UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO
CURSO DE ENGENHARIA MECATRÔNICA E ENGENHARIA DA COMPUTAÇÃO

RELATÓRIO

ATYSON JAIME DE SOUSA MARTINS: Nº 20190153956
JOSE LINDENBERG DE ANDRADE: Nº 20200150293
JÚLIA COSTA CORRÊA DE OLIVEIRA: Nº 20200149087
SAMUEL CAVALCANTI: Nº 20200149318

Natal-RN
2021

ATYSON JAIME DE SOUSA MARTINS: Nº 20190153956
JOSE LINDENBERG DE ANDRADE: Nº 20200150293
JÚLIA COSTA CORRÊA DE OLIVEIRA: Nº 20200149087
SAMUEL CAVALCANTI: Nº 20200149318

Relatório presentado à disciplina de Sistemas Robóticos Autônomos, correspondente à avaliação parcial da 2º unidade do semestre 2021.2 do curso de Engenharia Mecatrônica e Engenharia da Computação da Universidade Federal do Rio Grande do Norte, sob orientação do **Prof. Pablo Javier.**

Professor: Pablo Javier Alsina.

Natal-RN
2021

RESUMO

Relatório

Lista de Figuras

Sumário

1 INTRODUÇÃO

Com o intuito de desenvolver Planejadores de Caminhos para um robô móvel, que permitam ao mesmo executar movimentos especificados em espaço povoado de obstáculos, sem colidir com os mesmos. Foi incluído obstáculos poligonais no espaço de trabalho simulado e considerado que o robô tenha um formado retangular. Esse trabalho foi dividido em três metas, a primeira meta visa implementar um algoritmo que mapeie os obstáculos do espaço de trabalho para o espaço de configuração e visualizar o caminho no espaço de configuração e no espaço de trabalho. A segunda meta consiste em a partir dos polígonos convexos no espaço de configuração que representam os obstáculos deve-se gerar um grafo, a qual deve-se implementar um algoritmo que navegue nesse grafo e busque encontrar os caminhos a qual um controlador deve ser capaz de seguir-lo. A terceira meta busca implementar um planejador de caminhos baseado em campos de potenciais qual deve-se ser capaz de visualizar esses caminhos no espaço de configuração e trabalho, além do fato de implementar um controlador que permita seguir o caminho.

2 PRIMEIRA META

Para que faça sentido ter um espaço de configuração, foram adicionados no simulador, obstáculos que podem ser vistos na figura 1. Foram adicionados pequenos objetos planares da cor roxa que representam os vértices de cada obstáculo, de modo que se possa automatizar o processo de coleta desses vértices e visualizá-los durante a simulação, o mesmo foi feito com o robô, sendo que no caso do robô, o retângulo que o representa é maior do que ele para que seja possível o robô rotacionar dentro esse retângulo. Os obstáculos podem ser vistos na figura 2 e o robô na figura 3.

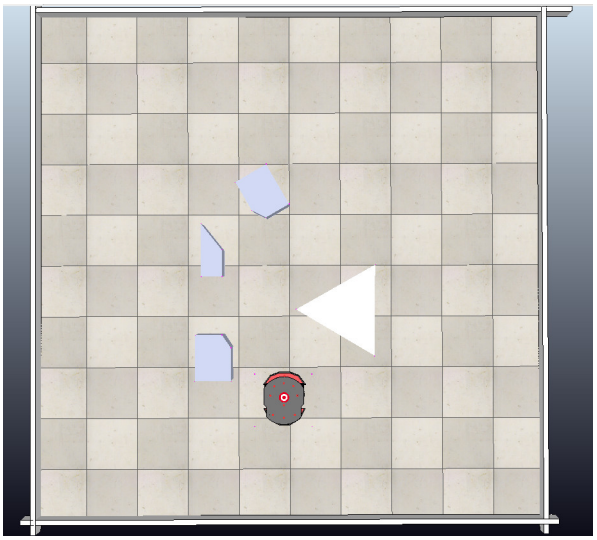

Figura 1: Espaço de trabalho CoppeliaSim com Obstáculos

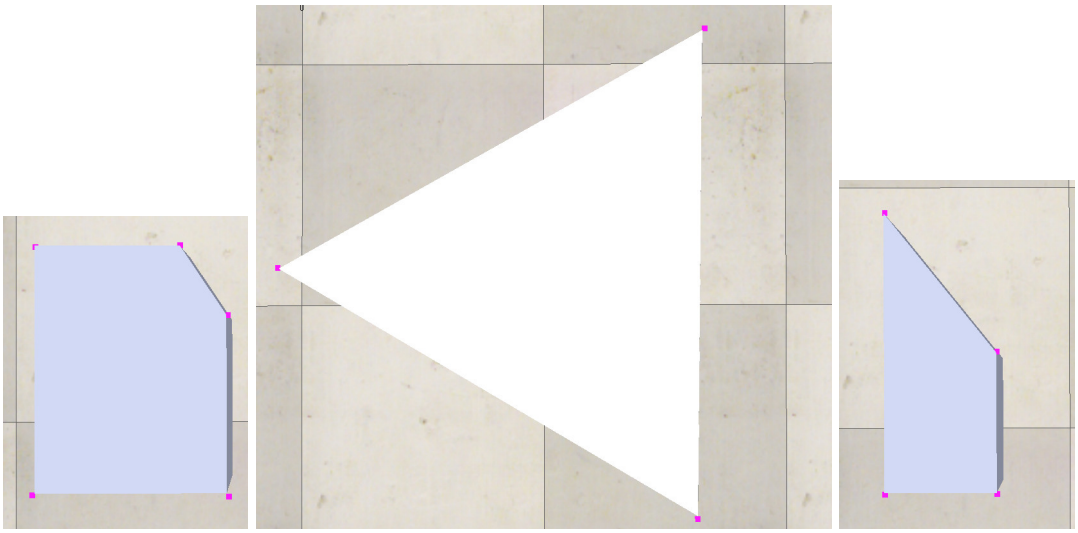

Figura 2: Obstáculos

O motivo pelo qual o robô tem um retângulo que lhe representa maior que o seu verdadeiro tamanho é para facilitar na geração do espaço de configuração, uma vez que tendo o espaço extra o robô pode girar e dessa forma, o espaço de configuração pode ser obtido apenas transladando o retângulo. Tendo todos os vértices necessários, foi gerado o espaço de configuração seguindo o seguinte algoritimo:

- Escolhendo o sistema de coordenadas tendo como vista o centro geométrico do polígono do robô, caso esse polígono seja convexo, a média dos vértices é o centro geométrico.
- Refletindo os vértices do Robô, ou seja, dada o conjunto de vértices do robô A é calculado $-A = \{-a | a \in A\}$
- Calculado a soma de Minkowski entre o conjunto $-A$ e um conjunto de vértices de um o obstáculo P_i
- Calculado o convex hull do resultado da soma de Minkowski, onde o resultado do convex hull é o conjunto de vértices de um obstáculo no espaço de configuração.

Esse algoritmo foi retirado de Paulina Varshavskaya, a qual foi passado o texto original para uma versão escrita em markdown [5].

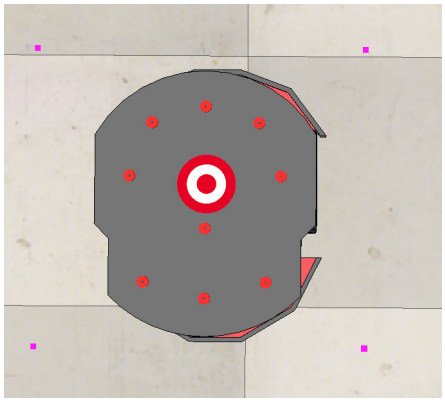

Figura 3: Robô

Utilizando o script que extrai os vértices da simulação, foi criado o gráfico do espaço de configuração. Para observarmos como seria um caminho no espaço de trabalho, foi utilizando, algoritmo de Diogo que cria um caminho usando polinômios interpoladores de terceiro grau, o espaço de trabalho e o caminho gerado pode ser visto na figura 4.

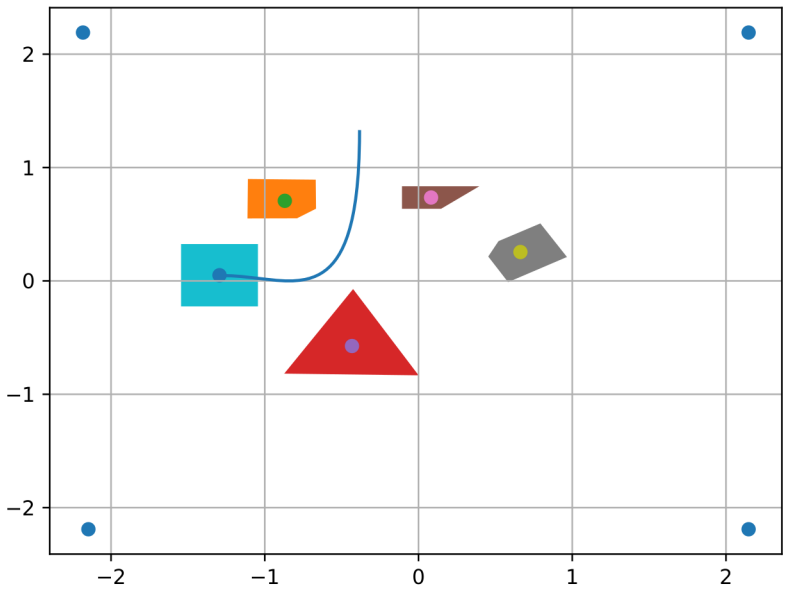

Figura 4: Gráfico do espaço de trabalho com o caminho gerado pelos algoritmo de Diogo.

Utilizando o algoritmo e o robô em formato retangular foi obtido o espaço de configuração da figura 5. Podemos observar nessa figura um caminho gerado pelos polinômios interpoladores de Diogo demonstra existir um risco do robô colidir com os obstáculos.

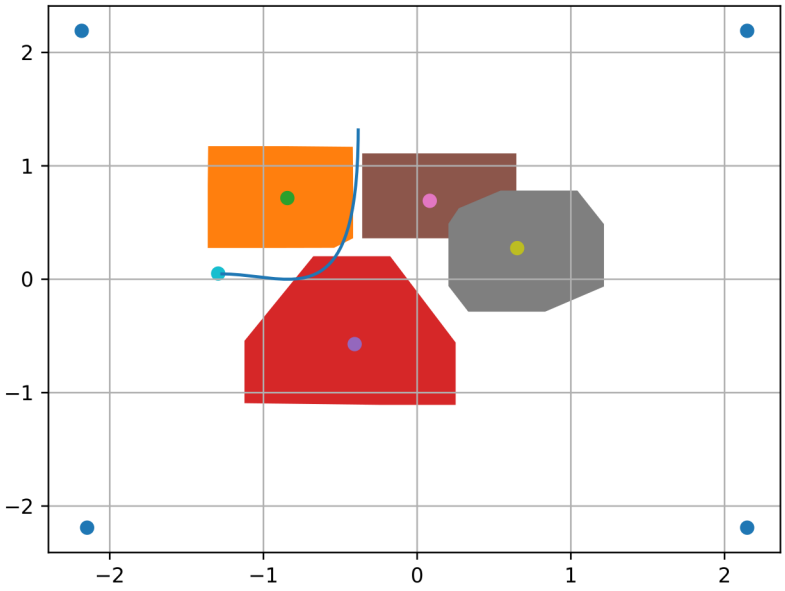

Figura 5: Gráfico do espaço configuração gerado a partir do espaço de trabalho.

3 SEGUNDA META

Nessa meta é mapeado o espaço de configuração para a geração de um grafo e utilizado o algoritmo A^* para gerar um caminho entre um ponto a outro nesse grafo. Para gerar o grafo foi criada uma malha igualmente espaçada 50x50, totalizando um grafo com 2500 nós, no entanto parte desses nós estão dentro dos polígonos do espaço de configuração que representam os obstáculos, ou seja, esses pontos não geram um caminho válido. Para resolver esse problema, foi utilizado um algoritmo que verifica se um ponto está ou não contido em um polígono. O algoritmo consiste em três passos simples:

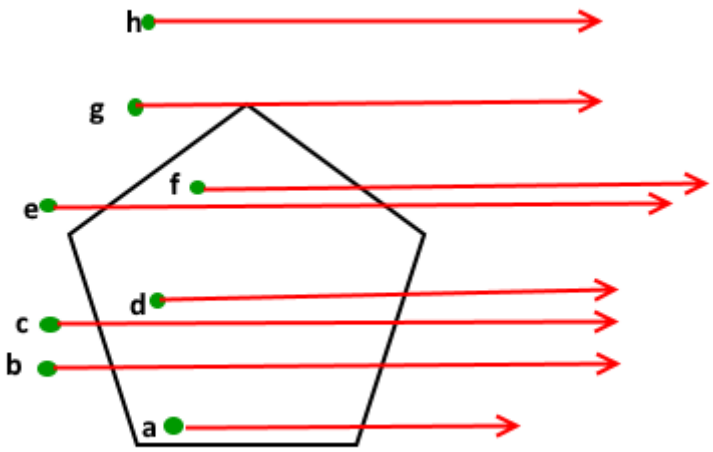

Figura 6: Polígono exmplo para o algoritmo

- desenhe uma linha horizontal a partir do lado direito do ponto até infinito
- contar o número de vezes que essa linha intersecta com os lados do polígono
- Se o ponto estiver dentro do polígono, o número de interseções é impar ou o ponto se encontra em cima do lado do polígono, caso contrário o ponto está fora do polígono.

Esse algoritmo foi retirado do geeksforgeeks [6]. Utilizando a malha, o algoritmo para calcular se um ponto está ou não dentro do polígono e A^* para determinar o melhor caminho, foram geradas duas figuras, a figura 7, que demonstra o funcionamento do o algoritmo de colisão do polígono com o ponto e também mostra, através do seguimento laranja, o caminho sugerido pela A^*.

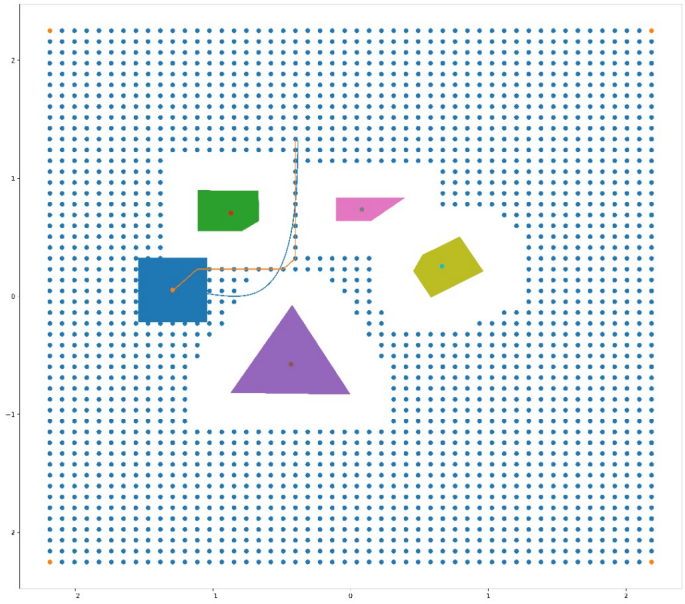

Figura 7: Espaço de Trabalho com um grafo do tipo malha 50x50

Já a figura 8 mostra as mesmas informações só que no espaço de configuração. Também foi feito um vídeo mostrando o controlador Frederico no seu trabalho de seguir o caminho gerado pela A^* no espaço de configuração [2]

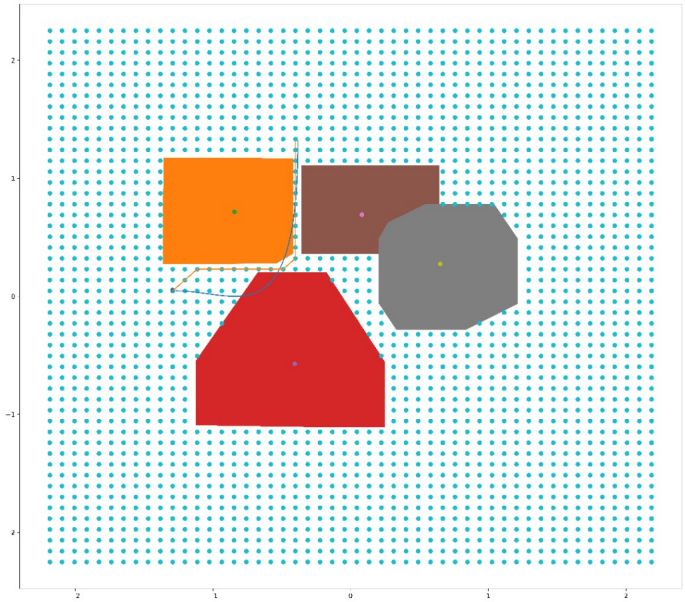

Figura 8: Espaço de Configuração com um grafo do tipo malha 50x50

4 TERCEIRA META

Nessa meta foi utilizado a ideia de campos potenciais artificiais para criação de caminhos. Onde a orientação final é uma partícula atrativa, que emite um campo artificial P_a dado por:

$$P_a = \frac{1}{2} K_p ||s_i - s_g|| \tag{1}$$

onde K_p é uma constante que em nossos testes foi assumido 0.5 e $||s_i - s_g||$ é a distância euclidiana entre as posição s_i e a posição objetivo (goal) s_g, ou seja, quando maior a distância entre s_i e s_g, maior é o valor campo. Já os obstáculos emitem um campo artificial repulsivo P_o dado por:

$$\begin{array}{ll} \frac{1}{2}\eta(\frac{1}{||s_i - s_o||}\frac{-1}{RR})^2 & \text{se } \frac{||s_i - s_o||}{2} \leq RR \\ 0 & \text{caso contrário} \end{array} \tag{2}$$

onde, nos nossos testes $\eta = 110$ e RR é um suposto raio do robô que no caso é a distância entre o centro do quadrilátero com o vértice mais distante do centro. Nó caso foi adotado que todo vértice de um obstaculo emite um valor de campo P_o, por exemplo, o obstáculo cujo o formato é de um triângulo possui três fontes de campo repulsivo. Nessa etapa foi utilizado 2 algoritmos baseados em campo potencial, o primeiro foi a decida do gradiente, e a segundo foi o algoritmo A^* com modificações. Uma vez que o valor do campo tende a decrescer com a distância entre a posição atual do robô e a posição desejada, é razoável pensar seguindo o menor valor do campo chega-se na posição desejada, mas a decida do gradiente só demonstrou atingir o objetivo em cenários mais simples como no caso da figura 9.

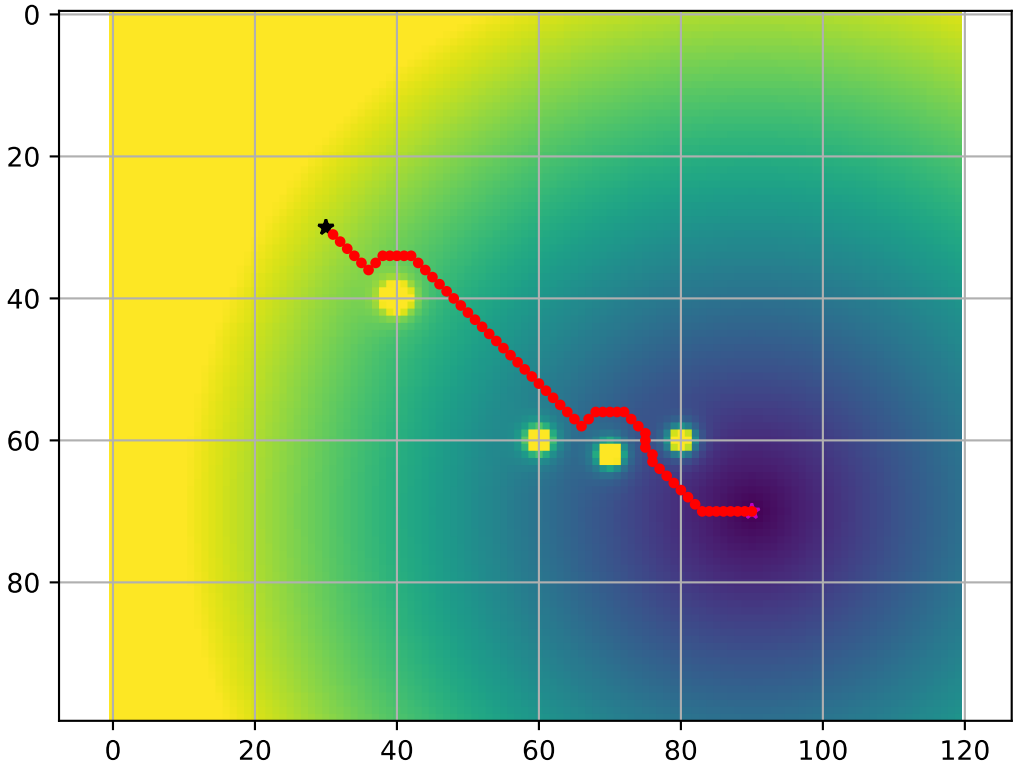

Figura 9: Cenário simples com obstáculos bem afastados

Já no nosso cenário, onde temos três obstáculos muitos próximos um dos outros. Observamos que a decida do gradiente não funciona, sua tentativa pode ser observada pela sequência de pontos vermelhos na figura 10 que em um dado momento se recusa a se afastar do ponto cinza, para futuramente encosta-la. Então aproveitando a implementação do algoritmo A^*, foram feitas duas modificações utilizando o campo potencial artificial. A primeira foi a o custo entre dois nós da malha que agora é dada pela seguinte equação:

$$C(o,t) = |P(t) - P(o)| \tag{3}$$

onde $|P(t) - P(o)|$ representa o módulo de $P(t) - P(o)$, o é o nó de origem, t o nó de destino e $P(n)$ é o valor do campo de potencial do nó n, então no caso o custo entre dois nós é a diferença em modulo dos valores dos campos artificiais do nó alvo(target) t e nó origem o, dessa forma a A^* vai buscar minimizar além dos menores valores do campo, como também as menores transições, lembrando que devido a equação de repulsão no momento que a distância da posição atual s_i é duas vezes menor que menor o raio do robô, então ocorre uma transição elevada, que as modificações da A^* tendem a evitar. Também foi feita uma segunda modificação, que no momento que o algoritmo busca os nós vizinhos do nó que está sendo visitado, é desconsiderado o vizinho que possui um valor de seu campo for maior que T_p, onde nos nossos testes $T_p = 50$. Dessa forma o algoritmo busca o menor potencial, com a transição menos brusca e em nós com valores de campo menores que 50. Podemos ver o caminho gerado pelo algoritmo A^* pela sequência de pontos da cor ciano na figura 10. Cores claras no mapa

de calor significa um alto valor de campo, cores mais escuras um baixo valor de campo.

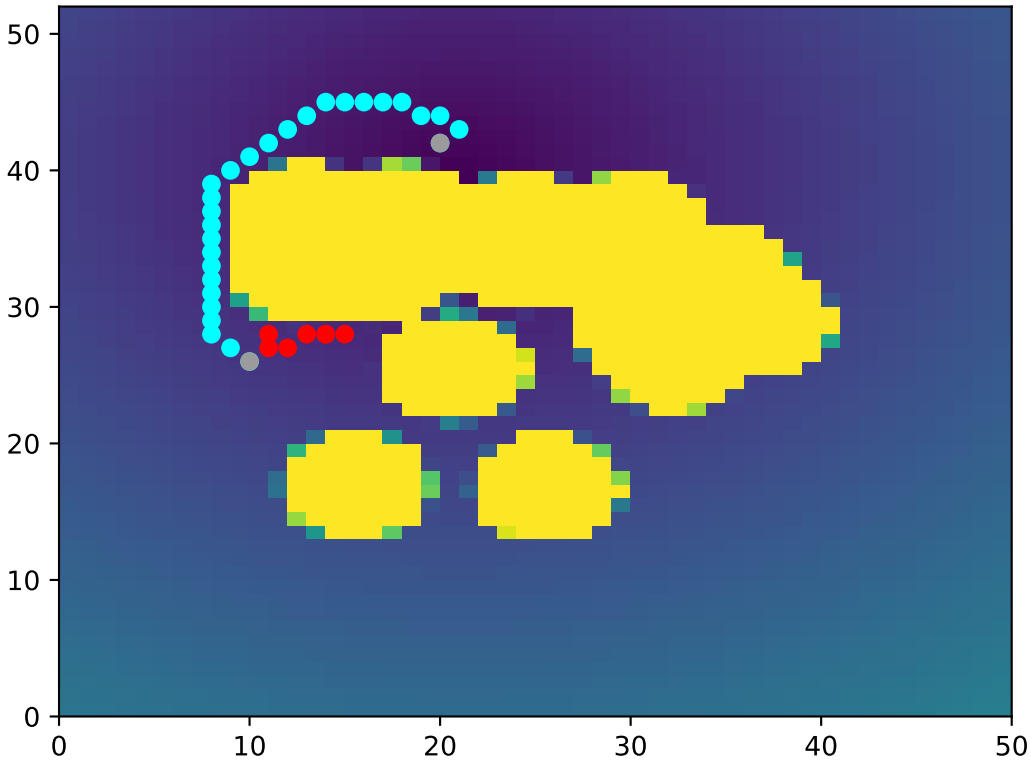

Figura 10: Mapa de calor do campo artificial

No espaço de configuração o caminho da A^* é a sequência de pontos amarelos e o caminho de pontos cianos é o algoritmo da decida do gradiente que ambos podem ser vistos na figura 11, essa mesma sequência de cores foi adotada no gráfico da figura 12 que mostram o caminho no espaço de configuração. Também foi feito um vídeo mostrando o controlador Frederico buscando seguir o caminho gerado pela A^* no campo artificial potencial [3]

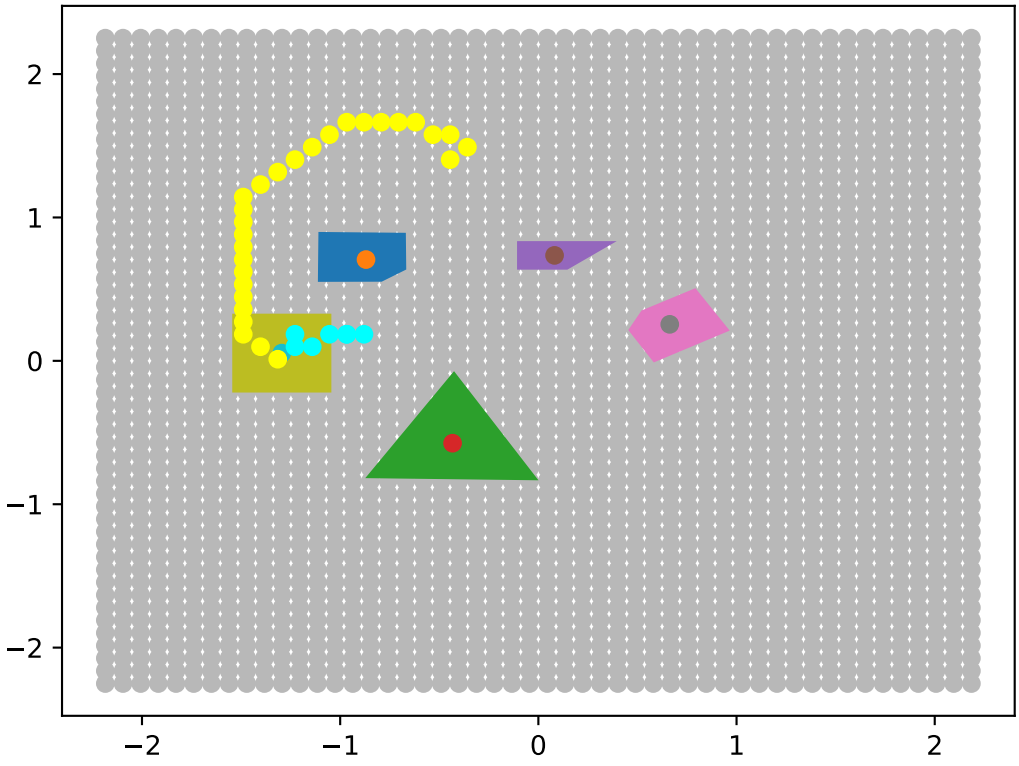

Figura 11: Caminho do campo artificial no espaço de trabalho

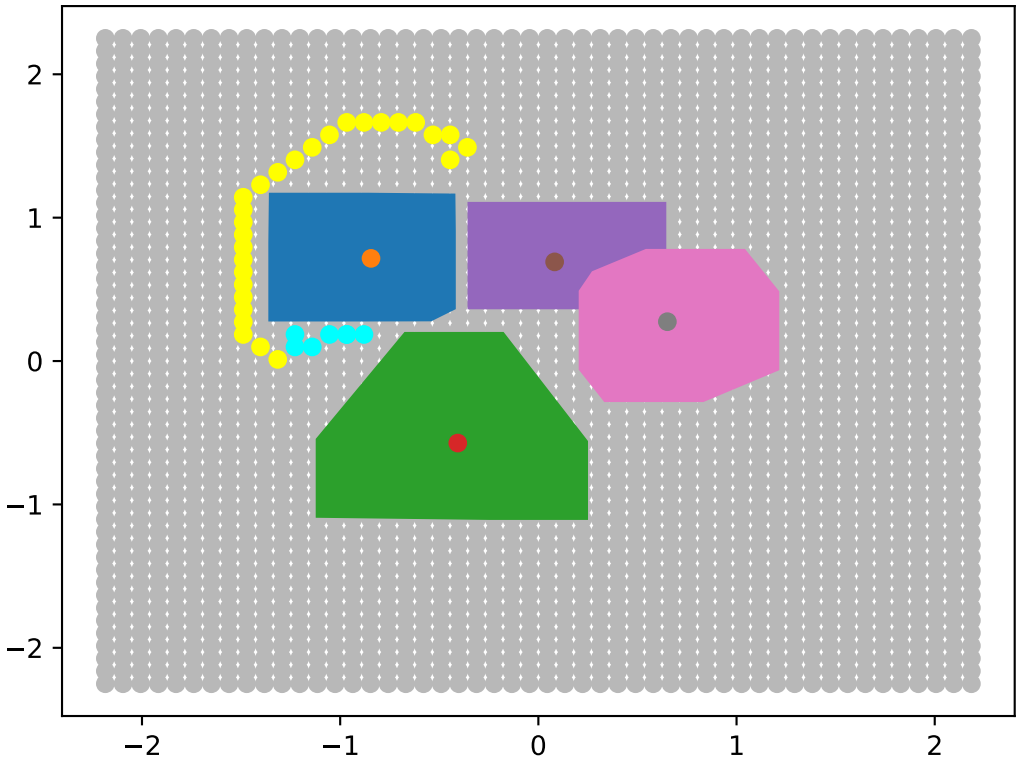

Figura 12: Caminho do campo artificial no espaço de configuração

Referências

[1] Simulador CoppeliaSim. Coppelia Robotics, 2021. Disponível em: https://www.coppeliarobotics.com/. Acesso em 22 de nov. de 2021.

[2] Controlador Federico Seguindo Caminho gerado pela A^* no Espaço de Trabalho: https://youtu.be/MZlBS-d6O-Y. Acesso em 10 de jan. de 2022.

[3] Controlador Federico Seguindo Caminho gerado pela A^* no Campo Potencial Artificial: https://youtu.be/jtnPh2pygI4. Acesso em 10 de jan. de 2022.

[4] Repositório contento todo código do projeto: https://github.com/samuel-cavalcanti/controle_cinematico_sistemas_autonomos_ufrn. Acesso em 10 de jan. de 2022.

[5] Obstacles in Configuration space by Paulina Varshavskaya: https://github.com/samuel-cavalcanti/controle_cinematico_sistemas_autonomos_ufrn/blob/main/docs/notes_on_configuration_space.md

[6] Como verificar se um ponto está contido em um polígono: https://www.geeksforgeeks.org/how-to-check-if-a-given-point-lies-inside-a-polygon/ Acesso em 10 de jan. de 2022