Ciencia Latina
Revista Multidisciplinar

DOI: https://doi.org/10.37811/cl_rcm.v6i6.4071

Subsistema de locomoción de módulos funcionales en crecimiento

PhD Jérôme Leboeuf Pasquier
jerome.lep@gmail.com
https://orcid.org/0000-0001-7613-277X

PhD Norma Ramírez Hernández
norma.rhernandez@academicos.udg.mx
https://orcid.org/0000-0002-2022-0444

Universidad de Guadalajara
Guadalajara Jalisco – México

RESUMEN

El enfoque del módulo de crecimiento funcional (GFM, por sus siglas en inglés) permite la generación gráfica de controladores basados en aprendizaje. Un controlador se crea interconectando y configurando cuatro tipos de componentes: objetivos globales, módulos de actuación, módulos de detección y sensaciones. La lista de sensaciones describe la retroalimentación del bucle de control, el conjunto de objetivos globales especifica las motivaciones intrínsecas del controlador y los módulos de actuación y detección desarrollan dinámicamente sus respectivas funcionalidades mientras interactúan con el entorno. Con el fin de posicionar a GFM como un paradigma potencial, este artículo describe el desarrollo de un subsistema de locomoción basado en el aprendizaje, este subsistema está compuesto por dos Módulos Actuales a cargo de la motricidad gruesa y fina, respectivamente. Se describe la implementación de estos nuevos módulos y se ilustra su funcionalidad: el primero está dedicado a mapear el entorno y el segundo a evitar obstáculos.

Palabras clave: *módulos de crecimiento funcional; control basado en el aprendizaje; robótica cognitiva del desarrollo; mapeo de realización;*

Correspondencia: jerome.lep@gmail.com
Artículo recibido 25 noviembre 2022 Aceptado para publicación: 25 diciembre 2022
Conflictos de Interés: Ninguna que declarar

Cómo citar: Leboeuf Pasquier, P. J., & Ramírez Hernández, P. N. (2022). Subsistema de locomoción de módulos funcionales en crecimiento. Ciencia Latina Revista Científica Multidisciplinar, 6(6), 9298-9321. https://doi.org/10.37811/cl_rcm.v6i6.4071

A growing functional modules locomotion subsystem

ABSTRACT

The Growing Functional Modules (GFM) approach allows the graphic generation of learning-based controllers. A controller is created by interconnecting and configuring four kinds of components: Global Goals, Acting Modules, Sensing Modules, and Sensations. The Sensations list describes the control loop feedback, the set of Global Goals specifies the intrinsic motivations of the controller, and the Acting and Sensing Modules dynamically develop their respective functionalities while interacting with the environment. In order to position GFM as a potential paradigm, this paper describes the development of a learning-based locomotion subsystem. This subsystem is consists of two new Acting Modules in charge of gross and fine motor skills, respectively. The implementation of these new modules is described and their functionality illustrated: The first is dedicated to mapping the environment and the second one to obstacle avoidance.

Keywords: *growing functional modules; learning-based control; cognitive developmental robotics; embodiment mapping;*

INTRODUCCIÓN

La robótica de desarrollo cognitivo (CDR), propuesta en (Asada, 2001) y ampliamente descrita en (Asada M. e., 2009; Lungarella, 2003; Meeden, 2006), se basa en las habilidades de una inteligencia de propósito general introducida por (Brooks, 1998) acerca de la personificación, desarrollo, interacción social e integración. La CDR se enfoca en diseñar arquitecturas cognitivas para el control de robots que permitan aprender mientras se realizan tareas complejas en entornos complejos. Además, se debe poder interactuar y trabajar codo con codo con los humanos. Algunos marcos como los propuestos en (Franklin, 2013; Goertzel, 2009; Bach, 2012) facilitan la construcción de arquitectura cognitiva a través de una interconexión de módulos, cada uno integrando un proceso de desarrollo. Los paradigmas correspondientes construyen una representación del conocimiento creciente utilizando redes neuronales que inicialmente están vacías y se entrenan o construyen utilizando la información recopilada a través de la exploración de entornos complejos utilizando las habilidades sensoriales y de actuación del robot.

De manera similar, el enfoque GFM, presentado en (Leboeuf Pasquier, 2005), permite el diseño de controladores basados en aprendizaje mediante la interconexión gráfica de cuatro tipos de componentes. Este paradigma potencial, desarrollado durante los últimos años, ha sido descrito recientemente en (Leboeuf-Pasquier, 2015). GFM puede entenderse como un cerebro artificial controlador basado en el aprendizaje que brinda una respuesta en tiempo real a los eventos que detecta. Además, durante este proceso almacenan experiencia relevante que luego puede ser utilizada. La arquitectura de GFM se basa en un bucle de control clásico (ver fig. 1), el controlador envía un comando para modificar la posición de un actuador específico de acuerdo con la amplitud correspondiente y luego recibe como retroalimentación una secuencia de n valores del sensor, 2 eventos discretos de una señal de audio y una matriz de píxeles de 8x8 que corresponde a una característica visual. Las referencias vienen dadas por una lista de Objetivos Globales internos. El sistema de control se ha dividido en dos áreas (Actuador y sensor), cada área está provista de habilidades de aprendizaje, funcionalidad y conectividad con otros componentes. En un video didáctico (Leboeuf-Pasquier, 2015), se muestra el diseño y la ejecución de un controlador GFM simple que realiza la búsqueda de rutas; recomendable para comprender el enfoque GFM.

Actualmente, a pesar de algunas aplicaciones de control exitosas, incluido el equilibrio humanoide (Leboeuf-Pasquier J. , 2015), el video o el subsistema auditivo (Leboeuf-Pasquier e. a., 2013), parece imposible evaluar la capacidad de diseño de GFM debido a la falta de un modelo matemático; el desarrollo de un subsistema de locomoción basado en el aprendizaje descrito en este artículo es un intento de resolver esta deficiencia.

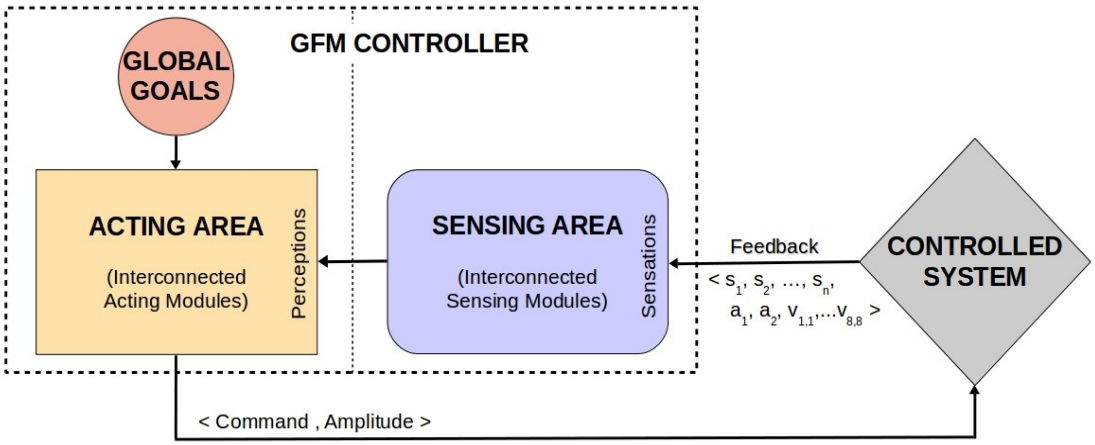

Figura 1. *El controlador GFM (dentro de las líneas de puntos), utiliza un bucle de control clásico para comunicarse con el robot: cada comando de salida va seguido de una retroalimentación correspondiente a una secuencia de sensores*

A diferencia de los subsistemas visuales o auditivos mencionados anteriormente, el GFM está concebido sobre un esquema de control con habilidades de aprendizaje que incluyen el mapeo, localización, búsqueda de rutas y la evasión de obstáculos. Estas habilidades se realizan en dos etapas: la primera es Localización y Mapeo Simultáneos (SLAM), que constituye la principal habilidad del subsistema de locomoción; el segundo es la evasión de obstáculos requerida para compensar la representación aproximada del mapeo. Se han desarrollado dos nuevos módulos de actuación para realizar estas habilidades: el primero está dedicado a mapear el entorno y el enrutamiento (*AmDmr-Mapeo Dinámico y Enrutamiento*); el segundo está a cargo de evitar obstáculos (AmNfp-Posicionamiento Neuro-Fuzzy), que se describen a continuación.

METODOLOGÍA

1. Arquitectura interna de un Módulo Actuante

Cada módulo de actuación abarca una funcionalidad específica que debe adquirirse dinámicamente a través de la interacción del robot con el entorno. La siguiente lista

ilustra el tipo de funcionalidades actualmente implementadas:

- *AmCi*: inferencia causal; su implementación se describe en (Pasquier, 2007)
- *AmCtg*: trayectoria cíclica de una extremidad involucrada en tareas como caminar o nadar.
- *AmTlp*: posiciona una extremidad de tres articulaciones para operaciones de agarre.
- *AmRtr*: realiza regulación en tiempo real para tareas de control automático, descrito en (Pasquier J. L., 2006).
- *AmHog*: genera una heurística que optimiza el comportamiento de alto nivel.

Conceptualmente, la funcionalidad de un módulo de actuación debe ser lo más "simple" posible para integrarse fácilmente en diferentes tipos de tareas de control. Para permitir la interconexión con los otros componentes del GFM, se debe construir un módulo activo según se muestra en la fig. 2. Donde en cada ciclo de propagación del nuevo módulo *AmDmr* debe completar cuatro pasos, cada paso se implementa con el método C++ de la clase *ActingModule*. Los pasos 2 y 3 caracterizan la funcionalidad del módulo: al diseñar un controlador, esta funcionalidad se asigna a cada módulo al especificar el tipo de parámetro.

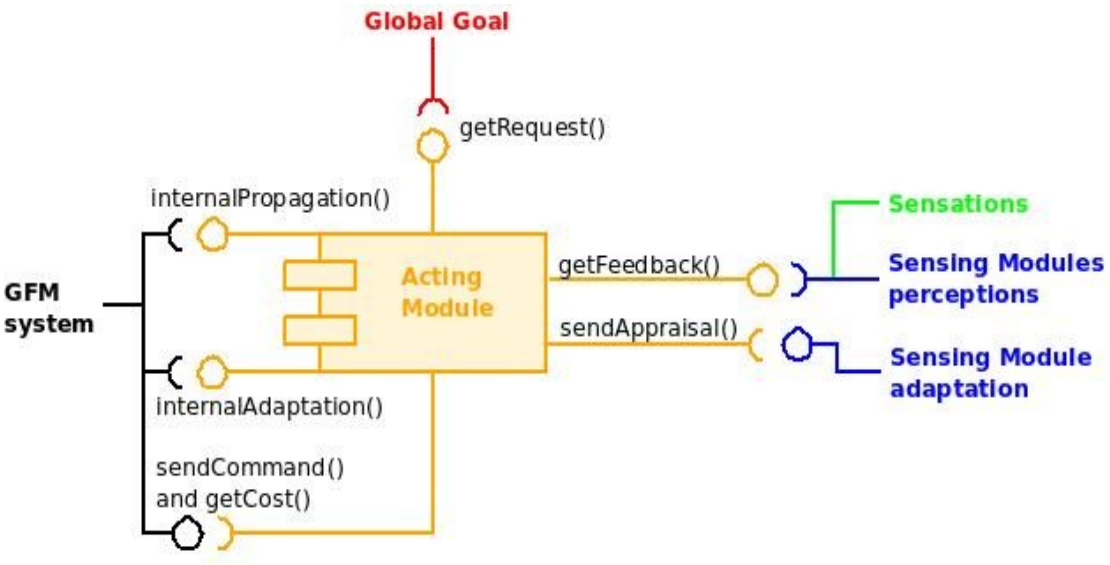

Figura 2. *Diagrama de componentes de un módulo de actuación, cada color está asociado a una clase de componente (rojo para el objetivo global, verde para sensores, azul para el módulo de detección y naranja para el módulo de actuación).*

1. *getRequest()*: El primer paso corresponde a la recepción de una solicitud entrante ; el valor de la solicitud no cambiará hasta que se satisfaga o hasta que el módulo activo emita un comando de Retiro. Una solicitud se satisface cuando su valor es igual al primer valor de la secuencia de retroalimentación.
2. *getFeedback(), getCost(), internalAdaptation()*: El segundo paso corresponde a la recepción de una retroalimentación compuesta por las Sensaciones y/o Percepciones del Área Sensorial. Esta retroalimentación reporta los efectos producidos por el comando previamente enviado por el módulo, que se "adapta" para ajustarse a esta retroalimentación. La primera sensación de retroalimentación siempre indica el costo en términos de energía utilizada por el robot para ejecutar el último comando. La propagación interna del módulo trata permanentemente de minimizar este costo.
3. *internalPropagation*(): El tercer paso consiste en la propagación interna del módulo para generar la secuencia de comandos con mayor probabilidad de satisfacer la solicitud actual. Para generar estos comandos, el método *internalPropagation()* procesa la representación dinámica del entorno construida por el método *internalAdaptation()*.
4. *sendAppraisal(), sendCommand()*: Durante el cuarto paso, el primer comando de la secuencia y su amplitud se envían al método de aplicación robótica, mientras que se envía una evaluación a los módulos sensores asignados. Para el *AmDmr*, el comando puede ser avanzar o retroceder y su amplitud un múltiplo de 45 grados. Esta valoración indica la percepción esperada por el módulo actuante en el próximo ciclo y es utilizada por los módulos sensores para realizar su propio proceso de adaptación. Este proceso de cuatro pasos se resume en el algoritmo 1

Cada módulo de actuación está asociado con una clase de C++ heredada de una clase base llamada *ActingModule* cuyos métodos de comunicación, propagación y adaptación son puramente virtuales. En consecuencia, el método de la clase derivada codifica un proceso que debe cumplir con el método genérico implementado en la clase base. Se aplica un principio similar a cada uno de los cuatro componentes: la herencia de una clase base refuerza la viabilidad de la conectividad entre los componentes porque los nuevos se construyen de acuerdo con un esquema similar al que se ilustra en la fig. 2. La

funcionalidad de un módulo se implementa principalmente por dos métodos predominantes *internalPropagation()* e *internalAdaptation()*.

```
procedure ActingModuleGenericInternalProcess( Request, Feedback ) ;
var Command
begin
      while not interrupted do
      begin
            Request = GetRequest() ;
            Feedback = GetFeedback() ;
            while not satisfied( Request )
            begin
                  if ThereIsCost() then
                        InternalAdaptation() ;
                  Command = InternalPropagation() ;
                  Send( Command) ;
                  Feedback = GetFeedback() ;
                  Cost = GetCost() ;
            end
      end
end ;
```

Algoritmo 1. *Se describe el proceso interno genérico de todos los módulos activos donde los métodos internalPropagation() e internalAdaptation() definen la funcionalidad del módulo; una Solicitud viene dada por una Meta Global o un Módulo Actuante superior; el FeedBack es proporcionado por el Área de Detección; se devuelve un Comando con su Amplitud correspondiente*

1. Módulo *AmDmr*

La funcionalidad del módulo de actuación *AmDmr* consiste en el mapeo dinámico del entorno y el enrutamiento del robot. La funcionalidad del módulo *AmDmr* se analiza en las siguientes subsecciones, donde se describen los métodos internalPropagation() e internalAdaptation().

a) Método de propagación AmDmr (internalPropagation())

Con cada ciclo, de acuerdo con la retroalimentación y la solicitud actuales, el método de propagación de un módulo activo determina qué comando de un conjunto predefinido se activará en la salida. A falta de conocimiento previo sobre la aplicación, la decisión resulta del proceso de propagación aplicado a la representación interna actual. En el caso del módulo AmDmr, la representación interna es una matriz cuyos elementos corresponden a una posición aproximadamente del tamaño del robot (Almansa-Valverde, 2012). Se usa comúnmente en CDR para realizar mapeo y SLAM en entornos desconocidos (Einhorn, 2015). Cada posición de la matriz se puede etiquetar como vacía, ocupada o inexplorada. En caso de ocupado, un valor indica qué tan

estática/dinámica es la posición de acuerdo con la retroalimentación anterior. En el proceso interno del módulo, la solicitud de entrada especifica la posición a alcanzar (objetivo); con cada ciclo, el comando de salida activado por el módulo indica la dirección en la que se debe realizar el siguiente paso. La retroalimentación especifica si la nueva posición es la misma que la esperada. En este caso, se activa el siguiente comando de dirección de paso a la meta. De lo contrario, se calcula una nueva ruta y se activa un nuevo comando de paso en la aplicación. El mapeo resultante permite una definición de los posibles caminos según el algoritmo 2.

```
Procedure AmDmrInternalPropagation( Request, Feedback )
var Result composed of a Command and its corresponding Amplitude
begin
while not interrupted
       begin
       while the requested position is not reached
              begin
              Get( Request ) from a Global Goal or an upper Acting Module ;
              Get( Feedback )from the Sensing Area ;
              Perform the adaptation of the map according to Feedback ;
              if Request changed or Feedback is different from expected
                     begin
                     Find a secure Path composed of unexplored positions ;
                     if Path not found
                            begin
                            Find the most feasible Path computing reliance ;
                            if Path not found
                                   begin
                                   Goback() performing backwards steps ;
                                          if Goback fails
                                          begin
                                          The trapped robot must give up ;
                                          Command is Withdrawal ;
                                          Wait for help ;
                                          End
                                   Get( NextStep ) from the current Path ;
                                   Get( Command, Amplitude) from NextStep ;
                                   end
                            else
                                   begin
                                   Get( NextStep ) from the feasible Path ;
                                   Get( Command, Amplitude) from NextStep ;
                                   end
                            end
                     else
                            begin
                            Get( NextStep ) from the secure Path ;
                            Get( Command, Amplitude) from NextStep ;
                            end
                     end
              end
       SendRobot( Command, Amplitude ) ;
       Get( Feedback ) from the Sensing Area ;
       Get( Cost ) from Feedback ;
       end
end
```

Algoritmo 2. *Este algoritmo es una versión del algoritmo genérico 1 diseñado para cumplir con la funcionalidad de mapeo/enrutamiento. Cuando la ruta actual falla, se calcula una nueva segura compuesta de posiciones vacías o inexploradas. Si no se encuentra ninguno, la propagación interna utiliza los valores de confianza para rastrear la ruta más segura. Si esto vuelve a fallar, se aplica el procedimiento Goback()*

Se activa un comando de retirada después de que se haya intentado repetidamente la secuencia interna de control "*if*" y si todas las rutas posibles han fallado, incluso cuando se han tenido en cuenta los obstáculos dinámicos. Una falla de todos los intentos significa que el robot ha sido conducido a un callejón sin salida, por lo tanto, se aplica el procedimiento *Goback()*. Fallar nuevamente significa que el robot está atrapado; esta situación se produce cuando el robot se ha metido en un callejón sin salida y todos los problemas se han cerrado tras él. Para encontrar un camino, se aplica un algoritmo inspirado en orientación del robot cuando no puede girar sobre sí mismo debido a la limitación del ángulo del volante (Tian, 2013; Duchoň, 2014), donde la dirección actual dada por la retroalimentación ofrece una mayor precisión y el comando de salida puede asumir tres valores: Adelante, Atrás y Retirada; la amplitud asociada con los comandos Adelante y Atrás especifica el ajuste del ángulo del volante.

b. Método de adaptación AmDmr (internalAdaptation())

Como se mencionó, la representación del mapa se actualiza cada vez que se proporciona una retroalimentación. A cada posición del mapa se le asigna una de las siguientes etiquetas: inexplorada, vacía u ocupada especificando el estado actual de esta posición. Se asigna un segundo atributo estático o dinámico para indicar si la posición parece estar "permanentemente ocupada" o no. Un tercer atributo de confianza refleja la probabilidad de cruzar con éxito esta posición. Finalmente, un atributo etiquetado como inhibido prohíbe cualquier nuevo intento de cruzar esta posición durante un corto período de tiempo. El proceso de adaptación se resume en:

- Una posición inexplorada se etiqueta como vacía y se le asigna un estado dinámico.
- Una posición que inicialmente no se puede alcanzar se ocupa con un valor de confianza bajo y un estado estático.
- La confianza de una posición ocupada etiquetada como estática aumenta ligeramente con cada intento fallido hasta alcanzar el valor máximo de 1. Al calcular una ruta factible, el valor de confianza se interpreta como la probabilidad de cruzar con éxito esta posición.
- Cuando se alcanza una posición estática, su dependencia se reduce considerablemente y su estado se actualiza a dinámico.

- La dependencia correspondiente a una posición etiquetada dinámica pertenece al intervalo unitario [0,1]; cuanto menor sea el valor, mayor será la probabilidad de cruzar con éxito esta posición.
- Cuando falla un intento de cruzar una posición, esta posición no puede realizar ningún nuevo intento durante un período corto cuyo valor se calcula en función de la confianza: cuanto mayor sea la confianza, mayor será la demora.

Por tanto, el mapa dinámico resultante indica la dependencia asociada a un camino que corresponde a una secuencia de posiciones; esto permite calcular la ruta más factible desde la posición actual del robot hasta la posición solicitada. Cada vez que una posición vacía resulta estar ocupada, el mapa se actualiza y se calcula una nueva ruta. Sin embargo, un conjunto de ocho posibles direcciones que corresponden a los comandos predefinidos, no ofrece suficiente precisión para controlar con precisión la ruta del robot. Por tanto, un nuevo módulo con un proceso interno basado en lógica neuro-difusa se encargará de suavizar el ajuste-comando resultante del volante teniendo en cuenta el entorno real. Este módulo, denominado *AmNfp*, se describe en la siguiente sección.

2. Módulo actuador *AmNfp*

En un entorno real, los obstáculos rara vez se alinean o cuadran como se representa en el mapeo del módulo AmDmr. Aunque esta representación simplificada y generada dinámicamente facilita el cálculo de la planificación de la ruta, la ruta resultante debe suavizarse para adaptarse a una situación real. Este proceso se ilustra en la fig. 3, la imagen de la izquierda muestra la representación aproximada generada por el módulo *AmDmr* con flechas verdes. El área sombreada se amplía en la figura de la derecha, donde la ruta mejorada seguida por el robot se indica con puntos amarillos. Este recorrido más suave lo produce el módulo AmNfp cuya retroalimentación, proveniente de sensores más precisos, ultrasónicos en la actualidad, permite ajustar el volante en cada paso; en cada paso de *AmDmr* (flecha verde), se solicita al módulo *AmNfp* active una secuencia de pasos (puntos amarillos).

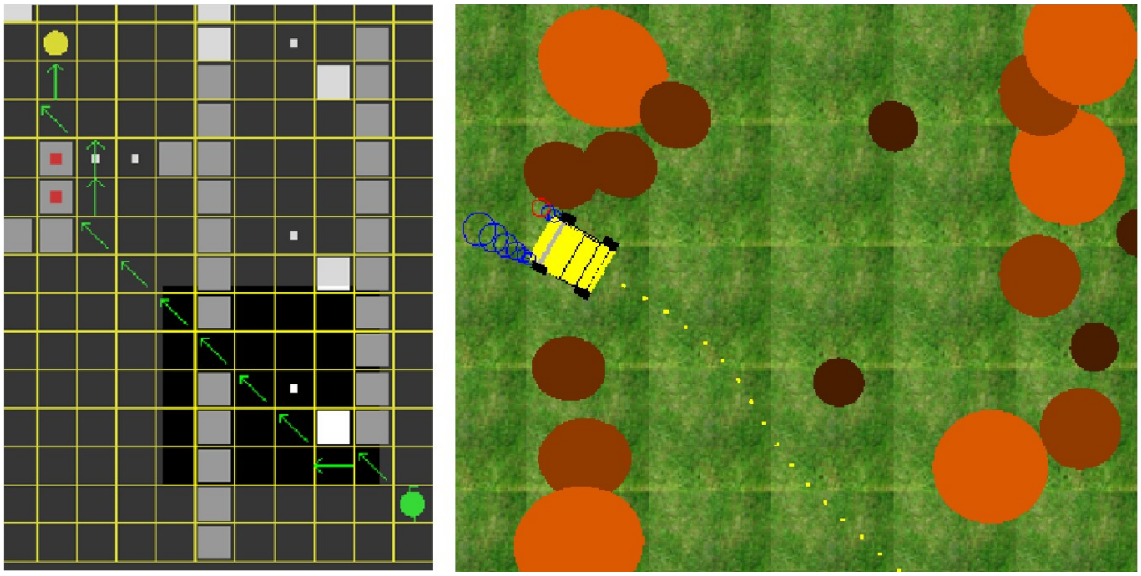

Figura 3. *La imagen de la izquierda representa la ruta aproximada planificada por el módulo AmDmr. La imagen de la derecha corresponde al área sombreada de la imagen de la izquierda: Esta amplificación muestra el ajuste resultante del camino realizado por el módulo AmNfp para evitar obstáculos*

De acuerdo con el método *internalPropagation()* del módulo activo, este proceso finaliza cuando se satisface la solicitud *AmDmr* o cuando el módulo *AmNfp* falla y envía un comando de retiro. Este comando finaliza la propagación interna del módulo e indica al módulo AmDmr que no se puede alcanzar la posición solicitada debido a la presencia de obstáculos inesperados. Luego se actualiza el mapeo de AmDmr y se calcula una nueva ruta tentativa, donde los métodos *internalPropagation()* e *internalAdaptation()* definen completamente la funcionalidad del módulo *AmNfp*.

3. Método de propagación AmNfp

La solicitud de entrada dada al módulo *AmNfp* es producida por el módulo *AmDmr*; es decir, el comando que indica avanzar o retroceder un paso en una de las ocho direcciones posibles constituye la solicitud del módulo *AmNfp*. Esta solicitud, junto con la retroalimentación obtenida de los sensores ultrasónicos, permite que *AmNfp* ajuste la trayectoria del robot. Por lo tanto, el comando producido por el módulo *AmDmr* se reemplaza por una secuencia de comandos-pasos más pequeños que permiten evitar los obstáculos cercanos a la trayectoria que se basa en la lógica difusa. Con cada ciclo interno y mientras no se alcance el paso solicitado, las ruedas del robot se inclinan con

mayor precisión. Esta ganancia en precisión resulta de la aplicación de las reglas difusas a las siguientes cuatro variables de entrada:

- *GoalAngle:* corresponde a la dirección del bloque a alcanzar, esta dirección viene dada por la solicitud del módulo AmDmr.
- *SteerAngle:* especifica la orientación de las ruedas del robot.
- *LeftSensor:* indica la proximidad de un objeto detectado por el sensor ultrasónico izquierdo.
- *RightSensor:* indica la proximidad de un objeto detectado por el sensor ultrasónico derecho

La variable de salida del sistema difuso *SteeringWheel* estipula la corrección que se debe aplicar a la orientación de las ruedas del robot. Independientemente de su implementación interna, ya sea mediante un grafo dinámico, una red neuronal o un sistema difuso, la arquitectura interna de un Módulo de Actuación debe cumplir con los requisitos expuestos en el apartado de la Arquitectura interna de un Módulo Actuante.

El método *internalPropagation()* del módulo *AmNfp* se implementa como se describe en el algoritmo 3. En cada instante t se actualizan *GoalAngle* y los valores de retroalimentación *SteerAngle*, *LeftSensor* y *RightSensor*. Los valores de *GoalAngle* y *SteerAngle* se validan de acuerdo con *veryNegative(), negative(), zero(), positive()* y *veryPositive()* representadas en la fig. 4, mientras que las otras dos variables de entrada *LeftSensor* y *RightSensor* se validan de acuerdo con las funciones *close()* y *far()*. Los operadores difusos *and* y *or* se definen utilizando las funciones *min()* y *max(),* respectivamente; el valor resultante, asignado a la variable de salida *SteeringWheel*, corresponde a la amplitud del comando enviado al sistema de dirección del robot por el módulo AmNfp.

Las reglas R1 a R25 descritas en el algoritmo 4, definen la funcionalidad del sistema de control difuso a cargo de guiar suavemente el robot hacia la meta, mientras que las reglas R26 a R28b están diseñadas para evitar obstáculos (aumentan el valor de *SteeringWheel* cuando un sensor ultrasónico señala la proximidad de un objeto a través de sus valores de retroalimentación *LeftSensor* y *RightSensor)*. La información proporcionada por estos sensores puede ser insuficiente para decidir si el robot debe rodear el obstáculo por el lado izquierdo o por el derecho, por lo tanto, eventualmente se hacen dos intentos: primero en el lado izquierdo aplicando la regla R28a y luego, en

caso de fallar, en el lado derecho aplicando la regla 28b. Finalmente, los valores nítidos *turnFullLeft*, *turnLeft, goStraight, turnRight, turnFullRight* asignados a la variable de salida *SteeringWheel*, se adaptan de acuerdo con la respuesta mecánica del volante.

```
procedure AmNfpInternalPropagation( GoalAngle, SteerAngle,
       Left/RightSensor feedback values )
var Command and Amplitude that will modify SteerAngle
while not interrupted
begin
       get( Request ) ;
       get( Feedback ) ;
       while the Goal is not reached that is the request is not satisfied
       begin
              if Cost has been received
                     Perform internal adaptation ;
              else Going forward
              begin
                     Get( Amplitude ) applying fuzzy rules ;
                     Memorise the current step for an eventual backward ;
                     if the resulting amplitude is not valid
                     begin
                            if a new attempt is still allowed
                                   Init a new backward attempt ;
                            else
                                   Send a Withdrawal command ;
                     end
              end
              else that is going backward
              begin
                     if initial position is reached
                     begin
                     Update strategy ;
                     Define a new forward attempt according to strategy ;
                     end
                     else
                     begin
                            Continue going backward using memorised steps ;
                     end
              end
```

Algoritmo 3. *Pseudocódigo del método AmNfp internalPropagation(); este algoritmo es una versión del algoritmo genérico 1 diseñado para cumplir con la funcionalidad de suavizado.*

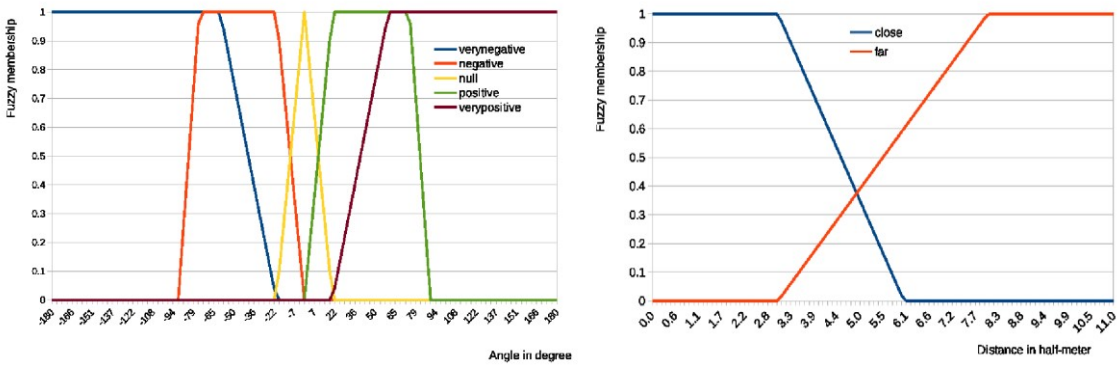

Figura 4. *Izquierda: funciones de pertenencia correspondientes a las variables de entrada GoalAngle y SteerAngle. Derecha: funciones de pertenencia correspondientes a las variables de entrada de distancia ultrasónicas LeftSensor y RightSensor*

```
procedure AmNfpInternalPropag( GoalAngle, SteerAngle, LeftSensor, RightSensor)
  var SteeringWheel which corresponds to the Command-Amplitude sent to the robot
        begin
        R1:    IF GoalAngle IS veryNegative AND SteerAngle IS veryNegative
        THEN SteeringWheel GET goStraight
        R2:    IF GoalAngle IS veryNegative AND SteerAngle IS negative
        THEN SteeringWheel GET turnRight
        R3:    GoalAngle IS veryNegative AND SteerAngle IS zero
        THEN SteeringWheel GET turnRight
        R4:    IF GoalAngle IS veryNegative AND SteerAngle IS positive
        THEN SteeringWheel GET turnFullRight
        R5:    IF GoalAngle IS veryNegative AND SteerAngle IS veryPositive
        THEN SteeringWheel GET turnFullRight

        R6:    IF GoalAngle IS negative AND SteerAngle IS veryNegative
        THEN SteeringWheel GET goStraight
        R7:    IF GoalAngle IS negative AND SteerAngle IS negative
               THEN SteeringWheel GET goStraight
        R8:    IF GoalAngle IS negative AND SteerAngle IS zero
        THEN SteeringWheel GET turnRight
        R9:    IF GoalAngle IS negative AND SteerAngle IS positive
        THEN SteeringWheel GET turnFullRight
        R10:   IF GoalAngle IS negative AND SteerAngle IS veryPositive
        THEN SteeringWheel GET turnFullRight
        R11:   IF GoalAngle IS zero AND SteerAngle IS veryNegative
        THEN SteeringWheel GET turnFullLeft
        R12:   IF GoalAngle IS zero AND SteerAngle IS negative
        THEN SteeringWheel GET turnLeft
        R13:   IF GoalAngle IS zero AND SteerAngle IS zero
        THEN SteeringWheel GET goStraight
        R14:   IF GoalAngle IS zero AND SteerAngle IS positive
        THEN SteeringWheel GET turnRight
        R15:   IF GoalAngle IS zero AND SteerAngle IS veryPositive
        THEN SteeringWheel GET turnFullRight
        R16:   IF GoalAngle IS positive AND SteerAngle IS veryNegative
        THEN SteeringWheel GET turnFullLeft
        R17:   IF GoalAngle IS positive AND SteerAngle IS negative
        THEN SteeringWheel GET turnLeft
        R18:   IF GoalAngle IS positive AND SteerAngle IS zero
        THEN SteeringWheel GET turnLeft
        R19:   IF GoalAngle IS positive AND SteerAngle IS positive
        THEN SteeringWheel GET goStraight
        R20:   IF GoalAngle IS positive AND SteerAngle IS veryPositive
        THEN SteeringWheel GET goStraight
        R21:   IF GoalAngle IS veryPositive AND SteerAngle IS veryNegative
        THEN SteeringWheel GET turnFullLeft
        R22:   IF GoalAngle IS veryPositive AND SteerAngle IS negative
        THEN SteeringWheel GET turnFullLeft
        R23:   IF GoalAngle IS veryPositive AND SteerAngle IS zero
        THEN SteeringWheel GET turnLeft
        R24:   IF GoalAngle IS veryPositive AND SteerAngle IS positive
        THEN SteeringWheel GET turnLeft
        R25:   IF GoalAngle IS veryPositive AND SteerAngle IS veryPositive
        THEN SteeringWheel GET goStraight
        R26:   IF LeftSensor IS close AND RightSensor IS far
        THEN SteeringWheel GET turnFullRight
        R27:   IF LeftSensor IS far AND RightSensor IS close
        THEN SteeringWheel GET turnFullLeft
        R28a:IF LeftSensor IS close AND RightSensor IS close
        THEN SteeringWheel GET turnFullLeft
        R28b:IF LeftSensor IS close AND RightSensor IS close
        THEN SteeringWheel GET turnFullRight
        end
```

Algoritmo 4. *Conjunto de reglas difusas implementadas en el método internalPropagation() del módulo AmNfp.*

4. Adaptación del Método AmNfp

La respuesta de AmNfp debe adaptarse con el tiempo porque se desconocen las características mecánicas del robot, como sus dimensiones, radio mínimo de giro y velocidad de dirección. Como se explicó anteriormente, el proceso interno del módulo

AmNfp se ha programado como un sistema neuro difuso para lograr la adaptación de los parámetros *turnFullLeft, turnLeft, ..., turnFullRight* para producir una respuesta adecuada. Este proceso de aprendizaje interno se lleva a cabo mientras se realizan varios intentos para evitar un obstáculo, como se muestra en la fig. 5. En la práctica, el robot es impulsado hacia adelante y hacia atrás mientras su radio de giro se incrementa gradualmente hasta que se logra un intento exitoso o se activa un comando de Retirada. En particular, para realizar la adaptación, el módulo AmNfp memoriza las últimas secuencias de pasos para retroceder al recibir la solicitud correspondiente del módulo AmDmr. Este proceso secuencial, junto con el sistema difuso adaptativo, permite que el robot muestre las habilidades motoras finas necesarias para maniobrar con precisión en un entorno real.

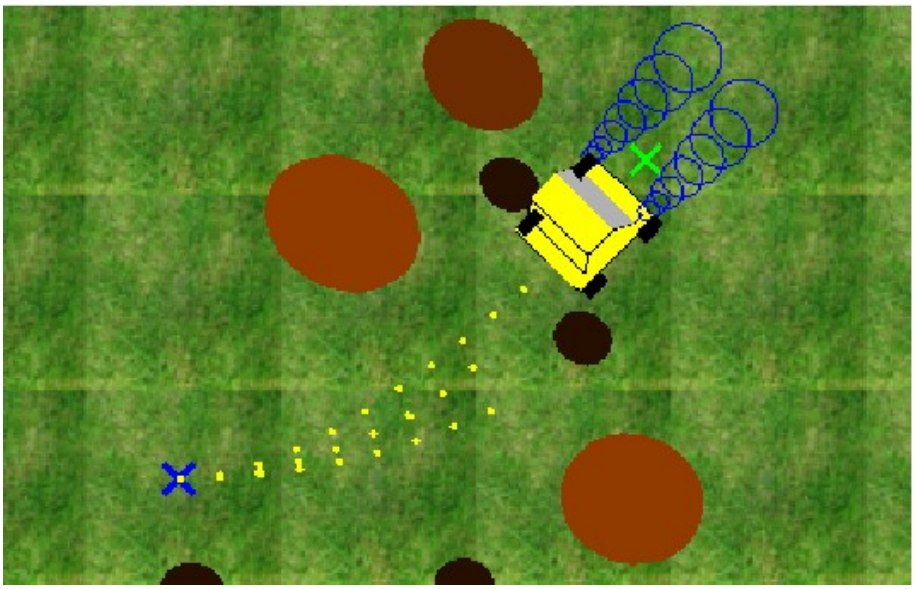

Figura 5. *Esta vista de la simulación muestra los tres intentos (líneas de puntos amarillas) generados por el proceso de adaptación del módulo AmNfp para permitir que el robot supere un obstáculo. Con cada intento fallido, el robot retrocede y aumenta su radio de giro más alto turnLeft y turnFullLeft*

RESULTADOS Y DISCUSIÓN

La implementación de los módulos AmDmr y AmNfp solo requirió 1400 y 950 líneas respectivamente de código C++ porque todas las funcionalidades genéricas ya están programadas dentro de la clase *ActingModule*. Sin embargo, las pruebas de funcionalidad requieren el desarrollo de una simulación para evaluar por separado cada nuevo módulo. Además, para cada prueba se diseña un controlador que incluye un

único módulo para realizar la prueba (ver fig. 6). Esta última fase no requiere programación adicional porque, como se mencionó en la introducción, los controladores se diseñan y configuran gráficamente utilizando el editor GFM. La configuración consiste básicamente en indicar el archivo ejecutable de la simulación correspondiente o la dirección IP del robot. Sin embargo, las simulaciones son mucho más complejas de desarrollar que los propios módulos, principalmente debido a la implementación de OpenGL necesaria para visualizar el comportamiento del robot. Las simulaciones AmDmr y AmNfp correspondientes requieren 1500 y 4700 líneas respectivamente de código C++. Los archivos fuente de ambos módulos y sus simulaciones se pueden descargar desde (Pasquier J. L., 2022). Las siguientes subsecciones presentan la evaluación del nuevo módulo de actuación descrito anteriormente.

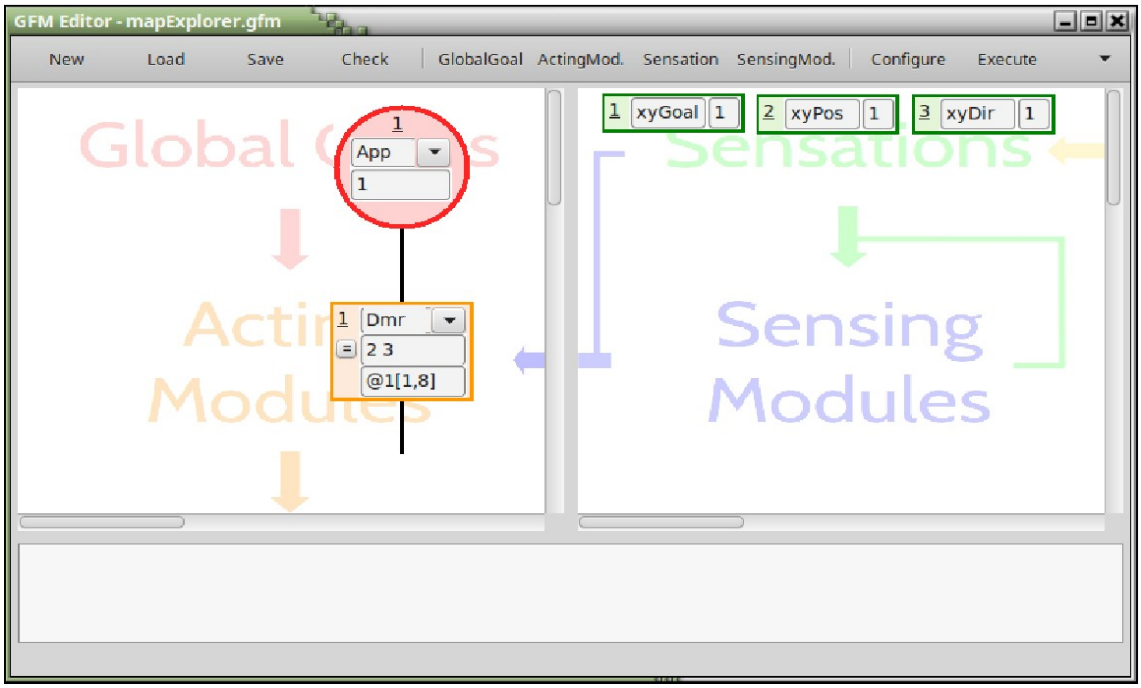

Figura 6. *Para evaluar la funcionalidad de AmDmr, un controlador incorpora este único módulo. Esta representación gráfica corresponde a la fase de programación: se definen las conexiones de entrada/salida, la funcionalidad del módulo y sus parámetros. La contraparte de la simulación se especifica haciendo clic en el botón "Configurar" en el menú; al hacer clic en el comando "ejecutar" se ejecuta simultáneamente el controlador y la simulación*

5. Habilidad motora básica del subsistema de movimiento.

El desarrollo del subsistema de movimiento está asociado a la implementación del módulo *AmDmr* previamente. Para evaluar la funcionalidad del módulo *AmDmr*, el controlador que se muestra en la fig. 6 se ha conectado a través de TCP/IP a la simulación robótica de un entorno de obstáculos estáticos y dinámicos. El experimento que se muestra en la fig. 7 representa el mapeo de un entorno previamente desconocido que muestra la posición inicial del robot (punto verde), su trayectoria (secuencia de flechas verdes), la meta (punto amarillo), los obstáculos estáticos (cuadros blancos) y los dinámicos (cuadros grises). De acuerdo con los requisitos explicados, el módulo debe realizar el mapeo y la planificación de rutas sin un conocimiento inicial del entorno. En particular, estas funcionalidades son independientes del hardware del robot para aumentar su portabilidad. Para probar las funcionalidades de *AmDmr*, el controlador que se muestra en la fig. 7 incluye un solo módulo; su conectividad y configuración especifican que:

- El comando de salida es único, pero incluye ocho valores de amplitud.
- La retroalimentación está compuesta por la posición del robot y la dirección del objetivo.
- La solicitud de entrada de *AmDmr* (opción "App") que se transmite por la sensación 1; esto permite que la simulación actúe como entrenador decidiendo cuándo establecer una nueva meta

Esto está conectado a través de TCP/IP a la simulación que muestra el comportamiento del robot y muestra la capacidad de aprendizaje del módulo para realizar mapeo y enrutamiento. Para que se considere que ha adquirido las habilidades requeridas, el módulo *AmDmr* debe ser capaz de:

1. Mejorar permanentemente el mapeo interactuando con el entorno.
2. Encontrar un camino adecuado a la meta; el controlador debe encontrar una ruta alternativa incluso en un entorno desconocido.
3. Manejar obstáculos dinámicos que cambian de posición al azar, lo que significa encontrar un camino incluso cuando el entorno está alterado.
4. Realizar una media vuelta, que requiere más espacio libre.
5. Moverse hacia atrás cuando el robot es conducido a un callejón sin salida.

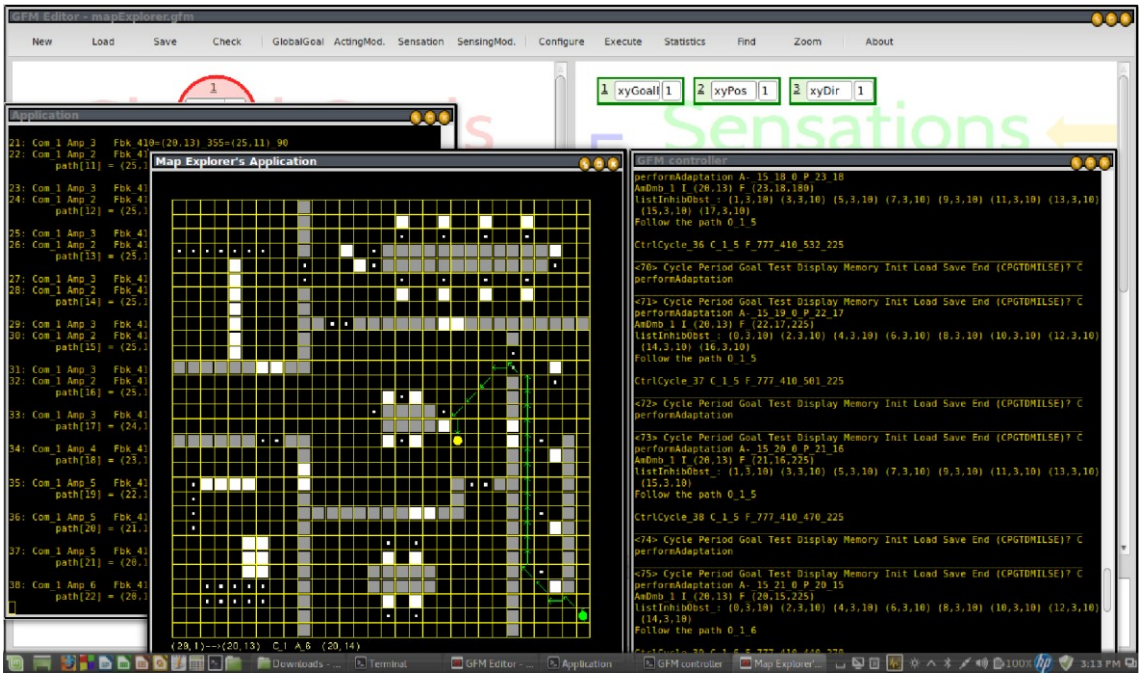

Figura 7. *Evaluación del módulo AmDmr: La ventana de fondo muestra el editor. El controlador se ejecuta en la ventana de terminal inferior derecha y la aplicación, en la parte inferior izquierda y se muestra gráficamente en primer plano. Ambas aplicaciones se comunican a través de TCP/IP.*

Diseñar y ejecutar con éxito el controlador junto con la simulación implica que el módulo *AmDmr* cumple completamente con el diagrama de componentes del módulo activo y el algoritmo genérico. El experimento muestra que se cumplen todas las especificaciones anteriores, el mapeo dinámico se adquiere rápidamente y el enrutamiento se realiza de manera efectiva por el módulo, lo que muestra no solo la viabilidad de la aplicación de movimiento, sino también los cimientos sólidos del enfoque GFM: una nueva funcionalidad se ha integrado fácilmente en un módulo de actuación genérico y se ha adquirido con éxito de manera dinámica.

Sin embargo, hay un inconveniente que debe mencionarse al tratar con posiciones dinámicas, la confianza de estas posiciones solo puede actualizarse cuando el robot está en contacto directo con un obstáculo. Esto significa que el robot actúa como una entidad ciega que intenta cruzar repetidamente una posición a pesar de que está equipado con sensores ultrasónicos que podrían evitar tantos y agotadores intentos. La pregunta es ¿Cómo mejorar el comportamiento del robot utilizando sensores de largo alcance para mejorar la actualización de mapas? Desafortunadamente, no hay una

respuesta fácil a esta simple pregunta debido a las estrictas restricciones de la arquitectura GFM, eventualmente el proceso de mapeo debe ser realizado en el Área de Detección por un conjunto de módulos provistos de diferentes tipos de sensaciones y percepciones, esto podría generar un mapa mucho más elaborado, diseñado no solo para realizar rutas sino también, por ejemplo, para memorizar recursos o posiciones estratégicas. En conclusión, el subsistema de locomoción aquí propuesto es adecuado principalmente para una entidad primitiva.

Una crítica más conceptual de *AmDmr* se refiere a su funcionalidad, que parece ser más especializada que la de los módulos de actuación existentes como AmCi. El punto sería determinar si dicho mapeo/enrutamiento podría aplicarse para realizar otro tipo de tareas de nivel superior o si está estrictamente limitado a la locomoción básica. El código fuente del módulo *AmDmr* se puede descargar desde (Pasquier J. L., 2022) y un video ilustrativo desde (Pasquier J. L., A basic gfm control illustration, building a pathfinder GFM controller, 2022).

6. Habilidad motora fina del subsistema de movimiento

Como se mencionó, el módulo *AmNfp* ha sido desarrollado para suavizar la ruta calculada por el módulo *AmDmr*. Para evaluar su funcionalidad, el controlador que se muestra en la fig. 8 ha sido diseñado para estipular que:

- Los comandos al volante del robot son disparados por el módulo AmNfp, su amplitud varía de -10 a 10.
- Su retroalimentación está compuesta primero por la posición de meta; segundo, por los valores de los sensores ultrasónicos que indican la distancia a un obstáculo potencial y tercero, por el ángulo entre la dirección del robot y la posición objetivo, correspondiente a los sensores #1,2,9 y 10 en la fig. 8.
- En teoría, las solicitudes de entrada de *AmNfp* deberían ser generadas por el módulo *AmDmr*, aunque actualmente estas solicitudes las proporciona la aplicación (el objetivo global de la aplicación) para evaluar la aptitud del módulo AmNfp independientemente del *AmDmr*.

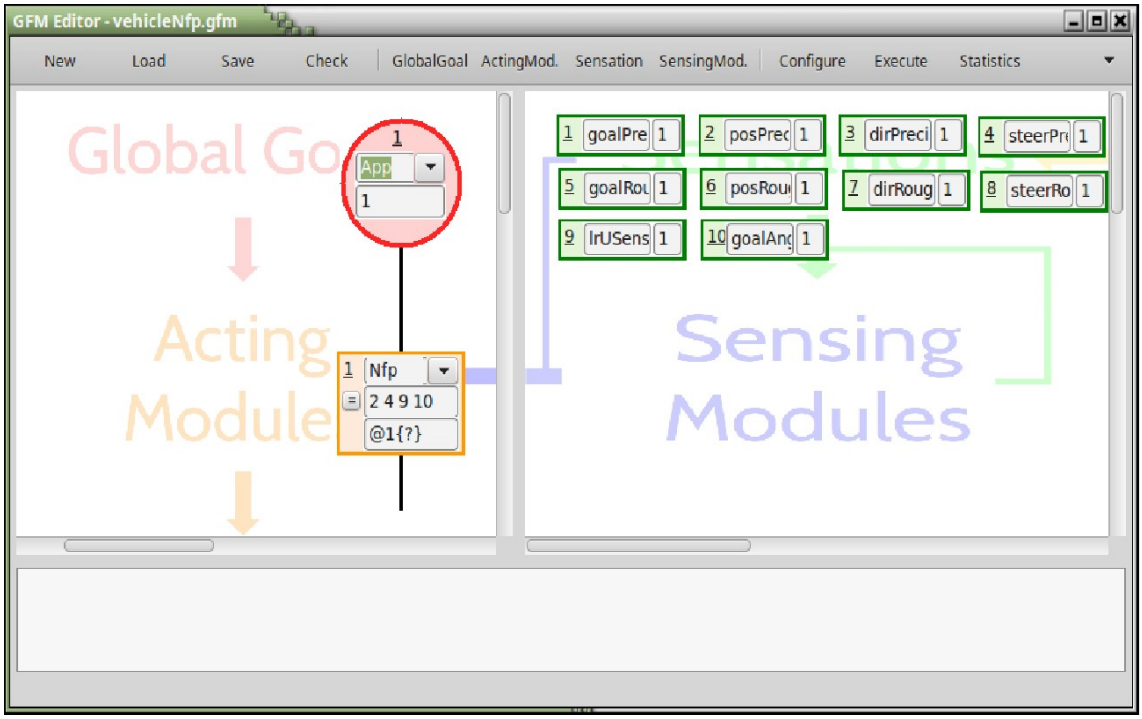

Figura 8. *El controlador solo integra un módulo AmNfp para probar su funcionalidad. El controlador está completamente definido por este diseño y las especificaciones del botón "Configurar". Al hacer clic en el botón "Ejecutar" se ejecuta el controlador y la simulación al mismo tiempo*

Este segundo controlador está conectado a través de TCP/IP interno a una simulación más realista del entorno donde los obstáculos exhiben diferentes tamaños, formas y disposiciones no geométricas (ver fig. 3). El robot puede detectar este entorno a través de dos sensores ultrasónicos cuyos rayos están representados por una secuencia de círculos azules que se ensanchan.

La fig. 5 muestra tres intentos realizados por el robot (líneas de puntos amarillas) para sortear el obstáculo. Con cada falla, el robot retrocede y luego realiza un nuevo intento, aumentando su radio de giro. Al hacerlo, el módulo *AmNfp* adapta sus respuestas a las características mecánicas del sistema de dirección del robot. Se han llevado a cabo con éxito diferentes experimentos relacionados con la simulación del vehículo. El video (Jerome, 2022) ilustra la capacidad específica del módulo *AmNfp* para complementar las del módulo *AmDmr*: Encontrar el mejor camino para llegar a la posición de destino durante cada paso-comando, Realizar varios intentos antes de activar un comando de Retirada, Retroceder cuando lo solicite el módulo *AmDmr* o cuando el camino esté

obstruido y Adaptación de la respuesta de control a las características del sistema de dirección.

Un aspecto particularmente importante del presente proyecto de investigación es haber separado con éxito los procesos de mapeo-enrutamiento y suavizado al integrar estas funcionalidades en dos módulos de actuación independientes. En la práctica, esta tarea fue el principal desafío en el desarrollo de este subsistema de locomoción. La relevancia es que un módulo de actuación debe estar diseñado para realizar una funcionalidad tan sencilla como sea posible; esto, con el fin de aumentar la flexibilidad y versatilidad del módulo. Podríamos aclarar esta afirmación por analogía con un juego de construcción: un mosaico más simple ofrece más opciones de conectividad. Sin embargo, debemos mencionar que no se ha realizado ningún experimento para conectar el comando de salida del módulo *AmDmr* como solicitud de entrada al módulo *AmNfp*.

Finalmente, el presente desarrollo eleva el número de módulos operativos de actuación de cinco a siete. La integración de estos nuevos componentes ayuda a reforzar las habilidades de aprendizaje de GFM y su capacidad de representación contribuyendo así a posicionar la GFM como un paradigma potencial. Reiteramos que el experimento tiene como objetivo mostrar la viabilidad de diseñar controladores basados en el aprendizaje utilizando el enfoque GFM, no mejorar las técnicas existentes de suavizado de rutas; por lo tanto, no se ha realizado ninguna comparación con otras propuestas,

REFERENCIAS

Almansa-Valverde, S. e. (2012). Mobile robot map building from time-of-flight camera. *Expert Systems with Applications*, 8835-8843.

Asada, M. e. (2001). Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Robotics and Autonomous systems*, 185-193.

Asada, M. e. (2009). Cognitive developmental robotics: A survey. . *IEEE transactions on autonomous mental development*, 12-34.

Bach, J. (2012). MicroPsi 2: the next generation of the MicroPsi framework. . *Springer, Berlin, Heidelberg.*, 11-20.

Brooks, R. A. (1998). Alternative essences of intelligence. *American Association for Artificial Intelligence*, 961-968.

Duchoň, F. .. (2014). Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, 59-69.

Einhorn, E. &. (2015). Generic NDT mapping in dynamic environments and its application for lifelong SLAM. . *Robotics and Autonomous Systems*, 28-39.

Franklin, S. e. (2013). LIDA: A systems-level architecture for cognition, emotion, and learning. *IEEE Transactions on Autonomous Mental Development*, 19-41.

Goertzel, B. (2009). OpenCogPrime: A cognitive synergy based architecture for artificial general intelligence. *IEEE International Conference on Cognitive Informatics* , 60-68.

Leboeuf Pasquier, J. (2005). Growing functional modules, a prospective paradigm for epigenetic artificial intelligence. . *Springer, Berlin, Heidelberg.*, 465-471.

Leboeuf-Pasquier, e. a. (2013). A preliminary auditory subsystem based on a gfm controller. *Proceedings of IWINAC*, 1-10.

Leboeuf-Pasquier, J. (2015). A basic Growing Functional Modules “artificial brain”. . *Neurocomputing*, 55-61.

Leboeuf-Pasquier, J. (2015). A Growing Functional Modules learning based controller designed to balance of a humanoid robot on one foot. *International Work-Conference on the Interplay Between Natural and Artificial Computation*, 240-250.

Lungarella, M. e. (2003). Developmental robotics: a survey. . *Connection science*, 151-190.

Meeden, L. A. (2006). Introduction to developmental robotics. *Connection Science*, 93-96.

Pasquier, J. L. (2006). Improving the RTR Growing Functional Module. *IECON Annual Conference on IEEE Industrial Electronics*, 3945-3950.

Pasquier, J. L. (2007). A growing functional module designed to trigger causal inference. *ICINCO-ICSO*, 456-466.

Pasquier, J. L. (01 de 07 de 2022). *A basic gfm control illustration, building a pathfinder GFM controller.* Obtenido de https://drive.google.com/open?id=1xl9eoyqv1Ci1z2bZ8F6GGeCp8LrxNmpA

Pasquier, J. L. (01 de 07 de 2022). *The source code of the GFM modules and the corresponding test applications.* Obtenido de

https://drive.google.com/file/d/1QDbOWLlvPLYBpb39isDk6kCsyEKcoshz/view?usp=sharing

Tian, B. e. (2013). RGB-D based cognitive map building and navigation. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1562-1567.