UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
CURSO DE MESTRADO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
PPGEEC2324 - SISTEMAS ROBÓTICOS AUTÔNOMOS

PROJETO 1 - Meta 2

GUILHERME FÉLIX DE MEDEIROS BRANDT - Nº 20241009450
JOSE LINDENBERG DE ANDRADE - Nº 20241009460
LARISSA SOARES DE SOUZA - Nº 20241028376
PIETRO AUGUSTO DE ALBUQUERQUE LIRA E SILVA - Nº 20241028474
RAFAEL AUGUSTO DE OLIVEIRA GUEDES - Nº 20241009530

Natal-RN
2024

GUILHERME FÉLIX DE MEDEIROS BRANDT - Nº 20241009450
JOSE LINDENBERG DE ANDRADE - Nº 20241009460
LARISSA SOARES DE SOUZA - Nº 20241028376
PIETRO AUGUSTO DE ALBUQUERQUE LIRA E SILVA - Nº 20241028474
RAFAEL AUGUSTO DE OLIVEIRA GUEDES - Nº 20241009530

SEMINÁRIO SOBRE O COPPELIASIM

Relatório apresentado à disciplina de Sistemas Robóticos Autônomos, correspondente à segunda meta de avaliação do 1º projeto do semestre 2024.2 do curso de Mestrado em Engenharia Elétrica e de Computação da Universidade Federal do Rio Grande do Norte, sob orientação do **Profº Drº Pablo Javier.**

Professor: Drº Pablo Javier Alsina.

Natal-RN
2024

RESUMO

Esse relatório tem como objetivo implementar o controlador cinemático de posição para o robô móvel que leve o robô a uma posição final especificada. Além de obter resultados de simulação (caminho seguido pelo robô, gráficos das variáveis de entrada e saída em função do tempo, entre outros.). O robô utilizado é o Pioneer P3Dx e possui um controlador cinemático de posição. Para realizar as simulações será utilizado o *software* CoppeliaSim.

Palavras-chave: Robô móvel; CoppeliaSim; Pioneer P3Dx; Controlador.

Lista de Figuras

Sumário

1 INTRODUÇÃO

A finalidade do presente trabalho é simular no *software* CoppeliaSim um robô móvel com acionamento diferencial, de maneira a que o mesmo receba os comandos das velocidades de referências para as rodas e retorne a posição e orientação do robô (x, y, θ) em um referencial global. E além do movimento do robô no espaço de trabalho, mostrar os gráficos de velocidades das rodas (entradas) em função do tempo; configuração do robô (x, y, θ) (saídas), em função do tempo; e o gráfico das posições (x(t), y(t)) seguidas pelo robô no plano xy.

1.1 O que é CoppeliaSim?

O simulador de robô CoppeliaSim, com ambiente de desenvolvimento integrado, é baseado em uma arquitetura de controle distribuída, assim, cada objeto/modelo pode ser controlado individualmente por meio de um *script* embutido (com as liguagens LUA ou Python), um *plugin* (em C, C++), um nó ROS ou BlueZero, um cliente API (*Application Programming Interface*) remoto ou uma solução customizada. Isso torna o CoppeliaSim uma ferramenta bastante versátil e ideal para aplicações multi-robôs. Os controladores podem ser escritos em C / C ++, Python, Java, LUA, Matlab ou Octave.

O CoppeliaSim é usado para desenvolvimento rápido de algoritmos, simulações de automação de fábrica, prototipagem e verificação rápida, educação relacionada à robótica, monitoramento remoto, verificação dupla de segurança, como gêmeo digital entre outras atribuições (Coppelia, 2024).

Figura 1: CoppeliaSim

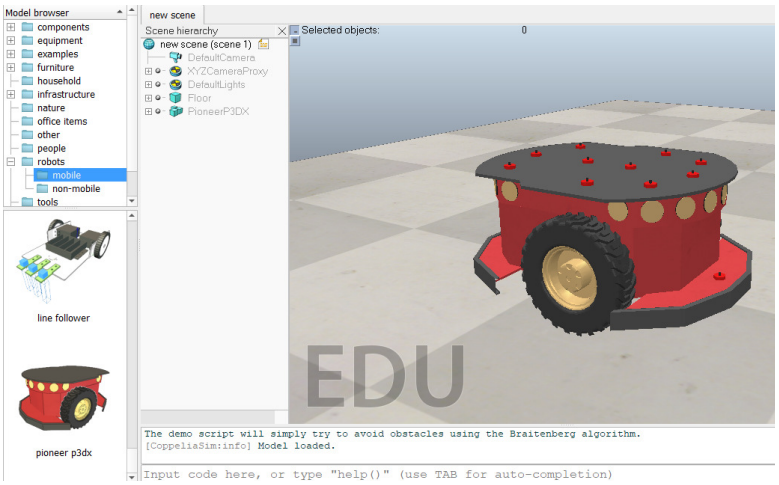
Fonte: Coppelia Robotics

2 DESENVOLVIMENTO

2.1 Como fazer uma simples simulação

O primeiro passo para realizar uma simulação é escolher o robô. Para isso é necessário observar uma caixa aberta no lado esquerdo do Coppelia onde tem o nome "Model Browser". Nesta existem várias pastas e é preciso clicar na que tem escrito "Robots"e, em seguida, no "Mobile". Após a realização desses passos, logo abaixo dessa caixa aparecerão vários tipos de robôs, mas o que é utilizado nesse trabalho é o Pioneer P3Dx. É possível ver essas características na Figura abaixo.

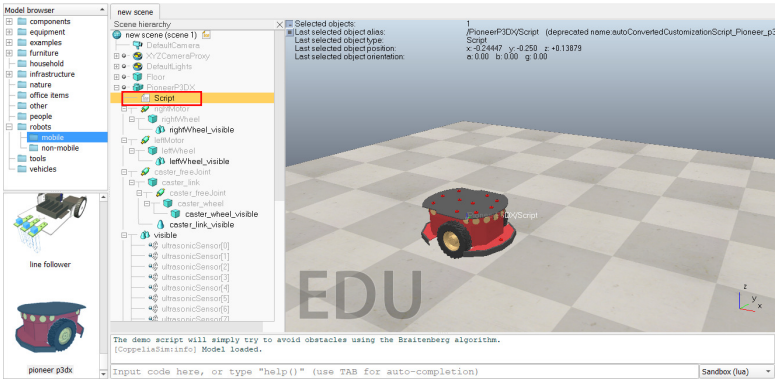
Figura 2: Caixa Model Browser e robô Pioneer P3Dx



Fonte: Elaborado pelos autores

Além da insersão do modelo no *software*, também é necessário inserir o código de atuação do robô utilizado, para isso é necessário abrir a árvore de ferramentas do robô localizada em *Scene hierarchy* e abrir o seu *script* para começar a programação, para o caso trabalhado, a linguagem utilizada foi LUA.

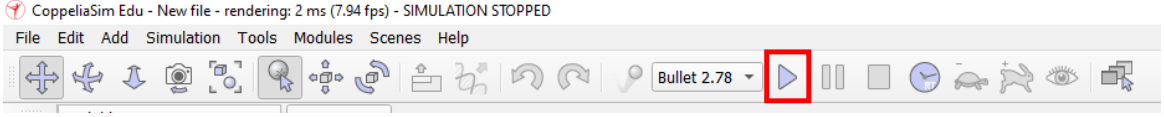
Figura 3: Árvore de ferramentas Pioneer P3Dx



Fonte: Elaborado pelos autores

Com tudo já pronto, para rodar a simulação é necessário apenas apertar a seta, mostrada na Figura 4, que fica na parte central do topo da página do simulador.

Figura 4: Controle de simulação.



Fonte: Elaborado pelos autores

2.2 Controle cinemático

O controle cinemático aplicado em um robô serve para determinar um conjunto de entradas (também conhecidas como as velocidades aplicadas nas rodas) apropriadas para levar o robô de uma determinada posição inicial até uma final (Macharet, 2024).

Como a simulação do Pioneer P3Dx está sendo realizada em um ambiente livre de obstáculos, para fazer o controle cinemático basta especificar um caminho e dividir o mesmo em segmentos bem definidos, esses segmentos podem ser retas ou arcos de circunferência, desde que o raio dessa circunferência seja de um tamanho satisfatório para o robô realizar as manobras.

O código do controle programado para o Pioneer P3Dx está descrito abaixo.

```
pos = sim.getObjectPosition(robot, -1)
theta = sim.getObjectOrientation(robot, -1)[3]

errorX = targetPos[1] - pos[1]
errorY = targetPos[2] - pos[2]
distance = math.sqrt(errorX^2 + errorY^2)

desiredTheta = math.atan2(errorY, errorX)
angleError = desiredTheta - theta

if angleError > math.pi then
    angleError = angleError - 2 * math.pi
elseif angleError < -math.pi then
    angleError = angleError + 2 * math.pi
end

if distance < distanceThreshold then
    sim.setJointTargetVelocity(motorLeft, 0)
    sim.setJointTargetVelocity(motorRight, 0)
    sim.addStatusbarMessage('Rob? chegou ao ponto final e parou.')
    return
end
```

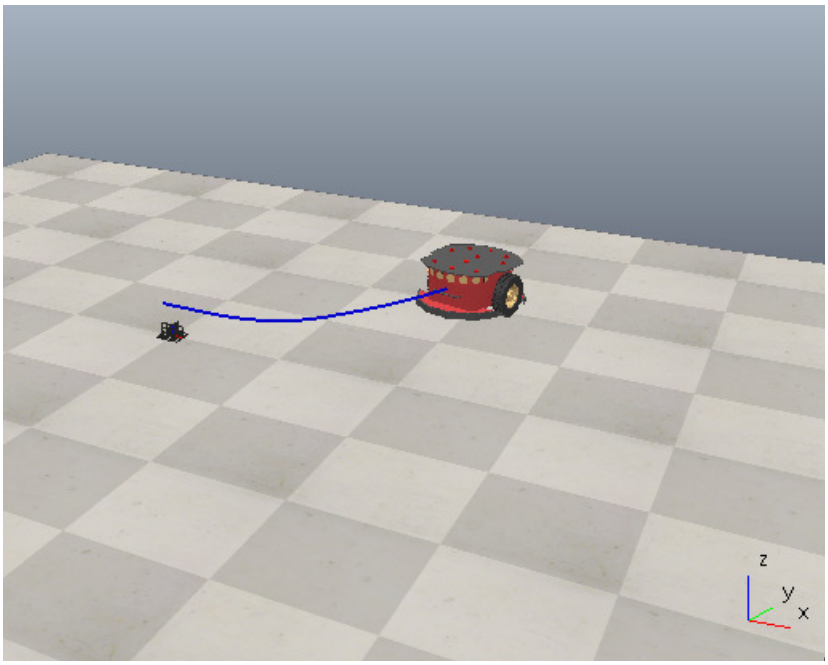
2.3 Resultados gráficos e códigos

Nessa seção está apresentado os resultados gráficos das curvas do robô e os trechos do código responsável pela plotagem dos gráficos.

2.3.1 Caminho seguido pelo robô

Com o código do robô realizado, foi coletado o caminho que o robô seguiu do seu ponto inicial até o ponto, final. No código, foi considerado o robô no centro do ambiente de simulação como ponto inicial e imposta a posição alvo na posição (1.0, 1.0) do plano (x, y), ao final do percurso, o caminho seguido pelo robô está traçado na Figura 5 com a linha azul.

Figura 5: Caminho seguido pelo robô



Fonte: Elaborado pelos autores

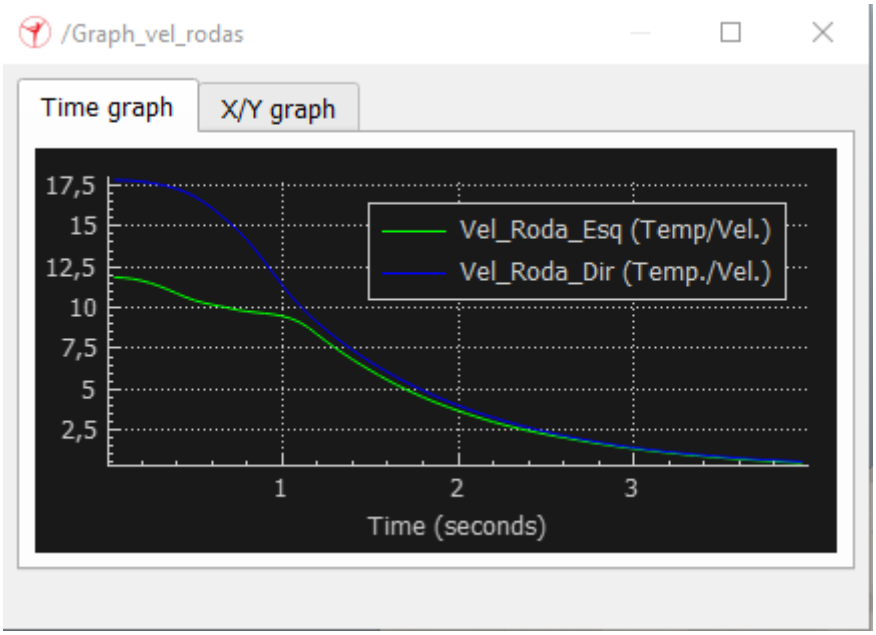
Para gerar essa trajetória foi utilizada a função *sim.addDrawingObject* que tem como função criar objetos de desenho dentro do ambiente gráfico do robô, podendo gerar de retas a outros pontos de interesse, nele é inserido o tipo de objeto desenhado, o tamanho desse objeto, a opção de visualização (se o objeto de desenhado deve ser sempre visível independente da perspectiva da câmera), e valores para determinar a posição, orientação e cor do objeto. O trecho de código utilizado para obter o caminho está abaixo.

```
trajectoryHandle = sim.addDrawingObject(sim.drawing_linestrip, 2, 0, -1, 9999, {0, 0, 1})
```

2.3.2 Gráfico de velocidades das rodas (entradas) em função do tempo

O primeiro gráfico (Figura 6) ilustra as velocidades das rodas do robô em função do tempo, apresentando como cada roda (direita e esquerda) se comporta individualmente, assim como o ângulo. O gráfico apresenta o comportamento do robô ao fazer curvas e mudanças de direção. A roda esquerda (linha verde) e a roda direita (linha azul) variam as suas velocidades para se ajustar à posição e angulação finais. Esse padrão de variação nas velocidades das rodas e no ângulo demonstra o comportamento do robô ao realizar manobras, como curvas e mudanças de direção.

Figura 6: Gráfico 1 - Velocidades das rodas



Fonte: Elaborado pelos autores

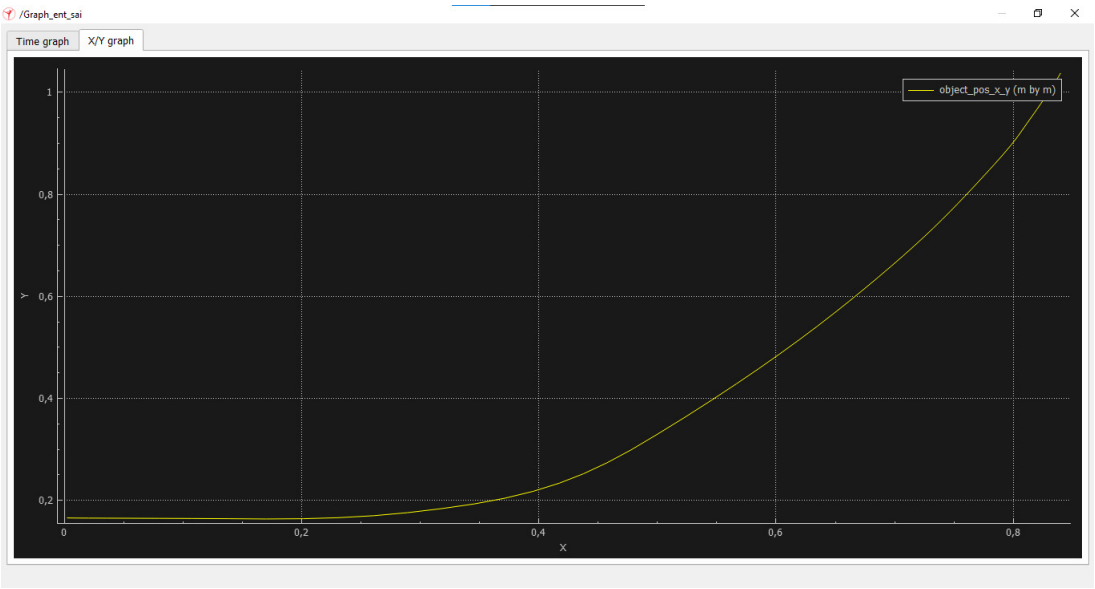
O código utilizado para gerar os dados da Figura 6 está representado abaixo.

```
graphVelRodas = sim.getObject('/Graph_vel_rodas')
left_wheel_graph = sim.addGraphStream(graphVelRodas, 'Vel Roda Esq', 'Temp/Vel.', 0, {0, 1, 0}, 1)
right_wheel_graph = sim.addGraphStream(graphVelRodas, 'Vel Roda Dir', 'Temp./Vel.', 0, {0, 0, 1}, 1)
```

2.3.3 Gráfico de configuração do robô (x, y e θ) (saídas) em função do tempo

O segundo gráfico (Figura 7) foi gerado a partir da coleta de dados da posição x, y e θ do robô em função do tempo. A variação conjunta dos valores de x e y indica que o robô está realizando um movimento curvo. A mudança do ângulo θ reflete as alterações na orientação do robô levando a uma alteração na sua trajetória. As oscilações no gráfico sugerem que essas variações no ângulo estão associadas às mudanças de direção ou sentido ao longo do percurso.

Figura 7: Gráfico 2 - Configuração do robô (x, y, θ)



Fonte: Elaborado pelos autores

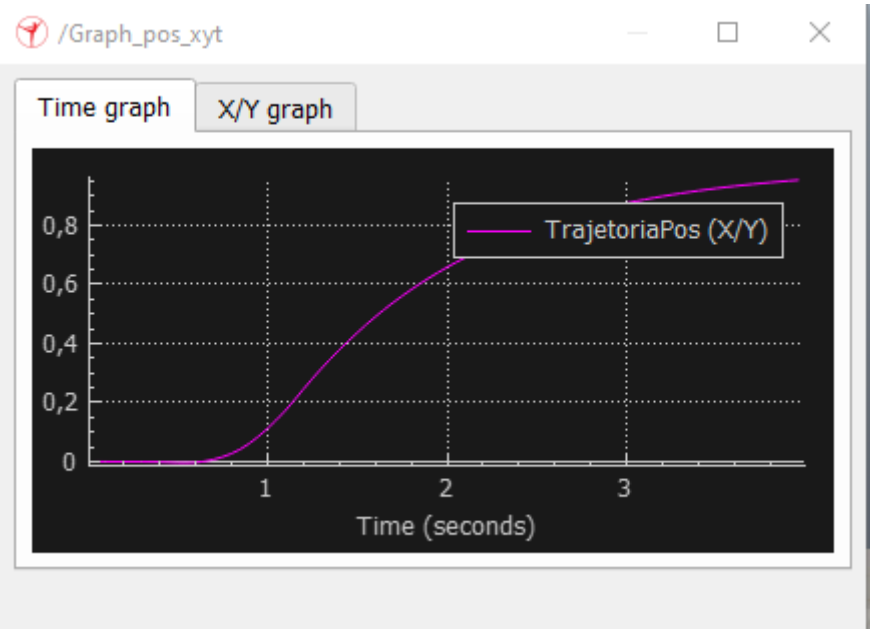
O código utilizado para gerar os dados da Figura 7 está representado abaixo.

```
graphEntSai = sim.getObject('/Graph_ent_sai')
objectPosX = sim.addGraphStream(graphEntSai, 'object pos x', 'm', 1)
objectPosY = sim.addGraphStream(graphEntSai, 'object pos y', 'm', 1)
sim.addGraphCurve(graphEntSai, 'object pos x/y', 2, {objectPosX, objectPosY}, {0, 0}, 'm by m')
```

2.3.4 Gráfico das posições (x(t), y(t)) seguidas pelo robô no plano xy

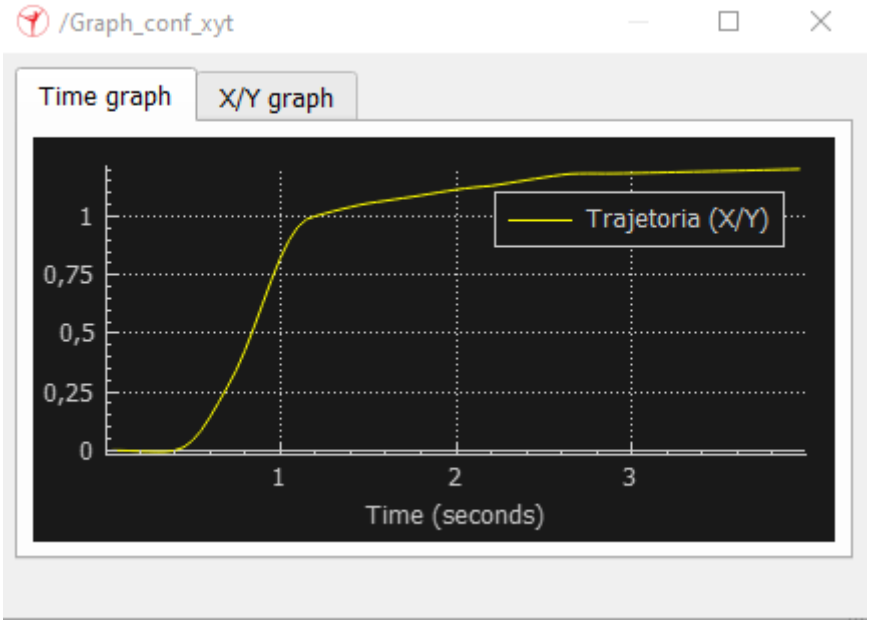
Para o gráfico das posições, foram plotadas duas figuras (8 e 9) em que foi aplicada uma relação de dependência entre os eixos x e y. Nos gráficos, é possível observar que a trajetória realiza uma curva acentuada para se ajustar à posição final.

Figura 8: Gráfico 3 - Configuração (y(t) *versus* x(t))



Fonte: Elaborado pelos autores

Figura 9: Gráfico 4 - Posição (y(t) *versus* x(t))



Fonte: Elaborado pelos autores

O código utilizado para gerar os dados das Figuras 8 e 9 está representado abaixo.

```
graphEntSai = sim.getObject('/Graph_ent_sai')
objectPosX = sim.addGraphStream(graphEntSai, 'object pos x', 'm', 1)
objectPosY = sim.addGraphStream(graphEntSai, 'object pos y', 'm', 1)
sim.addGraphCurve(graphEntSai, 'object pos x/y', 2, {objectPosX, objectPosY}, {0, 0}, 'm by m')
```

3 CONCLUSÃO

Foi possível realizar a implementação do controlador cinemático de posição para o robô móvel no CoppeliaSim Edu de forma eficiente em conduzi-lo à posição final especificada.

Os resultados das simulações apresentaram o comportamento esperado do robô, conforme observado nos gráficos gerados. O gráfico 'Graph_conf_xyt' confirmou que o objetivo foi atingido, respeitando as restrições cinemáticas, os gráficos 'Graph_ent_sai' e 'Graph_vel_rodas' mostraram um ajuste eficaz das velocidades das rodas, resultando em uma trajetória suave e o gráfico 'Graph_pos_xyt' validou a precisão do modelo implementado.

Através deste relatório e vídeo que evidenciaram o sucesso da implementação, oferecendo uma base sólida para futuras melhorias.

Referências

Simulador CoppeliaSim. Coppelia Robotics, 2024. Disponível em: <https://www.coppeliarobotics.com/>. Acesso em 17 de Out. de 2024.

CoppeliaSim User Manual, 2022. Disponível em: <https://manual.coppeliarobotics.com/>. Acesso em 17 de Out. de 2024.

DOUGLAS, G.; MACHARET. Robótica Móvel. [s.l: s.n.]. Disponível em: <https://homepages.dcc.ufmg.br/ doug/cursos/lib/exe/fetch.php?media=cursos:roboticamovel:aula09-controle-cinematico.pdf>. Acesso em: 24 out. 2024.