Integração de uma Estrutura Deliberativa para um Manipulador Robótico Inteligente

Guilherme I. S. Alves * **Luiz F. Pugliese** * **Diogo L. F. da Silva** *
Giovani B. Vitor *

* *Universidade Federal de Itajubá - Campus de Itabira*
Instituto de Ciências Tecnológicas
Rua Irmã Ivone Drumond, 200 - Distrito Industrial II - 35903-087
Itabira, Minas Gerais, Brasil
(e-mail: g_inaciio@hotmail.com, pugliese@unifei.edu.br),
diogoleonardof@unifei.edu.br, giovanibernardes@unifei.edu.br

Abstract: This work portrays the definition and integration of an architecture of a robotic manipulator, whose function is to organize, in an autonomous way, cubic blocks arranged on a bench. The structure is defined according to a deliberative approach that is based on SPA modeling (Sense-Plan-Act). The implementation is carried out by integrating the techniques of intelligent planning, computer vision and control of the robotic manipulator. MATLAB software is used to integrate the different modules of the system. Intelligent planning is performed by the Black Box planner who is responsible for defining the actions that must be taken during the execution of the manipulator. Therefore, the robotic arm can execute the actions established by the intelligent planner and organize the cubed blocks on the surface as defined by the user interface.

Resumo: Este trabalho retrata a definição e a integração de uma arquitetura de um manipulador robótico, cuja função é organizar de maneira autônoma, blocos cúbicos dispostos em uma bancada. A estrutura é definida segundo uma abordagem deliberativa que baseia-se na modelagem SPA (*Sense-Plan-Act*). A implementação é realizada através da integração das técnicas de planejamento inteligente, visão computacional e controle do manipulador robótico. O *software* MATLAB é utilizado para realizar a integração dos diferentes módulos do sistema. O planejamento inteligente é realizado pelo planejador *Black Box* que é responsável por definir as ações que devem ser tomadas durante a execução do manipulador. Assim, o braço robótico consegue executar as ações estabelecidas pelo planejador inteligente e organizar os blocos cúbicos na bancada como definido na interface do usuário.

Keywords: Robotic manipulator; Deliberative architecture; Intelligent planning; Computer vision

Palavras-chaves: Manipulador robótico; Arquitetura deliberativa; Planejamento inteligente; Visão computacional

1. INTRODUÇÃO

O conhecimento sobre inteligência artificial cresceu consideravelmente nos últimos anos, favorecendo o desenvolvimento de sistemas inteligentes que conseguem realizar uma tomada de decisão de forma planejada e autônoma. Com o crescimento da complexidade das tarefas a serem realizadas por esses agentes, tornou-se difícil o desenvolvimento desses sistemas, uma vez que robôs podem possuir diversos graus de liberdade (Wilkins, 2014).

Um manipulador inteligente pode ser definido como um sistema que está situado em um ambiente físico ou de *software* o qual ele faz parte. Ele deve ser capaz de extrair informações de seu ambiente realizando percepções através de sensores com objetivo de atuar em seu ambiente de forma autônoma através de atuadores, seguindo uma lista de tarefas, de forma a alcançar objetivos definidos por outro agente ou provenientes de sua própria necessidade. Inicialmente, as pesquisas na área de agentes inteligentes foram desenvolvidas com o uso de robôs móveis, porém, esta tecnologia teve sua aplicação estendida também para sistemas de *software* (Franklin and Graesser, 1996; Russell and Norvig, 2002).

Sistemas de visão computacional evoluíram ao longo do tempo, proporcionando a captura de parâmetros de câmeras por dispositivos computacionais (Faustinelli, 2021). As técnicas de aquisição, tratamento e processamento de imagens ficaram cada vez mais avançadas proporcionando também, o avanço da robótica devido à alta capacidade de sensoriamento do ambiente (Tenenbaum, 1970).

Há quatro tipos fundamentais diferentes de arquitetura para agentes inteligentes, que são classificadas como: arquitetura deliberativa, arquitetura reativa, arquitetura híbrida e arquitetura baseada em comportamento (Borgo et al., 2016; Pandey et al., 2016; Mahmoud Zadeh et al., 2017; Choi et al., 2018; Muñoz et al., 2019). Neste trabalho,

a arquitetura deliberativa será utilizada pois permite aplicações que requer do agente, a realização de objetivos de alto nível em um ambiente em que existe tempo suficiente para sintetizar uma sequência de ações e é possível obter todas as informações necessárias. Este tipo de arquitetura permite um raciocínio de ações para execução de tarefas e elabora um plano contendo essas ações a serem tomadas a fim de alcançar os seus objetivos (De Lorenci, 2004).

Existem vários domínios para o planejamento inteligente. O domínio utilizado neste trabalho é o domínio do mundo dos blocos, o qual é constituído de apenas objetos cúbicos, e o planejador inteligente é o *Black Box* (Kautz and Selman, 1999). Este planejador, através de seus resolvedores SAT (Satisfatibilidade), é capaz de desenvolver um plano de ações a ser enviado ao controlador para que o agente realize ações de pegar, empilhar e desempilhar blocos de acordo com a determinação do usuário (Kautz et al., 1992; Kautz and Selman, 1999; Hoffmann and Nebel, 2001).

Desse modo, tem-se a proposta deste trabalho que é realizar a integração de uma estrutura deliberativa para ação sobre um manipulador robótico inteligente. Tem-se como entrada do sistema, uma câmera realizando a aquisição da imagem inicial dos blocos no espaço de trabalho (Percepção). O estado de destino dos blocos é inserido pelo usuário por meio de uma interface entre usuário computador e a saída do sistema será composta pelas ações determinadas pelo planejador inteligente (Planejamento). Das ações determinadas, o manipulador robótico executará o movimento concebido pelo controlador (Ação), que é capaz de traduzir a linguagem recebida do planejador para atuação sobre os motores do manipulador robótico.

O trabalho está organizado da seguinte forma: A Seção 2 irá tratar sobre a arquitetura deliberativa descrevendo todos os elementos utilizados na integração. A Seção 3 descreverá toda a implementação e desenvolvimento dos módulos utilizados na obtenção das informações para planejamento e consequentemente execução do manipulador robótico bem como os resultados obtidos. Por fim, na Seção 4, apresenta-se as conclusões do trabalho.

2. ARQUITETURA DELIBERATIVA

O manipulador robótico precisa operar de maneira segura, sem colisões ou travamentos para poder alcançar o objetivo final. O conhecimento do ambiente de trabalho do robô é realizado por uma câmera *webcam* que terá a função de capturar o estado e organização dos blocos. O *software* MATLAB será utilizado na aquisição e processamento da imagem, gerando um arquivo de posição inicial dos blocos. Além disso, o MATLAB é responsável pela interface com um usuário, gerando o arquivo de posição final para os blocos bem como envia comandos para o controlador via comunicação USB. O planejador inteligente é uma ferramenta que utiliza dois arquivos em formato *".txt"* para criação de um plano de execução. As etapas de desenvolvimento são apresentadas na Figura 1.

2.1 Manipulador Robótico

O manipulador robótico utilizado será um robô didático conforme apresentado pela Figura 2. O robô AL5D da Lynxmotion foi desenvolvido para ensino técnico, pesquisa

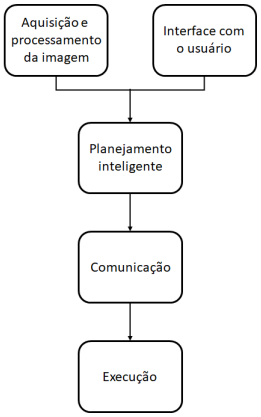


Figura 1. Etapas realizadas no projeto.

e atividades semi-industriais. Os movimentos utilizados neste trabalho pelo manipulador são rotação do ombro, cotovelo e pulso, além de fechar e abrir a garra.

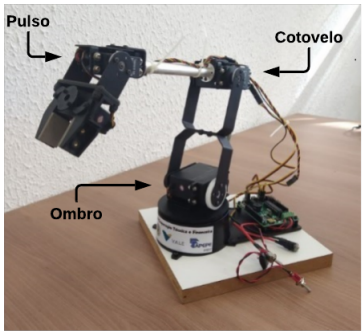


Figura 2. Manipulador Robótico AL5D.

O modelo do robô foi desenvolvido no ambiente do MATLAB, usando para isso uma caixa de ferramentas que é possível instalar no *software* (Corke, 1996). O espaço de atuação do robô é 2D devido ao ângulo de rotação da base ser sempre fixo. A cinemática direta do robô é calculada utilizando o método de Denavit-Hartenberg para obter os parâmetros das juntas e braços (Kucuk and Bingul, 2006). Esse método encontra quatro parâmetros, apresentados na Tabela 1, os quais são encontrados observando as posições de cada elo em relação ao elo anterior considerando um sistema de referência para cada eixo e as dimensões para cada estrutura conforme pode ser visto na Figura 3. Dois parâmetros descrevem o elo (a, α) e dois (θ, d) descrevem a conexão do elo com os elos adjacentes. O comprimento do elo é representado por a. O θ é o ângulo definido entre o eixo de um elo e o eixo do elo seguinte. O α é o ângulo de torção que o elo impõe desde o eixo anterior até o eixo seguinte. A distância entre elos medida ao longo do eixo da junta anterior é representado por d. Assim, étem-se a Tabela 1, a qual tem apenas os valores dos ângulos das juntas e o comprimento de cada elo.

Utilizando as matrizes de rotação R_z, R_x e as matrizes de translação T_x, T_z é possível obter a matriz A apresentada na equação (1) que é a matriz de transformação homogênea do manipulador robótico, a qual a partir dos valores dos ângulos permite a determinação da posição do objeto (Craig, 2012).

Tabela 1. Tabela de parâmetros Denavit-Hartenberg.

	θ	d	a	α
0-1	θ_1	0	0,1461	0
1-2	θ_2	0	0,1873	0
2-3	θ_3	0	0,075	0

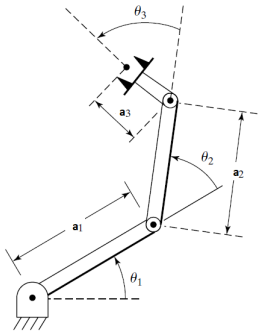


Figura 3. Eixos do Manipulador.

$$A = R_z(\theta)T_z(d)T_x(a)R_x(\alpha) =$$

$$\begin{bmatrix} cos(\theta) & -cos(\alpha)sin(\theta) & sin(\theta)sin(\alpha) & acos(\theta) \\ sin(\theta) & cos(\theta)cos(\alpha) & -cos(\theta)sin(\alpha) & asin(\theta) \\ 0 & sin(\alpha) & cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

2.2 Espaço de Trabalho

O espaço de trabalho é uma bancada conforme Figura 4, a qual contêm o manipulador robótico, uma câmera, um computador e um ambiente com cinco blocos cúbicos. Esse ambiente é todo de cor branca para auxiliar no contraste com a cor dos blocos e auxiliar na aquisição e processamento da imagem.

Os cinco blocos em formato cúbico possuem mesmo tamanho e volume, cor branca identificados com letras E, H, I, L e T na cor preta possuindo mesmo comprimento e fonte. A garra do manipulador robótico deve segurar um bloco de cada vez, realizando ações de pegar, empilhar e desempilhar.

Somente o robô deve ser capaz de alterar o espaço de trabalho e o número total de blocos cúbicos para este trabalho não pode ultrapassar a cinco. Cada bloco presente na bancada apresenta um local de ocupação, encontrando-se empilhados ou não. Essa posição somente é alterada com a ação do manipulador e o movimento sempre é esperado. Há vinte e cinco posições possíveis para a disposição dos cinco blocos.

2.3 Sensoriamento

A imagem para obter a posição inicial dos blocos é capturada por uma câmera *webcam* C922 Pro HD Stream como mostra a Figura 4, com resolução horizontal de 1920 pixels e vertical de 1080 pixels permitindo a transmissão e captura de imagens em HD com boa taxa de fps (30 a 60 quadros por segundo). Assim, a câmera é usada como sensor para capturar a imagem da disposição dos cinco blocos no ambiente de trabalho.

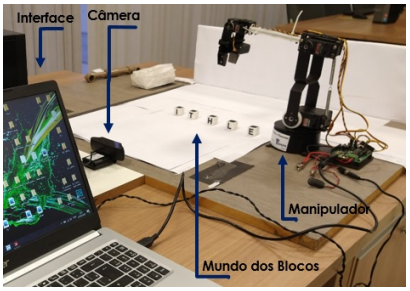


Figura 4. Disposição dos equipamentos na bancada.

A imagem é adquirida em escala de cinza e processada pelo *software* MATLAB percorrendo etapas durante o seu processamento, Figura 5. Cada etapa é melhor explicada na subseção 3.2.

Figura 5. Etapas para processamento e reconhecimento da posição inicial.

A extração de características das regiões permite identificar por meio de *scripts* do MATLAB os caracteres e a sua localização. A etapa de reconhecimento é responsável por distinguir os caracteres por meio das suas respectivas áreas. Outra função da parte de reconhecimento é definir a posição de cada letra na imagem. A posição (0,0) é o canto inferior esquerdo da imagem e é realizada uma varredura nas coordenadas da imagem da esquerda para a direita.

2.4 Interface do Usuário

Por meio de uma interface gráfica criada na *toolbox* GUIDE do *software* MATLAB, o usuário consegue realizar um teste no manipulador robótico e aquisitar a imagem em uma primeira interface criada. Já uma segunda janela permite ver a imagem inicial da posição dos blocos além de proporcionar a realização do processamento, definição da posição destino dos blocos e por fim iniciar a execução.

O destino dos blocos é inserido na interface e registrado em uma matriz que possui a função de guardar a posição de destino dos blocos. O usuário consegue apenas inserir as letras (H, L, T, I e E). Caso não seja inserido os cinco caracteres ou tenha algum espaço vazio abaixo de uma letra, aparece uma mensagem de erro ao usuário.

2.5 Agente Inteligente

O agente responsável por organizar os blocos na bancada, parte de um estado inicial com o objetivo de chegar a um estado de destino determinado por um usuário. O planejador inteligente que será utilizado é o *Black Box*, uma ferramenta que utiliza linguagem PDDL (*Planning Domain Definition Language*), que é uma linguagem de descrição de domínios de planejamento automático. Pela Figura 6, pode-se observar como é a atuação do planejador, o qual parte de arquivos de entrada e retorna um arquivo plano que contém as ações a serem executadas pelo manipulador robótico.

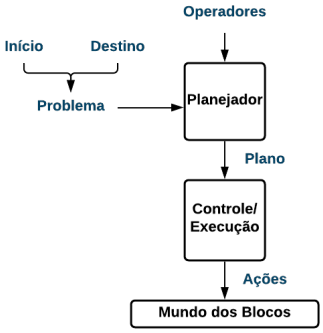


Figura 6. Arquitetura de planejamento inteligente.

Definido o arquivo problema que é composto pelo arquivo de estado inicial e o arquivo de estado objetivo, o planejador *Black Box*, através dos seus resolvedores SAT (Satisfatibilidade) desenvolve um arquivo com uma sequência de comandos responsáveis pela execução do manipulador robótico. O planejador é assim chamado pois trata os resolvedores SAT como "caixas-pretas", afinal nada se sabe a respeito dos mesmos (Kautz and Selman, 1999; Fox and Long, 2003).

O *Black Box* trabalha com dois arquivos de entrada, o arquivo "Problema.txt" e um arquivo de predicados, ações e pré-condições, "Operadores.txt" (De Lorenci, 2004). A saída do planejador é um arquivo "Plano.txt" que contém as informações de ações em alto-nível (*pickup, stack, unstack, putdown*) a serem executadas pelo manipulador robótico.

2.6 Controle e Execução

O controlador acoplado ao braço robótico é o *BotBoarduino*, um microcontrolador compatível com o Arduino feito especificamente para os robôs Lynxmotion, o qual tem a função de controlar os motores presentes nas juntas do manipulador.

A comunicação entre o controlador e o *software* MATLAB é feita via serial USB. O *software* é responsável por enviar até trinta posições por meio de *strings* para o controlador que reconhece as mesmas e através de condicionais, escreve o valor dos ângulos nas juntas. Esses valores são tabelados para cada uma das trinta posições manualmente, ou seja para cada posição ocorreram testes para coleta dos valores dos ângulos de cada junta para chegar a tal ponto. Para o movimento do manipulador ser suave, ou seja, com uma velocidade controlada, é feito um acréscimo ou decréscimo do ângulo inicial até o ângulo final com um tempo de espera.

A base do manipulador foi fixada em seu valor médio para atuar em um ambiente 2D. Na bancada há marcações onde é a localização de cada coluna para os blocos ficarem e são distanciados igualmente entre si.

Para executar a ação de segurar um bloco, o manipulador primeiro se desloca até a coluna desejada (*dummy point*) e depois até a linha onde o objeto está. Uma vez com o bloco pressionado, o manipulador move-se para o topo da coluna novamente (*dummy point*) passando sempre duas vezes pelo mesmo *dummy point*, ou seja, sempre é obedecido esses movimentos para evitar que derrube algum bloco acidentalmente. Para realizar a ação de soltar um bloco o braço se descola até a coluna desejada (*dummy point*) e posteriormente a linha para poder realizar a ação de soltar um bloco. Após soltar o bloco, o efetuador volta novamente para o *dummy point*.

Dessa maneira, há vinte e cinco posições possíveis de localização dos blocos já estabelecidas e o manipulador possui essas vinte e cinco posições mais cinco posições superiores a cada coluna também já determinadas chamadas *dummy points*, Figura 7. Sempre que for para uma posição possível de algum bloco, o manipulador passa por um ou mais *dummy points* primeiro, ou seja, para chegar até uma posição ipj (i,j = 0,1,2,3,4,5) o efetuador passa por uma posição superior primeiro (*dummy point*) e depois de realizar a ação retorna para o mesmo *dummy point*.

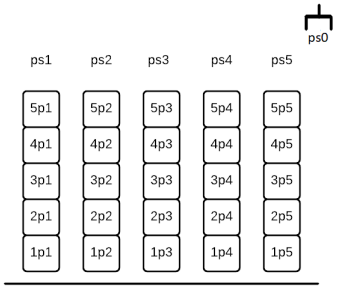


Figura 7. Posições possíveis do manipulador.

3. IMPLEMENTAÇÃO E RESULTADOS

3.1 Interface

A interface de comunicação com o usuário pode ser aberta por um arquivo executável "Planejador.exe", desenvolvido no MATLAB com o comando "deploytool".

A primeira interface, Figura 8, permite o usuário realizar a aquisição da imagem inicial e testar o manipulador para verificar a comunicação e o seu correto funcionamento. Esse movimento de teste é sempre o mesmo. Após realizar os procedimentos na primeira janela, o botão "OK" abre a segunda janela, Figura 9.

Figura 8. Janela 1.

Na segunda janela de inteface com o usuário é possível observar a imagem da posição inicial dos blocos capturada pela câmera, o botão do comando para realizar o processamento da imagem capturada e um espaço dedicado para o usuário registrar a organização final desejada

Figura 9. Janela 2.

para os blocos. As cinco letras (H, E, I, L e T) inseridas em condições reais são requisitos para gerar o arquivo "Destino.txt", Figura 10, o qual possui dados da posição final dos blocos dispostos em colunas, número da coluna, quantidade de blocos na coluna, letras e ordem das letras, respectivamente.

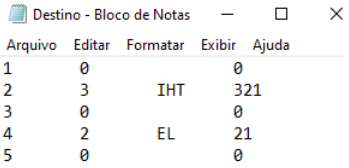


Figura 10. Composição do arquivo "Destino.txt".

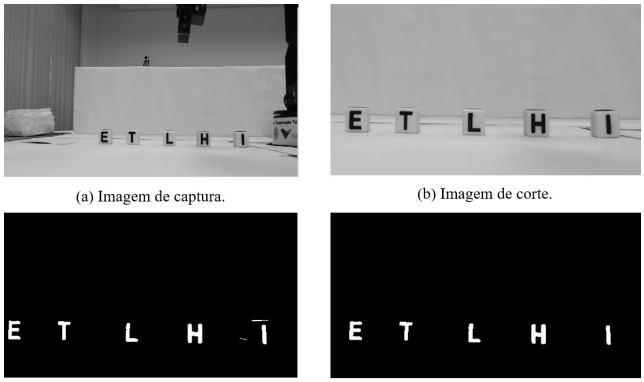
Outro comando presente na janela da segunda interface com o usuário, é um botão para a execução do sistema. Nesta etapa, utiliza-se os arquivos "Inicio.txt" e "Destino.txt" para gerar o arquivo "Problema.txt", o qual é escrito em linguagem PDDL permitindo o reconhecimento pelo planejador inteligente. O arquivo "Operadores.txt" que também é uma entrada do planejador contém um conjunto de predicados e ações definidas para o agente organizador de blocos, ou seja é um arquivo fixo.

3.2 Visão Computacional

A etapa de visão computacional utiliza a *toolbox* do MATLAB, *Image Aquisition Tool*. Depois da captura da imagem conforme a Figura 11(a), ocorre algumas etapas parciais como ajuste da imagem, extração de características tais como reconhecimento das letras e posição dos blocos no ambiente de trabalho.

Inicialmente, o ajuste da imagem é realizado com um corte da imagem inicial por um retângulo de corte através da função *"imcrop"*. Assim, obtem-se apenas a região principal dos blocos, ou seja a região de interesse conforme mostra a Figura 11(b).

Logo após é feito uma segmentação pelo método *thresholding* com o comando *"im2bw"* que utiliza um valor constante de intensidade e compara com cada pixel para converter a imagem em binária. O processo realiza a binarização da imagem pela limiarização, o qual separa as regiões da imagem em objeto e fundo facilitando a análise (Sezgin and Sankur, 2004). Se a intensidade do pixel for maior que um valor pré-definido, o pixel será branco caso contrário, será preto. Posteriormente, os tons da imagem (preto e branco) são invertidos como pode ser visto na Figura 11(c).



(a) Imagem de captura. (b) Imagem de corte.

(c) Imagem após limiarização. (d) Imagem final.

Figura 11. Imagem de captura, imagem de corte, imagem após limiarização e imagem final.

Com a imagem em preto e branco, o comando *"bwareaopen"* é responsável por remover todos os elementos da imagem que possuem um número de pixels menor do que a área da letra "I", a qual é o caractere com o menor número de pixels. A Figura 11(d) mostra a imagem processada com as regiões de interesse, ou seja, apenas as cinco letras para poder extrair as informações atreladas a cada letra ou região.

Na etapa de extração de características, é identificada regiões conectadas em uma imagem com o comando *"bwlabel"* que retorna uma matriz de rótulos e o número de regiões separadas. Caso a quantidade de regiões for diferente de cinco, o programa retorna um erro. Por fim, as características de cada região são geradas pelo comando *"regionprops"* como mostra a Figura 12 que retorna a *Área* em escalar que corresponde a soma do número total de pixels da região, o *BoundingBox* que é a menor caixa de pixels contendo a região e *Image* que é uma imagem de mesmo tamanho que a caixa delimitadora da região.

```
Caract(1).Area: 452;
Caract(1).BoundingBox: [42.5,182.5,23,31];
Caract(1).Image: [31x23 logical];

Caract(2).Area: 313;
Caract(2).BoundingBox: [137.5,183.5,25,31];
Caract(2).Image: [31x25 logical];

Caract(3).Area: 307;
Caract(3).BoundingBox: [267.5,186.5,23,31];
Caract(3).Image: [31x23 logical];

Caract(4).Area: 584;
Caract(4).BoundingBox: [386.5,187.5,27,33];
Caract(4).Image: [33x27 logical];

Caract(5).Area: 247;
Caract(5).BoundingBox: [527.5,189.5,10,32];
Caract(5).Image: [32x10 logical];
```

Figura 12. Características de cada região.

Com as características, é possível reconhecer os caracteres. São utilizadas a *Área* e a *Image*, as quais descrevem os pixels de cada região. Cada região é dividida entre quatro partes, parte superior, parte inferior, esquerda e direita, abrangendo uma certa quantidade de linhas e colunas, ou seja, não é preciso utilizar todos os pixels apenas os de interesse.

O caractere "I" possui a menor área total, logo é o mais fácil de ser reconhecido. Para reconhecer as outras letras são utilizados os pixels das quatro partes de cada região.

A letra “T” possui uma área da região superior maior do que sua área da região inferior. Por outro lado, a letra “L” possui uma área inferior maior do que sua área superior. Os caracteres “H” e “E” possuem o maior número de pixels, no entanto a área da parte esquerda do caractere “E” é maior do que a área do lado direito. Por fim, caso não entre em nenhuma condição para registro de caractere, a letra registrada é “H”. Logo, o processo de extração de característica possibilitou identificar e reconhecer qual letra está em cada região.

Com a característica *BoundingBox* extraída, que são as coordenadas na imagem da menor caixa contendo a região, há a possibilidade de calcular o centro geométrico da região e comparar com coordenadas da imagem. A origem (0,0) é na parte inferior esquerda da imagem. Com o centro geométrico das regiões é possível obter a localização de cada bloco.

Desse modo, a imagem é dividida em cinco regiões por colunas (p1, p2, p3, p4 e p5) e é realizada uma comparação com o centro geométrico de cada letra, permitindo identificar a posição de cada bloco na coordenada x da imagem.

Caso haja dois ou mais caracteres na coordenada x, que representa as colunas, é feita uma comparação com os valores da coordenada y, que representa as linhas, e o maior valor representa o topo da pilha de blocos. Por fim, com o reconheciento feito, um arquivo “Inicio.txt” é criado, como mostra a Figura 13, para registro das informações dispostas em colunas: número da coluna, quantidade de blocos por coluna, letras e ordem das letras, respectivamente.

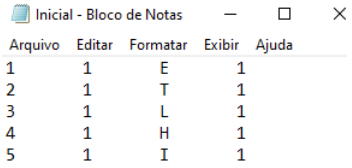


Figura 13. Composição do arquivo “Inicio.txt”.

3.3 Planejador Inteligente

O planejador inteligente é o responsável por elaborar as ações ou passos a serem realizados pelo manipulador robótico a fim de organizar os blocos de acordo com o desejo do usuário.

O botão referente ao comando “Executar” da segunda janela de interface é responsável por iniciar o planejamento inteligente, elaborando o plano de ação e logo após, envio dos comandos para o controlador de maneira a obedecer os passos descritos pelo planejador.

Para a elaboração do arquivo “Plano.txt” que contém as ações a serem realizadas, como mostra a Figura 14, o planejador necessita de um arquivo denominado “Problema.txt”, gerado, por meio de *scripts*, pela união das informações dos arquivos “Inicio.txt” e “Destino.txt”, e de um arquivo denominado “Operadores.txt”, o qual contém uma lista de predicados, ações e pré-condições. Os dois arquivos de entrada no *Black Box*, “Problema.txt” e “Operadores.txt”, utilizam a linguagem PDDL. O arquivo “Operadores.txt” é um arquivo interno do sistema de controle que implementa o planejamento inteligente contendo as ações, os blocos e suas posições. Já o arquivo “Problema.txt” contém a relação inicial dos blocos e o objetivo a ser alcançado. O planejador inteligente elaborou oito passos a serem executados em sequência para o exemplo apresentado.

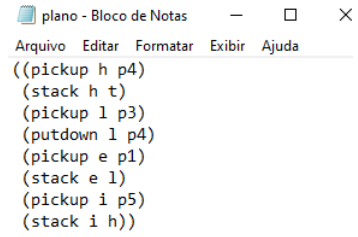


Figura 14. Composição do arquivo “Plano.txt”.

3.4 Execução do Manipulador

A etapa de execução recebe o arquivo “Plano.txt” oriundo do planejador. As funções de atuação para o manipulador são representadas por ações de alto nível (Ghallab et al., 2004). Cada uma das ações de alto nível definidas para o agente (*stack*, *unstack*, *put down* e *pick up*) corresponde a uma combinação de movimentos do robô para posições específicas e movimentos de abertura e de fechamento da garra do braço conforme apresentado na Tabela 2. Existem 25 posições “ipj” (i,j=1, 2, 3, 4, 5) possíveis pré-determinadas onde o manipulador pode efetuar uma ação (agarrar ou liberar) sobre um bloco e foram definidas 6 posições “psj” (j= 0, 1, 2, 3, 4, 5) adicionais que funcionam como *dummy points* e servem para criar o movimento vertical requerido para abaixar e levantar a garra. Toda vez que o sistema é inicializado ou termina a execução de uma requisição, o braço se posiciona em “ps0”, com a garra aberta, conforme esquemático da Figura 7. A Tabela 2 mostra as ações possíveis para o manipulador e as variáveis do bloco, posição e se o bloco está sobre outro representados por “?ob”, “?pos” e “?underob” respectivamente. Além disso exibe como é traduzido essas ações em movimento.

O controlador recebe posições e escreve um valor de ângulo já pré-determinado em cada motor, ou seja, os valores de cada ângulo foram tabelados para cada uma das 30 posições de atuação. A velocidade de atuação do manipulador pode ser controlada modificando o tempo de atraso de acréscimo ou decréscimo dos ângulos.

Para explicitar uma situação de operação do manipulador robótico após a geração do plano de atuação é utilizada a Figura 14 para obedecer as operações de execução e a Figura 7 para observar quais as posições o efetuador vai exibir e que serão enviadas ao controlador.

Encerrando o processo, o manipulador robótico volta para posição inicial, “ps0”, que é a mesma depois de realizar o teste. A realização de toda a execução pode ser consultada pelo link abaixo:

`https://tinyurl.com/y5ay5apg`

4. CONCLUSÃO

Com a realização deste trabalho, é evidente que aplicações de manipuladores robóticos possuem certo grau de com-

Tabela 2. Comandos do robô para cada ação gerada pelo manipulador robótico.

Ação do plano	Comandos que implementam a ação
pick up (?ob ?pos)	(Verifica posição ipj de ?ob) - Move para a posição psj - Move para a posição ipj - Fecha a garra - Move para a posição psj
put down (?ob ?pos)	(Verifica posição ipj de ?pos) - Move para a posição psj - Move para a posição ipj - Abre a garra (Atualiza a posição ipj de ?ob para a posição ipj corrente) - Move para a posição psj
stack (?ob ?underob)	(Verifica posição ipj de ?underob) - Move para a posição psj - Move para a posição (i+1)pj - Abre a garra (Atualiza a posição ipj de ?ob para a posição ipj corrente) - Move para a posição psj
unstack (?ob ?underob)	(Verifica posição ipj de ?ob) - Move para a posição psj - Move para a posição ipj - Fecha a garra - Move para a posição psj

plexidade. O sistema foi baseado no modelo SPA (*Sense-Plan-Act*), o qual consegue identificar e perceber aspectos do ambiente por meio de visão computacional e planejar de forma lógica uma solução por meio do planejador inteligente *Black Box*, para que o estado desejado pelo usuário seja alcançado. Para organizar os blocos de outra maneira, é preciso realizar todo o processo novamente. Assim, a repetibilidade do processo pode ser realizada e novas disposições dos blocos podem ser efetudas.

A estrutura de percepção do ambiente de trabalho e reconhecimento da imagem está sujeita a interferências de iluminação. Além disso, o sistema projetado não estará preparado para atuar caso um objeto diferente dos blocos esteja sobre a bancada de trabalho. Assim, o agente inteligente não previne eventuais falhas como essa, sendo que se o "objeto estranho" possuir uma quantidade de pixels maior do que um valor específico, uma mensagem de erro aparece na tela, pois o sistema de processamento de imagem detectou mais do que cinco regiões de interesse.

Como proposta para trabalhos futuros, pretende-se expandir o número de blocos e desse modo o sistema de visão necessita reconhecer outras letras do alfabeto ou outro tipo de caractere e o planejamento da tomada de devisão deve ser capaz de calcular e implementar ações para essa modificação. Além disso, podem ser utilizados outros tipos de planejadores inteligentes e com isso, análieses e comparações de desempenho podem ser realizadas. A partir do conceito de cinemática inversa também é possível desenvolver um sistema de controle e rastreamento de trajetória.

REFERÊNCIAS

Borgo, S., Cesta, A., Orlandini, A., and Umbrico, A. (2016). A planning-based architecture for a reconfigurable manufacturing system. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*. ICAPS.

Choi, D., Langley, P., and To, S.T. (2018). Creating and using tools in a hybrid cognitive architecture. In *2018 AAAI Spring Symposium Series*. AAAI.

Corke, P.I. (1996). A robotics toolbox for matlab. *IEEE Robotics & Automation Magazine*, 3(1), 24–32.

Craig, J.J. (2012). *Robótica*. Editora Pearson, São Paulo.

De Lorenci, E.V.N. (2004). *Definição e implementação de uma arquitetura deliberativa para um agente inteligente robótico*. Master's thesis, Universidade Federal de Itajubá - UNIFEI, Itajubá - MG.

Faustinelli, A.B. (2021). *Sistema de alinhamento de prótese de membro inferior utilizando visão computacional*. Master's thesis, Universidade Estadual Paulista - UNESP, Ilha Solteira - SP.

Fox, M. and Long, D. (2003). Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*, 20, 61–124.

Franklin, S. and Graesser, A. (1996). Is it an agent, or just a program?: A taxonomy for autonomous agents. In *International Workshop on Agent Theories, Architectures, and Languages*, 21–35. Springer.

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: theory and practice*. Elsevier.

Hoffmann, J. and Nebel, B. (2001). The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14, 253–302.

Kautz, H. and Selman, B. (1999). Unifying sat-based and graph-based planning. In *IJCAI*, volume 99, 318–325.

Kautz, H.A., Selman, B., et al. (1992). Planning as satisfiability. In *ECAI*, volume 92, 359–363. Citeseer.

Kucuk, S. and Bingul, Z. (2006). *Robot kinematics: Forward and inverse kinematics*. INTECH Open Access Publisher.

Mahmoud Zadeh, S., Powers, D.M., and Sammut, K. (2017). An autonomous reactive architecture for efficient auv mission time management in realistic dynamic ocean environment. *Robotics and autonomous systems*, 87, 81–103.

Muñoz, P., R-Moreno, M.D., Barrero, D.F., and Ropero, F. (2019). Mobar: A hierarchical action-oriented autonomous control architecture. *Journal of Intelligent & Robotic Systems*, 94(3-4), 745–760.

Pandey, A., Moreno, G.A., Cámara, J., and Garlan, D. (2016). Hybrid planning for decision making in self-adaptive systems. In *2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, 130–139. IEEE.

Russell, S. and Norvig, P. (2002). *Artificial intelligence: a modern approach*. Prentice Hall.

Sezgin, M. and Sankur, B. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging*, 13(1), 146–166.

Tenenbaum, J.M. (1970). Accommodation in computer vision. Technical report, Stanford Univ Ca Dept of Computer Science.

Wilkins, D.E. (2014). *Practical planning: extending the classical AI planning paradigm*. Elsevier.