Um Agente Autônomo Concorrente para o Manipulador Robótico Jaco Kinova

Edi Moreira M. de Araujo[1], **Augusto Loureiro da Costa**[2]

[1]Programa de Pós-graduação em Engenharia Elétrica
Universidade Federal da Bahia (UFBA)
Salvador – BA – Brazil

edimoreira13@yahoo.com.br, augusto.loureiro@ufba.br

Abstract. *This work presents the use of the Concurrent Autonomous Agent (CAA) in an JACO Kinova robot arm, enabling it to perform complex tasks in a completely autonomous way. The communication between the CAA and the manipulator will be made through the ROS (Robot Operating System), as well as the performance of the behaviors present in the reactive level. The Concurrent Autonomous Agent is an implementation of a cognitive agent architecture based on the Generic Cognitive Model for Autonomous Agents. This model over the years proved to be very effective, initially being used for the implementation of a distributed control system for multi-robot systems, called Mecateam, obtaining significant results in RoboCup's Latin America and Brazilians. The CAA has already been implemented in several successful applications, such as the NAO humanoid robot and the AxeBot omnidirectional robot. These results point to CAA as a model of well-known cognition for the training of robots to perform tasks that require a certain degree of cognition.*

Resumo. *Este trabalho tem como objetivo apresentar a implementação do Agente Autônomo Concorrente (AAC) para o manipulador robótico JACO Kinova, habilitando-o à execução de tarefas complexas de uma maneira completamente autônoma. A comunicação entre o AAC e o manipulador foi feita através do ROS (Robot Operating System), bem como a realização dos comportamentos presentes no nível reativo. O Agente Autônomo Concorrente é a implementação de uma arquitetura de agente cognitivo baseada no modelo cognitivo genérico para agentes autônomos. Este modelo ao longo dos anos mostrou-se bastante eficaz, sendo inicialmente utilizado para a implementação de um sistema de controle distribuído para sistemas multi-robôs, chamado Mecateam, obtendo resultados significantes em RoboCup's Latin América e Brasileiras. O AAC já foi implementado em diversas aplicações com êxito, como exemplo, o robô humanoide NAO e o robô omnidirecional AxéBot. Tais resultados apontam o AAC como um modelo de cognição bem indicado para a capacitação de robôs a execução de tarefas que solicitam um alto grau de cognição.*

1. Introdução

Manipuladores robóticos podem ser utilizados em diversas aplicações, porém muitos destes manipuladores estão limitados a movimentos sequenciais predefinidos ou necessitam de controle humano, isto é, não possuem a capacidade de identificar mudanças no ambiente ou de tomar decisões de uma maneira autônoma. O grau de autonomia presente nos

robôs está associado com a tarefa que o mesmo venha a executar. Em algumas linhas de montagem industrial, os comportamentos dos robôs podem ser supervisionados por uma máquina de estados discretos, que determina exatamente quais ações estão disponíveis em um dado estado, porém, em tarefas mais complexas, especialmente aquelas que exigem planejamento, o mecanismo de tomada de decisão deve ser mais robusto, de maneira que o robô venha adquirir a capacidade de realizar estas tarefas em um tempo hábil, com um alto grau de precisão e de uma forma completamente autônoma. De forma a lograr desta completa autonomia, estas máquinas devem ser dotadas de algum tipo de inteligência, ou seja, devem ser capazes de tomar decisões por conta própria. Para atender a estas necessidades, este trabalho propõe a implementação de um sistema inteligente, baseado em agentes cognitivos, no manipulador robótico JACO Kinova, capacitando-o desta forma a realização de tarefas complexas. A tarefa escolhida para realização de experimentos, foi a de habilitar o manipulador a montar uma Torre de Hanói com três discos (Figura 8).

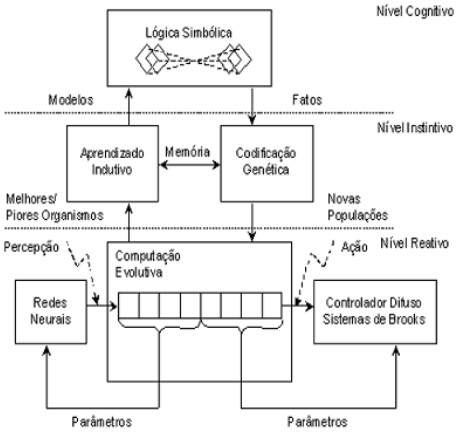

Figura 1. Modelo genérico para agentes cognitivos.

Em [Bittencourt 1997] um modelo genérico de agentes cognitivos é proposto (Figura 1). Este modelo consiste em uma arquitetura cognitiva utilizada para modelar agentes de qualquer natureza. O modelo genérico de agentes cognitivos é utilizado como base em [Costa and Bittencourt 1999] e [Bittencourt and Costa 2001] para a proposta de uma arquitetura de agentes autônomos inteligentes chamada de Agente Autônomo Concorrente (AAC). Este agente utiliza uma arquitetura de três camadas: cognitiva, instintiva e reativa (Figura 2). Cada uma delas é implementada como um processo e representa um nível decisório distinto que complementa as demais para a construção de um agente cognitivo. A complexidade do comportamento do agente é incrementada a cada camada. Resultados experimentais, utilizando o AAC, foram realizados com êxito como podemos verificar em [Costa et al. 2011], onde o mesmo teve o nível reativo embarcado no robô omnidirecional AxéBot. Neste caso o nível reativo teve como função rastrear as trajetórias geradas pelo nível instintivo do agente. Já em [Costa et al. 2015] o nível reativo foi implementado no modulo *Unifield Humanoid Robotics Software Platform* (UHRSP) do robô Humanoide NAO. O UHRSP disponibiliza uma plataforma de controle de caminhada chamado de *Zero Moment Point* (ZMP), que recebe do nível instintivo do AAC qual o comportamento será executado pelo controlador ZMP. Em [Ferreira 2014] uma rede de três microcontroladores foi projetada para embarcar o AAC no robô omnidirecional AxéBot. A rede foi concebida mimetizando a estrutura funcional do AAC. Os três níveis deste

último (a saber, o nível reativo, o nível instintivo e o nível cognitivo) foram embarcados em cada nó da rede.

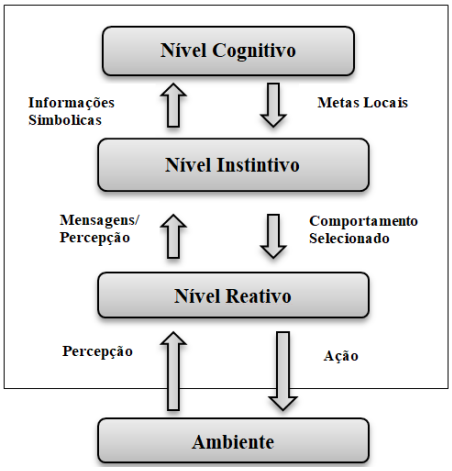

Figura 2. Fluxo de informação do Agente Autônomo Concorrente.

Com tais resultados, podemos chegar a conclusão de que o AAC pode ser implementado em qualquer sistema robótico, modificando-se apenas sua camada reativa, levando em conta a característica de hardware de cada robô. Faz-se também necessário a criação de bases de regras, para os níveis cognitivo e instintivo.

Neste presente trabalho, o nível reativo foi implementado utilizando o ROS. O ROS possui um conjunto de ferramentas e bibliotecas que permite ter o controle do manipulador JACO (Figura 14) Kinova tanto no espaço cartesiano, quanto no espaço de juntas.

Os principais objetivos e contribuições deste trabalho são:

- Implementação da camada reativa do AAC utilizando o ROS, algo nunca utilizado anteriormente e de extrema importância no auxilio a futuras implementações do AAC;
- Capacitação do manipulador robótico JACO Kinova a execução de tarefas complexas;
- Implementação do AAC em uma nova categoria de robôs (manipuladores robóticos).

O presente artigo está estruturado da seguinte forma: na seção 2 são apresentados os principais conceitos acerca do JACO Kinova e ROS. Em seguida, na seção 3 é apresentado o AAC; os experimentos e os resultados são apresentados na seção 4 e as conclusões e perspectivas futuras delineadas na seção 5.

2. JACO Kinova e ROS

2.1. Manipulador JACO Kinova

O robô JACO é um manipulador criado pela empresa canadense *Kinova Robotics*. Ele é dotado de seis juntas rotacionais, 6 elos (*links*) e um efetuador com 3 dedos (*fingers*). As seis juntas são movidas individualmente por seis motores *brushless* DC, onde dois tipos de módulos motor são usados. Nas juntas 1-3, onde a junta 1 é a mais próxima da base,

usa motores com torque de até 1 *Nm*, enquanto as articulações 4-6 usam pequenos motores com torque de no máximo 0.5 *Nm*. As seis juntas possuem *ranges* distintos, sendo que as juntas 1, 4, 5 e 6 possuem um *range* de [-10000°, 10000°] e as juntas 2 e 3 possuem o *range* de [42°, 318°] e [17°, 343°] respectivamente. Os elos são constituídos de fibra de carbono que abrigam os módulos motor. Cada dedo do efetuador possui um motor individual, totalizando assim 9 motores. A base do JACO abriga um processador de sinal digital (DSP) que envia informações para cada motor com base na entrada do usuário. Dispositivos que possuem o poder de comando são conectados a base do manipulador através de uma porta USB, configuração essa reconhecida pelo sistema de controle.

2.2. ROS

O *Robotic Operationg System* (ROS) é um software de código aberto, desenvolvido em 2007, que disponibiliza bibliotecas e ferramentas para ajudar desenvolvedores de programas robóticos a criarem suas aplicações, permitindo que diferentes grupos de pesquisa possam trocar informações e experiências, reutilizando códigos e ferramentas criadas com o seu auxílio [Barros 2014]. Ele fornece diversos serviços, incluindo abstração de hardware, implementação de bibliotecas, controle de baixo nível de dispositivos (*drivers*), gerenciamento de pacotes e troca de mensagens entre processos [Oliveira et al. 2013]. A empresa Kinova disponibiliza um conjunto de códigos e bibliotecas que proporcionam a comunicação e controle do JACO através do ROS [Kinova 2017]. Com isso, torna-se possível ter o controle tanto no espaço de juntas como o controle no espaço cartesiano do manipulador.

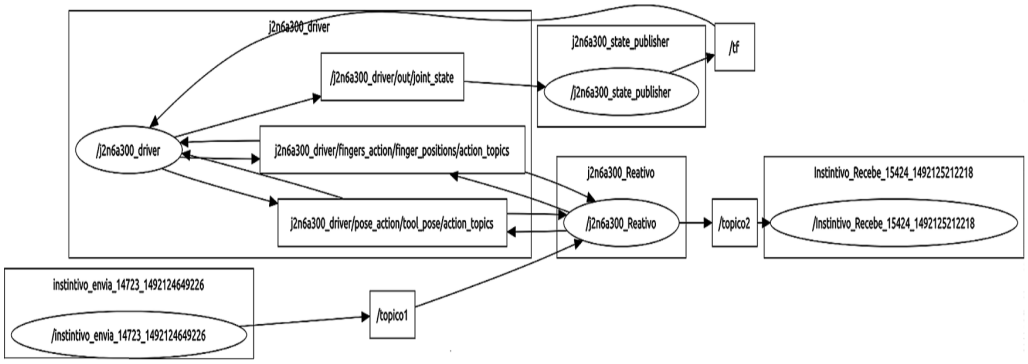

Figura 3. Fluxograma grafo do ROS (nível reativo).

Na Figura 3 podemos verificar o grafo de execução do ROS. Cada bloco representa um ”nó” em execução. ”Nós” são processos que executam instruções, geralmente utilizando bibliotecas do ROS. Cada nó comunica-se uns com os outros através de *streaming* de tópicos, serviços via RPC (*Remote Procedure Call*) e o servidor de parâmetros.

3. Agente Autônomo Concorrente

A arquitetura do AAC foi inspirada no modelo genérico para agentes cognitivos proposto em [Bittencourt 1997]. A ideia principal desta arquitetura era implementar percepção, ação, comunicação, cooperação, planejamento e tomada de decisão, utilizando a programação concorrente. Neste modelo concorrente, a arquitetura de tomada de decisão

era completamente centralizada, ocasionando alguns problemas de sincronismo entre o agente e o ambiente [Costa and Bittencourt 1998].

Com o intuito de solucionar esses problemas de sincronização, a arquitetura do agente cognitivo concorrente migrou de uma abordagem centralizada, para uma arquitetura descentralizada, através da sua implementação no *framework* Expert-Coop++ [Costa and Bittencourt 1999]. Esse modelo baseia-se na hipótese que as atividades cognitivas possuem três características principais: auto-organização, natureza evolutiva e dependência [Costa et al. 2011]. De acordo com esse modelo, um agente cognitivo é composto por três níveis decisórios: nível reativo, instintivo e cognitivo (Figura 2). Cada um desses níveis implementa um processo decisório distinto, complementando os demais níveis para a construção de um agente cognitivo.

Um componente importante presente em todos os níveis decisórios do AAC, o *mailbox*, tem um papel fundamental no funcionamento do agente. O *mailbox* consiste de um objeto instanciado em cada processo do Expert-Coop++ que oferece uma interface de comunicação baseada em soquetes UNIX e uma estrutura de dados que funciona como um *buffer*, armazenando as mensagens em uma fila de tamanho finito. Quando uma mensagem é lida, a mesma é removida da fila [Ferreira 2014].

3.1. Nível Cognitivo

A função do nível Cognitivo (Figura 4) é de possuir um modelo simbólico do ambiente, um conjunto de planos, uma base de conhecimento codificado em regras de produção com a qual é possível estabelecer metas e escolher o plano mais adequado para alcança-las. Estes planos são executados pelo nível instintivo. O nível cognitivo recebe informações simbólicas do nível instintivo e as mensagens dos outros agentes (quando inserido em um sistema multiagente). Um importante aspecto do AAC é que enquanto os níveis instintivo e reativo trabalham no alcance de uma meta local, o nível cognitivo pode, concomitantemente, se dedicar a tarefas de planejamento e criação de metas [Ferreira 2014].

Nos experimentos realizados na seção 4, o nível cognitivo possui uma representação do ambiente *E(k)*, uma base de regras de produção *Bi* contento um total de 25 regras (Figura 5), a qual contém o conhecimento necessário para associar ao estado corrente do ambiente *E(k)* e escolher o plano a ser executado pelo nível instintivo. Um motor de inferência, presente no nível cognitivo, de um único ciclo, utiliza o conhecimento armazenado em *Bi*, para escolher o plano a ser executado pelo nível instintivo.

3.2. Nível Instintivo

O nível instintivo (Figura 4) é o responsável pelo reconhecimento das mudanças do estado do ambiente, atualização das informações simbólicas utilizadas na base de conhecimento do nível cognitivo e pela escolha do comportamento a ser utilizado pelo nível reativo. O reconhecimento das mudanças do estado do ambiente é realizado após cada ciclo de percepção-ação executado pelo nível reativo. Este nível executa planos, que se bem sucedidos, levam a satisfação de metas locais, criadas pelo nível cognitivo.

O arquivo de regras, implementado pelo usuário, fornece as regras necessárias para classificar os estados do ambiente, escolher o comportamento mais adequado a este estado e gerar informações a ser enviadas ao nível cognitivo. Lógica e quadros são os formalismos admitidos para representação de conhecimento do agente. O nível instintivo

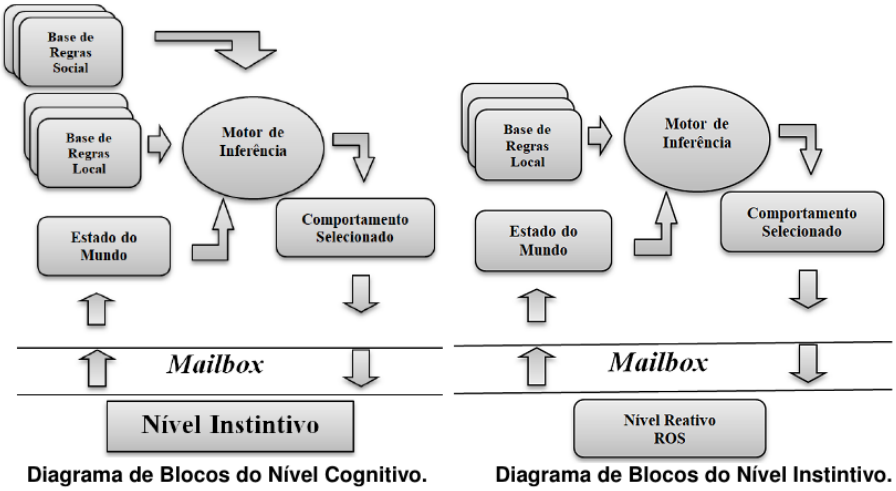

Figura 4. Diagrama de Blocos dos Níveis Cognitivo e Instintivo.

(Figura 4) utilizado neste trabalho tem como principal função executar o plano escolhido pelo nível cognitivo, reconhecer as mudanças do estado do ambiente e escolher o comportamento que será executado pelo nível reativo implementado no ROS. O reconhecimento das mudanças do estado do ambiente é realizado após cada ciclo de percepção-ação executado pelo nível reativo. Foram criados três arquivos de regras, sendo cada um destes arquivos um plano a ser escolhido pelo nível cognitivo.

Um Sistema Baseado em Conhecimento (SBC), composto por um Motor de Inferência (MI), Base de Regras (BR) e uma Base de Fatos (Estado do Mundo) são implementados neste nível.

3.3. Representação de conhecimento do AAC

Segundo Ferreira (2014), o AAC pode utilizar a Lógica de Primeira (LPO) Ordem ou quadros como método de representação de conhecimento, sendo o LPO o método escolhido de representação neste trabalho. Nas Figuras 5 e 6, temos o exemplo de algumas regras implementadas nos níveis cognitivo e instintivo utilizando o conceito de objeto, atributo e valor. As representações do estado do ambiente são identificados pelo *token* "if", ou seja, condições necessárias para a execução dos termo identificado pelo *token* "then". Todas as regras possuem uma estrutura semelhante as regras apresentadas nas Figuras 5 e 6, possuindo três condições necessárias e uma consequência para estas condições.

Como podemos observar na Figura 5, temos exemplos de três regras implementadas no nível cognitivo, sendo todas elas responsáveis por escolher o plano a ser executado no nível instintivo. Os termos delimitados pelo *token* "then", isto é, plano escolhido para ser executado pelo nível instintivo tem como estrutura: o termo "robo", o termo "acao" e por último o plano escolhido dentre as três opções possíveis (plano_1, plano_2 ou plano_3).

A Figura 5 mostra o exemplo de uma regra implementada no nível instintivo. As condições delimitadas pelo *token* "if" são: "disco_menor (amarelo)", "disco_medio (verde)" e "disco_maior (vermelho)" como os "objetos", o termo "posicao" como o estado e os termos "p1n3", "p1n2" e "p1n1" como os atributos. Estes atributos determinam a localização dos discos na Torre de Hanói, conforme a Tabela 1. Os termos delimitados

```
(regra_21
     (if   (logic ( disco_menor posicao p3n2 ))
           (logic ( disco_medio posicao p3n1 ))
           (logic ( disco_maior posicao p2n1 )))
     (then (logic ( robot acao plano_3 ))))
(regra_7
     (if   (logic ( disco_menor posicao p1n1 ))
           (logic ( disco_medio posicao p3n2 ))
           (logic ( disco_maior posicao p3n1 )))
     (then (logic ( robot acao plano_1 ))))
(regra_8
     (if   (logic ( disco_menor posicao p2n3 ))
           (logic ( disco_medio posicao p2n2 ))
           (logic ( disco_maior posicao p2n1 )))
     (then (logic ( robot acao plano_2 ))))
```

Figura 5. Exemplo de três regras implementadas no nível cognitivo.

```
(regra_1
      (if     (logic ( disco_menor posicao p1n3 ))
              (logic ( disco_medio posicao p1n2 ))
              (logic ( disco_maior posicao p1n1 )))
       (then  (logic ( robot acao p1n3_p3n1menor ))))
```

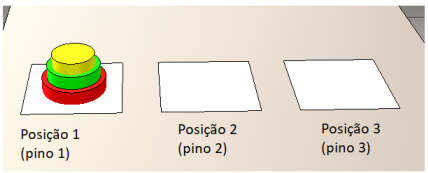

Figura 6. Exemplo de uma regra implementada no nível instintivo com o modelo do mundo.

pelo *token* "then", isto é, o comportamento escolhido para ser executado pelo nível reativo tem como estrutura: o termo "robot", o termo "acao" e o termo "p1n3_p3n1", informa que o manipulador tem que pegar o disco na posição "p1n3" e leva-lo a posição "p3n1", seguindo fluxograma de ações conforme a Figura 7

Tabela 1. Posições dos discos da Torre de Hanói utilizadas nos experimentos em relação a base do manipulador.

Localizações possíveis	Localização no espaço cartesiano	Localização e orientação desejada do manipulador em metros (x, y, z, α, β, γ)
p1n1	x = -0.5 , y = 0 e z = 0.0150	(-0.5 , 0, 0.670, -180, 0, 0)
p1n2	x = -0.5 , y = 0 e z = 0.0450	(-0.5 ,0 , 0.710, -180, 0, 0)
p1n3	x = -0.5 , y = 0 e z = 0.0750	(-0.5, 0, 0.730, -180, 0, 0)
p2n1	x = -0.5 , y = -0.25 e z = 0.0150	(-0.5 , -0.25, 0.670, -180, 0, 0)
p2n2	x = -0.5 , y = -0.25 e z = 0.0450	(-0.5 ,-0.25 , 0.710, -180, 0, 0)
p2n3	x = -0.5 , y = -0.25 e z = 0.0750	(-0.5, -0.25, 0.730, -180, 0, 0)
p3n1	x = -0.5 , y = -0.5 e z = 0.0150	(-0.5 , -0.5, 0.670, -180, 0, 0)
p3n2	x = -0.5 , y = -0.5 e z = 0.0450	(-0.5 ,-0.5 , 0.710, -180, 0, 0)
p3n3	x = -0.5 , y = -0.5 e z = 0.0750	(-0.5, -0.5, 0.730, -180, 0, 0)

Fonte: Criado pelo próprio autor.

3.4. Nível Reativo

O nível reativo é o responsável pela interação do agente com o ambiente. Ele recebe a percepção do ambiente e determina as ações que serão tomadas sobre ele. Tais ações são determinadas por um conjunto de comportamentos que serão implementados no nível

reativo do agente. A cada ciclo de ação-percepção apenas um comportamento está ativo. Os comportamentos podem ser entendidos como um conjunto de ações que devem ser executadas afim de alcançar determinados objetivos.

Neste trabalho o nível reativo foi implementado no ROS, tendo como objetivo realizar os comportamentos escolhidos e por enviar as mensagens que contém a nova configuração da Torre de Hanói (novo estado do mundo) ao nível instintivo, após cada comportamento realizado.

Para a implementação do nível reativo, criou-se um total de 3 arquivos executáveis (*Nós*) no ROS: */J2n6a300_Reativo*, */Instintivo_envia* e */Instintivo_recebe* (Figura 3). O *nó /J2n6a300_Reativo*, contém os 25 comportamentos criados, sendo ele o responsável por realizar a comunicação com o nó */J2n6a300_Driver* através dos tópicos *J2n6a300_driver /pose_action/tool_pose/action_topics* ou *J2n6a300_driver/pose_action/finger_position /action_topics* . Os *nós /Instintivo_envia* e */Instintivo_recebe*, são responsáveis por enviar a mensagem correspondente ao comportamento que será executado no nível reativo e por receber do nível reativo as modificações do estado do mundo *E(k)*. Estes blocos trocam mensagens com o *mailbox* presente no nível instintivo do AAC através de *sockets* UDP.

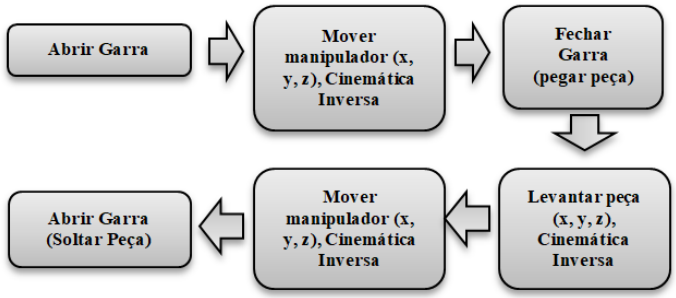

Figura 7. Fluxograma de ações de cada comportamento.

O *Nó /Instintivo_envia* (Figura 3) recebe do nível instintivo a mensagem contendo o nome do comportamento a ser executado em formato de *string* e a publica no tópico */tópico1*, que por sua vez, é subscrito por */J2n6a300_Reativo*, executando o comportamento escolhido. Após a execução de cada comportamento uma mensagem contendo a nova configuração da Torre de Hanói (estado do ambiente *E(k)*) é publicada no tópico */tópico2*, que é subscrito pelo *nó /Instintivo_recebe*, este responsável por enviar a mensagem ao nível instintivo do AAC.

4. Implementação das regras e Experimentos

A tarefa escolhida para a realização dos experimentos foi a de capacitar o manipulador JACO kinova a montar uma Torre de Hanói com 3 objetos, a partir de qualquer configuração inicial possível, dentre os 25 estados iniciais aceitos para uma Torre com 3 discos. O problema consiste em passar todos os discos de um pino para o último pino, usando um dos pinos como auxiliar, sendo considerado o final do jogo quando os três discos ficam montados, conforme a última imagem da Figura 8. Existem duas regras que devem ser seguidas na montagem da Torre:

- Só poderá movimentar um disco por vez;
- Nunca um disco maior pode estar sobre um disco menor.

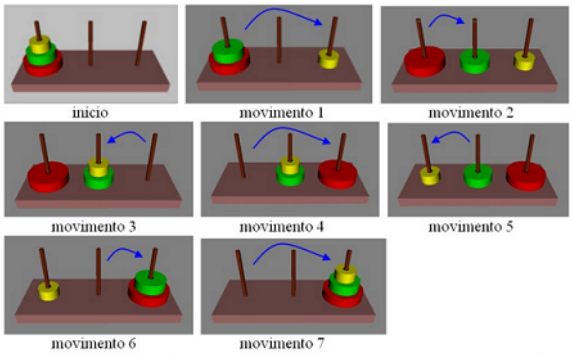

Figura 8. Exemplo e uma Torre de Hanói com três discos sendo montada.

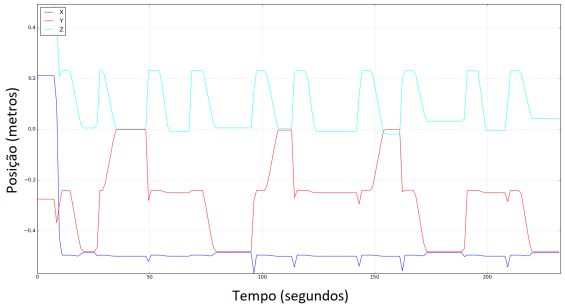

Figura 9. Posição do efetuador vs Tempo durante a montagem da Torre de Hanói do experimento II.

Foram feitos simulações e experimentos com todos os estados iniciais possíveis, onde em todos os casos o manipulador montou a Torre de Hanói com o menor número de movimentos. A seguir pode ser visto o resultado de dois experimentos com o robô JACO Kinova presente no Laboratório de Robótica na Universidade Federal da Bahia. Para o estado inicial da Torre de Hanói do primeiro experimento, o nível cognitivo escolheu o *plano_1* para ser executado no nível instintivo, podendo ser observado todas as regras utilizadas na Figura 12. A evolução no tempo da posição do efetuador durante este experimento pode ser observado na Figura 13.

O estado inicial do segundo experimento pode ser visto na Figura 15. Para este estado inicial as premissas da regra_16 presente no nível cognitivo foram aceitas, logo o plano_2 foi escolhido para ser executado pelo nível instintivo. Todas as regras para a execução deste experimento pode ser visto na Figura 10 e na Figura 14 podemos ver todos os comportamentos sendo executados pelo manipulador JACO.

A evolução no tempo da posição do efetuador durante este experimento pode ser observado na Figura 13.

5. Conclusão

Este trabalho teve como principal objetivo habilitar o manipulador JACO Kinova a execução de tarefas complexas de uma maneira completamente autônoma. Foi escolhido uma arquitetura de agente inteligente (Agente Autônomo Concorrente) para ser utilizado juntamente com o ROS para capacitar o manipulador a execução de tais tarefas. Os três níveis do AAC foram empregados, sendo que no nível cognitivo um Sistema Baseado em

```
Aguardando... (0)

Aguardando... (1)
(3 1) (1 1) (2 1)
p3n1
p1n1
p2n1
 - - - Facts Base - - -
(logic (disco_maior posicao p2n1))
(logic (disco_medio posicao p1n1))
(logic (disco_menor posicao p3n1))
(logic (robot acao p3n1_p1n2menor))

---------->robot acao p3n1_p1n2menor

Aguardando... (1)
(1 2) (1 1) (2 1)
p1n2
p1n1
p2n1
 - - - Facts Base - - -
(logic (disco_maior posicao p2n1))
(logic (disco_medio posicao p1n1))
(logic (disco_menor posicao p1n2))
(logic (robot acao p2n1_p3n1maior))

---------->robot acao p2n1_p3n1maior

Aguardando... (1)
(1 2) (1 1) (3 1)
p1n2
p1n1
p3n1
 - - - Facts Base - - -
(logic (disco_maior posicao p3n1))
(logic (disco_medio posicao p1n1))
(logic (disco_menor posicao p1n2))
(logic (robot acao p1n2_p2n1menor))

---------->robot acao p1n2_p2n1menor
```

```
Aguardando... (1)
(2 1) (1 1) (3 1)
p2n1
p1n1
p3n1
 - - - Facts Base - - -
(logic (disco_maior posicao p3n1))
(logic (disco_medio posicao p1n1))
(logic (disco_menor posicao p2n1))
(logic (robot acao p1n1_p3n2medio))

---------->robot acao p1n1_p3n2medio

Aguardando... (1)
(2 1) (3 2) (3 1)
p2n1
p3n2
p3n1
 - - - Facts Base - - -
(logic (disco_maior posicao p3n1))
(logic (disco_medio posicao p3n2))
(logic (disco_menor posicao p2n1))
(logic (robot acao p2n1_p3n3menor))

---------->robot acao p2n1_p3n3menor

Aguardando... (1)
(3 3) (3 2) (3 1)
p3n3
p3n2
p3n1
 - - - Facts Base - - -
(logic (disco_maior posicao p3n1))
(logic (disco_medio posicao p3n2))
(logic (disco_menor posicao p3n3))
(logic (robot acao pare_sucesso))

---------->robot acao pare_sucesso
```

Figura 10. Regras do plano_2, utilizadas para a realização do experimento II presente no nível instintivo.

Estado Inicial — Estado Final Objetivo

Figura 11. Estado Inicial e estado objetivo do experimento I.

```
(regra_1
        (if     (logic ( disco_menor posicao p1n3 ))
                (logic ( disco_medio posicao p1n2 ))
                (logic ( disco_maior posicao p1n1 )))
        (then   (logic ( robot acao p1n3_p3n1menor ))))

(regra_2
        (if     (logic ( disco_menor posicao p3n1 ))
                (logic ( disco_medio posicao p1n2 ))
                (logic ( disco_maior posicao p1n1 )))
        (then   (logic ( robot acao p1n2_p2n1medio ))))

(regra_3
        (if     (logic ( disco_menor posicao p3n1 ))
                (logic ( disco_medio posicao p2n1 ))
                (logic ( disco_maior posicao p1n1 )))
        (then   (logic ( robot acao p3n1_p2n2menor ))))

(regra_4
        (if     (logic ( disco_menor posicao p2n2 ))
                (logic ( disco_medio posicao p2n1 ))
                (logic ( disco_maior posicao p1n1 )))
        (then   (logic ( robot acao p1n1_p3n1maior ))))
```

```
(regra_5
        (if     (logic ( disco_menor posicao p2n2 ))
                (logic ( disco_medio posicao p2n1 ))
                (logic ( disco_maior posicao p3n1 )))
        (then   (logic ( robot acao p2n2_p1n1menor ))))

(regra_6
        (if     (logic ( disco_menor posicao p1n1 ))
                (logic ( disco_medio posicao p2n1 ))
                (logic ( disco_maior posicao p3n1 )))
        (then   (logic ( robot acao p2n1_p3n2medio ))))

(regra_7
        (if     (logic ( disco_menor posicao p1n1 ))
                (logic ( disco_medio posicao p3n2 ))
                (logic ( disco_maior posicao p3n1 )))
        (then   (logic ( robot acao p1n1_p3n3menor ))))

(regra_25
      (if   (logic ( disco_menor posicao p3n3 ))
            (logic ( disco_medio posicao p3n2 ))
            (logic ( disco_maior posicao p3n1 )))
     (then (logic ( robot acao pare_sucesso ))))
```

Figura 12. Regras do plano_1, utilizadas para a realização do experimento I presente no nível instintivo.

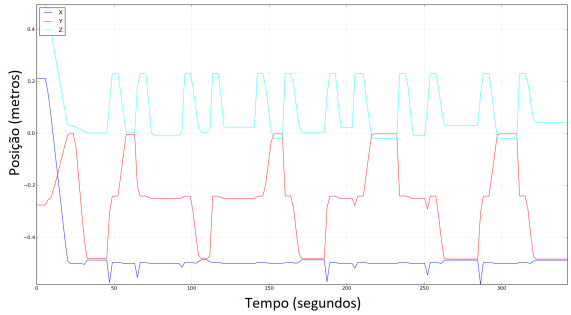

Figura 13. Posição do efetuador vs Tempo durante a montagem da Torre de Hanói do experimento I.

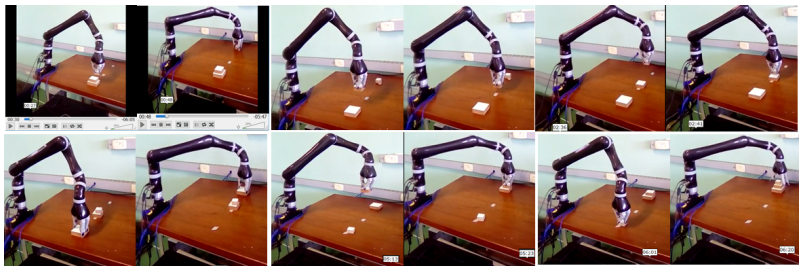

Figura 14. Manipulador JACO Kinova executando a montagem da Torre de Hanói do experimento I.

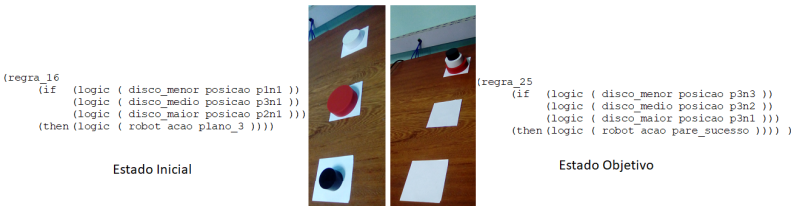

Figura 15. Estado Inicial e estado objetivo do experimento II.

Conhecimento (SBC) com uma base de regras em LPO, foi utilizado com o intuito de selecionar os planos a serem executados no nível instintivo. No nível instintivo um SBC foi implementado com três bases regras, sendo cada uma dessas um plano a ser executado. Como foi visto na seção 1, o AAC até então nunca havia sido utilizado juntamente com o ROS, sendo está uma das principais contribuições deste trabalho. O nível reativo responsável pela resposta em tempo real do agente, foi implementado então no ROS, utilizando o *metapackage* Kinova-ROS [Kinova 2017], possibilitando desta forma o controle total do manipulador, tanto no espaço cartesiano quanto no espaço de juntas.

Nos experimentos, conseguimos encontrar resultados satisfatórios, visto que, em todos os casos o manipulador executou as tarefas com o número mínimo de movimentos possíveis.

Este trabalho trouxe como principais contribuições:

- Utilização do AAC juntamente com o ROS, algo nunca feito antes e de extrema importância para futuras implementações do AAC, visto o grande número de bibliotecas e ferramentas que o ROS disponibiliza para ajudar desenvolvedores de programas robóticos a criarem sua aplicações. Outro ponto a ser considerado, é

que o código do nível reativo foi escrito totalmente em *Python* (linguagem de programação), enquanto o Expert-coop++ é escrito inteiramente em C++. O ROS possui a característica de multi-linguagem, ou seja, ele foi projetado para suportar diversos idiomas de programação como o C++, Python, Octave e Lisp. Desta forma, o número de possibilidades de utilização do AAC aumenta, visto que, sua utilização não fica restrita a apenas pessoas que programem em C++;
- Capacitou o manipulador JACO a execução de tarefas complexas de uma maneira autônoma. Com o nível reativo implementado no ROS, basta apenas rescrever as regras dos níveis cognitivos e instintivos, para capacitar o JACO a realização de novas tarefas em ambientes 3D, sem a necessidade de implementar um novo nível reativo. Como exemplo o manipulador poderia realizar tarefas como: "jogar"xadrez, simular situações em ambientes industriais (soldagem de peças, movimentar peças em estoques), manipulação de ampolas (amostras) em analises biomédicas, etc.;
- Implementou o AAC em um nova categoria de robôs (manipuladores robóticos);

Futuros trabalhos visam a utilização de cenários mais robustos, com a inserção de sistemas de visão computacional e a inserção do manipulador em uma comunidade multi-agente.

Referências

Barros, T. T. T. (2014). Modelagem e implementação no ros de um controlador para manipuladores móveis.

Bittencourt, G. (1997). In the quest of the missing link. *International Joint Conference of Artificial Inteligence*, pages 310–315.

Bittencourt, G. and Costa, A. d. (2001). Hybrid cognitive model.

Costa, A. C. P. L. and Bittencourt, G. (1998). Ufsc-team: A cognitive multi-agent approach to the robocup 98 simulator league. pages 371–376.

Costa, A. L. and Bittencourt, G. (1999). From a concurrent architecture to a concurrent autonomous agents architecture. pages 274–285.

Costa, A. L., Ferreira, D. S. F., Conceição, A. G. S., and Pappas, G. (2015). Embedding the concurrent autonomous agent into a humanoid robot. *IFAC-PapersOnLine*, 48(19):203–208.

Costa, A. L. d., Scolari, A. G., Ribeiro, T. T., and Junior, J. s. (2011). Embarcando o agente autônomo concorrente no robô móvel omnidirecional axébot. nível reativo. *X Simpósio Brasileiro de Automação Inteligente (SBAI)*.

Ferreira, D. S. F. (2014). Embarcando o agente autônomo concorrente em uma rede de microcontroladores de um robô móvel omnidirecional. Master's thesis, Dissertação de Mestrado-Universidade Federal da Bahia, Programa em Pós Graduação em Engenharia Elétrica, Salvador - BA.

Kinova (2017). *Ros Parameter Server*.

Oliveira, L. L. R. d. et al. (2013). Controle de trajetória baseado em visão computacional utilizando o framework ros. Master's thesis.