ARQUITECTURA NEURONAL PARA ROBOTS

AFFORDANCES CONTEXTUALES CON UNA ARQUITECTURA NEURONAL EN UN ESCENARIO ROBÓTICO

Francisco Cruz *Facultad de ingeniería*

Affordances son un método efectivo para anticipar el efecto de acciones ejecutadas por un agente al interactuar con objetos. En este trabajo, presentamos un tarea robótica de limpieza usando affordances contextuales, es decir, una extensión de affordances que toma en consideración el estado actual del agente. Para predecir el efecto de las acciones ejecutadas con diferentes objetos y evitar estados de falla, implementamos una arquitectura neuronal asociativa. Resultados experimentales en un ambiente robótico simulado muestran que nuestra memoria asociativa es capaz de aprender en corto tiempo y predecir estados futuros con gran exactitud.

INTRODUCCIÓN

Cada día más, robots están siendo usados en diversos campos de aplicación y se espera que ellos puedan llevar a cabo tareas hábilmente en tiempo real. Por lo tanto, la anticipación y resolución de situaciones conflictivas que pueden conllevar a errores o tareas incompletas es una propiedad deseable en los robots teniendo como objetivo operar exitosamente en escenarios del mundo real. En este trabajo, extendemos un escenario basado en aprendizaje por refuerzo que consiste de un robot en frente de una mesa con el objetivo de limpiarla [1]. Durante la ejecución de la tarea, el robot transitará por diferentes estados ejecutando acciones y usando objetos hasta que un estado final es alcanzado. Sin embargo, existen acciones que no deberían ser ejecutadas en ciertos estados debido a que pueden conllevar a un estado de fallo y de este modo impidiendo al robot finalizar la tarea exitosamente. Para tratar este problema, utilizamos affordances [2] que son un modelo de aprendizaje que permite predecir el efecto al ejecutar una acción utilizando un objeto. Sin embargo, este esquema no considera el estado actual de el agente y por lo tanto, la información de entrada necesaria para anticipar el efecto es incompleta. Al respecto, proponemos una extensión a este modelo llamada affordances contextuales [3] que considera el estado actual como una variable de entrada adicional con el fin de predecir con exactitud el efecto de una acción usando un objeto.

Para aprender y asociar las affordances contextuales, hemos implementado una arquitectura que contiene una capa con una neuronal cuadrática compleja [4]. La arquitectura asociativa crea una grilla virtual en un plano complejo para mapear las entradas en el espacio de salida. Esta arquitectura nos permite entrenar nuestro modelo con menos iteraciones obteniendo resultados exactos en un ambiente simulado con un robot humanoide que debe limpiar una mesa con diferentes objetos.

AFFORDANCES CONTEXTUALES

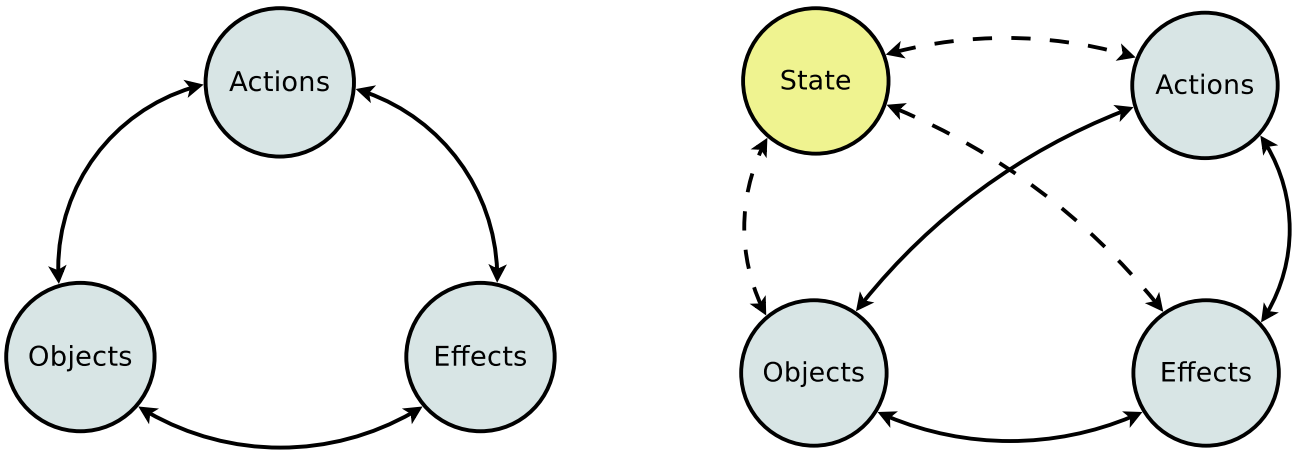
Affordances son posibilidades de acción disponibles para un agente en su ambiente [2]. Ellas representan características de la relación entre un agente y un objeto en términos de oportunidades que el objeto ofrece a un agente [5]. En robótica, affordances han sido usadas como una 3-tupla de la siguiente manera:

$$affordance :=< action, object, effect > \tag{1}$$

la cual codifica las relaciones entre sus componentes [6][7]. La Fig. 1a muestra el modelo clásico de affordances. Por lo tanto, es posible predecir el efecto usando acciones y objetos como variables de dominio, es decir:

$$effect = f(action, object) \tag{2}$$

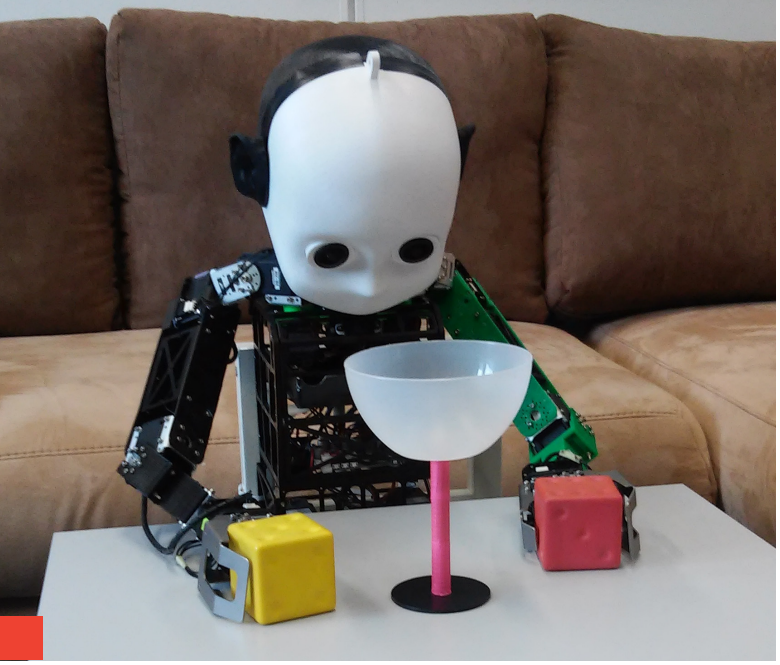
Sin embargo, aunque este modelo ha mostrado ser apropiado en muchos escenarios, affordances no incluye información de contexto la cual permite anticipar adecuadamente los efectos en todas las situaciones [8]. Es importante señalar que el hecho de poder usar o no una affordance en algún estado dado no determina la existencia de la affordance en si misma. Contrariamente, la affordance está aún presente pero no puede ser aplicada en ese estado, o puede implicar un efecto diferente usando ciertas acciones con algún objeto dado. Consideremos el escenario mostrado en la Fig 2a: una copa posee la affordance de ser tomable, así mismo un dado, pero en caso de que un agente tenga ambas manos ocupadas con dos dados, entonces el agente no podrá tomar la copa, es decir, la affordance está temporalmente no disponible.



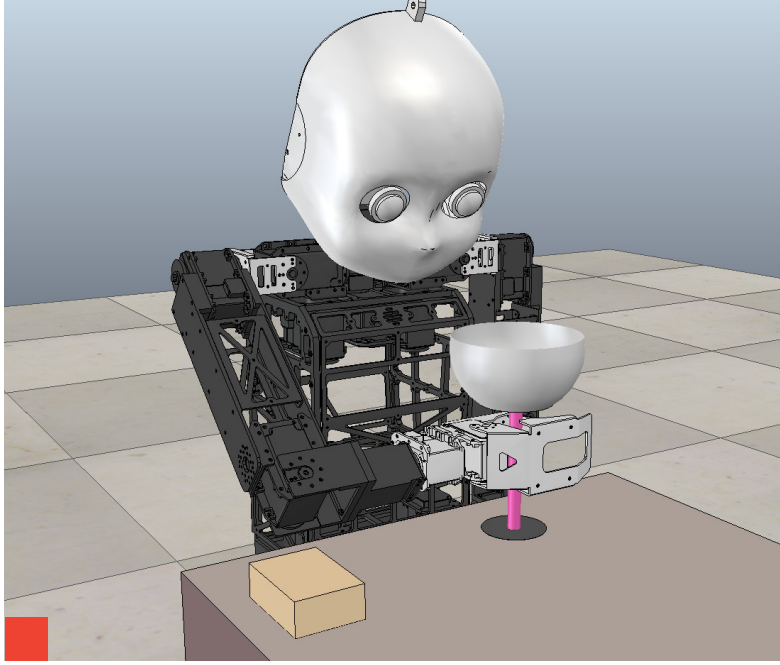
(a) Modelo de affordance **(b) Modelo de affordance contextual**

FIGURA 1

Modelo clásico de affordance (a) y la extensión propuesta (b) incluyendo información contextual.



(a) En este escenario, la affordance tomable está temporalmente no disponible.



(b) Nuestro escenario simulado con dos objetos: una copa y una esponja.

FIGURA 2

Escenario robótico real (a) y simulado (b).

Para superar este problema, es posible usar affordances contextuales donde consideramos una variable adicional para introducir información acerca del estado actual [8]. En este caso, la 3-tupla previa es ahora extendida a una 4-tupla de la siguiente manera:

(3) $$contextualAffordance :=< state, action, object, effect >$$

Usando esta tupla, podemos entonces predecir el efecto considerando la siguiente función:

(4) $$effect = f(state, action, object)$$

Por ejemplo, dado dos affordances usando la misma acción a y el mismo objeto o, pero desde diferentes estados $s_1 \neq s_2$, por lo que distintos efectos podrían ser generados $e_1 \neq e_2$. Entonces no es factible establecer diferencias entre estas affordances sin información del estado, dado que $e_1=f(a,o)$ y $e_2=f(a,o)$, lo que sugeriría que $e_1=e_2$. Por lo tanto, incorporando el estado actual $s_1 \neq s_2$, un agente podrá distinguir cada caso y, al mismo tiempo usando affordances contextuales, aprender a predecir los efectos $e_1 \neq e_2$ a través de $e_1=f(s_1,a,o)$ y $e_2=f(s_2,a,o)$ estableciendo claras diferencias entre ellos [3]. La Fig. 1b muestra el modelo de affordances contextuales incluyendo información acerca del estado actual del agente.

Al utilizar affordances, en algunos casos, el objeto puede ser una localización, por ejemplo una colina posee la affordance de ser escalable, si la acción es escalar y el objeto, o más bien la localización, es una colina. En general, usamos el término objeto para referirse a ambos, ya sean objetos o localizaciones.

Arquitectura Neuronal Asociativa

En este trabajo, hemos desarrollado un arquitectura neuronal asociativa con una neurona cuadrática de valor complejo [9] para definir una nueva grilla de dos dimensiones en el espacio de salida como presentada en [4]. Para un vector de entrada $X \in \mathbb{C}^n$, la salida compleja escalar es $y=X^*AX$, donde $A \in \mathbb{C}^{nxn}$ es la matriz de pesos y X* denota la matriz transpuesta conjugada. La salida puede ser descrita como la sumatoria de los términos individuales de los componentes de X y A :

(5) $$y = \sum_{j=1}^{n}\sum_{k=1}^{n} \bar{x}_j x_k a_{jk}$$

La regla de gradiente descendiente que minimiza el error cuadrático medio es:

(6) $$\triangle A = \alpha \varepsilon \bar{X} X^T$$

donde es una tasa de aprendizaje con un valor real pequeño. La neurona cuadrática tiene ciertas propiedades que otras neuronas usadas comúnmente no tienen, como las neuronas con funciones de transferencia sigmoidal. Por ejemplo, la salida se puede escalar y rotar simplemente multiplicando la matriz de pesos A por un factor escalar $\alpha \in \mathbb{R}$, y una rotación $e^{i\theta}$, donde θ es el ángulo de rotación. Si los vectores de entrada son normalizados, la traslación de la salida y por un escalar complejo z puede ser calculada reemplazando A por $A+zI$. Para un vector de entrada dado X, la salida deseada Y, usada en el algoritmo de aprendizaje, es definida como el punto de intersección más cercano de las lineas de la grilla del plano complejo. En la práctica una función Ψ es definida tal que redondee al entero más cercano para las lineas de la grilla a una distancia fija δ en ambas direcciones:

(7) $$\Psi(Y) = \frac{round(\delta Re(Y))}{\delta} + i\frac{round(\delta Im(Y))}{\delta}$$

Esta función crea una grilla virtual donde la salida se ajusta a la esquina más cercana en la grilla. El algoritmo de entrenamiento es definido de la siguiente manera:

1. Inicializar los pesos de la neurona con valores aleatorios
2. Calcular Y
3. Calcular $d=\Psi(Y)$
4. Actualizar los pesos de la neurona usando Eq. 6

En cada iteración, los pasos (2) al (4) son ejecutados para todos los vectores de entrada, de tal forma que un conjunto de datos en el espacio de entrada será mapeado a una región similar dentro del

espacio de salida debido a la continuidad de la función de activación. El criterio de término puede ser un número fijo de iteraciones, una tasa de aprendizaje decreciente o un error cuadrático medio mínimo sobre todas las entradas.

Escenario Robótico

La tarea consiste en un robot parado en frente de una mesa con el objetivo de limpiarla. El robot puede usar uno de sus brazos y su agarradera para manipular un conjunto de objetos con el fin de completar la tarea de limpieza. Para esta tarea, definimos objetos, locaciones y acciones. El escenario incluye dos objetos: una esponja y una copa. La mesa está dividida en tres zonas, la zona izquierda y derecha de la mesa y una posición adicional llamada home donde el robot puede colocar la esponja durante la ejecución de la tarea. El robot puede ejecutar cuatro acciones: get <*object*> (tomar un objeto), drop <*object*> (soltar un objeto), goto <*location*> (ir a una localización), y clean (limpiar) la sección de la mesa donde el brazo del robot esta localizado en ese momento. La tarea de limpieza robótica en un escenario simulado es representada en la Fig. 2b.
Cada estado del robot en el escenario toma en consideración cuatro variables:

TABLA 1
Representación de los datos de entrenamiento
usados por la arquitectura neuronal asociativa para clasificación.

DATA REPRESENTATION															
Side conditions					**Locations**				**Actions**					**Objects**	
dd	1	0	0	0	home	1	0	0	get	1	0	0	0	sponge	1	0
dc	0	1	0	0	left	0	1	0	drop	0	1	0	0	cup	0	1
cd	0	0	1	0	right	0	0	1	goto	0	0	1	0	free	0	0
cc	0	0	0	1	none	0	0	0	clean	0	0	0	1			

1. La posición de la mano del robot
2. El objeto en la mano del robot (si alguno)
3. La posición de la copa
4. La condición de cada lado de la mesa, es decir, si la superficie está limpia o sucia

El vector de estado está representado como:

(8) $s_t =< handPosition, handObject, cupPosition, sideCondition >$

Sin embargo, desde ciertos estados el robot podría ejecutar acciones que conlleven a estados de fallo, es decir, estados desde donde no es posible completar la tarea. Por ejemplo asumamos el estado actual como s_t =<*right, sponge, right, (dirty, dirty)*>, es decir, la copa se encuentra en el lado derecho de la mesa y la mano del robot se encuentra sobre ella sosteniendo la esponja. Entonces, si el robot limpia la sección derecha de la mesa podría romper la copa y, por lo tanto, no es posible finalizar la tarea de limpieza desde el siguiente estado s_{t+1}.

Codificamos todas las variables involucradas como se muestran en la Tabla 1, donde mostramos la representación de los datos para *side conditions, locations, actions y objects*. Dentro de *side conditions,* las letras *d (dirty)* y *c (clean)* representan el hecho de que la superficie se encuentre sucia o limpia respectivamente.

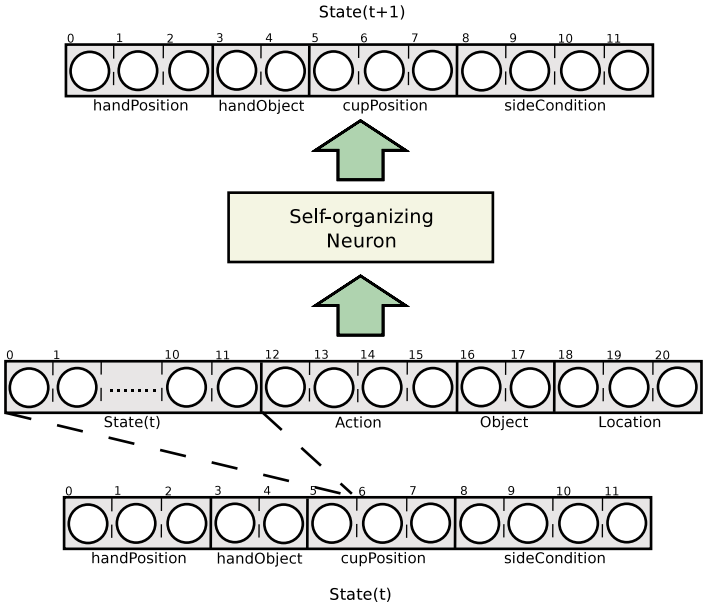


FIGURA 3

Arquitectura neuronal asociativa para la predicción del siguiente estado. En nuestro escenario, el estado alcanzado por el robot representa el efecto de la affordance.

RESULTADOS EXPERIMENTALES

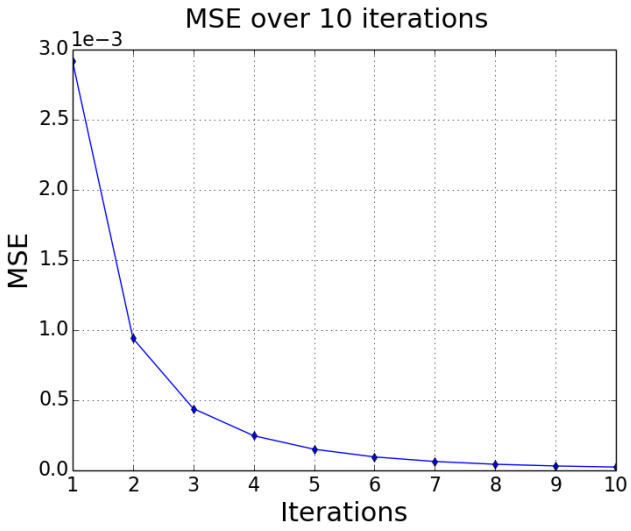
Nuestra estrategia desarrollada usa affordances contextuales para predecir el efecto de una acción ejecutada por el robot. Utilizamos la representación mostrada en la Tabla 1 para generar el conjunto de entrenamiento. Como entrada, utilizamos vectores con 21 variables conteniendo información acerca del estado actual, la acción, el objeto y/o la localización. Cada estado está contenido en las primeras 12 componentes del vector de entrada considerando las cuatro variables que definen a un estado (ver Fig. 3). Nuestra arquitectura contiene una capa neuronal asociativa que mapea el estado actual del sistema en el efecto esperado, que corresponde al efecto de las affordances contextuales codificado como un vector de 12 componentes que representa el siguiente estado. Cuando una acción conlleva a un estado de fallo, todos los componentes de el vector de salida son iguales a cero. Los datos fueron creados considerando todos los posibles estados junto a las acciones y objetos (o locaciones). El número total de ejemplos son 368 instancias para el entrenamiento de la capa asociativa.

Durante el entrenamiento, asociamos la etiqueta (o label) del estado a la salida deseada con fines de clasificación. Posterior a la fase de entrenamiento, cuando un nuevo ejemplo es presentado a la neurona, calculamos y retornamos la etiqueta del estado que minimiza . En nuestra implementación, utilizamos y una tasa de aprendizaje decreciente de la siguiente manera:

(9) $$\alpha_t = \alpha_0 * e^{\frac{-t(t+3)}{k}}$$

donde es el número de iteración, α_0 = 0.01 y k = 5000 .

Experimentos muestran que nuestra arquitectura con una capa asociativa es capaz de clasificar todas las instancias correctamente luego del entrenamiento. El error cuadrático medio disminuyó desde 2.92e-3 a 2.37e-5 después de 10 iteraciones como se muestra en la Fig. 4a. La distribución final de la salida después de 10 iteraciones es mostrada en la Fig. 4b, donde el eje x y el eje y son la partes real e imaginaria del plano complejo respectivamente.



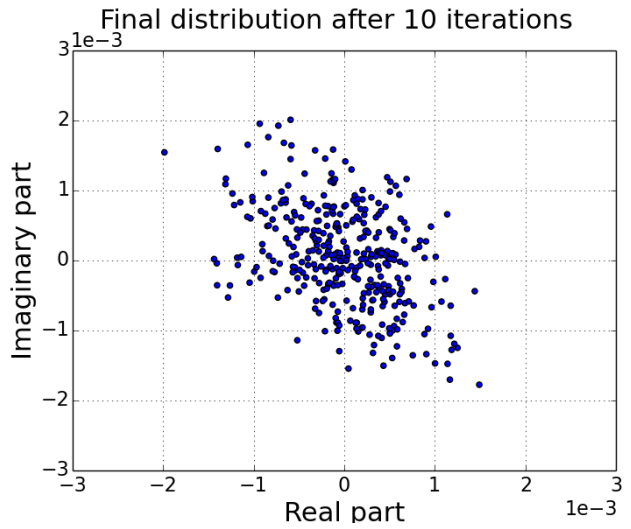


FIGURA 4

Error de entrenamiento (a) y distribución final (b) de la capa asociativa.

CONCLUSIONES Y TRABAJOS FUTUROS

La arquitectura propuesta ha sido capaz de predecir satisfactoriamente el efecto de ejecutar una acción usando un objeto utilizando affordances contextuales. Hemos utilizado información adicional del estado para distinguir diferentes situaciones en un escenario robótico de limpieza y hemos evitado estados de falla finalizando la tarea exitosamente. La arquitectura asociativa compleja permitió mapear los vectores de entrada en estados validos con pocas iteraciones de entrenamiento, lo cal representa una ventaja para aplicaciones de aprendizaje en linea donde el tiempo de respuesta juega un rol crucial.

Como trabajo futuro, el escenario experimental puede ser extendido para utilizar plataformas robóticas reales obteniendo el vector de entrada a través de un sensor de visión y el vector de salida desde el estado real del robot después de ejecutar la acción en el escenario de limpieza.

MÁS INFORMACIÓN

El presente reporte está basada en nuestro anteriores artículos: *Training agents with interactive reinforcement learning and contextual affordances, publicado en la revista IEEE Transactions on Cognitive and Developmental Systems (TCDS) [10] y Learning contextual affordances with an associative neural architecture, publicado en los Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN) [3].* Para mayores detalles, por favor consultar los artículos mencionados donde además es posible encontrar experimentos adicionales aplicados a un escenario de aprendizaje por refuerzo interactivo.

BIBLIOGRAFÍA

[1] F. Cruz, G. I. Parisi, J. Twiefel, and S. Wermter. Multi-modal integration of dynamic audiovisual patterns for an interactive reinforcement learning scenario. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 759–766, 2016.

[2] J. J. Gibson. The Ecological Approach to the Visual Perception of Pictures. Boston: Houghton Mifflin, 1979.

[3] F. Cruz, G. I. Parisi, and S. Wermter. (2016). Learning contextual affordances with an associative neural architecture. Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), pp. 665–670, 2016.

[4] G. Georgiou and K. Voigt. Self-organizing maps with a single neuron. Proceedings of the International Joint Conference on Neural Networks (IJCNN), pp. 1–6, 2013.

[5] T.E. Horton, A. Chakraborty, and R. S. Amant. Affordances for robots: a brief survey. AVANT: Journal of Philosophical-Interdisciplinary Vanguard, Vol. 2, No.1, pp. 70–84. 2012.

[6] Self-organizing maps with a single neuron E. Şahin, M. Çakmak, M. R. Doğar, E. Uğur, and G. Üçoluk. To afford or not to afford: A new formalization of affordances toward affordance-based robot control. Adaptive Behavior, Vol. 15, No. 4, pp. 447–472, 2007.

[7] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory-motor coordination to imitation. IEEE Transactions on Robotics, Vol. 24, No. 1, pp. 15–26, 2008.

[8] M. Kammer, T. Schack, M. Tscherepanow, and N. Yukie. From affordances to situated affordances in robotics - Why context is important. Frontiers in Computational Neuroscience, Conference Abstract IEEE ICDL-EpiRob, Vol. 5, No. 30, 2011.

[9] G. Georgiou. Exact interpolation and learning in quadratic neural networks. Proceedings of the International Joint Conference on Neural Networks (IJCNN), pp. 230–234, 2006.

[10] F. Cruz, S. Magg, C. Weber, and S. Wermter. Training agents with interactive reinforcement learning and contextual affordances. IEEE Transactions on Cognitive and Developmental Systems, Vol. 8, No. 4, 271–284, 2016.