UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

RENAN MOREIRA LÔBO

NOVAS ESTRATÉGIAS DE PLANEJAMENTO NO FIRN: UM FRAMEWORK PARA NAVEGAÇÃO INTELIGENTE DE ROBÔS

RIO DE JANEIRO

2012

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

RENAN MOREIRA LÔBO

NOVAS ESTRATÉGIAS DE PLANEJAMENTO NO FIRN: UM FRAMEWORK PARA NAVEGAÇÃO INTELIGENTE DE ROBÔS

Dissertação de Mestrado submetida ao Corpo Docente do Departamento de Ciência da Computação do Instituto de Matemática, e Instituto Tércio Pacitti (iNCE) da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para a obtenção do título de Mestre em Informática.

Orientadores: Prof. Dr. Josefino Cabral Melo Lima

Prof. Dr. Adriano Joaquim de Oliveira Cruz

RIO DE JANEIRO

2012

L799 Lôbo, Renan Moreira.
Novas estratégias de planejamento no FIRN: um framework para navegação inteligente de robôs. / Renan Moreira Lôbo. -- 2012.
174 f.: il.

Dissertação (Mestrado em Informática) – Universidade Federal do Rio de Janeiro, Instituto de Matemática, Núcleo de Computação Eletrônica, 2012.

Orientadore: Josefino Cabral Lima
Co-orientador Adriano Joaquim de Oliveira Cruz
1. Navegação Inteligente de Robôs. 2. Previsão de Tragetórias. 3.Algoritmos Bioinspirados 4. Particle Swarm Optimization – Teses. I. Lima, Josefino Cabral (Orient.). II. Cruz, Adriano Joaquiim de Oliveira. (Co-orient.). III Universidade Federal do Rio de Janeiro, Instituto de Matemática, InstitutoTércio Pacitti. III Título.

CDD

Dedico esta dissertação aos meus pais, aos meus grandes amigos que me acompanharam em toda minha vida e a Deus.

AGRADECIMENTOS

Meus agradecimentos pelo resultado deste trabalho e também por todo o período de desenvolvimento vão para os meus pais, que me apoiaram nas horas mais difíceis e me lembraram sempre daquilo que eu era capaz de fazer; para os meus amigos que indicaram caminhos e alternativas que me levaram a esta versão final, assim como participaram de minhas horas de lazer e descanso, sendo estas tão importantes quanto as horas de trabalho; aos meus orientadores, Prof. Dr. Josefino Cabral MELO LIMA e Prof. Dr. Adriano Joaquim de Oliveira CRUZ pelas sugestões, acompanhamentos, direções e também pelas palavras amigas nas horas de necessidade; à banca avaliadora, professores Max Suell Dutra, José Francisco Moreira Pessanha, Carla Amor Divino Moreira Delgado e Antônio Carlos Gay Thomé pela disponibilidade e sugestões durante a defesa.

RESUMO

LOBO, Renan Moreira. **Novas estratégias de planejamento no FIRN: um framework para navegação inteligente de robôs**. 2012, 174f. Dissertação (Mestrado em Informática) – Instituto de Matemática, Instituto Tércio Pacitti, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2012

A crescente aplicação de robôs em diversos tipos de tarefas requer que eles otimizem seus deslocamentos e minimizem colisões. Em ambientes dinâmicos, povoados por obstáculos que se movem, planejar uma trajetória segura e otimizada tem sido objeto de importantes pesquisas em robótica. Uma forma de tratar esse problema é predizer os deslocamento dos obstáculos móveis e calcular uma rota livre de confrontos, estimando as melhores direções a serem tomadas. Em muitos ambientes dinâmicos os obstáculos móveis não se movem aleatoriamente: eles seguem padrões específicos de movimento. Deslocamentos de carros nas ruas, deslocamentos de grupos de pedestres em uma cidade, deslocamentos de humanos dentro de um prédio, são exemplos de "obstáculos móveis" que seguem basicamente padrões específicos de movimentos, assim como as partículas suspensas em um fluido em que não haja turbulência ou dispersão significativa. Neste contexto, foi desenvolvido, pelo grupo de pesquisa MASI, o FIRN (*Framework for Intelligent Robot Navigation*), para navegação de um robô capaz de predizer o comportamento de um obstáculo móvel e em seguida planejar uma trajetória segura para o seu deslocamento. O FIRN é aplicável a ambientes simulados e trata com padrões determinísticos. Esta dissertação apresenta uma série de otimizações para o FIRN, capacitando-o para tratar com vários obstáculos móveis com deslocamento simultâneo. Essas otimizações, que incluem novos métodos de planejamento, englobam também contribuições advindas de *swarm intelligence*. As experimentações efetuadas comparando o FIRN com a navegação controlada por APF (*Artificial Potential Fields*) mostraram uma taxa de colisões 14.9% menor.

Palavras-chave: Navegação Inteligente de Robôs, Previsão de Trajetórias, Algoritmos Bioinspirados.

ABSTRACT

LOBO, Renan Moreira. **New planning strategies on FIRN: a framework for intelligent robot navigation**. 2012, 174f. Dissertação (Mestrado em Informática) – Instituto de Matemática, Instituto Tércio Pacitti, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2012

The increasing application of robots in several tasks requires them to optimize their path and minimize collisions. In dynamic environments, populated by moving obstacles, planning a safe and optimized trajectory has been the subject of important researches on robotics. One way to handle this problem is to predict the trajectories of the moving obstacles and to calculate a path free of collision, computing the best directions to take. In many dynamic environments moving obstacles do not move randomly: they follow specific movement patterns. Car traffic on streets, pedestrian traffic in a city, human traffic inside a building are all examples of "moving obstacles" that basically follow specific movement patterns, as well as particles in fluids where there is no significant turbulence or eddy diffusion In this context, the research group MASI developed FIRN, a Framework for Intelligent Robot Navigation, in which a robot is able to predict the behavior of moving obstacles and then to plan its safe trajectory. FIRN is suitable to simulated environments and it deals with deterministic patterns. This dissertation presents a series of enhancements to the FIRN, including the development of new planning methods based on swarm intelligence techniques. Conducted experiments comparing FIRN to an APF (Artificial Potential Fields) controlled navigation have shown that FIRN attains 14.9% less collisions.

Keywords: Intelligent Robot Navigation, Trajectory Prediction, Bio-inspired Algorithms.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABC	*Artificial Bee Colony Algorithm* (Algoritmo da Colônia de Abelhas)
ACO	*Ant Colony Optimization* (Otimização da Colônia de Formigas)
ANNPM	*Artificial Neural Networks Prediction Method*
APF	*Artificial Potential Field*
AprioriFRN	*Apriori For Robot Navigation* (Apriori para Navegação de Robôs)
DSM	*Direct Search Method*
E-LOFORM	*Enhanced LOoking FOward Robot Motions*
FIRN	*Framework for Intelligent Robot Navigation* (Framework para Navegação Inteligente de Robôs)
IR	*Intelligent Robot* (Robô Inteligente)
LADSM	*Look Ahead Direct Search Method*
LOFORM	*LOoking FOward Robot Motions*
MASI	Modelos e Arquiteturas para Sistemas Inteligentes
MO	*Mobile Obstacle* (Obstáculo Móvel)
POMDP	*Partially Observable Markov Decision Process* (Processo de Decisão de Markov Parcialmente Observável)
PNN	*Polynomial Neural Network* (Rede Neural Polinomial)
PPGI	Programa de Pós-Graduação em Informática
PSOSM	*Particle Swarm Optimization Search Method*
RBPD	*Rao-Blackwellised Particle Filter*
UFRJ	Universidade Federal do Rio de Janeiro
UGV	*Unmaned Ground Vehicle* (Veículo Terrestre Autônomo)
VGM	*Visibility Graph Methods* (Métodos com Grafos de Visibilidade)

SUMÁRIO

1 INTRODUÇÃO

Os avanços tecnológicos conquistados recentemente com o desenvolvimento de componentes cada vez menores, com maior capacidade de processamento, e de sensores de alta precisão estão permitindo a aplicação de robôs em tarefas diversas, tais como limpeza autônoma de cômodos residenciais, exploração de ambientes inacessíveis a humanos, suporte a vítimas de acidentes (CANGELOSO, 2011; WEBSTER, 2011; HORNYAK, 2011) etc. A robótica vem então se destacando no meio acadêmico e se consolidando como um campo multidisciplinar de pesquisa, envolvendo também a biologia, a psicologia e a medicina (BEER et al., 1998; ITOH; KATO; ITOH, 2009; MARTEL, 2008).

Ainda que as aplicações sejam bastante variadas, um problema comum a todas elas é a necessidade do robô se locomover. Seja na execução de tarefas manuais em uma indústria ou na interação social, é preciso que o robô possa planejar e realizar seus deslocamentos de forma a evitar obstáculos e otimizar seu caminho. Para resolver este problema diferentes métodos utilizando ferramentas distintas foram desenvolvidos. É possível classificar estes métodos de acordo com:

- o tipo de estratégia utilizada (reativa, deliberativa ou híbrida) (STONE; VELOSO, 2000);
- o tipo de algoritmo aplicado (clássico ou heurístico) (SAFFARI; MAHJOOB, 2009);
- a disposição dos obstáculos no ambiente de navegação (estáticos ou dinâmicos);

- a percepção que o robô tem do ambiente (visão via câmeras e sensores ou representação interna do ambiente através de mapas);

Quando se trabalha com ambientes dinâmicos não é possível contar com uma representação completa do local *a priori* devido aos obstáculos móveis. Isto dificulta a navegação do robô e limita as opções de métodos aplicáveis: métodos que se utilizam de estratégias deliberativas (onde o robô consulta a configuração do ambiente e calcula rotas otimizadas até o objetivo) necessitam de atualizações constantes do mapa do ambiente e são muito onerosos do ponto de vista computacional, já que a rota até o objetivo tem de ser atualizada constantemente. Nesses ambientes é mais comum a aplicação de métodos com estratégias reativas (onde o robô se move em direção ao objetivo e evita os obstáculos localmente, quando eles vão sendo identificados pelos sensores) ou de métodos com estratégias híbridas (onde um planejamento é realizado tomando por base a configuração estática do ambiente e medidas de emergência são adotadas apenas na iminência de colisões).

Com a crescente aplicação de ambientes de computação ubíqua, onde câmeras fornecem imagens do ambiente a todo momento e conexões com terminais de computação permitem processamento em qualquer local, a etapa de planejamento das estratégias híbridas tem a chance de lidar melhor com os obstáculos dinâmicos. Por exemplo, através da coleta de dados estatísticos do comportamento dos obstáculos no ambiente é possível estimar a rota de um obstáculo específico. Além disso, com a constante atualização da visualização do ambiente, o planejamento pode ser recalculado apenas quando os obstáculos dinâmicos interferem na rota planejada.

Tendo em mente este tipo de ambiente de aplicação, o grupo MASI do PPGI vem desenvolvendo há aproximadamente três anos, o FIRN, um *Framework* para Navegação Inteligente de Robôs, capaz de conduzir um robô em um ambiente dinâmico simulado onde os obstáculos se movem tão ou mais rapidamente que ele próprio. O FIRN é constituído de quatro etapas: uma de coleta de amostras, onde as posições ocupadas por obstáculos no ambiente em diferentes momentos são coletadas; uma de previsão que, a partir dos dados coletados, é capaz de estimar o deslocamento futuro dos obstáculos; uma de planejamento, onde uma rota que evite as trajetórias estimadas dos obstáculos é construída para o robô;

e uma de navegação, onde o robô se desloca de acordo com a trajetória planejada. Uma série de métodos podem ser aplicados nas etapas de previsão e planejamento, contemplando algoritmos baseados em Redes Neurais, Lógica Nebulosa e uma estratégia utilizada originalmente no contexto de Mineração de Dados para reconhecimento de padrões, como será explicado no Capítulo 3. O FIRN difere de estratégias aplicadas em outras pesquisas por calcular a previsão em tempo real e utilizar o resultado imediatamente no planejamento.

O FIRN é resultado de um desenvolvimento contínuo dentro do MASI. Foram finalizadas as seguintes tarefas pelo grupo antes da pesquisa que culminou com esta dissertação:

- Elaboração preliminar do *framework*, com definição do ambiente de aplicação, objetivo, restrições e etapas contempladas;
- Desenvolvimento dos métodos de previsão ANNPM (*Artificial Neural Network Precition Method*), baseado em Redes Neurais, LOFORM (*LOking FOrward Robot Motions*) e E-LOFORM (*Enhanced-LOoking FOrward Robot Motions*), ambos baseados no algoritmo AprioriAll, utilizado originalmente no contexto de Mineração de Dados;
- Desenvolvimento dos métodos de planejamento DSM (*Direct Search Method*), que utiliza uma heurística gulosa para calcular as trajetórias e FDSM (*Fuzzy Direct Search Method*), que utiliza controladores nebulosos para tal;
- Desenvolvimento de um simulador capaz de lidar com um obstáculo, chamado Simulador FIRNp;
- Experimentos preliminares relativos aos métodos de previsão no Simulador FIRNp;
- Experimentos preliminares relativos aos métodos de planejamento no Simulador FIRNp;

Os resultados obtidos no Simulador FIRNp foram muito favoráveis, incentivando um investimento adicional no *framework*. No entanto, é necessário considerar que um ambiente com apenas um obstáculo dificilmente apresentaria grandes desafios para o robô. Apenas com a extensão do simulador para o contexto de vários obstáculos é que os tempos de previsão e planejamento, a taxa de colisões e a qualidade do caminho retornado pelo planejamento

poderiam ser avaliados criteriosamente. O objetivo da pesquisa associada a esta dissertação foi efetuar esta extensão, com uma avaliação do comportamento dos métodos de previsão e planejamento disponíveis no contexto de vários obstáculos, e propor novos métodos de planejamento para resolver os problemas identificados nos métodos DSM e FDSM. Também foi feita uma comparação do FIRN com uma estratégia puramente reativa, evidenciando as vantagens da etapa de previsão na navegação. Os seguintes desenvolvimentos foram realizados dentro do escopo desta dissertação:

- A construção de um novo simulador capaz de confrontar o robô contra vários obstáculos simultaneamente, chamado de Simulador FIRNn;
- A extensão dos experimentos contemplando a etapa de previsão, desta vez considerando um ambiente com vários obstáculos;
- A realização de experimentos avaliando de forma mais criteriosa os métodos de planejamento disponíveis originalmente no Simulador FIRNp e o desempenho geral do FIRN aplicado em um contexto com vários obstáculos;
- A comparação direta entre o FIRN e estratégias puramente reativas utilizando APFs (*Artificial Potential Fields*);
- O desenvolvimento de novos métodos de planejamento (LADSM - *Look Ahead Direct Search Method*, PSOSMv1 - *Particle Swarm Optimization Search Method* versão 1 e PSOSMv2 - *Particle Swarm Optimization Search Method* versão 2), a fim de resolver as deficiências dos já existentes;
- A realização de experimentos avaliando os novos métodos de planejamento;

1.1 Motivação

Em muitos ambientes dinâmicos o comportamento dos obstáculos é imprevisível. É possível, no entanto, que em certas condições exista um padrão observável de movimento. Em ambientes fechados, como escritórios ou corredores, agentes humanos se deslocam com

intenção, delineando trajetórias em linha reta entre pontos objetivos (por exemplo, o caminho entre uma porta e um assento). Nas ruas, veículos se movimentam limitados às vias de tráfego, realizando curvas em pontos específicos e manobrando de acordo com as normas de trânsito locais. Em ambos os casos podem ocorrer desvios a fim de que colisões sejam evitadas, mas estes acontecem esporadicamente dependendo do congestionamento nos ambientes. Em oleodutos, macro-partículas suspensas em um fluído podem possuir comportamento identificável, dependendo das condições de viscosidade, densidade e pressão do fluído, além da região onde as macro-partículas viajam (próximas à parede do tubo, no centro da turbulência ou na camada de transição) (ESCOBEDO; MANSOORI, 2010). O mesmo pode ser dito para partículas metálicas presentes em gasodutos, onde modelos matemáticos são capazes de projetar estas trajetórias (AMARNATH et al., 1999; SAHU; AMARNATH, 2010). Um robô para inspeção e manutenção nesses tipos de ambientes, por exemplo, poderia obter informações relativas ao movimento de partículas grandes o suficiente para comprometer sua estrutura, podendo então calcular trajetórias mais otimizadas.

O FIRN (*Framework for Intelligent Robot Navigation*) é um *framework* para navegação de robôs que, partindo do princípio que os deslocamentos de obstáculos móveis não são completamente aleatórios, pode prever um conjunto de posições futuras desses obstáculos e utilizar esta informação a fim de calcular uma rota para o robô até o seu objetivo. O FIRN é aplicável em locais onde os obstáculos podem ser identificados com precisão em uma área limitada em volta de um robô. Inicialmente, o robô se desloca em direção a um objetivo sem se preocupar com obstáculos móveis e, a partir do momento que estes são identificados pelos sensores ou câmeras, o FIRN é iniciado. São quatro as etapas aplicadas no *framework*:

- Coleta de Amostras: o movimento dos obstáculos móveis é observado e amostras das posições ocupadas em instantes de tempo igualmente espaçados são coletadas;
- Previsão: As amostras coletadas na etapa anterior são processadas por um método de previsão, o que resulta em um conjunto de posições futuras estimadas para cada obstáculo;
- Planejamento: O conjunto de posições resultante da etapa anterior é enviado para um método de planejamento que calcula uma rota livre de colisões, considerando a posição

inicial e final do robô e as posições ocupadas pelos obstáculos a cada instante de tempo futuro;

- Navegação: A rota resultante da etapa de planejamento é transferida para o robô, que deve segui-la a fim de alcançar o objetivo sem colisão;

O *framework* deve ser aplicado sempre que um novo conjunto de obstáculos móveis é identificado, e o processo se repete até que o objetivo final seja alcançado. Dada a dificuldade encontrada em ambientes do tipo labirinto, onde existe a necessidade de localização e rastreamento do objetivo e o robô pode se prender em becos sem saída ou andar em ciclos, o FIRN não lida com estes ambientes.

A fim de avaliar a corretude dos métodos de previsão e planejamento desenvolvidos e o desempenho do FIRN em um ambiente simples, o Simulador FIRNp foi construído pelo grupo MASI. Este simulador contempla a situação onde o robô se encontra em uma posição inicial e um obstáculo móvel é identificado no ambiente. O *framework* deve calcular a rota prevista do obstáculo e fornecer uma rota sem colisões para o robô até um ponto objetivo. Experimentos revelaram que o FIRN consegue reproduzir as posições futuras de obstáculos se movimentando de acordo com padrões simples em tempos da ordem de nanossegundos e que a navegação é realizada com sucesso em até 94.10% dos casos. Ainda que o resultado tenha sido positivo, alguns fatores devem ser considerados: quando o robô é confrontado por apenas um obstáculo, o espaço para manobras no ambiente é grande, diminuindo a probabilidade de colisão; o tempo de processamento da previsão e do planejamento para um obstáculo não influencia negativamente a etapa de navegação, mas com o acréscimo de novos obstáculos este tempo aumenta e pode comprometê-la; nenhum experimento comparando o FIRN com uma estratégia reativa foi realizado anteriormente pelo grupo MASI, o que poderia melhor esclarecer as contribuições trazidas pela etapa de previsão.

1.2 Objetivo

O primeiro objetivo nesta pesquisa foi o de avaliar o desempenho dos métodos de previsão e planejamento desenvolvidos originalmente no Simulador FIRNp em um ambiente com vários obstáculos. Um novo simulador, chamado Simulador FIRNn, foi construído para

lidar com este tipo de situação, possibilitando ainda a aplicação de estratégias reativas comparativas. Dos experimentos realizados neste novo simulador foi possível aferir: o tempo de processamento do método de previsão E-LOFORM (considerado o de melhor desempenho no Simulador FIRNp) no contexto de vários obstáculos, o desempenho e o tempo de processamento dos métodos de planejamento DSM e FDSM também no contexto de vários obstáculos e o desempenho geral do *framework* proposto em relação a duas estratégias reativas baseadas em APFs.

Com os resultados obtidos na primeira rodada de experimentos ficou clara a deficiência do FIRN em ambientes congestionados, principalmente com relação aos métodos reativos de comparação. Como o tempo de processamento tomado na previsão não apresentou impacto significativo na navegação e o resultado da mesma foi correto, a deficiência foi atribuída aos métodos de planejamento. Dentre os problemas identificados, aqueles considerados mais graves foram a subutilização da previsão retornada pela etapa anterior, as limitações impostas pelos métodos na escolha de direções aplicáveis para o robô e a incapacidade dos mesmos de regular a velocidade de navegação.

A partir das deficiências identificadas, novos métodos de planejamento foram elaborados e implementados, sendo este o principal objetivo da dissertação. O método LADSM foi desenvolvido com o intuito de extrair informações relevantes da previsão e aplicá-las de forma inteligente no planejamento do caminho; o método PSOSMv1 foi desenvolvido a fim de flexibilizar as direções e velocidades disponíveis para o robô durante o planejamento, eliminando as restrições impostas pelos métodos DSM e FDSM; o método PSOSMv2 foi desenvolvido para identificar áreas congestionadas por obstáculos na previsão e distanciar o robô das mesmas durante o planejamento.

Com os novos métodos de planejamento desenvolvidos, uma segunda rodada de experimentos foi realizada para avaliar o impacto exercido por estes no FIRN. Os métodos LADSM, PSOSMv1 e PSOSMv2 foram comparados aos métodos DSM e FDSM, no que se refere ao tempo de processamento, número de colisões e qualidade do caminho, e a estratégia E-LOFORM + PSOSMv2 às estratégias reativas de comparação baseadas em APFs.

1.3 Escopo

Como o FIRN é aplicado a ambientes simulados na pesquisa associada a esta dissertação, não são discutidos aspectos físicos reais da movimentação de corpos, admitindo-se que eles se movem de acordo com padrões determinísticos, sem colisão entre si ou influências externas. Também não é abordada nesta dissertação a transição do modelo simulado para o mundo real. As métricas de qualidade e desempenho são todas baseadas na comparação com as estratégias reativas.

Como o objetivo principal é estender o simulador para ambientes com vários obstáculos e propor novos métodos de planejamento, foi optado por deixar alguns desenvolvimentos para o futuro, como:

- Desenvolvimento da etapa reativa na navegação: Para que seja classificada como uma estratégia de navegação híbrida, o FIRN necessita evitar colisões reativamente caso o planejamento falhe ou o robô encontre alguma situação não prevista. Como o objetivo é verificar o desempenho das estratégias de previsão e planejamento, não é necessário o desenvolvimento da etapa reativa do FIRN neste momento;

- Introdução de novos obstáculos no ambiente: Ainda que seja uma situação comum no mundo real, a introdução de novos obstáculos no ambiente de simulação não será realizada neste momento. Os obstáculos em cada cenário de simulação são somente aqueles definidos no início;

- Reavaliação da previsão: Dado que a previsão não é 100% correta, talvez seja necessário reavaliar a previsão em sua totalidade ou parcialmente caso seja detectado que os obstáculos não se movem da forma prevista. Como este é um detalhe pertinente à etapa de Previsão do FIRN, não foi trabalhado neste momento;

Nas simulações descritas, são feitas as seguintes suposições:

- O robô simulado é considerado omnidirecional, com capacidade de se mover livremente em qualquer direção. Ele não possui massa, é circular com 5 unidades de medida de

raio e não sofre de qualquer força ou influência física real. Os obstáculos possuem as mesmas características e se movem sob as mesmas condições;

- A velocidade máxima do robô é sempre igual ou menor que a velocidade dos obstáculos, o que destaca as contribuições que a etapa de previsão traz para a navegação, além de ser um diferencial desta pesquisa com relação a outras abordagens, que normalmente supõem que a velocidade dos obstáculos é menor;

- Por se tratar de um ambiente simulado, as posições dos obstáculos e do robô são representadas por coordenadas bidimensionais referentes à área de simulação. Não foi resolvido qualquer problema referente a tratamento de imagens para aferir a posição dos mesmos no ambiente;

- O campo de simulação representa a área coberta pelos sensores em comunicação com o robô. Desta forma, a posição dos obstáculos é conhecida e exata e a coleta de amostras é feita imediatamente ao início da simulação;

- Nas simulações, os objetos atualizam suas posições a cada rodada como se estivessem se movendo continuamente pelo ambiente com suas respectivas velocidades. Isto implica que o movimento no simulador é representado de forma discreta e qualquer problema referente à conversão de posições no simulador em movimentos para um robô do mundo real não foi abordado;

- Existe uma distância segura entre o robô e os obstáculos no início da simulação, de forma que colisões só podem ocorrer depois da etapa de coleta de amostras do *framework*. É necessário definir o tempo e o número de amostras suficientes para que os métodos de previsão funcionem corretamente e como estes parâmetros estão intimamente correlacionados com o ambiente de aplicação no mundo real, o cálculo destes deve ser realizado na ocasião da aplicação do FIRN em um robô físico;

- A velocidade dos obstáculos não é fornecida diretamente pelos sensores e é descoberta através do processamento dos métodos de previsão;

1.4 Contribuição

É esperado que as funcionalidades básicas do FIRN estejam completas com performance equivalente ou melhor do que estratégias puramente reativas para o ambiente de simulação, propondo o FIRN como uma nova estratégia de navegação a ser explorada. Considerando que os obstáculos realmente possuam um padrão de movimento identificável e que a previsão seja suficientemente correta, espera-se um número de colisões menor do que o de uma estratégia puramente reativa aplicada no mesmo ambiente.

Esta documentação formal sobre o FIRN poderá ser usada como base para diversos trabalhos futuros. As possibilidades são muitas: transferência do *framework* para o mundo real; extensão do simulador, de forma que o robô possa detectar novos obstáculos e reprocessar a previsão e o planejamento; simulação física real das partículas, a fim de aproximar o modelo à realidade; concepção de novos métodos de previsão e planejamento utilizando outros algoritmos.

1.5 Estrutura da Dissertação

A estrutura da dissertação é a seguinte:

- No Capítulo 2 é apresentada uma discussão mais detalhada sobre navegação de robôs em ambientes dinâmicos, bem como trabalhos relacionados ao tema aqui desenvolvido;
- No Capítulo 3 são discutidos os aspectos gerais do FIRN, com uma descrição do *framework* e o detalhamento dos métodos de previsão e planejamento desenvolvidos em momento anterior à dissertação;
- No Capítulo 4 são apresentados os dois simuladores desenvolvidos para o FIRN: o Simulador FIRNp para um obstáculo, onde os primeiros experimentos contemplando a etapa de previsão realizados anteriormente são abordados, e o Simulador FIRNn para vários obstáculos, com experimentos contemplando o desempenho do FIRN em um contexto com vários obstáculos e relativo a estratégias de navegação puramente reativas;

- No Capítulo 5, os novos métodos de planejamento são apresentados, explicitando as deficiências resolvidas por cada um, e experimentos avaliando o impacto das mudanças são realizados;
- No Capítulo 6 é delineada uma discussão geral sobre a dissertação, com uma conclusão sobre o FIRN, a aplicabilidade do mesmo no mundo real, e trabalhos futuros relacionados;
- No Apêndice A estão todos os algoritmos utilizados pelos métodos de previsão e planejamento do FIRN;
- No Apêndice B, os esquemas utilizados para gerar cenários de simulação nos experimentos;
- No Apêndice C, as instruções gerais de como utilizar o Simulador FIRNn para desenvolver novos métodos de previsão e planejamento.

2 ESTRATÉGIAS DE NAVEGAÇÃO EM AMBIENTES DINÂMICOS

2.1 Estado da Arte

Os tipos de estratégias disponíveis para navegação são: deliberativas, reativas e híbridas. Ainda que existam interpretações diversas a cerca de seus significados, a definição de STONE e VELOSO (2000) parece ser a mais apropriada: estratégias deliberativas são aquelas onde o agente inteligente possui recursos para ponderar sobre diversas possibilidades, manter estados internos e calcular os efeitos de suas ações; estratégias reativas são aquelas onde o agente inteligente toma suas decisões baseado em reflexos do ambiente, sem considerar os efeitos posteriores; estratégias híbridas são aquelas que reúnem características dos dois tipos. Existem outras definições comparativas entre estratégias deliberativas e reativas: agentes deliberativos utilizam um conhecimento prévio do ambiente para calcular rotas globais, enquanto que agentes reativos tomam decisões em tempo real, interpretando a leitura dos sensores (ZHANG; LIU; YANG, 2006); agentes deliberativos processam camadas hierárquicas de algoritmos, onde as camadas mais altas são responsáveis pelo ambiente completo e as mais baixas pelos arredores (ARKIN; MACKENZIE, ????); agentes reativos se movimentam sob a influência de campos de atração e repulsão no ambiente (JULIÁ et al., 2010).

Duas definições formais para ambientes dinâmicos são: ambientes com obstáculos em constante movimento (como pessoas, animais ou outros robôs) e ambientes que podem ter sua configuração alterada com o tempo (portas que se fecham, passagens que não estão mais disponíveis, etc). Ambientes do primeiro tipo serão chamados de “ambientes dinâmicos

ativos" (JARADAT; AL-ROUSAN; QUADAN, 2011; SINGH; PARHI, 2009) e os do segundo tipo de "ambientes dinâmicos reconfiguráveis" (MENG; XUEDONG, 2009; XU et al., 2007). Esta nomenclatura é utilizada nesta dissertação apenas com o intuito de eliminar a ambiguidade; nos trabalhos citados os ambientes são chamados apenas de dinâmicos.

Sob certa perspectiva, ambientes dinâmicos reconfiguráveis podem ser tratados como ambientes estáticos que devem ser reavaliados de tempos em tempos. Nestes casos, ainda é possível para o agente inteligente ter um mapa completo do ambiente à sua disposição. No entanto, quando se fala em ambientes dinâmicos ativos, este tipo de mapeamento se torna complicado e custoso, já que a configuração do espaço se altera em espaços muito curtos de tempo. Dada a necessidade de uma resposta rápida, as estratégias reativas e híbridas são mais apropriadas já que as evasões são realizadas apenas quando os obstáculos são identificados pelos sensores e o processamento é naturalmente rápido. Ambientes monitorados com integração via rede fornecem uma visão constante do ambiente e possibilitam que alguns tipos de estratégias deliberativas sejam aplicadas mesmo quando obstáculos dinâmicos ativos estão presentes, mas ainda assim é necessário velocidade no processamento. Este tipo de ambiente será cada vez mais comum com o advento da computação ubíqua, onde ambientes integrados por redes e monitorados permitem o processamento em qualquer lugar de forma descomplicada (WEISS; CRAIGER, 2002; KIM; KIM; LEE, 2004).

Exemplos de métodos empregados em estratégias reativas são: APFs (*Artificial Potential Fields*), que simulam campos potenciais em volta de obstáculos e objetivos, influenciando na navegação do robô (KHATIB, 1986); Sistemas de Decisão baseados em comportamento onde, dependendo do contexto do ambiente, o robô pode executar uma série de ações baseadas em seu comportamento ativo (evasão de obstáculos, *wall-following*, busca pelo objetivo, etc) (BROOKS, 1986); Lógica Nebulosa, um modelo capaz de representar o nível de incerteza natural do mundo real e do pensamento humano, utilizada tanto para controlar aspectos relativos à navegação quanto para executá-la diretamente (ZADEH, 1965).

Na etapa de planejamento global de sistemas híbridos, são comuns a aplicação de: VGMs (*Visibility Graph Methods*), onde o mapa do ambiente é dividido em vértices classificados como livres ou ocupados (com obstáculos) (NILSSON, 1969) seguido da aplicação do Algoritmo A* (HART; NILSSON; RAPHAEL, 1968); Redes Neurais Artificiais, um modelo

matemático baseado em neurônios biológicos que, através de um treinamento adequado com exemplos de aplicação, é capaz de generalizar sua resposta para outras situações no mesmo contexto (ROSENBLATT, 1958).

Uma das deficiências das estratégias reativas é a qualidade final do caminho. Como elas têm a evasão por prioridade, as rotas projetadas nem sempre são ótimas. Como mostra a Figura 2.1, o caminho construído por uma estratégia deliberativa possui ângulos mais suaves de direção que o caminho gerado pela estratégia reativa. As estratégias híbridas atenuam esta deficiência, já que a rota é planejada globalmente e sofre influências mínimas, mas necessárias, dos mecanismos de reação.

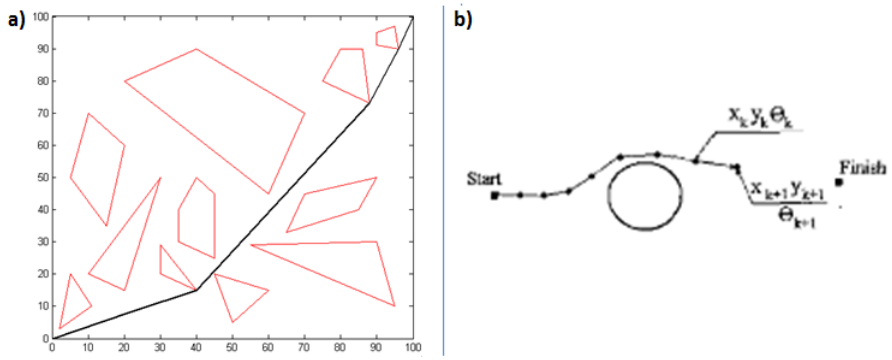

Figura 2.1: Resultado de uma estratégia: a) deliberativa; b) reativa. Figuras reproduzidas de SAFFARI e MAHJOOB (2009) e SFEIR, KANAAN e SAAD (2004).

2.2 Trabalhos Relacionados

A principal meta considerada durante o desenvolvimento do FIRN e desta dissertação foi a elaboração de uma estratégia de navegação em ambientes dinâmicos ativos, auxiliada por um mecanismo de aprendizagem capaz de gerar rotas otimizadas sem colisões. Nesta Seção são apresentados alguns trabalhos contemplando objetivos semelhantes, assim como pesquisas relacionadas aos métodos escolhidos para comparação, aqueles baseados na aplicação de APFs na navegação de robôs. Para uma visão geral do problema de planejamento de caminhos, SARIFF e BUNIYAMIN (2006) traçam uma linha do tempo: no período entre os anos 1980 e 1989, o Algoritmo A* era utilizado extensivamente no planejamento de rotas

otimizadas em representações distintas do ambiente, como por exemplo, aquelas geradas pela aplicação de VGMs; entre os anos 1990 e 1999 houve a ascensão dos algoritmos probabilísticos, como os Algoritmos Genéticos, e também daqueles baseados em APFs; no período entre 2000 e 2005 houve o surgimento de versões modificadas do Algoritmo A* que consideram heurísticas baseadas em dados coletados diretamente de ambientes reais ao invés daquelas derivadas de representações aproximadas dos mesmos, ou que agilizam a busca através do reaproveitamento de informações obtidas previamente.

Nas pesquisas que contemplam ambientes dinâmicos reconfiguráveis é comum a aplicação de estratégias híbridas, com uma fase global de planejamento que é reavaliada quando barreiras imprevistas são detectadas pelos sensores. Por exemplo, XU et al. (2007) realizam a projeção de um ambiente inicialmente conhecido em um *gridmap* 2D e empregam o Algoritmo A* em um primeiro momento para calcular a melhor rota. O trajeto entre os pontos inicial e final tem vários objetivos parciais que devem ser alcançados pelo robô e a heurística utilizada é a distância em linha reta entre um vértice e o objetivo. Quando um obstáculo não representado no mapa inicial é encontrado pelo robô, ao invés de reavaliar todo o caminho reprocessando o A*, ele utiliza um método proposto chamado de D* Lite que considera apenas os nós do grafo entre dois objetivos parciais adjacentes onde tenha ocorrido alteração no ambiente, agilizando a busca. No FIRN o ambiente não precisa ser projetado em um grafo ou qualquer outro tipo de representação alternativa. No entanto, considerando-se que as câmeras estarão espalhadas pelo ambiente, é necessário realizar uma transformação do eixo de visualização das câmeras para um plano ortogonal ao ambiente.

Em MENG e XUEDONG (2009) o mapeamento do ambiente é efetuado com o auxílio de agentes humanos. Durante a etapa de treinamento, um robô no mundo real realiza leituras do ambiente com seus sensores e grava a rota de navegação ao ser guiado pelo agente. Incorreções são contornadas através de ajustes na imprecisão dos sensores de ultrassom e câmera disponíveis. Durante a navegação autônoma o robô marca objetivos parciais no mapa e calcula a rota otimizada para trafegar entre eles, baseando-se no aprendizado realizado na etapa anterior. Quando os sensores registram que um dos objetivos parciais está obstruído por um obstáculo desconhecido o planejamento local é ativado para a evasão. O artigo não considera, no entanto, o movimento destes obstáculos, sugerindo que o ambiente é dinâmico

reconfigurável. No FIRN, a única manipulação necessária por agentes humanos é a seleção dos métodos de previsão e planejamento desejados e o ajuste dos parâmetros associados.

KIM et al. (2007) mostram uma alternativa para navegação de um robô no mundo real com uma câmera acoplada, que recebe imagens do ambiente à sua frente, e sensores de ultrassom. Para processar os obstáculos através das câmeras, é realizado um processamento de imagem capaz de definir as bordas dos objetos com relação ao fundo através da comparação do brilho entre os pixels limitantes (Detector de Bordas de Canny (CANNY, 1986)). O sistema é dividido em três camadas de atuação: quando os obstáculos estão muito distantes, uma câmera acoplada no robô identifica as bordas dos objetos e o robô processa o limite dos *pixels* que separam obstáculos do cenário, utilizando esta informação para visualizar o ambiente e calcular sua rota; quando numa distância de 2 metros, o robô calcula o centro do obstáculo com seus sensores de ultrassom (que cobre toda sua parte frontal) e determina o centro dos feixes, delineando uma barreira circular que deve ser evitada; quando o obstáculo é identificado muito próximo ou mais de um é encontrado simultaneamente, o robô calcula a área livre disponível, ou seja, a área que não recebe resposta do sensor de ultrassom e altera sua rotação até que consiga passar sem esbarrar seus limites nos obstáculos. Experimentos foram realizados em ambientes estáticos, mas o sistema pode ser estendido para ambientes dinâmicos reconfiguráveis sem perda de generalidade.

VOKRINEK, KOMENDA e PECHOUCEK (2010) apresentam uma solução para um grupo de robôs navegar em um ambiente *outdoor* simulado representando um vilarejo. O grupo conta com duas representações do mapa utilizadas pelo planejamento global, mas não é possível aferir a condição das estradas, que funcionam como obstáculos dinâmicos reconfiguráveis. O grupo utiliza um comboio de UGVs (*Unmanned Ground Vehicles*) mais velozes para encontrar estes obstáculos. Eles averiguam fronteiras marcadas no mapa global e marcam os pontos onde obstáculos desconhecidos são encontrados. Os experimentos mostram que um comboio de cinco UGVs melhora o tempo de navegação e otimiza a trajetória, já que o grupo principal não se prende em "becos sem saída".

No caso de ambientes dinâmicos ativos, onde se encaixa o FIRN, as pesquisas normalmente lidam com o aspecto reativo da navegação, demonstrando métodos de rápido processamento ou consulta para situações de risco. JARADAT, AL-ROUSAN e QUADAN

(2011) utilizam um método Q-Learning para navegar um robô em um ambiente dinâmico ativo simulado. Q-Learning é uma estratégia de Aprendizado de Máquina cujo objetivo é determinar o melhor conjunto de ações dada uma situação percebida no ambiente. Inicialmente, o robô não tem um comportamento bem definido mas através de um sistema de recompensas e penalidades consegue construir um modelo funcional e adaptá-lo para diversos ambientes de aplicação. O Q-Learning utiliza uma tabela como ferramenta, onde as linhas representam os estados possíveis percebidos no ambiente e as colunas as ações possíveis de serem tomadas. Os valores nas células da tabela são iniciados em zero e o robô escolhe, em um primeiro momento, as ações baseado em tentativa e erro. Se o resultado de uma ação aplicada a um estado for positivo, uma recompensa é mapeada na célula correspondente. Se o resultado for negativo, uma penalidade é associada à referida célula. Com o tempo, o robô é capaz de determinar as melhores ações para cada estado, alterando seu comportamento em tempo real através deste sistema de recompensas/penalidades. A contribuição de JARADAT, AL-ROUSAN e QUADAN (2011) se encontra no mapeamento eficiente do ambiente dinâmico em estados e ações finitas: os estados são relativos à presença de obstáculos e do objetivo em oito regiões angulares em volta do robô; as ações são o movimento em linha reta até o objetivo, um giro para a esquerda ou um giro para a direita com um determinado ângulo. Experimentos realizados mostraram que o robô foi capaz de navegar com índices de colisão variando entre 2% e 24.4% para 1 até 13 obstáculos dinâmicos ativos. No entanto, é assumido que a velocidade do robô é sempre maior ou igual a dos obstáculos. No FIRN, a suposição é exatamente a contrária: os obstáculos podem ter velocidade igual ou maior que o robô.

SINGH e PARHI (2009) utilizam uma Rede Neural de quatro camadas para navegar um robô em um ambiente dinâmico ativo. A rede recebe como entrada a leitura de sensores instalados nos lados e na dianteira do robô, assim como o ângulo da direção estimada ao objetivo. A saída é o ângulo de rotação do robô para o próximo passo. A rede é treinada com 200 casos que contemplam situações de evasão, rastreamento do objetivo e fuga de situações de *deadlock* (onde o robô fica preso atrás de um obstáculo tentando alcançar o objetivo sem sucesso). O resultado mostra que o robô desenvolve um comportamento de exploração pelas paredes, contornando os obstáculos de forma a não se prender indefinidamente.

TAN, ZHU e YANG (2009) apresentam um sistema contendo um controlador Nebuloso complementado por um módulo A/B (*acceleration/brake* - aceleração/freio). O sistema Nebuloso é utilizado para evitar obstáculos estáticos e direcionar o robô até o objetivo, tendo como entrada a distância ao obstáculo mais próximo com relação à dianteira, esquerda e direita do robô, a direção onde se encontra o objetivo e a velocidade instantânea do robô; e como saída a aceleração nas rodas esquerda e direita. O módulo A/B é responsável pelos obstáculos dinâmicos e, de acordo com a direção e velocidade dos mesmos, ajusta a saída do controlador Nebuloso de forma a evitar a colisão. Algoritmos Genéticos são utilizados para modificar as variáveis de entrada em tempo real, de acordo com o *feedback* recebido do ambiente. Os controladores nebulosos aplicados na etapa de planejamento do FIRN controlam apenas a direção do movimento, reagindo de acordo com a configuração prevista dos obstáculos no ambiente.

As soluções para ambientes dinâmicos reais apresentadas acima necessitam de uma fase de treinamento antes que o robô possa efetuar sua navegação autônoma. Uma opção que não necessita de treinamento prévio são as APFs. Elas foram sugeridas por KHATIB (1986) e figuram como uma das soluções reativas mais utilizadas em qualquer ambiente de aplicação. Para aplicá-las, o ambiente deve ser mapeado em um espaço operacional referenciável por coordenadas. Em seguida, campos de atração e repulsão são projetados, respectivamente, ao redor dos obstáculos e objetivos. Aplicando o gradiente negativo das combinações dos campos virtuais mapeados no ambiente, obtém-se a direção e a intensidade da força de deslocamento aplicada no robô. É necessário então que os campos sejam funções contínuas e diferenciáveis em todos os pontos. É importante também que o campo de repulsão tenha um limite de atuação, sendo ativado apenas quando o robô se aproxima de uma certa distância.

As APFs são largamente utilizadas devido à sua baixa complexidade computacional e implementação simples. No entanto, elas podem levar o robô a situações de "*deadlock*", onde ele se prende indefinidamente em um obstáculo por estar em um ponto de força resultante zero. Muitos dos trabalhos relacionados se concentram na modelagem de novas funções capazes de evitar estes mínimos (WARREN, 1989; AGIRREBEITIA et al., 2005; VOLPE; KHOSLA, 1990) ou então na aplicação das APFs em outros problemas (LEE; KROVI, 2006; JULIÁ et al., 2008).

KHATIB (1986) cita que as APFs podem ser estendidas sem retrabalho para ambientes dinâmicos reais, contanto que a localização dos obstáculos seja conhecida em qualquer instante de tempo. Existe um esforço em aprimorar as APFs originais para este tipo de ambiente, a fim de que seu desempenho seja otimizado. Por exemplo, VADAKKEPAT, TAN e MING-LIANG (2000) sugerem o uso de uma APF evolutiva, onde as forças de repulsão possuem variáveis que podem ser modificadas por um algoritmo genético. Desta forma, as forças de repulsão podem ser avaliadas de acordo com o tamanho dos obstáculos. A partir de um treinamento em um ambiente específico, estas variáveis regidas pelo algoritmo genético permitem que a fórmula de geração dos campos seja mais simples, fazendo com que casos de mínimos comuns sejam mais facilmente identificados e tradados, agilizando o tempo de reação. Os resultados mostram que o método funciona para ambientes dinâmicos reais, contendo inclusive objetivos móveis, e o caminho gerado é de maior qualidade que as APFs convencionais. No entanto, a aplicação de algoritmos genéticos aumenta a complexidade do método, o que é contrário à simplicidade natural das APFs.

Com o objetivo de lidar diretamente com ambientes dinâmicos reais, GE e CUI (2002) modelam suas funções de atração e repulsão levando em consideração a velocidade dos obstáculos e objetivo em relação ao robô. Quando entra no campo potencial de um obstáculo, o robô considera também a velocidade relativa do mesmo, de forma que possa realizar ajustes finos em sua própria velocidade. Assim, quando percebe que o obstáculo vem em sua direção, o robô regula sua velocidade e trajetória de forma que evite o ponto de colisão. O mesmo é dito para o objetivo, mas em uma relação de atração. Cada componente dos campos é potencializada por uma constante que deve ser estimada. Quando o valor da componente referente à velocidade é zero, o método se comporta como uma APF convencional. São realizadas simulações considerando um robô omnidirecional, com o objetivo de testar os melhores parâmetros para o modelo, e experimentos reais em um robô de duas rodas motorizadas. Os resultados mostram que o robô consegue alcançar o objetivo nas duas situações. O modelo é estendido por YIN e YIN (2008), adicionando também uma componente de aceleração que melhora o desempenho de robôs físicos que necessitam desacelerar para alcançar o objetivo. Dentre os problemas citados estão os mínimos comuns que devem ser tratados separada-

mente e a necessidade de regular as componentes que potencializam os campos de acordo com o ambiente e dimensões dos obstáculos.

Com relação à previsão do movimento de obstáculos, as pesquisas normalmente são voltadas a ambientes congestionados com alto tráfego de pessoas (conferências, escritórios, locais públicos). Em sua maioria, o ambiente é monitorado por câmeras, amostras de pessoas se movimentando são tomadas e algoritmos de *clusterização* utilizados para definir as rotas existentes entre objetivos, que podem ser entradas e saídas, computadores, assentos, terminais informativos, obras de arte e demais pontos de interesse encontrados nos ambientes de estudo. Métodos estatísticos são então utilizados para inferir a trajetória tomada por um humano trafegando no ambiente com base no banco construído pelos algoritmos de *clusterização*, atualizando a previsão sempre que desvios inesperados são projetados.

FOKA e TRAHANIAS (2010), por exemplo, propõem o uso de POMDP (*Partially Observable Markov Decision Process*) para navegar um robô em um ambiente fechado, conhecido e com tráfego de pessoas. O sistema parte do pressuposto que humanos se movem com alguma intenção, nunca aleatoriamente. Desta forma, eles se movem da forma mais direta e otimizada possível para alcançar seu objetivo. Inicialmente, é realizado um treinamento *offline* para mapear o ambiente estático e construir um *gridmap*. Neste *gridmap* estão marcados todos os obstáculos estáticos, e objetivos podem ser apontados. O mapa é acessado a cada passo do robô e um sistema de penalidade/recompensa é utilizado para determinar a próxima célula a ser ocupada. Para não colidir com os obstáculos móveis, o sistema utiliza um sistema de inferência a curto e longo prazo: no primeiro tipo, o robô utiliza uma PNN (*Polynomial Neural Network*) treinada com o auxílio de um algoritmo genético para determinar o próximo passo imediato do obstáculo; no segundo, o robô utiliza pontos de interesse marcados no *gridmap* e, de acordo com o campo de visão do obstáculo, estima seu objetivo e a trajetória em linha reta que o levará até lá. Ambos os tipos de inferência são agregados ao modelo POMDP e determinarão penalidades no *gridmap* de acordo com a previsão que o robô faz das posições futuras dos obstáculos. Os resultados mostram que o sistema é capaz de realizar desvios, ajustar sua velocidade ou replanejar completamente a rota caso uma colisão futura seja identificada. Além disso, o sistema de previsão melhora a qualidade do caminho final em relação ao mesmo sistema utilizando apenas o POMDP.

QIAN et al. (2010) utilizam um sistema semelhante ao de FOKA e TRAHANIAS (2010) para controlar um robô socialmente aceitável em ambientes com pessoas. Existe também um *gridmap* com um sistema de recompensas/penalidades, assim como um mecanismo de previsão a curto e longo prazo. Neste caso, é utilizado um algoritmo proposto (RBPD, *Rao-Blackwellised Particle Filter*) para mapear as trajetórias (identificadas por câmeras) dos agentes humanos no ambiente como partículas (quanto mais partículas, maior a probabilidade de uma pessoa ocupar aquele *grid*). Um processo de Gauss-Markov (RASMUSSEN; WILLIAMS, 2006) suaviza os agrupamentos de partículas, de forma que representem caminhos contínuos, e uma clusterização *fuzzy* (*k-means*) agrupa os pontos em trajetórias distintas. Para a previsão em longo prazo, o robô analisa o campo de visão dos agentes humanos e calcula a trajetória mais provável das disponíveis no *gridmap* e, para a previsão em curto prazo, analisa as últimas velocidade e direções tomadas para inferir as próximas posições. Para determinar os pesos em cada célula do *gridmap*, o robô analisa a previsão a curto e longo prazo e um sistema de interação social baseado no seguintes aspectos: proxêmica, onde o robô deve manter uma distância confortável para os humanos; visibilidade, onde o robô deve estar sempre no campo de visão dos humanos, para não surpreendê-los ao aparecer subitamente; manutenção do caminho, para que o robô não altere sua direção subitamente; prioridade para os humanos, para que o robô sempre ceda sua vez em passagens estreitas ou portais. Os resultados mostram que o sistema de inferência social funciona, ocorrendo situações onde o robô aguarda a passagem de humanos em locais de difícil tráfego, não se esconde atrás de pilastras ou paredes enquanto espera a passagem (ficando sempre visível) e desvia sua trajetória em uma única direção ao detectar um agente humano, evitando manobras desnecessárias. Entretanto, é mostrado que o resultado da previsão a longo prazo pode diminuir de maneira proporcional à quantidade de trajetórias mapeadas no *gridmap* e o número de pessoas trafegando no local. Em condições extremas, a taxa de acerto ficou em apenas 41.9%.

SASAKI, BRSCIC e HASHIMOTO (2010) utilizam as informações fornecidas por um ambiente monitorado por câmeras para aprender trajetórias efetuadas por pessoas, modelando um grafo com nós representando pontos de interesse e arestas representando as transições entre eles. As arestas recebem pesos de acordo com a relevância do trajeto considerado.

É necessário um sistema de câmeras capazes de se comunicarem com um servidor para analisar o comportamento das pessoas e processar as rotas. O algoritmo de Dijkstra é utilizado para calcular a melhor trajetória entre dois pontos de interesse. Ainda que seu foco não seja a evasão de obstáculos dinâmicos, é possível inferir o caminho mais provável que uma pessoa seguiria, dados os pesos atribuídos às arestas. Desta forma, o robô consegue projetar uma rota alternativa com baixo índice de colisão em um ambiente dinâmico ativo.

Numa pesquisa de base, TSUBOUCHI et al. (1992) apresentam uma estratégia de navegação para um ambiente congestionado com previsão iterativa e planejamento. A previsão faz uma projeção de cada obstáculo considerando que ele andará em linha reta com velocidade constante e é reavaliada constantemente. Quando os obstáculos realizam curvas suaves, a projeção em linha reta pode ser considerada verdadeira, contanto que a previsão seja reavaliada em espaços curtos de tempo. Esta projeção é mapeada em um espaço WS x t tridimensional, onde WS é a área de simulação bidimensional e t representa a passagem do tempo. Considerando-se que os obstáculos andarão em linha reta, a projeção dos mesmos são cilindros oblíquos em WS x t. O robô então é capaz de calcular uma rota entre o ponto inicial e o objetivo, ligando retas tangentes a áreas de segurança envolvendo os obstáculos. A estratégia é repetida em espaços de tempo pré-definidos, tomando como ponto inicial a posição atual do robô e recalculando a previsão e planejamento. Isto faz com que, mesmo que os obstáculos não estejam andando em linha reta, a previsão esteja correta por um curto espaço de tempo, o suficiente para que não haja colisão. Nas simulações, TSUBOUCHI et al. (1992) consideram que os obstáculos não realizam curvas muito acentuadas e o robô possui a velocidade operacional máxima no ambiente. A pesquisa é estendida com experimentos exaustivos no ambiente de aplicação (TSUBOUCHI; ANIMOTO, 1994) e com a extensão das capacidade de planejamento do robô, sendo este capaz de projetar rotas circulares entre os pontos de interesse, além daquelas em linha reta (TSUBOUCHI; KURAMOCHI; ANIMOTO, 1995). Para o FIRN, os obstáculos podem projetar curvas acentuadas e a velocidade dos mesmos é igual ou superior à do robô inteligente.

Por ser um *framework*, o FIRN permite a aplicação de diversos mecanismos de previsão na respectiva etapa. Diferente das abordagens de FOKA e TRAHANIAS (2010), QIAN et al. (2010) e SASAKI, BRSCIC e HASHIMOTO (2010), os métodos atualmente aplicáveis

calculam a previsão em tempo real, fornecendo o resultado que é imediatamente utilizado pelo método de planejamento. Com relação à pesquisa de TSUBOUCHI et al. (1992), o FIRN é capaz de prever não só a direção geral do movimento, como também o padrão de deslocamento dos obstáculos.

3 DESCRIÇÃO GERAL DO *FRAMEWORK*

O FIRN é um *framework* para navegação de robôs em ambientes dinâmicos ativos, onde os obstáculos se movem de acordo com um padrão de movimento resultante de uma influência externa ou com comportamento esperado. Um padrão de movimento é caracterizado por uma fração representativa que, quando extraída do movimento de um obstáculo dinâmico e reproduzida repetidamente, é capaz de gerar o movimento completo do mesmo. Por exemplo, na Figura 3.1, uma partícula qualquer se move em um ambiente projetando ondas. Como as ondas possuem a mesma amplitude e comprimento, as informações extraídas de uma única amostra poderiam ser utilizadas para gerar a continuação da rota.

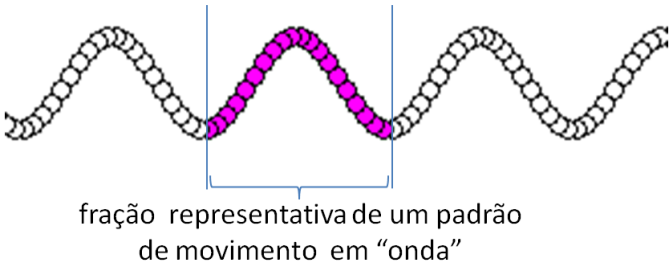

Figura 3.1: Fração representativa de um padrão

Para que o FIRN possa ser aplicado em um robô, este deve ter meios de acessar o ambiente e coletar amostras de posições dos obstáculos. Este conhecimento pode ser tomado a partir de câmeras e sensores acoplados ao robô ou presentes no ambiente. Dada a complexidade de alguns algoritmos aplicados no *framework*, a capacidade limitada de processamento individual de grande parte dos modelos de robôs disponíveis e a necessidade

de agilidade na execução, o cenário ideal de aplicação do FIRN é aquele no qual câmeras e sensores fornecem uma visão do ambiente a um computador remoto, que processa os métodos e envia direções para o robô.

No que se segue nessa dissertação, o robô que navega em um ambiente de acordo com o FIRN será chamado de IR (*Intelligent Robot*, Robô Inteligente) e os obstáculos de MOs (*Moving Obstacles*, Obstáculos Móveis).

Nas simulações abordadas nesta dissertação, o ambiente é visto como se por uma câmera perpendicular, que pode enviar informações relativas à posição do IR e dos MOs para uma estação remota. Inicialmente, o IR se movimenta em direção ao objetivo, sem se preocupar com os MOs e, a partir do momento que um grupo destes é identificado, o FIRN é inciado. A Figura 3.2 ilustra o cerne do FIRN, com as quatro etapas bem definidas:

- Etapa de Coleta de Amostras: São coletadas amostras de posições dos obstáculos identificados pela câmera (Figura 3.2.a). Considerando que um número n de MOs foram identificados e T_c é o tempo disponível para esta etapa, as posições $p_{moi}(t)$ são registradas para cada $t \in T_c$ e armazenadas em listas mo_{lpi}, onde $1 \leq i \leq n$ e t são instantes de tempo tomados em intervalos igualmente espaçados. Cada posição $p_{moi}(t)$ é representada por coordenadas bidimensionais relativas ao ambiente registrado pela câmera e as listas mo_{lpi} representam a trajetória identificada de um MO_i em termos de posições coletadas. As listas mo_{lpi} têm o formato descrito na Fórmula (3.1);

- Etapa de Previsão: Um método de previsão é responsável por processar as listas mo_{lpi} de amostras de cada MO_i, o que envolve a extração de informações relevantes das posições registradas e a aplicação de um algoritmo de aprendizado (Figura 3.2.b). O algoritmo deve identificar o padrão de movimento associado a cada MO_i e, a partir deste, construir as trajetórias mo_{tri}, que consistem de um conjunto de posições esperadas para cada MO_i no futuro;

- Etapa de Planejamento: Um método de planejamento deve acessar as listas mo_{tri} de trajetórias esperadas para os MOs e calcular uma rota para o IR do ponto inicial ao objetivo (Figura 3.2.c). As informações derivadas das listas mo_{tri} dependem do método aplicado, mas podem contemplar as direções tomadas pelos MOs, a velocidade

dos mesmos, locais onde existe grande concentração de MOs, áreas livres de MOs, etc. O resultado desta etapa é uma lista ir_{tr}, contendo a trajetória que o IR deve tomar para alcançar o objetivo sem colisões;

- Etapa de Navegação: O IR se desloca pelo ambiente de acordo com a trajetória descrita em ir_{tr} (Figura 3.2.d). Ainda que este desenvolvimento não seja abordado nesta dissertação, vale salientar que neste momento os mecanismos reativos de evasão e replanejamento na identificação de novos obstáculos devem ser ativados;

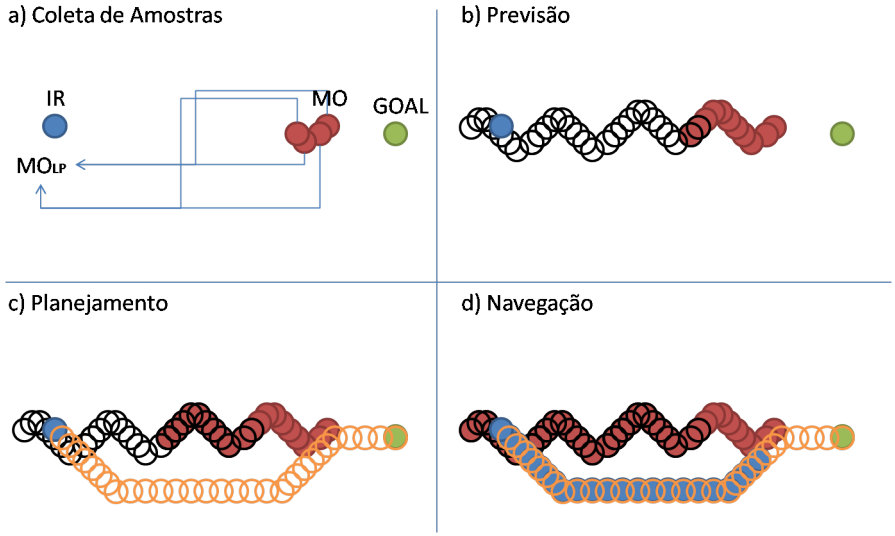

Figura 3.2: Etapas do FIRN

$$mo_{lpi} = \{(p_{moi}(t)|\forall t \in T_c\} = \{(x_{it}, y_{it})|\forall t \in T_c\} \tag{3.1}$$

Para as etapas de previsão e planejamento, existe uma série de métodos que podem ser aplicados. As únicas restrições em relação a isso são as entradas e saídas: os métodos de previsão devem estar prontos para receber as listas de posições mo_{lpi} dos obstáculos e gerar as listas de trajetórias previstas mo_{tri}; os métodos de planejamento devem estar prontos para receber a saída dos métodos de previsão e gerar a lista com a trajetória prevista ir_{tr} do IR. Os métodos originalmente desenvolvidos para o FIRN são enumerados no que se segue, começando pelos métodos de previsão. Todos os métodos descritos neste Capítulo

foram desenvolvidos pelo grupo MASI em momento anterior à pesquisa associada a esta dissertação.

3.1 Métodos de Previsão

Dada a importância da previsão para o funcionamento do FIRN, a primeira preocupação do grupo MASI foi desenvolver e aplicar métodos para esta etapa em um ambiente simulado. Todos os métodos possuem uma etapa de pré-processamento, onde as informações relevantes são derivadas de mo_{lpi}, como o vetor de velocidade ou a direção tomada entre duas posições consecutivas. A partir desta, o padrão de movimento é identificado e as posições futuras esperadas são derivadas aplicando-se o próximo passo do padrão encontrado na última posição registrada em mo_{lpi}. Considerando-se que o padrão identificado seja correto, este processo deve gerar a trajetória esperada. Está claro que no mundo real pode ocorrer a presença de ruídos oriundos de condições adversas nas amostras tomadas dos MOs, e neste caso, os métodos de previsão devem ser capazes de minimizar estas ocorrências.

A trajetória mo_{tri} resultante desta etapa contém as posições esperadas de MO_i para cada $tf \in T_f$, onde tf são instantes de tempo futuro em intervalos igualmente espaçados e T_f é o tempo total previsto pelo método de previsão, onde cada conjunto do tipo $mo_{tr1}(tf), mo_{tr2}(tf), ..., mo_{trn}(tf)$ é uma configuração estática para o ambiente no instante tf. A Figura 3.3 mostra um exemplo: existem três listas mo_{tr}, cada uma com T_f configurações estáticas para um MO.

Foram desenvolvidos três métodos: ANNPM (*Artificial Neural Network Prediction Method*), onde Redes Neurais são treinadas para, a partir de um conjunto anterior de vetores de velocidade tomados por um MO, retornar o próximo vetor esperado; LOFORM (*LOoking FOward for Robot Motions*), que toma as direções identificadas entre as posições da amostra de um MO, discretiza as mesmas em um conjunto finito de possibilidades e encontra sequências de direções frequentes através da aplicação do algoritmo AprioriFRN, baseado no algoritmo AprioriAll utilizado no campo de Mineração de Dados; E-LOFORM (*Enhanced LOoking FOward for Robot Motions*), que identifica os vetores de velocidade tomados entre duas posições respectivas de um MO, calcula a velocidade máxima estimada a partir deste

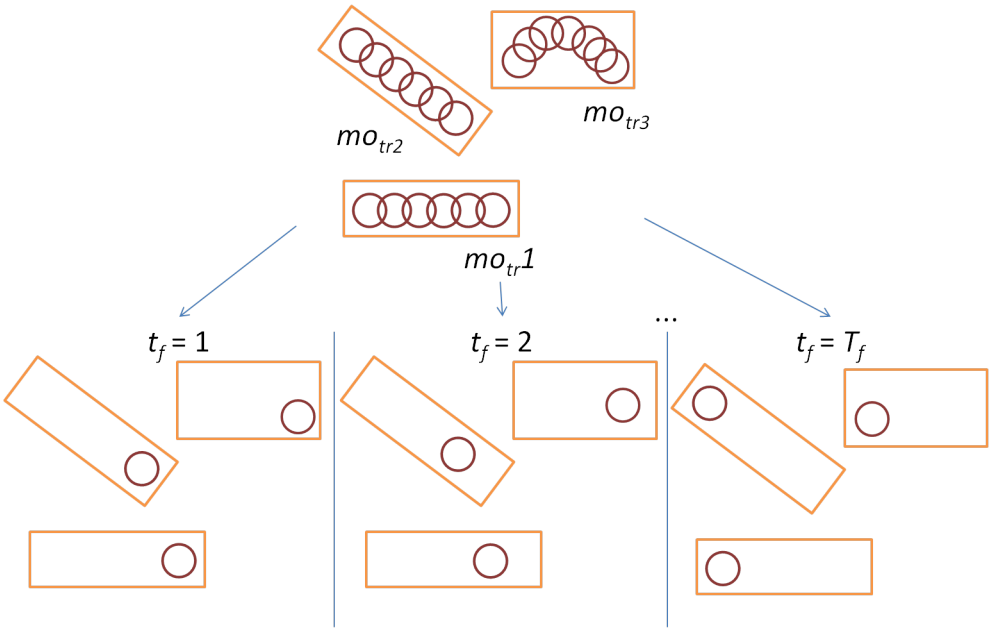

Figura 3.3: Configurações estáticas do ambiente: A cada instante de tempo tf, existe uma configuração estática de MOs associada

conjunto e discretiza os vetores de acordo com a magnitude das velocidades, aplicando em seguida o mesmo algoritmo AprioriFRN para detectar os padrões.

3.1.1 O Método de Previsão ANNPM

De forma resumida, Redes Neurais Artificiais foram propostas com o objetivo de simular o funcionamento do cérebro humano. Três aspectos biológicos básicos são representados: nós funcionam como os neurônios, tendo funções de ativação que podem ser calculadas a cada instante de tempo, dependendo dos estímulos externos; ligações entre os nós representam sinapses, tendo cada nó um conjunto de entradas e saídas interconectadas com outros nós; pesos atribuídos a cada entrada representam a propensão que um nó tem de ser ativado por um nó anterior. Os nós são divididos em camadas e o número de camadas e a maneira como elas se comunicam classificam o tipo de Rede (RUSSEL; NORVIG, 1995). As Redes Neurais são treinadas em uma fase preliminar, onde um conjunto de entradas e saídas esperadas são apresentados e a Rede deve recalcular os pesos dos nós de forma que o erro seja minimizado.

No método de previsão ANNPM (*Artificial Neural Network Prediction Method*) são utilizadas duas Redes Neurais Artificiais do tipo *Feedfoward* de três camadas para cada MO_i. Cada rede é responsável por uma das coordenadas na previsão (supondo que a representação do ambiente é bidimensional). São no total $2n$ redes, onde n é o número de MOs identificados.

Primeiramente, o método ANNPM realiza um pré-processamento para extrair informações relevantes de cada lista mo_{lpi}. O conjunto de posições bidimensionais presente nas listas não fornece informações imediatas sobre o padrão de movimento empregado, já que o deslocamento do MO_i no ambiente faz com que cada posição da amostra seja diferente da anterior. O que se repete de fato são os vetores representantes da velocidade entre cada par consecutivo de posições na amostra, como é ilustrado na Figura 3.4. A partir de mo_{lpi}, os vetores de velocidade são gerados aplicando-se uma subtração entre dois membros consecutivos da lista como mostra a Fórmula (3.2), resultando em uma nova lista de velocidades mo_{lvi}. Como as posições das amostras foram tomadas em instantes de tempo t igualmente espaçados, não é necessário considerar esta constate.

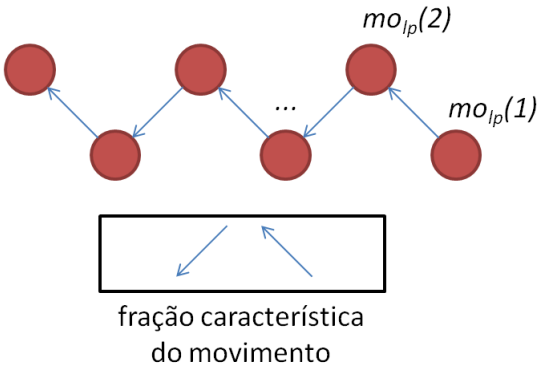

Figura 3.4: Vetores de velocidade extraídos de uma amostra de posições

$$mo_{lvi} = \{(x_i(t+1) - x_i(t), y_i(t+1) - y_i(t)) | \forall t \in T_c\} \tag{3.2}$$

A Figura 3.5 mostra a estrutura básica de uma das redes, ANN_{xi}, responsável pela coordenada x do MO_i. Como entrada são fornecidas as componentes x e y de nv vetores de velocidade consecutivos, onde $1 \leq nv \leq size(mo_{lvi})$ e $size(mo_{lvi})$ é o tamanho da lista mo_{lvi},

o que gera como saída a componente x do próximo vetor $nv+1$ de velocidade esperada. Neste caso, nv é determinado por experimentos, representando o número de vetores consecutivos necessários na entrada para que a rede forneça o resultado. A rede ANN_{yi} é equivalente, mas gerando como saída a componente y do próximo vetor. As redes são alimentadas com todas as coordenadas, o que aumenta a precisão do resultado.

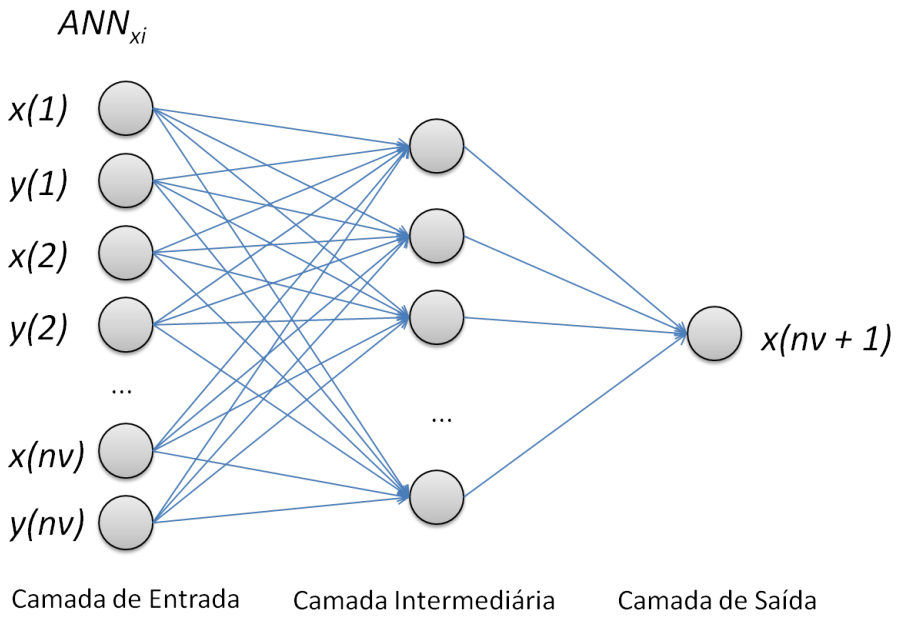

Figura 3.5: Estrutura da Rede ANN_{xi}

As listas mo_{lvi} são divididas em blocos b de tamanho $nv + 1$, contendo valores de velocidade consecutivos. Para o treinamento, os primeiros nv valores de cada bloco são fornecidos como entrada para as Redes, a componente x do valor $nv + 1$ como saída para a Rede ANN_{xi} e a componente y do valor $nv + 1$ como saída para a Rede ANN_{yi}. As Redes então reajustam os pesos de seus neurônios a fim de minimizar o erro.

Com as redes treinadas, elas devem ser capazes de fornecer o próximo vetor de velocidade esperado do obstáculo dados os nv vetores anteriores. As últimas nv velocidades conhecidas de cada MO, ou seja os últimos nv valores de mo_{lvi} para cada MO_i, são então fornecidos para as redes, que geram as componentes x e y da próxima velocidade estimada para o MO_i. A aplicação deste vetor na última posição conhecida do MO_i, ou seja o último

membro da lista mo_{lpi}, resulta na primeira posição prevista para o MO_i. Esta previsão é armazenada na lista mo_{tri} e o processo é repetido, tomando-se sempre o próximo vetor de velocidade previsto pelas Redes aplicado na última posição prevista armazenada em mo_{tri}. O resultado será a trajetória prevista para o MO_i. O Algoritmo 1 do Apêndice A desta dissertação ilustra a construção de mo_{tri} via o método ANNPM.

3.1.2 O Método de Previsão LOFORM

Ainda que as Redes Neurais tenham alta capacidade de aprendizado e flexibilidade, o treinamento das mesmas é uma fase computacionalmente custosa, notadamente se tiver que ser realizada para cada obstáculo individualmente. Como o FIRN necessita de um aprendizado *"ad-hoc"*, um método mais rápido e especializado para o problema se torna necessário. O campo de Mineração de Dados já é consolidado na busca automática ou semi-automática de padrões relevantes em bases de dados grandes e heterogêneas, visando a utilização prática deste conhecimento para alavancar vantagens comerciais ou identificar problemas em estratégias empresariais (WITTEN; FRANK, 2005). Dado o volume e diversidade de dados na busca, os algoritmos e estratégias empregados têm de funcionar da forma mais otimizada possível, agilizando o processamento e eliminando eventuais ruídos, como campos em branco ou dados fora do padrão esperado. Esta foi então a fonte explorada na busca por uma solução compatível com os requisitos de tempo na etapa de previsão do FIRN.

Uma das ramificações do campo de Mineração de Dados é a aplicação de algoritmos para a descoberta de padrões temporais ou sequências correspondentes em amostras distintas (AGRAWAL; RAMAKRISHNAN, 1994, 1995). Com a necessidade de processar as massas de dados rapidamente, uma estratégia comum é iniciar a busca com "sementes", micropartículas relevantes nas amostras, e estendê-las até que as unidades maximais (os padrões ou sequências de maior tamanho e com melhor índice de acerto) sejam encontradas. O conceito de "sementes" se assemelha às frações representativas de padrões de movimento no FIRN (Figura 3.1), o que sugere uma oportunidade de aplicação.

O LOFORM (*LOoking FOrward Robot Motions*) é um método de previsão que utiliza em seu núcleo um algoritmo inspirado nestas estratégias de descoberta de padrões, chamado

de AprioriFRN (*Apriori For Robot Navigation*). Especificamente, a base da construção do AprioriFRN é o algoritmo AprioriAll, utilizado para encontrar sequências temporais não contíguas maximais frequentes em bases de dados de transações comerciais realizadas por clientes de uma empresa (AGRAWAL; RAMAKRISHNAN, 1995). O AprioriAll é dividido em fases e aplicado sobre um banco de dados, onde as transações temporais de cada cliente são consideradas amostras:

1. *Sort Phase* (Fase de Agrupamento): o banco de dados é organizado de forma que todas as transações de um cliente estejam agrupadas cronologicamente. O conteúdo de cada transação é uma lista com os itens obtidos (a quantidade de cada item não é considerada);

2. *Litemset Phase* (Fase de Descoberta de Litemsets): No AprioriAll, um Litemset é um conjunto de L itens que tenha suporte mínimo, onde suporte é o número de clientes que tenham adquirido aquele conjunto de itens em uma transação. Por exemplo: se existe um Itemset I com os itens A e B, o suporte mínimo for de 25% e o número total de indivíduos no banco de dados for oito, pelo menos dois indivíduos têm de ter uma transação com os itens A e B para que o Itemset I seja um Litemset. A cada Litemset descoberto é atribuído um identificador único, um numeral inteiro de uma série contínua;

3. *Transformation Phase* (Fase de Transformação): Nesta fase, as transações de cada cliente são substituídas pelos respectivos Litemsets encontrados na etapa anterior. Se por acaso algum dos clientes não tiver nenhum Litemset em uma transacão, este cliente é eliminado da busca (ainda que ele faça peso na contagem de suporte mínimo). Esta fase simplifica a busca e diminui a complexidade do algoritmo.

4. *Sequence Phase* (Fase de Desoberta de Sequências): Cada Litemset encontrado na Fase de Descoberta de Litemsets é utilizado como semente para gerar sequências de dois ou mais Litemsets com suporte mínimo. Caso uma sequência deste tipo seja encontrada, ela pode ser utilizada como semente para uma busca posterior. O algoritmo continua até que novas sequências não sejam mais encontradas.

5. *Maximal Phase* (Fase de Descoberta de Maximais): Cada uma das sequências encontradas é averiguada e aquelas que não são subsequências de outras são selecionadas como maximais.

No pré-processamento do método LOFORM, as listas de amostras mo_{lpi} possuem as posições registradas de cada MO_i e é necessário converter estes dados para um conjunto de representação finita, de forma que o AprioriFRN possa identificar os padrões. Primeiramente, o ângulo α da direção é extraído de cada par de posições consecutivas em mo_{lpi}. Em seguida, α é mapeado para uma direção cardeal definida, como mostra a Figura 3.6. O resultado é uma lista mo_{ldi} de direções, como mostra a Fórmula (3.3), onde Ω representa a operação de mapeamento de uma direção em um ponto cardeal. Por exemplo, o ângulo 72º está entre 67.5º e 112.5º e, de acordo com a Figura 3.6, a direção Norte (N) será atribuída.

$$mo_{ldi} = \{\Omega(\alpha_{(t+1,t)}) | 0 \leq t \leq T_c\} \tag{3.3}$$

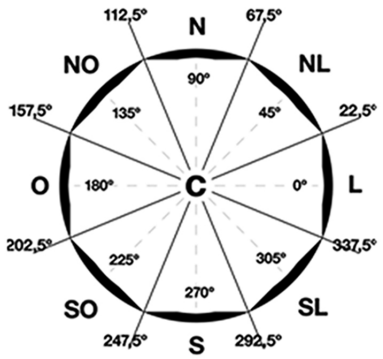

Figura 3.6: Conversão de ângulos em direções cardeais no pré-processamento do método LOFORM.

Como o deslocamento de cada MO_i é independente dos demais, o AprioriFRN é aplicado individualmente a cada lista mo_{ldi} e procura pelo padrão de movimento que se repete o maior número de vezes na própria lista. Dada esta mudança no cenário com relação ao AprioriAll, algumas adaptações foram necessárias:

- A Fase de Agrupamento do AprioriAll é utilizada para organizar cronologicamente as amostras referentes a um cliente. No AprioriFRN, as direções em uma lista mo_{ldi} são

a amostra referente a um MO_i e já se encontram organizadas cronologicamente. Desta forma, a Fase de Agrupamento é desnecessária;

- O AprioriAll aceita que Litemsets contendo apenas um item sejam escolhidos como sementes. No AprioriFRN, uma sequência deve ter um valor mínimo S_{min} para ser escolhida como semente (uma sequência com apenas uma direção não pode ser representativa do padrão de movimento);

- O conceito de suporte mínimo do AprioriAll foi convertido no AprioriFRN para a constante F_{min}, que representa a frequência mínima que uma sequência deve ter em uma lista para que seja considerada um padrão de movimento;

- Como o conceito de Litemset do ApririoriAll foi simplificado a uma constante no AprioriFRN, não faz sentido aplicar a Fase de Transformação, já que não existe mais a necessidade de converter transações em Litemsets;

Os valores de S_{min} e F_{min} são específicos para cada problema. Eles devem ser definidos de acordo com a complexidade esperada dos padrões de movimento e o tamanho das amostras. O valor de S_{min} não pode ser muito pequeno, já que ruídos poderiam ser erroneamente identificados como padrões e o valor de F_{min} não pode ser muito grande, o que dificultaria a descoberta de padrões caso a amostra seja muito pequena.

Tendo definido um valor para S_{min} e F_{min}, o AprioriFRN funciona da seguinte forma: uma sequência contígua de tamanho S_{min} é extraída de mo_{ldi} como semente. Em seguida, é verificado se esta semente se repete F_{min} vezes ou mais em mo_{ldi}. Em caso positivo, a semente é aumentada em uma unidade e o processo se repete até que a extensão da sequência cause um decréscimo na frequência. Quando isto ocorre, a sequência anterior (de maior tamanho e frequência) é escolhida como o padrão maximal atual cp_i (*chosen pattern*) para o MO_i e armazenada. Uma nova semente é retirada de mo_{ldi} e aumentada da mesma forma até que o novo padrão maximal seja encontrado. Caso este padrão tenha um tamanho s maior que o de cp_i, ele se torna o novo cp_i. O processo se repete até que mo_{ldi} seja completamente processada. O AprioriFRN completo pode ser encontrado no Algoritmo 2 do Apêndice A desta dissertação.

O processo iterativo aplicado no AprioriFRN é capaz de eliminar ruídos nas amostras e retornar o padrão de movimento maximal. Suponha que a ampliação de uma sequência $sq1$ de tamanho $sz1$ e frequência $fq1$ gere uma sequência $sq2$ de tamanho $sz1 + 1$ e frequência $fq2$. Existem três resultados possíveis:

- $fq2 = fq1$: A nova sequência $sq2$ possui a mesma frequência de $sq1$, sendo então uma extensão válida de $sq1$. A sequência $sq2$ será estendida mais uma vez na busca pelo padrão de movimento;

- $fq2 < fq1$: A nova sequência $sq2$ tem uma frequência menor que $sq1$. Isto significa que $sq2$ não é maximal ou que um ruído foi introduzido. Em qualquer dos dois casos $sq1$ será escolhido nesta iteração mas, no primeiro caso, $sq1$ é maximal e será o padrão escolhido no final do AprioriFRN e no segundo, $sq1$ é apenas um padrão interrompido por um ruído e a sequência maximal será encontrada a partir de outra semente da amostra;

- $fq2 = 0$: A nova sequência $sq2$ não se repete na amostra, o que sugere que $sq2$ introduziu um ruído que deve ser ignorado. A sequência $sq1$ anterior é escolhida nesta iteração, o que garante a remoção do ruído. O ruído poderia estar no meio de uma sequência maximal, fazendo com que $sq1$ seja apenas um padrão interrompido. Isto é resolvido em uma iteração posterior, que tomará como semente uma sequência onde o ruído não estará presente;

Com o padrão escolhido cp_i e a última posição conhecida (último membro de mo_{lpi}) de cada MO_i, é possível prever as próximas posições. É necessário no entanto descobrir em que etapa do padrão de movimento se encontrava o MO_i ao final da fase de coleta de amostras. Para tanto, um pareamento entre cp_i e as últimas direções conhecidas do MO_i, tomadas de mo_{ldi}, deve ser realizado, como mostra a Figura 3.7. As duas sequências são comparadas e, quando o pareamento é realizado com sucesso, cp_i é deslocado de forma que a próxima direção represente aquela que o MO_i tomará em seu próximo passo.

Cada direção em cp_i é convertida novamente em um ângulo de direção, mas existe uma perda de informação: qualquer ângulo que tenha sido classificado como um certo ponto

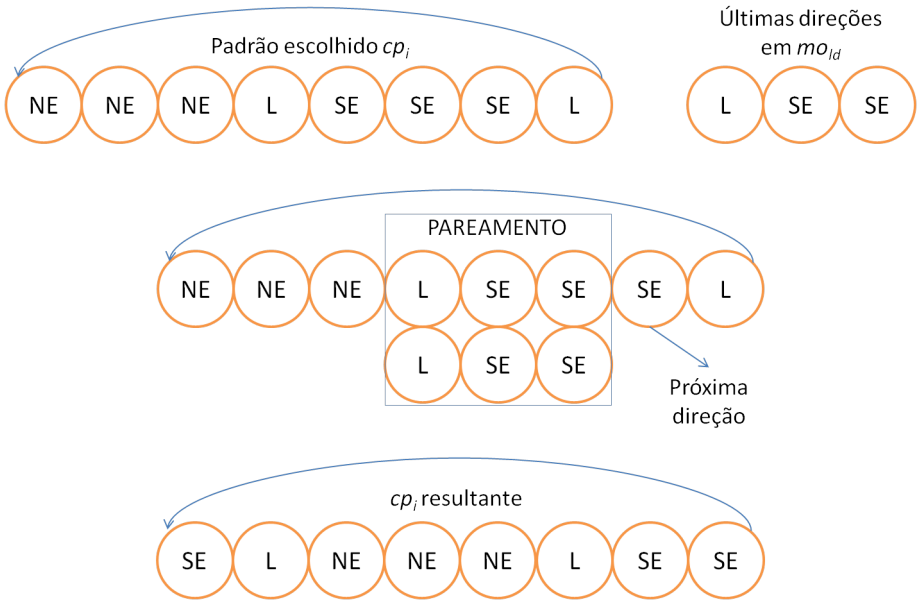

Figura 3.7: Alinhamento e deslocamento de *cp*, a fim de definir a próxima direção de movimento de um MO

cardeal no pré-processamento tomará o ângulo representante do ponto cardeal neste processo inverso. Ou seja, com base na Figura 3.6, qualquer direção entre 67.5º e 112.5º será classificada como Norte (N) no pré-processamento. Na conversão de *cp* novamente em ângulos, qualquer direção marcada previamente como Norte (N) será convertida no ângulo 90º com a horizontal. Este processo limita a capacidade de expressão do método LOFORM, mas é necessário já que o algoritmo AprioriFRN necessita de um conjunto finito de caracteres para encontrar os padrões.

Com o cp_i preparado, basta aplicar a primeira direção na última posição registrada em mo_{lpi} em conjunto à velocidade média conhecida AS_i (que pode ser extraída facilmente de mo_{lpi}), como mostram as Fórmulas (3.4) e (3.5), onde *mod* representa uma operação de módulo e $numPred$ é o número de previsões a serem feitas. Esta nova posição é armazenada em uma lista mo_{tri} e a próxima direção é aplicada nesta posição, gerando mais uma posição. O processo se repete, considerando cp_i como uma listra circular, até que a trajetória desejada do MO_i seja construída.

$$mo_{tri}(j+1) = (mo_{trix}(j+1), mo_{triy}(j+1)), \text{ para } j = 0, ..., numPred \tag{3.4}$$

$$\begin{cases} mo_{trix}(j+1) = mo_{trix}(j) + \cos(cp(\text{mod}(j, size(cp)))AS_i \\ mo_{triy}(j+1) = mo_{triy}(j) + \sin(cp(\text{mod}(j, size(cp)))AS_i \end{cases} \tag{3.5}$$

3.1.3 O Método de Previsão E-LOFORM

O método E-LOFORM possui as mesmas inspirações e idéias do método LOFORM, tendo o mesmo algoritmo AprioriFRN aplicado em seu núcleo. A diferença se encontra na etapa de pré-processamento. Ainda que funcional, o pré-processamento no LOFORM limita a capacidade de expressão, reduzindo todas as direções possíveis a grupo de apenas oito possibilidades (Figura 3.6). Uma discretização que tome como base as características próprias de cada obstáculo se tornaria mais eficiente que uma de caráter genérico, e é exatamente este problema que o E-LOFORM tenta resolver.

Considerando que um MO_i se movimenta de acordo com um padrão, as amostras de posições armazenadas em mo_{lpi} possuem informações relevantes sobre o aspecto geral do deslocamento do mesmo. Por exemplo, tomando o vetor da velocidade projetada pelo MO_i entre cada par de direções consecutivas na amostra, calculando o módulo dos mesmos e realizando um teste comparativo, é possível determinar qual a velocidade máxima aproximada projetada. Tendo este valor característico do MO_i, os demais vetores de velocidade do conjunto podem ser então classificados com relação ao módulo em um grupo finito, sendo esta solução mais específica do que a classificação geral baseada em pontos cardeais do método LOFORM. Uma análise ainda mais apurada pode ser realizada: dependendo do padrão de movimento, as componentes da velocidade no eixo x podem ser muito diferentes das componentes no eixo y, sendo mais apropriado classificar a magnitude da velocidade projetada em cada eixo independentemente.

Tomando como entrada a lista de amostras de posições mo_{lpi}, o E-LOFORM primeiramente calcula os vetores de velocidade entre cada par de posições consecutivas e os armazena em uma lista mo_{lvi}, como é feito no método de previsão ANNPM (Fórmula 3.2). Em seguida, cada vetor em mo_{lvi} é decomposto e as componentes x e y são armazenadas em listas mo_{lvix}

e mo_{lviy}. As componentes em ambas as listas são analisadas e os maiores valores em módulo encontrados são armazenados em MS_{xi} e MS_{yi}. O Algoritmo 3 do ApêndiceA dessa dissertação detalha esse processo.

Tendo como base as velocidades máximas projetadas em cada eixo, MS_{xi} e MS_{yi}, é possível classificar em grupos cada um dos valores em mo_{lvxi} e mo_{lvyi}. São estabelecidos nove níveis de velocidade onde serão classificados os mesmos. Como MS_{xi} ou MS_{yi} representam apenas a magnitude, é importante que os níveis representem também a direção do movimento. Desta forma, o nível 5 representa uma velocidade próxima a 0; os níveis $[6-9]$ representam velocidades progressivamente maiores na direção positiva do eixo, sendo o nível 9 mais próximo a MS_{xi} ou MS_{yi}; os níveis $[4-1]$ representam velocidades progressivamente maiores na direção negativa do eixo, sendo o nível 1 mais próximo a MS_{xi} ou MS_{yi}. A Figura 3.8 ilustra a divisão. A Fórmula (3.6) representa a operação de discretização das magnitudes dos vetores em mo_{lvxi} e mo_{lvyi}, convertendo cada valor em um nível apropriado.

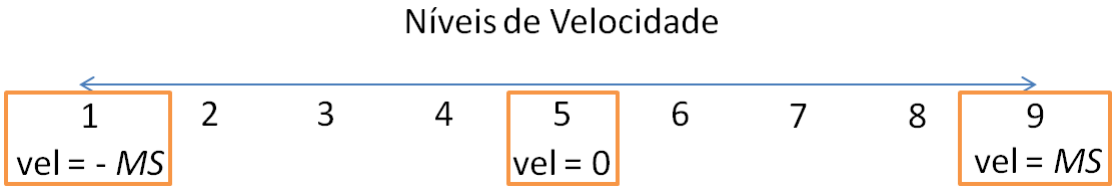

Figura 3.8: Níveis de velocidade relativos a MS no pré-processamento do método E-LOFORM

$$dmo_{lvai}(j)\Big|_{a=y}^{a=x} = \{round\left(\left(4 * \frac{mo_{lvai}(j) + MS_{ai}}{MS_{ai}}\right) + 1\right), j = 0, \ldots, mo_{lvi}(last)\} \tag{3.6}$$

Com o fim do pré-processamento, o movimento de cada MO_i será representado pelas listas discretizadas dmo_{lvix} e dmo_{lviy}, contendo os níveis de velocidade tomados entre cada posição da amostra original mo_{lpi}. Tendo os valores discretizados, já é possível aplicar o algoritmo AprioriFRN. Para cada MO_i, a lista dmo_{lvix} é dada como entrada, resultando no padrão escolhido para o eixo x cp_{xi}, seguida da lista dmo_{lviy}, resultando em cp_{yi}, da mesma forma como acontece com a lista mo_{ldi} no método LOFORM.

Os padrões escolhidos cp_xi e cp_yi são então alinhados respectivamente com os últimos membros de mo_{lvix} e mo_{lviy} (convertidos em níveis) a fim de definir a próxima velocidade em cada eixo a ser tomada pelo MO_i. O processo é análogo ao descrito na Figura 3.7, salvo que cada membro das listas representa níveis de velocidade, e não direções cardeais. Tendo os padrões escolhidos alinhados, os níveis de velocidade são reconvertidos em valores reais através da aplicação da Fórmula (3.7), que realiza operação inversa à Fórmula (3.6). Novamente, existe uma perda de informação já que todos os módulos de velocidade classificados em um nível são agrupados em um único valor representativo, mas desta vez esta discretização é feita baseada no módulo de velocidade máximo do próprio MO, e não baseada em um critério geral.

$$rcp_a\left(j\right)\left|\begin{array}{l} a=x \\ \\ a=y \end{array}\right. = \left\{\left(\frac{cp_a\left(i\right)-1}{4} * MS_{ai}\right) - MS_{ai}, i=0,\ldots,cp_a(last)\right\} \tag{3.7}$$

Tendo rcp_xi e rcp_yi, os padrões escolhidos alinhados e reconvertidos em valores reais nos eixos x e y, basta tomar a última posição conhecida do MO_i (o último membro de mo_{lpi}) e aplicar as Fórmulas (3.8) e (3.9), resultando na trajetória esperada mo_{tri} de cada MO_i.

$$mo_{tri}(j+1) = (mo_{trix}(j+1), mo_{triy}(j+1)), \text{ para } j = 0, ..., numPred \tag{3.8}$$

$$\begin{cases} mo_{trix}(j+1) = mo_{trix}(j) + rcp_x(\mathrm{mod}(j, size(cp))) \\ mo_{triy}(j+1) = mo_{triy}(j) + rcp_y(\mathrm{mod}(j, size(cp))) \end{cases} \tag{3.9}$$

3.2 Métodos de Planejamento

Os métodos de planejamento são responsáveis por gerar uma rota segura da posição inicial do IR até o objetivo. A entrada destes métodos são as listas mo_{tri} com a trajetória prevista de cada MO_i e a saída é a lista ir_{tr} com as posições a serem tomadas pelo IR para que alcance o objetivo. Esta trajetória considera apenas a previsão retornada na etapa anterior e podem existir colisões caso esta não seja correta e o IR não conte com um mecanismo

reativo emergencial. A corretude da previsão será discutida no Capítulo 4 e o mecanismo reativo não é contemplado nesta pesquisa.

Existem duas características comuns a todos os métodos: o planejamento é realizado antes do deslocamento, permitindo que algumas decisões sejam tomadas sem que o IR se submeta a uma situação de perigo; todos lidam com o conceito de "raio do medo" fr (*fear radius*), que representa o raio do círculo em volta do IR onde a presença de um MO pode gerar colisões e manobras de evasão devem ser tomadas imediatamente. Por exemplo, na Figura 3.9.a o MO_1 adentra a área delimitada por fr do IR, o que ocasiona o desvio da rota para evitar uma colisão (Figura 3.9.b). O MO_2 não gera nenhum tipo de reação, já que não intercepta a área do círculo em seu deslocamento. O IR não está efetivamente se movendo neste momento e os MOs também não estão realmente naquela posição. O que é exibido na Figura 3.9 é uma projeção da posição planejada pelo IR com relação às posições previstas dos MOs em $t_f \in T_f$, onde t_f representa um instante de tempo no futuro em que o IR espera que cada MO_i esteja em uma posição prevista $mo_{tri}(t_f)$, e T_f representa o tempo futuro total acessível pelo planejamento através das previsões. O comportamento evasivo (direção e velocidade da manobra) é particular de cada método.

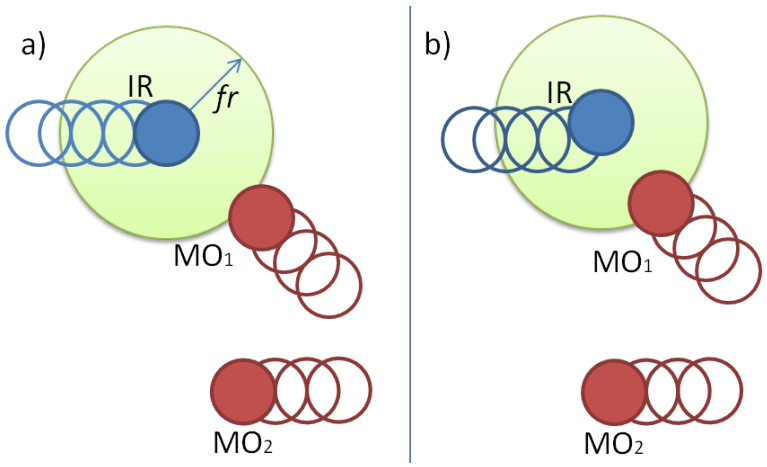

Figura 3.9: Exemplo de situação de risco: a) O MO_1 adentra o fr do IR; b) O IR inicia uma manobra de evasão.

As listas mo_{tri} possuem um número limitado de posições previstas para cada MO_i. Os métodos de planejamento consultam a posição estimada de MO_i a partir destas listas e em

algum momento não existirão mais posições futuras. Neste caso, o comportamento padrão dos métodos é planejar a trajetória até o objetivo, sem se preocupar mais com os obstáculos nesta etapa (certamente é interessante que o IR já tenha ultrapassado os obstáculos em seu planejamento neste momento).

Dois métodos foram originalmente desenvolvidos pelo grupo MASI: o método DSM (*Direct Search Method*) utiliza uma heurística gulosa, que considera para cada posição estimada de cada MO a melhor direção a ser tomada a partir de um conjunto limitado de possibilidades; o método FDSM (*Fuzzy Direct Search Method*) tem o mesmo comportamento do método DSM, mas ao invés de escolher uma direção de um conjunto limitado, consulta um controlador nebuloso que retorna a melhor direção para o planejamento.

3.2.1 O Método de Planejamento DSM

O primeiro método de navegação desenvolvido pelo grupo MASI, DSM (*Direct Search Method*), utiliza uma heurística gulosa para gerar uma trajetória do ponto inicial ao objetivo. Para cada instante de tempo futuro t_f, o IR considera a posição estimada de cada MO_i, $mo_{tri}(t_f)$, a localização do objetivo, e avalia uma direção que o aproxime o máximo do objetivo e o afaste dos MOs. O processo é repetido até que a próxima direção escolhida leve o IR ao objetivo ou não existam mais posições futuras estimadas nas listas mo_{tri} (neste caso, a heurística considera apenas a distância do IR ao objetivo).

Como o IR deve escolher uma direção a ser tomada a cada passo, o conjunto de possibilidades tem que ser limitado, a fim de que o processo transcorra em tempo hábil. Esta primeira versão do DSM utiliza apenas três direções possíveis: em frente (ângulo de 0^o com a horizontal); para a diagonal superior, em frente (ângulo de 45^o com a horizontal); para a diagonal inferior, em frente (ângulo de -45^o com a horizontal). Este conjunto garante que o IR irá sempre se movimentar em direção ao objetivo, sem realizar ciclos que poderiam comprometer a qualidade do caminho.

A Fórmula (3.10) mostra a heurística utilizada pelo método DSM, que consiste na soma da distância euclideana d entre o IR e o objetivo e do somatório de pe, uma penalidade inversamente proporcional à distância entre o IR e os MOs (Fórmula (3.11)). Na Fórmula

(3.11), o valor de pe_i é zero caso a previsão de posição em $\text{mo}_{tri}(tf)$ esteja fora da área delimitada por fr, ou a diferença entre fr e a distância euclideana entre o IR (p_{IRx}, p_{IRy}) e a posição prevista do MO_i, $mo_{tri}(tf)$, caso contrário. Existe ainda um peso associado w_f (*weight factor*) que simboliza a importância de o IR manter distância dos MOs (quanto maior w_f, maior será a penalidade).

A partir da posição inicial o IR planeja um passo nas três direções disponíveis e seleciona aquela que minimiza a heurística da Fórmula (3.10). Esta nova posição é armazenada em ir_{tr} e o processo continua até que a posição do objetivo seja escolhida. A velocidade do IR é constante na aplicação do DSM, então, independente da direção tomada, a distância percorrida será a mesma entre cada passo.

$$h = d + \sum_{i=1}^{n} pe_i,\ onde\ n\ é\ o\ número\ de\ MOs \tag{3.10}$$

$$pe_i = \begin{cases} 0,\ se\ dist(IR,\ mo_{tri}(tf)) > f_r \\ (f_r - dist(IR, mo_{tri}(tf))) * w_f,\ se\ dist(IR,\ mo_{tri}(tf)) \leq f_r \end{cases} \tag{3.11}$$

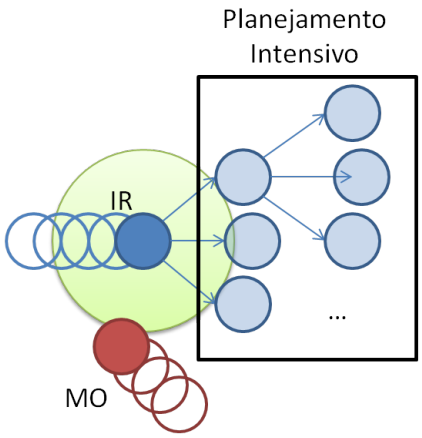

Figura 3.10: Busca intensiva no método DSM, a fim de encontrar o melhor caminho em volta do MO

Quando o planejamento leva o IR a uma situação onde um MO qualquer intercepta a área delimitada por fr, um comportamento mais cauteloso é tomado. Nestes casos, o IR

explora todas as direções possíveis até que não existam mais MOs na região de risco e o caminho selecionado é aquele de menor heurística (Figura 3.10). Este processo se assemelha a uma busca em largura em uma árvore e, dependendo da área coberta pelo MO, pode tomar um tempo incompatível com a proposta do FIRN. A limitação de três direções no método se dá por causa deste comportamento cauteloso e o tempo de processamento exigido.

O resultado do DSM é a lista ir_{tr}, com as posições que o IR deve tomar para alcançar o objetivo. O Algoritmo 4 do Apêndice A, detalha o método.

3.2.2 O Método de Planejamento FDSM

O método DSM apresenta algumas inconveniências devido à limitação nas direções disponíveis. Além de diminuir a capacidade de movimento, isto faz com que nem sempre o IR alcance seu objetivo, o que ocorre quando as manobras de evasão levam o IR a uma situação em que as direções possíveis não são suficientes para projetar uma rota de retorno até o objetivo. A fim de eliminar esta limitação e garantir a navegação, um novo método cujo cerne é a aplicação de dois Controladores Lógicos Nebulosos simples foi elaborado pelo grupo MASI. O objetivo é oferecer uma maior capacidade de movimento, sem a necessidade de realizar buscas taxativas.

Diferente da Lógica Booleana, a Lógica Nebulosa permite que a incerteza natural do mundo real seja representada por um modelo matemático bem fundamentado (ZADEH, 1965). Desta forma, um elemento não necessariamente necessita ser classificado exclusivamente como pertencente a um grupo específico, podendo pertencer a diferentes grupos com diferentes graus de pertinência. Um Controlador Lógico Nebuloso possui variáveis de entrada e saída nebulosas. Cada valor recebido como variável de entrada é avaliado e classificado em funções de pertinência com diferentes graus. Dependendo dos graus atribuídos a cada função para cada variável de entrada, um resultado é gerado pelas variáveis de saída. O problema de aplicação deve ser mapeado neste controlador, sendo necessário definir as variáveis e as funções de pertinência.

Foram desenvolvidos dois Controladores independentes para o método FDSM: um responsável por gerar o ângulo da direção do IR com relação ao objetivo, chamado de Con-

trolGOAL, e outro responsável por gerar o ângulo da direção do IR com relação aos MOs, chamado de ControlMO. O ControlGOAL regula a intensidade do ângulo de acordo com a distância no eixo y entre o IR e o objetivo, retornando o ângulo que os aproxime mais, enquanto que o ControlMO realiza operação semelhante mas de acordo com a distância entre o IR e o MO mais próximo, procurando o ângulo que os distancie mais. A distância entre o IR e os pontos de interesse no eixo x é levada em consideração antes de aplicar um Controlador e é ela que determina qual será utilizado: caso a distância entre o IR e algum MO seja menor que o raio do medo fr no eixo x, o ControlMO é ativado; caso contrário, o ControlGOAL é ativado.

Existem duas características que devem ser levantadas sobre a modelagem do problema: a independência entre os Controladores e o uso exclusivo da distância no eixo y. Os Controladores funcionam independentemente pois as manobras de evasão têm um peso maior que a aproximação ao objetivo. Desta forma, a ativação exclusiva do ControlMO em uma situação de perigo garante que o IR utilizará todos os seus esforços para desviar dos MOs. O movimento em direção ao objetivo só é retomado quando o IR encontra-se fora de uma situação de perigo. A independência também agiliza o processamento, já que apenas um controlador é ativado por vez. Os Controladores consideram apenas o eixo y pois neste método, assim como no DSM, o IR se movimenta sempre na direção do objetivo no eixo x, sem realizar recuos, bastando apenas direcionar o IR no eixo y para que o objetivo seja alcançado.

A Figura 3.11 mostra as variáveis de entrada e saída dos Controladores. A entrada do ControlGOAL (Figura 3.11.a) varia de -400 a 400, que é o limite do campo de simulação; a entrada do ControlMO (Figura 3.11.c) varia de $-fr$ a fr, sendo que qualquer valor acima ou abaixo é tratado como nos limites (na Figura 3.11.c, o exemplo toma fr como 100); a saída de ambos os controladores (Figuras 3.11.b e d) variam entre -90º e 90º graus com a horizontal, fazendo com que o IR nunca se distancie do objetivo durante manobras, evitando ciclos.

A Figura 3.12 é uma representação gráfica das regras de ativação dos mesmos. Em ambos os casos, a entrada é a distância do IR ao ponto de interesse no eixo y e a saída é o ângulo da direção. Como pode ser visto na Figura 3.12, quanto maior a distância, maior é a

intensidade da manobra. Existe no entanto um comportamento diferente quando a distância no eixo y é igual a zero. O ControlGOAL mantém um ângulo de direção de 0^o com a horizontal, a fim de encurtar a distância do IR ao objetivo, enquanto que o ControlMO força um desvio para cima, tirando o IR da direção do MO. A escolha da direção do desvio neste caso é arbitrária, e o objetivo é apenas tirar o IR de uma situação de inércia.

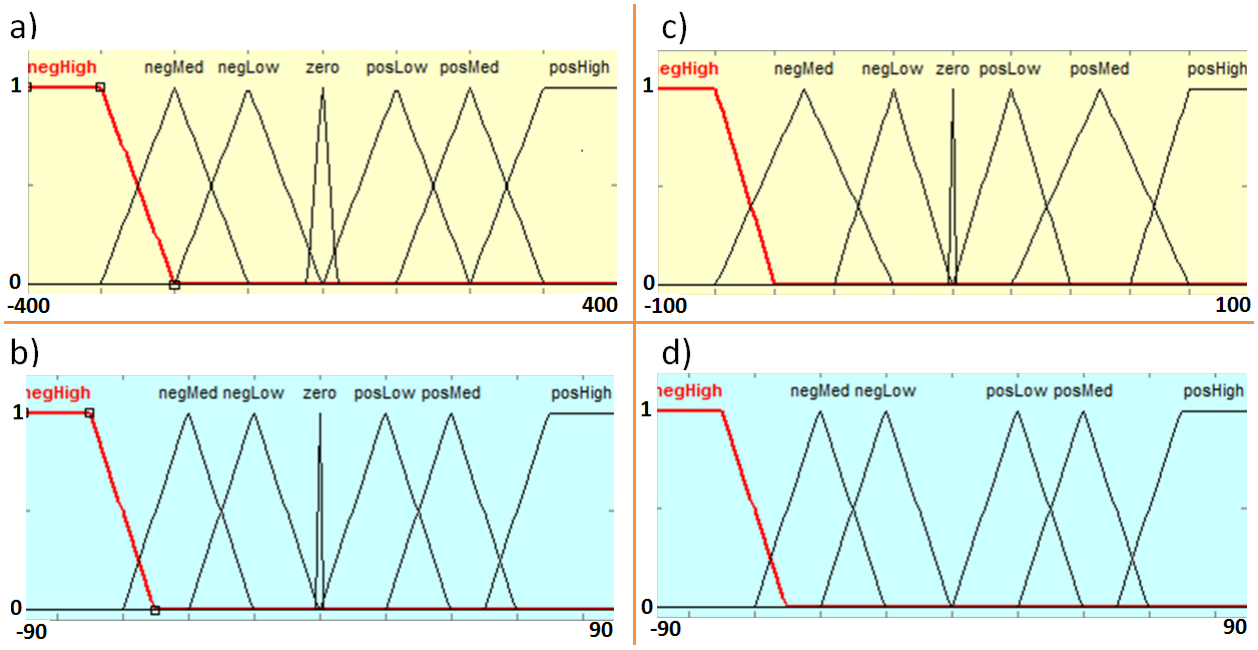

Figura 3.11: Variáveis dos Controladores Lógicos Nebulosos: a) Entrada "distY" para o ControlGOAL; b) Saída "angle" para o ControlGOAL; c) Entrada "distY" para o ControlMO; d) Saída "angle" para o ControlMO

O FDSM funciona da seguinte forma: a partir da posição inicial o IR avalia sua distância entre o objetivo e os MOs. Se algum MO se encontra dentro do raio do medo fr, o IR consulta o ControlMO tendo como entrada a distância no eixo y entre o IR e o MO mais próximo, o que resulta na direção do movimento. Esta direção é aplicada e o IR planeja um passo em frente. Caso contrário, o IR consulta o ControlIR tendo como entrada a distância no eixo y entre o IR e o objetivo e toma um passo nesta direção. O processo se repete até que o objetivo seja alcançado.

Apesar de garantir a flexibilidade na escolha das direções e agilizar o processamento, o método FDSM apresenta as seguintes limitações: como o número de MOs é variável e não

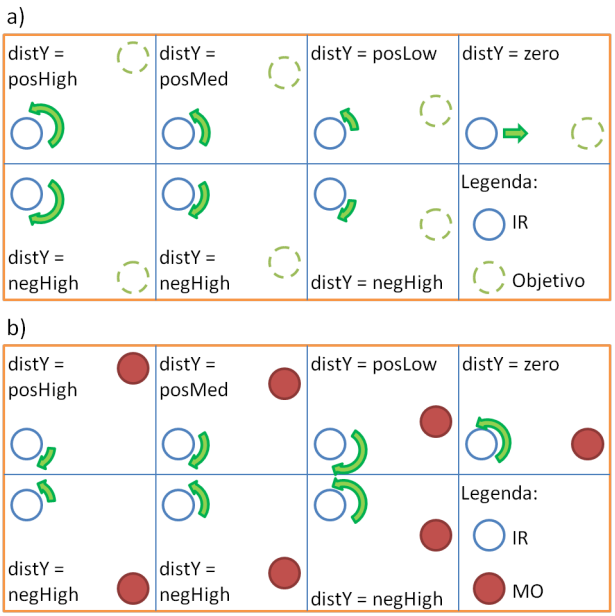

Figura 3.12: Representação gráfica das regras para os Controladores Nebulosos: a) ControlGOAL; b) ControlMO

foi encontrada uma modelagem capaz de considerar todos simultaneamente, o ControlMO considera apenas o MO mais próximo ao determinar a direção da manobra; assim como no método DSM, a velocidade do IR é constante.

Neste Capítulo foram abordados os aspectos gerais do FIRN e os métodos de previsão e planejamento desenvolvidos pelo grupo MASI. O *framework* é aplicável em ambientes onde as posições de MOs próximos ao IR podem ser obtidas através de câmeras ou sensores. São contempladas quatro etapas: Coleta de Amostras, Previsão, Planejamento e Navegação. Os métodos de previsão, ANNPM, LOFORM e E-LOFORM recebem como entrada amostras de posições tomadas pelos MOs e devem processá-las a fim de identificar padrões de movimento replicáveis, capazes de fornecer as trajetórias estimadas. Além disso, eles possuem mecanismos de redução de ruídos nas amostras. Os métodos de planejamento DSM e FDSM

consultam a previsão resultante e planejam uma rota até o objetivo, evitando colisões no caminho. O método DSM aplica uma etapa de busca intensiva ao identificar um MO próximo durante o planejamento, calculando o melhor caminho para evasão. O método FDSM não possui este mecanismo, mas possibilita que qualquer direção entre -90º e 90º com a horizontal seja escolhida

4 SIMULADORES

Neste capítulo serão apresentados os simuladores desenvolvidos para a execução de experimentos de aplicação do FIRN. O primeiro simulador foi desenvolvido em momento anterior pelo grupo MASI, tendo como principal característica a capacidade de lidar com um obstáculo dinâmico se movendo de acordo com um padrão de movimento definido. Ainda em momento anterior à pesquisa associada a esta dissertação, experimentos de avaliação da corretude dos métodos de previsão foram realizados neste simulador, cujos resultados serão enunciados na Seção 4.2.

O segundo simulador abordado foi desenvolvido para esta dissertação, constituindo portanto o primeiro produto do trabalho desta pesquisa. Ele é capaz de lidar com vários obstáculos simultaneamente, possibilitando que experimentos de avaliação do tempo de processamento sejam melhor desenvolvidos. Além disso, este simulador possibilita o acoplamento de novos métodos de previsão e planejamento, além da instauração de outras estratégias de navegação e não somente aquelas empregadas no FIRN. Este simulador será a base para os experimentos delineados na Seção 4.4 e no Capítulo 5.

A fim de eliminar qualquer ambiguidade e diferenciar os dois simuladores no que se segue, o primeiro será chamado de Simulador FIRNp (Simulador para o FIRN preliminar) e o segundo de Simulador FIRNn (Simulador para o FIRN novo).

Afora as características distintas de cada simulador, o funcionamento básico é o mesmo: Inicialmente são definidas as posições e os atributos do IR e dos MOs, assim como a posição do objetivo. O movimento dos MOs e as ações do IR são executadas por rodadas, que duram uma quantidade de tempo definida no simulador. A cada rodada, os MOs dão

um passo com velocidade e direção definidas no início da simulação, o que representa de forma discreta o movimento contínuo dos obstáculos no mundo real. O comportamento do IR depende da etapa aplicada do FIRN naquela rodada:

- Coleta de Amostras: A cada rodada o IR registra a posição corrente dos MOs e as armazena em vetores de amostras de posições;
- Previsão e Planejamento: Nestas etapas, o processamento do IR não está atrelado às rodadas do simulador. Enquanto os MOs continuam tendo suas posições atualizadas a cada rodada, o IR processa os métodos de previsão e planejamento livremente, tomando o tempo necessário para concluí-las sem interrupção;
- Navegação: Na etapa de navegação, o IR volta a sincronizar suas ações com as rodadas do simulador, dando um passo de acordo com o caminho planejado na etapa de Planejamento a cada rodada.

4.1 Simulador FIRNp

Tendo a concepção inicial do *framework* e os primeiros métodos de previsão e planejamento, surgiu a necessidade de testar sua aplicabilidade. Como a prioridade inicial era avaliar a corretude da etapa de previsão, o Simulador FIRNp tem suporte para apenas um MO. Os métodos de previsão e planejamento apresentados no Capítulo 3 foram implementados para o IR, mas dada a limitação no número de MOs, experimentos realizados contemplando o planejamento tiveram caráter preliminar, sem uma definição rigorosa dos casos de teste e parâmetros de avaliação. O simulador foi construído na linguagem Java.

Com o objetivo de desafiar um IR que adota o *framework* proposto para navegar, foram decididos para os experimentos quatro padrões básicos de movimento para o MO, cada um com características próprias:

- Linha: Este é o padrão de movimento mais simples, mas pode ser encontrado com frequência no mundo real (Figura 4.1.a). Por exemplo, pesquisas avaliando o deslocamento de agentes humanos em ambientes onde existem pontos de interesse bem

definidos concluíram que grande parte dos deslocamentos se dão entre estes pontos, existindo trajetórias comuns que são adotadas pelos agentes. Estas trajetórias normalmente são o menor caminho entre os pontos de interesse, o que caracteriza um conjunto de rotas em linha reta conectadas, evitando eventuais barreiras ou obstáculos (FOKA; TRAHANIAS, 2010; TSUBOUCHI et al., 1992);

- Onda: Ainda que seja mais incomum de ocorrer no mundo real, o padrão de movimento em Onda apresenta mudanças suaves na direção do movimento que podem confundir o IR (Figura 4.1.b). Além disso, a forma da onda possui a flexibilidade de ser alterada facilmente através de parâmetros como amplitude e período;

- Quadrado: O movimento em forma de Quadrado pode ser considerado uma complicação adicional do movimento em Linha (Figura 4.1.c). Ainda que pareça tão simples quanto o movimento supracitado, a mudança súbita de direção nos vértices do quadrado impossibilitam o uso de alguns métodos de previsão, como será visto na Seção 4.2.

- Oito: O padrão de movimento em Oito é uma complicação adicional do movimento em Onda e apresenta alterações na direção do movimento nos eixos x e y, assim como velocidade variável entre cada passo (Figura 4.1.d). É o movimento que apresenta maiores dificuldades durante a evasão, já que a troca constante de direção e velocidade demandam um número maior de manobras;

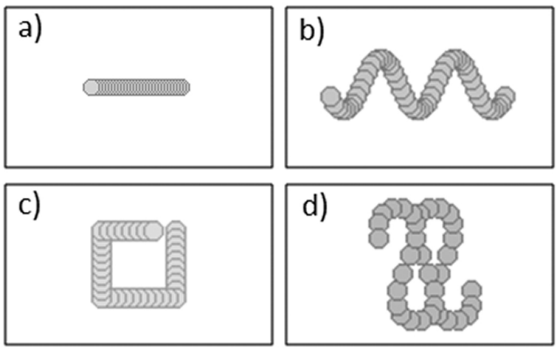

Figura 4.1: Padrões de movimento disponíveis para os MOs nos simuladores

A interface do Simulador FIRNp se encontra na Figura 4.2. Existe uma área de visualização, onde todas as etapas do FIRN podem ser acompanhadas graficamente. Ela exibe os objetos como se fosse uma câmera perpendicular ao ambiente. Todas as coordenadas representando posições são fornecidas de acordo com este ponto de vista. Abaixo da área de visualização estão os controles do simulador, onde é possível selecionar:

- O raio do medo fr do IR;
- A posição do objetivo;
- O padrão de movimento, posição inicial, velocidade e outras características como direção, amplitude e período do MO;
- Os métodos de previsão e planejamento a serem utilizados;

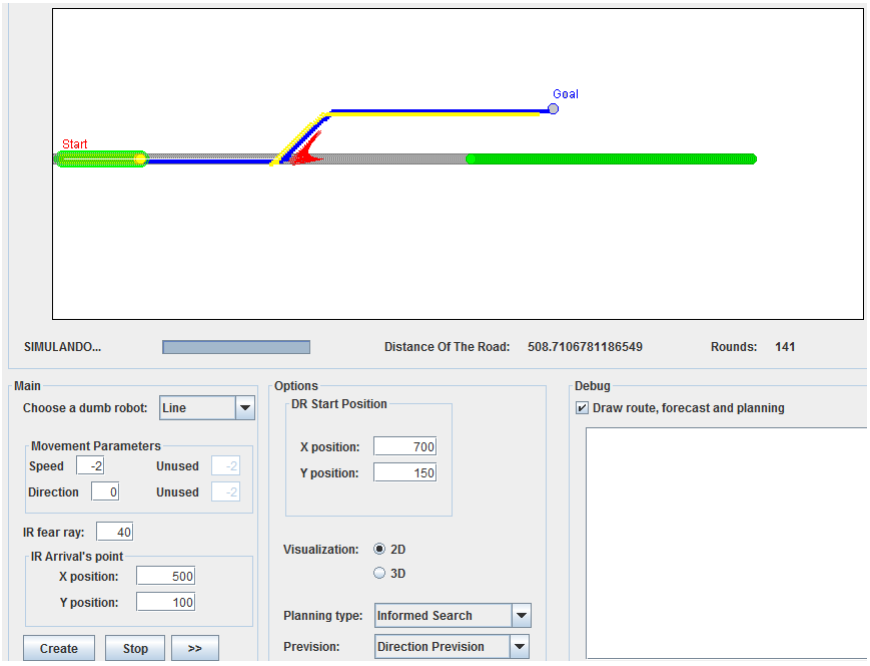

Figura 4.2: Interface do Simulador FIRNp

Houve o interesse do grupo MASI em inserir um mecanismo de visualização 3D, além do convencional em 2D, mas este recurso foi abandonado em seu desenvolvimento devido a complicações adicionais no tempo de processamento do Simulador FIRNp.

Alguns parâmetros da simulação são exibidos em tempo real tal como a distância do caminho gerado pelo planejamento e a rodada atual, onde cada rodada representa a passagem de uma quantidade constante t de tempo. Ao final da simulação, são exibidos o resultado final (colisão ou sucesso), o tamanho do caminho, tempo de previsão e planejamento, assim como o tempo total de simulação.

4.2 Desempenho da Previsão

Para avaliar a performance da previsão, foram elaborados casos onde o MO se encontra em uma posição fixa e começa a se mover de acordo com um dos padrões de movimento ilustrados na Figura 4.1. A posição inicial do MO para um experimento deve ser distante o suficiente do IR, para que o mesmo tenha tempo de coletar as amostras e executar o método de previsão e não necessita variar, já que não existe relação entre posição inicial e o resultado da previsão. Os parâmetros de cada padrão foram definidos como sendo:

- Padrão Linha: velocidade de 2 unidades de medida/rodada;
- Padrão Quadrado: velocidade de 2 unidades de medida/rodada;
- Padrão Onda: velocidade mínima de 2 unidades de medida/rodada; amplitude de 5 unidades de medida; período da onda igual a 20 rodadas;
- Padrão Oito: velocidade mínima de 2 unidades de medida/rodada; amplitude no eixo x de 5 unidades de medida; amplitude no eixo y de 10 unidades de medida; período igual a 20 rodadas;

A direção em todos os casos é de 0^o com a horizontal. Para os padrões Onda e Oito, a velocidade é definida como mínima já que para manter a forma do padrão, o MO deve projetar velocidades maiores que a determinada entre alguns passos (para o formato Onda, por exemplo, a velocidade do MO é maior nas proximidades dos nós das ondas). A velocidade máxima projetada pelos padrões Oito e Onda é aproximadamente duas vezes maior que a velocidade mínima definida. O padrão Oito possui amplitude nos eixos x e y, definidas como na Figura 4.3. Cada rodada no simulador dura 0.05s.

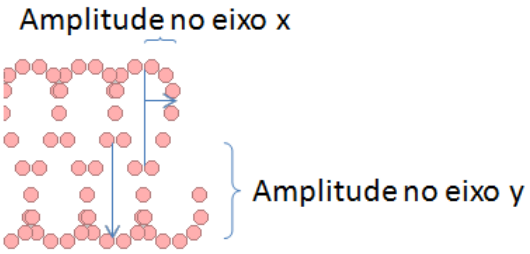

Figura 4.3: Amplitude no padrão de movimento "Oito".

Cada método de previsão apresentado na seção 3.1 deve processar a amostra de posições tomada de um MO se movendo, a cada simulação, de acordo com um padrão de movimento. A amostra é um conjunto de posições bidimensionais do MO tomadas durante a etapa de coleta de amostras. O tempo de coleta de amostras foi definido como sendo de 100 rodadas (aproximadamente 5s), sendo este então o número de registros de posições disponível para a previsão e o tamanho da lista mo_{lp}.

Para definir os melhores parâmetros de cada método de previsão no ambiente de simulação, um MO se movimentando de acordo com o padrão de movimento Oito, considerado o mais complexo do ponto de vista matemático, é utilizado e cada método é aplicado com um conjunto de parâmetros distintos. Aquele de melhor resultado, foi o escolhido:

- Para as redes do método ANNPM, o número de nós mais apropriado para cada camada foi de 16 nós na entrada, 26 nós internos e 1 nó na saída. Além disso, o erro mínimo no treinamento das redes foi definido como sendo 10^{-5} e o número máximo de épocas para treinamento como sendo 3000;

- Para os métodos LOFORM e E-LOFORM, os parâmetros de melhor resultado foram $S_{min} = 10$ e $F_{min} = 2$;

A configuração da máquina utilizada para estes experimentos é a seguinte: processador AMD Athlon64 X2 2.21 GHz, 2GB de memória RAM e sistema operacional Windows 7. As medidas de interesse tomadas foram o tempo de processamento e a corretude da previsão, que é a distância euclideana entre uma posição prevista $mo_{tri(tf)}$ e a posição real do

MO em um instante de tempo futuro tf, onde $tf \in T_f$ e T_f é o tempo total previsto pelos método de previsão. Como o tempo de processamento pode ser diferente entre uma execução e outra, cada experimento foi repetido 10 vezes e a média foi tomada como resultado final. Desta forma, foram realizadas 120 simulações (3 métodos de previsão x 4 padrões de movimento x 10 repetições).

Com relação à corretude da previsão, a Figura 4.4 exibe o erro a cada instante de tempo futuro tf de cada método para cada padrão de movimento. Ou seja, quando $tf = 1$, a primeira previsão de posição retornada por um método é comparada com a posição real do MO, e a distância entre as duas posições é calculada. Um erro crescente com o tempo indica que a direção prevista para o MO foi incorreta. Para o padrão em Linha, os 3 métodos retornaram resultados 100% corretos, não havendo necessidade de representá-lo graficamente.

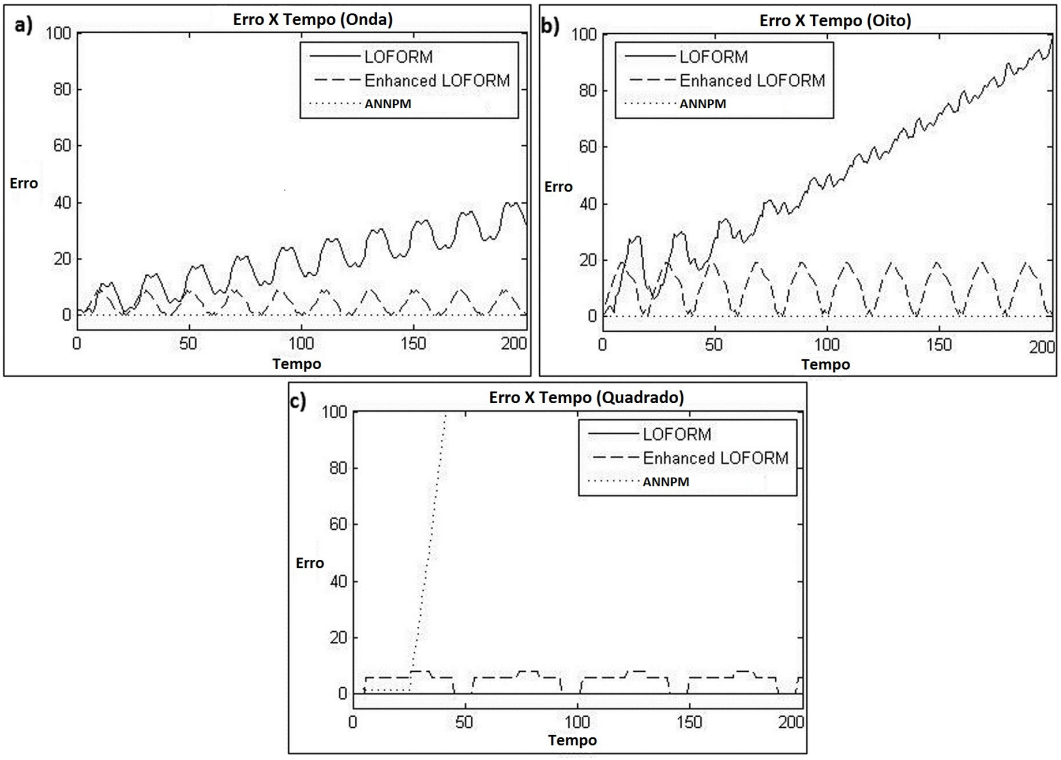

Figura 4.4: Erro x Tempo na previsão, comparando a posição prevista para o MO com relação à posição real em um instante tf: a) Onda; b) Oito; c) Quadrado.

Nas Figuras 4.4.a e 4.4.b, são apresentados os erros para os formatos Onda e Oito. O comportamento dos métodos para estes dois é semelhante: o método ANNPM foi capaz de prever ambos os padrões com 100% de acerto, o método E-LOFORM apresentou um erro constante e o método LOFORM um erro que aumentou linearmente com o tempo. A Figura 4.5, ilustra com um exemplo este resultado. Para o método LOFORM (Figura 4.5.a), a onda não só é representada com a amplitude incorreta, como também sofre um desvio de direção que distancia cada vez mais as previsões das posições reais. Para o método E-LOFORM, existe apenas um pequeno deslocamento das posições previstas, mas sem comprometer a forma ou a direção das mesmas (Figura 4.5.b). Isto confirma que a discretização realizada no método LOFORM acaba por limitar a capacidade de expressão do método. A mesma situação é observada para o padrão Oito.

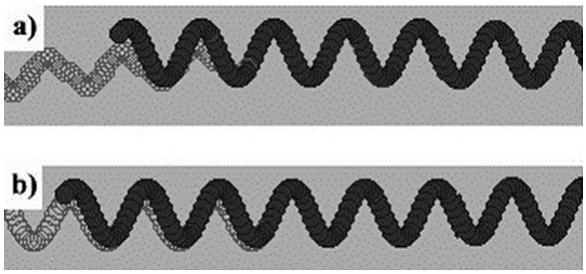

Figura 4.5: Erro na previsão para o padrão Onda: a) LOFORM; b) E-LOFORM. As formas escuras representam o movimento real e as formas transparentes, o movimento previsto

Para o padrão Quadrado, o resultado na Figura 4.4.c mostra que o LOFORM foi capaz de prever o movimento com 100% de acerto, o E-LOFORM com um erro muito baixo e constante e o método ANNPM com um erro que aumenta de forma exponencial. A Figura 4.6 ilustra o ocorrido. O método E-LOFORM desloca o formato do quadrado ligeiramente, mas sem deformar a forma ou alterar a direção (Figura 4.6.a). Já o método ANNPM deforma completamente o quadrado, o que se deve à falta de memória das Redes utilizadas para determinar os pontos de curva do padrão (Figura 4.6.b). As Redes fornecem sua saída baseadas apenas nas entradas fornecidas na iteração atual. Como o padrão Quadrado consiste de deslocamentos em linha reta durante um intervalo de tempo e subitamente força um giro de 90^{o}, as Redes não têm como prever quando se dará esta situação excepcional sem considerar as iterações anteriores.

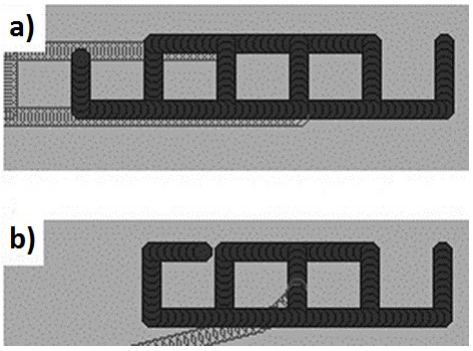

Figura 4.6: Erro na previsão para o padrão Quadrado: a) E-LOFORM; b) ANNPM. As formas escuras representam o movimento real e as formas transparentes, o movimento previsto

A Tabela 4.1 mostra o tempo tomado por cada método para processar a previsão. A primeira observação, e a mais relevante, é que o método ANNPM toma muito tempo para retornar os resultados. O desempenho das Redes é da ordem de 10^{-2}s, obtendo resultados ainda piores para o padrão Quadrado, já que as Redes esgotam o número de épocas permitido tentando diminuir o erro. Este tempo se justifica pelo fato de as Redes terem de ser treinadas para cada novo obstáculo, um processo que toma tempo.

Os métodos LOFORM e E-LOFORM possuem tempo de processamento da ordem de 10^{-4}s, sendo mais apropriados que as Redes neste sentido. É interessante observar que, para estes dois, o tempo de processamento não é proporcional à complexidade do padrão. O processamento dos padrões Onda e Oito toma menos tempo que o padrão em Linha. Isto se deve ao fato de o algoritmo AprioriFRN encontrar novas sequências no padrão Linha a cada iteração. Desta forma, o algoritmo precisa parar e armazenar a sequência estendida para depois retomar o processamento. Novas sequências não aparecem tão frequentemente para os padrões Onda e Oito.

Tabela 4.1: Tempo de processamento para os métodos de previsão

Tempo em Segundos (Método x Padrão de Movimento)	**Linha**	**Onda**	**Quadrado**	**Oito**
LOFORM	0.0048	0.0024	0.0019	0.0031
E-LOFORM	0.0063	0.0034	0.0040	0.0046
ANNPM	0.8523	1.6328	8.2635	1.3074

Dada a corretude e velocidade de processamento do método E-LOFORM, tendo sempre retornado previsões com erro baixo e tempo compatível com o método LOFORM, ele foi adotado como método padrão para realização dos experimentos delineados nas Seções 4.4 e 5.5. Na Seção 4.4, aproveitando a aplicação do FIRN no Simulador FIRNn, será feita uma avaliação rápida do desempenho da previsão nestes casos.

As Redes Neurais apresentaram um tempo de processamento incompatível com o *framework* proposto, tempo este que certamente irá aumentar com a introdução de mais obstáculos. Por isso o método, da forma como está implementado, foi abandonado.

4.3 Simulador FIRNn

Alguns experimentos preliminares foram realizados no Simulador FIRNp contemplando a etapa de planejamento. Os resultados mostram que o IR utilizando o método de previsão E-LOFORM e o método de planejamento DSM é capaz de alcançar o objetivo com taxa de sucesso de 94%. É um resultado bastante promissor, ainda que alguns pontos tenham que ser considerados:

- Quando o IR é confrontado com apenas um obstáculo, existe um grande espaço para manobras de evasão. Mesmo quando o MO é inserido entre o IR e o objetivo, a probabilidade de ocorrer uma colisão é pequena;
- A previsão e o planejamento dificilmente influenciarão na performance do *framework* ao processar apenas um obstáculo. No entanto, ao aumentar este número, é possível que o processamento tome tanto tempo que a previsão não será mais válida durante a navegação;
- O resultado do FIRN pode parecer positivo, mas se um método que não realize previsão fornecer resultados melhores nas mesmas configurações e condições, a contribuição do *framework* pode ser questionada;

Ficou então clara a necessidade de atualizar o Simulador FIRNp a fim de possibilitar experimentos com vários obstáculos simultaneamente. O ideal seria a extensão do código já

existente de forma a preencher o requisito básico, mas a falta de documentação e maneira como o programa foi organizado dificultaram esta atualização. Primeiramente, alguns detalhes referentes às rodadas e ao simulador em geral estavam sendo tratados dentro de classes referentes aos objetos individuais, o IR e o MO. Desta forma, qualquer alteração feita nas classes superiores traziam complicações adicionais que não eram facilmente rastreáveis. Além disso, um sistema robusto de *threads* não foi implementado no início e qualquer adaptação neste sentido resultou em travamentos e interrupções inesperadas no simulador. Ainda que um sistema deste tipo tenha sido implementado em uma tentativa preliminar de aproveitar o Simulador FIRNp, o programa continuou a abortar constantemente, ocorrendo em situações inesperadas e difíceis de diagnosticar. Sem *threads*, os MOs seriam incapazes de se movimentar enquanto o IR processasse as etapas de previsão e planejamento. O acoplamento de qualquer outro método de planejamento ou mesmo outra estratégia de navegação no Simulador FIRNp também seria problemático, dadas as falhas na hierarquia das classes. Desta forma, cada novo método de planejamento teria que lidar com as falhas do simulador e outros problemas poderiam aparecer. Com relação às estratégias de navegação comparativas baseadas em APFs, é necessário que elas estejam disponíveis no próprio simulador, a fim de que as comparações sejam feitas sob as mesmas condições. Mais uma vez, esta implementação seria complicada devido aos problemas apontados. Finalmente, o Simulador FIRNp não está preparado para lidar com execuções de experimentos em série de forma confiável.

Dado este cenário, decidiu-se por elaborar um novo simulador, Simulador FIRNn, tomando como base as idéias delineadas no Capítulo 3, mas considerando as novas necessidades. Do Simulador FIRNp foram herdados os códigos referentes ao cerne dos métodos de previsão e planejamento, descritos nas seções 3.1 e 3.2. A plataforma escolhida foi o Matlab, devido às facilidades proporcionadas com relação à manipulação de vetores e matrizes e a disponibilidade de módulos prontos (Redes Neurais, Lógica Nebulosa, entre outros).

O Simulador FIRNn foi construído de forma que novos métodos de previsão e planejamento possam ser acoplados facilmente no futuro. O único requisito para que os métodos funcionem são a entrada e saída, que devem estar no modelo esperado. Outras estratégias para comparação também podem ser inseridas, bastando sobrescrever a rotina de movimento

do IR. Os padrões de movimento dos MOs são os mesmos, com as mesmas características presentes no Simulador FIRNp.

O Matlab não conta com um sistema robusto de *threads*, mas objetos do tipo *timer* podem ser utilizados para simular a sincronia:

- Antes de iniciar uma simulação, um objeto *timer* é criado para cada obstáculo. Eles serão ativados em intervalos de tempo constantes t, onde t é o tempo de duração de uma rodada;

- Durante a etapa de coleta de amostras (Figura 3.2.a), cada *timer* dispara a rotina de movimento associada ao tipo de obstáculo (Linha, Quadrado, Onda ou Oito) nos intervalos determinados e calcula o próximo passo de acordo com a velocidade (tamanho do passo/t) e a direção definida. A nova posição do obstáculo é passada para o IR e armazenada em uma matriz. Como as únicas ações neste momento são o movimento dos obstáculos, as ações são executadas imediatamente;

- Durante as etapas de previsão e planejamento (Figuras 3.2.b e 3.2.c), o Matlab estará ocupado processando os métodos. Então, o movimento dos MOs não poderá ser calculado ou desenhado na área de simulação. No entanto, quando um *timer* não pode ser tratado imediatamente, a chamada associada é inserida em uma fila. Desta forma, quando o processamento atual cessa, a próxima ação a ser executada é aquela na primeira posição da fila. Terminado o planejamento, todas as chamadas na fila são tratadas e os MOs se encontram na posição que estariam caso o movimento fosse tratado em tempo real;

- Antes da etapa de navegação (Figura 3.2.d), um *timer* é criado para o IR, de forma que ele possa se movimentar de acordo com a trajetória retornada pelo planejamento. Como a prioridade deste novo *timer* é igual à dos demais, o IR se movimentará respeitando as rodadas. Ele também não poderá realizar qualquer movimento antes que todas as chamadas dos *timers* dos obstáculos disparadas durante as etapas anteriores sejam tratadas. Desta forma, o sincronismo é mantido e nenhum objeto dará mais de um passo por rodada.

A interface do Simulador FIRNn está ilustrada na Figura 4.7. Os controles estão melhor distribuídos, com os parâmetros do IR e dos MOs agrupados. Além do que já existia no Simulador FIRNp, agora é possível:

- Definir o número de MOs, bem como as características de cada um;
- Selecionar o tipo de simulação: manual, onde a simulação toma os parâmetros definidos nos controles principais; automática, onde os parâmetros são lidos de um arquivo e simulações são executadas em série; APF, onde o robô se movimenta de acordo com a estratégia de navegação baseada em APFs; APF Auto, idêntico ao anterior, mas tomando os parâmetros definidos a partir de um arquivo de entrada;
- Definir o tamanho de uma rodada;

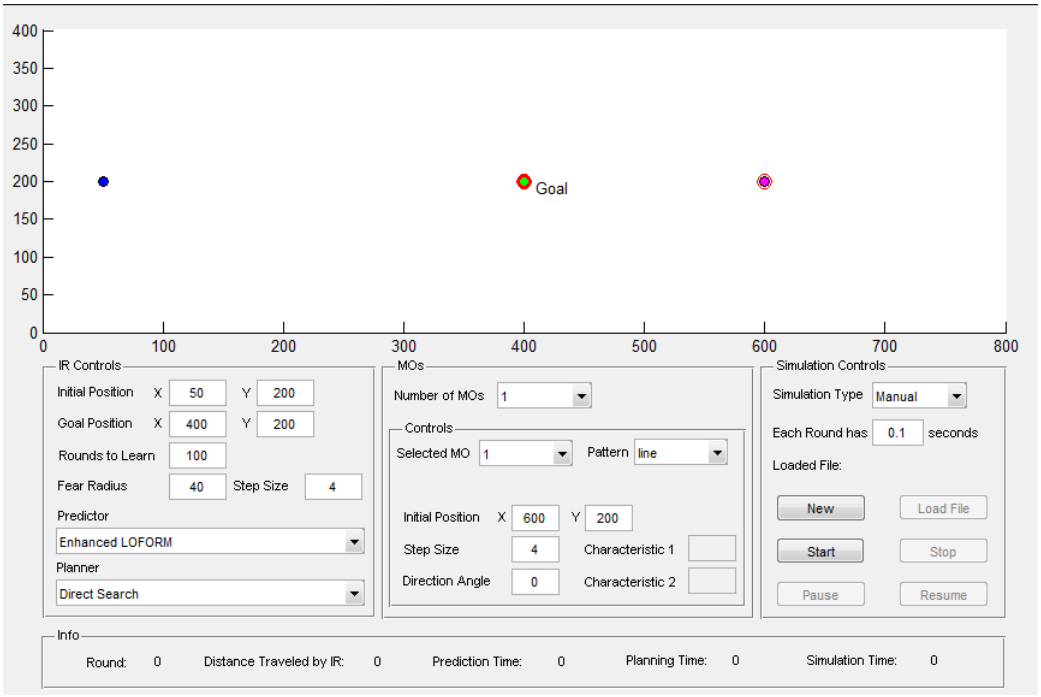

Figura 4.7: Interface do Simulador FIRNn

O número máximo de MOs foi definido como sendo 9. Isto se deu porque, a partir de 9 obstáculos, o simulador começou a apresentar algumas quedas na taxa de quadros com

que os desenhos eram apresentados na tela. Isto sugere um decréscimo na performance geral da simulação. O número pode ser aumentado ajustando-se um único parâmetro dentro do simulador.

Com o intuito de realizar experimentos em série, o Simulador FIRNn foi preparado para executar diversas simulações sem pausa, armazenando os resultados em um arquivo de saída. Teve-se cuidado de que nenhum detalhe de uma simulação fosse passado para a simulação seguinte, mantendo a memória utilizada do sistema sempre constante. Além disso, um programa à parte foi elaborado para construir casos de simulação de forma rápida. Não cabe, no contexto desta dissertação, apresentar este componente em detalhes, mas ele possibilita a definição de áreas específicas para a alocação de obstáculos e gera diversas situações com posições ligeiramente diferentes, fornecendo controle, flexibilidade e agilidade na operação.

Em relação aos métodos de previsão e planejamento, a única alteração feita foi no método DSM, que apresentou problemas ao ser aplicado a múltiplos MOs. Como o número de situações de conflito entre IR e MOs aumenta, o tempo tomado durante a busca intensiva se torna excessivamente longo (Figura 3.10), fazendo com que o planejamento perca sua validade quando o IR começa a se mover efetivamente. Desta forma, optou-se por aplicar o DSM em sua forma puramente gulosa no Simulador FIRNn, ignorando a busca intensiva e tomando sempre a melhor direção disponível no momento. Ainda que a abordagem gulosa diminua a performance do método quanto ao número de colisões, o tempo disponível aumenta ao ignorar a busca intensiva. Desta forma, duas versões do DSM foram aplicadas no Simulador FIRNn: o DSM3, contendo as três possibilidades de direção originais e o DSM7, permitindo que o IR escolha as seguintes direções: 0^o, 25^o, -25^o, 45^o, -45^o, 75^o e -75^o com relação ao eixo horizontal.

Um ponto negativo deste simulador com relação ao anterior são os requisitos do sistema. O simulador para vários obstáculos necessita de recursos computacionais mais avançados a fim de ser executado de forma satisfatória. Mesmo quando apenas um obstáculo é inserido na simulação, o programa utiliza mais recursos que o anterior elaborado em Java. Isto se deve aos próprios requisitos mínimos exigidos pelo Matlab. No entanto, como todas

as simulações são realizadas no mesmo contexto, as medidas tratadas nas Seções 4.4 e 5.5 permanecem válidas.

4.4 Experimentos no Simulador FIRNn

Simulando apenas um obstáculo não é possível observar o impacto do tempo de processamento no resultado da navegação. Como o FIRN utiliza apenas as posições tomadas na etapa de coleta de amostras, quando o processamento das etapas seguintes é finalizado, o IR considera que os MOs ainda estejam nas últimas posições registradas. Como o movimento continua em tempo real durante o processamento, o MO pode estar mais à frente do que o esperado. A Figura 4.8 mostra um exemplo: no instante de tempo t_f, o IR na posição $p_{ir}(t_f)$ colide com o MO_i na posição $p_{moi}(t_f)$, já que a posição prevista $mo_{tri}(t_f)$ não é mais válida. Este cenário acontece com pouca frequência quando o *framework* tem que processar apenas um obstáculo, mas pode se tornar comum com a introdução de outros MOs. O primeiro experimento desta seção tem então por objetivo avaliar a performance do FIRN contra vários obstáculos, aumentando o número de MOs gradativamente.

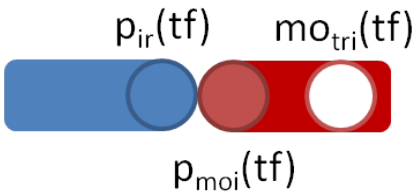

Figura 4.8: Colisão inesperada no FIRN devido ao tempo de processamento elevado. O IR na posição $p_{ir}(tf)$ colide com o MO_i na posição mo_{tri}, já que a previsão indica que o MO_i estaria na posição $mo_{tri}(tf)$ no instante de tempo tf

O segundo experimento realizado é comparativo. Como o FIRN tem por proposta fornecer uma navegação segura em um ambiente dinâmico, com a restrição de que os obstáculos possuam padrões de movimento reconhecíveis, espera-se então que a previsão contribua de tal forma que os resultados, tanto com relação ao número de colisões quanto à qualidade do caminho, sejam melhores que uma estratégia completamente reativa nos mesmos cenários.

Todos os experimentos desta Seção e do Capítulo 5 foram realizados no Simulador FIRNn, em um PC com a seguinte configuração: processador Intel Core i7 de 3.07 GHz, 8 GB de memória RAM, sistema operacional Windows 7 e versão R2009b do Matlab. As medidas de tempo tomadas são referentes a esta configuração. Os parâmetros gerais no simulador foram os seguintes:

- Área de simulação com dimensões de [0 800] no eixo x e [0 400] no eixo y;
- Velocidade do IR de 3 unidades de medida/rodada;
- Tempo de coleta de amostras de 100 rodadas;
- Duração de 0.1s para cada rodada;
- Nos padrões Linha e Quadrado, a velocidade do MO é de 3 unidades de medida/rodada;
- No padrão Onda, a velocidade mínima do MO é de 3 unidades de medida/rodada (chegando a um máximo de 6 unidades de medida/rodada), amplitude de 5 unidades de medida e período igual a 10 rodadas;
- No padrão Oito, a velocidade mínima do MO e de 3 unidades de medida/rodada (chegando a um máximo de 6 unidades de medida/rodada), amplitude no eixo x de 5 unidades de medida, amplitude no eixo y de 10 unidades de medida e período igual a 10 rodadas;

Antes de descrever os experimentos e expor os resultados, as estratégias de comparação serão apresentadas, com seus respectivos modelos matemáticos.

4.4.1 Estratégias de Comparação: $\mathbf{APF}_k$ e $\mathbf{APF}_p$

Foram adotadas estratégias baseadas em APFs nos experimentos comparativos por sua boa eficiência, modelos matemáticos bem fundamentados, simplicidade teórica e computacional. KHATIB (1986) introduziu o modelo geral, citando aplicações como a manipulação de braços mecânicos e a navegação autônoma de robôs. O modelo é extensível naturalmente para ambientes dinâmicos, bastando que os obstáculos próximos sejam rastreáveis a todo instante, o que é o caso do ambiente de aplicação do FIRN.

As estratégias de navegação utilizando APFs mapeiam campos de atração em volta de pontos objetivos e campos de repulsão em volta de obstáculos. Os campos devem ser funções contínuas e diferenciáveis e o robô se move de acordo com o gradiente negativo da combinação destes campos. Um exemplo se encontra na Figura 4.9, com os vetores da soma das forças dos campos visíveis. Foram utilizadas duas propostas: a original de KHATIB (1986), e a de YIN e YIN (2008) que considera, além da posição, a velocidade e aceleração dos pontos de interesse na modelagem do campo.

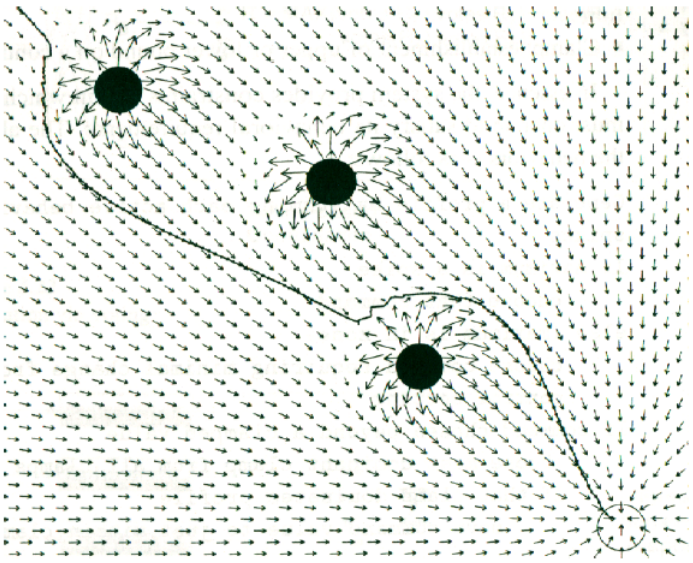

Figura 4.9: Exemplo de navegação utilizando APFs. O objetivo projeta um campo artificial de atração e os obstáculos, campos artificiais de repulsão. O robô se movimenta de acordo com a força resultante dos campos. Figura reproduzida de COSTA e PACHECO (2002)

Para as APFs originais de KHATIB, que a partir deste momento serão chamadas de APF_k, o robô é influenciado pelos campos considerando-se apenas a distância entre ele e os pontos de interesse. Para os obstáculos, quanto menor a distância, maior será a repulsão; para o objetivo, quanto menor a distância, menor será a atração.

Os campos de atração influenciam a direção do robô em qualquer ponto do ambiente e, se este fosse o único campo existente, o robô projetaria uma rota em linha reta até se aproximar do objetivo. Neste momento, a força do campo começa a diminuir, fazendo com que o robô reduza sua velocidade até parar no ponto desejado. A área dos campos de

repulsão é limitada e não influencia o robô de forma alguma enquanto este não se aproximar o suficiente. Quando se dá a aproximação, o campo em volta do obstáculo começa a repelir o robô, alterando a velocidade e direção do mesmo. Tanto a área em volta do objetivo, onde o robô sofre redução na velocidade, quanto a área em volta de um obstáculo são chamadas de áreas de atuação.

As áreas de atuação de uma APF podem ser modeladas de diversas formas. Por exemplo, se forem cilindros, a força exercida no robô é a mesma para qualquer ponto dentro da projeção circular no plano bidimensional; se forem cones, a magnitude da força se altera com a proximidade do robô ao centro da projeção, que no caso pode ser o obstáculo ou o objetivo. Na Figura 4.9, os campos em volta dos obstáculos são modelados como cones (quanto mais próximo do centro, maiores são os vetores representativos da força de repulsão). KHATIB (1986) delineia características gerais para vários formatos das APFs. O formato escolhido nesta pesquisa para a implementação da APF_k no simulador foi o cone. O raio da projeção do cone será chamado de *sp* (*spread*, propagação).

O deslocamento do robô em uma rodada é dado pela soma das forças exercidas pelo objetivo e n MOs (Fórmula (4.1)). Para o objetivo, primeiro é necessário avaliar a distância: se o robô encontra-se dentro do raio do objetivo, o deslocamento relativo deve ser zero (o objetivo foi alcançado); se o robô encontra-se dentro da área de atuação do campo de atração, mas fora do raio do objetivo, a velocidade deve diminuir de acordo com a proximidade; se o robô encontra-se fora da área de atuação, deve se mover com velocidade máxima até o objetivo (Fórmula (4.2)). Para os obstáculos, a distância também deve ser considerada: quando o robô se encontra dentro do raio do obstáculo (uma situação que deve ser evitada), o deslocamento deve ser o maior possível na direção contrária ao MO; quando o robô se encontra dentro do campo de atuação do obstáculo, mas fora do raio do mesmo, deve se afastar com intensidade inversamente proporcional à sua distância até o MO; quando o robô se encontra fora da área de atuação, nenhum deslocamento relativo é efetuado (Fórmula (4.3)). Se todas as influências forem somadas, o robô se movimentará de acordo com a disposição dos pontos de interesse.

Nas Fórmulas (4.2) e (4.3): α é uma constante que representa a intensidade do campo de atração; β é uma constate que representa a intensidade do campo de repulsão; dist(rob,

obj) é a distância euclideana entre o robô e o objetivo; dist(rob, MO_i) é a distância euclideana entre o robô e um MO; r é o raio de um ponto de interesse; θ é ângulo da direção entre o robô e um ponto de interesse; e sp é o raio da área de atuação de um campo.

$$\Delta APF_k = \Delta obj + \sum_{i=1}^{n} \Delta mo_i \tag{4.1}$$

$$\Delta obj = \begin{cases} \begin{cases} \Delta x = 0 \\ \Delta y = 0 \end{cases}, se\ dist(rob, obj) \leq r; \\ \begin{cases} \Delta x = \alpha(dist(rob, obj) - r)\cos(\theta) \\ \Delta y = \alpha(dist(rob, obj) - r)\sin(\theta) \end{cases}, se\ r \leq dist(rob, obj) \leq sp + r; \\ \begin{cases} \Delta x = \alpha sp\cos(\theta) \\ \Delta y = \alpha sp\sin(\theta) \end{cases}, se\ dist(rob, obj) > sp + r; \end{cases} \tag{4.2}$$

$$\Delta moi = \begin{cases} \begin{cases} \Delta x = -\infty\cos(\theta) \\ \Delta y = -\infty\sin(\theta) \end{cases}, se\ dist(rob, MO_i) \leq r; \\ \begin{cases} \Delta x = -\beta(sp + r - dist(rob, MO_i))\cos(\theta) \\ \Delta y = -\beta(sp + r - dist(rob, MO_i))\sin(\theta) \end{cases}, se\ r \leq dist(rob, MO_i) \leq sp + r; \\ \begin{cases} \Delta x = 0 \\ \Delta y = 0 \end{cases}, se\ dist(rob, MO_i) > sp + r; \end{cases} \tag{4.3}$$

As APFs são deslocadas junto dos pontos de interesse e a direção e velocidade do robô são calculadas a cada rodada. Diferentemente do IR no FIRN, o robô utilizando a estratégia APF_k se movimenta desde a primeira rodada da simulação. Não existe previsão, nem planejamento.

A simplicidade da estratégia já pode ser contemplada: são apenas três parâmetros a serem definidos e os cálculos são muito simples. A cada iteração da estratégia, são feitas $n+1$ considerações (*loops* do tipo *if*) e as operações mais complexas são o cálculo de distância e os cálculos de seno e cosseno, que são processadas rapidamente.

A segunda implementação é a de YIN e YIN (2008), que além da distância entre o robô e os pontos de interesse, considera também a velocidade e aceleração relativas. Quando o obstáculo está próximo do robô, não necessariamente ele está indo de encontro ao mesmo. Se o vetor de velocidade do obstáculo tem mesma direção que o do robô, eles estão andando no mesmo sentido e o obstáculo pode não ser uma ameaça. Da mesma forma, a aceleração reflete o comportamento imediato do obstáculo, indicando quando este pretende mudar de direção. Considerar estas variáveis na composição dos campos pode levar a uma navegação otimizada para ambientes dinâmicos, mas que possui uma penalidade: a implementação é mais complexa que a APF_k, com diversas constates que devem ser reguladas. Esta implementação será chamada de APF_p.

Dada a complexidade, a análise será feita mais detalhadamente. A começar pelo campo de atração: como mostra a Fórmula (4.4), o campo de atração diminui de acordo com a diferença entre a posição, velocidade e aceleração do robô e do objetivo. Quanto menor for a diferença, menor a intensidade do campo. ξ_q, ξ_v e ξ_a são constantes escalares e i, j e k são constantes exponenciais.

$$U_{att}(q, v, a) = \xi_q \|q - q_g\|^i + \xi_v \|v - v_g\|^j + \xi_a \|a - a_g\|^k \tag{4.4}$$

A partir de $U_{att}(q, v, a)$, é possível determinar a força de atração $F_{att}(q, v, a)$, como mostra a Fórmula (4.5): e_{qrg} é um vetor unitário que aponta para a posição do obstáculo a partir da posição do robô; e_{vrg} é um vetor unitário com direção igual a da velocidade relativa do objetivo para com o robô; e_{arg} é um vetor unitário com direção igual a da aceleração relativa do objetivo para com o robô.

$$\begin{gathered}
F_{att}(q, v, a) = -\nabla U_{att}(q, v, a) = \\
= -\nabla q U_{att}(q, v, a) - \nabla v U_{att}(q, v, a) - \nabla a U_{att}(q, v, a) = \\
= -\frac{\partial U_{att}(q,v,a)}{\partial q} - \frac{\partial U_{att}(q,v,a)}{\partial v} - \frac{\partial U_{att}(q,v,a)}{\partial a} = \\
= \xi_q i \|q - q_g\|^{i-1} e_{qrg} + \xi_v j \|v - v_g\|^{j-1} e_{vrg} + \xi_a k \|a - a_g\|^{k-1} e_{arg}
\end{gathered} \tag{4.5}$$

Para o campo de repulsão na Fórmula (4.6), dado que v_{rmoi} e a_{rmoi} são, respectivamente, a velocidade e a aceleração do MO_i relativas ao robô e dist(robô, MO_i) é a distância euclideana entre os dois: quando o robô se encontra dentro do campo de atuação delimitado por sp e v_{rmoi} e a_{rmoi} são maiores do que 0, o campo repele o robô com intensidade inversamente proporcional à distância e proporcional à v_{rmoi} e a_{rmoi}; quando o robô se encontra dentro do campo de atuação, mas a_{rmoi} é menor do que 0, o campo repele o robô considerando apenas a distância e a velocidade; finalmente, quando o robô está fora do campo de atuação e a velocidade v_{rmoi} é menor do que 0, a intensidade do campo é nula. Ainda na Fórmula (4.6), η_1, η_2 e η_3 são constantes escalares.

$$U_{repi}(q,v,a) = \begin{cases} \eta_1 \left(\frac{1}{dist(rob,MO_i)} - \frac{1}{sp}\right) + \eta_2 v_{rmoi} + \eta_3 a_{rmoi}, \\ se\ dist(rob,MO_i) \leq sp\ e\ v_{rmoi} > 0\ e\ a_{rmoi} > 0; \\ \\ \eta_1 \left(\frac{1}{dist(rob,MO_i)} - \frac{1}{sp}\right) + \eta_2 v_{rmoi}, \\ se\ dist(rob,MO_i) \leq sp\ e\ v_{rmoi} > 0\ e\ a_{rmoi} \leq 0; \\ \\ 0, \\ se\ dist(rob,MO_i) > sp\ e\ v_{rmoi} \leq 0; \end{cases} \tag{4.6}$$

A força exercida pelo campo U_{repi} se encontra na Fórmula (4.7). Ela é encontrada da mesma forma que a força atrativa. Neste caso, e_{moir} é um vetor unitário apontando do MO_i para o robô, e_{rmoi} é um vetor unitário apontando do robô para o MO_i, $v_{rmoi}\perp e_{rmoi}\perp$ é a componente da velocidade perpendicular a $v_{rmoi}e_{rmoi}$ e $a_{rmoi}\perp e_{moi}\perp$ é o análogo para $a_{rmoi}e_{rmoi}$.

$$F_{repi}(q,v,a) = \begin{cases} \eta_1 \frac{e_{moir}}{dist(rob,MO_i)^2} + \eta_2 \frac{v_{rmoi}\perp e_{rmoi}\perp}{dist(rob,MO_i)^2} + \eta_3 \frac{a_{rmoi}\perp e_{rmoi}\perp}{dist(rob,MO_i)^2}, \\ se\ dist(rob,MO_i) \leq sp\ e\ v_{rmoi} > 0\ e\ a_{rmoi} > 0; \\ \\ \eta_1 \frac{e_{moir}}{dist(rob,MO_i)^2} + \eta_2 \frac{v_{rmoi}\perp e_{rmoi}\perp}{dist(rob,MO_i)^2}, \\ se\ dist(rob,MO_i) \leq sp\ e\ v_{rmoi} > 0\ e\ a_{rmoi} \leq 0; \\ \\ 0, \\ se\ dist(rob,MO_i) > sp\ e\ v_{rmoi} \leq 0; \end{cases} \tag{4.7}$$

A força total é a soma de $F_{att(q,v,a)}$ e $F_{repi(q,v,a)}$, para todo MO_i em n (Fórmula (4.8)). Finalmente, o deslocamento da estratégia APF_p é dado tomando-se a componente x e a componente y do vetor representante da força exercida em F_{apfp}.

$$F_{apfp}(q, v, e) = F_{att}(q, v, e) + \sum_{i}^{n} F_{repi}(q, v, e) \tag{4.8}$$

$$\Delta APFp = (\Delta x\,(F_{apfp})\,, (\Delta y\,(F_{apfp}))) \tag{4.9}$$

Duas observações que são válidas tanto para a estratégia APF_k, quanto para a APF_p:

- O robô está sempre limitado à sua velocidade máxima. Portanto, mesmo que o campo exerça uma força de magnitude alta o suficiente a ponto de ultrapassar o deslocamento máximo permitido, o deslocamento será restringido e a direção mantida;
- KHATIB (1986) comenta que as APFs devem ser utilizadas junto a uma estratégia de planejamento. Isto se dá porque as APFs podem prender o robô em pontos sem saída quando este é inserido em um labirinto complexo. No ambiente de simulação do FIRN não existem labirintos deste tipo e todos os obstáculos se movem dinamicamente, isto é, não existem situações onde o robô permanece estático em um ponto, tornando as APFs uma estratégia viável para o ambiente do simulador e, por isso, um bom parâmetro de comparação.

4.4.2 Definição de Parâmetros para os Métodos e as Estratégias

Na estratégia APF_k, faz-se necessário definir as constantes α de atração, β de repulsão e o raio da área de atuação dos campos, *sp*. Para os dois primeiros, deve-se escolher valores de tal forma que os campos de repulsão sejam mais intensos que os de atração a fim de evitar colisões. Fixando α em 1, é necessário se preocupar apenas com o valor de β e *sp*. Estes parâmetros são relacionados e definidos juntos.

Para realizar estas definições, foi construído um pequeno conjunto de testes contemplando situações com um e oito obstáculos: nas situações com um obstáculo, o objetivo é

determinar se a estratégia está funcionando corretamente com os parâmetros definidos; com oito obstáculos, o objetivo é determinar o desempenho dos atributos escolhidos dada uma situação complexa. Este conjunto de testes é composto por 40 casos, 20 contemplando um obstáculo e 20 contemplando oito obstáculos.

Na primeira metade, o robô é fixado na posição (50, 200) e o MO e objetivo podem tomar posições do tipo (x_{ip}, 200), onde x_{ip} deve ser maior que 400 a fim de garantir espaço entre o robô e o MO. Cada caso contempla uma situação onde o obstáculo inicia na frente ou atrás do objetivo e a distância entre os três objetos varia. O ângulo do MO é sempre de 0º com a horizontal, fazendo com que o MO sempre confronte o robô, e os padrões de movimento foram selecionados de forma aleatória com distribuição uniforme.

Para os 20 casos com 8 obstáculos, o esquema da Figura 4.10 é usado como referência. A posição do robô é fixada mais uma vez em (50, 200). Em seguida, um MO é inserido em cada área numerada de 1 a 6 na Figura 4.10. Tomando como exemplo a área 1, um MO é inserido em uma posição aleatória dentro dos limites definidos pelo retângulo na imagem. Existe uma seta indicando a direção geral do movimento que um MO inserido nesta área tomará, a fim de que aconteça um confronto. Tendo definido os 6 primeiros MOs, a posição do objetivo é fixada como sendo (600, 200) ou (700, 200) (Goal 1 e Goal 2 respectivamente na Figura 4.10). Finalmente, dois MOs são inseridos na área 7 ou na área 8, dependendo da posição do objetivo. Estes últimos MOs estão restritos à posição 200 no eixo y, fazendo com que eles estejam obrigatoriamente entre o robô e o objetivo. Os MOs inseridos nas áreas 1 e 2 se movimentam sempre no padrão Quadrado; os demais MOs têm padrão de movimento aleatório com distribuição uniforme.

Um robô utilizando a estratégia APF_k deve navegar em direção ao objetivo evitando os obstáculos. O objetivo neste momento é avaliar os atributos β e sp, que são a intensidade e o raio da área de atuação dos campos de repulsão. O valor de α é fixado em 1 e o de β é testado como 2 e 4. O experimento é repetido para valores de sp entre 20 a 70 (sendo este último correspondente a 17.5% do tamanho da área de simulação no eixo y). No total são 480 casos de teste (40 cenários * 2 valores de β * 6 valores de sp). Os parâmetro de avaliação são o número de colisões e a qualidade do caminho em relação a um caminho em linha reta entre a posição inicial e o objetivo (quanto mais próximo de 100% a qualidade,

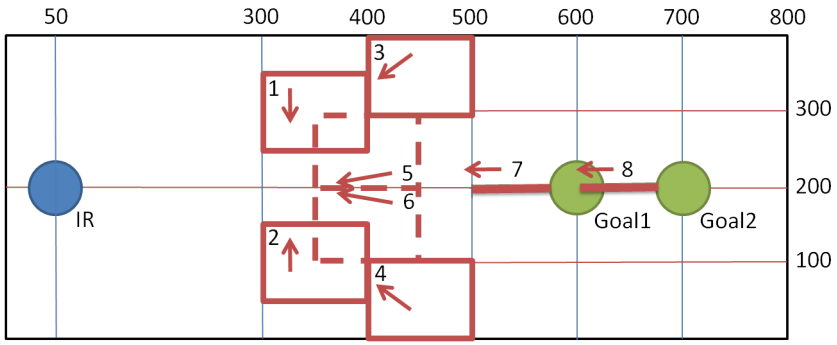

Figura 4.10: Esquema para geração de casos de testes relativos à definição de atributos. Oito MOs são distribuídos, um em cada região numerada de 1 a 8. A direção do movimento é indicada pelas setas

mais otimizado é o caminho). A qualidade do caminho é tomada apenas em casos de sucesso, já que a rota não é finalizada quando existe colisão.

É importante observar que, uma vez definidas as posições e padrões de movimento dos obstáculos em cada um dos 40 cenários de avaliação, estes parâmetros serão os mesmos para todos os experimentos. Ou seja, o robô com $\beta = 2$ e $sp = 30$ enfrenta exatamente as mesmas situações que o robô com $\beta = 4$ e $sp = 50$.

A Tabela 4.2 mostra o número de colisões para cada valor de β sobre cada valor de sp. Comparando as duas linhas, representativas dos valores de β determinados, é possível perceber que $\beta = 4$ fornece um menor número de colisões que $\beta = 2$ para qualquer valor de sp. A diferença é mais acentuada para valores de sp muito baixos ou muito altos: nas colunas onde $sp = 20$ e $sp = 70$ é possível perceber que a diferença é de 7 e 8 colisões respectivamente, enquanto que para $sp = 40$, a diferença é de apenas 2. Em relação a sp, a tendência é que o número de colisões diminua com o aumento de sp, mas existe uma discrepância entre $sp = 40$ e $sp = 50$, que será analisada posteriormente.

Tabela 4.2: Número de colisões para a estratégia APF_k com diferentes valores de β e sp

Nº de colisões (β x sp)	$sp = 20$	$sp = 30$	$sp = 40$	$sp = 50$	$sp = 60$	$sp = 70$
$\beta = 2$	23	15	10	12	10	9
$\beta = 4$	16	14	8	9	3	1

A Tabela 4.3 mostra a qualidade do caminho para cada valor de β sobre cada valor de sp. Comparando as duas linhas da tabela, a qualidade do caminho é maior para $\beta = 2$, independente de sp. A diferença se acentua quando sp é muito grande, chegando a um máximo de 13% quando $sp = 70$. De acordo com os dados das Tabelas 4.2 e 4.3, existe uma relação inversa entre o número de colisões e a qualidade do caminho: enquanto $\beta = 2$ alcança os melhores valores de qualidade do caminho, $\beta = 4$ fornece o menor número de colisões. Esta relação é natural, pois quanto maior o valor de β, maior a intensidade da força exercida pelos campos de repulsão. Uma força repulsiva maior ajuda o robô a evitar colisões, mas afeta a qualidade do caminho já que os desvios são mais expressivos. A qualidade do caminho diminui estritamente com o aumento de sp, sofrendo uma queda de 19% ao longo do experimento para $\beta = 4$.

Tabela 4.3: Qualidade do caminho para a estratégia APF_k com diferentes valores de β e sp

Qual. do caminho (β x sp)	$sp = 20$	$sp = 30$	$sp = 40$	$sp = 50$	$sp = 60$	$sp = 70$
$\beta = 2$	98.91%	97.40%	95.34%	94.40%	92.58%	90.81%
$\beta = 4$	96.87%	94.79%	91.98%	90.62%	85.68%	77.89%

Em termos gerais, é possível concluir que valores de sp muito baixos ou altos não são desejáveis. Quando $sp < 40$, o número de colisões é grande, principalmente para $\beta = 2$, havendo uma diferença significativa de 13 colisões entre $sp = 20$ e $sp = 40$ (Tabela 4.2). O decréscimo na qualidade do caminho entre $sp = 50$ e $sp = 70$ para $\beta = 4$ é de quase 13%. Restam então os valores 40 ou 50 elegíveis para sp. Como sp representa a área de atuação de um campo potencial, espera-se que uma área maior de atuação induza o robô a evitar colisões mais cedo, diminuindo o número. No entanto, entre $sp = 40$ e $sp = 50$ o comportamento é justamente o contrário, independente de β. A Figura 4.11 ilustra o ocorrido. Para sp entre 20 e 40, o robô não realiza ciclos no deslocamento, aplicando apenas um desvio na direção contrária do MO no eixo y. A partir de $sp = 50$, o robô começa a recuar junto do MO, tentando se afastar do mesmo. Em $sp = 50$, o resultado é a colisão pois o MO tem velocidade máxima maior. A partir de $sp = 60$, o robô consegue ultrapassar o obstáculo por se antecipar o suficiente, mas não antes de diminuir consideravelmente a qualidade do

caminho. Então, para o ambiente simulado onde serão realizados os experimentos, o melhor conjunto de valores para as AFP_k é $\alpha = 1$, $\beta = 4$ e $sp = 40$.

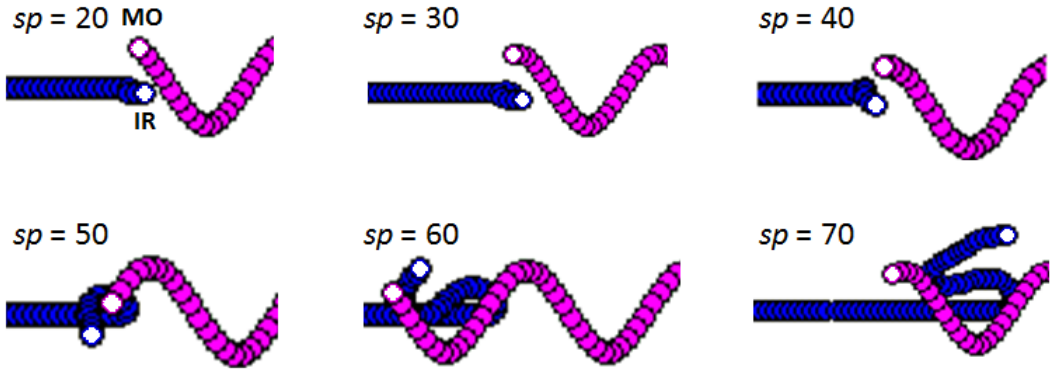

Figura 4.11: Colisão de um robô utilizando a estratégia APF_k para diferentes valores de sp. O valor ideal no simulador é $sp = 40$. A partir de $sp = 50$ o robô realiza desvios desnecessários que podem ocasionar colisão e diminuem a qualidade final do caminho

A análise para definição de parâmetros da estratégia APF_p é mais complexa, já que são 10 parâmetros: ξ_q, ξ_v, ξ_a, i, j, k, η_1, η_2, η_3 e sp. Alguns deles podem ser eliminados para aplicação no ambiente de simulação: ξ_v, ξ_a, j e k são parâmetros relativos à velocidade e aceleração do objetivo. Observando a Fórmula (4.4) e considerando que o objetivo não se move, qualquer valor diferente de zero para estes parâmetros irá apenas aumentar a intensidade do campo de atração U_{att} sem qualquer relação com o objetivo, já que v_g e a_g serão sempre zero. A única característica que irá regular a direção e diminuir a velocidade do robô será a diferença entre a posição do robô e do objetivo, sendo então possível simplificar o modelo igualando ξ_v, ξ_a, j e k a zero, sem nenhuma perda de informação.

Na definição dos demais parâmetros, YIN e YIN (2008) recomendam apenas que os valores de i, j e k sejam os mesmos, e que os valores de ξ_q, ξ_v e ξ_a não sejam muito distantes entre si. Uma diferença muita alta entre ξ_q e os outros dois podem gerar trajetórias muito oscilantes ou alterações muito pequenas na velocidade do robô. Como alguns destes parâmetros foram definidos como zero, esta preocupação não é necessária. YIN e YIN não abordam em detalhes a relação entre os demais parâmetros. Algumas observações tomadas durante a pesquisa associada a esta dissertação:

- Os valores de i e ξ_q devem ser menores com relação a η_1, η_2 e η_3, já que i é exponencial. No entanto, se forem muito baixos, o robô traçará percursos oscilantes, mantendo apenas uma direção muito vaga para o objetivo e evitando fortemente os obstáculos. O contrário faz com que o robô se movimente em linha reta até o objetivo, ignorando os obstáculos;

- Valores muito diferentes entre η_1, η_2 e η_3 também tornam a trajetória do robô muito oscilante. Valores similares ou iguais contribuem para rotas mais comportadas

Assim sendo, os melhores valores encontrados para os parâmetros da estratégia APF_p foram: $\xi_q = 3$, $i = 1.4$ (constante exponencial), $\eta_1 = 50$, $\eta_2 = 50$ e $\eta_3 = 50$. Dada a quantidade de parâmetros, não foi possível realizar testes extensivos, como foi feito para a estratégia APF_p. Estes valores resultaram em uma navegação com poucas oscilações e com reações ao campo de repulsão dos obstáculos. O parâmetro *sp* definido como 40 também funciona bem para esta estratégia, sendo então tomado como tal.

Para o FIRN, o método de previsão utilizado é o E-LOFORM, postos os resultados apresentados na Seção 4.2. Os atributos S_{min} e F_{min} do método também são os mesmos: 10 e 2, respectivamente. O método tem de prever 2000 posições à frente para cada MO ($size(mo_{tri}) = 2000$).

Os parâmetros dos métodos de planejamento do *framework* são: w_f para o método DSM (Fórmula 3.11) e fr tanto para o DSM, quando para o FDSM. Para definir o valor de w_f, primeiro é levantado o seguinte: para o DSM3 e DSM7, quando o IR está distante dos MOs ele segue em direção ao objetivo tomando sempre a direção que diminua o máximo possível a distância entre os dois. Quando um MO se aproxima do IR, este deve realizar manobras evasivas com intensidade regulada pelo parâmetro w_f (quanto maior o valor, mais acentuados são os desvios). Dado que a maior preocupação é evitar a colisão e o IR está limitado a três ou sete direções de movimento em direção ao objetivo no eixo x, eliminando qualquer chance de o mesmo andar em círculos ou recuar indefinidamente, a melhor escolha para w_f é um número próximo a $+\infty$. Isto faz com que o planejamento escolha sempre as direções que mais afastem o IR dos MOs, sem diminuir a qualidade do caminho.

O parâmetro fr é definido como o raio do circulo que envolve o IR, onde a detecção de qualquer MO fará com que o IR adote manobras de evasão. Esta definição é muito parecida com o sp dos campos de atração e repulsão das estratégias APF_k e APF_p. Se algum método tivesse um valor de sp ou fr muito maior que o de outro, certamente conseguiria um número menor de colisões em detrimento da qualidade do caminho. Desta forma, é interessante que este valor seja o mesmo para todos os métodos e estratégias. Dado o valor escolhido para as estratégias APF_k e APF_p, basta checar se o mesmo é compatível com os métodos de navegação do FIRN. Para testar é utilizado o mesmo conjunto de testes onde o valor de β e sp foram encontrados para as APFs. Um IR navega de acordo com o FIRN, aplicando cada método de navegação nos 40 cenários definidos anteriormente, num total de 120 experimentos (3 métodos de navegação x 40 cenários). Os critérios de avaliação são, mais uma vez, o número de colisões e a qualidade do caminho.

A Tabela 4.4 exibe os resultados referentes ao número de colisões para cada método de navegação aplicado com um valor diferente de fr. O número de colisões diminui com o aumento de fr, já que quanto maior o raio de percepção dos obstáculos, mais cedo o robô inicia os desvios. O método DSM3 não apresenta muitas variação no número de colisões, tendo uma diferença máxima de apenas 6 entre $fr = 20$ e $fr = 70$. Além disso, não existem variações significativas para valores entre 30 e 70. No método DSM7, percebe-se uma variação mais acentuada, sendo a máxima de 15 colisões entre $fr = 20$ e 70. Entre $fr = 30$ e 40 não existe muita variação. Para o FDSM, o comportamento é semelhante, mas percebe-se que o número de colisões só diminui efetivamente a partir de $fr = 60$. Não cabe aqui uma comparação direta entre os métodos, pois o conjunto de testes para definição de parâmetros é limitado.

Tabela 4.4: Número de colisões para os métodos de navegação do FIRN em relação a fr

Nº de colisões (Mét. de nav. x fr)	$fr = 20$	$fr = 30$	$fr = 40$	$fr = 50$	$fr = 60$	$fr = 70$
DSM3	26	22	22	22	20	20
DSM7	28	23	22	19	16	13
FDSM	23	23	23	22	17	13

A Tabela 4.5 apresenta os resultados referentes à qualidade do caminho para cada método de navegação aplicado com um valor diferente de fr. Entre $fr = 20$ e $fr = 70$, o método DSM3 sofre uma queda de 8.46% na qualidade, o método DSM7 uma queda de 15.56% e o método FDSM uma de 11.38%. Comparando as Tabelas 4.4 e 4.5, a relação entre o número de colisões e a qualidade do caminho é: quanto maior fr, menor o número de colisões e menor a qualidade do caminho.

Tabela 4.5: Qualidade do caminho para os métodos de navegação do FIRN em relação a fr

Qual. do caminho (Método x fr)	$fr = 20$	$fr = 30$	$fr = 40$	$fr = 50$	$fr = 60$	$fr = 70$
DSM3	99.18%	97.19%	95.62%	94.46%	91.87%	90.72%
DSM7	98.61%	96.18%	93.40%	90.85%	87.96%	83.05%
FDSM	98.93%	97.33%	95.55%	93.61%	92.66%	87.55%

Para compatibilizar com os valores adotados para as APFs o ideal é que fr seja escolhido como 40. De acordo com as Tabelas 4.4 e 4.5, não existem muitas variações entre $fr = 30$ e $fr = 50$ e estes valores representam a melhor relação entre o número de colisões e a qualidade do caminho. Não há, então, problemas em definir $fr = 40$ para todos os métodos.

Resumidamente, os parâmetros definidos são então:

- APF_k: $\alpha = 1$, $\beta = 4$ e $sp = 40$;
- APF_p: $\xi_q = 3$, $\xi_v = 0$, $\xi_a = 0$, $i = 1.4$, $j = 0$, $k = 0$, $\eta_1 = 50$, $\eta_2 = 50$, $\eta_3 = 50$ e $sp = 40$;
- DSM3 e DSM7: $w_f = +\infty$ e $fr = 40$;
- FDSM: $fr = 40$;

4.4.3 Desempenho do FIRN para Vários Obstáculos

Com o Simulador FIRNn preparado para lidar com vários obstáculos e a motivação de avaliar o FIRN em seu limite, foi realizado um experimento onde um IR, navegando de acordo com o *framework*, é confrontando com cenários de navegação contendo números

progressivamente maiores de MOs. O número máximo de MOs em um grupo é nove, devido a limitações no processamento do próprio simulador.

Foram elaborados conjuntos de 120 cenários de teste para cada grupo de MOs, totalizando 1080 cenários. O IR utiliza o método de previsão E-LOFORM para prever o movimento dos obstáculos e cada método de planejamento é executado, utilizando os parâmetros definidos na Seção 4.4.2. No total são 3240 simulações (3 métodos de navegação x 1080 casos de teste).

Para todos os cenários de teste, a posição inicial do IR é definida como sendo (50, 200) e a posição do objetivo como (750, 200). A Figura 4.12 fornece a base para a elaboração dos cenários com relação à posição inicial e direção dos MOs. O IR está fixado em sua posição inicial e o espaço de simulação foi subdivido em diversas pequenas áreas, numeradas de 1 a 16, onde os MOs podem ser alocados. Quando um MO é inserido em uma área deste tipo, ele pode tomar qualquer posição dentro dos limites da área, mas deve ter sua direção apontada para o IR, garantindo um conflito na simulação. As áreas "L1" até "L12" são mais limitadas, já que a posição dos MOs só pode ser deslocada no eixo x.

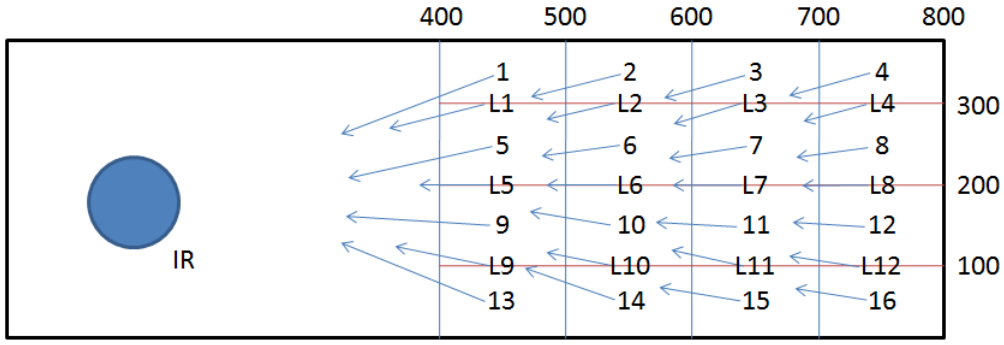

Figura 4.12: Esquema geral para geração de cenários de teste. Os MOs são distribuídos nas regiões numeradas de 1 a 16 e de "L1" até "L12". As setas indicam a direção do movimento

Tomando como exemplo o grupo de cenários de teste para dois obstáculos, a Figura 4.13 ilustra o esquema desenvolvido. São três sub-esquemas, cada um representando uma visão simplificada do esquema geral da Figura 4.12. Como para cada grupo de MOs são elaborados 120 cenários de teste, então de cada sub-esquema são derivados 40 cenários. Na Figura 4.13.a existem duas setas, uma passando pelas áreas 5 a 8 da Figura 4.12 e outra

passando pelas áreas L8 a L5. Isto significa que, dos 40 cenários construídos utilizando este sub-esquema, nos 10 primeiros existe um MO localizado na área 5 e um MO localizado na área L8; nos 10 seguintes existe um MO na área 6 e um MO na área L7; e assim por diante, seguindo a direção das setas. Os outros 80 cenários são definidos através dos sub-esquemas expostos nas Figuras 4.13.b e 4.13.c. O processo é repetido para todos os 1080 cenários, cada conjunto de 120 representando um grupo com i MOs, onde $1 \geq i \geq 9$. Os esquemas de todos os conjuntos podem ser encontrados no Apêndice B, sempre tomando como base as áreas estabelecidas na Figura 4.12. O aspecto geral dos esquemas abordados no Apêndice B deixa claro que um conjunto de experimentos contemplando i MOs é independente de um conjunto contemplando $i - 1$ MOs. Ou seja, os conjuntos subsequentes não são gerados apenas adicionando-se um novo MO em uma configuração anterior. Isto foi feito a fim de explorar situações interessantes, dado o número de MOs corrente.

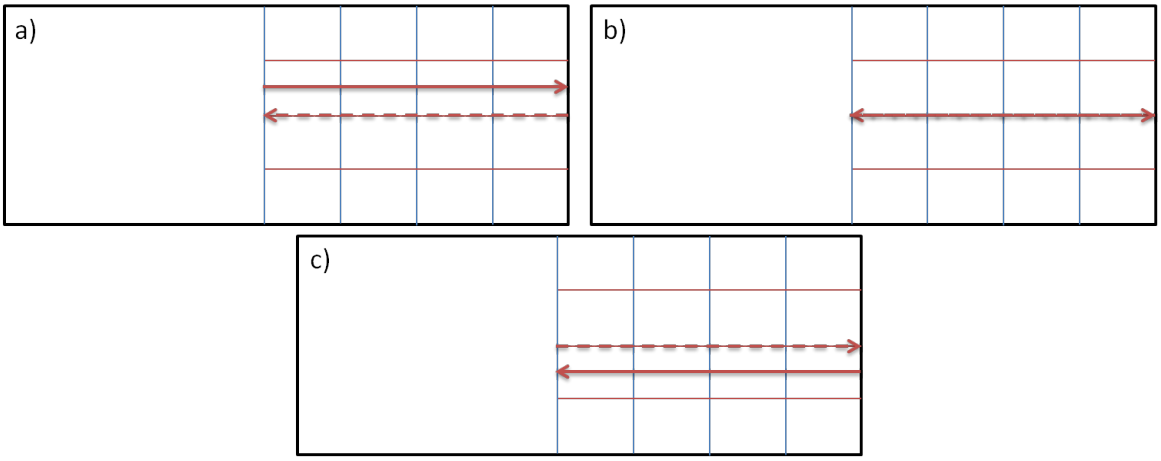

Figura 4.13: Esquema para geração de cenários de teste para dois obstáculos. Cada sub-esquema (a, b, c) é utilizado para gerar 40 cenários de simulação. As setas indicam a área ocupada por cada MO (cada seta ocupa quatro áreas, onde o MO estará em 10 dos 40 cenários). Estes sub-esquemas são representações simplificadas do esquema da Figura 4.12

Assim como nos cenários construídos para selecionar os melhores atributos para a estratégia APF_k, os cenários de simulação desenvolvidos permanecem constantes durante todos os experimentos. Ou seja, todos os experimentos expostos nesta seção são realizados em 1080 cenários de simulação previamente definidos, sem alterações de posição ou padrão

de movimento dos MOs após a atribuição inicial. Esta é provavelmente a única maneira de assegurar o aspecto comparativo entre as estratégias.

O objetivo deste experimento é verificar o impacto do número de MOs nos tempos de previsão e planejamento e a taxa de colisões (cenários onde existe colisão entre o IR e um MO) na aplicação de cada método de planejamento. As APFs não foram consideradas neste momento, limitando-se a análise aos métodos do FIRN.

A Figura 4.14 mostra o tempo de previsão alcançado ao aplicar o E-LOFORM antes de cada método de planejamento para todos os conjuntos de MOs. O que é mostrada é a média do tempo alcançado pelo E-LOFORM para cada conjunto de cenários contemplando um certo número de MOs (120 casos para cada conjunto). Primeiramente, fica claro o aspecto modular do FIRN: como o processamento da previsão é independente do método de planejamento, o tempo de previsão é o mesmo para todos os três. Ele cresce linearmente com o número de obstáculos e o cálculo é simples: o tempo de inicialização do método e o processamento da previsão de um MO tomam aproximadamente 0.03s e cada amostra adicional derivada de um MO adiciona aproximadamente 0.015s ao tempo total. Ao processar nove obstáculos, o E-LOFORM toma aproximadamente 0.16s o que atrasa o IR em dois passos, ou seja, quando o IR inicia a navegação de fato, os MOs já estarão dois passos à frente que o esperado no simulador. O resultado obtido para o conjunto de cenários com um obstáculo difere em relação ao encontrado na Tabela 4.1, sendo o tempo de previsão cerca de cinco vezes maior no Simulador FIRNn do que no Simulador FIRNp. Isto se dá devido à plataforma utilizada, o Matlab, cuja execução é mais pesada que um executável gerado em Java.

Na Figura 4.15 consta o tempo de planejamento para cada método de navegação aplicado a cada conjunto de MOs. Mais uma vez, é mostrada a média alcançada por cada método de planejamento ao ser aplicado a cada conjunto de cenários contemplando um certo número de MOs. Ao analisar a performance de cada método de forma independente, o tempo de planejamento aumenta linearmente, com taxas de crescimento da ordem de 10^{-3}s para os métodos DSM3 e FDSM e da ordem de 10^{-2}s para o DSM7 a cada MO introduzido na simulação. No entanto, o tempo base gasto para o processamento de um único MO já é elevado em comparação ao tempo de previsão, sendo de 0.7s para o método DSM7. O MO está cerca de sete posições à frente do que o esperado na etapa de navegação.

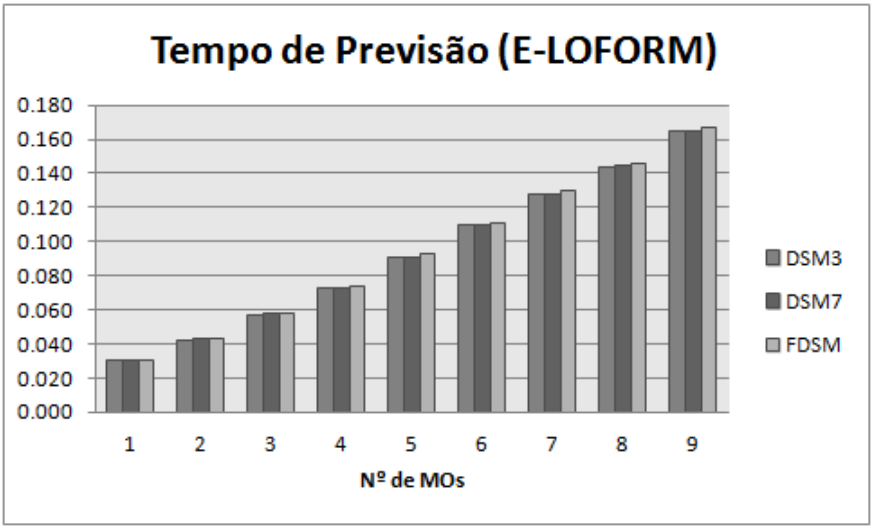

Figura 4.14: Tempo médio de previsão para o E-LOFORM antes da aplicação de cada método de planejamento, considerando-se cada grupo de n MOs (nº de MOs x tempo (s)).

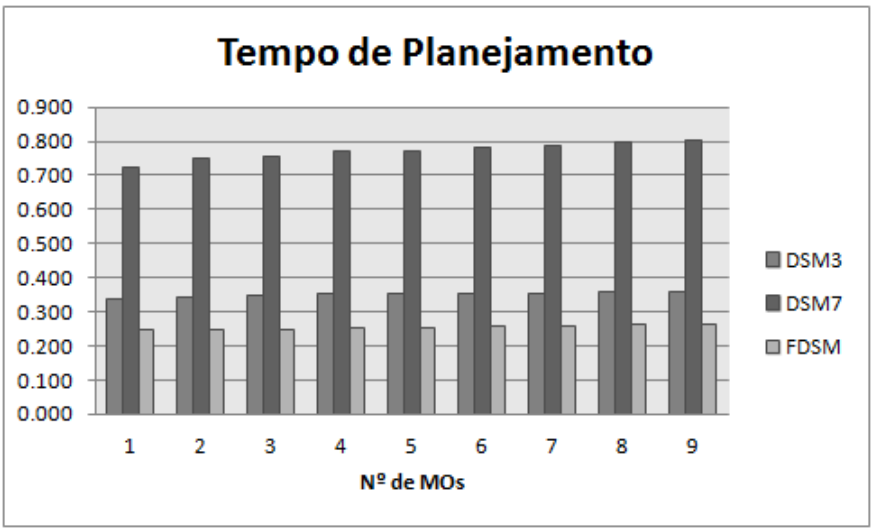

Figura 4.15: Tempo de planejamento para os métodos de planejamento do FIRN, considerando-se cada grupo de n MOs (nº de MOs x tempo (s))

Ainda na Figura 4.15 fica clara a diferença entre o tempo de processamento do método DSM7 e do método DSM3. Faz-se então necessária uma análise da complexidade que direções adicionais trazem ao método: existem dois ciclos do tipo "enquanto" aninhados no Algoritmo 4 do Apêndice A desta dissertação. Um dos ciclos corresponde ao número de passos que o IR tem de tomar para chegar ao objetivo e o segundo ao número de direções a serem consideradas no planejamento. Por sua vez, o acréscimo de MOs adiciona apenas cálculos de distância e somas na heurística (Fórmula (3.10)), que são operações simples do ponto de vista computacional. Dado que o DSM7 tem mais que o dobro de direções possíveis a serem tomadas, o que influencia diretamente o segundo ciclo do algoritmo 4 e o número de vezes que a heurística é considerada, é natural que o tempo de planejamento seja o dobro do método DSM3. O acréscimo de MOs, no entanto, tem um impacto muito pequeno. O método FDSM possui o menor tempo de planejamento, já que é feito apenas um acesso ao controlador nebuloso por ciclo do planejamento e, como o método considera apenas o MO mais próximo na avaliação de distância, não é influenciado diretamente pela quantidade de obstáculos no ambiente.

A Figura 4.16 mostra o número de simulações onde ocorre colisão entre o IR e o MO ao empregar cada método de planejamento. Ainda que seja observável uma tendência no crescimento de colisões com o número de MOs, existem conjuntos destoantes: os cenários de simulação contemplando cinco e seis MOs apresentam menos dificuldades para o IR na navegação do que os cenários com quatro. Isto acontece dada a forma como os conjuntos foram construídos: experimentos contemplando um número i de MOs não são construídos com base nos cenários de $i-1$ MOs. Desta forma, as situações enfrentadas pelo IR ao navegar nos cenários com cinco ou seis MOs são completamente distintas daquelas enfrentadas nos cenários com quatro MOs. Este detalhe não é importante para a análise, já que a diferença de colisões entre os cenários com um único obstáculo daqueles contemplando nove obstáculos é visível e comprova a deficiência do FIRN atual para vários obstáculos: são aproximadamente 80 colisões a mais, chegando a um total de 108 colisões em 120 casos para o DSM7, um número alto e indesejável. Dado que a previsão é correta e o IR tem espaço suficiente até os MOs para completar o processamento, são três fontes possíveis para o problema: os métodos de previsão e planejamento estão tomando tanto tempo no processamento que durante a etapa

de navegação os MOs já estão muitos passos à frente do esperado e a previsão e planejamento perdem sua validade; os métodos de planejamento não estão utilizando adequadamente a previsão para processar a rota; o ambiente de simulação é desafiador, já que a velocidade dos obstáculos pode ser maior que a do IR em alguns instantes. Será feita agora uma avaliação destas três fontes de problemas.

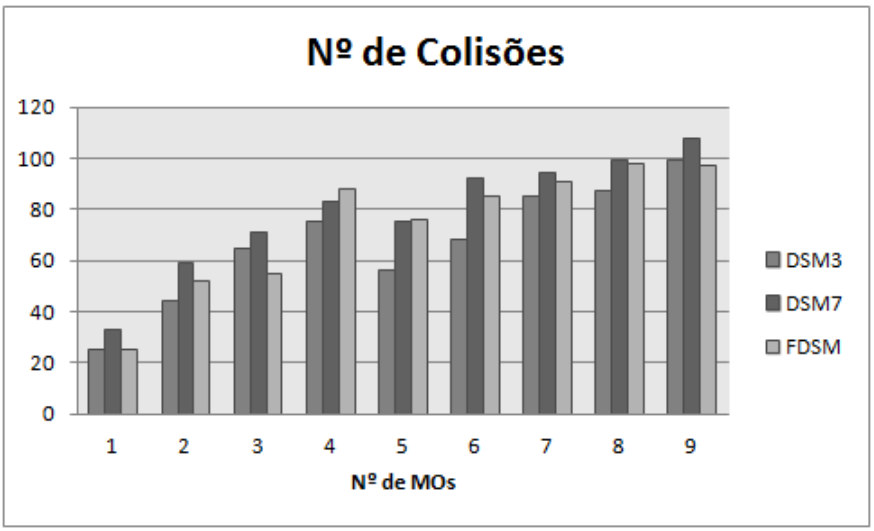

Figura 4.16: Número de colisões para os métodos de planejamento do FIRN, considerando-se cada grupo de n MOs (nº de MOs x nº de colisões)

Como mostra o gráfico da Figura 4.16, o DSM7 gera mais colisões que o DSM3 em todos os conjuntos de MOs. A Figura 4.17 mostra um exemplo da navegação do IR utilizando as rotas geradas pelos métodos DSM3 e DSM7 no mesmo cenário de simulação. O tempo de processamento do DSM3 foi de 0.40s e o do DSM7 de 0.72s. Na Figura 4.17.a, o IR consegue passar pelo MO sem colidir, enquanto que o DSM7 na Figura 4.17.b colide. O tempo de processamento do DSM7 faz com que o MO esteja sete passos a frente do esperado. Se o mesmo estivesse ainda na posição mais abaixo no eixo y, como era esperado pela previsão, o IR conseguiria realizar o desvio sem colidir. No entanto, o atraso no processamento fez com que o IR iniciasse a navegação tardiamente. Dado que a liberdade de movimento do DSM7 é maior às custas do tempo de planejamento, a vantagem do DSM3 indica que a troca não foi vantajosa, sendo, deste ponto de vista, melhor investir em um menor tempo de planejamento.

Este atraso derivado do tempo de processamento explica apenas a performance do DSM7 com relação à sua versão de três de direções e não o alto índice de colisões geral.

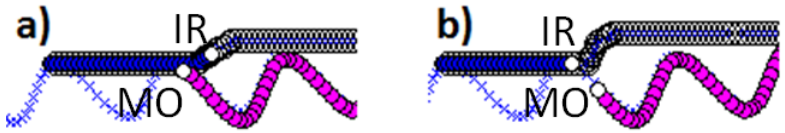

Figura 4.17: Efeitos do tempo de planejamento na navegação: a) DSM3 ; b) DSM7. Em b), o IR colide com o MO pois o tempo de processamento foi muito longo, invalidando o planejamento

A Figura 4.18 ilustra uma ocorrência comum quando o IR é confrontado por um MO se movendo nos padrões Onda ou Oito. O IR já está na etapa de navegação, adotando a rota retornada pelo planejamento e o MO está se movendo no padrão Onda, sendo a previsão correta neste caso. Dado que a seleção de uma direção nos métodos DSM3, DSM7 e FDSM é feita considerando-se apenas a posição prevista do MO em um instante de tempo tf, o IR não tem noção da direção que o MO está tomando e planeja apenas a melhor direção naquele instante. Durante a previsão, o MO estaria acima do IR no eixo y, na crista da Onda. Como o IR utilizando o DSM3 pode selecionar apenas três direções, a que mais o distância do MO é aquela em um ângulo de -45º com a horizontal. Mas a projeção do caminho do MO mostra que este começa a descer da crista da onda e o IR, limitado a três direções e sem um mecanismo de replanejamento, não tem nenhuma opção a não ser continuar seu planejamento para baixo tentando reduzir a heurística atual da melhor maneira possível. No FDSM, o IR pode adotar qualquer direção entre -90º e 90º com a horizontal, mas como as previsões futuras não são observadas e a velocidade do MO é elevada, o IR mais uma vez não tem opções a não ser continuar seu caminho para baixo. O ideal seria que os métodos observassem a direção do movimento do obstáculo e se antecipassem, passando por cima da onda.

Para verificar a dificuldade oferecida pelo ambiente, é necessário consultar o desempenho das estratégias baseadas em APFs, o que será feito na Seção 4.4.4.

Uma análise a ser feita sobre a Figura 4.16 é sobre o desempenho geral inferior do FDSM com relação ao DSM3. O método FDSM consulta um controlador nebuloso para

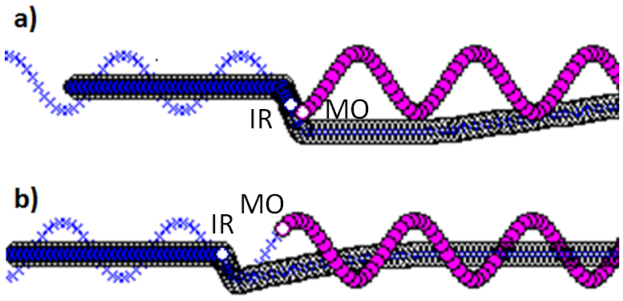

Figura 4.18: Subutilizacão da previsão pelos métodos de planejamento: a) DSM3 ; b) FDSM. Os métodos consideram apenas a configuração estática dos MOs em um instante de tempo tf para planejar uma posição para o IR, sem considerar o movimento futuro dos mesmos

definir a próxima direção do movimento dada a configuração atual do ambiente, contando com maior liberdade de direções e gastando um menor tempo de processamento (Figura 4.15). Inicialmente, o controlador também retornaria a velocidade ideal para o IR a cada consulta, mas não foram encontrados critérios para regular a mesma; como o IR deve se movimentar com velocidade máxima em direção ao objetivo na ausência de obstáculos, a redução se daria apenas quando MOs são encontrados. No entanto, como o controlador nebuloso sabe apenas a posição prevista dos obstáculos a cada instante, não há meios de descobrir quando uma redução de velocidade evitaria uma colisão. Ou seja, a direção do MO não é derivada da previsão e o controlador nebuloso não teria como encontrar situações onde diminuir a velocidade. Além disso, diferente do DSM3, o FDSM considera apenas o MO mais próximo ao definir a próxima direção a ser tomada, o que pode levar a colisões inesperadas se dois ou mais MOs estão se movendo muito próximos uns dos outros, justificando sua performance não satisfatória.

Somando o número de colisões, o DSM3 levou o IR a colisões 604 vezes num total de 1080 cenários, o DSM7 em 714 vezes e o FDSM em 667 vezes.

4.4.4 FIRN x APFs

Analisados os resultados gerais obtidos com o FIRN, agora são consideradas as estratégias APF_k e APF_p. O objetivo é o de comparar a performance das estratégias baseadas em APFs com as estratégias aplicadas no FIRN. O último fator que justifica o alto índice de

colisões sofrido pelo IR navegado segundo o FIRN é a complexidade do ambiente, que será avaliada de acordo com os resultados obtidos pelas estratégias comparativas. Os atributos utilizados para os métodos são aqueles descritos na Seção 4.4.2. As velocidades do robô e dos MOs são as mesmas dos experimentos realizados anteriormente. Como as estratégias baseadas em APFs não possuem tempo de previsão e planejamento, as únicas métricas consideradas são a taxa de colisão e a qualidade do caminho com relação a uma linha reta do ponto inicial ao objetivo (quanto mais próximo de 100% for este valor, mais otimizado foi o caminho). A qualidade do caminho é tomada apenas em simulações onde o robô alcança o objetivo sem colisões, já que em caso contrário o caminho não é traçado completamente. Para simplificar a análise dos resultados, as APFs são comparadas apenas ao método DSM3 do FIRN, pois este foi o que obteve menor número de colisões no geral.

O gráfico da Figura 4.19 apresenta o número de colisões sofridas pelo robô ao navegar com cada estratégia. As estratégias APF_k e APF_p sujeitam o robô a 45 e 57 colisões respectivamente no conjunto de cenários contemplando nove MOs. Visto que os números representam aproximadamente 42.5% de colisões do total de 120 cenários, as estratégias também encontram dificuldades na navegação, devido principalmente à velocidade elevada tomada pelos MOs com relação ao robô. No entanto, ao conferir o resultado com relação à estratégia empregada pelo FIRN, existe uma grande vantagem: o método DSM3 do FIRN sujeita o IR ao dobro de colisões que as estratégias baseadas em APFs sujeitam o robô, quando cada conjunto de MOs é examinado individualmente. Isto sugere que, além da dificuldade apresentada pelo ambiente, o tempo de processamento e a qualidade do planejamento estão afetando negativamente a performance do *framework*.

Com relação apenas às estratégias baseadas em APFs, o desempenho depende do número de obstáculos no ambiente. A estratégia APF_k obtém melhor resultado quando existem mais de cinco MOs no ambiente, enquanto que a estratégia APF_p obtém melhor resultado nos demais conjuntos. A Figura 4.20 ilustra um cenário de navegação com apenas um obstáculo, onde um robô navega com as duas estratégias. Para a estratégia APF_k, o robô é influenciado somente pela distância até o MO, o que faz com que o robô se aproxime muito do MO até iniciar as manobras de evasão. Quando o campo de repulsão começa a influenciar o robô de fato, o MO já está muito próximo e o robô manobra em sua velocidade

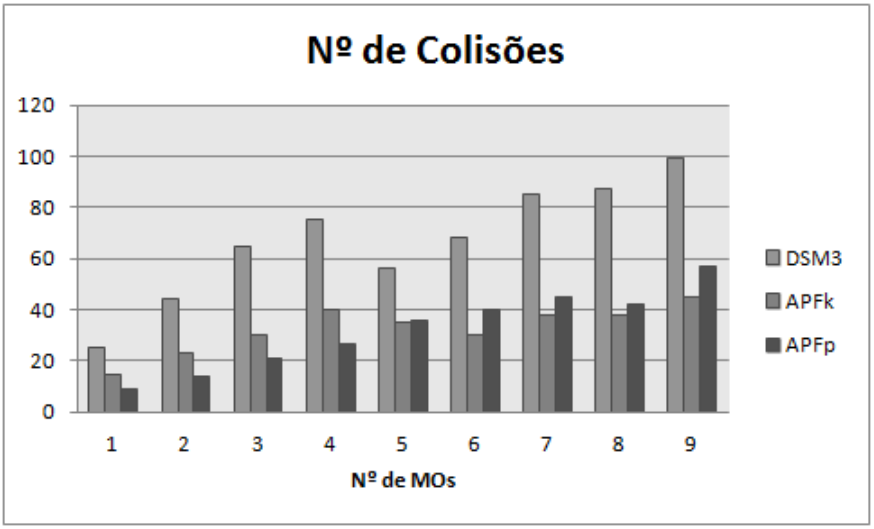

Figura 4.19: Número de colisões para o DSM3 e as APFs, considerando-se cada grupo de n MOs (nº de MOs x nº de colisões)

máxima, recuando sempre que o campo de repulsão se torna muito intenso. No momento que o MO se encontra no vale do Oito, o campo de atração acaba por atrair o robô em direção ao objetivo, mas como o MO começa a subir de forma súbita e em alta velocidade, o robô não tem velocidade suficiente para escapar. A estratégia APF_p é capaz de regular a força dos campos de repulsão considerando também a velocidade e aceleração tomadas pelo MO. Assim sendo, o robô diminui sua velocidade ao se aproximar do MO, e inicia seu desvio com velocidade positiva no eixo y quando detecta que o MO está projetando uma velocidade negativa no eixo y. Isto faz com que o robô já comece a se distanciar do obstáculo. Quando o robô percebe que o MO está mudando de direção no eixo y, ele aumenta sua velocidade a fim de estar a uma distância segura quando o MO atinge a crista do formato Oito e pode ultrapassar o obstáculo sem colisões. A Figura 4.21 ilustra um cenário de navegação com nove obstáculos (nem todos os MOs estão representados nas imagens) com o robô navegando de acordo com as duas estratégias. Como os campos de repulsão da estratégia APF_k dependem apenas da distância entre o robô e o MO, o robô acaba desviando com mais intensidade ao encontrar grupos de obstáculos aglomerados (os campos de repulsão são somados), evitando áreas de alto risco naturalmente. No caso da estratégia APF_p, os campos de repulsão também são regulados pela velocidade relativa dos obstáculos com o robô. Desta forma, a repulsão

em alguns casos não é tão forte fazendo com que o robô, ao perceber que os MOs não estão vindo em sua direção no momento, se insira dentro de espaços perigosos, tentando encontrar uma posição segura de estacionamento até que os MOs o ultrapassem. Como a velocidade dos MOs pode ser maior que a do robô na maioria dos casos, ele acaba se chocando quando um dos MOs faz uma curva inesperada em alta velocidade.

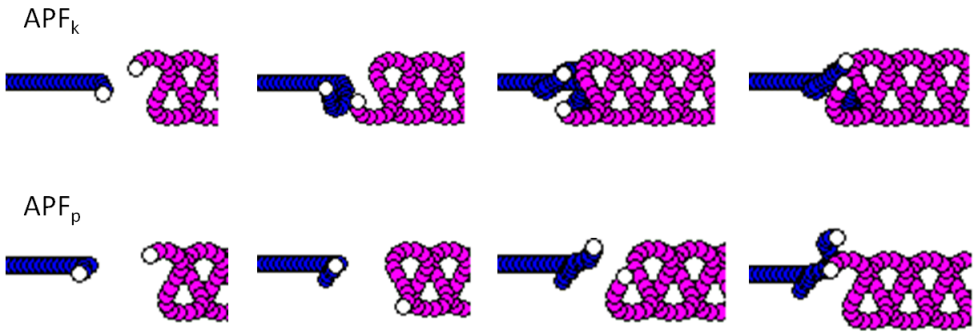

Figura 4.20: APF$_k$ x APF$_p$ em um cenário com um obstáculo. A estratégia APF$_k$ colide o IR pois reage apenas de acordo com a distância entre o IR e o MO. A estratégia APF$_p$ evita a colisão pois dirige o IR em uma direção contrária àquela delineada pelo MO, antecipando a evasão

O gráfico da Figura 4.22 mostra os resultados obtidos por cada estratégia no que se refere à qualidade média do caminho para cada conjunto de MOs, excluindo-se os casos onde existe colisão. O DSM3 alcança um resultado melhor que as APFs em todos os conjuntos de MOs. No entanto, não se pode concluir que o DSM3 seja de fato melhor nesta característica: como a qualidade do caminho é tomada apenas em casos de sucesso e o DSM3 colide nos cenários de simulação mais complicados, a qualidade do DSM3 é tomada baseando-se apenas em casos simples, onde o IR não precisa realizar muitos desvios para alcançar o objetivo. As estratégias baseadas em APFs conseguem alcançar o objetivo mesmo quando a configuração dos obstáculos induz vários desvios na rota, o que justifica a qualidade geral menor no caminho. Com isso, a métrica de qualidade do caminho só pode ser observada relacionada com o número de colisões. Por exemplo, é possível afirmar que a estratégia APF$_k$ obtém melhor desempenho geral que a estratégia APF$_p$ nos conjuntos contemplando cinco a nove obstáculos já que, além de conseguir um menor número de colisões (Figura 4.19), retorna rotas de melhor qualidade (Figura 4.22).

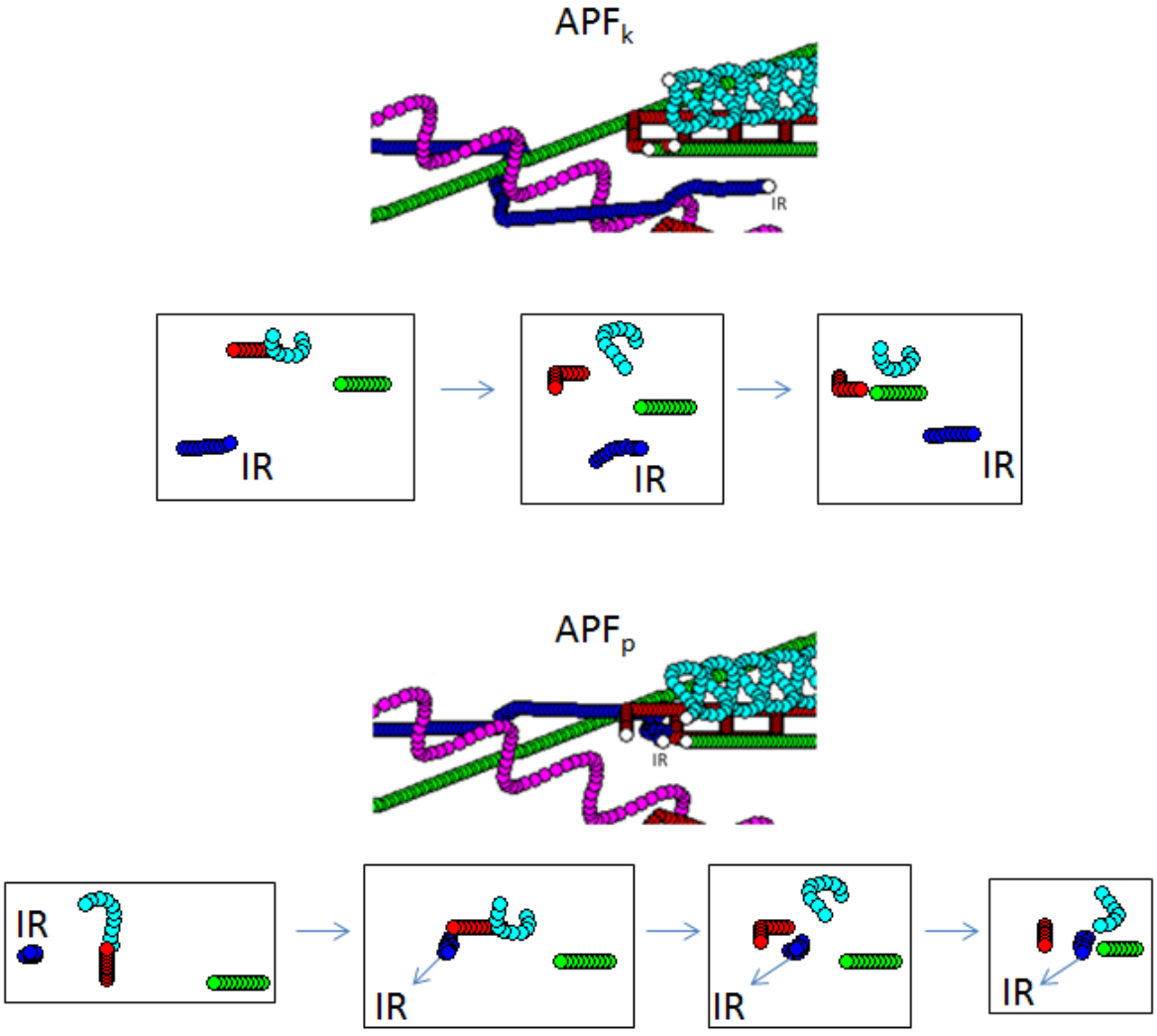

Figura 4.21: APF$_k$ x APF$_p$ em um cenário com nove obstáculos. A estratégia APF$_k$ evita a colisão pois a soma dos campos artificiais emitidos pelos obstáculos acaba por repelir o IR. Na estratégia APF$_p$ existe colisão pois alguns MOs estão se movimentando na mesma direção do IR, o que faz com que este entre em áreas perigosas. Em destaque estão as situações como vistas no simulador. Abaixo das mesmas, capturas de telas sequenciais ilustrando o ocorrido

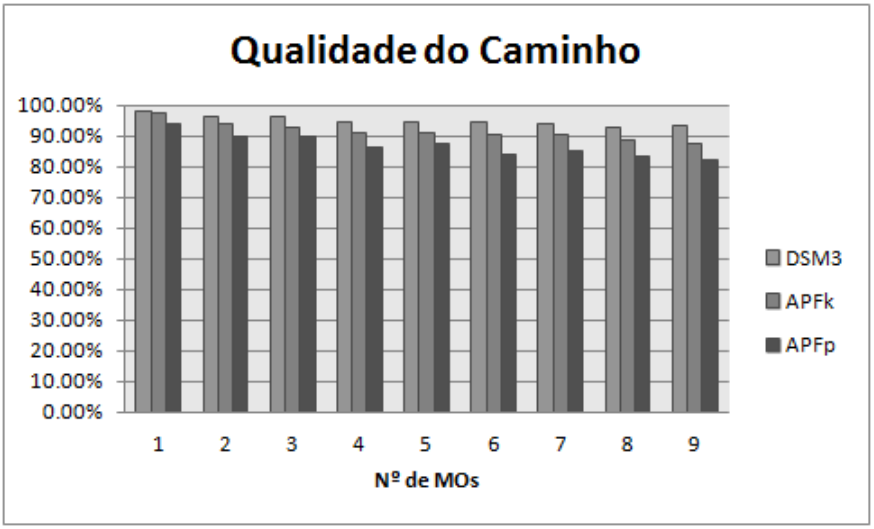

Figura 4.22: Qualidade do caminho média para o DSM3 e as APFs, considerando-se cada grupo de n MOs (nº de MOs x nº de colisões (nº de MOs x qualidade)

Somando o número de colisões, a estratégia APF_k sujeitou o robô a 294 colisões num total de 1080 cenários, e a estratégia APF_p a 291 colisões.

Neste Capítulo foram apresentados os simuladores utilizados para medir a performance dos métodos de previsão e planejamento do FIRN. O Simulador FIRNp é capaz de lidar apenas com um obstáculo e os experimentos de corretude da previsão nele realizados revelaram que o método E-LOFORM é capaz de prever o padrão dos obstáculos no ambiente de simulação em tempos da ordem de 10^{-3}.

O Simulador FIRNn é capaz de lidar com vários obstáculos simultaneamente e resolve algumas das deficiências encontradas no Simulador FIRNp. Foram realizados experimentos medindo o impacto do tempo de previsão e planejamento para a navegação no contexto de vários obstáculos. O tempo de previsão cresce linearmente com o número de MOs no ambiente, alcançando um tempo máximo de 0.160s ao lidar com grupos de 9 MOs. Os métodos de planejamento apresentaram tempo de processamento com crescimento linear muito baixo,

mas com um tempo inicial para cenários com um MO já elevados. Em relação ao número de colisões, ainda que os resultados tenham sido críticos, o DSM3 teve a melhor performance com relação aos métodos DSM7 e FDSM, colidindo o IR 604 vezes nos 1080 cenários construídos, o que corresponde a uma taxa de 55.9% de colisões. As deficiências levantadas foram a subutilização da previsão, o tempo de processamento elevado, especialmente no método DSM7, e a dificuldade do ambiente, onde os obstáculos se movem com velocidades elevadas.

Nos experimentos comparativos com as estratégias baseadas em APFs, a estratégia APF_k apresentou uma taxa total de colisão de 27.22% e a estratégia APF_p uma taxa total de 26.9%. No entanto, a estratégia APF_k obteve melhor performance em cenários com vários MOs, o que sugere uma melhor adaptabilidade desta com relação à estratégia APF_p para o ambiente de aplicação. Independente deste resultado, a taxa de colisões das estratégias comparativas foi 28.7% menor que as estratégias aplicadas no FIRN, evidenciando as deficiências atuais do *framework*. Nada pôde ser comprovado sobre à qualidade do caminho do FIRN em relação às estratégias baseadas em APFs.

5 NOVOS MÉTODOS DE PLANEJAMENTO

Pelos resultados obtidos no Capítulo 4, fica clara a deficiência dos métodos de planejamento em comparação às estratégias baseadas em APFs. Os problemas levantados foram:

1. Subutilização das informações geradas pela etapa de previsão por parte dos métodos de planejamento, onde apenas uma previsão é consultada em cada instante tf;
2. Número limitado de direções disponíveis para o IR nos métodos de planejamento DSM3 e DSM7, restringindo a capacidade de movimentação;
3. Restrição do FDSM na identificação de obstáculos, tendo um controlador nebuloso que considera apenas o obstáculo mais próximo para decidir qual direção tomar;
4. Tempo de processamento elevado, em especial aquele despendido durante a etapa de planejamento, o que faz com que a previsão e a rota estejam inválidas quando a navegação de fato se inicia;
5. Velocidade constante empregada pelo IR em todos os métodos;
6. Dificuldade apresentada pelo ambiente, com obstáculos com velocidades maiores que o robô e mudanças constantes na direção;

Para solucionar o problema 1, o comportamento dos métodos de planejamento tem de ser reconsiderado: atualmente, os métodos consultam cada membro das listas mo_{tri} de forma isolada, aplicando a melhor direção possível para cada configuração estática prevista. Dado que o planejamento não é recalculado quando uma situação de perigo é identificada,

a navegação funciona exatamente da mesma forma que um método reativo aplicando uma estratégia gulosa. Na Figura 5.1 o problema fica claro: o efeito da aplicação da estratégia E-LOFORM + DSM3 é exatamente o mesmo que navegar o robô em tempo real utilizando a heurística da Fórmula (3.10). O mesmo comportamento é válido para os métodos DSM7 e FDSM. O planejamento tem que extrair informações relevantes da previsão, como a direção geral do MO, ou reavaliar o planejamento ao se deparar com situações como a da Figura 3.10. Inicialmente, na aplicação do DSM3 no Simulador FIRNp, o método realizava uma busca intensiva para determinar o melhor caminho em volta de um obstáculo o que evitava estas situações, como ilustra a Figura 3.10. No entanto, esta busca seria muito taxativa no contexto de vários obstáculos, sendo então mais vantajoso explorar as informações relevantes na previsão antes de iniciar o planejamento.

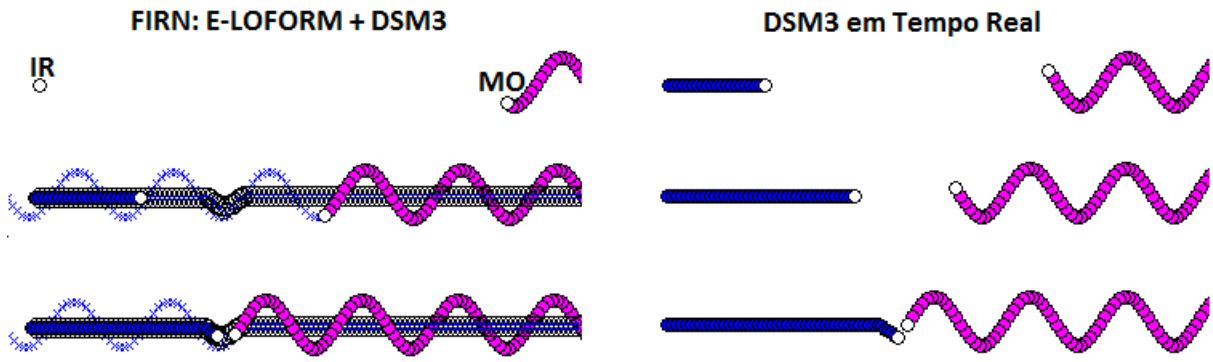

Figura 5.1: No método DSM3 a previsão é subutilizada, resultando em uma navegação puramente reativa

Os problemas 2 e 3 são deficiências específicas dos métodos DSM e FDSM. Para o FDSM, o controlador nebuloso aplicado teria de ser reestruturado, a fim de possibilitar o tratamento de vários obstáculos simultaneamente a cada passo do planejamento. Para o método DSM, o processo de tomada de decisão teria que ser completamente reformulado, eliminando a restrição do movimento. O problema 4 é influenciado diretamente pelo comportamento do controlador nebuloso ou do processo de tomada de decisões do DSM3. Dado o interesse desta pesquisa em apontar novas soluções para o FIRN (e não em uma extensão de um modelo nebuloso), optou-se por investir na solução do problema 2. Independente do resultado, o tempo de planejamento ideal é algo entre aquele alcançado pelo DSM3 e aquele

alcançado pelo DSM7, considerando-se que a complexidade certamente será maior do que aquela empregada no método DSM3.

O problema 5 requer informações da previsão para que seja solucionado. Não existem meios eficientes de regular a velocidade considerando apenas a distância relativa entre o IR e os obstáculos. A partir do momento que alguma métrica de direção ou velocidade dos obstáculos for derivada da previsão, algumas possibilidades nestes sentido podem ser consideradas.

O problema 6 é uma situação determinada pelo ambiente de aplicação e não pode ser solucionada a partir do IR.

Neste capítulo são apresentados novos métodos de planejamento para o FIRN, a fim de resolver os problemas 1, 2, 4 e 5 apontados. Eles são baseados no DSM3 e cada uma das restrições é eliminada progressivamente. Em seguida, experimentos são realizados a fim de atestar os impactos das otimizações, comparando o desempenho dos novos métodos com as estratégias APF_k e APF_p.

5.1 Método de Planejamento LADSM

Uma deficiência grave a ser solucionada no FIRN é a subutilização da previsão. Dado que o método E-LOFORM retorna listas mo_{lpi} com as posições esperadas de cada MO_i em instantes de tempo futuro tf, existem informações que podem ser extraídas naturalmente. Por exemplo, examinando uma série de posições subsequentes em uma lista mo_{tri} é possível definir a direção geral do movimento de um MO. No entanto, os métodos de planejamento atuais se preocupam apenas em definir posições vantajosas para o IR para cada configuração estática dos MOs, onde entende-se por configuração estática as posições $mo_{tri}(tf)$ de todos os i MOs, tal que $1 \leq i \leq n$, n é o número total de MOs e tf é um dado instante de tempo futuro.

A maneira mais simples de indicar a direção de movimento de um MO é considerar a cada passo não só a posição esperada para o instante tf, mas também um conjunto la de posições à frente (*Look Ahead*, antecipado). Na Figura 5.2.a o planejamento é realizado pontualmente no método DSM3. Como $mo_{tri}(tf)$ é uma previsão do MO_i em que a posição

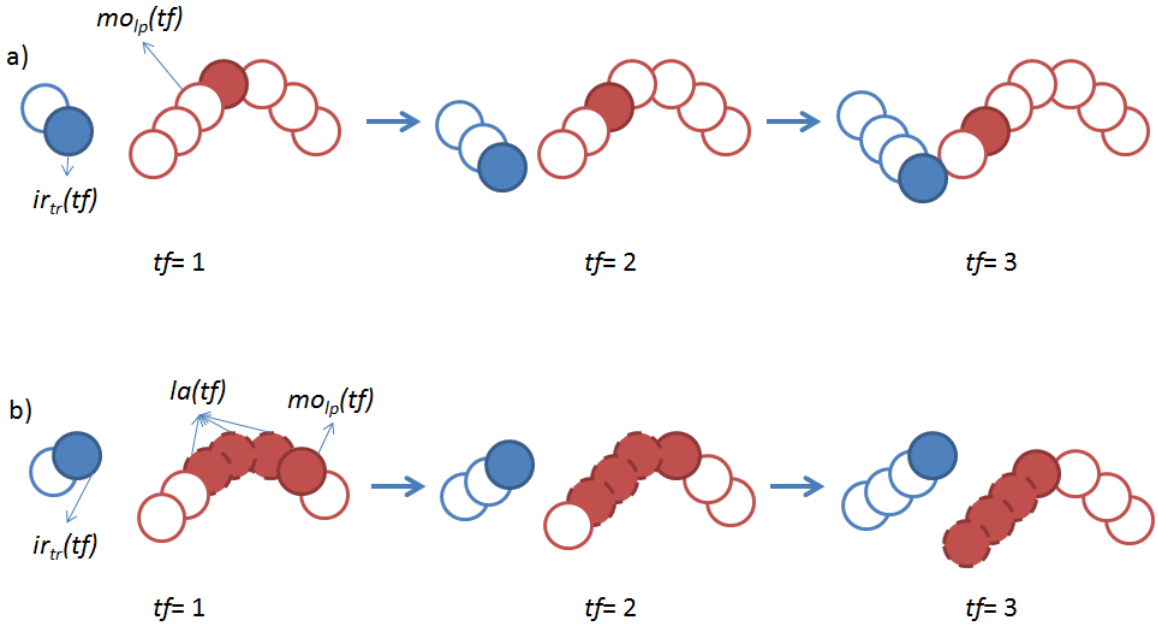

Figura 5.2: Uso da previsão: a) planejamento pontual (DSM3); b) planejamento antecipado (LADSM). No DSM3, o IR planeja a primeira posição de forma que o IR se afaste o máximo possível da posição estimada para o MO_i no tempo futuro tf. No entanto, como o MO_i está descendo no ambiente simulado, haverá uma colisão no futuro. No método LADSM, várias posições da previsão são consideradas simultaneamente, o que faz com que o planejamento se antecipe e contorne a trajetória esperada do MO_i

se encontra acima da posição planejada do IR, a direção que mais distância os dois é a de -45º para baixo. No entanto, a previsão diz que o MO_i está se deslocando para baixo e o DSM3, alheio a este detalhe, continua planejando a suposta melhor direção pro IR, que na verdade causará a colisão. Durante a navegação, a colisão acaba ocorrendo pois o IR tem velocidade total menor do que a do MO e a posição real do mesmo está defasada com relação à previsão. Na Figura 5.2.b, um conjunto $la(tf)$ de 3 posições é considerado para o instante $tf = 1$, fazendo com que a direção geral do deslocamento seja percebida antecipadamente. A janela de previsões é deslocada em $tf = 2$, confirmando a direção do MO. Dada esta situação, se a heurística do DSM3 for calculada passando como entrada não somente a posição prevista no instante tf, mas também as posições antecipadas, a direção mais vantajosa será em 45º com a horizontal, evitando a colisão. O método de planejamento LADSM (*Look Ahead Direct Search Method*) aplica exatamente este conceito.

O LADSM funciona da seguinte maneira: existem "janelas de previsão" que englobam $la + 1$ membros de cada lista mo_{tri} a partir de $mo_{tri}(tf)$, ou seja, a previsão do MO_i para um dado instante tf acrescido de la posições à frente, onde la representa a antecipação do planejamento. O método então calcula a heurística da Fórmula (5.1) considerando todas as posições previstas contempladas pelas janelas de previsão, sendo as posições antecipadas tratadas na heurísticas como se fossem MOs distintos. Quando a melhor direção para o IR é escolhida e inserida na lista ir_{tr}, as janelas de previsão são deslocadas, iniciado agora em $mo_{tri}(tf+1)$ e o processo se repete, até que a posição planejada para o IR seja a do objetivo ou as listas mo_{tri} se esgotem.

$$h = d + \sum_{i=1}^{n} \sum_{j=0}^{la} pe_{i,j}, \tag{5.1}$$
onde n é o número de MOs e la é o número de posições antecipadas

$$pe_{i,j} = \begin{cases} 0, \ se\ dist(IR,\ mo_{tri}(tf+j)) > f_r \\ (f_r - dist(IR, mo_{tri}(tf+j))) * w_f, \ se\ dist(IR,\ mo_{tri}(tf+j)) \leq f_r \end{cases} \tag{5.2}$$

Como o método DSM7 apresenta um tempo de processamento elevado e no LADSM são consideradas $n * la$ posições a mais por iteração, ficou decidido por aplicar apenas as direções disponíveis no DSM3. Uma outra otimização também é possível. Como mostra a Fórmula (5.2), a penalidade associada à posição prevista de um MO é igual a zero quando a distância entre ele e a posição planejada do IR é menor do que fr. Se não existe nenhuma previsão de obstáculo em volta da posição considerada pelo IR no planejamento, o somatório das penalidades é zero e a direção tomada é aquela que mais aproxime o IR do objetivo. O planejamento é realizado desde a posição inicial do IR e mesmo quando os obstáculos estão distantes a heurística é calculada. Esta ação consome tempo de processamento desnecessário, já que o objetivo nestes casos é diminuir a distância até o objetivo desconsiderando os obstáculos. A heurística só precisa, então, ser avaliada quando algum MO se aproxima do IR de uma distância de fr. Em todos os outros casos, o IR simplesmente projeta um caminho em linha reta até o objetivo. Esta alteração, além de diminuir o tempo de processamento, permite que o IR navegue livremente em qualquer direção na ausência de obstáculos. Esta é

a primeira medida de relaxamento no grupo de direções do IR. Os impactos das modificações no tempo de processamento e performance da navegação serão discutidos na Seção 5.5. O Algoritmo 5 do Apêndice A detalha o funcionamento do LADSM.

5.2 Algoritmos Bioinspirados

Dada a natureza bioinspirada das soluções existentes no FIRN, contando com métodos de previsão e planejamento baseados em Redes Neurais Artificiais e Lógica Nebulosa, e o interesse de expansão da pesquisa neste assunto, um estudo foi realizado com o objetivo de encontrar novas estratégias aplicáveis ao contexto do *framework*. Mais especificamente, a pesquisa foi direcionada para os princípios e aplicação de *Swarm Intelligence* (Inteligência de Enxames). Segundo ZHU e TANG (2010), *Swarm Intelligence* é o comportamento coletivo de um grupo a fim de realizar um objetivo, onde não existe um agente controlador central. O que acontece é uma troca de informações a fim de identificar e compartilhar pontos de vantagem para a exploração, que a parte disso é realizada individualmente. Este comportamento é observado em grupos de animais como formigas, abelhas, peixes, aves etc. De acordo com MILLONAS (1993), são cinco princípios que um sistema de *Swarm Intelligence* deve seguir:

- Proximidade: Os agentes de um enxame devem ser capazes de avaliar medidas de espaço e tempo, a fim de calcular os custos de uma ação em resposta ao ambiente;
- Qualidade: Os agentes devem ser capazes de calcular o valor de uma nova fonte de exploração, que no contexto dos enxames pode ser uma fonte de alimento;
- Resposta diversificada: Os agentes não devem se focar em apenas um ponto ótimo de exploração. Eles devem se distribuir, de forma que novas opções sejam encontradas, evitando que o sistema falhe por completo ao esgotar um ponto de exploração específico;
- Estabilidade: O sistema não deve ser suscetível a toda e qualquer mudança que ocorra no ambiente. Deve existir um parâmetro de impedimento, de forma que esforços não sejam gastos descontroladamente a cada novo ponto de exploração encontrado;

- Adaptabilidade: O grupo deve ser capaz de adaptar o comportamento geral quando novos pontos de exploração mais valiosos são encontrados. Este princípio é o oposto da Estabilidade e a chave para o sucesso do grupo está no equilíbrio entre estes dois;

Existem duas vertentes contemplando a aplicação de *Swarm Intelligence* no contexto de robótica: a aplicação de inteligência em grupos cooperativos de robôs e a elaboração de algoritmos de otimização da navegação, baseados na troca de informações. Dentre as vantagens encontradas neste tipo de modelagem para grupos de robôs, ZHU e TANG (2010) destacam a distribuição do processamento e organização individual, já que cada agente tem a capacidade de agir por conta própria na falta de comunicação, a robustez, devido à descentralização da inteligência, a escalabilidade, que permite a entrada e saída de novos robôs sem qualquer tipo de adaptação, a longevidade, referente à vida útil do grupo com relação a um robô individual e o baixo custo, já que cada robô não necessita de *hardware* muito sofisticado. No caso dos algoritmos de otimização, o caráter social de compartilhamento de informações é utilizado para que pontos ótimos sejam conhecidos pelo grupo, além do equilíbrio entre os princípios da Estabilidade e Adaptabilidade serem diretamente correspondentes à necessidade de evitar convergências em pontos ótimos locais. Nesta dissertação, o estudo de *Swarm Intelligence* será focado na aplicação de algoritmos na otimização de funções, com as pesquisas relacionadas a esta vertente de aplicação.

O primeiro algoritmo estudado foi o ACO (*Ant Colony Optimization* - Otimização da Colônia de Formigas), proposto por COLORNI, DORIGO e MANIEZZO (1991). Ele é baseado no comportamento de formigas na busca por fontes de alimentos próximas da colônia. Inicialmente, as formigas andam de forma aleatória pelo ambiente, deixando uma trilha de feromônios pelo caminho, a qual evapora com o tempo. Quando uma nova fonte é encontrada, a formiga que efetuou a descoberta começa a se movimentar com intenção em linha reta entre a comida e a colônia. As outras formigas são influenciadas probabilisticamente pelas trilhas deixadas, aumentando a probabilidade com a quantidade de feromônios. Visto que a formiga que se move com um objetivo reforça o feromônio pelo caminho, as outras formigas são compelidas a seguir a mesma rota. Com o aumento no número de formigas, aumenta a quantidade de feromônio deixada, elevando a popularidade da rota. A distância até a fonte de alimento também influencia na quantidade de formigas: quando a rota é muito grande, os

feromônios evaporam antes que as formigas possam completar o caminho, enquanto que rotas menores permitem maior acúmulo. Graças ao caráter probabilístico empregado quando uma formiga escolhe qual rota seguir, a exploração do ambiente acontece mesmo quando fontes de alimentos são conhecidas.

O algoritmo foi aplicado por COLORNI, DORIGO e MANIEZZO (1991) para resolver o problema do "caixeiro viajante", onde em um grafo com vértices representantes de cidades e arestas representantes de trilhas, deve-se encontrar o menor caminho que contemple todas as arestas sem repetições. Uma quantidade de formigas é distribuída nos vértices do grafo e pequenas doses de feromônio são inseridas nas arestas. Cada formiga deve escolher a melhor aresta, baseada em uma probabilidade relativa à quantidade de feromônio e à distância entre as cidades vizinhas. Constantes α e β regulam a importância dada pela formiga ao feromônio e à distância, respectivamente. A cidade visitada pela formiga é inserida em uma lista chamada "tabu", que proíbe que a mesma revisite cidades em um ciclo do algoritmo. A quantidade de feromônio nas arestas é atualizada, considerando a passagem de formigas e a evaporação. Quando a lista "tabu" de uma formiga contém todas as cidades, o menor caminho alcançado pela colônia é armazenado e o ciclo se repete, tendo agora cada aresta uma quantidade de feromônio determinada pela quantidade de formigas que ali trafegaram. Se a quantidade de formigas, valores de α e β e o índice de evaporação de feromônio forem bem definidos, o algoritmo consegue retornar uma rota de boa qualidade, na maioria das vezes ótima, com processamento rápido.

No contexto de navegação de robôs, BRAND et al. (2010) mapeiam o ambiente como um *gridmap*, onde cada célula representa um espaço equidistante. Cada célula do mapa possui um índice de feromônio associado, cujo valor é inicialmente igual a 0.1. O ponto inicial do robô é sempre na célula diagonal superior esquerda, e o objetivo na célula diagonal inferior direita. No princípio, não existem obstáculos no ambiente. Um certo número de formigas é alocado no ponto inicial e o algoritmo ACO é inicializado. Cada formiga pode dar um passo para uma das quatro células adjacentes, sendo que o algoritmo define a melhor opção avaliando a quantidade de feromônio e as probabilidades associadas. Quando uma formiga alcança uma nova célula, o nível de feromônio associado aumenta. Quando uma delas alcança o ponto objetivo, o algoritmo é finalizado e a menor rota é armazenada. Os

índices de feromônios nas células são atualizados com a evaporação e o algoritmo ACO é processado novamente até que um número de iterações máximo seja alcançado ou não existam mais melhorias significativas no caminho. A cada execução do algoritmo, existe uma chance de que obstáculos dinâmicos reconfiguráveis sejam alocados ou movidos, fazendo com que os níveis de feromônios tenham de ser reajustados. Duas possibilidades foram exploradas: uma chamada de "global", onde a cada reconfiguração os índices de feromônios de todo o mapa são reinicializados para 0.1; outra chamada de "local", onde apenas as células ao redor dos novos obstáculos têm seus níveis de feromônios reajustados. Os resultados mostram que a abordagem "local" é capaz de retornar melhores caminhos num menor número de iterações do algoritmo ACO, comprovando que o sistema é adaptativo ao ambiente.

TAN; ZHU; YANG (2009) sugerem uma estratégia híbrida de navegação envolvendo ACO e APFs para ambientes dinâmicos reconfiguráveis. O ambiente estático inicial é mapeado com um modelo especial de VGMs, onde os obstáculos e as áreas lives são delimitados por polígonos. Quando dois polígonos delimitadores de áreas livres estão conectados, um vértice artificial é mapeado no ponto médio do lado comum aos dois polígonos. Estes vértices são ligados por arestas, representando os caminhos possíveis para o robô. Nesta configuração, o algoritmo ACO é executado várias vezes e as formigas devem encontrar o melhor caminho pelas arestas do ambiente mapeado. Durante a navegação, quando um obstáculo estático inesperado é detectado pelos sensores, o robô deve utiliza APFs para desviar. Dado o mapeamento do ambiente, o robô e obstáculo estão dentro de um polígono representante de uma área livre. O robô entra neste polígono artificial por um ponto médio de entrada e deve sair pelo ponto médio de saída. Um campo de repulsão é mapeado em volta do obstáculo inesperado e um campo de atração em volta do ponto médio de saída. Quando o ponto de saída é alcançado, o robô retoma o planejamento global resultante do algoritmo ACO.

O segundo algoritmo estudado foi o ABC (*Artificial Bee Colony Algorithm* - Algoritmo da Colônia de Abelhas), proposto por KARABOGA (2005). Ele é baseado no comportamento das abelhas na procura e compartilhamento de fontes de alimento em volta da colônia. Existem três tipos básicos de abelhas que trabalham com a coleta de alimentos: as escoteiras (*scouts*), as espectadoras (*onlookers*) e as coletoras empregadas (*employed foragers*). As abelhas escoteiras são aquelas que procuram por fontes de alimento aleatoria-

mente em volta da colônia, as espectadoras são aquelas que aguardam na colônia informações sobre novas fontes de alimento e as coletoras empregadas são as que estão associadas a uma fonte de alimento, viajando entre a colônia e a fonte. Quando uma abelha coletora leva o alimento até a colônia, ela tem três opções: abandonar a fonte de alimento e se tornar uma escoteira ou espectadora; recrutar abelhas espectadoras para a fonte de alimento associada; continuar a coletar alimento sem convocar ajuda. Esta decisão é probabilística e depende da qualidade e distância do alimento, número de abelhas espectadoras disponíveis e o número de abelhas coletando alimento na fonte associada. As abelhas espectadoras esperam em uma câmara específica da colônia por abelhas coletoras que informam as características de uma fonte de alimento através de uma dança. Dependendo das características passadas pela dança, a espectadora pode decidir adotar esta fonte de alimento. Finalmente, as abelhas escoteiras representam o caráter exploratório do modelo, procurando novas fontes mesmo quando outras são conhecidas.

O algoritmo foi usado por KARABOGA (2005) para otimizar funções matemáticas complexas multidimensionais. Um número de abelhas é definido no início da simulação. Metade destas serão abelhas coletoras empregadas e a outra metade, abelhas espectadoras. Um número muito pequeno das abelhas coletoras (algo em torno de 5 a 10%) serão definidas como escoteiras. As abelhas coletoras empregadas são distribuídas na função a ser otimizada, buscando um melhor resultado em uma área limitada da mesma. As abelhas espectadoras vão sendo alocadas em regiões próximas às abelhas coletoras empregadas de acordo com a qualidade das soluções encontradas na região. As abelhas escoteiras procuram soluções ótimas aleatoriamente. O melhor resultado da colônia é conhecido por todos os membros e quando uma escoteira encontra um resultado promissor ao acaso, ela se torna uma abelha coletora empregada, as abelhas espectadoras podem trocar de posições e a abelha coletora de pior resultado se torna uma escoteira, para ocupar o lugar da anterior. Além disso, quando uma abelha coletora está em uma região pouco promissora, ela abandona o local voluntariamente e se torna uma escoteira por conta própria. O algoritmo é repetido por um certo número de iterações, e no final o melhor resultado global é retornado. Os resultados mostram que o algoritmo converge para soluções muito próximas das ótimas em três funções complexas, sendo uma delas em um espaço de 10 dimensões. O limite de iterações máximo

foi definido como sendo o número de abelhas espectadoras multiplicado pela dimensão da função aplicada.

A aplicação do ABC em estratégias de navegação, especialmente no planejamento de caminhos, não é tão comum, já que não existe uma correspondência direta entre o algoritmo e o problema do planejamento. SAFFARI e MAHJOOB (2009) aplicam ABC em um ambiente específico para planejar uma rota em um ambiente estático, onde os obstáculos são considerados polígonos convexos e o robô tem acesso a uma representação do ambiente fornecido por uma câmera perpendicular ao espaço de navegação. Inicialmente, o robô calcula uma rota em linha reta do ponto inicial ao objetivo. Certamente esta reta passará por obstáculos, então qualquer segmento da reta que ultrapasse um polígono é moldado de forma que o caminho contorne o obstáculo pela parede e depois retome o caminho em linha reta. Como todos os obstáculos são considerados polígonos convexos, o caminho acaba sendo um conjunto de segmentos de retas conectados por "pontos de quebra" (curvas com ângulos acentuados). Este caminho não é o mais otimizado e existem pontos mais afastados dos obstáculos que retornariam rotas de menor caminho. O ABC é aplicado para encontrar estes pontos. Em volta de cada ponto de quebra é delimitado um círculo de busca. As abelhas coletoras são enviadas para estas áreas e devem procurar por novos pontos de quebra mais afastados dos obstáculos. Cada abelha coletora considera dois tipos de caminho ao escolher um novo ponto na área alocada: deslocar o ponto de quebra atual para sua posição ou recalcular todo o caminho projetando uma linha reta do ponto inicial até o novo ponto de quebra e uma linha do mesmo até o objetivo, quebrando a rota em novas pontos quando as linhas passam por outros obstáculos. Considerando os dois tipos de caminho, uma função de avaliação é consultada que escolhe os melhores caminhos de acordo com o tamanho total do mesmo e os ângulos que o robô teria de girar em cada ponto de quebra (ângulos muito agudos são indesejáveis). As abelhas espectadoras são enviadas para as áreas de busca de abelhas coletoras com resultados promissores. Abelhas escoteiras procuram por pontos de curva aleatoriamente no ambiente. O algoritmo é realizado por um certo número de iterações ou até que a qualidade do caminho se estabilize. Os resultados mostram que o ABC é capaz de encontrar caminhos com qualidade 20% maior que o inicial, pontos de quebra com ângulos mais suaves e em um bom tempo.

O último algoritmo estudado foi o PSO (*Particle Swarm Optimization* - Otimização por Enxame de Partículas), proposto por KENNEDY e EBERHART (1995). O algoritmo tem inspirações no movimento e formação relativa de bandos de pássaros e cardumes de peixes. Inicialmente o objetivo dos autores era simular o comportamento de pássaros de um bando em um modelo tridimensional. Tentando alcançar uma modelagem cada vez mais verossímil, os autores perceberam a existência de um aspecto aleatório que fazia com que alguns agentes se separassem dos demais para posteriormente retomar o movimento sincronizado. Em algumas destas situações, uma posição vantajosa no ambiente era encontrada e o bando inteiro convergia junto deste agente autônomo. Além disso, a velocidade da convergência podia ser regulada, fazendo com que os agentes alcançassem um objetivo instantaneamente, sem chances de exploração, ou navegassem suavemente, tendo oportunidades de inovar. Os autores perceberam então um conceito de exploração em grupo com troca de informações e a capacidade de exploração, bastando apenas incluir a probabilidade dos agentes de seguirem um explorador autônomo para constituir uma *Swarm Intelligence*. As variáveis desnecessárias foram eliminadas e a necessidade dos agentes de se movimentarem como um bando foi descartada, simplificando o modelo para um enxame de partículas inteligente.

No modelo final de KENNEDY e EBERHART (1995), aplicado a otimização de funções multidimensionais, as partículas possuem uma memória individual e uma coletiva. Na primeira está armazenada a melhor posição encontrada pela partícula e no segundo a melhor posição encontrada pelo grupo ao qual ela pertence. A velocidade e direção do movimento da partícula no plano são influenciadas por estas memórias, cada uma com um peso associado e uma variável aleatória de ativação, a fim de permitir a exploração autônoma. A particularidade do PSO é a velocidade das partículas, que permite que novos pontos de exploração sejam encontrados durante o deslocamento. O algoritmo foi utilizado para ajustar os pesos de Redes Neurais, alcançando resultados tão bons quanto métodos de ajuste já consolidados, e foi capaz de encontrar pontos ótimos globais em funções com vários ótimos locais em tempo compatível com outras abordagens.

LU e GONG (2008) aplicam PSO no contexto de navegação de robôs em um ambiente dinâmico ativo, mapeando o problema da navegação em um de otimização de funções. Para

tanto, foi criada uma função a ser minimizada que considera a distância do robô ao objetivo e aos obstáculos. Quanto mais próximo dos obstáculos, maior o valor da heurística, ocorrendo o inverso para o objetivo. Existe um peso associado à distância dos obstáculos e um à distância do objetivo, sendo possível regular a importância que o robô emprega na ação de evitar colisões. Ao aplicar um algoritmo de PSO nesta função, um ponto ótimo será marcado no ambiente e se o robô se deslocar até o mesmo estará livre de colisões. O robô sabe a cada momento a posição representante do mínimo global alcançado pelo PSO, alterando sua direção sempre que uma nova posição é encontrada. Como o ambiente é dinâmico ativo, a função muda a cada instante e o PSO responde naturalmente a estas alterações. LU e GONG (2008) consideram que o robô tem limitações nos sensores e pode detectar apenas os obstáculos próximos. Como o robô se move em tempo real, os autores consideram que os obstáculos possuem velocidade inferior, a fim de que o robô tenha tempo para processar a função de minimização em tempo real. Em um ambiente simulado, tendo o robô uma velocidade de 30cm/s e sensor de proximidade com raio de 60cm, o obstáculo dinâmico uma velocidade de 13cm/s e alguns obstáculos estáticos distribuídos, o robô foi capaz de alcançar o objetivo sem colisões, trilhando um caminho muito próximo do ótimo.

QIN et al. (2004) utilizam uma representação bidimensional em grafo MAKLINK de um ambiente estático com obstáculos na forma de polígonos convexos. Nesta representação, os vértices dos obstáculos são mapeados como vértices de limite em um grafo, que representam as divisórias entre os obstáculos e as áreas livres. Em seguida, estes vértices de limite são usados para encontrar vértices livres no grafo, que representam regiões das áreas livres equidistantes dos obstáculos identificados. Arestas interconectam estes vértices livres e o robô pode trafegar pelas mesmas. Em seguida, o algoritmo de Dijkstra é utilizado para encontrar o menor caminho de um ponto inicial a um ponto objetivo no grafo. Entretanto, os vértices livres representam apenas um ponto na área livre, havendo pontos adjacentes que retornariam um caminho de menor qualidade se explorados. Um algoritmo PSO é utilizado para encontrar estes pontos. As partículas são alocadas em volta de cada vértice livre e devem encontrar novos pontos de deslocamento vantajosos, respeitando-se o tamanho da área livre. QIN et al. (2004) aplicam PSO com um fator de mutação nas partículas a fim de alterar aleatoriamente a velocidade das mesmas e dificultar a convergência a pontos ótimos

locais. Através de simulações, os autores observaram a convergência do modelo para o caminho ótimo em um cenário de aplicação em 80% das simulações com o PSO convencional e em 100% das simulações com o PSO com fator de mutação. Além disso, o tempo de convergência do PSO com mutação foi apenas 2.1s maior que o PSO convencional.

5.3 Método de Planejamento PSOSMv1

O método de planejamento LADSM contorna a deficiência do método DSM3 em lidar adequadamente com o resultado da previsão. Além disso, o tempo de processamento foi otimizado e o IR tem liberdade de se mover em qualquer direção na ausência de obstáculos próximos. É necessário que esta liberdade seja estendida para as situações realmente críticas na navegação, aquelas onde existem um ou mais MOs próximos do IR. A Figura 5.3 representa o deslocamento máximo do IR em uma rodada, onde a velocidade máxima é $max\ desl/n^o\ de\ rodadas$. Na Figura 5.3.a estão as opções de deslocamento que os métodos DSM3 e LADSM oferecem a cada iteração, sendo o tamanho do deslocamento o máximo possível (velocidade máxima constante). A Figura 5.3.b representa a área permitida de deslocamento desejada para o IR, ou seja, qualquer posição cujo centro do IR se encontre dentro da área de deslocamento máximo. Isto não só permitiria ao IR se deslocar em qualquer direção, como também ajustar sua velocidade. O método proposto nesta seção, PSOSMv1 (*Particle Swarm Optimization Search Method* versão 1), permite que o IR escolha qualquer posição dentro desta área de deslocamento, através de um método de otimização de PSO aplicado à heurística apresentada na Fórmula (5.1).

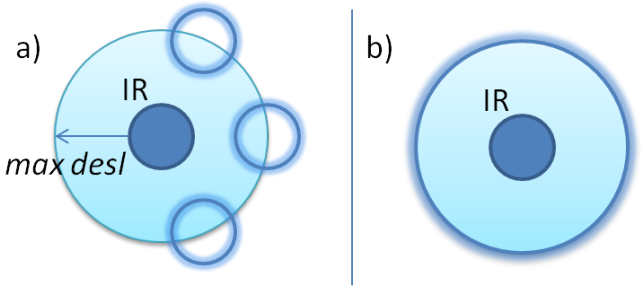

Figura 5.3: Área de deslocamento do IR em uma rodada do simulador: a) opções de deslocamento possíveis no DSM3 e LADSM; b) deslocamento desejável

A decisão por um algoritmo de PSO se dá devido à simplicidade do modelo e à aplicabilidade direta no problema a ser resolvido. Os algoritmos baseados em ACO são mais apropriados para buscas em grafos e, dado que a função de aplicação é bidimensional, com limites bem definidos e poucos ótimos locais, aqueles baseados em ABC são excessivamente complexos para a função proposta.

No algoritmo de PSO, um número pt de partículas é distribuído na função de aplicação, dentro dos limites estabelecidos. O valor da função para cada partícula é calculado e o melhor resultado do grupo é armazenado em uma variável *gbest*, e o melhor resultado individual da partícula i em uma variável $pbest_i$, onde $1 \leq i \leq pt$. Cada partícula possui velocidade dada pela Fórmula (5.3), onde v_{xi} e v_{yi} são as velocidades atuais nos eixos x e y, w_p é um peso associado à melhor coordenada ($pbestx_i$, $pbesty_i$) da partícula, $rand()$ é um fator de aleatoriedade, x_i e y_i são as coordenadas da posição atual da partícula e w_g é um peso associado à melhor coordenada ($gbestx$, $gbesty$) alcançada pelo grupo. Os pesos w_p e w_g definem a importância que as partículas atribuem para as melhores posições encontradas localmente e globalmente, regulando a convergência, e $rand()$ define a aleatoriedade necessária para a exploração. A posição de uma partícula é dada na Fórmula (5.4). As partículas se movem pela função, atualizando sua velocidade e posição a cada iteração do algoritmo PSO.

$$v_i = \begin{cases} v_{xi} = v_{xi} + w_p * rand() * (pbestx_i - x_i) + w_g * rand() * (gbestx - x_i) \\ v_{yi} = v_{yi} + w_p * rand() * (pbesty_i - y_i) + w_g * rand() * (gbesty - y_i) \end{cases} \tag{5.3}$$

$$p_i = \begin{cases} x_i = x_i + v_{xi} \\ y_i = y_i + v_{yi} \end{cases} \tag{5.4}$$

Dado que o PSOSMv1 é uma extensão do LADSM, o conceito de antecipação também é aplicado, assim como a avaliação da heurística apenas quando MOs se encontram próximos do IR. A aplicação de PSO no PSOSMv1 se assemelha ao proposto por LU e GONG (2008), com algumas particularidades. Nesta última o algoritmo de PSO é avaliado em todas as ocasiões, fazendo com que o tempo de processamento não permita a aplicação em ambientes complexos; a função heurística prioriza posições que aproximem o robô do objetivo e o

distanciem dos obstáculos, mas não existe nenhum fator limitante relacionado à velocidade do robô, o que dá a entender que as partículas do PSO estão livres para considerar qualquer posição no ambiente, aumentando o tempo de processamento; a pesquisa não explicita o erro mínimo buscado pelo algoritmo nem o número de iterações máximo, não sendo possível avaliar o tempo de processamento; o robô não dispõe da previsão dos obstáculos; o robô se movimenta em tempo real sem planejamento e está, desta forma, sujeito a colisões caso a velocidade dos obstáculos seja muito elevada. Em contrapartida, no PSOSMv1 o algoritmo de PSO é consultado apenas quando MOs se encontram próximos do IR, este se movendo em direção ao objetivo caso contrário; a função heurística, ainda que diferente da apresentada em LU e GONG (2008), também considera a melhor posição para o robô com relação aos obstáculos e objetivo, mas as partículas estão limitadas apenas à área de deslocamento máxima em volta do robô, reduzindo o tempo de processamento; o PSOSMv1 conta com a previsão do movimento dos obstáculos e, através da aplicação do *look ahead*, conhece a direção geral do movimento dos mesmos; o robô realiza um planejamento antes de iniciar a navegação, o que aliado à previsão permite que o robô navegue em ambientes com obstáculos com maior velocidade.

O fluxograma da Figura 5.4 ilustra o método PSOSMv1. O IR se encontra em sua posição inicial e tem à sua disposição as listas mo_{tri} com as trajetórias dos MOs, onde $1 \leq i \leq n$ e n é o número de MOs. As janelas de previsão estão localizadas inicialmente na primeira posição das listas $mo_{tri}(1)$.

A primeira ação do PSOSMv1 é verificar se alguma das previsões de posições dos obstáculos se encontra próxima da posição do IR considerada no planejamento. Em caso negativo, o LADSMv1 calcula o ângulo entre o IR e o objetivo e planeja um passo com velocidade máxima nesta direção. Senão, as previsões contempladas pela janela de previsão são alimentadas à heurística da Fórmula (5.1) em forma de obstáculos e o algoritmo PSO deve encontrar a melhor posição para o IR no planejamento. Em qualquer dos dois casos, se a posição planejada é o objetivo, o planejamento se encerra e a etapa de navegação se inicia. Caso contrário, a posição planejada do IR é inserida na lista ir_{tr}, as janelas de previsão são deslocadas em uma unidade para a direita e o processo se repete.

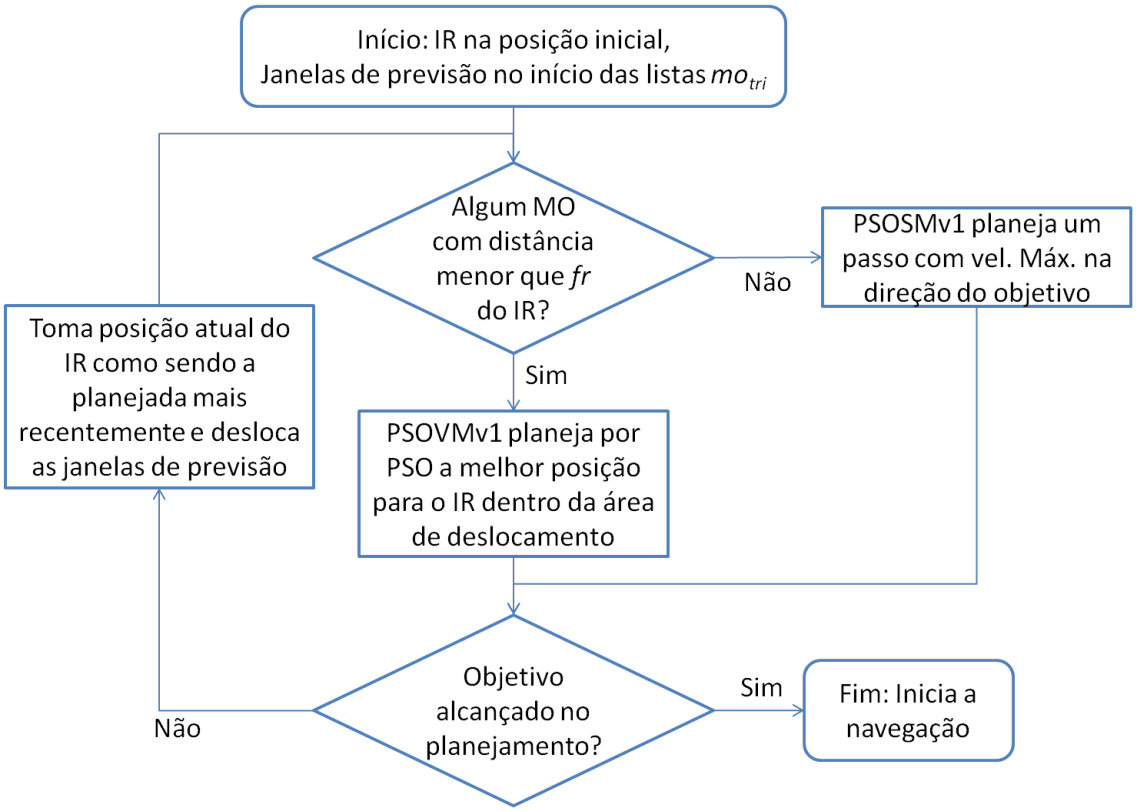

Figura 5.4: Fluxograma de etapas do método PSOSMv1

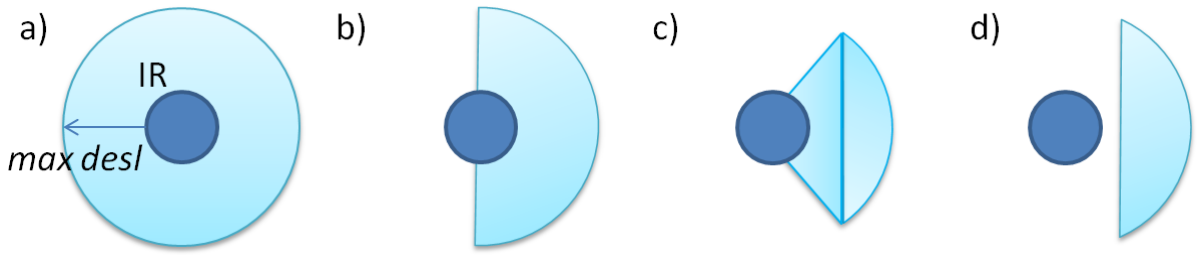

Figura 5.5: Exemplos de áreas de deslocamento para o método PSOSM

Assim como no método LADSM, a preocupação imediata na aplicação do PSOSMv1 é o tempo de planejamento, já que além do *look ahead*, um algoritmo de PSO tem de ser executado a cada avaliação da heurística da Fórmula 5.1. No entanto, o tempo poupado com a otimização do LADSM na avaliação da heurística, junto da simplicidade da função e os limites para atribuição de pontos do algoritmo PSO permitem a aplicação do método, como será visto na Seção 5.5 de experimentos.

Um detalhe interessante a ser ainda abordado com relação ao PSOSMv1 é a sua área de deslocamento. Como mostra a Figura 5.4, a área disponível para deslocamento, onde os pontos do algoritmo PSO são alocados e avaliados na heurística, é um círculo em volta do IR com raio *max desl*. Esta área pode ser moldada de forma que algumas limitações sejam estabelecidas. A Figura 5.5 ilustra alguns exemplos. A área da Figura 5.5.b representa um caso onde o robô não pode se deslocar para trás, evitando assim ciclos que poderiam comprometer a qualidade do caminho. A área da Figura 5.5.c limita os ângulos que podem ser tomados pelo IR, evitando que as curvas sejam muito acentuadas, aproximando o modelo a um robô de quatro rodas no mundo real. A área da Figura 5.5.d limita não só os ângulos de direção disponíveis, como também a velocidade mínima do robô.

Através do mecanismo de modelagem da área de deslocamento o método PSOSMv1 permite uma aproximação do ambiente simulado para o ambiente real. Os robôs aplicados no mundo real normalmente são do tipo diferencial com duas rodas motorizadas, fazendo com que eles não possam se movimentar livremente em todas as direções em um instante de tempo. Uma modelagem da área de deslocamento de forma que as restrições do movimento

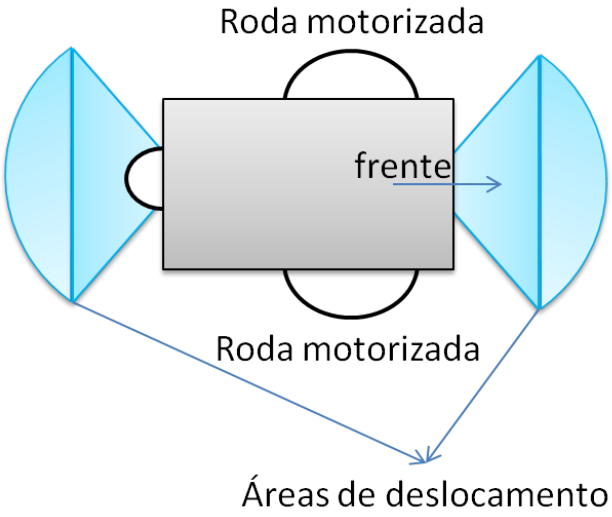

Figura 5.6: Robô diferencial e área permitida de deslocamento no PSOSMv1

do robô de aplicação sejam consideradas permitiria a transferência do modelo para o mundo real. A Figura 5.6 mostra um exemplo de uma área viável para um robô diferencial.

A melhor área para o IR no simulador será definida na Seção 5.5.

5.4 Método de Planejamento PSOSMv2

O PSOSMv2 é uma extensão do PSOSMv1 e foi desenvolvido para solucionar uma deficiência identificada na estratégia APF_p. A Figura 4.21 mostra uma situação onde o robô se insere em uma área perigosa, onde os obstáculos o cercam e ocasionam uma colisão. Este tipo de situação se agrava com a quantidade crescente de MOs no ambiente. Dado que a previsão no FIRN fornece uma visão geral dos obstáculos, seria interessante marcar os pontos onde existem grandes concentrações de obstáculos em um instante tf de tempo futuro, de forma que o IR evite entrar nestas áreas. Inicialmente, foram empregadas estratégias de *clusterização* e PSO para demarcar estas áreas, mas o tempo de processamento associado não pôde ser ajustado de forma a se tornar compatível com o *framework*. Em alguns casos críticos, o sistema demorava até 8s para retornar os pontos de risco.

Uma solução para o problema é instituir um raio de medo fr dinâmico. Como fr representa o raio do círculo em volta do IR, onde obstáculos são classificados como perigosos, um aumento dinâmico neste parâmetro poderia englobar um grupo de MOs à frente, fazendo eventualmente com que todos venham a ser considerados na heurística de evasão e as manobras se iniciem antecipadamente. A Figura 5.7 ilustra com um exemplo. Durante o planejamento, um conjunto de MOs próximos é identificado no instante tf na previsão. Para que o IR evite esta região, o raio de medo em volta do mesmo é estendido e todos os MOs são considerados perigosos. O PSOSMv2 começa então a planejar o deslocamento para longe do grupo.

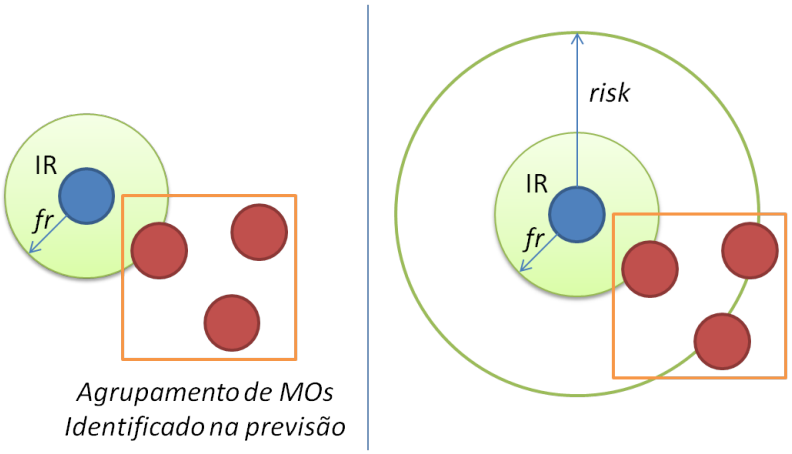

Figura 5.7: No método PSOSMv2, o raio do medo é ajustado de acordo com o número de MOs identificados em um grupo próximo à posição planejada para o IR

Uma nova constante, chamada $maxFearFactor$ determina o fator máximo de multiplicação para o raio do medo fr. O valor de fr é multiplicado por min($numMosG$, $maxFearFactor$), onde $numMosG$ é o número de MOs agrupados em uma área. Os MOs devem ter uma distância máxima $groupRadius$ entre si para que sejam considerados do mesmo grupo. O conceito por trás da constante $maxFearFactor$ é o seguinte: quanto mais MOs presentes em uma área, mais perigosa esta se torna. Caso o IR adentre esta área, a chance de colisão aumenta proporcionalmente ao número de MOs na mesma. No entanto, a reação não deve ser exagerada a ponto de comprometer a qualidade do caminho, tendo que haver um limite no crescimento. O valor de $maxFearFactor$ deve ser regulado de forma que

o IR se distancie de uma área perigosa sem exageros, considerando a frequência de situações onde obstáculos se deslocam em grupos e o número mais frequente de MOs esperados em um grupo no ambiente de estudo. Além disso, um conjunto de MOs não representa necessariamente um perigo, já que podem estar distantes das posições planejadas para o IR. Nestas situações, o grupo deve ser ignorado ate que exista uma aproximação.

O fluxograma da Figura 5.8 mostra o funcionamento do método, com o novo módulo inserido na versão 2 destacado. No início da simulação, o IR está na posição inicial e as janelas de previsão englobam os primeiros membros de mo_{tri}, assim como no PSOSMv1. Uma variável chamada *risk* é iniciada com o valor de fr e duas condições são testadas para definir se existem grupos de risco próximos ao MO: primeiro é definido se realmente existe um grupo de risco no conjunto de previsões contempladas pelas janelas de previsão. Em caso positivo, deve-se também definir se a distância entre qualquer um deles e a posição no planejamento atual do IR é menor que fr * min($numMosG$, $maxFearFactor$), já que um grupo de risco distante do caminho do IR não representa perigo. Se alguma das duas condições for falsa, o valor de *risk* continua sendo fr e o método procede exatamente como no PSOSMv1. Caso as duas condições sejam verdadeiras, o valor de *risk* é atualizado como sendo fr * min($numMosG$, $maxFearFactor$) - $minDistMoG$, onde $minDistMoG$ é a distância mínima entre dois MOs dentro do grupo (quanto menor $minDistMoG$, mais agrupados estão os MOs e mais perigoso pode ser se aproximar muito). O método prossegue como no PSOSMv1, mas qualquer MO com distância menor que o novo valor de *risk* será considerado uma ameaça.

5.5 Experimentos com os Novos Métodos de Planejamento

Nesta Seção serão descritos os experimentos avaliando o desempenho dos novos métodos desenvolvidos para o FIRN. Assim como na Seção 4.4, primeiramente são tomados os tempos de planejamento e número de colisões referentes a cada método do FIRN. Com estas métricas é possível avaliar o impacto das novas implementações no processamento e na qualidade da navegação. Definido o melhor método de navegação disponível do FIRN, os experimentos comparativos relativos às APFs são tomados.

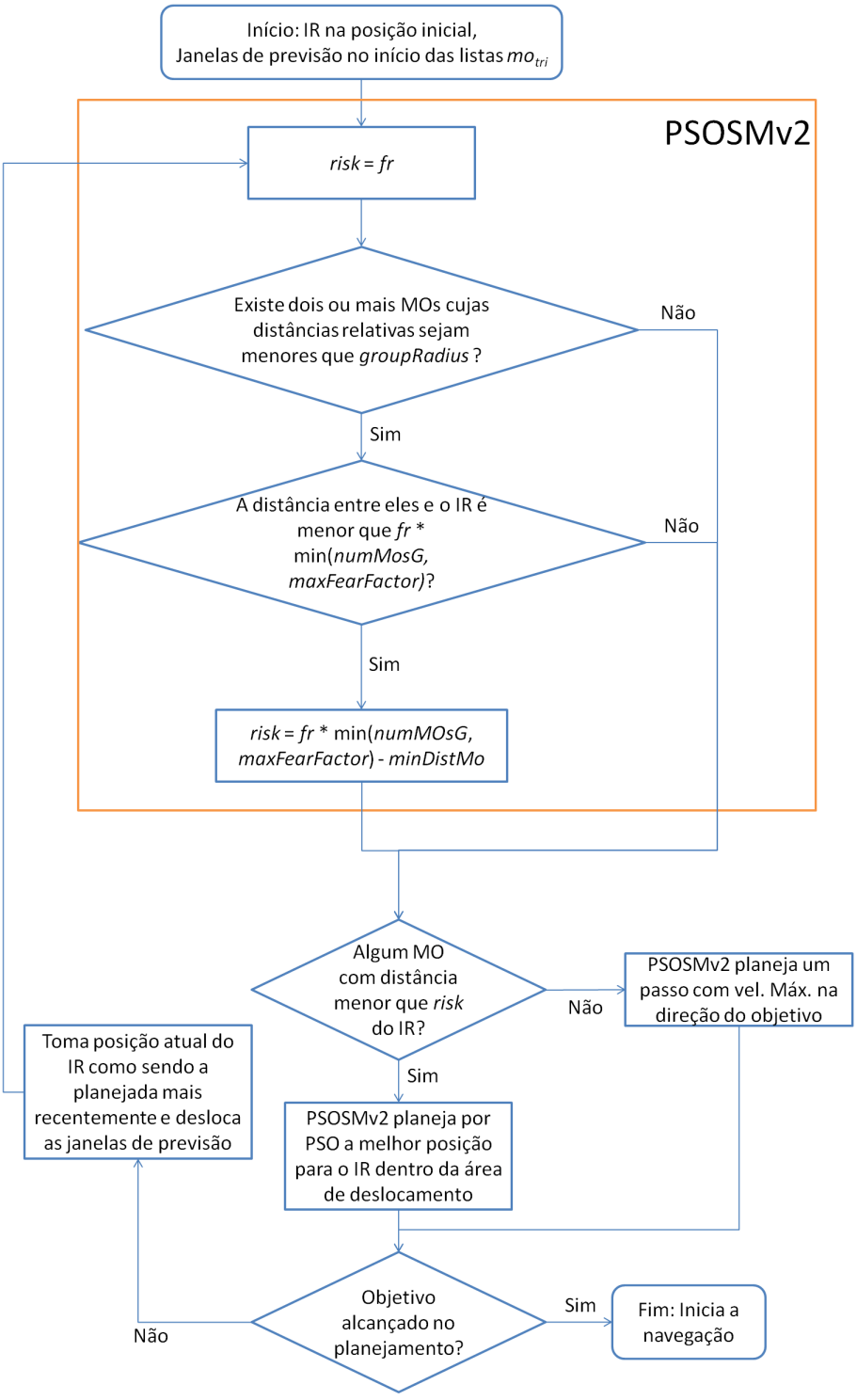

Figura 5.8: Fluxograma de etapas do método PSOSMv2

As características gerais no Simulador FIRNn, assim com a configuração da máquina de testes, são as mesmas da Seção 4.4. Primeiramente são definidos os parâmetros de cada método introduzido neste Capítulo.

5.5.1 Definição de Parâmetros para os Novos Métodos

Para o método LADSM, é introduzida a constante la (*look ahead*) de antecipação do planejamento. Quanto maior for o la, maior será a janela de deslocamento nas listas mo_{tr} e, consequentemente, mais previsões à frente o método considera a cada passo do planejamento. Para definir o melhor valor para este parâmetro foram realizados testes onde o IR deveria navegar em um ambiente contendo um MO com o padrão de movimento em Onda. A Figura 5.9 mostra a diferença no comportamento do IR para valores de $la = 0$ (DSM3) até $la = 16$. Apenas a partir de $la = 8$ o robô consegue evitar a colisão, mas ainda existe uma pequena oscilação (o planejamento faz o robô descer um pouco no eixo y para em seguida subir). Com $la = 16$, a oscilação é eliminada e a colisão evitada. Este valor foi o escolhido. Os demais atributos, w_f na penalidade da heurística (Fórmula 5.2) e fr continuam os mesmos: $+\infty$ e 40, respectivamente.

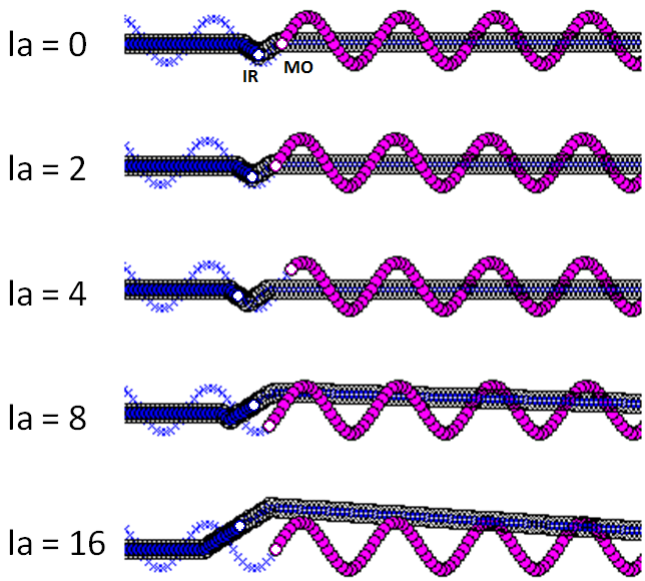

Figura 5.9: Variação do parâmetro la no método LADSM

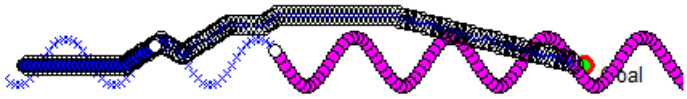

Figura 5.10: Resultado do planejamento do método DSM3 com $fr = 120$

Pode-se contestar que a variável la tenha o mesmo efeito que um aumento na variável fr, já que a primeira se refere a uma antecipação da previsão no planejamento e a segunda ao início antecipado das manobras de evasão, já que MOs mais distantes já são considerados ameaças. O resultado no entanto é bem diferente. Como mostra a Figura 5.10, o planejamento do DSM3 com $fr = 120$ (valor três vezes maior que o aplicado nos experimentos da Seção 4.4) resulta em oscilações na trajetória, diminuindo consideravelmente a qualidade do caminho. Mesmo que os desvios sejam realizados muito antes do MO se aproximar, a previsão continua sendo consultada pontualmente e o planejamento não tem noção da direção tomada pelo MO. Isso não acontece quando se altera a variável la.

No método PSOSMv1 é introduzido o algoritmo PSO e as constantes associadas devem ser reguladas. É necessário definir o número de partículas pr, os pesos associados à convergência local e global w_p e w_g, o erro mínimo esperado e o número de iterações. Para w_p e w_g, parece não haver uma maneira concreta de definir qual deve ter o maior valor e KENNEDY e EBERHART (1995) sugerem que os dois sejam substituídos por uma constante. A partir disto e considerando que o ambiente de busca é bem limitado, o valor escolhido foi de 0.5 para ambos.

Para definir o valor de pr e o número de iterações necessário para convergência, um cenário de teste contemplando nove MOs em uma situação complexa foi elaborado. Dado que existe um fator de aleatoriedade na convergência do algoritmo PSO, o valor ideal para estes dois atributos é aquele de menor tamanho, garantindo o menor tempo de processamento possível, mas que retorne uma trajetória consistente quando aplicado ao mesmo cenário. Então, a partir de $pr = 5$ e $nI = 1$ (número de iterações), o método PSOSMv1 é executado 100 vezes no cenário de teste desenvolvido. Os valores são aumentados iterativamente até que a média da qualidade do caminho seja aproximadamente igual à da rota ideal e o desvio padrão seja baixo o suficiente, mostrando que o caminho foi consistente em todas as itera-

ções. Entende-se por caminho ideal a rota resultante do método PSOSMv1 rodando com os atributos $pr = 25$ e $nI = 100$, o que retorna um caminho com 86.7% de qualidade em um tempo aproximado de 7s (um tempo muito alto para ser aplicado ao *framework*).

A Tabela 5.1 mostra os resultados para alguns parâmetros compatíveis com o tempo de processamento disponível para o método. Estão dispostos a média e desvio padrão referentes à qualidade do caminho (QC) e o tempo de planejamento (TP) nas 100 rodadas na simulação. A qualidade do caminho é tomada como sendo a porcentagem que representa o tamanho do caminho gerado pelo método de planejamento com relação a um caminho em linha reta entre o ponto inicial e o objetivo (quanto mais próximo de 100%, maior a qualidade). A Tabela 5.1 mostra um resultado interessante: quando o número de partículas é igual a 15 e o número de iterações do algoritmo é igual a 1, a qualidade do caminho já é próxima da qualidade do caminho ideal, havendo apenas um desvio padrão de 2.27%. Como as partículas são inicialmente distribuídas de forma equidistante no espaço da função, que no caso do FIRN é mapeado como sendo a área de máximo deslocamento em volta do IR, e a heurística não possui ótimos locais significantes, esta distribuição já dá uma estimativa boa para o PSO de qual direção é a mais vantajosa a se tomar. Uma das quinze partículas distribuídas já está normalmente mapeada na direção de deslocamento ideal para o IR naquela iteração do planejamento. O algoritmo PSO é utilizado apenas para refinar o resultado e um máximo de três iterações já é suficiente para aproximar a qualidade da rota escolhida para a qualidade da rota ideal em todas as iterações (o desvio padrão quando $nI = 3$ é de apenas 0.45%). O tempo de processamento médio do PSOSMv1 é de 0.406s para o algoritmo PSO com parâmetros $nR = 3$ e $pr = 15$, o que é compatível com o ambiente de aplicação.

Tabela 5.1: Qualidade do caminho e tempo de planejamento para diferentes parâmetros do algoritmo PSO

	QC (média)	QC (desv. padr.)	TP (média)	TP (desv. padr.)
$nI = 1$, $pr = 5$	57.93%	31.13%	0.637	0.462
$nI = 1$, $pr = 15$	83.64%	2.27%	0.324	0.022
$nI = 3$, $pr = 5$	84.73%	1.23%	0.398	0.018
$nI = 3$, $pr = 15$	86.44%	0.45%	0.406	0.016

A Figura 5.11 mostra o caminho resultante do PSOSMv1 utilizando um algoritmo PSO com os parâmetros $pr = 15$, $nI = 3$, e um algoritmo PSO com os parâmetros $pr = 25$, $nI = 100$. Os MOs foram omitidos na imagem para favorecer a visualização. Não existem diferenças substanciais entre os dois caminhos, o que sugere que o algoritmo PSO converge rapidamente no ambiente de aplicação. Dado que o desvio padrão foi de apenas 0.45%, este caminho não só é o ideal, como também foi tomado consistentemente em todas as 100 simulações.

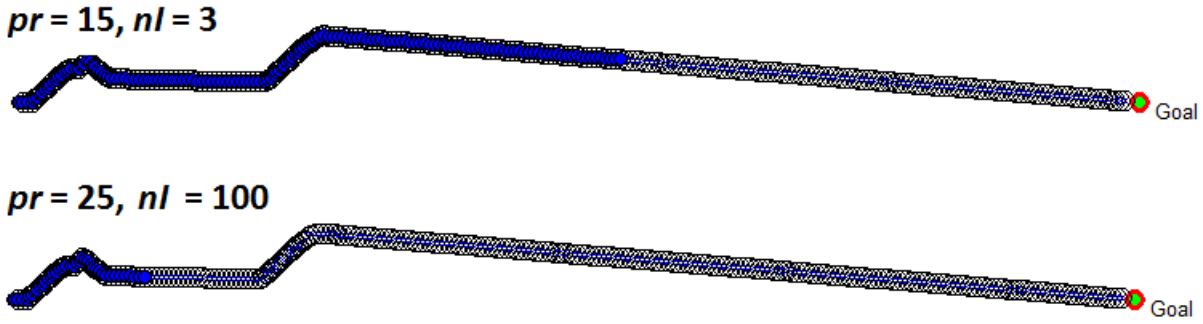

Figura 5.11: Caminho retornado pelo PSOSMv1, utilizando o algoritmo PSO com diferentes parâmetros

Dados os resultados, o conjunto de atributos escolhido para o algoritmo PSO foi o seguinte: $pr = 15$, $w_p = w_f = 0.5$, erro mínimo esperado indefinido (o algoritmo é executado até que o número de iterações de esgote) e número de iterações máximo de 3.

Um último parâmetro a ser definido para o método PSOSMv1 é a área de deslocamento em volta do IR onde partículas do PSO podem ser distribuídas. Na Figura 5.5 foram dados alguns exemplos de áreas possíveis. A escolhida para o ambiente simulado foi a representada na Figura 5.5.d, fazendo com que o IR se dirija sempre em direção ao objetivo e sua velocidade varie apenas entre max/2 e max. A primeira restrição evita que o IR projete ciclos ao desviar dos obstáculos, o que poderia comprometer a qualidade do caminho, enquanto que a segunda evita que o IR estacione em locais perigosos, esperando que os MOs atravessem, mas colida por causa de um movimento súbito de um MO em alta velocidade.

Para o método PSOSMv2, os valores de $maxFearFactor$ e $groupRadius$ devem ser definidos. Dado que o ambiente de simulação possui dimensões [800, 400] e o tamanho

mais provável de MOs em um grupo é três, este parece ser um valor apropriado, já que $3 * fr = 120$ será o tamanho máximo do fr dinâmico e 120 é aproximadamente 1/3 da área total disponível para deslocamento no eixo y. O valor de *groupRadius*, a distância máxima entre MOs para que sejam considerados de um mesmo grupo, foi definido como sendo 80. Os demais parâmetros são os mesmos do método PSOSMv1.

5.5.2 Desempenho dos Novos Métodos de Planejamento

O intuito é avaliar o impacto dos novos métodos desenvolvidos para a etapa de planejamento com relação aos disponíveis inicialmente. Os cenários de simulação são aqueles desenvolvidos na Subseção 4.4.3, totalizando 1080 cenários, divididos em grupos de 120, cada grupo contendo números progressivamente maiores de MOs a partir de 1 até um máximo de 9. Os parâmetros para os métodos DSM3, DSM7 e FDSM são aqueles definidos na Seção 4.4.2 e os parâmetros gerais do Simulador FIRNn são os enunciados na Seção 4.4.

O gráfico da Figura 5.12 mostra o tempo médio de planejamento de uma versão preliminar do LADSM (LADSMα), onde somente o *lookahead* é aplicado, ou seja, a função heurística é avaliada a todo momento, mesmo quando as posições previstas dos obstáculos estão distantes do ponto planejado para o IR. Os métodos de comparação são o DSM3 e o DSM7 (o método FDSM não será mais abordado, dada a direção tomada na dissertação). Como foi discutido na Seção 4.4.3, o algoritmo aplicado no DSM possui dois ciclos do tipo "enquanto" aninhados, um correspondente ao número de passos que o IR tem de tomar para chegar ao objetivo e o segundo ao número de direções a serem consideradas no planejamento. O acréscimo de MOs adiciona operações simples de soma e cálculo de distância entre pontos. Como o LADSMα é baseado no DSM3 e o algoritmo possui os mesmos ciclos, a estrutura é acessada $3 * size(mo_{tr})$ vezes, onde $size(mo_{tr})$ é o tamanho final da lista de posições planejadas para o IR. Dado que cada previsão antecipada considerada é tratada na heurística como um MO distinto, o valor de la é 16 e a adição de um MO no LADSMα acrescenta um tempo da ordem de 10^{-3}s, o acréscimo de tempo com relação ao DSM3 é de aproximadamente $numMos * la * 10^{-3}$s, onde $numMOs$ é o número de MOs na simulação e la é o *look ahead*. Para o grupo de 9 obstáculos, o tempo adicional tomado pelo LADSMα é de aproximadamente $9 * 16 * 10^{-3} = 0.144$s, representando um atraso de apenas um passo

dos MOs no processamento. O tempo total de processamento ainda é menor que no método DSM7 para 9 obstáculos, mas cada MO adicional é tratado como *la* MOs a mais, justificando o maior crescimento com o aumento de MOs.

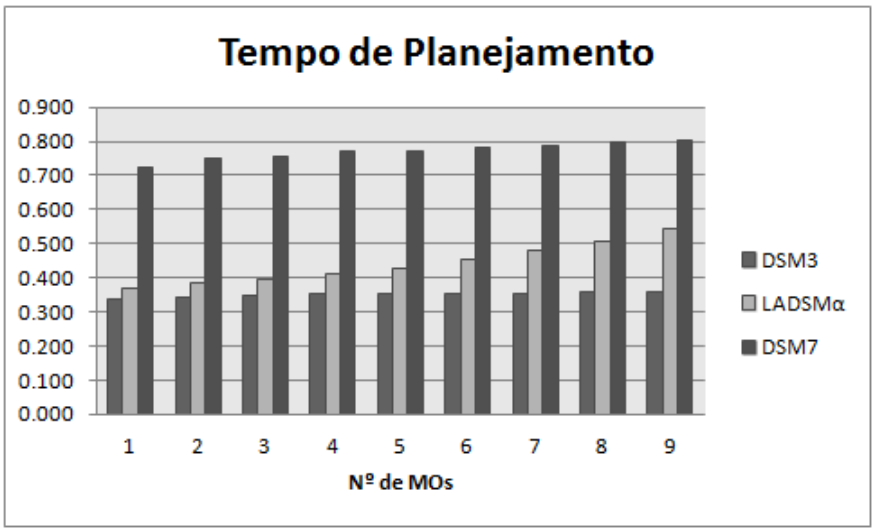

Figura 5.12: Tempo de planejamento para o método LADSM sem otimização de tempo (nº de MOs x tempo (s))

O gráfico da Figura 5.13 mostra o tempo médio de planejamento do LADSM final, desta vez calculando a heurística apenas quando a previsão de algum MO está próximo de uma distância fr do ponto planejado para o IR. O tempo tomado pelo LADSM é menor até mesmo que o do DSM3 para todos os grupos de obstáculos, comprovando que o cálculo desnecessário da heurística na ausência de MOs tem um grande peso no tempo total de processamento do método. Com o acréscimo de MOs, o tempo de processamento do LADSM aumenta rapidamente, quase se aproximando do DSM3 em cenários com 9 MOs, já que os casos de confronto aumentam com o número dos mesmos. Não é possível precisar o tempo adicional do LADSM com o acréscimo de obstáculos, já que isto depende do número de confrontos que efetivamente ocorrerão no caminho, mas a taxa de crescimento é da ordem de 10^{-2}s.

Na Figura 5.14 consta o tempo médio de planejamento do método PSOSMv1 em relação aos demais. Como a heurística é avaliada apenas na presença de obstáculos próximos ao ponto planejado para o IR, o tempo de processamento no caso de um MO é inferior ao

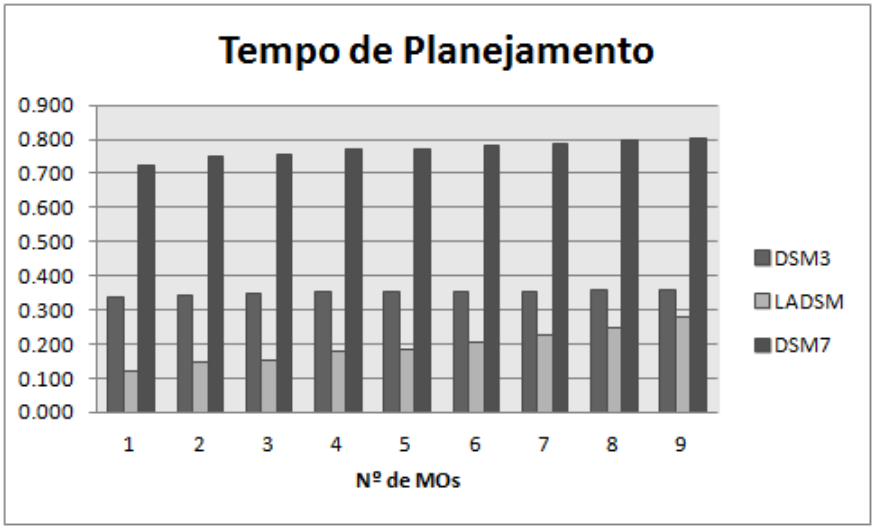

Figura 5.13: Tempo de planejamento médio para o método LADSM, considerando-se cada grupo de n MOs (nº de MOs x tempo (s))

do método DSM3, mas não do que o do LADSM, já que este último avalia apenas três direções na função heurística e o PSOSMv1 processa uma execução do algoritmo PSO. O método consegue ser mais rápido que o DSM3 até o grupo de sete MOs. Mesmo para nove MOs o método é mais rápido do que o DSM7. Como o PSO tem sempre três rodadas para minimizar a função heurística, o acréscimo do tempo com o aumento do número de MOs se dá apenas pelas operações adicionais ao avaliar a heurística (somas + cálculos de distância) e o maior número de confrontos, aumentando o número de vezes que o PSO é processado. O tempo de uma execução completa do PSO para calcular uma posição do planejamento com os parâmetros utilizados e a configuração da máquina é de $0.0044 + n * la * 125 * 10^{-7}$s, onde n é o número de MOs e la é *lookahead*. O tempo geral de planejamento do PSOSMv1 parece subir de forma escalar com o número de MOs, observando-se um acréscimo de 0.06s a cada MO inserido, mas depende do número de confrontos de MOs com o IR. O crescimento pode ser considerado alto, mas o tempo de planejamento só será igual ao do DSM7 em cenários com 15 obstáculos, considerando-se o tempo médio no simulador.

O gráfico da Figura 5.15 mostra o tempo médio de planejamento para o método PSOSMv2. A única diferença entre este e a versão 1 é o caráter dinâmico de fr, podendo aumentar em até três vezes para os parâmetros definidos. Dado que o aumento em fr

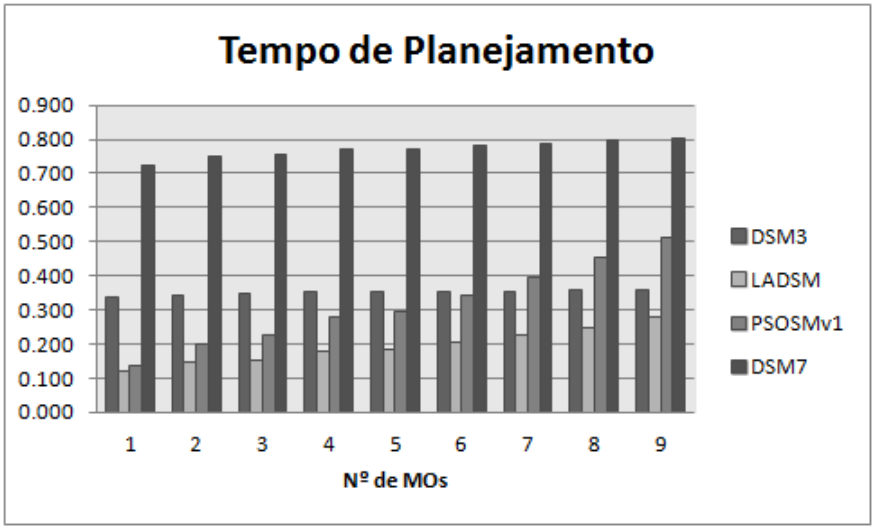

Figura 5.14: Tempo de planejamento médio para o método PSOSMv1, considerando-se cada grupo de n MOs (nº de MOs x tempo (s))

gera mais confrontos entre o IR e os MOs, o número de vezes que o PSO é executado e, consequentemente o tempo de processamento do método, também cresce. O custo de cada execução individual do algoritmo PSO é o mesmo do PSOSMv1. Como pode ser visto, a diferença entre as duas versões aumenta com o número de MOs no cenário, simbolizando o aumento no número de confrontos. No entanto, o tempo de processamento ainda é menor que no DSM3 para cenários com um MO, característica que é mantida até que o número de MOs alcance seis, e menor que o DSM7 em todos os casos. O crescimento médio no tempo de processamento do método a cada MO adicionado é de 0.07s, 0.01s a mais que o PSOSMv1 e alcançará o tempo do DSM7 quando o número de MOs no ambiente for igual a 13, considerando-se o tempo médio no simulador.

A Tabela 5.2 mostra os resultados gerais com relação ao tempo de planejamento para os métodos de navegação baseados no DSM do FIRN. Existem casos onde o tempo máximo de planejamento para os métodos PSOSMv1 e PSOSMv2 ultrapassa o do método DSM7 mas, conferindo a média e o desvio padrão, fica claro que estes casos não se repetem com frequência. É interessante observar que os dois métodos que aplicam PSO possuem o tempo mínimo menor que o LADSM, considerado o mais otimizado com relação ao tempo de processamento. Estes casos, também raros, acontecem quando existe apenas um MO no

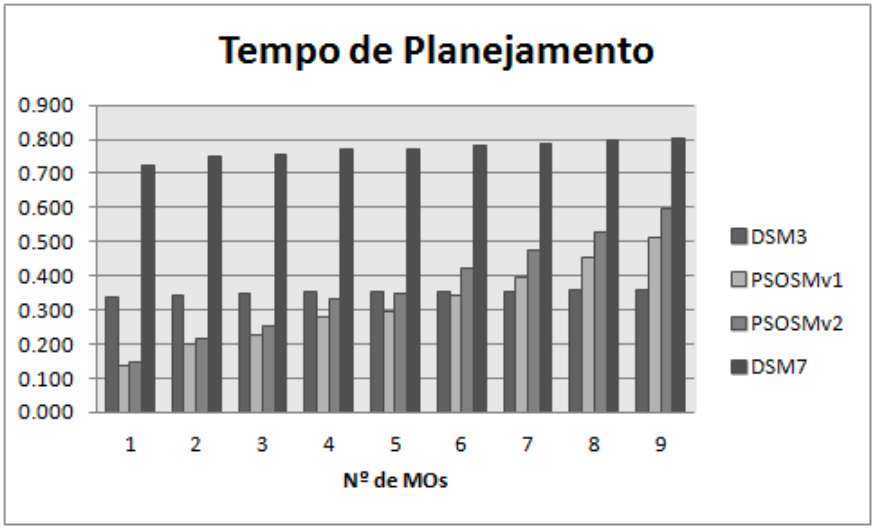

Figura 5.15: Tempo de planejamento médio para o método PSOSMv2, considerando-se cada grupo de n MOs (nº de MOs x tempo (s))

ambiente e o conjunto mais abrangente de direções possíveis distanciam o IR do MO mais rapidamente.

Tabela 5.2: Tempo geral de planejamento

	Média	Desvio Padrão	Mínimo	Máximo
DSM3	0.351s	0.010s	0.327s	0.428s
DSM7	0.771s	0.035s	0.692s	0.888s
LADSM	0.193s	0.054s	0.100s	0.396s
PSOSMv1	0.316s	0.138s	0.079s	0.996s
PSOSMv2	0.369s	0.162s	0.087s	1.055s

A Figura 5.16 mostra um gráfico com o número de colisões do IR para cada novo método de navegação proposto aplicado a cada grupo de MOs. Imediatamente fica clara a redução na taxa de colisões dos novos métodos comparados ao DSM3. O método LADSM, que utiliza a previsão de forma inteligente, é o principal responsável na redução, o que comprova que a proposta do FIRN é de fato válida, bastando apenas que as informações importantes derivadas da previsão sejam utilizadas adequadamente. O tempo de processamento otimizado certamente contribui para o resultado, mas de certo o índice na redução de colisões não teria sido tão significante se a deficiência apontada na Figura 4.18 não tivesse

sido resolvida. Além disso, esta solução é mais eficiente do ponto de vista computacional que a busca exaustiva empregada no método DSM3 do Simulador FIRNp (Figura 3.10), principalmente no contexto de vários obstáculos.

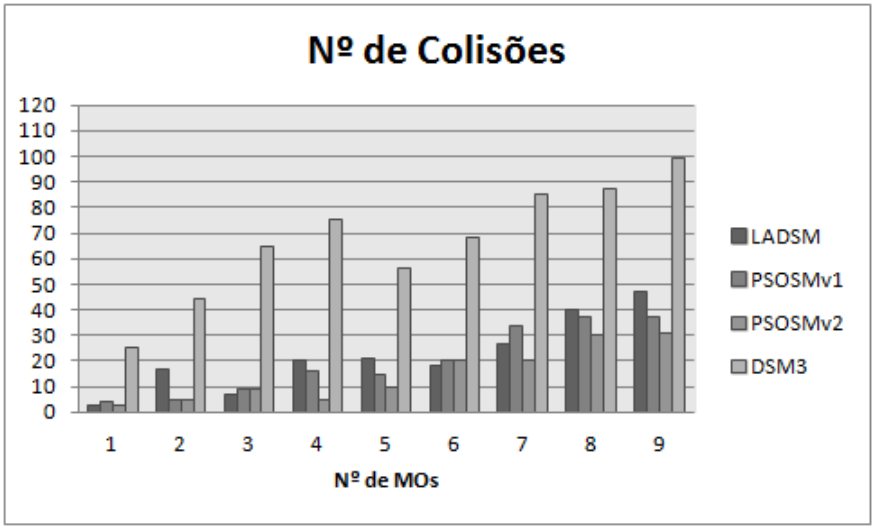

Figura 5.16: Número de colisões para os novos métodos, considerando-se cada grupo de n MOs (nº de MOs x tempo (s))

Considerando-se apenas os novos métodos desenvolvidos, no geral é observado um crescimento no número de colisões com a introdução de MOs, o que é natural dada a dificuldade adicional. Comparativamente, os métodos PSOSMv1 e PSOSMv2 apresentaram um menor número de colisões que o LADSM, comprovando a eficiência associada à flexibilidade nas direções tomadas no planejamento. Existem apenas dois conjuntos destoantes, o de três e o de seis MOs, onde o LADSM registra uma colisão a menos. A Figura 5.17 mostra um destes casos. A verdadeira causa da colisão é uma pequena deficiência na previsão. Ainda que os erros na previsão não influenciem no número de colisões na maior parte das vezes, a discretização empregada pelo E-LOFORM neste caso acaba por induzir a colisão. Os métodos PSOSMv1 e PSOSMv2 esperam que o MO, cuja previsão foi incorreta, esteja um pouco abaixo de onde realmente está e, como têm mais liberdade no movimento, acabam priorizando a distância em relação ao obstáculo de cima, colidindo com o MO de baixo.

O método PSOSMv2 sujeita o IR a um índice de colisões ligeiramente menor que o PSOSMv1, resultado da aplicação do fr dinâmico. Quando vários MOs são identificados uns

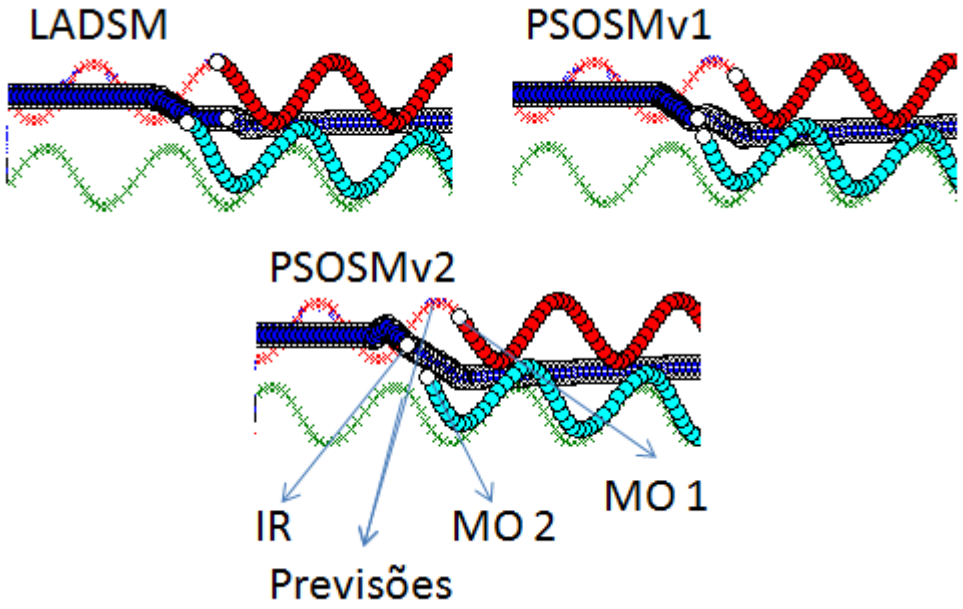

Figura 5.17: Colisão nos métodos PSOSMv1 e PSOSMv2 devido a falhas na previsão. Existem dois MOs se deslocando no padrão de movimento Onda e suas trajetórias previstas estão ilustradas. O IR se move de acordo com o planejamento, mas como a previsão é incorreta para o MO 2, existe uma colisão

próximos dos outros, o IR antecipa a evasão, escapando de situações de perigo desnecessárias. A Figura 5.18 ilustra uma situação típica onde o PSOSMv2 identifica um grupo de obstáculos muito próximos uns dos outros. Na Figura 5.18.a, o método PSOSMv1 identifica apenas um obstáculo em seu caminho e considera, pela heurística, que a melhor direção a ser tomada é para baixo. No entanto, existe um agrupamento de quatro obstáculos na região, ocasionando uma colisão. Já na Figura 5.18.b, o método PSOSMv2 identifica antecipadamente o agrupamento de MOs, aumentando o raio do medo fr, fazendo com que todo o grupo seja visto como uma ameaça, forçando uma evasão antecipada e consciente por parte do IR.

O índice total de colisões para cada método foi de 200 colisões para o LADSM, 177 colisões para o PSOSMv1 e 133 colisões para o PSOSMv2, num total de 1080 cenários.

5.5.3 Novos Métodos de Planejamento x APFs

Agora são apresentados os resultados comparativos das estratégias baseadas em APFs e os novos métodos de planejamento desenvolvidos para o FIRN. Cada estratégia foi aplicada no conjunto de 1080 cenários de testes apresentado na Seção 4.4.3, assim como demais parâmetros do robô e obstáculos. Os quesitos comparativos são o número de colisões e a qualidade do caminho com relação a um caminho em linha reta do ponto inicial ao objetivo.

O gráfico da Figura 5.19 fornece os resultados obtidos nos experimentos de avaliação para o número total de colisões em cada grupo de MOs. Estão contemplados o método PSOSMv2 do FIRN, que apresentou a melhor performance em seu domínio, e as estratégias APF_k e APF_p. O resultado mostra que o método PSOSMv2 sujeitou o IR a um menor número de colisões para todos os grupos de MOs com relação a um robô se deslocando no mesmo ambiente utilizando APFs. A diferença é mais acentuada nos grupos com poucos obstáculos, onde o IR navegando na trajetória gerada pelo PSOSMv2 chega a conseguir um índice de 29.2% de colisões a menos no grupo de cenários contemplando 4 MOs (APF_k x PSOSMv2). No grupo com 9 MOs, a diferença chega a 21.7% (APF_p x PSOSMv2).

A Figura 5.20 apresenta uma situação onde existe colisão entre o robô navegando de acordo com a estratégia APF_k e o MO em um cenário com apenas um obstáculo. A estratégia APF_k colide apenas em 15 dos 120 casos contemplados neste grupo, ocorrência

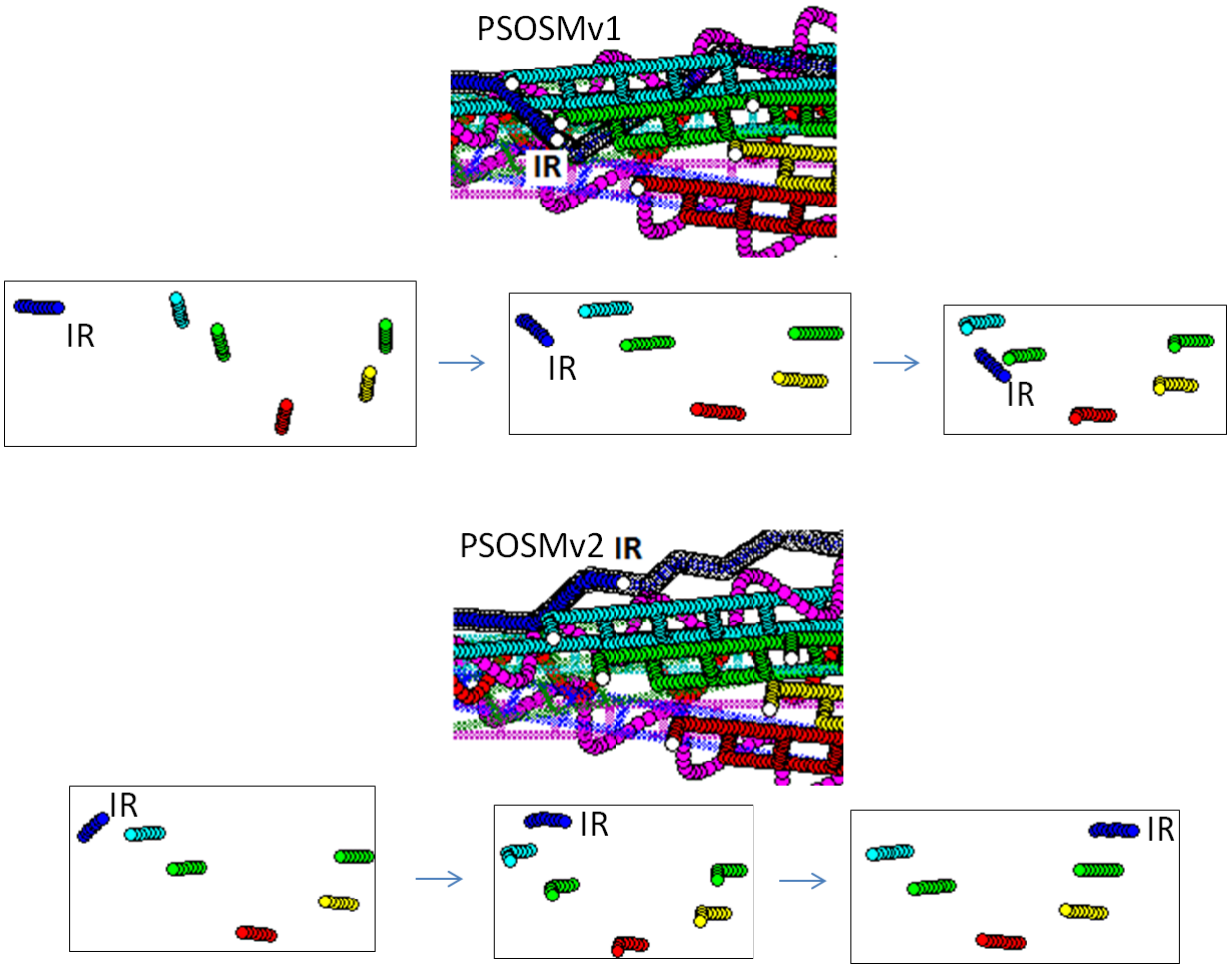

Figura 5.18: Evasão de um grupo de obstáculos agrupados: a) PSOSMv1 b) PSOSMv2. Em a), o IR é inserido em uma área de risco, com vários MOs. Em b), o planejamento faz com que o IR evite a área de concentração antecipadamente. Em destaque estão as situações como vistas no simulador. Abaixo das mesmas, capturas de telas sequenciais ilustrando o ocorrido

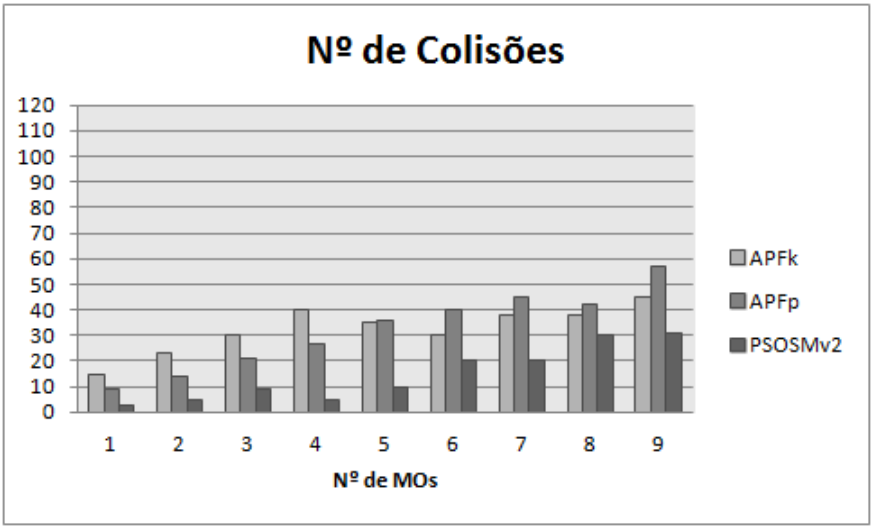

Figura 5.19: Número de colisões para estratégias E-LOFORM + PSOSMv2, APF_k e APF_p, considerando-se cada grupo de n MOs (nº de MOs x tempo (s))

que depende do ângulo e do padrão de movimento e acontece devido à velocidade elevada do obstáculo. O IR navegando com o PSOSMv2 conhece a direção do movimento do MO e desvia com antecedência.

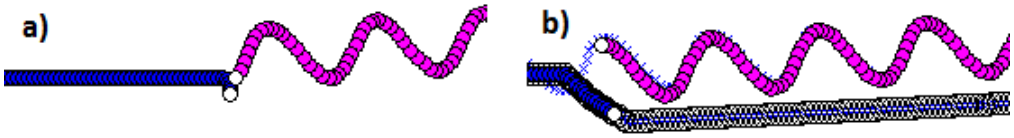

Figura 5.20: Situação de colisão para a estratégia APF_k em cenário com um obstáculo: a) APF_k, b) PSOSMv2. Em a), o ângulo da direção e a velocidade do MO fazem com que o robô colida. Em b), o planejamento antecipa o movimento do MO e evita a colisão

Na Figura 5.21 é observado um cenário com nove MOs onde o robô navegando com a estratégia APF_k colide e o IR não. O robô adentra uma região de risco, com três MOs se movendo no padrão Onda e dois no padrão Linha. O IR identifica o agrupamento de MOs e realiza um desvio antecipado, se distanciando o máximo possível dos obstáculos.

O gráfico na Figura 5.22 mostra a qualidade média do caminho obtido pelas três estratégias para todos os grupos de MOs. Existe uma diferença que varia entre 0 a 2%

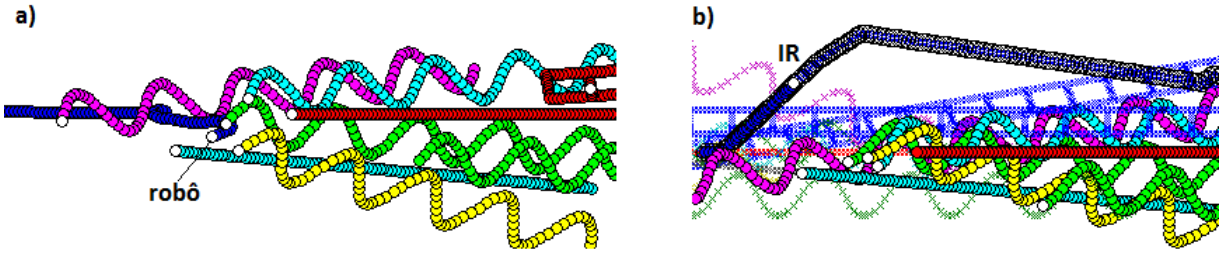

Figura 5.21: Situação de colisão para a estratégia APF_k em cenário com nove obstáculos: a) APF_k, b) PSOSMv2. Em a), o robô adentra em uma região com muitos MOs, culminando em uma colisão. Em b), o planejamento identifica a região perigosa e desvia o IR antecipadamente

na qualidade entre o PSOSMv2 e a estratégia APF_k, com uma pequena vantagem para o PSOSMv2, mas parece não haver qualquer relação com o número de MOs. O método APF_p tem sua qualidade afetada com o acréscimo de MOs e nos cenários com 9 MOs a qualidade do caminho do método PSOSMv2 é 7.16% maior. Independente da estratégia, o métodos PSOSMv2 obteve uma qualidade de caminho consistentemente maior para todos os cenários de simulação.

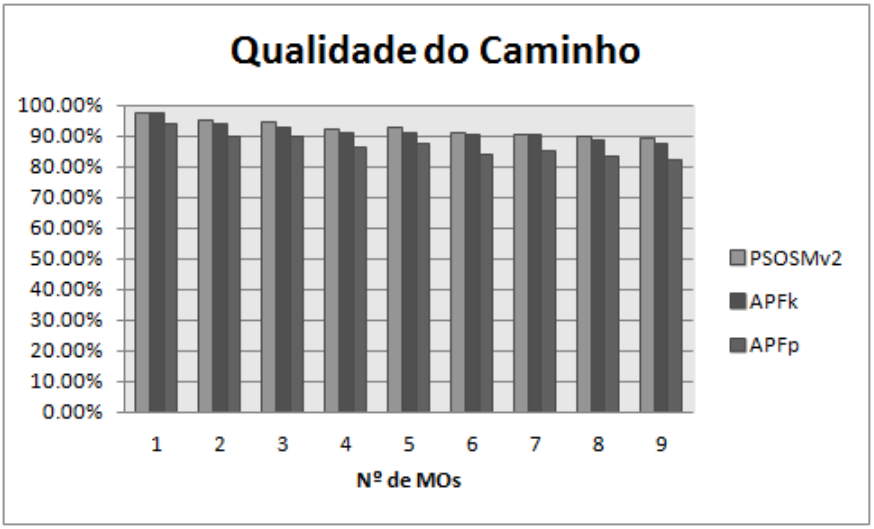

Figura 5.22: Qualidade do caminho para as estratégias E-LOFORM + PSOSMv2, APF_k e APF_p, considerando-se cada grupo de n MOs (nº de MOs x tempo (s))

A Tabela 5.3 mostra os resultados gerais referentes à qualidade do caminho para o FIRN ao aplicar o método PSOSMv2 e para as estratégias APF_k e APF_p. É possível verificar que além de ter obtido a maior média o PSOSMv2 também obteve os maiores valores para o mínimo e máximo caminhos alcançados e o menor desvio padrão. Como o número de colisões no PSOSMv2 também foi menor, pode-se dizer que a qualidade geral do caminho foi maior para o FIRN.

Tabela 5.3: Qualidade geral do caminho resultante de cada estratégia de navegação

	Média	Desvio Padrão	Mínimo	Máximo
PSOSMv2	92.59%	3.60%	76.67%	100.00%
APF_k	91.39%	5.98%	30.71%	99.57%
APF_p	86.71%	6.00%	65.00%	97.46%
Apenas para caminhos sem colisão				

A Figura 5.23 ilustra um caso onde a qualidade do caminho para o IR utilizando a estratégia PSOSMv2 é mais otimizada. Os caminhos foram destacados do simulador, a fim de simplificar a visualização. Como o robô adentra a área de perigo, tem que realizar mais curvas e ciclos a fim de alcançar o objetivo. O IR traça sua trajetória livre de confrontos diretos, projetando menos curvas. Ainda que ambos os caminhos sejam grandes, aquele projetado pelo PSOSMv2 sujeita o IR a menos desvios.

Neste Capítulo foram apresentados os métodos de planejamento propostos nesta dissertação: LADSM, PSOSMv1 e PSOSMv1. O método LADSM utiliza um mecanismo de antecipação da previsão para aferir a direção e velocidade dos obstáculos, além de diminuir o tempo de processamento ao avaliar a heurística apenas na identificação de MOs próximos. O método PSOSMv1 aplica um algoritmo de PSO para permitir que o IR tome qualquer direção e velocidade dentro do seu raio de deslocamento máximo em uma rodada. O método PSOSMv2, além das otimizações apresentadas nos demais métodos, procura na previsão por áreas congestionadas, fazendo com que o IR evite estes espaços.

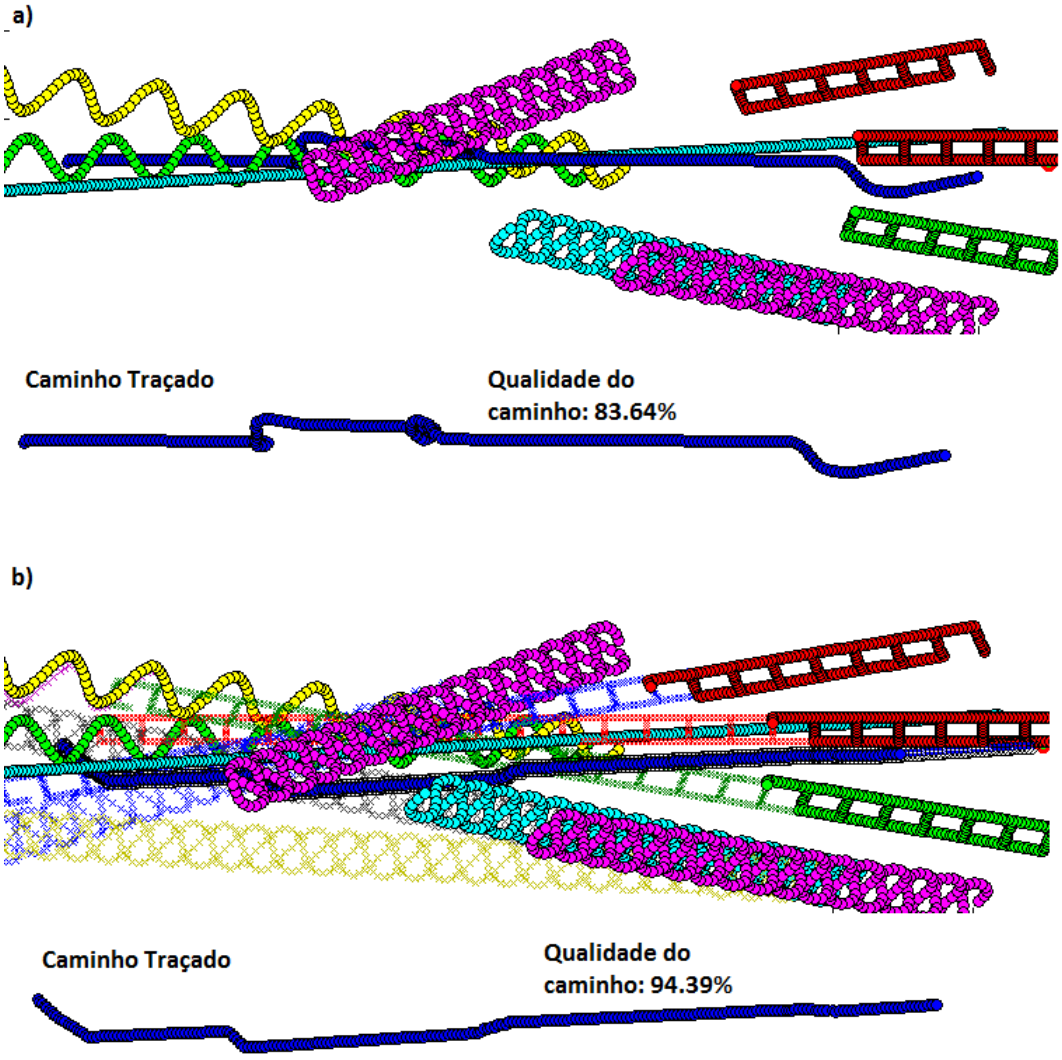

Figura 5.23: Exemplo da qualidade do caminho traçado por cada estratégia: a) APF_k, b) E-LOFORM + PSOSMv2. A estratégia APF_k sujeita o robô a desvios desnecessários, que afetam a qualidade final do caminho

A Tabela 5.4 apresenta os resultados gerais obtidos. Os métodos propostos LADSM, PSOSMv1 e PSOSMv2 conseguem taxas de colisão menores que as estratégias baseadas em APFs, sendo o maior resultado positivo de 14.91% para o método PSOSMv2 com relação à estratégia APF_k. A qualidade do caminho é tomada apenas em caminhos livres de colisão e, desta forma, só pode ser analisada relacionada à taxa de colisões. Dado que a qualidade do PSOSMv2 é maior e a taxa de colisões é menor que das estratégias APF_k e APF_p, é possível dizer que o FIRN consegue calcular rotas mais otimizadas no ambiente de aplicação.

Tabela 5.4: Resultados gerais para os métodos do FIRN e as estratégias baseadas em APFs

	Taxa de colisões	Qualidade do Caminho*	Tempo de Planejamento (s)
DSM3	55.93%	94.80%	0.351
DSM7	66.11%	92.59%	0.771
FDSM	61.76%	93.74%	0.255
LADSM	18.52%	94.40%	0.193
PSOSMv1	16.39%	93.17%	0.316
PSOSMv2	12.31%	92.36%	0.369
APF_k	27.22%	91.39%	N/A
APF_p	26.94%	86.71%	N/A
*Apenas para caminhos sem colisão			

6 CONCLUSÃO E TRABALHOS FUTUROS

O FIRN, um *Framework* para Navegação Inteligente de Robôs desenvolvido pelo grupo MASI, é aplicável a ambientes dinâmicos simulados. Um robô com navegação controlada pelo FIRN (IR, *Intelligent Robot*) é capaz de planejar rotas até um destino evitando obstáculos móveis (MO, *Moving Obstacle*) se deslocando segundo padrões determinísticos com velocidades superiores àquela do robô. O FIRN é constituído por quatro etapas:

- coleta de amostras: amostras de posições ocupadas pelos MOs são tomadas no ambiente;
- previsão: o movimento dos MOs é processado e uma projeção de suas trajetórias é construída;
- planejamento: a previsão é utilizada para calcular uma rota minimizando as colisões;
- navegação: o IR se desloca de acordo com a rota planejada.

Esta dissertação aborda os seguintes avanços relativos ao FIRN:

1. Desenvolvimento de um novo simulador (chamado Simulador FIRNn), permitindo a experimentação do FIRN com vários MOs;
2. Experimentos de avaliação dos métodos de planejamento existentes antes da pesquisa associada a esta dissertação;

3. Desenvolvimento de três novos métodos para a etapa de planejamento que contornam as deficiências encontradas nos métodos disponíveis anteriormente;

4. Experimentos de avaliação dos novos métodos de planejamento com relação aos métodos disponíveis anteriormente e estratégias reativas baseadas em APFs.

A partir dos experimentos citados no item 2 acima foram constatadas deficiências nos métodos de planejamento previamente desenvolvidos, o que levou ao desenvolvimento do item 3, os novos métodos de planejamento. As principais deficiências observadas foram:

(a) Subutilização da previsão por parte dos métodos de planejamento, pois estes consideram as posições previstas dos MOs de forma isolada, sem verificar características como direção ou velocidade dos obstáculos;

(b) Tempo de processamento elevado, especialmente para o método DSM7, comprometendo a previsão e o planejamento durante a navegação do IR;

(c) Limitação na capacidade de deslocamento do IR nos métodos DSM3 e DSM7, nos quais o IR só pode andar em três ou sete direções diferentes e com velocidade constante.

(d) Incapacidade dos métodos de identificar áreas congestionadas na previsão e evitá-las durante o planejamento.

O primeiro método proposto nesta dissertação, o LADSM (*Look Ahead Direct Search Method*), consulta um conjunto antecipado de previsões das posições dos obstáculos a cada instante de tempo durante o planejamento a fim de identificar *"a priori"* a direção de deslocamento dos mesmos. O tempo de processamento também é reduzido por se avaliar heurísticas de movimento apenas em situações de confronto com os obstáculos, tomando apenas um caminho em linha reta até o objetivo em caso contrário. Com isto, são resolvidas as deficiências (a) e (b) acima identificadas.

O método PSOSMv1 (*Particle Swarm Optimization Method* versão 1) utiliza um algoritmo de PSO (*Particle Swarm Optimization*) ao considerar novas posições no planejamento, flexibilizando as direções possíveis de serem tomadas pelo robô e a velocidade, que

deixa de ser constante, resolvendo então a deficiência (c) acima identificada. Como este método foi desenvolvido com base no LADSM, a otimização no tempo de processamento torna a estratégia viável no ambiente de aplicação.

O método PSOSMv2 (*Particle Swarm Optimization Method* versão 2) é uma atualização do PSOSMv1, capaz de identificar futuras áreas de risco com muitos MOs concentrados em um espaço delimitado, consultando os dados de previsão dos mesmos de forma relacionada, resolvendo a deficiência (d) identificada. A aplicação deste método introduz apenas um pequeno acréscimo no tempo de processamento mas evita colisões adicionais.

Com os novos métodos de planejamento desenvolvidos, o FIRN tornou-se um *framework* mais robusto e confiável. A aplicação do método PSOSMv2 no ambiente do Simulador FIRNn, onde os MOs possuem velocidades iguais ou maiores que o IR, mostrou que a utilização apropriada das informações derivadas da etapa de previsão leva o robô a menos colisões do que métodos estritamente reativos. As estratégias reativas baseadas em APFs introduziram o robô em áreas de risco, onde a probabilidade de colisões é elevada e qualquer tentativa de manobra diminui a qualidade geral do caminho. O FIRN aplicando a estratégia E-LOFORM + PSOSMv2 foi capaz de desviar destas áreas "*a priori*", evitando colisões e mantendo uma boa qualidade média do caminho. Os resultados mostraram que o índice de colisões do FIRN foi 14.91% menor que as estratégias baseadas em APFs e a qualidade do caminho foi consistentemente maior.

Certamente que são feitas algumas suposições para que o FIRN funcione no ambiente simulado, posto que os MOs possuem padrões de movimento determinísticos, a distância entre o IR e os MOs nos simuladores viabiliza a etapa de coleta de amostras e os sensores são considerados precisos, mas todas as estratégias, inclusive as reativas, foram avaliadas, por questão de coerência, sob as mesmas condições. Este resultado inicial incentiva investimentos futuros no *framework*, a fim de que o ambiente de aplicação do FIRN se aproxime cada vez mais do mundo real.

Como vantagem adicional, destaca-se que nos métodos PSOSMv1 e PSOSMv2 é possível aproximar o modelo para o mundo real de forma mais simples, moldando a área de deslocamento válida em volta do IR. Conforme mostrou a Seção 5.3, a área de deslocamento

pode ser definida de forma que alguns ângulos e velocidades sejam limitados, o que certamente facilita a aplicação em robôs que não podem tomar qualquer direção a todo momento. No entanto, o conjunto de parâmetros que devem ser definidos para que o FIRN funcione corretamente em um novo ambiente é maior que aquele para uma estratégia simples baseada em APFs, sendo necessário que algumas combinações sejam testadas para novos ambientes.

As estratégias baseadas em APFs utilizam mecanismos alternativos para lidar com "*deadlocks*" (situações onde o robô permanece preso atrás de uma barreira) em ambientes estáticos do tipo labirinto, como a técnica de "*wall-following*" (andar rente a uma parede até encontrar um novo caminho). Um robô no mundo real deve ser capaz de sair deste tipo de situação. Não foi objeto desta pesquisa como o FIRN, em seu estado atual, trataria deste tipo de ambiente, sendo necessário trabalhar esta característica no futuro.

Em termos de trabalhos futuros, visando notadamente preparar o FIRN para o mundo real, estão:

- A introdução de ruídos nos padrões de movimento dos MOS: tanto o Simulador FIRNn quanto o Simulador FIRNp lidam com obstáculos com padrões de movimento determinísticos. Como existem diversos ruídos introduzidos no mundo real por características físicas, como o atrito entre o meio de contato do obstáculo e o chão, e por imprecisões dos sensores e câmeras, é interessante abandonar o conceito de movimento determinístico nas simulações. Inicialmente algumas perturbações nestes padrões devem ser introduzidas, a fim de simular um comportamento menos previsível e avaliar o desempenho dos métodos de previsão atualmente disponíveis nestas situações;

- A identificação de novos MOs no ambiente após o início da simulação e a consequente reavaliação da previsão e planejamento: a identificação de novos MOs no ambiente de simulação representa uma situação onde um MO acaba de ser detectado pelos sensores e câmeras disponíveis. O comportamento natural nestas situações seria reavaliar a previsão e o planejamento, mas esta aplicação certamente traria diversos problemas que alongariam de forma proibitiva o tempo de desenvolvimento desta pesquisa. O IR teria que encontrar um local seguro para estacionar e reavaliar a trajetória, o que poderia acarretar uma colisão (dependendo do número de MOs e distância até o IR).

Uma possibilidade seria fazer o reprocessamento de forma global (reavaliar a previsão e planejamento relativo a todos os MOs) ou local (introduzir apenas as modificações necessárias para tratar apenas do novo MO). Outra proposta seria lidar com o novo obstáculo utilizando exclusivamente métodos reativos;

- O desenvolvimento de um mecanismo reativo de emergência para evitar obstáculos imprevistos ou cujas previsões fossem incorretas: como a proposta nesta pesquisa foi desenvolver métodos de planejamento que livrassem o IR de situações de colisão "*a priori*", foge deste escopo o desenvolvimento de métodos reativos de emergência. Esta questão terá de ser abordada novamente apenas quando os problemas acima listados (introdução de ruídos nos padrões de movimento ou de novos MOs no ambiente) forem tratados.

Referente à aplicação direta do FIRN no mundo real, é necessário inicialmente um estudo sobre os ambientes almejados:

- No momento, o grupo MASI empenha-se em transpor o ambiente simulado para um espaço de experimentação no mundo real, em uma configuração semelhante, contando com uma câmera perpendicular ao ambiente de navegação e um robô físico simulando os padrões de movimento determinísticos. Deste desenvolvimento, medidas de distância e tempo são tomadas a fim de calcular a velocidade e distâncias necessárias para que a coleta de amostras de posições e o processamento dos métodos de previsão e planejamento sejam realizados sem colisão;

- Na navegação em espaços dentro de edifícios, existe uma oportunidade de aplicação imediata. Dado o aspecto modular do FIRN, algumas aplicações com mecanismos de previsão apresentados em outras pesquisas podem ser utilizados em conjunto com o módulo de planejamento. Por exemplo, o sistema de previsão a longo e curto prazo proposto por FOKA e TRAHANIAS (2010) determina o campo de visão dos obstáculos e, de acordo com os objetivos demarcados no ambiente identificados na frente dos mesmos, estima a trajetória provável (longo prazo); uma PNN (*Polynomial Neural Network*) é usada para calcular mudanças locais na trajetória devido a outros obstáculos ou barreiras (curto prazo). Este sistema poderia ser acoplado nos módulos de

coleta de amostras e previsão do FIRN, já que a trajetória resultante poderia ser convertida no modelo esperado pelos métodos de planejamento (listas mo_{tri} de posições futuras esperadas). A previsão a curto prazo funcionaria como reavaliação da previsão, podendo então o IR replanejar a rota ou ativar um mecanismo de reação emergencial. Abordagens semelhantes podem ser tomadas a partir das pesquisas de QIAN et al. (2010) e SASAKI, BRSCIC e HASHIMOTO (2010). O único problema a ser resolvido é a maneira como o IR irá tratar os obstáculos estáticos não pontuais, como barreiras, paredes, portais etc.

Outros desenvolvimento suplementares ainda no ambiente simulado poderiam ser:

- Aplicação do Simulador FIRNn em um ambiente de computação paralela: atualmente, o processamento é realizado em apenas um processador, que tem que lidar com o processamento dos métodos de previsão e planejamento, além da navegação do IR, MOs e os desenhos na área de simulação. A partir de dez MOs, o simulador começa a apresentar lentidão (levando em consideração a configuração do computador utilizado) na execução por ter que lidar com tantos objetos simultaneamente. Os resultados obtidos nesta dissertação são promissores e podem melhorar ainda mais caso as simulações sejam realizadas em um ambiente de computação paralela. Tendo uma máquina responsável pela inteligência e movimento do IR, uma pelo movimento dos obstáculos e uma última lidando com os detalhes operacionais do simulador, a carga do processador relativo ao IR seria aliviada, permitindo maior velocidade na execução;

- Aplicação de processamento paralelo na etapa de previsão do FIRN: como as trajetórias dos obstáculos não são relacionadas e os métodos de previsão as processam independentemente, um ganho de tempo considerável pode ser obtido na paralelização desta etapa;

- Desenvolvimento de novos métodos de previsão: ainda que o método ANNPM, baseado na aplicação de Redes Neurais tenha sido deixado de lado nesta dissertação, uma nova tentativa utilizando Redes com memória certamente possibilitaria a identificação de padrões de movimento como o "Quadrado". Além disso, um banco de previsões

conhecidas poderia ser elaborado, de forma que Redes já treinadas fossem armazenadas e consultadas quando novos obstáculos com padrão de movimento semelhantes fossem identificados. Como em um ambiente de aplicação espera-se que os obstáculos tenham padrões de movimento em comum, esta medida pouparia tempo de processamento ao evitar que o treinamento das Redes fosse realizado em todas as situações.

- Desenvolvimento de novos métodos de planejamento: nesta pesquisa decidiu-se por aprimorar o método DSM, eliminando as restrições impostas ao movimento e diminuindo o tempo de processamento. O método FDSM também poderia ser explorado, inicialmente com uma modelagem que considere vários MOs simultaneamente, ao invés de apenas o mais próximo. Um mecanismo de antecipação parecido com o do método LADSM também poderia ser instaurado, de forma que o controlador nebuloso possa prever a direção do movimento dos obstáculos e ajustar sua velocidade. O resultado seria um método robusto e com tempo adequado de processamento.

REFERÊNCIAS

AGIRREBEITIA, J. et al. A new APF strategy for path planning in environments with obstacles. **Mechanism and Machine Theory**, Elmsford, NY, v. 40, n. 6, p. 645-658, Jun. 2005.

AGRAWAL, R. ; RAMAKRISHNAN, S. Fast algorithms for mining association rules. In: VLDB CONFERENCE, 20., 1994, Santiago, Chile. **Proceedings…** San Francisco: Morgam Kaufmmann, 1994. p.487-499.

_____. Mining sequential patterns. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 11., 1995, Taipei, Taiwan. **Proceedings …** New York: IEEE, 1995. p.3-14.

AMARNATH, J.; SINGH, B. P.; KAMAKSHAIAH, S.; RADHAKRISHNA, C. Determination of Particle Trajectory in Gas insulated Bus Duct by Monte-Carlo Technique. In: CONFERENCE ON ELECTRICAL INSULATION AND DIELECTRIC PHENOMENA, 1999. **Proceedings. . .** IEEE, 1999. v.1, n.1, p.399-402.

ARKIN, R. C. ; MACKENZIE, D. C. **Planning to behave: a hybrid deliberative/reactive robot control architecture for mobile manipualtion**. Atlanta: Georgia Institute of Technology, 1994.

BEER, R. D. et al. Biorobotic approaches to the study of motor systems. **Current Opinion in Neurobiology**, London, v. 8, n. 6, p. 777-782, Dec. 1998.

BRAND, M. et al. Ant colony optimization algorithm for robot path planning. In: 2010 CONFERENCE ON COMPUTER DESIGN AND APPLICATIONS, 2010, Qinhuangdao, Hebei, China. **Proceedings …** Piscatawey, NJ: IEEE, 2010. p.436-440.

BROOKS, R. A. A Robust layered control system for a mobile robot. **Journal of Robotics and Automation**, New York, v. 2, n. 1, p. 14-23, Mar. 1986.

CANGELOSO, S. **iRobot roomba 700 series on sale as of today**. Abr. 2011. Disponível em: <http://www.geek.com/articles/gadgets/irobot-roomba-700-series-on-sale-as-of-today-20110421/>. Acesso em: 12 out. 2011.

CANNY, J. A Computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Learning**, New York**, v. 8, n. 6, p. 679-698, Nov. 1986.

COLORNI, A. ; DORIGO, M. ; MANIEZZO, V. Distributed optimization by ant colonies. In: VARELA, F. J. ; BOURGINE, P. (Eds.) **Toward a practiceof autonomous systems.** Proceedings of the first european conference on artificial life, Paris, 1991. Cambridge: MIT Press, 1991. p.134-142.

COSTA, A. H. R. ; PACHECO, R. N. Navegação de robôs móveis utilizando o método de campos potenciais. In: WORKCOMP'2002 - WORKSHOP DE COMPUTAÇÃO, 2., 2002. São José dos Campos. **Anais....** São José dos Campos: ITA, 2002. v. 1. p.125-130.

ESCOBEDO, J. ; MANSOORI, G. A. Prefouling behavior of suspended particles in pretroleum fluid flow. **Transactions C: Chemistry and Chemical Engineering**, [S.l.], v.1 7, n. 1, p.77-85, 2010.

FOKA, A. F. ; TRAHANIAS, P. E. Probabilistic autonomous robot navigation in dynamic environments with human motion prediction. **Internatinal journal of. Social Robotics**, Amsterdam, v. 2, n. 1, p.79-94, Mar. 2010.

GE, S. S. ; CUI, Y. J. Dynamic motion planning for mobile robots using potential field method. **Autonomous Robots**, Dordrecht, v. 13, n. 3, p. 207-222, 2002.

HART, P. E. ; NILSSON, N. J. ; RAPHAEL, B. A Formal basis for the heuristic determination of minimum cost paths. **IEEE Transactions of Systems Science and Cybernetics**, New York, v. 4, n.2, p.100-107, 1968.

HORNYAK, T. **Rescue robots deployed in Japan earthquake ops**. Mar. 2011. Disponível em: <http://news.cnet.com/8301-17938_105-20042945-1.html>. Acesso em: out. 12, 2011.

ITOH, C. ; KATO, S. ; ITOH, H. Mood-transition-based emotion generation model for the robots personality. In: IEEE SMC, 2009 – 2009 IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS. 2009. San Antonio TX. **Proceedings …** New York: IEEE, 2009. p. 2878-2883.

JARADAT, M. A. K. ; AL-ROUSAN, M. ; QUADAN, L. Reinforcement based mobile robot navigation in dynamic environment. **Robotics and Computer-Integrated Manufacturing**, New York, v. 27, n. 1, p. 135-149, 2011.

JULIÁ, M. et al. Potential field based integrated exploration for multi-robot teams. In: **ICINCO – RA, 2.** 2008. INTERNATIONAL CONFERENCE ON INFORMATICS IN CONTROL, AUTOMATICS AND ROBOTICS, 5., Robotics and Automation 2., 2008. Funchal Madeira, Portugal. **Proceedings …** Setubal: INSTICC Pres, 2008. p. 308-314.

JULIÁ, M. et al. A Hybrid solution to the multi-robot integrated exploration problem. **Engeneering. Applications. of Artifcial Intelligence**, [S.l.], v. 23, n. 4, p. 473-486, June 2010.

KARABOGA, D. **An idea based on honey bee swarm for numerical optimization**. Kayseri, Turkey: University of Erciyes, Faculty of Engineering, 2005. (TR 06).

KENNEDY, J. ; EBERHART, R. Particle swarm optimization. In: PROCEEDINGS OF THE IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS, 1995, Perth, WA. **Proceedings …** Perth WA: IEEE, 1995. p.1945-1948.

KHATIB, O. Real-time obstacle avoidance for manipulators and mobile robots. **The International Journal of Robotics Research**, Cambridge Mass, v. 5, n. 1, p. 90-98, 1986.

KIM, J.-H. ; KIM, Y.-D. ; LEE, K.-H. The Third generation of robotics: ubiquitous robot. In: INTERNATIONAL CONFERENCE ON AUTONOMOUS ROBOTS AND AGENTS, 2., 2004, Palmerston North, New Zealand. **Proceedings ...** Palmerston North, New Zealand: IIST, 2004. p. 1-7.

KIM, P. G. et al. Obstacle avoidance of a mobile robot using vision system and ultrasonic sensor. In: ADVANCED INTELLIGENT COMPUTING THEORIES AND APPLICATIONS. WITH ASPECTS OF

THEORETICAL AND METHODOLOGICAL ISSUES, 3., 2007, Qingdao China. **Proceedings …** Heidelberg: Springer, 2007. p.545-553.

LEE, L.-F. ; KROVI, V. A Standardized testing-ground for artificial potential-field based motion planning for robot collectives. In: PERFORMANCE METRICS FOR INTELLIGENT SYSTEMS WORKSHOP, 6., 2006, Washington DC. **Proceedings ...** Gaithersburg MD: NIST, 2006. p. 232-239.

LU, L. ; GONG, D. Robot path planning in unknown environments using particle swarm optimization. In: INTERNATIONAL CONFERENCE ON NATURAL COMPUTING, 4., 2008, Jinan, China. **Proceedings …** Los Alamitos: IEEE, 2008. v. 4. p.422-426.

MARTEL, S. Nanorobots for microfactories to operations in the human body and robots propelled by bacteria. **Facta Universitatis. Series, Mechanics, Automatic Cotrol and Robotics**, Nis Serbia, v. 7, n. 1, p. 1-8, 2008.

MENG, W. ; XUEDONG, H. A Robot's navigation method based on information fusion in dynamic unknown environment. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL INTELLIGENCE AND SOFTWARE ENGINEERING, 2009. Wuhan China. **Proceedings …** Piscataway NJ: IEEE, 2009. p. 1-4.

MILLONAS, M. M. Swarms, phase transition, and collective intelligence. In: LANGTON, C. (Ed.) **Artificial life III.** Reading, MA: Addison Wesley, 1993. p. 417-445.

QIAN, K. et al. Robotic etiquette: socially acceptable navigation of service robots with human motion pattern learning and prediction. **Journal of Bionic Engineering**, [S.l.], v. 7, n. 2, p. 150-160, Jun. 2010.

QIN, Y.-Q. et al. Path planning for mobile robot using the particle swarm optimization with mutation operator. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND CYBERNETICS, 3., 2004, Shangai, China. **Proceedings …** Shangai, China: IEEE, 2004. v. 4. p. 26-29.

RASMUSSEN, C. E. ; WILLIAMS, C. K. I. Gaussian markov processes. In: DIETTERICH, T. (Ed.). **Gaussian processes for machine learning**. Cambridge Massachusetts: MIT Press, 2006. p. 207-219.

ROSEN, C. A. et al. **Research on intelligent automata**. Melon Park, CA: Stanford Research Institute, 1969. (Proposal for Resarch, SRI no. ESU 69-68)

ROSENBLATT, F. The Perceptron: a probabilistic model for information storage and organization in the brain. **Psychological Review**, Washington DC, v. 65, n. 6, p. 386-408, Nov. 1958.

RUSSEL, S. J. ; NORVIG, P. **Artificial intelligence** - a modern a pproach. 1. ed. Upper Saddle River NJ: Prentice Hall, 1995.

SAFFARI, M. H. ; MAHJOOB, M. J. Bee colony algorithm for real-time optimal path planning of mobile robots. In: INTERNATIONAL CONFERENCE ON SOFT COMPUTING COMPUTING WITH WORDS AND PERCEPTIONS IN SYSTEM ANALYSIS DECISION AND CONTROL, 5., 2009, Famagusta, Chipre. **Proceedings …** Famagusta, Chipre IEEE, 2009. p.1-4.

SAHU, K. B. M. ; AMARNATH, J. Effect of various parameters on the movement of metallic particles in a single phase gas insulated bus duct with image charges and dielectric coated electrodes. **ARPN Journal of Engineering and Applied Sciences**, [S.l.], v. 5, n. 6, p. 52-60, June 2010.

SARIFF, N. ; BUNIYAMIN, N. An overview of autonomous mobile robot path planning algorithms. In: SCOReD 2006 STUDENT CONFERENCE ON RESEARCH AND DEVELOPMEN. 4., 2006. Selangor, Malaysian. **Proceedings …** Selangor, Malaysian: IEEE, 2006. p. 183-188.

SASAKI, T. ; BRSCIC, D. ; HASHIMOTO, H. Human-observation-based extraction of path patterns for mobile robot navigation. **Journal of Bionic Engineering**, [S.l.], v. 57, n. 4, p. 1401-1410, Apr. 2010.

SFEIR, J. ; KANAAN, H. Y. ; SAAD, M. A neural-network-based path generation technique for mobile robots. In: ICM'04 - IEEE INTERNATIONAL CONFERENCE ON MECHATRONICS. 2004. Istanbul. **Proceedings ...** Piscataway, NJ: IEEE, 2004. v.5, n.3, p. 176-181.

SINGH, M. K. ; PARHI, D. R. Intelligent neuro-controller for navigation of mobile robot. In: ICAC3-09 INTERNATIONAL CONFERENCE ON ADVANCES IN COMPUTING, COMMUNICATION AND CONTROL, 2009. Mumbai Indias. **Proceedings …** New York: ACM, 2009. p. 123-128.

STONE, P. ; VELOSO, M. M. Multiagent systems: a survey from a machine learning perspective. **Autonomous Robots**, Dordrecht, v. , n. 3, p. 345-383, 2000.

TAN, S. ; ZHU, A. ; YANG, S. X. A GA-based fuzzy logic approach to mobile robot navigation in unknown dynamic environments with moving obstacles. In: IEEE INTERNATIONAL CONFERENCE ON GRANULAR COMPUTING, 2009. Nanchang, China. **Proceedings …** Nanchang, China: IEEE, 2009. p. 529-534.

TSUBOUCHI, T. ; ANIMOTO, S. Behavior of a mobile robot navigated by an iterated forecast and planning scheme in the presence of multiple moving obstacles. In: 1994 INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 1994. New Orleans. **Proceedings …** New Orleans: IEEE, 1994. v. 3. p. 2470-2475.

TSUBOUCHI, T. ; KURAMOCHI, S. ; ANIMOTO, S. Iterated forecast and planning algorithm to steer and drive a mobile robot in the presence of multiple moving objects. In: 1995 IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, HUMAN ROBOTIC INTERACTION AND COOPERATIVE ROBOTIS. 1995. Pittsburgh Pennsylvania. **Proceedings …** Washington DC: IEEE, 1995. v. 2, p. 33-38.

TSUBOUCHI, T. et al. A Mobile robot navigation scheme for an environment with multiple moving obstacles. In: INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, 1992. Raleigh NC. **Proceedings ...** Piscataway NJ: IEEE, 1992. v. 3, p. 1791-1798.

VADAKKEPAT, P. ; TAN, K. C. ; MING-LIANG, W. Evolutionary artificial potential fields and their application in real time robot path planning. In: CEC00 - CONGRESS ON EVOLUTIONARY COMPUTATION, 2000. La Jolla, CA. **Proceedings …** Piscataway NJ: IEEE, 2000. v. 1. p. 256-263.

VOKRINEK, J. ; KOMENDA, A. ; PECHOUCEK, M. Cooperative agent navigation in partially unknown urban environments. In: INTERNATIONAL SYMPOSIUM ON PRACTICAL COGNITIVE AGENTS AND ROBOTS, 3., 2010. Toronto. **Proceedings ...** New York**:** ACM, 2010. p. 41-48.

VOLPE, R.; KHOSLA, P. Manipulator control with superquadric artificial potential functions: theory and experiments. **IEEE Transactions on Systems, Man and Cybernetics**, New York, v. 20, n. 6, p. 1423-1436, Nov./Dec. 1990.

WARREN, C. W. Global path planning using artificial potential fields. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 1989. Scottsdade, Arizona. **Proceedings …** Washington DC: IEEE, 1989. v. 1. p. 316-321.

WEBSTER, G. **New animation depicts next mars rover in action**. Jun. 2011. Disponível em: <http://www.nasa.gov/mission_pages/msl/news/msl20110624.html>. Acesso em: 12 out. 2011.

WEISS, R. J. ; CRAIGER, J. P. Ubiquitous computing. **TIP: The Industrial-Organizational Psychologist**, Bowling Green OH,, v. 39, n. 4, p. 44-52, Apr. 2002.

WITTEN, I. H. ; FRANK, E. **Data mining**: practical machine learning tools and techniques. 2. ed. San Francisco: Elsevier, 2005.

XU, L. et al. The Mobile robot navigation in dynamic environment. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND CYBERNECTICS, 2007. Hong Kong. **Proceedings …** Hong Kong: IEEE, 2007. v. 1. p. 566-571.

YIN, L.; YIN, Y. An improved potential field method for mobile robot path planning in dynamic environments. In: WORLD CONGRESS ON INTELLIGENT CONTROL AND AUTOMATION, 7., 2008. Chongqing China. **Proceedings …** Chongqing China: IEEE, 2008. p. 4847-4852.

ZADEH, L. A. Fuzzy sets. **Information and Control**, New York, v. 8, n. 3, p. 338-353, 1965.

ZHANG, H. ; LIU, S. ; YANG, S. X. A Hybrid robot navigation approach based on partial planning and emotion-based behavior coordination. In: IROS – IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, 2006. Beijing. **Proceedings ...** East Lansing MI: IEEE, 2006. p. 1183-1188.

ZHU, Y-F. ; TANG, X-M. Overview of swarm intelligence. In: 2010 INTERNATIONAL CONFERENCE ON COMPUTER APPLICATION AND SYSTEM MODELING, 2010, Shanxi, Taiyuan. **Proceedings …** Piscataway NJ: IEEE, 2010. v. 9. p. 400-403.

APÊNDICE A ALGORITMOS

Algoritmo 1 Elaborando mo_{tr} a partir do resultado das Redes Neurais

1: **para** cada MO, **faça**
2: Tome os últimos n valores de velocidade do MO e p_{mo}, sua última posição conhecida;
3: Forneça os n valores de velocidade como entrada para as Redes ANN_x e ANN_y;
4: **enquanto** mo_{tr} não estiver completo **faça**
5: ANN_x computa o valor $x_{v(n+1)}$, a componente x da velocidade esperada para o MO;
6: ANN_y computa o valor $y_{v(n+1)}$, a componente y da velocidade esperada para o MO;
7: Tome pv como sendo $(x_{v(n+1)}, y_{v(n+1)})$;
8: Calcule $(p_{mo} + pv)$, a próxima posição estimada para o MO, insira em MO_{TR} e faça dele o próximo p_{mo};
9: Remova a entrada mais antiga de ANN_x e ANN_y e substitua por pv;
10: **fim enquanto**
11: Retorne mo_{tr};
12: **fim para**

Funções utilizadas pelo Algoritmo 2:

- Def find_sequence(S_{min}, F_{min}, *list*, *start*): retorna a primeira sequência em *list* com tamanho = S_{min} e frequência $\geq F_{min}$ a partir da posição *start* de *list*;
- Def calculate_frequency(*sequence*, *list*): retorna a frequência de *sequence* em *list*;
- Def append_next_symbol(*sequence*, *list*): retorna *sequence* acrescido do próximo símbolo consecutivo em *list*;
- Def return_size(*sequence*): retorna o tamanho de *sequence*;

Algoritmo 2 AprioriFRN

```
1: start = 1;
2: cp = Vazio;
3: enquanto sequence_temp != Vazio faça
4:    sequence_temp = find_sequence(S_min, F_min, list, start);
5:    se sequence_temp != Vazio então
6:       F_temp = calculate_frequency(sequence_temp, list);
7:       F_sequence = F_temp;
8:       enquanto F_temp >= F_sequence faça
9:          sequence = sequence_temp;
10:         sequence_temp = append_next_symbol(sequence, list);
11:         F_temp = calculate_frequency(sequence, list);
12:      fim enquanto
13:      se return_size(sequence) > return_size(cp) então
14:         cp = sequence
15:      fim se
16:      start = start + return_size(word);
17:   fim se
18: fim enquanto
19: retorne cp;
```

Algoritmo 3 Calculando MS_x e MS_y no Método E-LOFORM

```
1: Tome mo_lvx, a lista de velocidades de MO no eixo x;
2: Tome mo_lvy, a lista de velocidades de MO no eixo y;
3: MS_x = 0;
4: MS_y = 0;
5: s_x = 0;
6: s_y = 0;
7: para i de 0 a (size(mo_lvx) − 1) faça
8:    s_x = |mo_lvx(i)|;
9:    se s_x > MS_x então
10:      MS_x = s_x;
11:   fim se
12:   s_y = |mo_lvy(i)|;
13:   se s_y > ms_y então
14:      ms_y = s_y;
15:   fim se
16: fim para
17: retorne ms_x e ms_y;
```

Algoritmo 4 Calculando ir_{tr} no Método DSM

1: Tome mo_{trk}, a trajetória esperada do MO_k, onde $k = 1$, ..., n e n é o número de MOs;
2: Tome p_{ir0}, a posição inicial do IR;
3: Tome p_g, a posição do objetivo;
4: Tome irs, a velocidade do IR;
5: p_{irx} = componente x de p_{ir0};
6: p_{iry} = componente y de p_{ir0};
7: $i = 0$, $e = \nearrow \infty$;
8: $ir_{tr}(i) = p_{ir0}$
9: **enquanto** $(ir_{tr}(i)! = p_g)$ **faça**
10: **enquanto** (ainda houverem ângulos de direção α disponíveis) **faça**
11: $p_{temp} = (p_{irx} + \cos(\alpha) * irs\ ,\ p_{iry} + \sin(\alpha) * irs)$;
12: Calcule heurística de p_{temp}, tomando $mo_{trk}(i)$, $\forall k \in n$;
13: **se** $(h < e)$ **então**
14: $e = h$;
15: $p = p_{temp}$
16: **fim se**
17: **fim enquanto**
18: $i = (i + 1)$, $e = \nearrow \infty$;
19: $ir_{tr}(i) = p$;
20: p_{irx} = componente x de p;
21: p_{iry} = componente y de p;
22: **fim enquanto**
23: retorne ir_{tr};

Algoritmo 5 Calculando ir_{tr} no Método LADSM

1: Tome mo_{trk}, a trajetória esperada do MO_k, onde $k = 1$, ..., n e n é o número de MOs;
2: Tome p_{ir0}, a posição inicial do IR;
3: Tome p_g, a posição do objetivo;
4: Tome irs, a velocidade do IR;
5: p_{irx} = componente x de p_{ir0};
6: p_{iry} = componente y de p_{ir0};
7: $i = 0$, $e = \nearrow \infty$;
8: $ir_{tr}(i) = p_{ir0}$
9: **enquanto** $(ir_{tr}(i)! = p_g)$ **faça**
10: **se** $\min(\mathrm{dist}(ir_{tr}(i), mo_{trk}(i)) \geq fr, \forall k \in n$ **então**
11: $p = (p_{irx} + \cos(\beta) * irs$, $p_{iry} + \sin(\beta) * irs)$, onde β é o ângulo entre o IR e o objetivo;
12: **senão**
13: **enquanto** (ainda houverem ângulos de direção α disponíveis) **faça**
14: $p_{temp} = (p_{irx} + \cos(\alpha) * irs$, $p_{iry} + \sin(\alpha) * irs)$;
15: Calcule heurística de p_{temp}, tomando $mo_{trk}(i)$, $\forall k \in n$;
16: **se** $(h < e)$ **então**
17: $e = h$;
18: $p = p_{temp}$
19: **fim se**
20: **fim enquanto**
21: **fim se**
22: $i = (i + 1)$, $e = \nearrow \infty$;
23: $ir_{tr}(i) = p$;
24: p_{irx} = componente x de p;
25: p_{iry} = componente y de p;
26: **fim enquanto**
27: retorne ir_{tr};

APÊNDICE B ESQUEMAS PARA ELABORAÇÃO DE CENÁRIOS DE NAVEGAÇÃO: 1 A 9 OBSTÁCULOS

A Figura B.1 fornece a base para a elaboração dos cenários de simulação com relação à posição inicial e direção dos MOs. O IR está fixado em sua posição inicial e o espaço de simulação foi subdivido em diversas pequenas áreas, numeradas de 1 a 16, onde os MOs podem ser alocados. Quando um MO é inserido em uma área deste tipo, ele pode tomar qualquer posição dentro dos limites da área, mas deve ter sua direção apontada para o IR, garantindo um conflito na simulação. As áreas "L1" até "L12" são mais limitadas, já que a posição dos MOs só pode ser deslocada no eixo x.

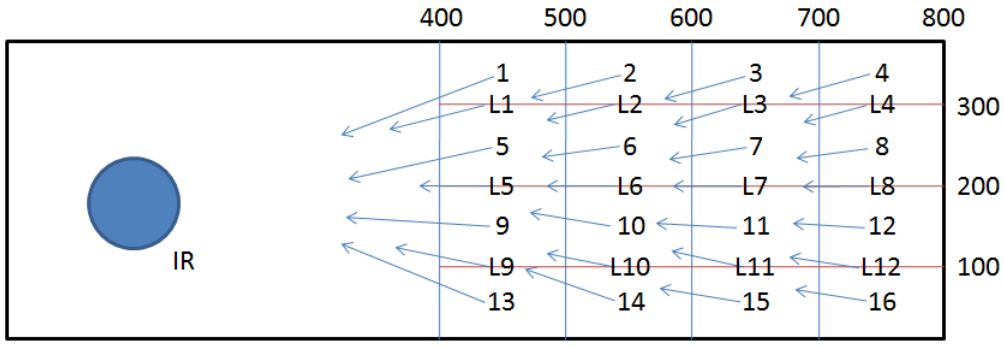

Figura B.1: Esquema geral para geração de cenários de teste. Os MOs são distribuídos nas regiões numeradas de 1 a 16 e de "L1" até "L12". As setas indicam a direção do movimento

Os esquemas a seguir representam a disposição inicial dos MOs para cenários contemplando 1 até 9 MOs. Estes esquemas representam de forma simplificada a Figura B.1. São três sub-esquemas para cada grupo de MOs, totalizando 120 casos de simulação para

cada grupo. Destes 120 casos, 40 são elaborados usando-se o esquema a), 40 usando-se o esquema b), e 40 usando-se o esquema c). Uma seta que intercepta um conjunto de quatro áreas nos esquemas a seguir indica que existe um MO que ocupará cada uma destas áreas em 10 simulações. A posição dentro da área é definida aleatoriamente, mas a direção do MO é representada na Figura B.1.

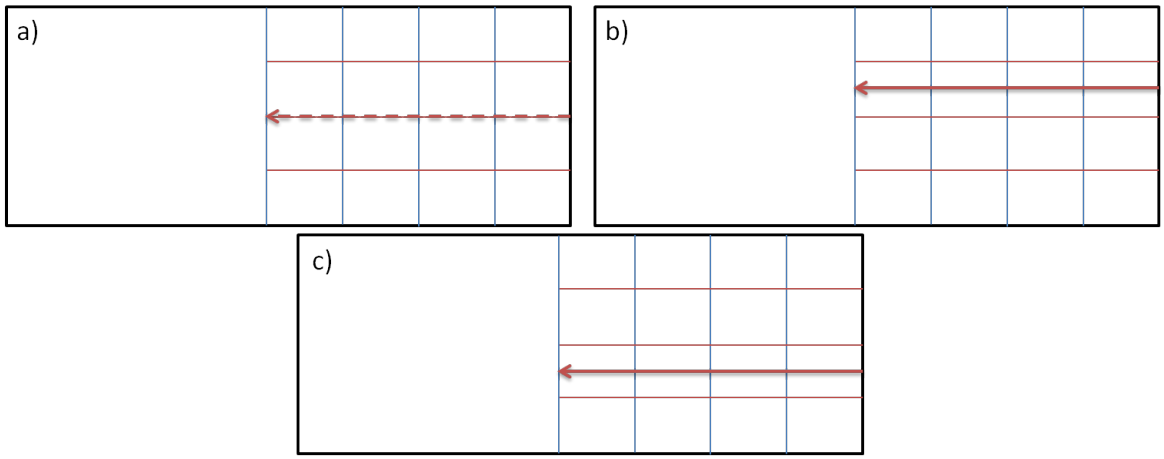

Figura B.2: Esquema para geração de cenários de teste para um obstáculo

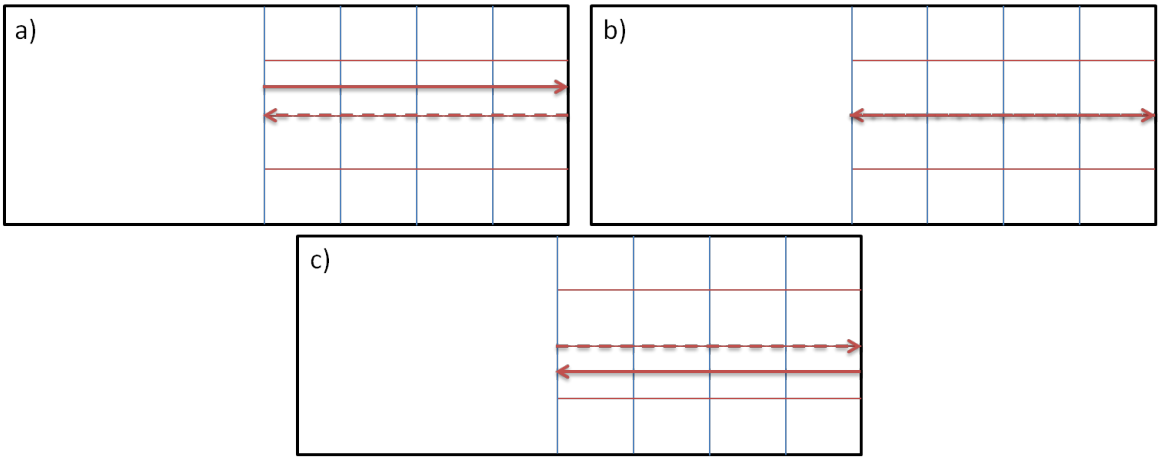

Figura B.3: Esquema para geração de cenários de teste para dois obstáculos

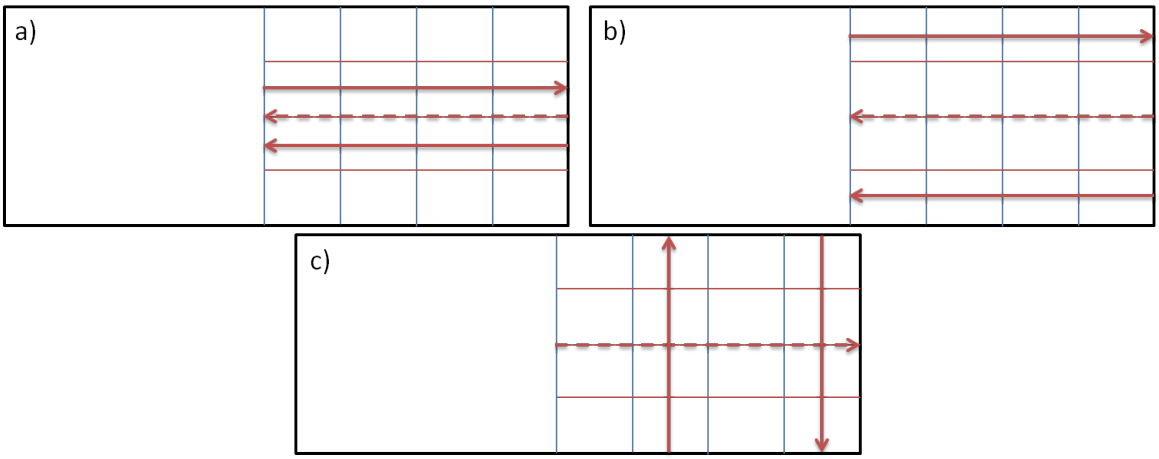

Figura B.4: Esquema para geração de cenários de teste para três obstáculos

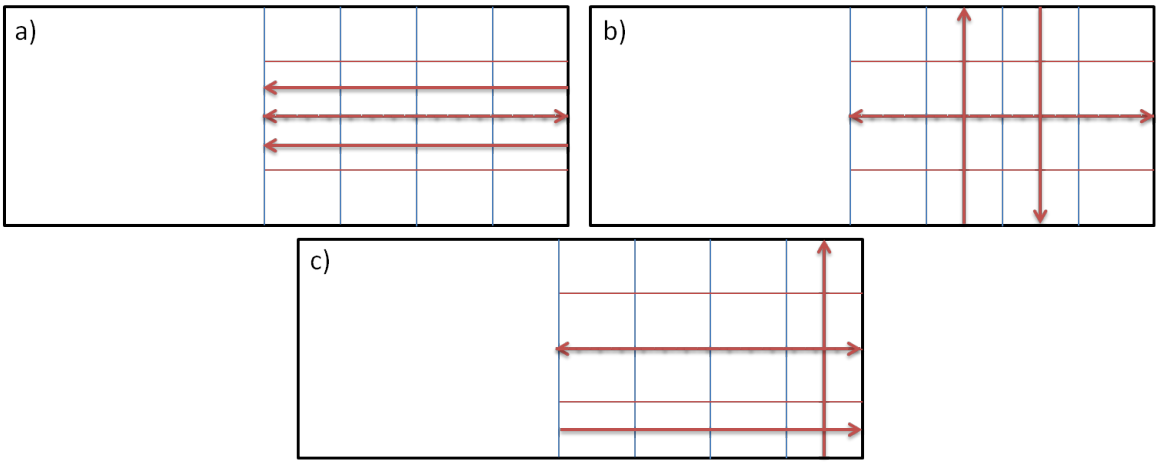

Figura B.5: Esquema para geração de cenários de teste para quatro obstáculos

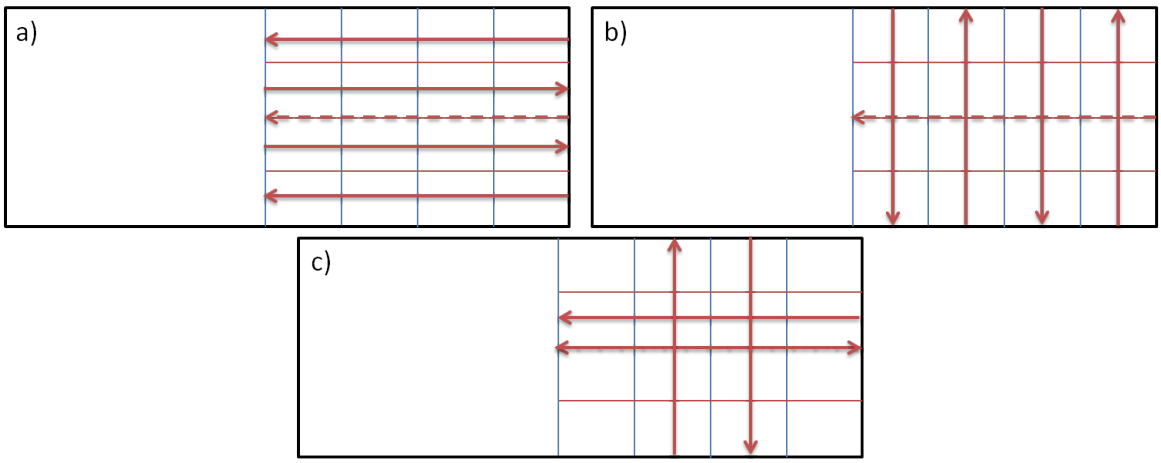

Figura B.6: Esquema para geração de cenários de teste para cinco obstáculos

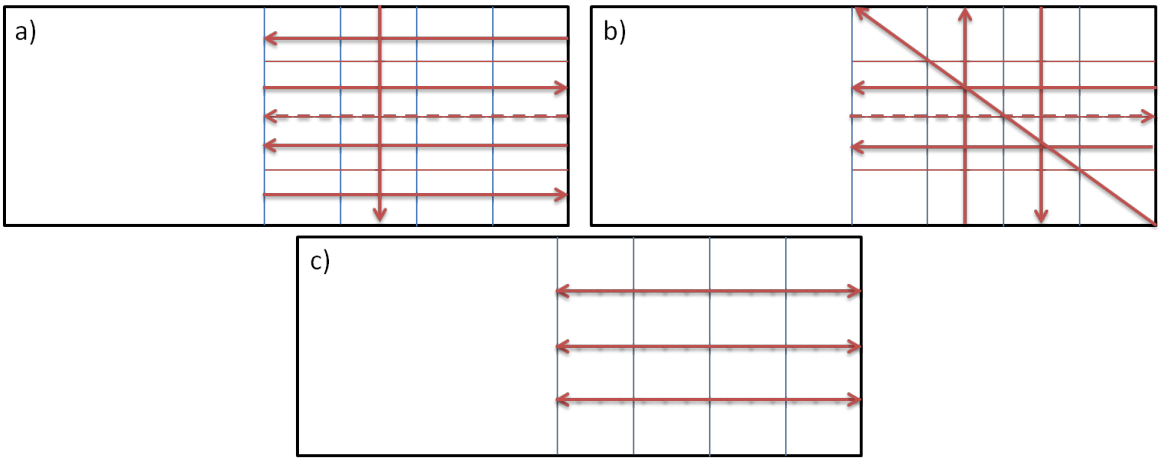

Figura B.7: Esquema para geração de cenários de teste para seis obstáculos

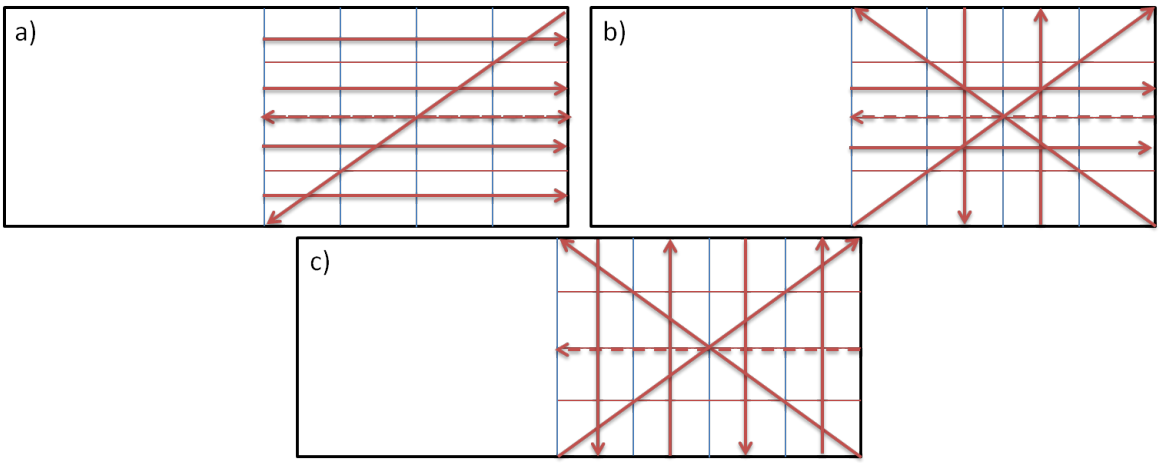

Figura B.8: Esquema para geração de cenários de teste para sete obstáculos

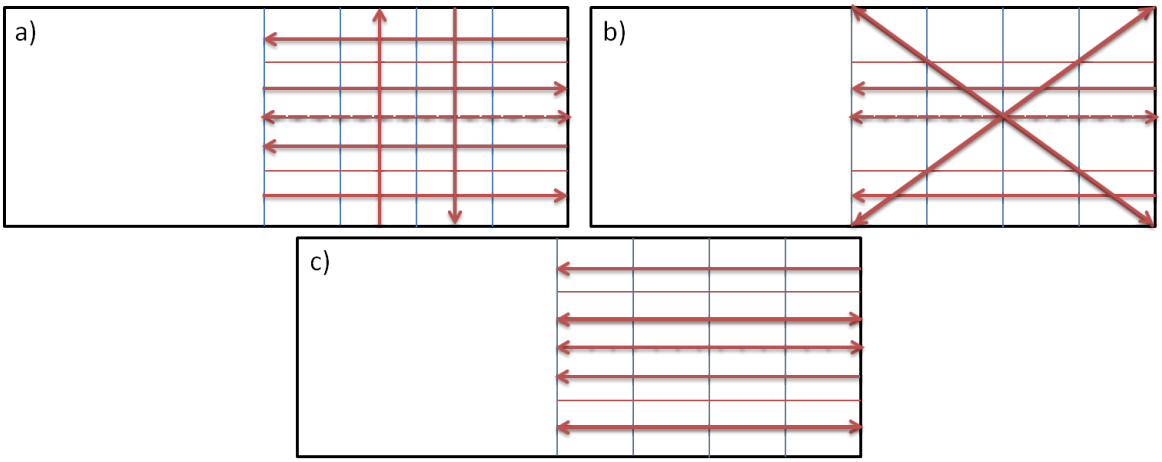

Figura B.9: Esquema para geração de cenários de teste para oito obstáculos

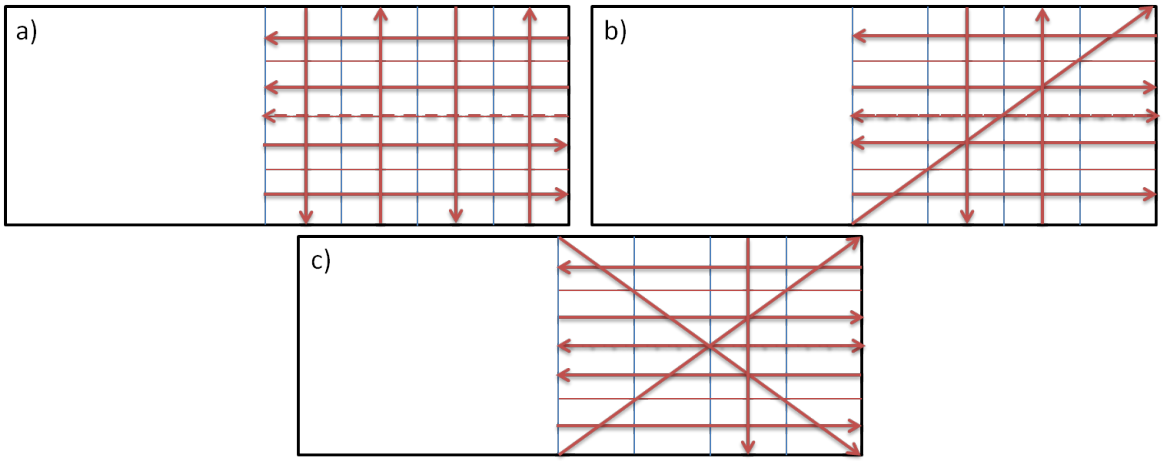

Figura B.10: Esquema para geração de cenários de teste para nove obstáculos

APÊNDICE C INSTRUÇÕES DO SIMULADOR FIRNN

Para solicitar uma cópia do Simulador FIRNn: Enviar email para renanmlobo at gmail.com

Versão necessária do Matlab: R2009b ou superior (deve suportar estrutura de classes);

Instalação:

- Descompactar o arquivo .rar em uma pasta;
- Inserir a pasta PSOt ao caminho de execução do Matlab (*File -> Set Path -> Add with Subfolders*, reiniciar o Matlab, testar se o comando "*Pso_Trelea_vectorized('f6,2)*" está funcionando);
- Supondo que o *path* do Matlab esteja na pasta raiz do simulador, executar o comando "StartSim" para iniciar o simulador;
- Para fechar o simulador, executar o comando "EndSim", também na pasta raiz do simulador;
- Observação: Fechar a janela do simulador não é suficiente para fechar o programa completamente. Algumas variáveis ainda estarão no Matlab e não será possível abrir um novo simulador enquanto o "EndSim" não for executado. Pode também travar o Matlab;

Para adicionar novos padrões para o MO:

- Os padrões dos MOs ficam na pasta "+patterns";
- Todo padrão herda a classe "Pattern", que implementa o método de rotação "calculateRot" (basta passar as coordenadas X e Y da posição antiga, as coordenadas X e Y da nova posição e o ângulo de rotação;
- Um padrão de movimento deve receber como entrada as coordenadas X e Y da posição anterior, o step (deslocamento máximo) e o ângulo de rotação (em 0, o MO andará num ângulo de 0^o com a horizontal);
- Uma fórmula deve ser aplicada na posição anterior de forma que a nova posição seja retornada, considerando o step do MO;
- Ao final, o método "patterns.Pattern.calculateRot" deve ser chamado para calcular o ângulo de rotação;
- A classe retorna as coordenadas X e Y da nova posição para o MO;

Para adicionar um novo método de Previsão:

- Os métodos de Previsão ficam na pasta "+predictors";
- Na classe control.m, é necessário adicionar o nome do método na linha 217, para que ele apareça na *gui*;
- Na classe control.m, é necessário adicionar um case no *switch* da linha 1997 com o nome do método na *gui* e o valor que será passado para o IR;
- Na classe IR.m, dentro da pasta "+robots", é necessário inserir um case no *switch* da linha 152, contendo o valor correspondente ao método escolhido (resultado do *switch* no tópico anterior) e setar a variável "ir.appliedPlanner" com o arquivo .m correspondente ao método;
- As entradas destes métodos são duas matrizes, posX e posY, contendo as coordenadas X e Y das posições coletadas dos MOs na etapa de coleta de amostras. A sintaxe de posZ, onde Z = X ou Z = Y é:

-
PosZ1MO1 PosZ1MO2 PosZ1MO3 ... PosZ1MOM
PosZ2MO1 PosZ2MO2 PosZ2MO3 ... PosZ2MOM
...
PosZMMO1 PosZMMO2 PosZMMO3 ... PosZNMOM
-

Onde PosZ1MO1 é a coordenada Z da primeira posição registrada para o MO1, PosZ1MO2 é a coordenada Z da primeira posição registrada para o MO2, PosZ2MO1 é a coordenada Z da segunda posição registrada para o MO1, etc.

- A saída destes métodos deve ser uma matriz com as posições esperadas para os MOs em tempos futuros. A sintaxe é a seguinte:

-
PosX1MO1 PosY1MO1 PosX1MO2 PosY1MO2 ... PosX1MOM PosY1MOM
PosX2MO1 PosY2MO1 PosX2MO2 PosY2MO2 ... PosX2MOM PosY2MOM
...
PosXNMO1 PosYNMO1 PosXNMO2 PosXNMO2 ... PosXNMOM PosYNMOM
-

Onde PosX1MO1 é a coordenada X da posição prevista 1 do MO1, PosY1MO1 é a coordenada Y da posição prevista 1 do MO1, PosX1MO2 é a coordenada X da posição prevista 1 do MO2, PosX2MO1 é a coordenada X da posição prevista 2 do MO1, etc.

Para adicionar um novo método de Planejamento:

- Os métodos de Planejamento ficam na pasta "+planners";
- Na classe control.m, é necessário adicionar o nome do método na linha 221, para que ele apareça na *gui*;
- Na classe control.m, é necessário adicionar um case no *switch* da linha 1997 com o nome do método na *gui* e o valor que será passado para o IR;

- Na classe IR.m, dentro da pasta "+robots", é necessário inserir um case no *switch* da linha 165, contendo o valor correspondente ao método escolhido (resultado do *switch* no tópico anterior) e setar a variável "ir.appliedPlanner" com o arquivo .m correspondente ao método;

- As entradas destes métodos são a posição inicial do IR, a posição do objetivo, uma matriz com a previsão dos MOs (saída dos métodos de previsão), o deslocamento máximo step do IR, o fRadius (Raio do Medo), e o *threshold*, que indica qual a distância entre o IR e um MO que caracteriza colisão. A matriz de previsão tem a seguinte sintaxe:

-
PosX1MO1 PosY1MO1 PosX1MO2 PosY1MO2 ... PosX1MOM PosY1MOM
PosX2MO1 PosY2MO1 PosX2MO2 PosY2MO2 ... PosX2MOM PosY2MOM
...
PosXNMO1 PosYNMO1 PosXNMO2 PosXNMO2 ... PosXNMOM PosYNMOM
-

Onde PosX1MO1 é a coordenada X da posição prevista 1 do MO1, PosY1MO1 é a coordenada Y da posição prevista 1 MO1, PosX1MO2 é a coordenada X da posição prevista 1 do MO2, PosX2MO1 é a coordenada X da posição prevista 2 do MO1, etc.

- A saída destes métodos deve ser uma matriz com as posições planejadas para o IR do ponto inicial ao objetivo, considerando sua velocidade máxima. A sintaxe é:

-
PosIRX1 PosIRY1
PosIRX2 PosIRY2
...
PosIRXN PosIRYN
-

Onde PosIRX1 é a coordenada X da primeira posição planejada para o IR, PosIRY1 é a coordenada Y da primeira posição planejada para o IR, PosIRX2 é a coordenada X da segunda posição planejada para o IR.

Estrutura de arquivos:

\Control.m -> Arquivo de controle principal do programa. Interliga a *gui* com o modelo interno.
\EndSim.m -> Este arquivo deve ser chamado para encerrar o programa.
\Heuristic.m -> Heurística utilizada pelo método de planejamento PSOSMv1.
\Heuristic4.m -> Heurística utilizada pelo método de planejamento PSOSMv2.
\Maingui.fig -> Arquivo que armazena a estrutura gráfica da *gui* (gerado com o 'guide' do MATLAB).
\Maingui.m -> Arquivo que armazena as propriedades e estruturas da *gui* (gerado com 'guide' do MATLAB).
\StartSim.m -> Este arquivo inicializa o programa.
\+patterns\Eigth.m -> Arquivo que executa o padrão 'Oito' para um MO.
\+patterns\Line.m -> Arquivo que executa o padrão 'Linha' para um MO.
\+patterns\Pattern.m -> Arquivo pai dos padrões de movimento. Armazena algumas propriedades comuns e implementa o método de rotação para um padrão.
\+patterns\Square.m -> Arquivo que executa o padrão 'Quadrado' para um MO.
\+patterns\Wave.m -> Arquivo que executa o padrão 'Onda' para um MO.
\+planners\DirectSearch.m -> Arquivo que implementa o método de planejamento 'DSM'.
\+planners\FDirectSearch.m -> Arquivo que implementa o método de planejamento 'FDSM'.
\+planners\fdsmGoal.fis -> Arquivo .fis que implementa a lógica nebulosa do controlador controlGoal para o método 'FDSM'.
\+planners\fdsmMO.fis -> Arquivo .fis que implementa a lógica nebulosa do controlador controlMO para o método 'FDSM'.
\+planners\LookAheadDirectSearch.m -> Arquivo que implementa o método de planejamento 'LADSM'.
\+planners\ParticleSwarmOptimizationSearchV1.m -> Arquivo que implementa o método

de planejamento 'PSOSMv1'.

\+planners\ParticleSwarmOptimizationSearchV2.m -> Arquivo que implementa o método de planejamento 'PSOSMv2'.

\+predictors\ANN.m -> Arquivo que implementa o método de previsão 'ANNPM'.

\+predictors\Eloform.m -> Arquivo que implementa o método de previsão 'ELOFORM'.

\+predictors\Loform.m -> Arquivo que implementa o método de previsão 'LOFORM'.

\+robots\Ir.m -> Modelo do IR. Possui as propriedades e os métodos relacionados.

\+robots\Mo.m -> Modelo do MO. Possui as propriedades e os métodos relacionados.

\+robots\Robot.m -> Modelo de um robô qualquer. Possui propriedades comuns ao IR e aos MOs.

\+tools\Lccs.m -> Retorna o tamanho do maior pareamento entre duas strings de caracteres. Utilizado no pareamento dos métodos LOFORM e ELOFORM.

\+tools\LoformAlg.m -> Implementa o algoritmo AprioriFRN, utilizado pelos métodos LOFORM e ELOFORM (Loform Alg era o nome antigo do algoritmo).

\+tools\Pos2ang.m -> Converte as amostras de posições tomadas dos MOs em ângulos de direção. Utilizado no pré-processamento do método LOFORM.

\+tools\Pos2vel.m -> Converte as amostras de posições tomadas dos MOs em vetores de velocidades. Utilizado no pré-processamento dos métodos ANNPM e ELOFORM.

\Bibliotecas\PSOt -> Biblioteca que implementa o algoritmo PSO para os métodos PSOSMv1 e PSOSMv2. Vide instalação acima.

\Gerador de Samples\gui_samples.m -> Arquivo que carrega a interface para criação de casos de simulação. O arquivo gerado deve ser usado como entrada para uma simulação automática no simulador (é um programa a parte, não faz parte do simulador).

\Gerador de Samples\gui_samples.fig -> Arquivo com a interface gráfica do criador de casos de simulação.

\Gerador de Samples\randomPatterns.m -> O gerador de casos de simulação gera apenas MOs com o padrão de movimento linha. Se a saída do programa *gui*_samples for alimentada a este programa, os padrões de movimento serão definidos aleatoriamente com distribuição normal.

\Legacy -> Métodos de planejamento que não funcionaram corretamente, ou são protótipos de outros métodos.