RobôCIn
EQUIPE DE ROBÓTICA

PS 2025 - Especificação Projeto de Software

No contexto de competições de robótica autônoma, como a **RoboCup**, os robôs devem ser capazes de executar tarefas complexas de maneira autônoma, adaptando-se em tempo real a um ambiente dinâmico. Nessas competições, sistemas de software são fundamentais para implementar soluções eficientes e inteligentes que permitam o robô tomar decisões e agir rapidamente no ambiente de competição.

Para um desempenho competitivo, é necessário que o software possa integrar diversos algoritmos em um sistema robusto e modular a fim de lidar com os desafios existentes no ambiente de competição. Dentre esses desafios, ganham destaque o planejamento de trajetórias para evitar obstáculos e a tomada de decisão para distribuir, de forma eficiente, tarefas entre diferentes agentes. A integração correta desses tipos de algoritmos é fundamental para assegurar que os robôs possam navegar no ambiente, identificar alvos e realizar ações estratégicas com precisão.

1. Descrição do problema

Em competições de robótica autônoma, os robôs são colocados à prova em ambientes altamente dinâmicos, precisando executar uma variedade de tarefas em tempo real. Para realizar uma atuação eficiente dos robôs no ambiente, o software precisa ser capaz de guiá-los, evitando obstáculos enquanto planeja a melhor estratégia para resolver os desafios do ambiente da competição. Ao lidar com vários robôs, o software também precisa ser capaz de coordenar a ação conjunta deles, fazendo com que cada robô execute uma tarefa que irá contribuir para atingir um determinado objetivo. Assim, um desafio que surge e ganha destaque é a atribuição eficiente de tarefas entre os robôs de um time.

Esses desafios destacam a necessidade de dois tipos principais de algoritmos:

- **Planejamento de Trajetória**: Para determinar o melhor caminho até o destino, evitando áreas de risco ou obstruções.
- **Atribuição de tarefas**: Para definir qual agente deve executar cada tarefa de forma eficiente e com base nas condições atuais do ambiente.

Para que os sistemas robóticos funcionem e não se tornem ainda mais complexos do que já são, é imprescindível o bom uso dos fundamentos de orientação a objetos, implementação de códigos legíveis e modularizados, bem como a escolha e utilização de algoritmos que resolvem as particularidades do problema da maneira mais viável possível.

2. Projeto da Seletiva

O objetivo deste projeto é desenvolver e implementar algoritmos eficientes para planejamento de trajetória e atribuição de tarefas, permitindo a cada agente no ambiente ser designado a um objetivo específico e navegar por um cenário com obstáculos até o destino desejado.

O software base para o desenvolvimento está disponibilizado na seção 3. O ambiente tem obstáculos, que podem ser tanto estáticos quanto dinâmicos, e destinos, que serão posicionados de forma aleatória e reposicionados sucessivamente ao serem alcançados. A quantidade de destinos disponíveis simultaneamente no ambiente pode aumentar para mais de um, criando a necessidade de atribuir robôs diferentes a cada destino. O projeto será dividido em 4 fases, em que cada uma terá um nível de dificuldade diferente:

1. Obstáculos estáticos e 10 destinos gerados sucessivamente;
2. Obstáculos dinâmicos, com 10 destinos gerados sucessivamente;
3. Obstáculos dinâmicos, atribuição de robôs para destinos cuja quantidade aumenta de 1 até 4 destinos simultâneos, com 10 gerações de destinos após atingir o máximo;
4. Obstáculos dinâmicos, atribuição de robôs para destinos cuja quantidade aumenta de 1 até 6 destinos simultâneos, com 10 gerações de destinos após atingir o máximo;

As fases 3 e 4 começam com 1 destino, aumentando progressivamente até o máximo de destinos: a cada vez que todos os destinos do ambiente forem alcançados, a quantidade aumenta e, consequentemente, mais robôs deverão ser atribuídos. Após atingir o máximo de destinos simultâneos, o ambiente mantém essa configuração por 10 rodadas adicionais.

Dica 1: *O arquivo `agent.py` possui a decisão de cada agente no campo e as modificações feitas podem ser integradas a partir dele.*
Dica 2: *Vocês **não** devem fazer alterações que modifiquem o comportamento dos obstáculos nem que mudem a dificuldade das fases.*
Dica 3: *Pode ser útil separar o código desenvolvido em arquivos diferentes e apenas usar o necessário ao fazer a integração com o ambiente.*

Formato da Entrega Parcial:

1. Um documento de uma página que descreva o progresso até a entrega parcial, podendo indicar não só o que foi programado, mas também os assuntos que foram estudados e as abordagens definidas;
2. Link para o repositório do projeto no Github.

Na entrega parcial, tente responder às seguintes perguntas:

- Você está tendo dificuldades para a realização desse projeto? Se sim, quais?
- Na sua opinião, quais são os maiores desafios que o ambiente de testes disponibilizado propõe?
- Quais abordagens foram estudadas para tratar os problemas mencionados no projeto?
- Alguma abordagem já foi definida? Se sim, qual e por quê?

Formato da Entrega Final:

1. Uma apresentação em *PowerPoint/slides* dos estudos, dos resultados obtidos e dificuldades enfrentadas durante o projeto, que será apresentada para a equipe em até 15 minutos;
2. Link para o repositório do projeto no Github;
3. Vídeo do projeto rodando e funcionando no ambiente simulado, que pode estar integrado nos slides da apresentação.

OBS: É importante ressaltar que **não será eliminatório deixar de concluir todas as fases.** Isso porque não será avaliado somente o resultado do projeto, mas também o empenho, a organização de código e os fundamentos da programação, mesmo que o resultado final não seja o esperado.

Na sua apresentação, tente responder às seguintes perguntas:

- Na sua opinião, e após sua pesquisa durante o desenvolvimento do projeto, quais são os maiores desafios na hora de fazer este projeto?
- O que você mais gostou de ter sido capaz de realizar durante o desenvolvimento do projeto?
- Você acredita que aprendeu algo novo durante o desenvolvimento do projeto? Se sim, o quê? Se não, por quê?
- Quais abordagens você escolheu para tratar os 2 problemas mencionados no projeto?
- Quais algoritmos foram usados e por que eles foram escolhidos? Caso tenha pesquisado sobre várias opções de algoritmos para cada problema, pode comentar sobre as vantagens dos que foram escolhidos.
- Existem pontos em que sua abordagem possa ser melhorada ou otimizada?

- Como você estruturou e organizou o desenvolvimento do código?

3. Links Úteis

- Visual Studio Code - aqui
- Software base para o desenvolvimento - aqui
- Tutorial de Git e Github - aqui

Alguma dúvida? Entra em contato com a gente! 🫡