Análise das Implicações da adição de Planejamento de Trajetórias no Controle de Robôs Omnidirecionais Aplicado ao Futebol de Robôs

Riei Joaquim Matos Rodrigues (rjmr@cin.ufpe.br)

Universidade Federal de Pernambuco
secgrad@cin.ufpe.br
www.cin.ufpe.br/~graduacao

Recife
2024

Riei Joaquim Matos Rodrigues (rjmr@cin.ufpe.br)

Análise das Implicações da adição de Planejamento de Trajetórias no Controle de Robôs Omnidirecionais Aplicado ao Futebol de Robôs

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Área de Concentração: *Controle de Robótica Móvel*
Orientador: *Edna Natividade da Silva Barros (ensb@cin.ufpe.br)*
Co-Orientador: *Hansenclever de França Bassani (hfb@cin.ufpe.br)*

Recife
2024

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Rodrigues, Riei Joaquim Matos.
Análise das Implicações da adição de Planejamento de Trajetórias no Controle de Robôs Omnidirecionais Aplicado ao Futebol de Robôs / Riei Joaquim Matos Rodrigues. - Recife, 2024.
61

Orientador(a): Edna Natividade da Silva Barros
Cooorientador(a): Hansenclever de França Bassani
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Engenharia da Computação - Bacharelado, 2024.

1. Controle. 2. Robótica Móvel. 3. Planejamento de Caminhos. 4. Planejamento de Trajetórias. 5. Trajetórias Bang-Bang. I. Barros, Edna Natividade da Silva. (Orientação). II. Bassani, Hansenclever de França . (Coorientação). IV. Título.

000 CDD (22.ed.)

RIEI JOAQUIM MATOS RODRIGUES

Análise das Implicações da adição de Planejamento de Trajetórias no Controle de Robôs Omnidirecionais Aplicado ao Futebol de Robôs

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em 2023.2 da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de bacharel em Engenharia da Computação.

Aprovado em: 19/03/2024

BANCA EXAMINADORA

_____

Profa. Dra. Edna Natividade da Silva Barros (Orientadora)

Universidade Federal de Pernambuco

_____

Prof. Dr. Adiel Teixeira de Almeida Filho (Examinador Interno)

Universidade Federal de Pernambuco

AGRADECIMENTOS

Primeiramente, quero agradecer a meus pais, Paulo e Edna, meus avós Joaquim e Romilde, que sempre renunciaram a muitas coisas para proverem suporte aos meus estudos e sonhos, que mesmo com medo do que poderia acontecer possibilitaram um jovem do interior da Bahia desbravar o mundo e contra as probabilidades ir mais longe do que jamais sonharam que seria possível. Agradeço muito a todos os meus familiares, em especial minha tia Edite e minha prima Danielle, amigos e conhecidos que tenho na Bahia, que mesmo de longe torcem muito por cada passo meu, comemorando cada conquista.

Quero agradecer a todos os professores que me ajudaram nesta jornada até aqui, desde do ensino médio em especial a Ivan, e principalmente aos professores do Centro de Informática (CIn) que me ensinaram muito e me fizeram sentir parte do CIn, em especial Edna Barros, Hansenclever Bassani e Alexandre Mota que me acolheram nos projetos RobôCIn, CIn/Motorola e CIn/Softex, nos quais ao longo dos anos pude evoluir e amadurecer muito tanto do ponto de vista profissional quanto do pessoal.

Além disso, gostaria de agradecer aos amigos que fiz em Recife ao longo dos anos por conta do pensionato, da graduação e do trabalho em especial a Artur Santos, Marcelo Fernandes, Ramiro Neto, Victor Miguel, Victor Hugo, Victor Ximenes, Pedro Nogueira, Zilde Neto, Lucas Ambrósio, Fernando Neto, Daniel Perrazo, José Carlos, Andresa Silva, Jefferson Norberto, Jairton Falcão, Felipe Augusto e Júlia Melo.

Por último, gostaria de agradecer ao RobôCIn e tudo que ele me proporcionou durante esses anos, onde pude aprender, evoluir e coordenar conquistar incríveis, além de momentos inesquecíveis e as amizades construídas para além do ambiente do projeto, em especial gostaria de agradecer a Lucas Cavalcanti, Roberto Fernandes, Victor sabino, Felipe Martins, Cecília Virgínia, José Victor, Matheus Teotônio, Ryan Morais e Thiago Araújo por terem me ajudado ao longo dessa jornada e terem construído junto comigo essa história.

"Run"
"Live to fly"
"Fly to live, do or die"
"Fly to live, aces high!"
— Aces High – Iron Maiden

RESUMO

A navegação de robôs móveis de forma robusta pelo espaço é um problema computacional amplamente estudado, sendo fundamental para aplicações de robótica móvel autônoma. O processo de navegação é iniciado com aplicação de um planejamento de caminhos para traçar uma rota factível e sem colisões até o destino, e é finalizado pela aplicação do controle que define diretamente as ações tomadas pelo robô, como, por exemplo, as velocidades das rodas. O planejamento de trajetórias surge entre esses dois pontos do processo, para suprir lacunas relacionadas à dinâmica da movimentação do robô. Ao adicionar esse planejamento de trajetória, surge a problemática de como avaliar os ganhos de desempenho obtidos em relação às estratégias sem esse módulo e sem as melhorias de estimação de estado propostas neste trabalho para conseguir extrair mais desempenho. A partir da execução da metologia de testes propostas em um ambiente dinâmico de obstáculos e de objetivos, um jogo de futebol de robôs, que exploram várias capacidades das estratégias, a introdução das trajetórias Bang-Bang em conjunto com o preditor de Smith foi avaliada. Com isso, foi possível determinar as vantagens de cada abordagem, com os quais é possível constatar que ao aplicar as Trajetórias Bang-Bang com o estado atual sincronizado, obteve-se um ganho de 25% da velocidade média durante a movimentação e uma redução de 15% do tempo necessário em relação à estratégia com o grafo de visibilidade.

Palavras-chave: Controle. Robótica Móvel. Planejamento de Caminhos. Planejamento de Trajetórias. Trajetórias Bang-Bang.

ABSTRACT

Navigating mobile robots robustly through space is a widely studied computational problem, being fundamental for autonomous mobile robotics applications. The navigation process begins with the application of path planning to trace a feasible and collision-free route to the destination, and is completed by the application of control that directly defines the actions taken by the robot, such as, for example, wheel speeds. Trajectory planning appears between these two points in the process, to fill gaps related to the dynamics of the robot's movement. When adding this trajectory planning, the problem arises of how to evaluate the performance gains obtained in relation to strategies without this module and without the state estimation improvements proposed in this work to extract more performance. From the execution of the proposed testing methodology in a dynamic environment of obstacles and objectives, a robot soccer game, which explore various capabilities of the strategies, the introduction of Bang-Bang trajectories in conjunction with the Smith predictor was evaluated. With this, it was possible to determine the advantages of each approach, with which it is possible to verify that when applying the Bang-Bang Trajectories with the current synchronized state, a gain of 25% of the average speed during movement was obtained and a reduction 15% of the time needed in relation to the strategy with the visibility graph.

Keywords: Control. Mobile Robotics. Path Planning. Trajectory Planning. Bang-Bang Trajectories.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

FIFA	Federação Internacional de Futebol
LARC	Latin American Robotics Competition
RMSE	Erro Quadrático Médio
SSL	Small Size League
TBB	Trajetórias Bang-Bang
TBBES	Trajetórias Bang-Bang de Estado Sincronizado
VG	Grafo de Visibilidade

LISTA DE ALGORITMOS

SUMÁRIO

1

INTRODUÇÃO

O uso de robôs móveis autônomos dentro do dia a dia das pessoas vem se tornando mais comuns nos últimos tempos, principalmente devido às relevantes aplicações direcionadas para um público mais amplo como aspiradores de pó, entregas feitas por drones e até carros autônomos. Todas essas aplicações compartilham a necessidade fundamental de navegação pelo ambiente de forma autônoma, respondendo dinamicamente a mudanças de forma eficiente para cumprir sua tarefa de navegação, de ir de um ponto a outro no espaço.

Diversos estudos têm sido realizados por pesquisadores sobre as aplicações e os desafios envolvendo a robótica móvel. Alatise et al [3] discutiram os desafios envolvidos em aplicações de robótica móvel autônoma, considerando os principais módulos na ampla maioria dos sistemas. Cada um deles possui uma ampla área do conhecimento associada e em expansão, dessa forma, a conclusão da autora aponta a robótica autônoma como vários desafios inerentes ainda em aberto, dado o amplo de potencial idealizado em comparação com a perspectiva explorada atualmente.

Como desafios, temos a capacidade de locomoção do robô, aspectos de projeto que não dependem somente de qual ambiente ele vai navegar, mas também da manobrabilidade e da controlabilidade possibilitada, além da flexibilidade para conseguir superar variações de condições do terreno, onde essas escolhas impactam diretamente a eficiência do sistema.

Outro desafio é a capacidade de percepção do meio a sua volta, com um subconjunto de sensores, dentre os vários possíveis, e uma série de algoritmos de consolidação de informações. O robô deve conseguir se situar no ambiente, e, de forma eficiente, definir sua posição relativa aos obstáculos. Se um robô não tiver um sistema robusto de percepção, ele pode não conseguir reagir às mudanças do mundo externo, prejudicando o desvio de obstáculos. Esta capacidade é, atualmente, um grande desafio para áreas mais críticas como os carros autônomos, que buscam dividir espaço com motoristas humanos e pedestres nas ruas das cidades, como discutido por Hussain et al. [7].

Por último, temos a capacidade de navegação, diretamente ligada a como o robô atua no meio. Para isso, é fundamental saber em que ponto da tarefa de navegação o robô se encontra, qual o próximo destino e como o robô vai fazer para chegar lá. Essa capacidade possui uma forte relação com a capacidade de percepção, pois a resposta fornecido por ela vai impactar na ação tomada pelo controle dos atuadores, que irão gerar o movimento.

A RoboCup é uma competição mundial de diversas modalidades de robôs autônomos existente desde 1997, cujo objetivo é desenvolver uma equipe de robôs humanoides capazes de derrotar a mais recente equipe campeã da Copa do Mundo da Federação Internacional de Futebol (FIFA) [8]. Essa competição cria um ambiente altamente dinâmico onde os robôs devem jogar futebol de forma autônoma contra outro time, sendo um contexto perfeito para o desenvolvimento de tecnologias interdisciplinares, pois possui obstáculos e posições alvo que se movem a cada segundo. Essa dinâmica requer detecção precisa da situação de campo, combinada com algoritmos de decisão rápidos e principalmente uma navegação que corresponda com uma velocidade e precisão elevadas.

Neste trabalho serão realizadas análises de técnicas de navegação em um ambiente que oferece uma grande variação de cenários, serão utilizados os robôs de Small Size League (SSL) para a validação da navegação durante um jogo de futebol, no qual os outros robôs se movem pelo campo representando obstáculos dinâmicos e as áreas dos goleiros, representando obstáculos estáticos. Por regra, a bola não pode ser empurrada por mais de um metro e precisa ser chutada, alcançando uma velocidade de até 6.5 m/s, esta regra cria uma mudança de objetivos e contextos repentinamente. Além disso, o fato dos robôs serem omnidirecionais força a exploração de sua manobrabilidade.

1.1 OBJETIVOS

A introdução do planejamento de trajetórias promete uma melhoria da conformidade entre o planejamento e a execução de uma movimentação, possibilitando o aumento do desempenho do sistema de navegação. Entretanto, avaliar esses ganhos de forma sistemática e entender as vantagens e desvantagens das implementações e otimizações de uma estratégia de navegação não é uma tarefa simples ou bem estabelecida para robôs moveis, por dependerem muito do contexto em que a robótica é aplicada.

O objetivo geral do trabalho é estudar e analisar as possíveis melhorias de desempenho no controle e navegação decorrentes da introdução de um planejamento de trajetórias em conjunto com um planejamento de caminhos para a navegação de robôs omnidirecionais, a parir de uma metologia para avaliação proposta.

A equipe de SSL do Centro de Informática (CIn), do RobôCIn, anteriormente possuía um sistema de navegação baseado apenas na execução de caminhos, com baixo acoplamento cinemático ao robô. Neste cenário, basicamente, se tinha uma lista de pontos no espaço obtidos por um planejamento de caminho, no qual o controle apenas buscava alcançar o próximo ponto na sequência até chegar ao objetivo final.

Recentemente, no RobôCIn, o desempenho do sistema de navegação foi visto como um gargalo para o desempenho atual da equipe, com isso, esse trabalho objetiva a implementação de uma estratégia de navegação que possui um planejamento de trajetórias, isto é, a descrição cinemática do robô ao longo do tempo, que acopla fortemente o planejamento ao contexto do

robô. Isso se reflete em um sistema de controle voltado para a aplicação de trajetórias planejadas ao robô, resultando em um potencial ganho de eficiência para a navegação. No entanto, para garantir que esta estratégia representa uma melhoria efetiva no sistema completo, uma análise de desempenho resultante se faz necessária.

Neste trabalho será realizada uma análise comparativa entre as estratégias de navegação implementadas, considerando uma série de cenários dinâmicos e realistas preestabelecidos, buscando entender e metrificar as vantagens e desvantagens obtidas, além de buscar aprofundar os fundamentos teóricos envolvidos e formalizar um sistema de testes e métricas analisados para validar uma estratégia de navegação.

Serão comparadas três estratégias de navegação, a primeira é a Grafo de Visibilidade (VG) que é uma técnica de navegação ponto a ponto, com os pontos vindos do planejador de caminhos, a segunda é o algoritmo de Trajetórias Bang-Bang (TBB) que aplica o planejamento de trajetórias à navegação diretamente e por fim a estratégia Trajetórias Bang-Bang de Estado Sincronizado (TBBES), que aplica o planejamento de trajetória em conjunto com outros módulos de predição e sincronia. A partir dessa análise, será possível discutir limitações e diferenças, bem como mensurar as reais vantagens e desvantagens das estratégias avaliadas.

Esse trabalho é dividido nas seguintes partes: O Capítulo 2 apresenta o referencial teórico, utilizado para compreender o funcionamento das técnicas abordadas neste trabalho, além da descrição de trabalhos relacionados à análise de desempenho de controle. No Capítulo 3 são apresentadas, em detalhes, as estratégias que serão analisadas e quais otimizações por módulos de predição e sincronia foram adotadas. Já no Capítulo 4 é apresentada a metodologia para a criação e execução dos testes para cada estratégia, além da justificativa dos cenários utilizados, assim como a explicação de quais métricas serão avaliadas durante esses testes. O Capítulo 5 apresenta os resultados obtidos com a execução dos testes em cada estratégia e suas respectivas análises. Por fim, o Capítulo 6 apresenta algumas conclusões obtidas, considerações finais e possíveis trabalhos futuros.

2

REFERENCIAL TEÓRICO

A navegação é a habilidade que possibilita robôs móveis se moverem de um ponto para outro no espaço. O desenvolvimento de estratégias para fazer os robôs desempenharem essa atividade representa uma grande área da robótica. Neste capítulo, são apresentados os conceitos e referenciais teóricos necessários para o entendimento das estratégias de navegação apresentadas nos próximos capítulos desse trabalho, além de servirem de referência para a análise comparativa entre as estratégias.

A navegação de um robô acontece em uma sequência de passos, feita em um ciclo de execução ao longo do tempo. Para fazer isso de forma autônoma, além de conseguir atuar sobre o meio, é necessário que o sistema robótico seja capaz de interpretar o mundo à sua volta, seus obstáculos e definir como navegar pelo ambiente até seu destino.

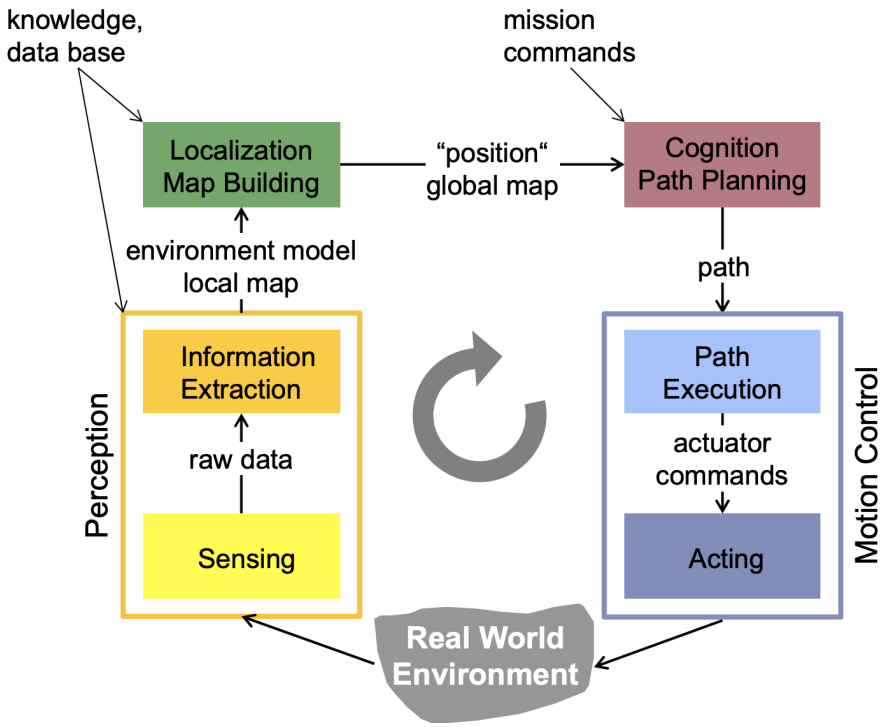


Figura 1: Esquema de controle de um robô móvel. Fonte: Siegwart et al. [15].

Segundo a definição de Siegwart et al. [15] mostrada na Figura 1, a navegação autônoma se dá em quatro etapas: O primeiro passo é a percepção, que baseada nas informações adquiridas por sensores, como encoders e câmeras, levantam dados sobre o atual contexto do mundo. Em seguida vem a etapa de localização, que, a partir desses dados, vai processar qual o estado atual do robô em relação a si (ângulo, velocidade, posição) e em relação aos demais objetos no mundo, os obstáculos, fechando a parte sobre o entendimento do mundo.

As duas etapas restantes dizem respeito sobre a atuação do robô no mundo para executar uma movimentação. Na etapa da cognição, ele planeja qual será sua ação no mundo, qual caminho será tomado até o objetivo, dado o contexto atual do robô no mundo. E por fim, na etapa de controle de movimento, o robô define qual vai ser o comando enviado para ser executado pelos atuadores.

Estratégias de navegação dizem respeito principalmente a como o robô vai planejar e atuar no mundo para concretizar uma tarefa de movimentação. Desse modo, este trabalho foca em definir e discutir as etapas de cognição e controle de movimentação no contexto de futebol de robôs.

Quanto à parte de percepção, ela será uma constante para todas as estratégias, sendo realizada em um sistema centralizado fora do campo, onde os robôs recebem os comandos de movimentação a partir de um módulo central do sistema, que possui visão global sobre o ambiente e poder computacional suficiente para planejar de forma robusta as ações a serem tomadas pelo robô. Como mostrado na Figura 2, nesse sistema, os robôs se movem no plano do campo junto a bola. Alguns metros acima do campo, uma ou a soma de múltiplas câmeras visualizam o campo por completo. Com isso, baseados na localização dos padrões de cores na imagem, as posições dos objetos são calculadas e enviadas para os módulos de controle, através do SSL-Vision [21].

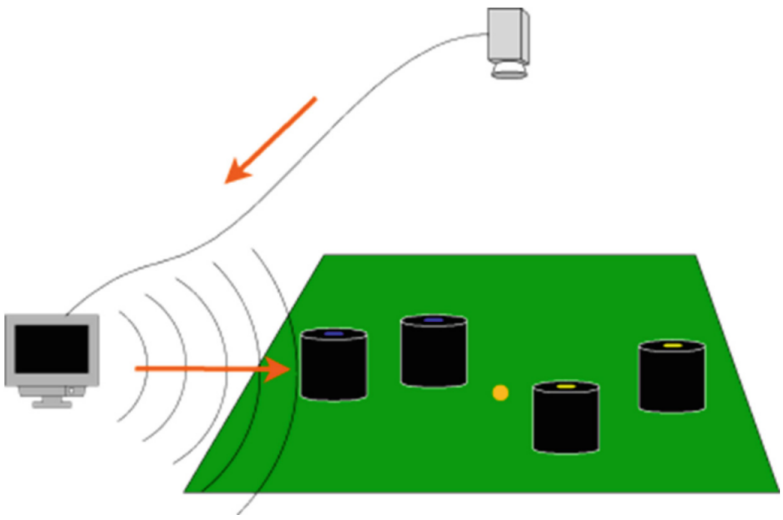


Figura 2: Estrutura geral de percepção do sistema autônomo de competição da SSL. Fonte: Weitzenfeld[18].

Por se basear em detecção de imagens, com processamento de cores e padrões para

o rastreamento dos objetos em cena, as informações do mundo são um tanto ruidosas, além de possuírem um natural atraso entre a captura da imagem e o envio da informação, gerando instabilidade e incerteza. Para minimizar isso, a maioria das equipes de SSL usam estratégias para melhorar a confiabilidade das informações, processando as informações de visão do mundo com filtros, como o de Kalman[20], que leva em conta a sequência atual de pacotes de visão e não só o último, assim, melhorando a consistência dos dados, porém retardando a percepção de mudanças no mundo. Além disso, ainda temos a limitação da frequência de atualização possível com câmeras RGB, que ficam no entorno de 60hz, gerando uma janela de aproximadamente 0.016 segundos para obter uma atualização do mundo, a cada ciclo.

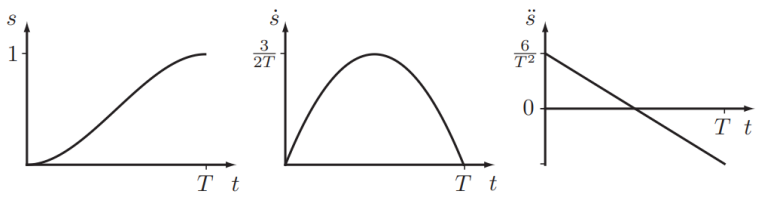
2.1 TRAJETÓRIAS, CAMINHOS E CONTROLE

Conceitualmente, trajetórias e caminhos são objetos de naturezas diferentes. Para a navegação, um caminho considera apenas a perspectiva geométrica, olhando os pontos no espaço, já uma trajetória está associada diretamente ao robô que vai executar aquela movimentação, pois, analisa a perspectiva dinâmica da cinemática do comportamento do robô se movendo ao longo do tempo. Segundo definição de Lynch et al. [9], trajetória é a combinação de um caminho com uma lista de posições geométricas, com um dimensionamento temporal que respeite as capacidades de movimentação do robô, sendo a resolução de tempo entre os pontos, valor que possibilite uma navegação suave e fluida.

Como demonstrado por Lynch et al. [9], a geração de trajetórias a partir de um caminho geométrico $s(t)$ depende de uma relação de derivações e de uma aproximação polinomial. Derivando a posição obtemos velocidade $s(\dot{t})$, derivando novamente obtemos a aceleração $s(\ddot{t})$. A relação gráfica mostrada na Figura 3(a), mostra como essas relações podem ser aplicadas a um polinômio de terceira ordem, que ao ser resolvido em função do tempo, resulta nas desigualdades com os limites da capacidade de atuação do robô, a partir das quais podemos obter o estado (posição, velocidade, aceleração) em função do tempo de forma confiável.

Segundo o trabalho de Lynch et al. [9], ao introduzir as constantes das capacidades do robô, obtemos um controle de aceleração *on/off* dividido em três segmentos, aceleração máxima, velocidade máxima com aceleração zero que pode não ser atingida e desaceleração máxima com aceleração negativa, ao desenhar a velocidade ao longo do tempo forma-se um trapézio que também nomeia a abordagem, como mostrado na Figura 3(b). A movimentação em velocidade possui a desvantagem das descontinuidades e alguns impulsos abruptos sobre o sistema com o chaveamento de zero aceleração para a máxima. Para minimizar as descontinuidades, a abordagem de contorno possível, apresentada pelo Lynch et al. [9], denominada S-Curves quebra a aceleração em mais estágios trazendo mais suavidade a elas.

Idealmente, o planejamento de trajetória seria algo realizado depois do caminho e antes do cálculo da atuação concreta do robô. Entretanto, por simplicidade ou limitações de implementação esse módulo acaba sendo ignorado, criando grandes limitações no controle



(a) Gráficos de s(t), $s(\dot{t})$ e $s(\ddot{t})$ para uma escala de tempo polinomial de terceira ordem.

(b) Gráficos de $s(\dot{t})$ para um perfil de movimento trapezoidal.

Figura 3: Gráficos das relações de derivações em função do tempo e o perfil de velocidade trapezoidal. Fonte: Lynch et al. [9]

quando o mesmo for gerado para a atuação no robô.

Como definido por Gasparetto et al.[6], idealmente uma tarefa de navegação é concretizada em três etapas a partir da sua definição abstrata:

1. Planejamento de caminho: Etapa responsável por viabilizar um caminho possível da posição atual do robô até o objetivo considerando o estado atual do mundo e montando a representação dos obstáculos. Considerando as capacidades de movimentação do robô, essa etapa ajuda principalmente a entender a interferência de obstáculos dinâmicos pelo caminho. Essa fase recebe um objetivo geométrico com a posição destino e uma configuração (regras de negócio relacionadas exclusivamente à busca, exemplo o que deve ser considerado como obstáculo). As dimensões do robô são utilizadas para verificar colisões, mas o robô é tratado como um ponto. Por fim, a saída desse processamento é uma lista de pontos intermediários que o robô deve passar sequencialmente para chegar ao destino.

2. Planejamento de trajetória: Etapa é responsável por computar as representações da interação da dinâmica do robô com os obstáculos e da atuação do robô no mundo ao longo do tempo, utilizando as informações do estado atual do robô (posição, velocidade, aceleração) e suas capacidades de movimentação (acelerações e velocidades, mínimas e máximas). Esse estágio recebe a lista de pontos geométricos computada pela etapa anterior e uma configuração (regras de negócio relacionadas ao comportamento da atuação do robô, exemplo velocidade mínima e máxima, vetor de velocidade objetivo no final, tempo requisitado para a movimentação ser completada). Por fim, a saída desse processamento é uma lista de estados de movimentação do robô ao longo do tempo que conectam o estado atual ao estado final desejado, respeitando suas restrições e capacidades. Para cada ponto no tempo temos a tupla que inclui Posição, vetor de velocidade, vetor de aceleração, bem como o tempo até ponto destino.

3. Controle: Etapa é responsável por processar todo o planejamento anterior resultando

uma atuação efetiva para o robô, a partir do estado atual do robô (posição, velocidade, aceleração) e suas capacidades de movimentação (acelerações e velocidades, mínimas e máximas). Com isso, a diferença entre o estado planejado e o atual é calculada, e a partir desse erro, o comando para robô é criado buscando sincronizar o estado com o planejamento a cada ciclo.

2.1.1 Planejamento de Movimento

Como explicado por Lynch et al. [9], planejar um movimento consiste em encontrar uma sequência de estados que conectem o estado inicial ao estado final evitando os obstáculos e satisfazendo as restrições de limites de velocidade ou aceleração. As configurações de estados possíveis para o robô estão representadas no espaço de configuração ou espaço-C, como por exemplo, a configuração de um braço robótico com n articulações pode ser representada como uma lista de n posições articulares q=(θ1, θ2, ..., θ_n).

Nem sempre algumas definições são claras, como explica Lynch et al. [9], planejamento de caminho e de movimento são coisas diferentes, o problema de planejamento de caminho é um subproblema do problema geral de planejamento de movimento.

Planejar caminhos é um problema puramente geométrico de encontrar um caminho livre de colisão de uma configuração inicial para uma configuração final, sem preocupações com a dinâmica, a duração do movimento ou restrições no movimento, ou nas entradas de controle, pois, é assumido que o caminho retornado pode ser dimensionado no tempo em um ponto futuro para criar uma trajetória viável.

Ter um planejador de movimento eficiente parte da premissa de uma representação fiel do mundo, das configurações possíveis do robô juntamente com sua dinâmica de movimentação. Além disso, é assumido um sistema de controle capaz de fazer o robô seguir com uma certa fidelidade do planejamento na execução. Para responder a possíveis mudanças no mundo ou erros de execução, o planejamento de movimento pode ser utilizado em uma abordagem online ao longo da execução.

Para planejar movimentos, vários métodos podem ser utilizados e os principais foram discutidos por Lynch at el. [9]. Para este trabalho são importantes explicar duas abordagens:

1. Métodos completos: estes métodos criam uma representação geométrica exata da topologia de obstáculos no mundo garantindo a completude de localização de um movimento possível se existir. Entretanto, exceto para problemas simples ou de baixo grau de liberdade, essas representações são computacionalmente proibitivas dada a complexidade do grafo gerado.

2. Métodos de amostragem: métodos baseados em amostragem geométrica em conjunto com uma função avaliadora para saber se o ponto pertence ao espaço-C a partir do estado atual. Neste método, os caminhos são agrupados normalmente em uma árvore de

possíveis caminhos utilizando uma função de proximidade entre estados e coerência na sequência das quais fazem parte. Estes métodos, tendem a ser probabilisticamente abrangentes na exploração e podem até resolver problemas de planejamento de movimento de alto grau de liberdade. As soluções tendem a ser satisfatórias, não ótimas, e pode ser difícil caracterizar a complexidade computacional.

2.1.2 Controle

O módulo de controle, como descrito por Lynch at el. [9], tem como função determinar a baixo nível a ação do controlador do robô, assim convertendo a especificação da tarefa em forças ou torques nos atuadores. As estratégias de controle que alcançam os comportamentos descritos acima são conhecidas como controle de movimento, ou controle de força, a abordagem aplicada depende do contexto da aplicação e do ambiente, se o robô está em contato aplicando força, ou se está se movendo no espaço livre.

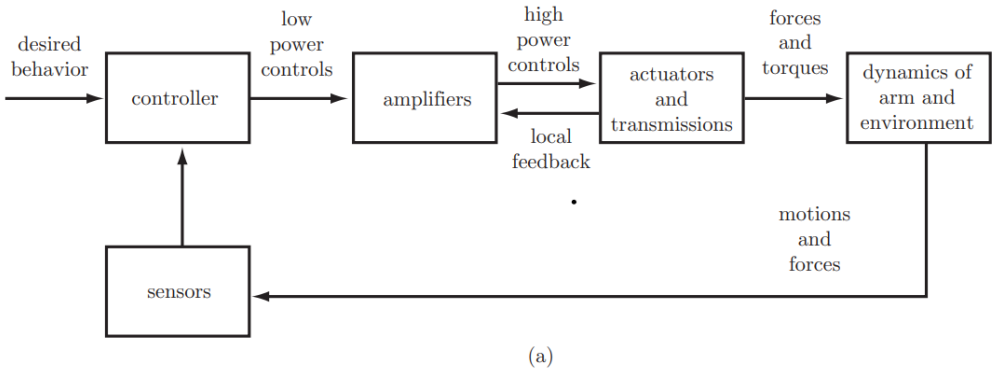


Figura 4: Sistema tipico de controle de robôs. Fonte: Lynch et al.[9].

A Figura 4, apresenta um diagrama clássico de um fluxo de controle, no qual os atuadores recebem um comando e ao aplicar este comando o robô começa a agir no mundo. Com o *feedback* capturado sobre os efeitos dessa ação e das anteriores, o bloco chamado de *Controller* ajusta os comandos de baixo nível para os atuadores a fim de garantir que o resultado produzido seja condizente com o esperado, utilizando técnicas como os controladores PID, que buscam minimizar erros para um ponto de referência.

2.2 GRAFO DE VISIBILIDADE

Um dos métodos completos, descrito na seção anterior, foi a representação do espaço-C livre em uma malha de grafo. Esta estratégia possui um custo computacional proibitiva, mas com otimizações para o contexto de navegação em um espaço 2D são consideradas estratégias clássicas de planejamento de caminhos para robôs móveis por sua relativa qualidade e simplici-

dade de implementação, bem como pelo fato dos dados de saídas em aplicações mais simples, o processo pode ser visto na Figura 5.

A técnica, segundo define Siegwart et al. [15], consiste em englobar os obstáculos em polígonos, adicionar como arestas o ponto de partida e o destino e por fim conectar todos os vértices visíveis com arestas com o peso associado ao custo delas. Com isso, constrói-se uma representação topológica do mundo conectando a posição atual ao destino se for possível.

Tendo o grafo montado, basta executar um algoritmo de menor custo de um caminho em grafo como o algoritmo de Dijkstra. Para limitar os custos dessa busca, somente arestas visíveis são conectadas no VG, mas mesmo assim, se os polígonos dos obstáculos forem complexos, teremos uma busca proibitiva em tempo e uso de memória. O caminho resultante vai contornar os obstáculos pelos vértices criando o caminho mínimo possível, o euclidiano mais curto, mas também bastante suscetível a colisões e invasões, pois, o caminho é feito contornando os obstáculos pelos vértices que forma os polígonos dos obstáculos. Além disso, o caminho gerado, pois, dificuldades que são consequências do uso de polígonos simples, a mudança abrupta de direção e velocidade presentes, que geram instabilidade e perda de tempo, conforme apresentado por Balkon et al.[4].

Para uso em ambientes mais dinâmicos, a limitação de usar polígonos simples cria a dificuldade de representar obstáculos de formas variadas. Os obstáculos são representados como uma lista de pontos conectados dentro do grafo, estáticos, dificultando a avaliação de colisões ao longo do tempo, bem como sobreposição de caminhos. Além disso, como precisamos expandir os obstáculos conforme as dimensões do robô, passamos a ter sobreposição de obstáculos se eles estão próximos, isto exige um pós-processamento do grafo antes de serem feitas as buscas para unificar os obstáculos.

2.3 TRAJETÓRIAS BANG-BANG

Na Seção 2.1, foi discutido o processo de criação de trajetórias a partir de uma especificação geométrica, produzindo um perfil de velocidade trapezoidal, como explica Lynch et al. [9]. Esse perfil de velocidade considera o melhor desempenho, menor tempo, apenas para movimentações em linha reta, nas quais a inércia tangencial e a aceleração são constantes. Para um caso geral, onde pode ser preciso variar a aceleração por conta dos limites dos atuadores, a capacidade da trajetória passa a depender do estado atual do robô.

Para o caso geral, Lynch et al. [9] demonstra que, a partir de um caminho $\theta(s)$ especificado e um perfil trapezoidal, podemos aplicar outras manipulações matemáticas para encontrar o tempo ótimo e um perfil de velocidade que respeite o movimento possível de ser executado pelos atuadores.

Otimizações da escala do tempo podem ser desenvolvidas em função do plano cartesiano $\theta(s,\dot{s})$. Aplicando algumas manipulações matemáticas obtemos as funções que limitam a aceleração máxima e miníma possíveis em função do estado, para todo instante no tempo

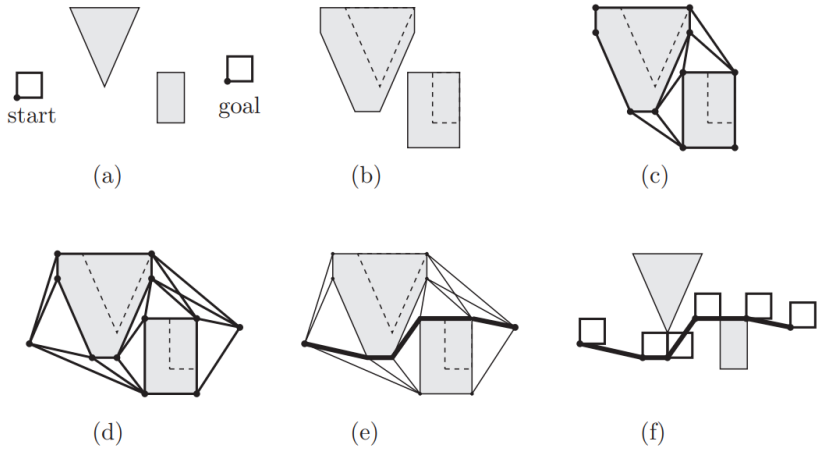


Figura 5: (a) A configuração de início e objetivo para um robô móvel quadrado em um ambiente com um obstáculo triangular e um retangular. (b) Os obstáculos com a margem adicionada da dimensão do robô. (c) As conexões do grafo de visibilidade no mapeamento R, representando o espaço-C livre. (d) O grafo completo consiste em R nós partindo do "start"até o destino "goal", juntamente com os links que conectam esses nós aos nós visíveis de R. (e) A busca no grafo resulta no caminho mais curto, mostrado em negrito. (f) O robô é mostrado atravessando o caminho. Fonte: Lynch et al.[9]

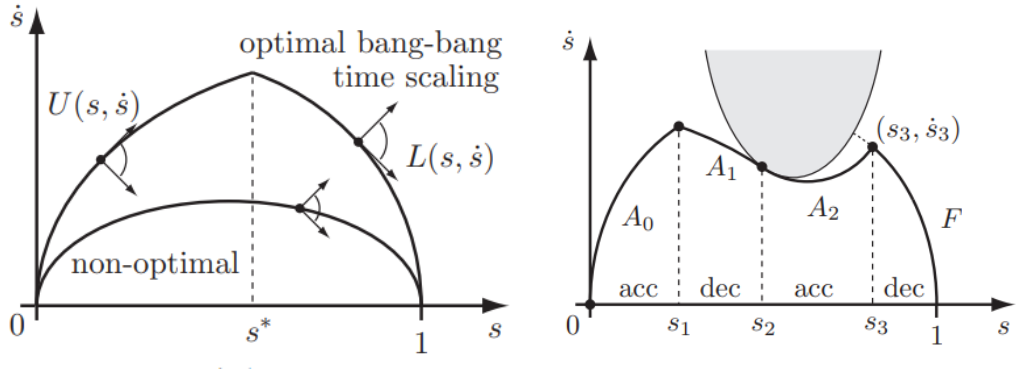
$L(s,\dot{s}) \leq \ddot{s} \leq U(s,\dot{s})$. Integrando essas funções, obtemos a curva de velocidade limite, mostrada na Figura 6(a). Para uma escala de tempo satisfazer as restrições de aceleração, a tangente da curva de escala temporal deve estar dentro do cone viável para aproximadamente todos os pontos da curva.

Tendo definido os limites de operação para minimizar o tempo total, precisamos que a trajetória desenhada no plano $\theta(s,\dot{s})$ tenha a maior área possível respeitando os limites. Com isso, teremos uma escala de tempo que sempre opera nos limites chaveando entre um extremo ou outro. Este é o princípio da técnica BANG-BANG (TBB), um perfil que acelera ao máximo na curva $U(s,\dot{s})$ até um certo $S*$ e depois desacelera ao máximo na curva $L(s,\dot{s})$.

Nem sempre é possível ter um único ponto $S*$ que satisfaça a tarefa de navegação especificada, dado o custo computacional de analisar vastamente os possíveis $S*$, para minimizar esse problema são usadas abordagens de dividir para conquistar, pois o problema de encontrar um ponto de inversão de aceleração na tarefa pode ser subdividido em N subproblemas responsáveis por segmentos da trajetórias que ao serem somados representam a tarefa completa, ideia mostrada na Figura 6(b), para definir a quantidade de segmentos e os pontos de cortes é comum estratégias de amostragem aleatórias.

Para aplicações em robôs omnidirecionais, onde os atuadores acoplados a rodas não-holonômicos somam suas forças para produzir um vetor velocidade que possibilita a navegação em todas as direções, algumas otimizações e simplificações são possíveis, como desenvolvido por Purwin et al. [13].

A partir da modelagem de corpo livre mostrada na Figura 7, podemos criar o envelope de aceleração baseado nos limites das capacidades do robô no plano XY, ao sincronizar os limites



(a) Gráfico de uma escala de tempo não ótima e da escala com o perfil TBB.

(b) Dependendo do estado do robô não será possível executar apenas um chaveamento entre aceleração e desaceleração, a região cinza representa uma zona proibida.

Figura 6: Gráficos no plano $\theta(s,\dot{s})$ com os perfis de velocidade para uma tarefa de navegação. Fonte: Lynch et al. [9].

para velocidades lineares e rotações, podemos simplificar o envelope a um cone de aceleração, uma expansão da curva apresentada na Figura6, que possibilita computar trajetórias em tempo real mantendo um alto desempenho dos resultados e utilizando a manobrabilidade facilitada dos robôs omnidirecionais, os envelopes podem ser vistos na Figura 8.

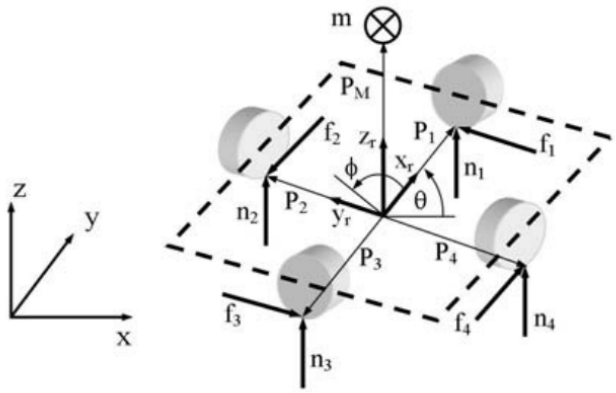
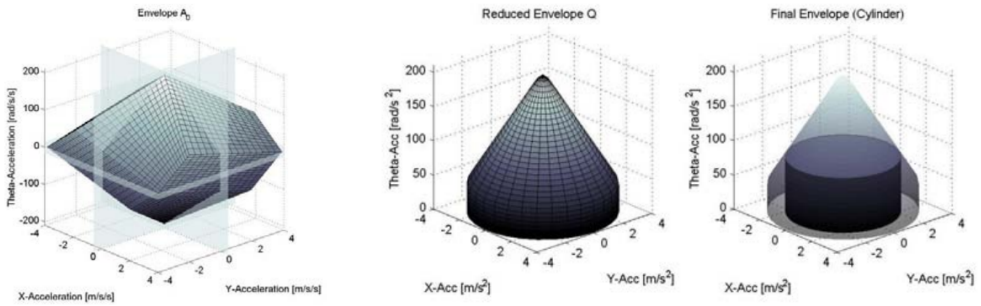


Figura 7: Diagrama de corpo livre de robôs omnidirecionais redondos, ajustando a força aplicada por cada motor é possível gerar um vetor velocidade para qualquer direção. Fonte: Purwiw et al. [13]

Tendo o cone de acelerações, como desenvolvido por Purwin et al. [13], podemos simplificar as equações, agrupando elas em cinco casos:

1. Caso 1: A velocidade está negativa em relação à trajetória, o veículo deve atuar para fazer o robô chegar a velocidade zero.

2. caso 2.1: A velocidade está abaixo da possível para o estado, o veículo deve atuar para acelerar até a velocidade máxima.



(a) Envelope de aceleração bruto modelado a partir da aplicação das capacidades do robô no modelo de corpo rígido.

(b) Cone de aceleração obtido a partir da sincronização das velocidade lineares e angulares possíveis.

Figura 8: Envelope de acelerações modelado a partir da Figura 7. Source: Purwiw et al. [13].

3. caso 2.2: A velocidade é máxima para o estado atual, o veículo deve atuar para se manter com velocidade constante.

4. caso 2.3: A partir de uma velocidade positiva em relação à trajetória, o veículo deve atuar para fazer o robô chegar a velocidade zero.

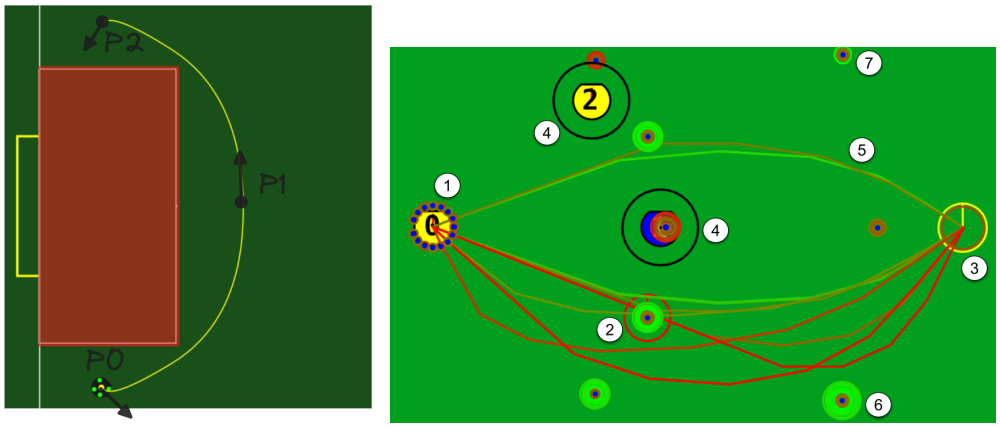
5. caso 3: A velocidade é maior do que a permitida para o estado atual, o veículo deve atuar para reduzir sua velocidade até a máxima permitida.

Com isso, determinando o caso do estado atual, podemos manter o robô se movendo em alta performance calculando de forma online a sua trajetória.

Por fim, as implementações da estratégia baseada na técnica TBB para a tarefa de navegação também foi explorada dentro do contexto de SSL, baseando se nos trabalhos descritos por Purwin at el.[13] e Balkon et al.[4]. As equipes de robótica Tigers[12] e Er-Force[19] também se basearam nestes trabalhos para suas implementações de perfis de velocidade Bang-Bang a partir de trajetórias trapezoidais, e que foram disponibilizadas como código aberto [1, 2]. A variação entre as abordagens se dá justamente na heurística de como dividir a tarefa de computar a trajetória da origem até o destino em sub-problemas, a equipe Tigers cria uma constelação de pontos fixos ao redor da posição atual do robô e testa as combinações possíveis em árvore até alcançar o destino, já a equipe ER-Force busca aleatoriamente um estado de pivô entre a estado atual e o destino que priorizando a velocidade e evita a colisão com o obstáculo mais próximo prioritariamente, essa diferença pode ser vista na Figura 9.

2.4 TRABALHOS RELACIONADOS

Trabalhos comparando famílias de algoritmos de planejamentos de caminhos e abordagens de controles direcionado a robôs móveis são amplamente estudados, entretanto, pesquisas



(a) Trajetória mais rápida, utilizando o estado inicial P1, o pivô no estado P1 e o estado final P2, que ao conectar eles é possível contornar a área do goleiro. Source: Andreas at el.[19]

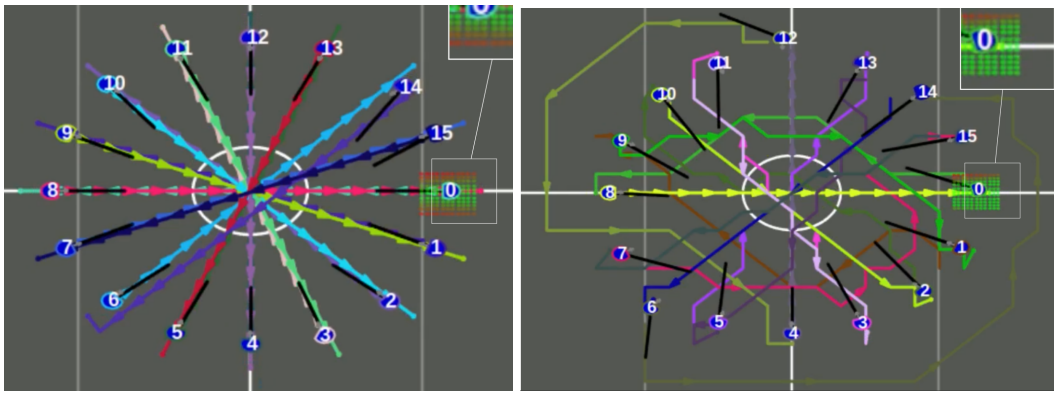
(b) Busca de trajetórias desenvolvido para a Tigers: (1) Posição do robô, (2) Melhor destino intermediário, (3) Destino, (4) Obstáculo ativo, (5) melhores trajetórias (verde: melhor, vermelho: pior) e (6) destino intermediário (azul). Source: Nicolai at el.[12].

Figura 9: Heurísticas de busca e amostragens de trajetórias dos times de SSL.

voltadas para analisar e comparar a qualidade de uma estratégia de navegação de forma mais ampla são mais restritas, muitas vezes, se prendendo a gerar os caminhos e analisar eles de forma separada da execução da navegação.

A dissertação de Costa [5] explora essa analise mais ampla da qualidade olhando para a navegação resultante em condições ambientais não ideais com ruído e incerteza, além de ambientes mais complexos por envolver a coordenação de múltiplos robôs. Costa [5] busca analisar a capacidade de três algoritmos: Multi Robot A Star (MRA*), Dynamic Visibility Graph A Star (DVG+A*) e Probabilistic Safety Barrier Certificates (PrSBC). Neste trabalho foi analisada a tarefa de coordenar a navegação de múltiplos robôs em um ambiente incerto e dinâmico em um contexto de uma partida de futebol de robôs com da liga de SSL, foi apresentado o contexto das suas implementações e de como os testes eram executados.

Para validar cada algoritmo, Costa definiu cenários de testes com a movimentação antipodal, na qual os robôs ficam distribuídos em uma circunferência e recebem a tarefa de movimentação de ir para a posição oposta na circunferência, como mostrados na Figura 10. Além do teste mencionado, foram feitos testes de uma situação típica de um jogo de SSL considerando a movimentação sobre marcação. O trabalho realizou os testes analisando as seguintes métricas: tempos de planejamento, atualizações do mapa, navegação e de desvio de obstáculos, velocidade média geral, distância mínima entre os robôs e número de colisões. Com isso, foi possível estabelecer vantagens e desvantagens de cada estratégia avaliada, e fornecer tanto um modelo de testes e análises para outros objetivos ou algoritmos em contextos próximos que podem servir de base para definir a melhor estratégia dada a restrições do contexto usado.



(a) Planejamento inicial para a movimentação antipodal com 16 robôs usando o DVG+A*.

(b) Planejamento ajustado para a movimentação antipodais realizados com 16 robôs usando o MRA*.

Figura 10: Gráficos do planejamento de execução dos testes antipodais. Source: Costa [5].

3

ESTRATÉGIAS DE NAVEGAÇÃO

Neste capítulo, são apresentadas as estratégias que serão analisadas e comparadas neste trabalho, explorando seus conceitos, regras de implementação e o fluxo de dados presente.

Uma estratégia de navegação consiste em todo o processo necessário para produzir atuação no robô para movê-lo de um ponto até outro no espaço. A entrada do sistema é uma tarefa de navegação, que informa qual a posição destino deve ser alcançada. Internamente, cada estratégia pode incluir uma quantidade de camadas e abordagens diferentes para fazer o processo de interpretação do contexto do mundo, de planejamento e criação de atuações, ao final, as atuações correspondentes são executadas pelo robô produzindo movimento.

Ao analisar a tarefa de navegação recebida e o resultado obtido na navegação como o desempenho na execução e a trajetória feita, temos uma forma mais genérica de avaliar que abrange diversas arquiteturas e abordagens de navegação. Para esse trabalho serão consideradas trés estrategias: A técnica do VG e as trajetórias Bang-Bang em duas versões, uma versão simples em aplicação direta, o TBB, e outra versão com adaptações e melhorias de fluxos para extrair mais desempenho TBBES.

3.1 ARQUITETURA DO SISTEMA

Em sistemas de robótica móvel é comum separar as responsabilidades dentro da arquitetura. Como a navegação é uma atividade base para a maioria das atividades feitas por um robô móvel, ela é abstraída a partir de uma rotina de navegação para o restante do sistema. Com isso, as funcionalidades mais complexas podem usar facilmente a navegação apenas fazendo uma chamada de processo com os parâmetros desejados. Essa abordagem foi usada no sistema de controle de robôs do RobôCIn, o SSL-unification [11], que aplica a navegação a um contexto bem mais amplo, fazendo robôs jogarem bola. Desde 2019, quando entrou na categoria de SSL, o RobôCIn utilizava por padrão uma estratégia baseada somente no planejamento de caminhos [16], com o grafo de visibilidade, que serviu aos propósitos de estabelecer uma versão funcional do sistema, mas que ao longo dos anos foi se mostrando um gargalo para extrair mais desempenho dos robôs, levando a criação de muita lógica especifica integrada para melhorar o funcionamento.

O SSL-Unification implementa o fluxo de dados mostrado na Figura 11, a cada ciclo

o dados do mundo são recebidos e um processamento em cascata é feito, onde cada módulo processa as entradas e produz saídas correspondente para o módulo seguinte. Dentro desse fluxo, quando o comportamento é planejado, ele define a ação atual e se isso for dependente de uma navegação, o processamento resulta em um pacote com uma tarefa de navegação para o robô ir até um ponto. Ao ser recebido pelo módulo de *Planning*, essa tarefa é calculada com base nos parâmetros recebidos e uma atividade de atuação, a partir do planejamento, é criada. Por último no módulo de *Navigation* a atividade de atuação é calculada resultando em comandos de velocidade para o robô.

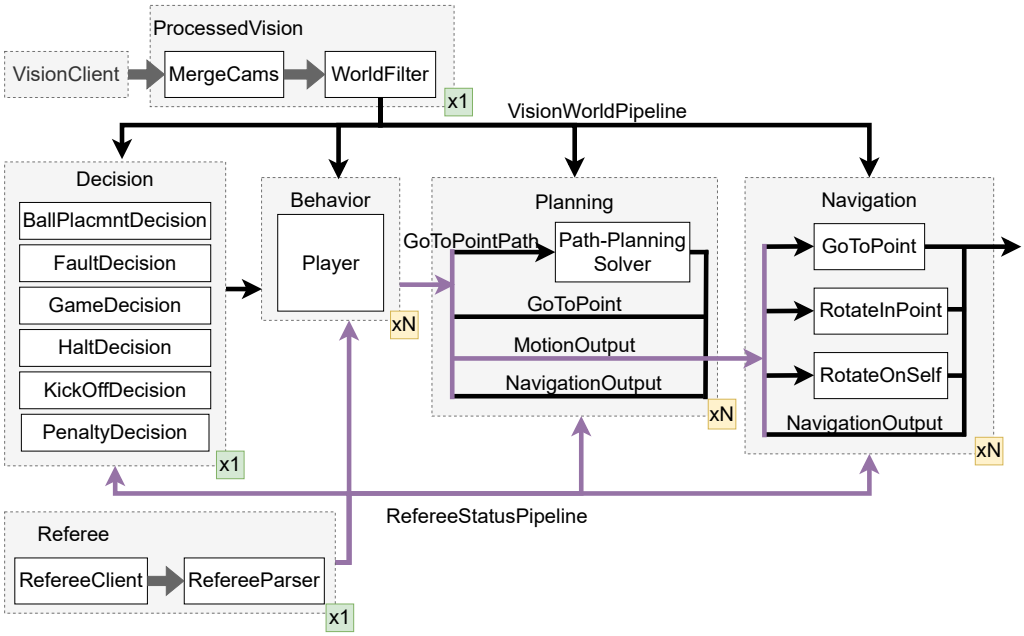


Figura 11: Arquitetura dos módulos do software SSL-Unification, no fluxo todas as ações possuem uma tarefa de navegação associada. Fonte: Oliveira et al. [11]

Dentro dessa arquitetura de módulos podem ser implementadas várias estrategias devido a facilidade de trocar e combinar módulos que processam os pacotes, mantendo a interface de comunicação entre eles. As estratégias de navegação apresentadas a seguir serão comparadas com as etapas do fluxo ideal de concretização de uma tarefa de navegação, apresentado no Capítulo 2.

3.2 GRAFO DE VISIBILIDADE COM NAVEGAÇÃO DISCRETA

A Figura 12 mostra como o fluxo de dados passa pelos módulos intermediários para processar a tarefa de navegação em uma execução concreta. A partir de uma tarefa de navegação temos dois módulos principais, nos quais o planejamento de movimentação aparece implicitamente apenas na navegação.

Planejamento de caminho (Grafo de visibilidade): Recebe um objetivo geométrico de posição destino e regras de negócio relacionadas tanto à busca, quanto a como os pontos de

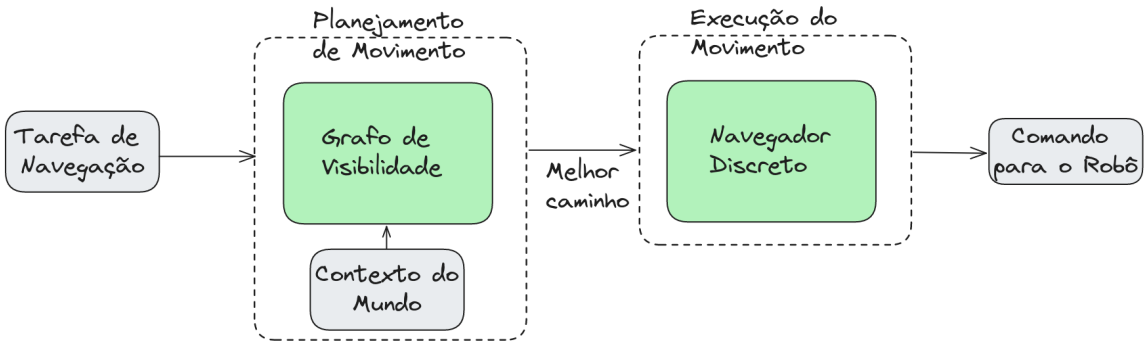


Figura 12: Fluxo da execução da estratégia do grafo de visibilidade. Fonte: Autor.

saída vão se relacionar com o controle. Dependendo das informações do mundo e das dimensões do robô, o submódulo de contexto do mundo cria uma representação em grafo, inscrevendo os obstáculos em polígonos e conectando seus vértices, tendo como saída uma lista de pontos para alcançar o destino.

O principal problema do processo de concretização em relação ao ideal, é que o planejamento de caminhos se liga diretamente à camada de controle, sem computar a trajetória previamente e, consequentemente, sem planejar o movimento a ser feito de forma explicita. Com isso, o controle fica limitado e instável principalmente quando no caso de altas velocidades altas e quando uma maior responsividades em desvios de obstáculos se fizer necessária.

Navegador Discreto: Recebe a lista de pontos e o estado atual do robô (posição, velocidade), capacidades de movimentação do robô (acelerações e velocidades, mínimas e máximas). Por conta das limitadas informações planejadas, apenas o caminho geométrico, todo o controle de estado é feito somente baseado em posição usando cada posição intermediária como um objetivo que ao ser alcançado é trocado pelo próximo até o último. A cada ciclo um vetor da posição atual do robô para o próximo destino é criado, ele é redimensionado em função das velocidades mínimas e máxima permitidas. Este vetor é a velocidade que será enviada no comando de atuação do robô para aquele ciclo.

O chaveamento entre destinos intermediários na navegação, como pode ser visto na Figura 13, é a principal fonte de gargalo de desempenho da estratégia, pois, para evitar colisões e conseguir seguir o planejamento, a velocidade do robô é reduzida para evitar inércia resultante faça o robô distorcer o caminho por uma força não compensada, gerando colisões e invasões.

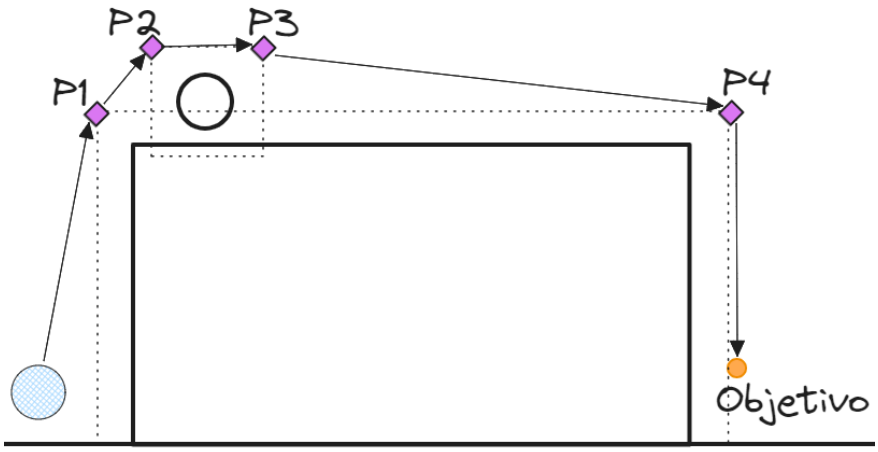


Figura 13: Diagrama do caminho planejada pelo grafo de visibilidade para mover robô ao redor da área do goleiro, os polígonos dos obstáculos estão marcados pela linha pontilhada. P1, P2, P3 e P4 são arestas dos polígonos dos obstáculos que quando ligados no grafo representam o menor caminho. Fonte: Autor.

3.3 TRAJETÓRIAS BANG-BANG COM NAVEGAÇÃO BASEADA EM ESTADOS

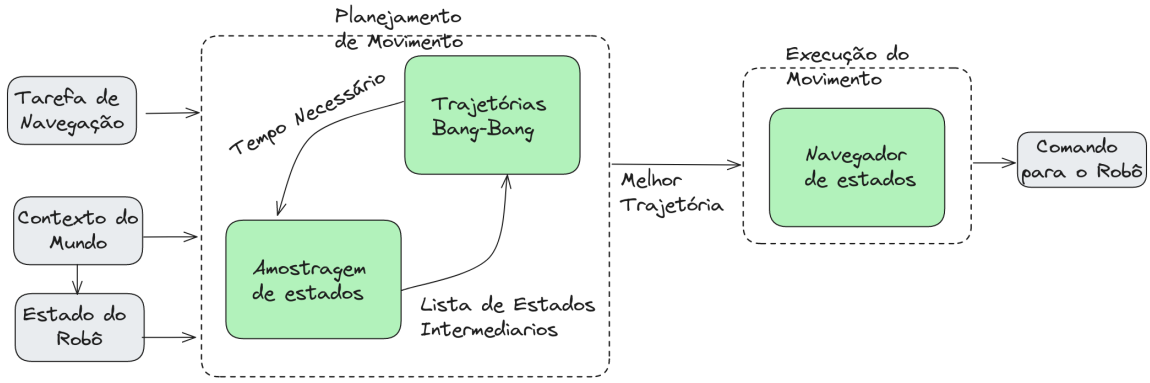


Figura 14: Fluxo da execução da estratégia das trajetórias Bang-Bang. Fonte: Autor.

Como descrito no Capítulo 2, as trajetórias Bang-Bang se baseiam no controle *on/off*, calculando a cinemática do robô entre o estado atual e destino e fazendo uma busca binária para encontrar a melhor combinação de estados dos atuadores. Ao ser combinada com um algoritmo exploratório de amostragem do espaço fora dos obstáculos, é possível classificar essas amostras pelo tempo total de movimentação que levariam. Assim, ao reunir a sequência de amostras de menor custo, temos uma trajetória que aproximadamente representa o tempo mínimo naquele cenário. A trajetória resultante desse processo pode ser visualizada na Figura 15.

A versão implementada apresentada na Figura 14 e demonstrada na Figura 15, foi baseada no código aberto disponibilizado pela equipe ER-Force[1]. Adicionalmente foram feitos ajustes em parâmetros internos de pesquisa e amostragem, além disso, foi adotado o modelo para a

construção de obstáculos e a possibilidade de mais pontos intermediários como a equipe Tigers implementou [2].

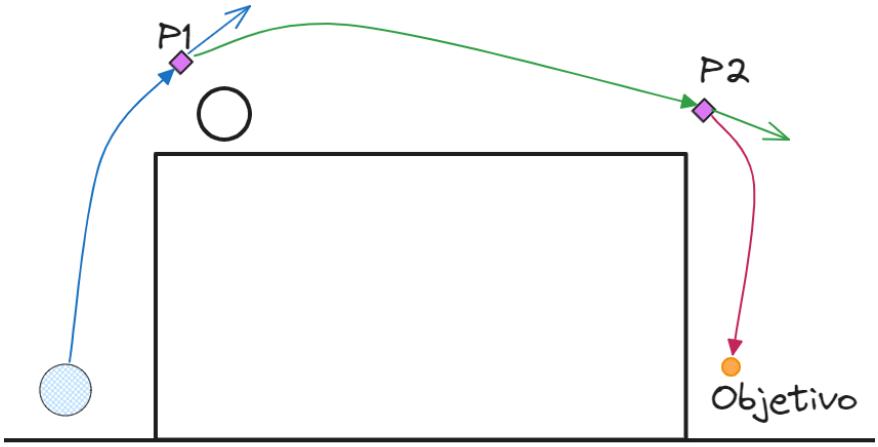


Figura 15: Diagrama da trajetória planejada pelo bang-bang para mover robô ao redor da área do goleiro, os obstáculos de um tamanho muito próximo ao dos elementos em cena. P1 e P2 são estados intermediários utilizados como pivô entre os segmentos de trajetória, em diferentes cores, para conseguir alcançar o objetivo, minimizando o tempo de execução. Fonte: Autor.

A Figura 14 mostra como o fluxo de dados passa pelos módulos intermediários para processar a tarefa de navegação em uma execução concreta. A partir de uma tarefa de navegação temos a participação dos três módulos definidos no fluxo ideal por Gasparetto et al. [6]:

Planejamento de caminho (Amostragem de estados): depende das informações do mundo e das dimensões do robô para montar sua representação espacial dos obstáculos e relacionar os pontos buscados no espaço. Consiste em um algoritmo exploratório, que busca pontos de pivô para ser os pontos intermediários entre a origem e o destino final que evitem o obstáculo mais próximo prioritariamente. Recebe um objetivo geométrico de posição, destino e uma configuração (regras de negócio relacionadas exclusivamente à busca, por exemplo, o que deve ser obstáculo). As dimensões do robô são utilizadas para verificar colisões, mas o robô é tratado como um ponto. Por fim, tem como saída uma lista de estados (Posição, velocidade, aceleração).

Planejamento de Trajetórias Bang-Bang: depende das informações do estado atual do robô (posição, velocidade, aceleração), capacidades de movimentação do robô (acelerações e velocidades, mínimas e máximas) para montar a representação da interação da dinâmica do robô com os obstáculos e da atuação do robô no mundo. Recebe uma lista de estados intermediários e uma configuração (regras de negócio relacionadas ao comportamento da atuação do robô, por exemplo, velocidade mínima e máxima, vetor de velocidade objetivo no final). Para a trajetória final serão somados o estado atual, os intermediários e final em uma lista de estados e Para cada estado ele é conectado ao estado seguinte interpolando o estado de navegação entre eles por uma trajetória Bang-Bang gerada por uma busca binária da combinação de aceleração máxima, desaceleração máxima e velocidade constante. Este módulo tem como saída uma lista

de segmentos de trajetórias do robô ao longo do tempo que conectam o estado atual ao estado final desejado, respeitando suas restrições e capacidades.

Esses dois módulos, descritos anteriormente, representam o planejamento do movimento que ao ser executado por iterações suficientes dos planejamentos de caminhos e de trajetórias torna possível obter uma trajetória com uma boa aproximação para o menor tempo. Este planejamento de movimento produz como saída uma lista de estados de movimentação do robô ao longo do tempo, onde para cada dado tempo tem-se a tupla (Posição, vetor de velocidade, vetor de aceleração, tempo até aquele ponto).

Navegador de estados: dependendo das informações do estado atual do robô (Posição, velocidade, aceleração), capacidades de movimentação do robô (acelerações e velocidades, mínimas e máximas), este módulo converte o planejamento na atuação do robô no mundo. O navegador recebe a lista de estados ao longo do tempo planejados para o robô tendo como base o estado estimado atual e o estado atual percebido para o robô. Assim, o módulo calcula a atuação necessária para seguir o planejado tendo como saída a velocidade de atuação que o robô deve aplicar naquele momento para os seus motores.

3.4 OTIMIZAÇÃO DE ESTADO E DE CONTROLE

No contexto da técnica VG, otimizações que melhorem o desempenho elevam muito a complexidade do sistema, pois o custo computacional de aplicar o Dijkstra no grafo não é viável. Dessa forma, a inclusão da dinâmica da movimentação é feita em um pós-processamento restrito ao que já foi planejado, buscando acrescentar informações que melhorem o processo de controle ao longo da execução de uma navegação.

Diante disso, as otimizações recaem sobre a execução do controle que possui poucas informações dentro do planejamento para criar a atuação, apenas uma sequência de deslocamentos. Por conta das mudanças bruscas de direção da lista de pontos intermediários, vindos de arestas dos obstáculos, a execução do controle acaba por priorizar manter a fidelidade e evitar colisões, pois, estará contornando os obstáculos, sacrificando a velocidade.

Por outro lado, aplicações envolvendo técnicas baseadas em trajetórias Bang-Bang possibilitam uma série de otimizações em etapas prévias e depois do planejamento das trajetórias, uma vez que o algoritmo é muito dependente do valor preciso do estado atual do robô, e da execução da sequência de estados planejada de forma fiel, o que melhora diretamente seu desempenho em relação a um caso base mais simplificado.

No caso mais simplificado de implementação, é feita tanto uma aproximação de entrada quanto de saída para o planejamento de trajetórias. Na entrada do sistema é assumido que o estado do robô visto e processado atualmente pela visão corresponde ao estado atual dele, o que é uma premissa problemática por conta do atraso implícito que existe no sistema de percepção global externo ao robô, como descrito no Capítulo 2. Na saída do sistema é assumido que o robô não precisa de nenhum esforço adicional de correção para alcançar o estado desejado no

planejamento, assim, a velocidade de atuação para o robô executar é a planejada diretamente, o que pode gerar distorções e erros durante a execução de uma navegação.

Buscando pontos de melhorias em relação a uma aplicação simples de trajetórias Bang-Bang, podemos adicionar módulos fluxo da uma navegação. Como mostrado na Figura 16, os módulos adicionados buscam melhorar a responsividade do sistema levando a um ganho de desempenho.

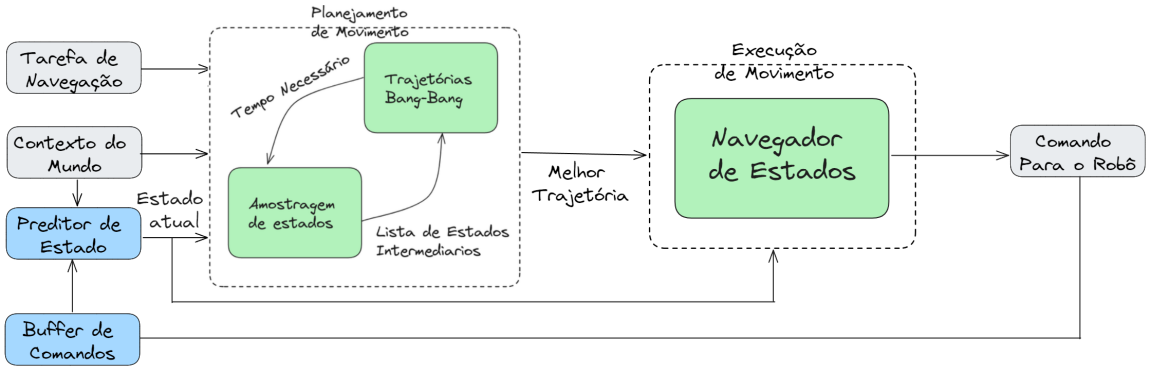


Figura 16: Fluxo da execução da estratégia das trajetórias bang-bang melhorado, em azul temos os módulos adicionados. Fonte: Autor.

3.4.1 Preditor de Estado

Problemas de atrasos de comunicação e obtenção de informação em sistemas robóticos móveis são amplamente estudados e em um contexto onde o sistema de controle recebe um comando enviado para a execução remotamente no robô, representam um desafio ao controle robusto. Um dos principais impactos é na responsividade do sistema, pois os atrasos fazem os comandos serem executados atrasados pelo robô e a percepção dos efeitos desses comandos também demoram a serem notadas.

Diante desse contexto, e buscando minimizar os impactos desse atraso de comunicação e percepção, uma abordagem possível é predizer em qual estado o robô está a partir da informação atual. Com isso, o ciclo de emissão de um comando e percepção de sua execução no robô passa a ser considerado, podendo assim, fazer o rastreio da trajetória de forma mais eficiente e, consequentemente, melhorando a resposta transitória de controle do sistema por evitar gerar comandos que não correspondem ao estado atual do robô.

Temos várias formas conhecidas na literatura para estimar o estado dos objetos em um tempo futuro, mas a maioria deles apenas estende o estado atual ao longo do tempo, modelando como o mundo interfere na sua propagação. No entanto, para o robô isso não é muito útil, pois o robô ativamente pode mudar seu estado atual com seus atuadores.

Pensando nisso, foram estudados trabalhos, como o desenvolvido por Velasco-Villa et. al. [17], que aplica um compensador baseado no preditor de Smith para minimizar os erros associados ao rastreio da trajetória de robôs móveis omnidirecionais, que se mostrou eficiente em antecipar o estado percebido para o atual do robô. Inspirado nesse conceito, para a técnica

TBBES em sistemas de tempo real, foi pensado um preditor que, baseado no estado atual percebido pela visão global externa ao robô, com atraso em relação ao real, e um *buffer* com os comandos enviados dentro da janela temporal de atraso de atuação, é capaz de prever o estado atual aplicando ao estado visto da visão esses N comandos e estimando seus efeitos. A estrutura deste módulo pode ser vista na Figura 17.

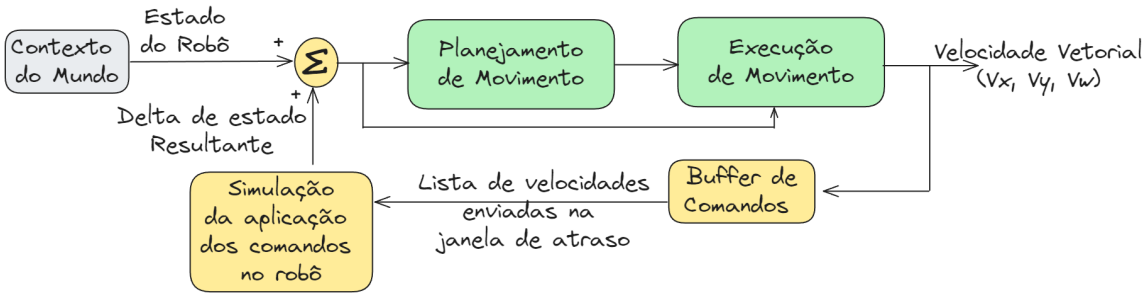


Figura 17: Fluxo do preditor de estado integrado ao fluxo de execução de navegação, amarelo estão os módulos utilizados para a predição. Fonte: Autor.

Para definir a janela de atraso de atuação do sistema, foi realizado o seguinte experimento com o sistema. Com o robô partindo do repouso, foi aplicado a ele uma função de velocidade linear em formato quadrado, em 50% do ciclo foi enviado a velocidade máxima e na outra metade foi enviado velocidade zero, a uma frequência baixa o suficiente para o robô conseguir uma velocidade próxima da máxima e também desacelerar com os comandos de velocidade zero. Como mostrado na Figura 18, ao analisar a velocidade enviada e a percebida ao longo do tempo, foi percebido um deslocamento temporal entre o último comando de velocidade máxima e o pico de velocidade percebida para o robô na visão global externa, ao repetir o processo 30 vezes, foi possível obter um ΔT médio de 0.082 segundos que representa o atraso de atuação considerando todos os passos e atrasos de processamentos ao longo do ciclo do sistema.

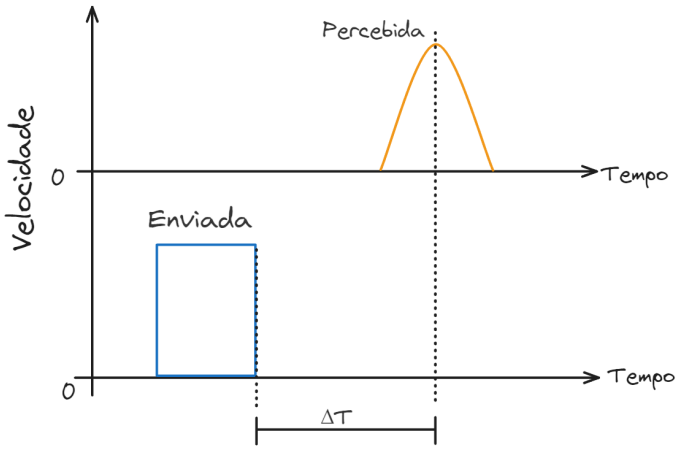
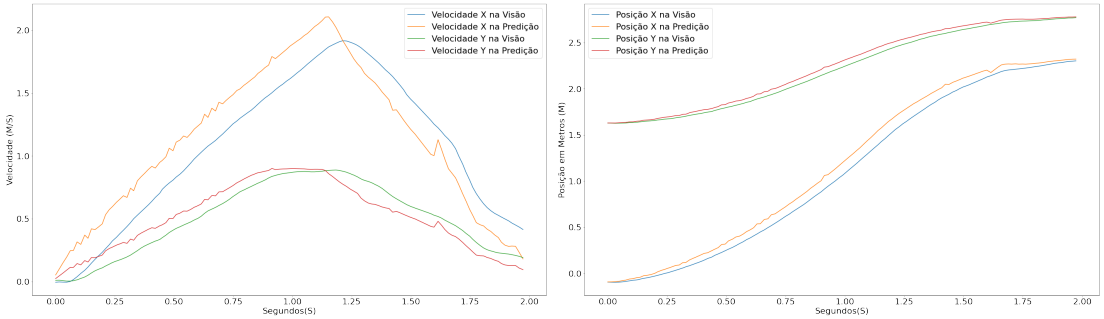


Figura 18: Diagrama da análise das velocidades ao longo do tempo no experimento para definir o atraso de atuação do ciclo do sistema. Fonte: Autor.

Uma vez definida uma janela de atraso de atuação condizente foi possível estimar com

mais fidelidade o estado atual. Como pode ser vista na Figura 19 para o movimento linear e na Figura 20 para o movimento angular, existe um efeito antecipatório em relação à percepção de mundo da visão global, a velocidade calculada responde rapidamente na Figura 19(a) e consequentemente pela maior velocidade, o descolamento do robô predito fica adiantado ao da visão conforme Figura 19(b), comportamento similar ocorre na Figura 20.



(a) Velocidade percebida pela visão vs a estimada pelo preditor ao longo do tempo.

(b) Posição percebida pela visão vs a estimada pelo preditor ao longo do tempo.

Figura 19: Comparativo entre o estado do robô percebido pela visão e o estimado pelo preditor de estados. Fonte: Autor

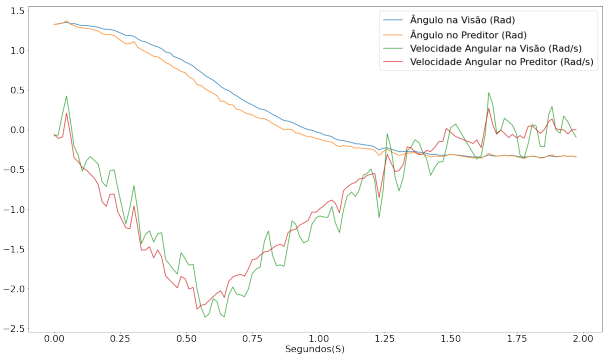


Figura 20: Ângulo e a velocidade angular do robô, comparativamente mostrando o percebido pela visão e a estimada pelo preditor ao longo do tempo. Fonte: Autor.

Nesse capítulo foram definidas e discutidas três implementações de estratégias para a realização de tarefas de navegação. Nos próximos capítulos nos concentramos em analisar comparativamente o desempenho das estratégias do VG e para as trajetórias Bang-Bang teremos duas versões, uma versão simplificada, TBB, e outra versão integrando o preditor de estados, TBBES, com o fluxo apresentado na Figura 16.

4

METODOLOGIA

Neste capítulo, é apresentada a metodologia para definição dos cenários de testes utilizados, o domínio dos testes, as métricas consideradas e como foram analisados os dados obtidos nos experimentos. Além disso, será apresentado o detalhamento dos testes que foram executados para definir o desempenho das estratégias de navegação analisadas.

A definição da qualidade da execução da navegação de um robô móvel para um cenário altamente dinâmico de forma precisa é algo complexo, pois uma grande variedade de cenários podem ocorrer, é difícil estimular a execução de navegações com os mais variados casos e, principalmente, é difícil traçar métricas numéricas consistentes para basear análises.

A metodologia apresentada neste capítulo busca explorar a definição de experimentos que estimulem contextos recorrentes na navegação de robôs móveis. Além disso, a definição de quais dados coletar durante a execução e como fazer o posterior processamento em métricas que resultam em valores numéricos que possibilitem a análise direta, buscando julgar a qualidade baseado principalmente no tempo necessário para a execução das tarefas e a precisão de execução quando comparado com o planejamento.

4.1 ARQUITETURA DA APLICAÇÃO DOS TESTES

Todo o experimento será conduzido utilizando o software de controle de alto nível do RobôCIn, o SSL-Unification, e para possibilitar a automação e replicação de testes, o controle do robô provido pelo software será aplicado no Simulator-CLI [1], simulador desenvolvido pela equipe ER-Force. O Simulator-CLI cria um campo virtual com os robôs, envia os mesmos pacotes de visão que o SSl-Vision[21] apresentando ruído e falhas de detecção com uma distribuição realista. Além disso, recebe comandos similares aos enviados aos robôs pelas equipes via rádio no contexto real. Ao enviar esses comandos se definem as velocidades de forma vetorial, o time e o robô, que executará o comando dentro da física simulada.

A arquitetura do sistema de testes é mostrada na Figura 21, nela é possível ver o fluxo de dados que corresponde à execução e análise dos testes. O SSL-Unification recebe pacotes de visão com as informações do estado atual do mundo no simulador. O módulo *TesterCoach* determina o comportamento a ser executado para todos os robôs de acordo o caso de teste selecionado

e as informações definidas para a sua execução pelo arquivo de test-case correspondente. Em seguida, os módulos de *Planning* e de *Navigation* planejam e criam a atuação em velocidade a ser enviada pela comunicação para o simulador, além disso, no módulo de navegação, a cada ciclo de execução, salva as informações atuais do planejamento e execução da navegação.

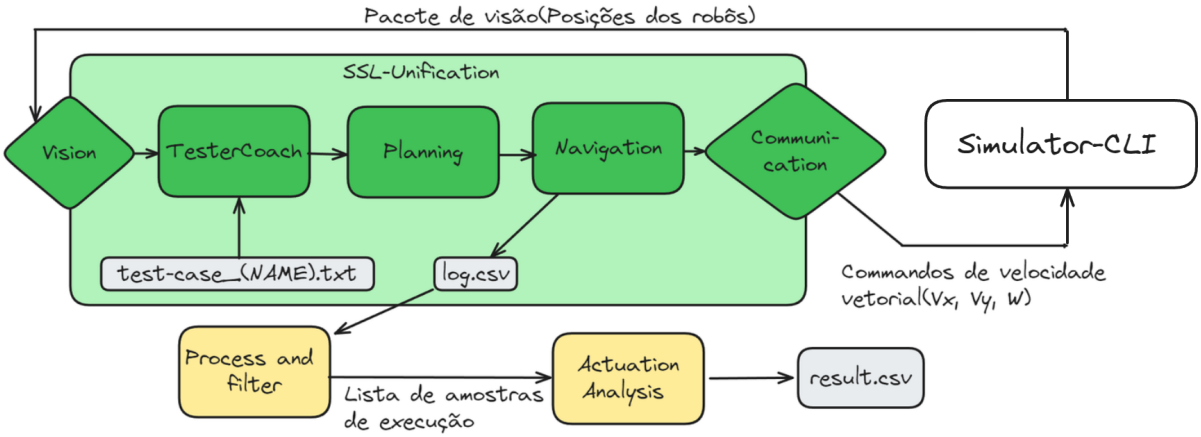


Figura 21: Arquitetura do método desenvolvido para teste de estrategias de navegação. Fonte: Autor

Após a execução de cada caso de teste descrito no arquivo de test-case, o log em CSV é filtrado e processado em uma lista de amostras contendo os dados estruturados. Por último, um *script* de análise é executado produzindo um arquivo CSV com os resultados de cada amostra e uma análise geral do grupo de amostras.

Como descrito ao longo do capítulo 3, cada abordagem de navegação possui um fluxo de elementos interconectados de forma específica dependendo da forma de planejamento de caminho e da estratégia de atuação/controle implementada. Isso cria uma série de variações e dependências que complicam ainda mais a análise e comparações. Assim, buscando aproximar o resultado mensurável comum a todos, a navegação será analisado de forma completa, considerando a ação de sair de um ponto para outro no espaço, respeitando restrições impostas ou que surgem dinamicamente e seguindo, dentro do possível, o planejado.

4.2 MODELAGEM DE CENÁRIOS DE NAVEGAÇÃO EM SSL

Como introduzido no capítulo 1, o contexto de uma partida de SSL é altamente dinâmico, volátil e até certo ponto caótico quanto às atribuições de atividades de navegação. Com isso, para validar o desempenho e analisar de forma coerente estratégias de navegação, é necessário buscar cenários que representem adequadamente as situações encontradas em partidas reais para aproximar ainda mais a coerência da análise.

Analisar a navegação no contexto real da aplicação gera muitas incertezas e variações entre as execuções, pois depende muito de como o jogo se desenvolve, tornado complexa uma análise comparativa entre estratégias de forma simples. Por conta disso, será analisada a

execução do time durante jogos e serão criados casos de testes fixos que sigam a distribuição e característica das navegações durante os jogos.

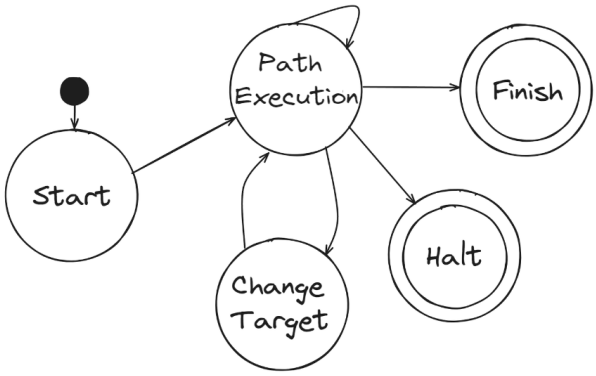


Figura 22: Estados da navegação em uma partida de SSL, a cada ciclo uma transição é executada, o estado inicial é indicado com uma seta de entrada e os estados terminais com um círculo duplo. Fonte: Autor

Durante uma partida de SSL a navegação de cada robô é executada continuamente, tendo seus estágios descritos pela máquina de estados apresentada na Figura 22. Em um fluxo sem interrupções ou mudanças, o robô em repouso, por ter concluído o objetivo de navegação anterior ou por não ter recebido atribuições ainda. Quando recebe uma nova atividade de navegação com um novo objetivo, o robô inicia execução no estado "Start". No próximo ciclo o robô vai para o estado de "Path Execution"e até atingir o objetivo fica em um laço executando uma transição para ele mesmo. Por fim ao conseguir cumprir o objetivo, o robô transiciona para o estado de "Finish", finalizando a tarefa. Além desse fluxo, por conta da dinamicidade de uma partida, a cada ciclo no estado de "Path Execution"podem acontecer mudanças de efeito imediato, o objetivo na tarefa de navegação pode mudar, transicionando para o estado de "Change Target"e depois voltando ao estado "Path Execution"com a nova tarefa definida. Por último, a qualquer momento no jogo, uma parada imediata pode ser solicitada pelo juiz e o robô vai para o estado terminal "Halt", no qual todos os robôs devem parar de se mover.

Dentre as ferramentas de softwares oficiais da liga, as ferramentas para a gravação e replay de logs são utilizadas para os jogos oficiais de competições como os da RoboCup e jogos da Latin American Robotics Competition (LARC). Os logs estão disponíveis publicamente e permitem executar novamente jogos específicos fornecendo ao software do time o mesma entrada de dados do jogo original. Com isso, tendo modelado os estados de navegação durante uma partida e executando novamente a partida a partir dos logs, é possível gerar estatísticas sobre o comportamento da navegação dentro da partida. A partir desses dados, é possível replicar a distribuição de comportamentos em cenários isolados, tornando eles condizentes com os cenários reais.

Para obter esses dados de distribuição foram analisados dois jogos oficiais do RobôCIn disputados recentemente com a última versão do SSL-Unification, as finais da Robocup 2023 e da LARC 2023. Essa análise de distribuição apresenta duas limitações pela dependência da

decisão do nosso software tomar a mesma ação com o mesmo robô, como na execução original, desse modo o RobôCIn precisa ser um dos times jogando e precisa que o software esteja na versão usada no originariamente.

Com a execução do replay dos logs foram obtidas 3105 execuções da máquina de navegação, partindo do estado de "Start"até o de "Finish". As movimentações interrompidas por um comando de halt vindas do juiz foram ignoradas, elas representaram 8.5% das execuções, analisando as 91.5% restantes das execuções se obteve a distribuição mostrada na Tabela 1. Dos dados obtidos uma coisa notável é o tamanho do desvio padrão em relação à média, que para parâmetros como o tempo total até a finalização chega a ser maior do que o tempo médio, isso ocorre pela grande variação de contexto coletados, durante o jogo o robô pode querer se mover por alguns milímetros ou alguns metros e, além disso, durante o percusso pode trocar de objetivo, levando o robô a percorrer mais alguns metros, esse desvio será usada para gerar a mesma variação presente nos cenários reais.

Tabela 1: Distribuição estatística dos fluxos de execução da máquina de estados da navegação, da máquina de estados na Figura 22.

Parâmetro	**Média**	**Desvio Padrão**
Tempo de Execução Total, até a Finalização (S)	3.95	4.05
Tempo de Execução do Estado "Change Target"até a Finalização (S)	0.35	0.43
Taxa de Conclusão até Objetivo ao entrar no Estado "Change Target"(%)	0.89	0.13
Distância para o Objetivo no Estado de "Start"(M)	0.75	1.25
Distância para o Objetivo ao sair do Estado "Change Target"(M)	1.47	1.43
Tempo para a Primeira Passagem pelo Estado "Change Target"(S)	0.06	0.12
Tempo entre as Passagens pelo Estado "Change Target"(S)	0.10	0.17
Quantidade de Passagens pelo Estado "Change Target"	24	28

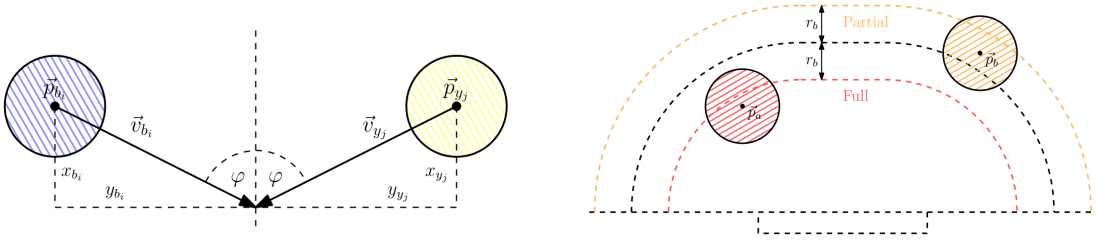
4.3 DADOS E MÉTRICAS AVALIADAS

Dentro do contexto de SSL temos uma relação direta entre velocidade e o desempenho do time, uma vez que assim como um jogo de futebol com humanos, ser mais rápido possibilita a inversão de jogadas, chegar primeiro em bolas chutadas, surpreender a defesa inimiga antes que se organize completamente obtendo uma grande vantagem. Além disso, também é necessário ser preciso nas movimentações, não é eficiente ser somente rápido se o robô não consegue cumprir a tarefa de navegação para se alinhar corretamente no chutar, ou descumpre regras por acabar colidindo com outros robôs, ou invade áreas indevidas.

Um jogo de SSL possui uma série de regras que podem ser consultadas no site oficial [14] e são julgadas de forma automática durante as partidas pelo software AutoRef [10]. Dentre as regras da categoria existem duas que especialmente testam a capacidade de navegação segura dos times: durante toda a partida é proibido entrar nas áreas dos goleiros e, dependendo do

comando atual do juiz, outras áreas são proibidas também como a zona próxima da bola em posicionamento para chutes e um certo raio da linha da bola até sua posição destino durante o reposicionamento automático. Além disso, durante toda a partida, o contato e impactos substanciais entre robôs são punidos com faltas que culminam em cartões de forma similar ao futebol humano. Desse modo, ter um planejamento de caminhos e de trajetória robusto alinhado a uma estratégia de navegação que permita a execução do que foi calculado de forma estável e consistente se mostra essencial para ter um alto desempenho sem infringir as regras. A análise do contexto do mundo utilizadas para julgar faltas por colisões e invasões é mostrada na Figura 23, nela é definido o diagrama de velocidades usado para medir a velocidade relativa em contato de robôs e a posição relativa à área do goleiro que geraram falta para os times.

Em uma partida de SSL, ao violar regras, faltas são anotadas para o time. Para as faltas de maior gravidade somente uma resulta em um cartão amarelo, mas nos casos de invasões e colisões são necessárias quatro faltas acumuladas dado o alto número de eventos que podem acontecer em um curto espaço de tempo. Um cartão amarelo fica ativo por dois minutos de jogo valido e obriga o time a tirar um robô de campo para cumprir a punição ao longo desse tempo. Um time pode ficar com até dois cartões amarelos ativos, o terceiro é convertido em um cartão vermelho que tira um robô do jogo em definitivo. Durante a partida ter robôs a menos, assim como no futebol de humanos, impacta diretamente na capacidade do time de jogar, pois a vantagem numérica do adversário em posicionamento, marcação e na manutenção da posse da bola cresce bastante.



(a) Velocidade vetorial relativa entre os robôs utilizados para o cálculo de colisão. Segundo as regras de implementação de 2023, se os robôs estiverem a uma distância menor de 200 *mm* e uma diferença vetorial de velocidade maior ou igual a 1.5 m/s a colisão é acusada.

(b) Posição relativa do robô a área do goleiro, onde pode ocorrer uma invasão parcial ou total, atualmente a área do goleiro é retangular, mas as mesmas regras se aplicam. Após o primeiro evento de falta o robô tem 2 segundos para sair da situação ou receberá uma nova falta.

Figura 23: Análise para aplicação automática de faltas. Fonte: Lukas [10].

Para a posterior aferição do desempenho, durante a execução de cada teste serão salvos logs em arquivos para análise dos dados da execução da máquina de navegação de forma offline. Para cada estado serão coletados a marcação de tempo atual e os seguintes dados:

- Start: Distância do robô para o objetivo e tempo estimado.
- Path Execution: Lista de estados (Posição, velocidade, tempo) calculados pela trajetória até o destino, posição e velocidade da visão, posição e velocidade referência

da trajetória, posição correta considerando o tempo de execução, erro de posição, erro de posição acumulado, erro de tempo, erro para a posição correta e a velocidade e posição calculada pelo preditor.

- Change Target: Distância do robô para o objetivo antigo e distância do robô para o objetivo novo, tempo estimado para o novo objetivo e o tempo de execução atual.
- Finish: O tempo atual e o tempo total da execução desde do início.

Além das informações por estado, descritas acima, durante toda execução dos testes o AutoRef será executado e eventos de colisões e invasões serão anotados no log. Os eventos de colisão descrevem os robôs envolvidos e qual foi a velocidade de colisão registrada, o evento de invasão detalha qual foi o tipo de obstáculo e qual distância o robô invadiu.

Diante desse requisitos, serão analisados as seguintes métricas para definir o desempenho das estratégias de navegação:

- Velocidade média e tempo de execução: métrica clássica para dar um quantitativo de quão rápido o robô se moveu durante a execução da tarefa.
- Consistência do caminho gerado ao longo da execução, utilizando o algorimo de dispersão 1, conforme o caminho é recalculado: para analisar a precisão da execução, o quanto a atuação de navegação está distorcendo o caminho planejado ao tentar aplicá-lo ao robô.
- Erro Quadrático Médio (RMSE) do Tempo estimado pelo real: Métrica aplicada a estratégias com planejamento de trajetórias, para a comparação do tempo estimado e o tempo real da execução.
- RMSE da posição no tempo entre o estado atual e o esperado: Métrica aplicada a estratégias com planejamento de trajetórias, para análise de sincronia entre o estado estimado e o estado real do robô.
- Invasões e colisões cometidas: Métrica para analisar a segurança e a precisão da movimentação.

Grande parte das métricas são valores quantitativos e erros a serem minimizados, entretanto, a métrica de consistência do caminho gerado exige uma análise dos dados de forma mais complexa. Para esse objetivo, foi desenvolvido o Algoritmo 1, que analisa a dispersão de caminho resultando uma nota a ser minimizada. A ideia principal do algoritmo é, a partir de uma matriz de zeros com as dimensões do campo, a cada ciclo iterar sobre o caminho planejado até o destino final e escrever na posição correspondente a do planejamento na matriz um valor que para ciclo começa com o valor um e vai com decaimento ao longo dos itens pela distância percorrida. Depois essa matriz é somada e normalizada pela distância real percorrida naquela

navegação, assim se nenhuma distorção ou incerteza de caminho ocorrer a nota resultante tende ao valor um.

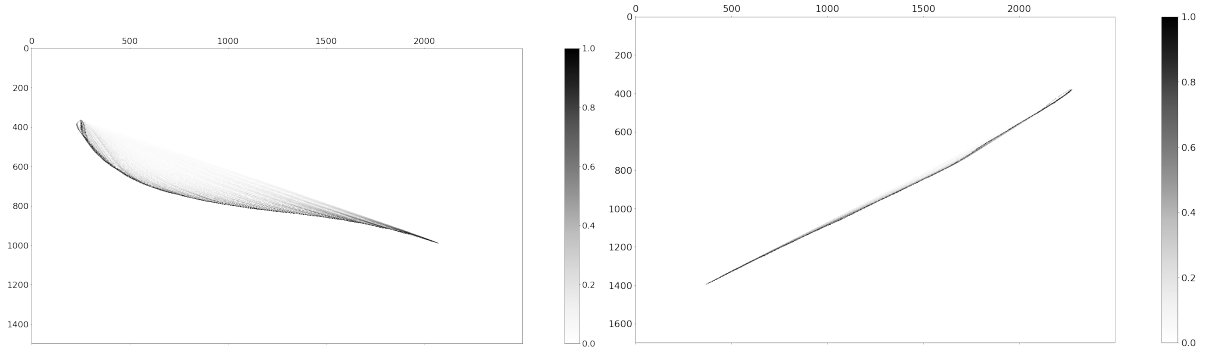
Algoritmo 1: Avaliação da consistência da trajetória a partir dos dados coletados no estado "Path Execution".

```
1  //Preprocessor logs in list of dataclass entities;
2  entities_list = convert_data_to_entity(files);
3  field_matrix = MatZeros(field_width_mm, field_length_mm);
4  execution_sample_size = entities_list.size();
5  //Execution Loop;
6  for i, from 1 to (execution_sample_size -1) do
7      running_node = entities_list[i];
8      gamma = 0.95;
9      alfa = 1.0;
10     current_travel_distance = 0 ;
11     for j, from 1 to (running_node.trajectory.size() -1) do
12         //Interpolate points with line;
13         state_current = running_node.trajectory[j];
14         state_next = running_node.trajectory[j+1];
15         x_serie = [state_current.position.x, state_next.position.x];
16         y_serie = [state_current.position.y, state_next.position.y];
17         m, c = linear_regression(x_serie, y_serie);
18         interpolate_points = values_in_range(m, c, x_serie[0], x_serie[1]);
19         for v, from 0 to interpolate_points.size() do
20             //Apply points in matrix;
21             pos_x = interpolate_points[v].x;
22             pos_y = interpolate_points[v].y;
23             field_matrix[pos_x][pos_y] = alfa;
24             current_travel_distance += dist_to(Point2d(x_abs, y_abs), lst_point)
25             if current_travel_distance >= 20 then
26                 //Apply decay in alpha;
27                 current_travel_distance = 0 ;
28                 alfa = alfa ∗ gamma;
29 //Return sum normalized in matrix for evaluate;
30 return sum(field_matrix)/full_travel_distance(sample);
```

4.4 TESTES DA TAREFA DE NAVEGAÇÃO

Analisando os jogos e buscando características recorrentes dentro de uma partida e os caminhos possíveis na máquina de estados da Figura 22, encontramos três grupos de características de navegação:

- Movimentações partindo do repouso e sendo executadas completamente, como as



(a) Desenho da matriz de uma amostra com um alto volume e com isso, uma alta dispersão.

(b) Desenho da matriz de uma amostra com um baixo volume e com isso baixa dispersão.

Figura 24: Desenho da matriz de pontos resultantes da aplicação do algoritmo de dispersão 1, na qual é possível ver o caminho executado próximo de um e a sua dispersão em uma escala de cinza, na qual para cada posição valores próximos de 1 são representados em tons de cinza mais escuros. Fonte: Autor

que ocorrem durante um posicionamento de um jogador.

- Movimentação recalculada, quando o robô inicia o movimento para um destino, mas durante o percurso é redirecionado para outro destino porque alguma coisa mudou no contexto do jogo.

- Movimentação de marcação evitando as áreas de proibidas, como a bola durante o comando de stop do juiz ou a área do goleiro durante todo o jogo.

Tendo os parâmetros das execuções analisadas e os cenários interessantes para serem analisados, os testes executados durante os experimentos deste trabalho foram gerados aleatoriamente seguindo uma distribuição predefinida e foram validados para simular, em execução controlada, o contexto realista de uma partida de SSL.

A seguir são apresentados e discutidos os cenários de testes considerados para a validação e a análise das estratégias pela perspectiva do controle e da atuação resultante na navegação do robô pelo meio.

Para cada cenário foi criado um caso de teste com 100 tarefas de navegação completas, na qual para cada um deles foram aplicadas as três estratégias de navegação analisadas: o grafo de visibilidade VG, as trajetórias Bang-Bang de forma simples TBB e as trajetórias Bang-Bang de estado sincronizado.

4.4.1 Navegação isolada

Neste tipo de cenário de teste, o robô é controlado pelo comportamento de navegação até uma posição específica do campo. Nos testes, o robô deve partir do repouso e chegar no destino determinado, ao atingir um objetivo o robô é desacelerado para o repouso e segue para o novo objetivo estabelecido. Para criar a lista de objetivos dentro do campo, isto é, a distância até o

objetivo no "Start"e no "Change Target", a quantidade de trocas de objetivos, em qual tempo elas ocorrem e qual a sua frequência, foram utilizados os parâmetros de distribuição apresentados na Tabela 1 na geração aleatória de tarefas de navegação e suas possíveis modificações.

Como temos outros robôs para criação de obstáculos e como forma de aumentar a complexidade da navegação, foram estabelecidas três variações dos experimentos de navegação isolada, nas quais a complexidade de executar uma tarefa de navegação vai aumentando gradativamente.

No primeiro experimento, apenas o robô controlado no campo com as áreas dos goleiros como obstáculos estáticos (cena 1). No segundo experimento, os outros 11 robôs parados foram espalhados aleatoriamente pelo campo (cena 2). No último experimento, os outros 11 robôs podem trocar de lugar com outro robô aleatoriamente, se movendo pelo campo para fazer essa troca, criando um cenário de interação com obstáculos dinâmicos que interferem diretamente no caminho possível para o robô, testando a capacidade de navegação segura e sem colisões (cena 3).

4.4.1.1 Movimento Simples

A ideia da execução é mostrada na Figura 25, o robô parte do repouso e chega até seu destino evitando obstáculos passando pela seguinte sequência de estados na máquina de estados de navegação: Start →Path Execution →Finish.

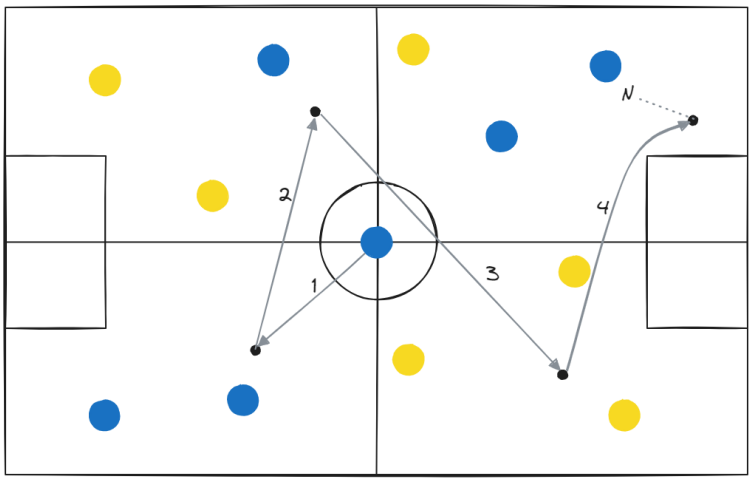


Figura 25: Teste de uma sequência de movimentações completas do robô dentro do campo de SSL, sem que sofrer mudança de objetivo, os círculos azuis representam os robôs do time aliado e os amarelos do time inimigo. Fonte: Autor.

4.4.1.2 Movimentação com Redirecionamento

A ideia da execução é mostrada na Figura 26, o robô parte do repouso e chega no destino determinado passando por N mudanças de objetivo ao longo da trajetória percorrida. O robô passa pela sequência de estados: Start →Path Execution ↔ Change Target ↔ Path Execution →Finish.

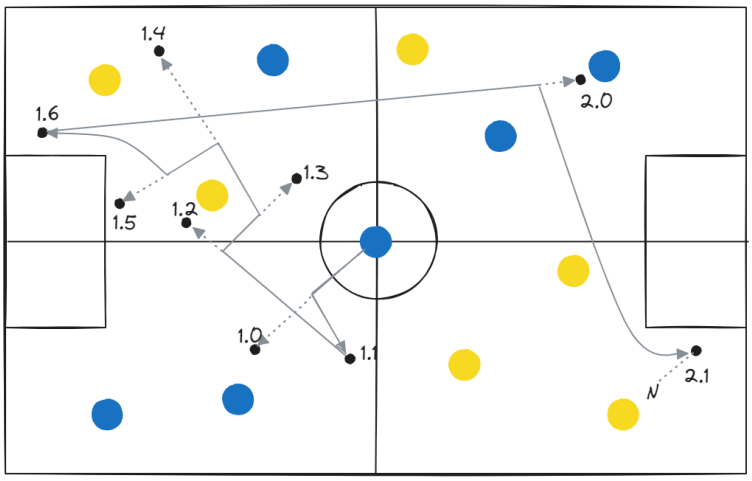


Figura 26: Teste de uma sequência de movimentações completas do robô dentro do campo de SSL, sofrendo mudanças de objetivo no meio da movimentação, os círculos azuis representam os robôs do time aliado e os amarelos do time inimigo. Fonte: Autor.

4.4.2 Marcação em torno da área do goleiro

Neste cenário de teste, são avaliados diretamente a eficiência e a capacidade de navegação segura, sem invadir a área do goleiro, diante de uma condição de estresse, onde jogadores inimigos se movem aleatoriamente no entorno da área gerando perigo para acionar a marcação.

Uma vez que, um robô inimigo se aproxima da área do goleiro, o SSL-Unification[11] entende ele como risco e aloca um robô para marcá-lo e fechar seu ângulo para o gol, evitando riscos de gols. Neste cenário, é importante alcançar a posição de marcação rapidamente, entretanto, sem invadir a área do goleiro ou colidir com outros robôs. Para definir a posição de marcação, o software traça uma linha do centro do robô inimigo para o centro do gol aliado e define um ponto próximo ao robô inimigo sobre a linha.

A ideia da execução é mostrada na Figura 27, neste caso tem-se um ambiente de estresse e espaço reduzido para a navegação onde os robôs inimigos vão se movendo para pontos aleatórios dentro da região de risco, precisando se mover enquanto evitam colisões com os outros robôs e sem invadir a área dos goleiros. A cada ciclo conforme eles se movem, o robô defensor tem sua posição recalculada buscando o novo objetivo em uma combinação de movimentações para os robôs na defesa que minimize a movimentação total, com isso, ao longo das movimentações pode-se ter trocas de pares de robô aliado e inimigo na marcação.

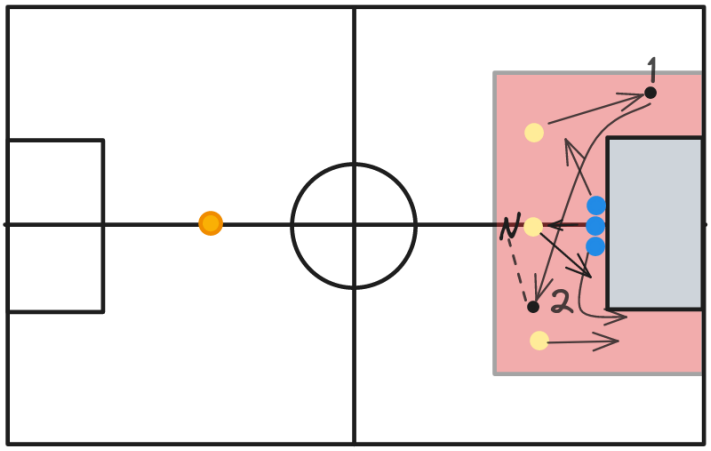


Figura 27: Teste de movimentação de marcação defensiva, o objetivo do robô é bloquear a linha para o gol dos robôs inimigos, reduzindo o risco desse robô fazer gol caso receba a bola. No diagrama os robôs amarelos se movem para posições livres da marcação e os azuis de forma reativa, continuamente buscam manter os robôs inimigos com suas linha para o gol bloqueadas, como demonstram as setas. Fonte: Autor.

5

RESULTADOS

Neste capítulo são apresentado os resultados obtidos com a aplicação da metodologia e casos de testes apresentada no Capítulo 4 considerando as estratégias de navegação apresentadas no Capítulo 3. Ao aplicar o processamento nos logs gerados pelo SSL-Unification como mostrado no fluxo da Figura 21, todos os resultados obtidos foram agrupados nas Tabelas 2, 3 e 4, as interpretações e análises sobre os dados das tabelas serão feitas ao longo do capítulo.

Os fatores relacionados ao desempenho de uma estratégia de navegação dados pelo tempo necessário e a velocidade média são apresentados, enquanto a dispersão (discrepância de planejamento e execução da trajetória) se relaciona a navegação segura e otimizada, podendo ser tolerados alguns erros em prol de desempenho. Esses dados estão na Tabela 2, quando relacionados descrevem inclusive o quanto, proporcionalmente, o robô fez de esforço adicional em relação à distância para chegar ao destino.

Além disso, para relacionar a precisão de uma estratégia de navegação podem ser avaliados os erros que ocorrem durante a execução da tarefa de movimentação, além da própria dispersão, apresentados na Tabela 3.

Por último, para avaliar a segurança na execução da tarefa de navegação de forma mais direta, podem ser avaliadas as quantidade de eventos de invasão e colisões registrados, além da média de distância da invasão ou da velocidade respectivamente, esses dados são apresentados na Tabela 4.

Nos dados de tempo e dispersão encontramos um alto desvio padrão devido ao grande grau de variabilidade do contexto de cada navegação. Mesmo com a padronização de tarefas, o resultado obtido com as trajetórias Bang-Bang não é tem oscilação por ser uma estratégia que envolve amostragem de pontos aleatoriamente, não sendo determinística. Somando a isso, cenários que envolvem a movimentação de outros robôs podem interferir de forma diferente no robô analisado entre as execuções, e por último a grande variação de distâncias e chaveamentos trazem um certo ruído para a análise.

Tabela 2: Resultados principais para avaliar o desempenho, com a média e o desvio padrão, representado por *s*, para cada cenário e estratégias propostas.

Caso de Teste	Estratégia	Dispersão	Velocidade Média (m/s)	Tempo Total (s)
Movimentação Simples (Cena 1)	VG	8.63 s 2.3	0.49 s 0.25	1.49 s 0.70
	TBB	10.71 s 3.64	0.49 s 0.22	1.42 s 0.82
	TBBES	5.71 s 1.3	0.64 s 0.29	1.09 s 0.51
Movimentação Simples (Cena 2)	VG	4.29 s 1.22	0.57 s 0.3	1.77 s 0.9
	TBB	11.06 s 3.66	0.52 s 0.23	1.65 s 0.88
	TBBES	5.63 s 1.06	0.66 s 0.32	1.27 s 0.68
Movimentação Simples (Cena 3)	VG	4.39 s 2.18	0.47 s 0.25	1.93 s 1.24
	TBB	9.13 s 3.09	0.45 s 0.22	1.72 s 1.02
	TBBES	5.21 s 1.55	0.54 s 0.3	1.33 s 0.85
Movimentação Redirecionada (Cena 1)	VG	7.9 s 1.61	0.89 s 0.20	9.15 s 5.9
	TBB	13.5 s 2.06	0.81 s 0.13	11.38 s 8.8
	TBBES	8.96 s 1.1	1.08 s 0.23	7.26 s 5.38
Movimentação Redirecionada (Cena 2)	VG	3.12 s 1.84	0.735 s 0.19	9.88 s 8.81
	TBB	11.94 s 2.39	0.73 s 0.2	12.05 s 11.47
	TBBES	8.0 s 1.1	1.02 s 0.28	7.9 s 6.8
Movimentação Redirecionada (Cena 3)	VG	3.69 s 0.87	0.82 s 0.23	8.21 s 7.57
	TBB	11.94 s 1.91	0.75 s 0.14	11.78 s 10.31
	TBBES	7.79 s 1.21	1.05 s 0.19	8.45 s 6.39
Marcação em Torno da Área do Goleiro	VG	6.84 s 4.45	0.57 s 0.19	3.25 s 2.3
	TBB	15.09 s 7.81	0.41 s 0.18	2.39 s 2.37
	TBBES	12.88 s 8.8	0.48 s 0.22	1.95 s 1.9

5.1 ANÁLISE DE DESEMPENHO

Ao analisar comparativamente a Tabela 2 com os resultados e a Figura 28, vemos que a estratégia TBBES conclui em menos tempo sua movimentação. Comparando com a técnica do VG temos uma redução global media de 15%, desconsiderando como caso fora da curva, o cenário da movimentação redirecionada (cena 3). Na cena 3, onde os outros robôs são obstáculos dinâmicos, a técnica VG foi 2% melhor, por conta da indecisão da trajetória Bang-Bang sobre qual caminho seguir para evitar os obstáculos móveis, esta constatação é reforçada por uma dispersão mais de 100% maior da estratégia TBBES em relação ao VG nesse cenário específico.

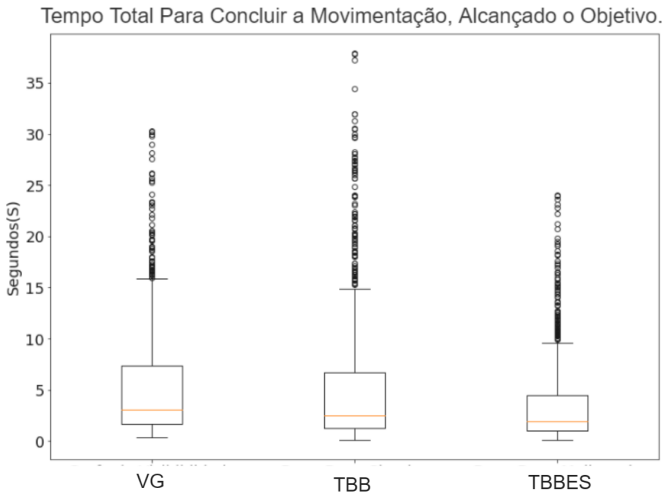


Figura 28: Distribuição do tempo total para a execução de uma tarefa de navegação, contabilizando todos os testes, divididos por estratégia. Fonte:Autor

Um fator que reforça os ganhos de desempenho da técnica TBBES é a velocidade média superior obtida, como mostrada na Figura 29. Enquanto a versão simplificada do TBB apresenta velocidades similares a estratégia do VG, a versão melhorada do TBBES apresentou em média uma melhoria de 25% na velocidade média, desconsiderando como caso fora da curva, o cenário da movimentação em torno da área do goleiro, onde a indecisão de uma trajetória que evitasse os outros robôs e a área do goleiro, influenciou negativamente o TBBES em comparação com os dados obtidos pela técnica VG para a velocidade media aplicada na atuação.

Como apresentado nas discussões acima, uma limitação de modelagem/implementação notável para as estratégias que utilizam as trajetórias Bang-Bang foram as incertezas de caminhos para cenários com obstáculos dinâmicos mais complexos. Nestes cenários, a trajetória Bang-Bang parece ser muito sensível a variações de obstáculos chegando a ter um desempenho similar a técnica VG.

Como a técnica TBB depende de uma entrada do estado atual do robô de forma precisa e que rapidamente responda à aplicação da atuação dos motores para poder calcular a trajetória possível do estado de entrada até o estado destino, percebe-se que utilizar diretamente a visão global de alto-nível do software fornece dados precisos, mas que possuem um atraso de percepção

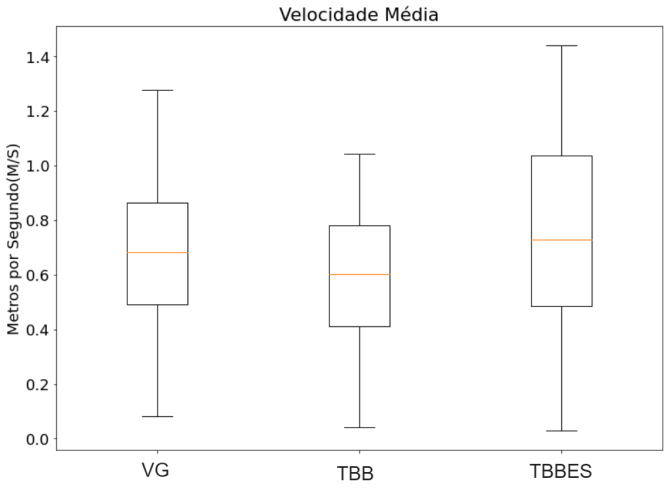


Figura 29: Distribuição da velocidade média durante a execução de uma tarefa de navegação, contabilizando todos os testes, divididos por estratégia. Fonte: Autor.

o que impacta a responsividade do sistema de controle em perceber a resposta do robô a atuação aplicada. Esta característica gera uma aparente inércia para executar mudanças conforme a trajetória planejada, com isso, a dispersão entre as trajetórias Bang-Bang simples TBB, em relação à versão melhorada TBBES é em média 61% maior.

Ainda olhando para a dispersão, é notável que tirando o cenário de movimentação simples sem outros robôs, no qual o TBBES representou uma redução de 33% em relação ao VG, para todos os outros cenários a menor dispersão presente foi a da técnica VG, dado reforçado pela Figura 30. A baixa dispersão é obtida por consequência direta da menor velocidade do robô, o que está refletido na menor velocidade média e um maior tempo para a execução da navegação apresentada. Essas reduções, dependendo da aplicação e inclusive para o contexto de SSL, são mais impactantes negativamente do que a dispersão em si podendo ser toleradas até certo ponto.

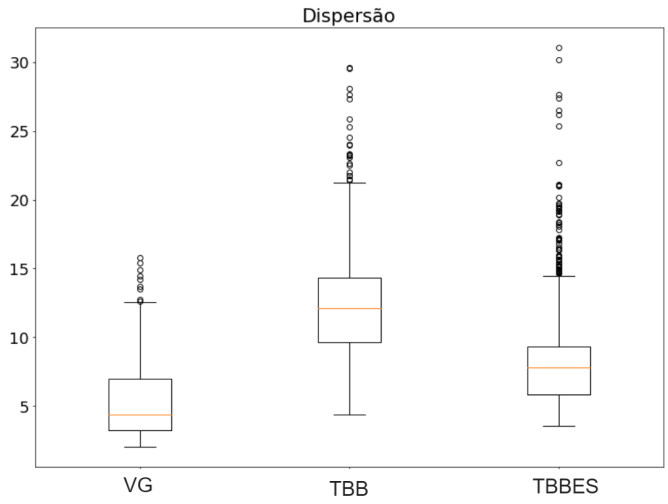


Figura 30: Distribuição da dispersão para a execução das tarefas de navegação, contabilizando todos os testes, divididos por estratégia. Fonte: Autor

Partindo para a comparação entre estratégias é notável como uma versão robusta e mais

Tabela 3: Resultados para avaliar a precisão das estratégias para cada cenário proposto, com os erros de execução médio para a posição e o tempo.

Caso de Teste	Estratégia	RMSE Posição no Tempo (m)	RMSE Tempo Estimado (s)
Movimentação Simples (Cena 1)	TBB	0.042 s 0.01	0.005 s 0.006
	TBBES	0.06 s 0.01	0.007 s 0.009
Movimentação Simples (Cena 2)	TBB	0.047 s 0.016	0.005 s 0.004
	TBBES	0.063 s 0.022	0.01 s 0.014
Movimentação Simples (Cena 3)	TBB	0.043 s 0.014	0.01 s 0.01
	TBBES	0.05 s 0.02	0.016 s 0.021
Movimentação Redirecionada (Cena 1)	TBB	0.07 s 0.01	0.013 s 0.003
	TBBES	0.1 s 0.026	0.016 s 0.007
Movimentação Redirecionada (Cena 2)	TBB	0.062 s 0.018	0.019 s 0.015
	TBBES	0.09 s 0.03	0.02 s 0.01
Movimentação Redirecionada (Cena 3)	TBB	0.064 s 0.012	0.016 s 0.005
	TBBES	0.09 s 0.02	0.02 s 0.01
Marcação em Torno da Área do Goleiro	TBB	0.038 s 0.012	0.03 s 0.01
	TBBES	0.05 s 0.01	0.04 s 0.03

otimizada para o fluxo do software, o VG supera uma implementação simplificada das trajetórias Bang-Bang, TBB. Vemos que os ganhos em redução de tempo nos cenários de movimentação simples, aparecem nos que envolvem redirecionamento, cenário de maior ocorrência. Somando a isso, uma maior dispersão em relação à versão melhorada demonstra a complexidade de integração e a necessidade de um fluxo adequado para a necessidade mais restrita sobre o estado atual, mesmo existindo implementações de código aberto disponíveis.

5.2 ANÁLISE DE PRECISÃO

As duas estratégias que utilizam trajetórias Bang-Bang e que estimam o estados desejado para o robô possibilitaram a análise de erros entre o estado estimado e o estado atual ao longo da execução. Analisando os valores RMSE, é notável que globalmente o TBBES possui um maior erro associado em módulo, tanto para velocidade quanto para a posição. Entretanto, pela diferença de tempo total nas execuções e da dispersão discutidos anteriormente, é possível concluir que, enquanto os erros da técnica TBB tem um viés de atraso para o robô chegar as posições e cumprir o tempo estimado, a técnica TBBES por conta de ruídos do estimador de estados, notável na Figura 19 e 20, a sincronia de estado estimado pela real tem um viés antecipatório que em alguns momentos ultrapassa o atraso do sistema por conta de oscilações.

5.3 EVENTOS DE COLISÃO E INVASÃO

Além dos dados diretamente associados ao desempenho da navegação, durante os testes foram analisadas as invasões de áreas proibidas e as colisões com outros robôs, buscando avaliar a capacidade de navegação segura possibilitada por cada estratégia.

Cenários propensos a invasões foram testados principalmente nos testes de movimentação de marcação no entorno da área do goleiro, sendo obtidos dados suficientes para uma análise mais sólida. Entretanto, eventos de colisão entre os robôs foram raros, aparecendo nos cenários com os outros robôs se movendo, ao contrário dos testes antipodais que foi apresentado por Costa[5], que realmente testam os limites das estratégias para evitar colisões, mas que são mais distantes da realidade de uma partida SSL.

Para ter cenários realistas da quantidade de colisões, provavelmente seriam necessários testes de disputa de bola entre os atacantes dos dois times que possuem uma complexidade alta de analisar por sua alta variabilidade de contextos resultantes de comportamentos complexos e dificuldade de padronizar as execuções.

Tabela 4: Resultados com a soma total e as médias dos dados dos eventos de colisão e invasão ao longo dos testes, divididos por estratégia.

Estratégia	Quantidade de Colisões	Velocidade Média das Colisões (M/S)	Quantidade de Invasões	Tamanho Médias das Invasões (M)
VG	3	1.696	67	0.01363
TBB	3	1.867	28	0.00373
TBBES	0	N/A	70	0.00686

A Tabela 4 apresenta os resultados obtidos dos eventos de colisão e invasão seguindo as regras do juiz. Quanto as colisões é possível ver que não forram registradas colisões utilizando o TBBES, sua responsividade do controle para obstáculos dinâmicos fizeram a diferença, sendo a única estratégia a não registrar eventos desse tipo. Por outro lado, a estratégia que menos invadiu a área do goleiro foi a técnica do TBB dada a quantidade e pelo tamanho médio das invasões, enquanto isso o VG e o TBBES ficaram próximos no entorno de 140% a mais de invasões, a vantagem do TBBES foi que a distância da invasão foi a metade do VG.

Na estratégia VG, as invasões ocorrem porque o robô navega na fronteira dos polígonos dos obstáculos. Dessa forma, erros de correspondência da navegação planejada com a navegação executada levam a invasões. Mesmo possuindo planejamento de trajetórias, a estratégia TBBES apresentou número de invasões similares, entretanto, neste caso o motivo dessas invasões foram os ruídos na estimativa de estado atual, levando o preditor a possuir um viés antecipatório exagerado na navegação prejudicando a conclusão da manobra de desvio dos obstáculos.

As invasões e colisões são um dos parâmetros principais de navegação segura de um robô móvel. No contexto de SSL, existe uma margem pequena para que esses eventos ocorram sem

prejudicar o time. Por conta do ruído de predição na utilizada no TBBES, ele não gerou ganhos em relação ao VG como a estratégia base, o TBB, entretanto, como a versão base apresenta problemas de desempenho mesmo nos casos mais simples, como discutido nas seções anteriores, a estratégia do TBBES permanece preferível. Esse fator limitante no ganho demonstra como a definição do estado atual é crucial e provavelmente o atraso de atuação no tempo é uma variável ao longo do tempo e não uma constante.

6

CONCLUSÃO

No contexto da movimentação de robô de forma autônoma navegar com eficiência é essencial. Ao longo desse trabalho foram detalhadas e analisadas estratégias para a execução concreta de uma tarefa de navegação a partir de uma tarefa abstrata, escopo amplamente relevante pelos desafios envolvidos e a quantidade de combinações de técnicas possíveis.

Como pretendido no início, além do aprofundamento na especificação das estratégias, foram definidos testes, após a captura dos dados forram feitas análises comparativas, que de forma numérica apontaram claramente as vantagens e desvantagens de cada estratégia estudada nos cenários propostos. Os cenários avaliados estão no domínio de uma partida de SSL por serem realistas, mas que são limitados em explorar amplamente características como o desvio ativo de obstáculos das estratégias.

Com os resultados discutidos no Capítulo 5, são notáveis os ganhos envolvidos pela introdução de um sistema de planejamento de trajetórias juntamente com a necessidade de utilizar um preditor de estados para realmente aproveitar suas vantagens, mesmo com o aumento da dispersão, tolerado pelo aumento de velocidade obtido. Assim, com este trabalho foi possível mensurar de forma sistemática os ganhos da técnica TBBES em velocidade e redução do tempo necessário, experimentados pelo RobôCIn durante o último ano, com a utilização dessa estratégia de navegação. Além disso, foi possível estabelecer pontos de melhorias que ainda são um gargalo para extrair mais desempenho da estratégia.

Para trabalhos futuros, é importante um estudo que defina o índice de cobertura de cenários e comportamentos obtidos com os testes propostos, o que possibilitaria entender definitivamente as limitações dos testes feitos, sendo possível ajustar os teste atuais ou incluir novos testes para aumentar o índice de cobertura. Um ponto relevante de melhoria para a robustez das estratégias que envolvem trajetórias Bang-Bang é a minimização da incerteza ao definir o caminho, demonstrada em alguns cenários. Além disso, a versão da estratégia com melhor desempenho apresentada foca apenas em melhorar a sincronia prévia ao módulo de planejamento. Para um ambiente real com mais ruídos e distorção entre o planejado e o executado seria importante um processamento para garantir que o robô execute realmente a atuação planejada, que sincronizasse esses dois estados ativamente minimizando os erros. Adicionalmente, seria interessante replicar os testes no ambiente real fazendo um comparativo

com a execução simulada.

REFERÊNCIAS

[1] (2024). *Open Source Framework*. Robotics Erlangen e.V. Team.

[2] (2024). *Sumatra - Central AI of TIGERs Mannheim*. Robotics TIGERs Mannheim Team.

[3] Alatise, M. B. & Hancke, G. P. (2020). A review on challenges of autonomous mobile robot and sensor fusion methods. *IEEE Access*, 8:39830–39846.

[4] Balkcom, D. J., Kavathekar, P. A., & Mason, M. T. (2006). Time-optimal trajectories for an omni-directional vehicle. *The International Journal of Robotics Research*, 25(10):985–999.

[5] Costa, L. d. S. & Tonidandel, F. (2023). Análise de técnicas de navegação de robôs autônomos em ambientes dinâmicos e incertos. Dissertação de mestrado, Centro Universitário FEI, São Bernardo do Campo. https://repositorio.fei.edu.br/handle/FEI/5194.

[6] Gasparetto, A., Boscariol, P., Lanzutti, A., & Vidoni, R. (2015). *Path Planning and Trajectory Planning Algorithms: A General Overview*, 3–27. Springer International Publishing, Cham.

[7] Hussain, R. & Zeadally, S. (2019). Autonomous cars: Research results, issues, and future challenges. *IEEE Communications Surveys Tutorials*, 21(2):1275–1313.

[8] Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., & Osawa, E. (1997). Robocup: The robot world cup initiative. In *Proceedings of the First International Conference on Autonomous Agents*, 340–347.

[9] Lynch, K. M. & Park, F. C. (2017). *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, USA, 1st edition.

[10] Magel, L. (2016). Development of an autonomous referee software for the small size league. Robocup Small Size League, Mannheim, Germany, 2016.

[11] Oliveira, A., Gomes, C., Silva, C., Alves, C. M., Souza, D., Xavier, D., Silva, E., Martins, F., Cavalcanti, L. H., Maciel, L., Paixão, M., Vasconcelos, M., Vinícius, M., Melo, J. G., Moura, J. P., Silva, J. R., Cruz, J. V., Santana, P. H., Oliveira, P. P., Rodrigues, R., Fernandes, R., Morais, R., Teobaldo, T., Silva, W., & Barros, E. (2023). Robôcin small size league extended team description paper for robocup 2023. Robocup Small Size League, Recife, Brazil, 2023.

[12] Ommer, N., Ryll, A., & Geiger, M. (2019). Tigers mannheim (team interacting and game evolving robots) extended team description for robocup 2019. RoboCup Small Size League, Mannheim, Germany, 2019.

[13] Purwin, O. & D'Andrea, R. (2005). Trajectory generation for four wheeled omnidirectional vehicles. In *Proceedings of the 2005, American Control Conference, 2005.*, 4979–4984 vol. 7.

[14] RoboCup (2024). *Laws of the RoboCup Small Size League 2023*. Small Size League Technical Committee.

[15] Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011). Autonomous mobile robots. *A Bradford Book*.

[16] Silva, C., Martins, F., Machado, J. G., Cavalcanti, L., Sousa, R., Fernandes, R., Araújo, V., Silva, V., Barros, E., Bassani, H. F., de Mattos Neto, P. S. G., & Ren, T. I. (2019). Robôcin 2019 team description paper. Robocup Small Size League, Recife, Brazil, 2019.

[17] Velasco-Villa, M., del Muro-Cuellar, B., & Alvarez-Aguirre, A. (2007). Smith-predictor compensator for a delayed omnidirectional mobile robot. In *2007 Mediterranean Conference on Control Automation*, 1–6.

[18] Weitzenfeld, A., Biswas, J., Akar, M., & Sukvichai, K. (2015). Robocup small-size league: Past, present and future. In Bianchi, R. A. C., Akin, H. L., Ramamoorthy, S., & Sugiura, K., editors, *RoboCup 2014: Robot World Cup XVIII*, 611–623.

[19] Wendler, A. & Heineken, T. (2020). Er-force 2020 extended team description paper. RoboCup Small Size League, Erlangen, Germany, 2020.

[20] Yang, S. & Baum, M. (2017). Extended kalman filter for extended object tracking. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4386–4390.

[21] Zickler, S., Laue, T., Birbach, O., Wongphati, M., & Veloso, M. (2010). Ssl-vision: The shared vision system for the robocup small size league. In Baltes, J., Lagoudakis, M. G., Naruse, T., & Ghidary, S. S., editors, *RoboCup 2009: Robot Soccer World Cup XIII*, 425–436.