores azul y verde. Luego de segmentar, se encuentra el punto más cercano a la cámara en el contorno superior (x_0, y_0, z_0) e inferior (x_1, y_1, z_1). La orientación se calcula utilizando los ángulos XY y YZ (Figura **4-16**).

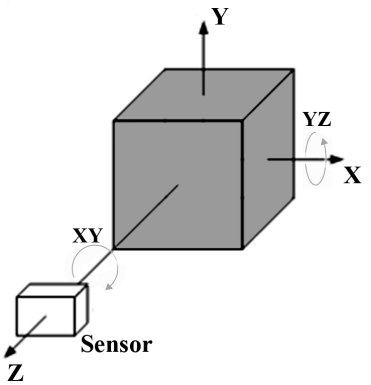

Figura 4-16: Orientación del objeto según ángulos XY y YZ.

Como se observa en la Figura **4-16**, el ángulo XY se refiere a la inclinación del objeto en un plano paralelo al plano de la cámara, es decir, el balanceo a izquierda o derecha con respecto a la imagen captada por la cámara (Ecuación 4-12). El ángulo YZ, se refiere a la inclinación del objeto hacia la cámara, es decir, el balanceo adelante o atrás según la profundidad del punto superior con respecto al inferior (Ecuación 4-13).

$$\tan\theta = \frac{x_1 - x_0}{y_1 - y_0}, \tag{4-12}$$

$$\tan\theta = \frac{z_1 - z_0}{y_1 - y_0}, \tag{4-13}$$

En las Tablas **4-1** y **4-2**, el error se calculó como el valor real (software de procesamiento de imágenes) menos el valor medido sobre el máximo valor posible, en este caso 90º. Con diez pruebas, el error para el cilindro se mantuvo inferior a 2.1 %, con una desviación estándar cercana al 1 %. En el caso del cilindro, el error es menor a 3.5 %, con desviación máxima de 2.3 %. Ambos errores se consideran bajos dada la actividad de agarre de objetos.

Tabla 4-1: Cálculo de la orientación del cilindro

	Cilindro - Programa		**Cilindro - Real**			
Iter.	**XY (°)**	**YZ (°)**	**XY (°)**	**YZ (°)**	**Error XY**	**Error YZ**
1	-2.61	-5.73	-1.9	-4.6	0.79 %	1.26 %
2	4.06	12.68	2.0	11.3	2.29 %	1.53 %
3	15.75	5.84	12.8	4.3	3.28 %	1.71 %
4	2.49	10.78	3.0	13.3	0.57 %	2.80 %
5	-12.02	0.73	-12.7	3.7	0.76 %	3.30 %
6	4.68	8.66	6.9	5.7	2.47 %	3.29 %
7	9.47	18.66	8.3	18	1.30 %	0.73 %
8	9.1	0.71	9.0	1.3	0.11 %	0.66 %
9	3.67	4.36	4.0	1.9	0.37 %	2.73 %
10	1.27	-16.94	3.0	-19.3	1.92 %	2.62 %
				Promedio	1.38 %	2.06 %
				Desviación	1.05 %	1.01 %

Tabla 4-2: Cálculo de la orientación del cubo

	Cubo - Programa		**Cubo - Real**			
Iter.	**XY (°)**	**YZ (°)**	**XY (°)**	**YZ (°)**	**Error XY**	**Error YZ**
1	5.91	-2.42	3.8	-5.1	2.34 %	2.98 %
2	-6.19	14.3	-8.4	11.1	2.46 %	3.56 %
3	-12.51	-5.95	-13.7	-6.1	1.32 %	0.17 %
4	10	-7.83	9.4	-13.9	0.67 %	6.74 %
5	-1.99	-1.07	-1.3	-1.3	0.77 %	0.26 %
6	-1.25	-9.07	-1.1	-10.8	0.17 %	1.92 %
7	9.83	-11.93	11.5	-16.3	1.86 %	4.86 %
8	6.49	-1.95	2.9	-4.8	3.99 %	3.17 %
9	0.79	-11.81	3.5	-17.8	3.01 %	6.66 %
10	-5.07	15.06	-5.6	11.8	0.59 %	3.62 %
				Promedio	1.72 %	3.39 %
				Desviación	1.23 %	2.28 %

4.4. Discusión

Los ángulos de las articulaciones que se calcularon representan de manera fiel los valores reales, lo cual se evidencia al replicar los movimientos realizados durante la demostración sobre el cuerpo del robot.

Los errores obtenidos al calcular la orientación de los objetos se consideran bajos, ya que durante el agarre, la mano siempre presenta una apertura mayor al ancho del objeto, lo cual aumenta significativamente la probabilidad de un agarre exitoso aún si existen pequeños errores en la posición y orientación calculada.

5 Generalización de trayectorias dadas nuevas posiciones iniciales y finales

Se propone generalizar a nuevas trayectorias a partir de una base de datos de movimientos demostrados. Para esto, en este capítulo se presenta la configuración de los experimentos para crear la base de datos, la introducción e implementación de la técnica empleada para la generalización de tanto el punto inicial como el final de la trayectoria, la técnica usada para comparar el desempeño, y la plataforma de simulación utilizada para comprobar el funcionamiento de los algoritmos.

5.1. Configuración del experimento

Los experimentos se realizaron como lo describe la Figura **5-1**. Quien realiza las demostraciones está de pie frente a la mesa donde están indicados los nueve valores finales de las trayectorias. Los nueve valores iniciales se calculan teniendo en cuenta la posición de la mano, cuando el brazo está extendido completamente junto al cuerpo, y la ubicación de la cintura. La base de datos consiste de 81 trayectorias, nueve posiciones iniciales de la mano derecha (Figura **5-1**a), de las cuales se realiza el movimiento a cada uno de los nueve valores finales (Figura **5-1**b). En una mesa, más adelante, se ubica el sensor kinect, con el cual se obtienen las imágenes 2D y 3D que permiten utilizar el algoritmo para la ubicación de los puntos (X, Y, Z) de las articulaciones.

La diferencia con otros trabajos, radica principalmente en que se generan nuevas trayectorias a partir de nuevos valores, tanto iniciales como finales. Se utiliza GPR para predecir los valores de los ángulos iniciales de acuerdo al punto de inicio (punto de consulta q_*), y GPR para determinar los parámetros de nuevas DMPs, que produzcan curvas suaves entre los nuevos ángulos iniciales y finales de las articulaciones.

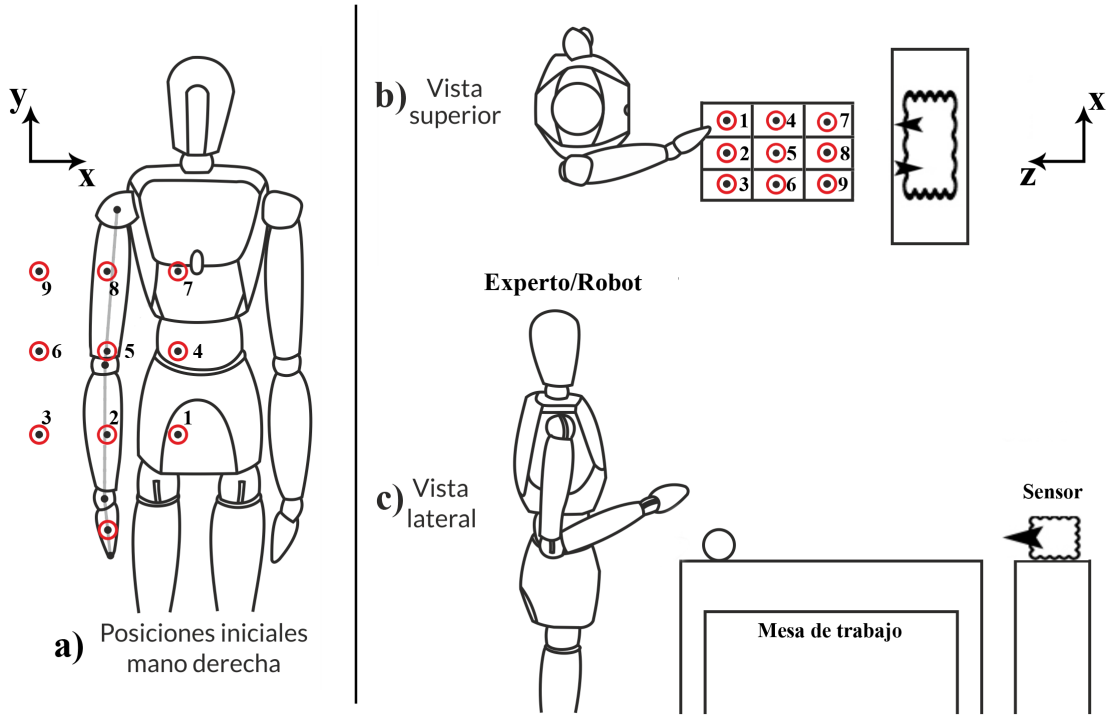

Figura 5-1: Imágenes del experimento realizado. a) Posiciones iniciales de la trayectoria. b) Posiciones finales de la trayectoria (vista superior). c) Configuración general del experimento (vista lateral).

Para contar con una base de datos con mayor información, se realizan ajustes a los valores iniciales y finales de los ángulos obtenidos, cubriendo una mayor área antes de llevar las trayectorias al robot (Figura **5-2**).

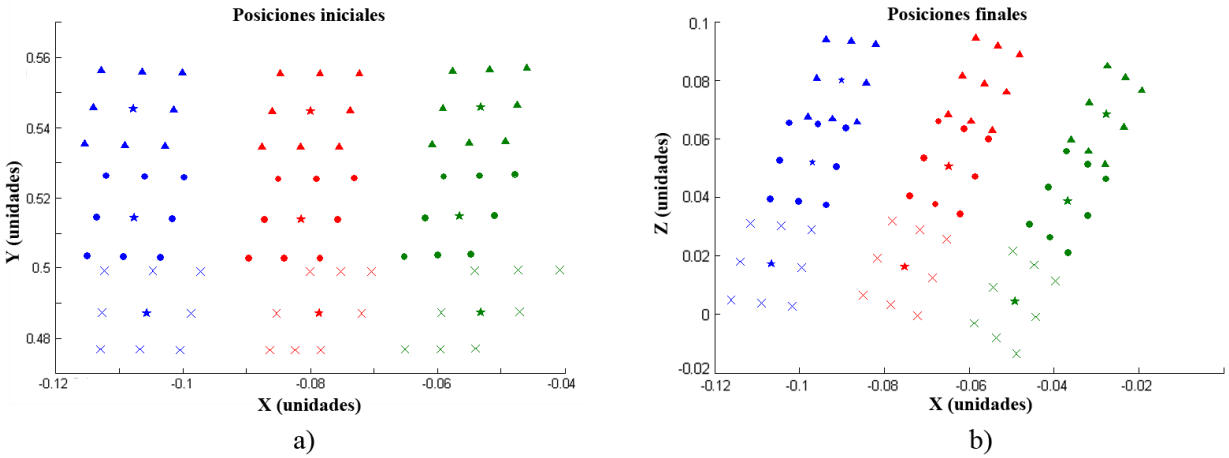

Figura 5-2: Puntos de entrenamiento en el plano cartesiano del robot. a) Posiciones iniciales de las trayectorias. b) Posiciones finales de las trayectorias.

La Figura **5-2**a representa el plano (X, Y) de las posiciones iniciales, mientras que la Figura **5-2**b representa el plano (X, Z) de las posiciones finales, ambas en coordenadas cartesianas del entorno del robot. Las estrellas indican las posiciones originales de inicio y final, mientras que los símbolos restantes son nuevas posiciones estimadas modificando la trayectoria de las articulaciones, movimientos que no son realizados por el usuario. Las posiciones iniciales y finales no se encuentran en una malla cuadrada, tal como se presenta al crear la base de datos (Figura **5-1**), debido a las diferencias en longitud entre el brazo de quien realiza la acción y el brazo del robot, además de ángulos en el operario que no son tenidos en cuenta.

Para realizar la comparación entre DMP con GPR, y la aproximación utilizando la distancia de Mahalanobis y distribución gaussiana, se crearon 30 puntos de consulta q_* en el plano (X, Z) de las posiciones finales utilizando una función aleatoria uniforme (Figura **5-3**). Para cada punto de consulta se desean calcular seis nuevas trayectorias, una para cada grado de libertad, de tal forma que al finalizar el movimiento, la mano del robot se encuentre en la posición cartesiana q_*. DMP con GPR lo logra prediciendo unos nuevos valores (w_*, g_*, τ_*), es decir, una nueva DMP para cada articulación, mientras que la distancia de Mahanobis con distribución gaussiana genera las nuevas curvas promediando las trayectorias cuyos valores finales son cercanos a q_*, y asignando pesos según la distancia.

Como ejemplo de lo que se desea lograr en la comparación, en la Figura **5-4** se presenta uno de los puntos de consulta (Figura **5-4**a) junto a las trayectorias de entrenamiento (líneas azules) y trayectoria generalizada (línea roja) para una de las articulaciones (Figura **5-4**b) obtenida utilizando DMP con GPR.

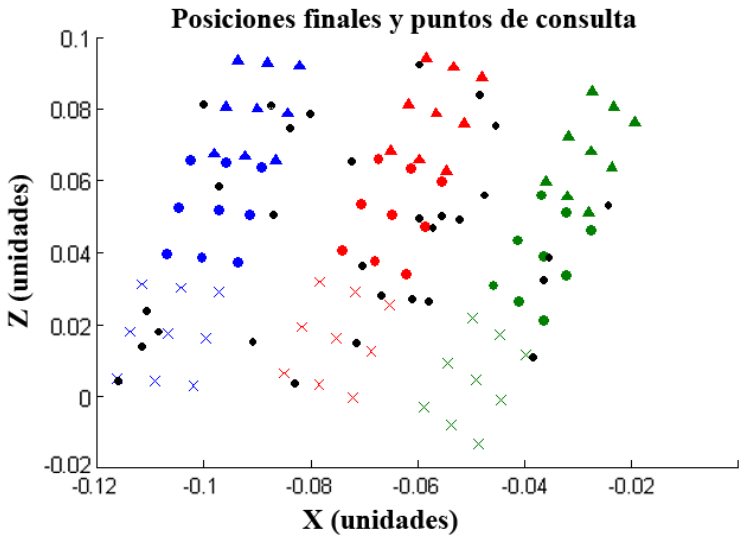

Figura 5-3: Posiciones finales (formas de colores) y puntos de consulta (puntos negros).

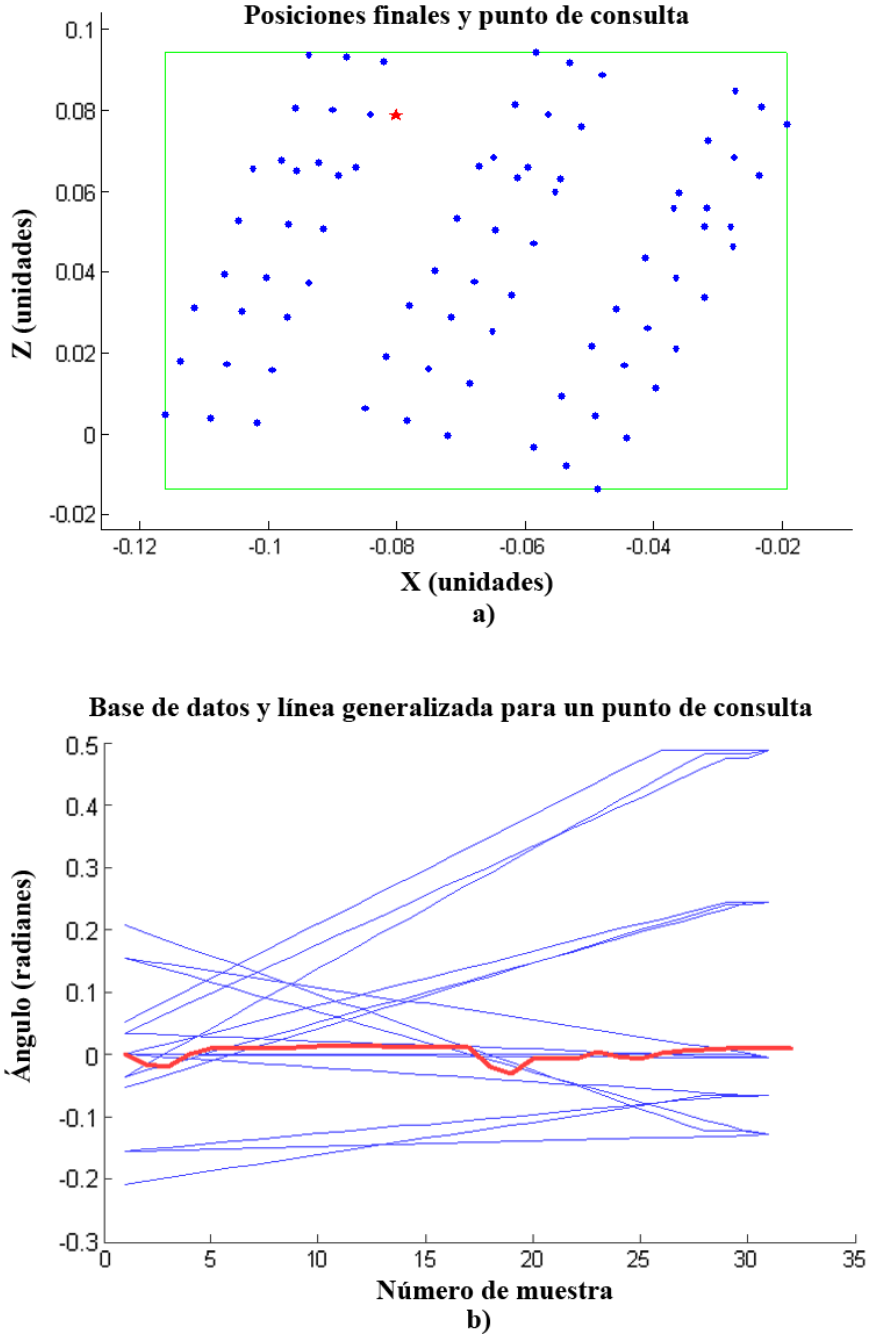

Figura 5-4: Generalización de trayectorias. a) Ubicación del punto de consulta en rojo. b) Algunas curvas de la base de datos (azul) y curva generalizada para la articulación guiñada del hombro (rojo).

5.2. Regresión de procesos gaussianos y primitivas de movimiento dinámico

Se propone generalizar a nuevas trayectorias a partir de otras ya conocidas, lo cual puede verse como un problema de predicción. Si se tienen observaciones con ruido de una variable dependiente en ciertos valores de la variable independiente x, se quiere estimar el valor de la variable dependiente en un nuevo valor x_*.

Si se asume cierto patrón sobre los datos de entrada, se pueden utilizar diferentes métodos para ajustar $f(x)$ a una línea recta, o una función cuadrática, cúbica, o incluso no-polinómica. La regresión de procesos gaussianos (GPR por sus siglas en inglés) es una aproximación más fina, ya que no relaciona $f(x)$ a un modelo específico, sino que por medio de un proceso gaussiano se representa $f(x)$ de forma indirecta, pero rigurosa.

En el Algoritmo 5, se presentan los pasos para utilizar la técnica regresión de procesos gaussianos. La fundamentación matemática se encuentra más adelante. Para ilustrar los resultados obtenidos al utilizar esta técnica, se presenta la Figura **5-5**, donde a partir de seis datos de entrenamiento y calculando la predicción de 1000 puntos de consulta, puede observarse la curva generada por los valores predichos de la variable dependiente (línea negra). La información de covarianza permite conocer la confianza de los valores calculados (sombra roja).

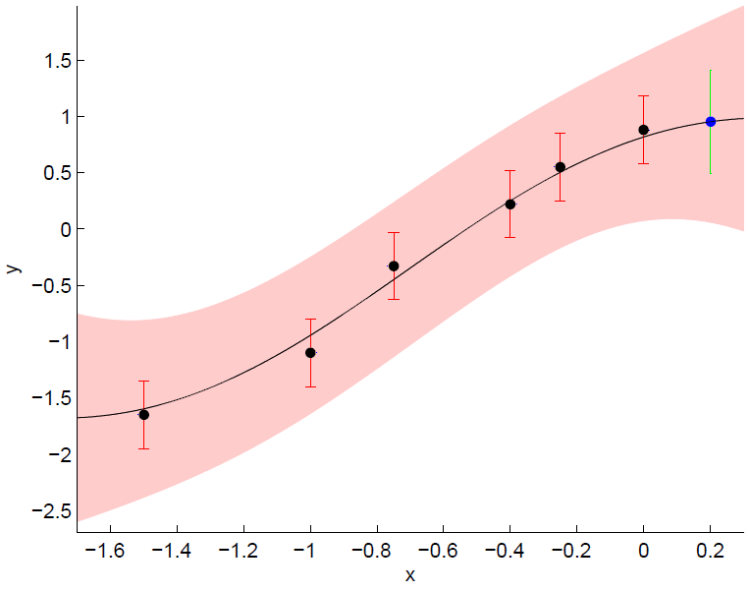

Figura 5-5: GPR utilizado para estimar 1000 valores de $f(x_*)$ (línea negra) a partir de los datos de entrenamiento (puntos negros) e intervalo de confianza del 95 % (sombra roja), calculado utilizando la covarianza de $f(x_*)$. Tomado de [3].

Algoritmo 5 Algoritmo implementado para regresión de procesos gaussianos

1: **procedimiento** Regresión de procesos gaussianos
2: Ajuste de los datos de entrenamiento.
3: Entrenamiento de GPR para ajustar los hiperparámetros.
4: Predicción de los parámetros w, g y τ a partir del punto de consulta x_*, utilizando la Ecuación 5-4.
5: Reproducción de la nueva trayectoria utilizando la DMP obtenida.
6: **end procedimiento**

Una de las ventajas de GPR se presenta en el ajuste de los hiperparámetros. En el Algoritmo 5, la técnica se entrena con los datos de entrada, y con el cálculo del logaritmo negativo de la probabilidad marginal [81], se obtienen unos valores apropiados. Luego, se procede a realizar la predicción para cada término de la nueva DMP teniendo en cuenta el punto de consulta definido.

En este trabajo, se generaliza a nuevos movimientos a partir de los parámetros de pesos (w), objetivo (g) y variable de tiempo (τ) de las DMP obtenidas de las demostraciones (Ecuación 5-1) [34, 30, 33].

$$\mathbf{G}\left(\{\mathbf{w_i}, \mathbf{g_i}, \tau_\mathbf{i}; \mathbf{q_i}\}_{\mathbf{i=1}}^{\mathbf{NumEj}}\right) : \mathbf{q}_* \mapsto (\mathbf{w}_*, \mathbf{g}_*, \tau_*), \tag{5-1}$$

donde los parámetros w_i, g_i y τ_i, son los parámetros de las DMP de entrenamiento, y están ligados a un punto q_i en la base de datos, el cual puede representar la posición inicial o final de la trayectoria en el espacio de trabajo. Cuando q_i se refiere al valor inicial del movimiento, los objetivos g_i son los valores iniciales de las articulaciones, mientras que cuando q_i es el valor final de la trayectoria, g_i se refiere a los valores finales de las articulaciones al realizar el movimiento. Por su parte, $NumEj$ es el número de ejecuciones o demostraciones que componen la base de entrenamiento, a partir de la cual se estiman los parámetros de una nueva DMP (w_*, g_*, τ_*) dado un punto de consulta inicial y/o final q_*.

Para esto, la regresión de procesos gaussianos se basa en el modelo probabilístico bayesiano [82]. Dado un conjunto de n datos de entrenamiento $\{x_i, y_i\}_{i=1}^n$, se desea encontrar una función $f(x_i)$ que transforma el vector de entrada x_i en el objetivo y_i, dado un modelo $y_i = f(x_i)+\epsilon_i$, donde ϵ_i es ruido gaussiano con media cero y varianza σ_n^2. Como resultado, los objetivos observados pueden describirse por una distribución gaussiana $y \sim \mathcal{N}(0, K(X, X)+\sigma^2 I)$, donde X es el grupo que contiene todos los puntos de entrada x_i y $K(X, X)$ la matriz de covarianza calculada utilizando la función de covarianza dada. Una de las funciones más utilizadas se basa en kernels gaussianos (Ecuación 5-2).

$$k(x_i, x_*) = \sigma_s^2 \exp(-\frac{1}{2}(x_i - x_*)^T W (x_i - x_*)), \tag{5-2}$$

donde σ_s^2 es la varianza de la señal y W representa el ancho del kernel gaussiano. La distribución conjunta de los valores del objetivo observado y predicho $f(x_*)$, para un punto de consulta x_*, está dado por la Ecuación 5-3.

$$\begin{bmatrix} y \\ f(x_*) \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) + \sigma_n^2 I & k(X,x_*) \\ k(x_*,X) & k(x_*,x_*) \end{bmatrix}\right). \tag{5-3}$$

El valor esperado $f(x_*)$, asociado al nuevo punto de consulta x_*, con su correspondiente covarianza $V(x_*)$, está dado por las Ecuaciones 5-4 y 5-5, respectivamente.

$$f(x_*) = k_*^T (K + \sigma_n^2 I)^{-1} y = k_*^T \alpha, \tag{5-4}$$

$$V(x_*) = k(x_*, x_*) - k_*^T (K + \sigma_n^2 I)^{-1} k_*, \tag{5-5}$$

donde $k_* = k(X, x_*)$, $K = K(X, X)$, y α denota el vector de predicción, el cual depende de los datos de entrenamiento, los datos de la variable dependiente y los valores objetivo.

La parte computacionalmente más costosa es $(K + \sigma_n^2 I)^{-1}$, pero dado que esta matriz sólo depende de los datos de entrenamiento, los cálculos necesarios pueden realizarse fuera de línea. Si definimos:

$$z = (K + \sigma_n^2 I)^{-1} y, \tag{5-6}$$

de la Ecuación 5-4, entonces el parámetro $f(x_*)$ puede escribirse como:

$$f(x_*) = \sum_{i=1}^{NumEj} k(x_*, x_i) z_i, \tag{5-7}$$

donde, al ser usado para determinar los parámetros de una nueva DMP como en este caso, $f(x_*)$ se refiere a $\bar{\tau}_*$, $\bar{g}_*$ y $\bar{w}_*$. Por lo tanto, los datos de entrenamiento tienen un peso basado en la distancia entre el punto de consulta de entrenamiento y el punto de consulta actual. Esto resulta en que puntos de entrenamiento más cercanos tienen mayor influencia en el resultado, por lo que puede pensarse en GPR como un método de regresión local.

5.3. Distancia de Mahalanobis y distribución gaussiana

Para comparar el desempeño de la técnica GPR, se propone realizar la generalización de los movimientos en la base de datos a nuevas trayectorias utilizando un algoritmo de asignación

de pesos locales, la cual se basa en distancia de Mahalanobis y distribución gaussiana. Se calcula la distancia de las posiciones finales al punto de consulta o nueva posición final deseada, utilizando la distancia de Mahalanobis [83], la cual tiene en cuenta la correlación en los datos, ya que se calcula usando la inversa de la matriz varianza-covarianza (Σ) del conjunto de datos de interés (Ecuación 5-8).

$$d_m(\overrightarrow{x}, \overrightarrow{y}) = \sqrt{(\overrightarrow{x} - \overrightarrow{y})^T \Sigma^{-1} (\overrightarrow{x} - \overrightarrow{y})}. \tag{5-8}$$

Basado en la distancia, se asignan pesos a las trayectorias utilizando una distribución gaussiana (Ecuación 5-9).

$$w = \exp(-0{,}5 * d_m). \tag{5-9}$$

De acuerdo a los pesos, sólo las trayectorias por encima de cierto umbral se tienen en cuenta para la generalización. Los pesos de dichas trayectorias se normalizan a valores entre 0 y 1, y se promedian con los pesos correspondientes. En la Figura **5-6**, se presenta como ejemplo la distancia de Mahalanobis calculada para uno de los puntos de consulta. Los puntos que se encuentran en el elipse, presentan una distancia constante a la posición representada con una estrella. Dado que el cálculo tiene en cuenta la relación entre las variables X y Z, la elipse se expande en el eje X, para ajustarse al espacio que hay entre los grupos de datos (columnas cuasi-paralelas al eje Z).

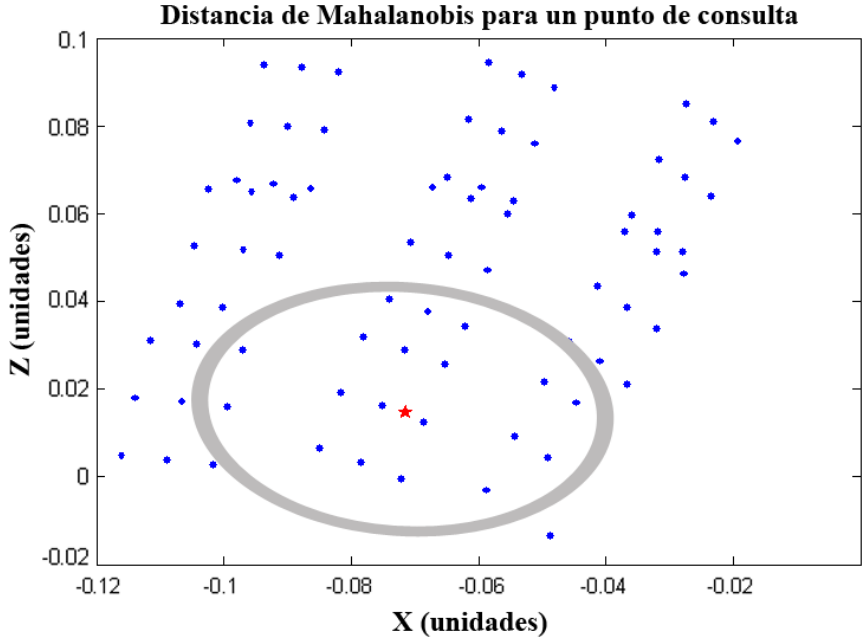

Figura 5-6: Distancia de Mahalanobis. El círculo representa valores a la misma distancia del punto de consulta.

5.4. Plataforma de simulación del robot iCub

El robot iCub fue diseñado como una plataforma común para investigadores interesados en el estudio de sistemas cognitivos artificiales y ha sido adoptado por cerca de 20 laboratorios a nivel mundial [13, 14].

La plataforma se ha utilizado para comprobar la obtención de habilidades cognitivas para interactuar con el mundo físico que lo rodea y manipular objetos de manera flexible [84, 85]. De forma más específica, se le han enseñado habilidades como la arquería [86] y el disparo a objetos estacionarios y en movimiento [87], utilizando visión por computador y aprendizaje de máquina.

Para el presente trabajo, se utiliza el simulador del robot (Figura **5-7**), el cual permite la programación de la plataforma de forma idéntica a como se realiza en el robot real.

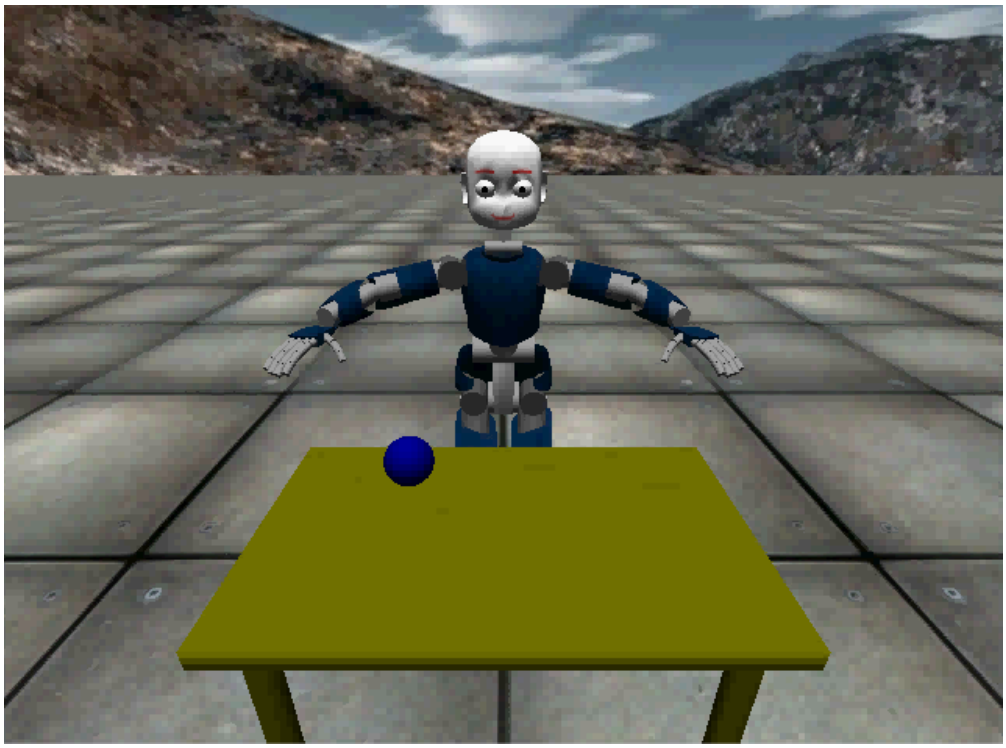

Figura 5-7: Ambiente simulado del robot iCub.

En la Figura **5-7**, se observa que en el entorno virtual del robot es posible crear objetos como mesas y esferas. Para el proyecto, se utilizan las esferas para marcar las posiciones objetivo, para obtener información visual del error.

A modo de ejemplo, en la Figura **5-8** se muestra una secuencia del movimiento en dirección al objetivo, realizada al crear la base de datos (izquierda) y al reproducirla en el robot (derecha).

En la Figura **5-8**, se presenta una de las secuencias que se utilizan como entrenamiento para el robot. Del movimiento del operario, se obtienen los valores de las articulaciones al realizar la acción, las cuales se ajustan y se codifican como DMP, para limitar el uso de espacio en memoria, y para la posterior generalización utilizando GPR.

Para el cálculo del error al objetivo se utiliza la Ecuación 5-10.

$$err_{ob} = \frac{q*_{x,z} - q'_{x,z}}{h_{x,z}}, \tag{5-10}$$

donde $q*_{x,z}$ es la coordenada del punto de consulta en X o en Z, $q'_{x,z}$ es el valor final de la reproducción en el robot (en X o en Z) al utilizar las técnicas GPR y distancia de Mahalanobis con distribución gaussiana, y $h_{x,z}$ es el ancho, es decir, la distancia entre los puntos de entrenamiento más a la izquierda y a la derecha para el eje X, y más arriba y abajo para el eje Z.

Para cada una de las 30 trayectorias generadas a los nuevos objetivos (Figura **5-3**), se calcula el error en los ejes X y Z al realizar el movimiento en el robot simulado iCub (Figura **5-9**). En la Tabla **5-1**, se presenta una síntesis de los resultados obtenidos, calculando los valores máximo, mínimo, promedio y desviación estándar.

Tabla 5-1: Comparación entre la técnica empleada y la técnica basada en distancia de Mahalanobis y distribución gaussiana.

	GPR con DMP		Mahal. y gauss.	
Error	x	z	x	z
Máximo	2,06 %	5,97 %	8,78 %	7,52 %
Mínimo	0,12 %	0,01 %	0,07 %	0,12 %
Promedio	0,86 %	1,52 %	2,82 %	2,98 %
Desviación estándar	0,55 %	1,42 %	2,60 %	2,33 %

Como se observa en la Figura **5-9**, donde se organizaron los errores de menor a mayor según los valores de la técnica GPR, en algunos puntos de consulta el error obtenido con la distancia de Mahalanobis y distribución gaussiana es menor. GPR compensa esto presentando un menor error promedio, como lo presenta la Tabla **5-1**. Además, el error en x presentado por GPR es cuatro veces más pequeño que el encontrado con la técnica de comparación. La desviación estándar para la regresión de procesos gaussianos se mantiene menor a 1,5 %

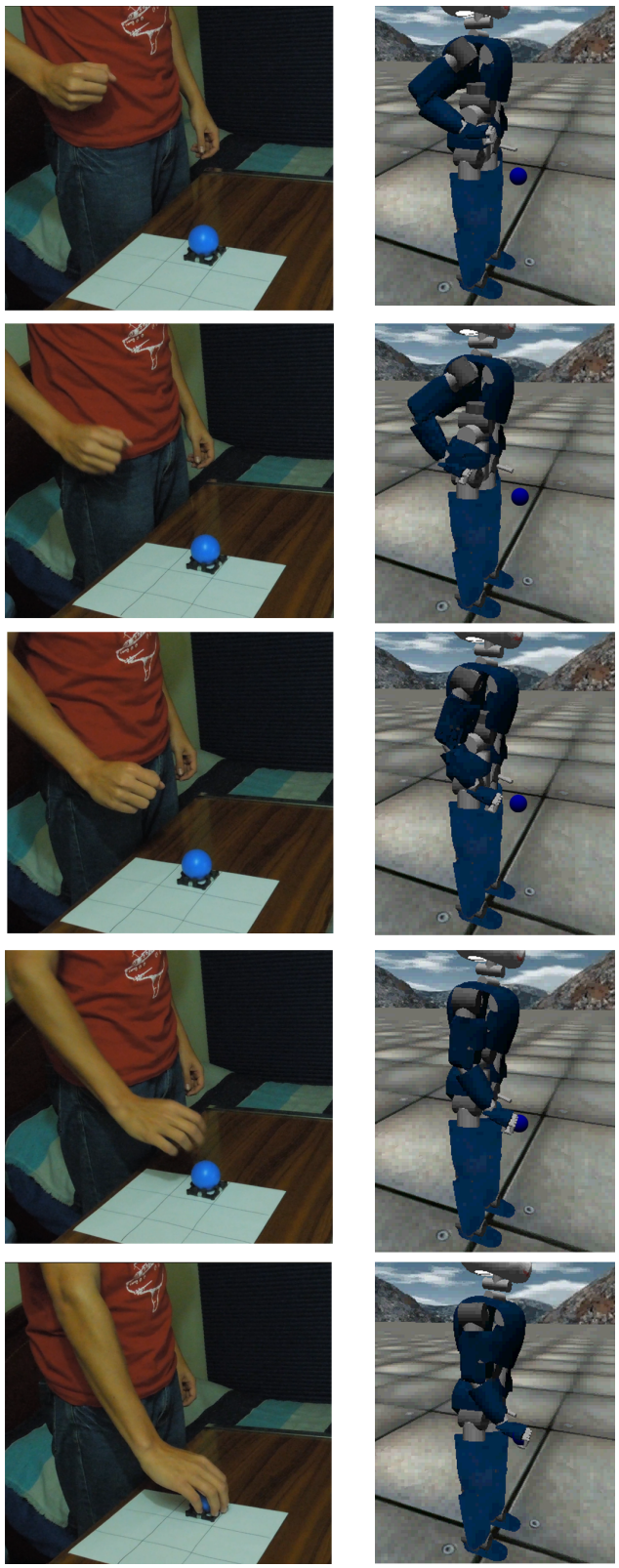

Figura 5-8: Secuencia de agarre de un objeto. Izquierda: Acción realizada por quien realiza la demostración. Derecha: Acción realizada en el robot simulado iCub.

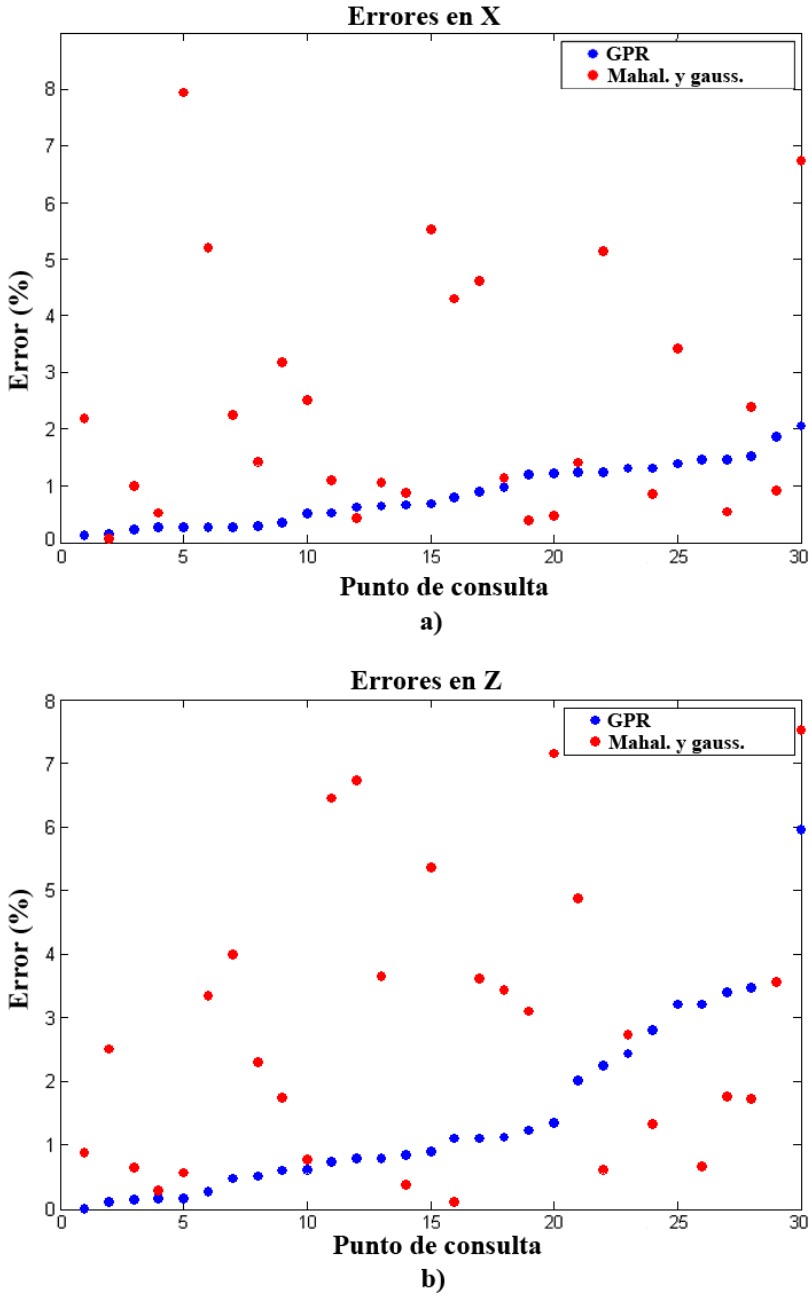

Figura 5-9: Errores al objetivo utilizando GPR y la distancia de Mahalanobis con distribución gaussiana. a) Errores en el eje X. b) Errores en el eje Z.

tanto en x como en z, sobre un valor promedio de 1,52 % en z, mientras que la distancia de Mahalanobis y distribución gaussiana presenta una desviación estándar mayor a 2,3 % sobre un promedio de 2,82 % en x.

Por motivos de comparación, hasta este punto solo se han mostrado resultados de predicción de la trayectoria para alcanzar una nueva posición final. La técnica que utiliza la distancia de Mahalanobis y distribución gaussiana no puede realizar generalización cuando se varían tanto el punto de inicio como el punto final, ya que las trayectorias que tienen más peso para la generalización del punto de inicio, no necesariamente serán las mismas trayectorias con mayor prioridad para generalizar el punto final.

Para comprobar el comportamiento de la técnica GPR cuando se proponen nuevos valores iniciales y finales, se crean puntos de consulta en el área de inicio y fin (Figura **5-10**), los cuales sirven como posiciones iniciales y finales de una nueva trayectoria, la cual se estima generalizando a partir de la base de datos de trayectorias conocidas.

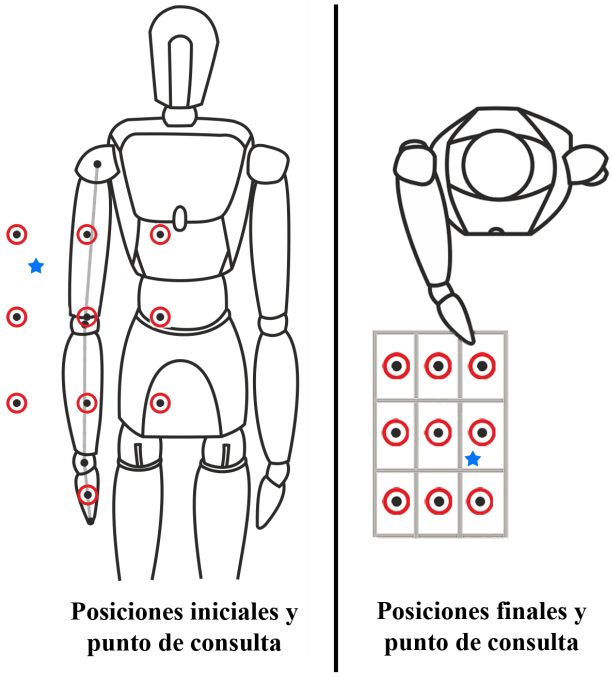

Figura 5-10: Nuevos puntos de consulta inicial y final.

Se crearon 30 puntos de consulta en el área de entrenamiento de las posiciones iniciales (Figura **5-11**), cada uno de los cuales se emparejó con uno de los 30 puntos de consulta en el área de entrenamiento de las posiciones finales (Figura **5-3**).

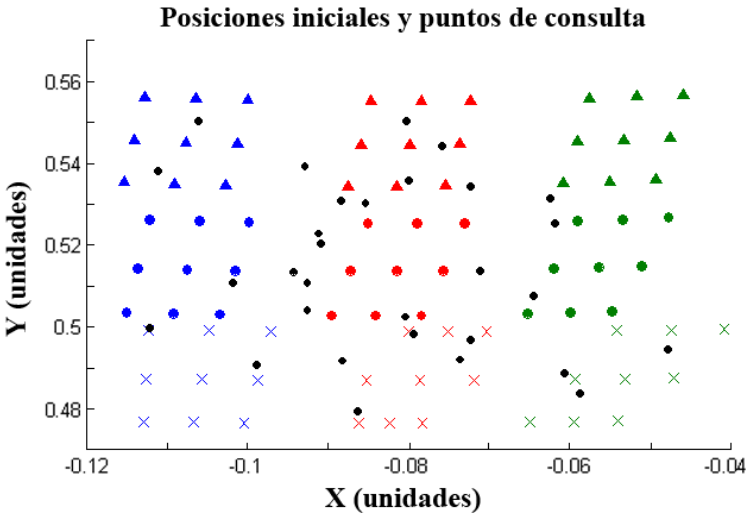

Figura 5-11: Posiciones iniciales y puntos de consulta.

Tanto los puntos de consulta de posición inicial como final, se crearon utilizando una función aleatoria uniforme. En la Tabla **5-2** se presenta una síntesis de los datos de error obtenidos al generalizar a un punto de consulta de inicio.

Tabla 5-2: Desempeño de la técnica empleada al generalizar el punto de inicio.

	GPR con DMP	
Error	x	y
Máximo	2,68 %	1,44 %
Mínimo	0,08 %	0,01 %
Promedio	0,78 %	0,72 %
Desviación estándar	0,61 %	0,48 %

De la Tabla **5-2** se observa que el error promedio se mantiene por debajo del 0,8 %, tanto en el eje X como en el eje Y. Este error se considera bajo.

5.5. Discusión

Un análisis más detallado de los errores al utilizar las técnicas GPR, y distancia de Mahalanobis con distribución gaussiana, permite observar que los máximos se producen cuando los puntos de consulta se encuentran en los extremos del área de entrenamiento (Figura **5-12**). Es importante anotar que aunque se habla de los errores máximos, el error en GPR es menor al presentado en la técnica de comparación.

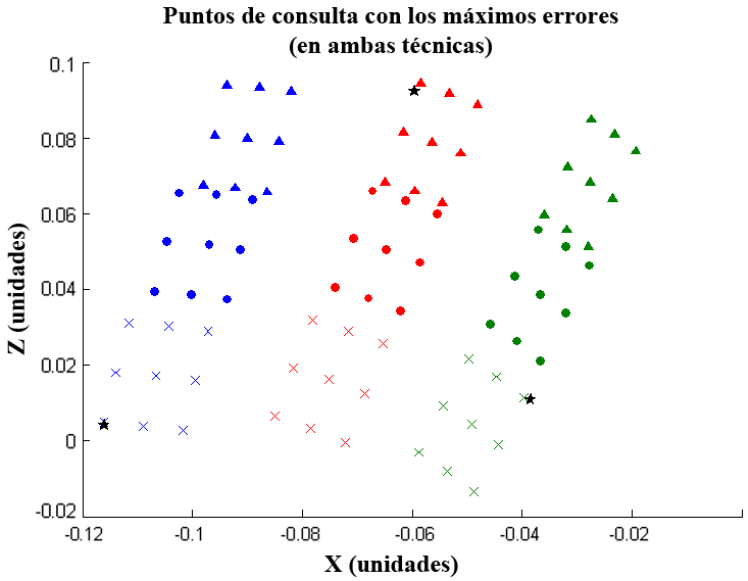

Figura 5-12: Puntos de consulta (estrellas) con los máximos errores al utilizar tanto GPR como distancia de mahalanobis y distribución gaussiana.

Los errores promedio de la técnica GPR, para generalización de punto de inicio (Tabla **5-2**) y del punto final (Tabla **5-1**), nunca fueron superiores al 1,6 %. Este error se considera bajo, más aún cuando la desviación estándar se mantuvo menor al 1,5 %. Estos resultados superan a la generalización utilizando la distancia de Mahalanobis y distribución gaussiana, que aunque en ciertos puntos presenta un error menor, el análisis estadístico de los 30 experimentos sugiere que su desempeño general es más bajo.

La generalización de trayectorias se ha utilizado para varias aplicaciones, por ejemplo, para enseñar a un robot a atar nudos sin importar la posición inicial de la cuerda [88], para generación de letras a mano alzada a partir de muestras dadas [51], y para generalizar el movimiento al lanzar dardos y al golpear en ping-pong [89]. Los artículos revisados suelen enfocarse en cómo debe actuar el robot ante un cambio en la posición final o un cambio en el objetivo. En este trabajo, se presenta una técnica que permite una variación tanto en la posición inicial como final, la cual aprovecha la característica de DMP de crear trayectorias suaves, después de hallar los nuevos valores iniciales y finales para las articulaciones del robot.

6 Conclusiones

En este trabajo se propuso la generalización de nuevas trayectorias para un brazo robótico, a partir de una base de datos de movimientos obtenidos por demostración de un operario humano. Los movimientos constan de las trayectorias de los seis grados de libertad necesarios para alcanzar un objeto ubicado en una mesa frente al operario o robot. La ubicación y orientación del objeto se calcula procesando imágenes de color y profundidad, obtenidos con un sensor RGBD. Los movimientos se codificaron mediante una técnica de aprendizaje por imitación, y cuando el objeto se encuentra en una posición que no se encuentra en la base de datos de entrenamiento, se genera una nueva trayectoria a partir de los datos conocidos utilizando una técnica de regresión.

Fue posible obtener nuevas trayectorias de los seis grados de libertad analizados, mediante el uso de la técnica de regresión de procesos gaussianos. Dicha técnica permite utilizar la base de datos de entrenamiento ya codificada, para generar nuevos movimientos cuando el objeto ubicado sobre la mesa no está en una posición demostrada previamente.

Las pruebas con las trayectorias generalizadas mostraron errores pequeños. Para llegar a estos resultados, se reprodujeron los movimientos de entrenamiento y generalizados en la plataforma de simulación del robot iCub. En esta, la programación se lleva a cabo de forma idéntica a como se realiza sobre el robot real. Se obtuvieron las trayectorias estimadas al crear 30 puntos de consulta para los valores iniciales, así como 30 puntos de consulta en la posición final de las trayectorias. Para la comparación, se implementó también una técnica basada en distancia de Mahalanobis y distribución gaussiana. Los resultados estadísticos señalan un mayor desempeño por parte de la técnica regresión de procesos gaussianos.

La selección de la técnica de programación por demostración utilizada para codificar los movimientos del operario humano, se realizó luego de comparar varios de los algoritmos más utilizados en la literatura. Para esto, se partió de la definición de una taxonomía, la cual agrupa las técnicas según alguna característica específica. Se analizó el comportamiento de las técnicas seleccionadas, según el error al valor final de la trayectoria, la diferencia entre la señal de entrada y salida por medio de la correlación cruzada, y el tiempo que les toma realizar la codificación y reproducción. Los resultados de esta comparación mostraron de forma cuantitativa, el desempeño de cada una de las cuatro técnicas estudiadas.

El resultado del análisis de las técnicas está limitado a las librerías y los parámetros utilizados. Se relacionan otros trabajos donde las capacidades individuales de los algoritmos se aumentan, mediante cambios en las ecuaciones base o en la técnica de reconstrucción utilizada, lo cual no se tuvo en cuenta en la ejecución de este proyecto. De acuerdo a estos resultados, se seleccionaron las primitivas de movimiento dinámico para la codificación y reconstrucción de las trayectorias de entrenamiento.

Los errores obtenidos al calcular la orientación de los objetos, se consideran pequeños dada la actividad de agarre. Para determinar el error, se utilizó un sensor RGBD, el cual produce imágenes 2D y 3D, o imágenes de color y profundidad, respectivamente. La ubicación de los objetos ubicados en la mesa frente al operario (esfera, cilindro, cubo), está dada por los valores (x, y, z) del centroide del objeto, y la orientación se define por dos ángulos de inclinación, los cuales se comparan con los ángulos calculados con un programa comercial de procesamiento de imágenes.

La adquisición del movimiento del operario humano se logra también utilizando el sensor RGBD. Para esto, se utilizan librerías diseñadas específicamente para obtener la posición de las articulaciones de quien realiza la demostración. Con esta información, se presentó paso a paso el cálculo del valor de seis grados de libertad del cuerpo, que tienen relación con el movimiento de llevar la mano hacia una posición deseada en el espacio de trabajo, movimientos con los que se crea la base de datos para la posterior generalización a nuevos movimientos.

Se propone como trabajo futuro, la implementación de las técnicas DMP y GPR para generalización a nuevas trayectorias, sobre un brazo robótico real. Para esto, se deberá estimar una matriz de transformación para relacionar las posiciones alcanzadas por el operario y las alcanzadas por el robot, teniendo en cuenta las diferencias en alcance por las longitudes de los enlaces. Además, después de llevar la mano a la posición del objeto sobre la mesa, capturar información del agarre, que pueda ser reproducida luego por el efector final del brazo robótico y que puede codificarse utilizando las técnicas se aprendizaje por imitación estudiadas en este trabajo.

Bibliografía

[1] Micha Hersch, Florent Guenter, Sylvain Calinon, and Aude G Billard. Learning dynamical system modulation for constrained reaching tasks. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 444–449. IEEE, 2006.

[2] T. Inamura, H. Tanie, and Y. Nakamura. Keyframe compression and decompression for time series data based on the continuous hidden markov model. In *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003. IROS 2003.*, volume 2, pages 1487 – 1492 vol.2, oct. 2003.

[3] M Ebden. Gaussian processes for regression: A quick introduction. *The Website of Robotics Research Group in Department on Engineering Science, University of Oxford*, 2008.

[4] M. Lopes and J. Santos-Victor. Visual transformations in gesture imitation: what you see is what you do. In *IEEE International Conference on Robotics and Automation, 2003. ICRA '03.*, volume 2, pages 2375 – 2381 vol.2, sept. 2003.

[5] M. Lopes and J. Santos-Victor. A developmental roadmap for learning by imitation in robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(2):308 –321, april 2007.

[6] M. Lopes, F. Melo, L. Montesano, and J. Santos-Victor. Abstraction levels for robotic imitation: Overview and computational approaches. In Olivier Sigaud and Jan Peters, editors, *From Motor Learning to Interaction Learning in Robots*, volume 264 of *Studies in Computational Intelligence*, pages 313–355. Springer Berlin / Heidelberg, 2010.

[7] Adrián León, Eduardo Morales, Leopoldo Altamirani, and Jaime Ruiz. Teaching a robot new tasks through imitation and feedback. In *Proc. Workshop: New Developments in Imitation Learning, as part of the International Conference on Machine Learning*, 2011.

[8] Adrián León, Eduardo Morales, Leopoldo Altamirani, and Jaime Ruiz. Teaching a robot new tasks through imitation and on-line feedback. In *Proc. of the 16th. Iberoamerican Congress on Pattern Recognition*, 2011.

[9] Monica N. Nicolescu and Maja J Matarić. Task learning through imitation and human-robot interaction. In *Models and Mechanisms of Imitation and Social Learning in Ro-*

bots, Humans and Animals: Behavioural, Social and Communicative Dimensions, pages 407–424. University Press, 2005.

[10] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[11] Wolfram Erlhagen, Albert Mukovskiy, Estela Bicho, Giorgio Panin, Csaba Kiss, Alois Knoll, Hein van Schie, and Harold Bekkering. Action understanding and imitation learning in a robot-human task. In *Artificial Neural Networks: Biological Inspirations. ICANN 2005*, volume 3696 of *Lecture Notes in Computer Science*, pages 261–268. Springer Berlin / Heidelberg, 2005.

[12] A Billard, S Calinon, R Dillmann, and S Schaal. Handbook of robotics chapter 59: Robot programming by demonstration. *Robotics*, chapter 59:1371–1394, 2007.

[13] Giorgio Metta, Giulio Sandini, David Vernon, Lorenzo Natale, and Francesco Nori. The icub humanoid robot: an open platform for research in embodied cognition. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, PerMIS '08, pages 50–56, New York, NY, USA, 2008. ACM.

[14] N.G. Tsagarakis, G. Metta, G. Sandini, D. Vernon, R. Beira, F. Becchi, L. Righetti, J. Santos-Victor, A.J. Ijspeert, M.C. Carrozza, and D.G. Caldwell. icub: the design and realization of an open humanoid platform for cognitive and neuroscience research. *Advanced Robotics*, 21(10):1151–1175, 2007.

[15] Sylvain Calinon, Petar Kormushev, and Darwin G Caldwell. Compliant skills acquisition and multi-optima policy search with em-based reinforcement learning. *Robotics and Autonomous Systems*, 61(4):369–379, 2013.

[16] Jens Kober and Jan Peters. Movement templates for learning of hitting and batting. In *Learning Motor Skills*, pages 69–82. Springer, 2014.

[17] Stefan and Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233 – 242, 1999.

[18] Peter Englert, Alexandros Paraschos, Jan Peters, and Marc Peter Deisenroth. Model-based imitation learning by probabilistic trajectory matching. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1922–1927. IEEE, 2013.

[19] Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340, 2011.

[20] Itauma Isong Itauma, Hasan Kivrak, and Hatice Kose. Gesture imitation using machine learning techniques. In *2012 20th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4. IEEE, 2012.

[21] Fabian Chersi. Learning through imitation: a biological approach to robotics. *Autonomous Mental Development, IEEE Transactions on*, 4(3):204–214, 2012.

[22] Milad S Malekzadeh, Sylvain Calinon, Danilo Bruno, and Darwin G Caldwell. Learning by imitation with the stiff-flop surgical robot: a biomimetic approach inspired by octopus movements. *Robotics and Biomimetics*, 1(1):1–15, 2014.

[23] Dana Kulić, Christian Ott, Dongheui Lee, Junichi Ishikawa, and Yoshihiko Nakamura. Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research*, page 0278364911426178, 2011.

[24] Stephan Al-Zubi and Gerald Sommer. Learning to imitate human movement to adapt to environmental changes. *International Conference on Pattern Recognition*, 1:191–194, 2006.

[25] Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. *Advanced Robotics*, 25(5):581–603, 2011.

[26] Amir Sadeghipour and Stefan Kopp. Embodied gesture processing: Motor-based integration of perception and action in social artificial agents. *Cognitive computation*, 3(3):419–435, 2011.

[27] Bogdan Moldovan, Martijn van Otterlo, Plinio Moreno, José Santos-Victor, and Luc De Raedt. Statistical relational learning of object aff ordances for robotic manipulation. *Latest advances in inductive logic programming*, page 6, 2012.

[28] Bogdan Moldovan, Plinio Moreno, Martijn van Otterlo, José Santos-Victor, and Luc De Raedt. Learning relational affordance models for robots in multi-object manipulation tasks. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4373–4378. IEEE, 2012.

[29] Zhikun Wang, Marc Peter Deisenroth, Heni Ben Amor, David Vogt, Bernhard Schölkopf, and Jan Peters. Probabilistic modeling of human movements for intention inference. In *In Proceedings of Robotics: Science and Systems (R: SS). 99*. Citeseer, 2012.

[30] Denis Forte, Aleš Ude, and Andrej Gams. Real-time generalization and integration of different movement primitives. In *2011 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 590–595. IEEE, 2011.

[31] Huan Tan, Erdem Erdemir, Kazuhiko Kawamura, and Qian Du. A potential field method-based extension of the dynamic movement primitive algorithm for imitation learning with obstacle avoidance. In *2011 International Conference on Mechatronics and Automation (ICMA)*, pages 525–530. IEEE, 2011.

[32] Bojan Nemec and Ales Ude. Action sequencing using dynamic movement primitives. *Robotica*, 30(5):837, 2012.

[33] Ales Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, 26(5):800–815, 2010.

[34] Denis Forte, Andrej Gams, Jun Morimoto, and Aleš Ude. On-line motion synthesis and adaptation using a trajectory database. *Robotics and Autonomous Systems*, 60(10):1327–1339, 2012.

[35] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(2):286–298, 2007.

[36] Sylvain Calinon and Aude Billard. Statistical learning by imitation of competing constraints in joint space and task space. *Advanced Robotics*, 23(15):2059–2076, 2009.

[37] Kaushik Subramanian. Task space behavior learning for humanoid robots using gaussian mixture models. In *AAAI*, 2010.

[38] Carol E Reiley, Erion Plaku, and Gregory D Hager. Motion generation of robotic surgical tasks: Learning from expert demonstrations. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 967–970. IEEE, 2010.

[39] Muhammad Umar Suleman and Mian M. Awais. Learning from demonstration in robots: Experimental comparison of neural architectures. *Robotics and Computer-Integrated Manufacturing*, 27(4):794 – 801, 2011.

[40] Masato Ito and Jun Tani. On-line imitative interaction with a humanoid robot using a dynamic neural network model of a mirror system. *Adaptive Behavior*, 12(2):93–115, 2004.

[41] M Stoica, GA Calangiu, F Sisak, and I Sarkany. A method proposed for training an artificial neural network used for industrial robot programming by demonstration. In *2010 12th International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)*, pages 831–836. IEEE, 2010.

[42] Cemil Oz and Ming C Leu. American sign language word recognition with a sensory glove using artificial neural networks. *Engineering Applications of Artificial Intelligence*, 24(7):1204–1213, 2011.

[43] P. Azad, T. Asfour, and R. Dillmann. Toward an unified representation for imitation of human motion on humanoids. In *2007 IEEE International Conference on Robotics and Automation*, pages 2558 –2563, april 2007.

[44] Dennis Herzog, Ales Ude, and Volker KrUger. Motion imitation and recognition using parametric hidden markov models. In *8th IEEE-RAS International Conference on Humanoid Robots, 2008*, pages 339–346. IEEE, 2008.

[45] Hossein Hajimirsadeghi, Majid Nili Ahmadabadi, Mostafa Ajallooeian, Babak Nadjar Araabi, and Hadi Moradi. Conceptual imitation learning: An application to human-robot interaction. *Journal of Machine Learning Research-Proceedings Track*, 13:331–346, 2010.

[46] Stefan Schaal, Jan Peters, Jun Nakanishi, and Auke Ijspeert. Control, planning, learning, and imitation with dynamic movement primitives. In *Proceedings of the Workshop on Bilateral Paradigms on Humans and Humanoids, IEEE 2003 International Conference on Intelligent RObots and Systems (IROS)*, pages 27–31, 2003.

[47] Takamitsu Matsubara, S Hyon, and Jun Morimoto. Learning stylistic dynamic movement primitives from multiple demonstrations. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1277–1283. IEEE, 2010.

[48] Minija Tamosiunaite, Bojan Nemec, Aleš Ude, and Florentin Wörgötter. Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives. *Robotics and Autonomous Systems*, 59(11):910–922, 2011.

[49] Freek Stulp, Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. Learning to grasp under uncertainty. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5703–5708. IEEE, 2011.

[50] Peter Pastor, Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, and Stefan Schaal. Skill learning and task outcome prediction for manipulation. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3828–3834. IEEE, 2011.

[51] Tomas Kulvicius, K Ning, Minija Tamosiunaite, and F Worgotter. Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting. *IEEE Transactions on Robotics*, 28(1):145–157, 2012.

[52] Miguel Prada, Anthony Remazeilles, Ansgar Koene, and Satoshi Endo. Implementation and experimental validation of dynamic movement primitives for object handover. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pages 2146–2153. IEEE, 2014.

[53] Zhaojie Ju and Honghai Liu. Fuzzy gaussian mixture models. *Pattern Recognition*, 45(3):1146–1158, 2012.

[54] V Kruger, Vadim Tikhanoff, Lorenzo Natale, and Giulio Sandini. Imitation learning of non-linear point-to-point robot motions using dirichlet processes. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2029–2034. IEEE, 2012.

[55] Antoine De Rengervé, Florent D'halluin, Pierre Andry, Philippe Gaussier, Aude Billard, et al. A study of two complementary encoding strategies based on learning by demonstration for autonomous navigation task. In *Proceedings of the tenth international conference on Epigenetic Robotics*, pages 105–112, 2010.

[56] Krishna Kumar Narayanan, Luis-Felipe Posada, Frank Hoffmann, and Torsten Bertram. Scenario and context specific visual robot behavior learning. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1180–1185. IEEE, 2011.

[57] Pedro Neto, J Norberto Pires, and A Paulo Moreira. High-level programming and control for industrial robotics: using a hand-held accelerometer-based input device for gesture and posture recognition. *Industrial Robot: An International Journal*, 37(2):137–147, 2010.

[58] Wei-Po Lee and Tsung-Hsien Yang. Combining GRN modeling and demonstration-based programming for robot control. *Neural Computing and Applications*, 20(6):909–921, 2011.

[59] Tetsunari Inamura, Iwaki Toshima, Hiroaki Tanie, and Yoshihiko Nakamura. Embodied symbol emergence based on mimesis theory. *The International Journal of Robotics Research*, 23(4-5):363–377, 2004.

[60] Volker Kruger, Dennis Herzog, Sanmohan Baby, Ales Ude, and Danica Kragic. Learning actions from observations. *IEEE Robotics & Automation Magazine*, 17(2):30–43, 2010.

[61] Sylvain Calinon, Florent D'halluin, Eric L Sauser, Darwin G Caldwell, and Aude G Billard. Learning and reproduction of gestures by imitation. *IEEE Robotics & Automation Magazine*, 17(2):44–54, 2010.

[62] Aleksandar Vakanski, Iraj Mantegh, Andrew Irish, and Farrokh Janabi-Sharifi. Trajectory learning for robot programming by demonstration using hidden markov model and

dynamic time warping. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42(4):1039–1052, 2012.

[63] Leonel Rozo, Pablo Jiménez, and Carme Torras. Force-based robot learning of pouring skills using parametric hidden markov models. In *2013 9th Workshop on Robot Motion and Control (RoMoCo)*, pages 227–232. IEEE, 2013.

[64] Andrej Gams and Ales Ude. Generalization of example movements with dynamic systems. In *9th IEEE-RAS International Conference on Humanoid Robots, 2009. Humanoids 2009*, pages 28–33. IEEE, 2009.

[65] S. Calinon. *Robot Programming by Demonstration: A Probabilistic Approach*. EPFL/CRC Press, 2009. EPFL Press ISBN 978-2-940222-31-5, CRC Press ISBN 978-1-4398-0867-2.

[66] Howard Demuth, Mark Beale, and Martin Hagan. Neural network toolbox™ 6. *User Guide*, 2008, 1992.

[67] Hao Yu and Bogdan M Wilamowski. Levenberg-marquardt training. *The Industrial Electronics Handbook*, 5:1–15, 2011.

[68] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[69] Abhishek Kar. Skeletal tracking using microsoft kinect. *Methodology*, 1:1–11, 2010.

[70] E.A. Suma, B. Lange, A. Rizzo, D.M. Krum, and M. Bolas. Faast: The flexible action and articulated skeleton toolkit. In *2011 IEEE Virtual Reality Conference (VR)*, pages 247 –248, march 2011.

[71] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC 2011*. BMVA, 2011.

[72] Y. Toda, Y. Kodai, E. Hiwada, and N. Kubota. Human motion tracking for cognitive rehabilitation in informationally structured space based on sensor networks. In *2011 IEEE International Conference on Fuzzy Systems (FUZZ)*, pages 1459 –1465, june 2011.

[73] T. Kaneko, T. Ono, and N. Munakata. Implementation of context-adaptive physical imitation between humans and robots. In *2011 IEEE RO-MAN*, pages 187 –191, aug. 2011.

[74] Z. Ren, J. Yuan, and Z. Zhang. Robust hand gesture recognition based on finger-earth movers distance with a commodity depth camera. In *ACM International Conference on Multimedia (ACM MM)*, november 2011.

[75] V. Frati and D. Prattichizzo. Using kinect for hand tracking and rendering in wearable haptics. In *2011 IEEE World Haptics Conference (WHC)*, pages 317 –321, june 2011.

[76] A. D. Wilson. Using a depth camera as a touch sensor. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 69–72, New York, NY, USA, 2010. ACM.

[77] M. Fiala and A. Ufkes. Visual odometry using 3-dimensional video input. In *2011 Canadian Conference on Computer and Robot Vision (CRV)*, pages 86 –93, may 2011.

[78] Norman Villaroman, Dale Rowe, and Bret Swan. Teaching natural user interaction using openni and the microsoft kinect sensor. In *Proceedings of the 2011 conference on Information technology education*, pages 227–232. ACM, 2011.

[79] H. J. Hsu. The potential of kinect as interactive educational technology. In *2011 2nd International Conference on Education and Management Technology IPCSIT*, volume 13, pages 334–338. IACSIT Press, 2011.

[80] G Wiselin Jiji and L Ganesan. Comparative analysis of colour models for colour textures based on feature extraction. *International Journal of Soft Computing*, 2(3):361–366, 2007.

[81] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced Lectures on Machine Learning*, pages 63–71. Springer, 2004.

[82] Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters. Model learning with local gaussian process regression. *Advanced Robotics*, 23(15):2015–2034, 2009.

[83] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1–18, 2000.

[84] Vadim Tikhanoff, Angelo Cangelosi, and Giorgio Metta. Integration of speech and action in humanoid robots: icub simulation experiments. *IEEE Transactions on Autonomous Mental Development*, 3(1):17–29, 2011.

[85] Lorenzo Natale, Francesco Nori, Giorgio Metta, Matteo Fumagalli, Serena Ivaldi, Ugo Pattacini, Marco Randazzo, Alexander Schmitz, and Giulio Sandini. The icub platform: a tool for studying intrinsically motivated learning. In *Intrinsically motivated learning in natural and artificial systems*, pages 433–458. Springer, 2013.

[86] Petar Kormushev, Sylvain Calinon, Ryo Saegusa, and Giorgio Metta. Learning the skill of archery by a humanoid robot icub. In *2010 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 417–423. IEEE, 2010.

[87] Vishnu K Nath and Stephen E Levinson. Learning to fire at targets by an icub humanoid robot. *Urbana*, 51:61801, 2012.

[88] Alex X Lee, Sandy H Huang, Dylan Hadfield-Menell, Eric Tzeng, and Pieter Abbeel. Unifying scene registration and trajectory optimization for learning from demonstrations with application to manipulation of deformable objects. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pages 4402–4407. IEEE, 2014.

[89] Jens Kober, Erhan Oztop, and Jan Peters. Reinforcement learning to adjust robot movements to new situations. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 2650, 2011.