CENTRO UNIVERSITÁRIO FEI
AISLAN CESAR DE ALMEIDA

**FILTRO DE PARTÍCULAS APLICADO À LOCALIZAÇÃO DE ROBÔS MÓVEIS
NO DOMÍNIO DA ROBOCUP HUMANOIDE**

São Bernardo do Campo

2017

AISLAN CESAR DE ALMEIDA

**FILTRO DE PARTÍCULAS APLICADO À LOCALIZAÇÃO DE ROBÔS MÓVEIS
NO DOMÍNIO DA ROBOCUP HUMANOIDE**

Dissertação de Mestrado apresentada ao Centro Universitário FEI para obtenção do título de Mestre em Engenharia Elétrica. Orientado pelo Prof. Dr. Reinaldo Augusto da Costa Bianchi.

São Bernardo do Campo

2017

Almeida, Aislan Cesar de.

Filtro de partículas aplicado à localização de robôs móveis no domínio da RoboCup Humanoide / Aislan Cesar de Almeida. São Bernardo do Campo, 2017.

100 p. : il.

Dissertação - Centro Universitário FEI.

Orientador: Prof. Dr. Reinaldo Ausgusto da Costa Bianchi.

1. Filtro de Partículas. 2. Localização de Monte-Carlo. 3. Localização de Robô. 4. Robô Humanoide. 5. Futebol de Robôs. I. Bianchi, Reinaldo Ausgusto da Costa, orient. II. Título.

Aluno: Aislan César de Almeida

Matrícula: 115301-4

Título do Trabalho: Filtro de partículas aplicado à localização de robôs móveis no domínio da robocup humanoide.

Área de Concentração: Inteligência Artificial Aplicada à Automação

Orientador: Prof. Dr. Reinaldo Augusto da Costa Bianchi

Data da realização da defesa: 18/09/2017

ORIGINAL ASSINADA

Avaliação da Banca Examinadora:

São Bernardo do Campo, 18 / 09 / 2017.

MEMBROS DA BANCA EXAMINADORA

Prof. Dr. Reinaldo Augusto da Costa Bianchi Ass.: _____

Prof. Dr. Paulo Eduardo Santos Ass.: _____

Prof. Dr. Valdinei Freire da Silva Ass.: _____

A Banca Julgadora acima-assinada atribuiu ao aluno o seguinte resultado:

APROVADO ☒

REPROVADO ☐

VERSÃO FINAL DA DISSERTAÇÃO

**APROVO A VERSÃO FINAL DA DISSERTAÇÃO EM QUE
FORAM INCLUÍDAS AS RECOMENDAÇÕES DA BANCA
EXAMINADORA**

Aprovação do Coordenador do Programa de Pós-graduação

Prof. Dr. Carlos Eduardo Thomaz

Dedico este trabalho a Deus.

AGRADECIMENTOS

Inicialmente a Deus.

Aos amigos do laboratório que têm ajudado a sanar minhas dúvidas e têm dividido esta jornada comigo.

Aos professores que com competência têm me ensinado as artes da pesquisa científica e da inteligência artificial.

Ao Centro Universitário da FEI, por me prover uma formação de excelência.

Ao CNPq por me conceder a bolsa auxílio.

Aos meus familiares pelo pronto suporte em todas as situações.

E acima de tudo a minha noiva, por me incentivar à obter o título de mestre.

“Após eliminar tudo que for impossível, então o que restar, ainda que improvável, deverá ser a verdade.”

Sir Arthur Conan Doyle

RESUMO

Para que robôs humanoides possam jogar futebol competitivamente de maneira autônoma é necessário que os robôs conheçam suas posições no campo, essa informação é essencial para o desenvolvimento de estratégias. A posição pode ser estimada a partir do conhecimento de como o robô se move pelo domínio e por observações feitas pelo próprio robô. Mas isso não é uma tarefa trivial. Os movimentos executados pelos robôs são imprecisos, além de problemas não modeláveis que surgem por problemas da parte física do robô. As observações feitas pelo robô são ruidosas, o que impedem que informações precisas de direção e distância sejam obtidas, pois seus poucos sensores estão em constante movimento devido ao balanço necessário para manter o robô em movimento. É comum encontrar trabalhos acadêmicos sobre localização de robôs autônomos para diversos domínios, mas são poucos os trabalhos que lidam com um domínio tão restrito quanto o deste trabalho. Além disso, os trabalhos sobre este domínio apresentam algoritmos e resultados que não são reproduzíveis, devido às diferenças de *hardware* e *software* dos robôs utilizados. Assim, este trabalho implementa um sistema de localização, baseado no algoritmo de localização de Monte-Carlo, para que robôs humanoides autônomos sejam capazes de estimar suas posições no domínio. O sistema implementado apresenta um método para estimar o quanto o robô se move ao longo do tempo e métodos diferentes para calcular quanto cada partícula representa a posição do robô real, além de métodos para se recuperar de erros de estimativa, para alterar a quantidade de partículas conforme o necessário e para estimar qual a melhor observação que o robô poderá fazer em instantes futuros. Foram realizados experimentos simulados e em robôs reais que validam os métodos implementados e mostram que os métodos propostos são eficientes para resolver o problema de localização. Por fim, trabalhos futuros incluem verificar o funcionamento do sistema em situação de jogo, além da expansão do sistema para um domínio genérico para observar o funcionamento dos métodos propostos e compará-las à outros métodos do estado da arte.

Palavras-chave: Filtro de Partículas. Localização de Monte-Carlo. Localização de Robô. Robô Humanoide. Futebol de Robôs.

ABSTRACT

In order to enable humanoid robots to play soccer competitively and autonomously, the robot needs to know its own position on the field, this information is important for strategy development. Informations regarding the movement of the robot and the observation made by the robot can be used to estimate its position on the domain. However, this is not a trivial task. The movements executed by the robots are inaccurate, and there are problems that can not be modeled which emerges from the physical problems of the robot. There is noise in the observations of the robot, which prevents the acquisition of precise informations regarding distance and direction, caused by the swing needed to keep the movement of the robot. Academic works about autonomous robot localization can be found for various domains, but there are a small quantity of works about the restricted domain of this work. Furthermore, the works presenting algorithms and results for this domain are not reproducible, because of the differences in hardware and software of the robots used. Thus, this work implements a localization system, based on Monte-Carlo Localization, in order to enable autonomous humanoid robots to estimate their position on the domain. The implemented system presents a method which estimates the movement of the robot over time, and methods to compute how much a particle represents the real position of the robot, besides a method to recover from position estimation errors, a method to adapt the quantity of particles in order to represent the probability distribution and a method to estimate which observation should give the best information in a near future. Experiments in simulated and real environments were performed which validates the implemented methods, and the results show that the proposed methods can effectively solve the localization problem. Finally, future works includes verifying the operation of the system in game, besides the expansion of the system to a generic domain, in order to observe the implemented methods and compare then with state of the art methods.

Keywords: Particle Filter. Monte-Carlo Localization. Robot Localization. Humanoid Robot. Robot Soccer.

LISTA DE ILUSTRAÇÕES

Figura 1	– Imagem vista pela câmera do robô em movimento	21
Figura 2	– Rede Bayesiana com variáveis dinâmicas	25
Figura 3	– Algoritmo de Localização de Monte-Carlo	27
Figura 4	– Composição do movimento do robô	38
Figura 5	– Modelo de Movimento	41
Figura 6	– Distância d e direção φ relativas entre partícula e <i>landmark</i>	42
Figura 7	– <i>Landmark</i> distante $5.5m$ do robô	43
Figura 8	– Gráfico do valor de $W_{\delta}(z_k l_k)$ em função da distância	44
Figura 9	– Comparação entre ângulos	45
Figura 10	– Algoritmo de cálculo de peso utilizando <i>landmarks</i>	46
Figura 11	– Correlação entre um ponto P no mundo real e um ponto p na imagem . . .	47
Figura 12	– Algoritmo de cálculo de peso usando pontos-chaves	48
Figura 13	– Cálculo da distância até a borda	49
Figura 14	– Algoritmo de cálculo de peso usando distância para a borda	50
Figura 15	– Algoritmo de cálculo de peso usando a orientação do robô	51
Figura 16	– Método de Reamostragem de Partículas	52
Figura 17	– Método de reamostragem de partículas	53
Figura 18	– Distribuição de probabilidades e partículas	55
Figura 19	– Quantidade de partículas em função do desvio padrão de suas posições . .	56
Figura 20	– Método que varia quantidade de partículas	56
Figura 21	– Método que estima a melhor observação	58
Figura 22	– Algoritmo de Localização de Monte-Carlo	59
Figura 23	– Simulador RoboFEI-HT	62
Figura 24	– Caminho feito pelo robô em simulação	62
Figura 25	– Erros de orientação e posição em função do tempo do experimento de referência	64
Figura 26	– Campo de visão do robô simulado	65
Figura 27	– Erros de orientação e posição em função do tempo do experimento utilizando <i>landmarks</i>	67
Figura 28	– Posições dos pontos-chaves	68
Figura 29	– Erros de orientação e posição em função do tempo do experimento utilizando pontos-chaves	69
Figura 30	– Erros de orientação e posição em função do tempo do experimento com adaptação da quantidade de partículas	71
Figura 31	– Valores-p ao longo do tempo, obtidos pela aplicação do teste <i>T</i> de <i>Student</i> comparando o experimento de referência e o experimento que varia a quantidade de partículas	72

Figura 32 – Comparação entre os resultados obtidos com e sem a adaptação da quantidade de partículas	72
Figura 33 – Adaptação da quantidade de partículas em função do tempo	73
Figura 34 – Direções que o robô pode observar	74
Figura 35 – Erros de orientação e posição em função do tempo do experimento com o método da melhor observação	75
Figura 36 – Valores-p ao longo do tempo, obtidos pela aplicação do teste <i>T</i> de <i>Student</i> comparando o experimento de referência e o experimento que utiliza o algoritmo de melhor observação	76
Figura 37 – Comparação entre os resultados obtidos utilizando o método da melhor observação e não utilizando	76
Figura 38 – Caminho feito pelo robô após ele ser movido	77
Figura 39 – Erros de orientação e posição em função do tempo do experimento com o método de recuperação de erros de estimativa	78
Figura 40 – Valores-p ao longo do tempo, obtidos pela aplicação do teste <i>T</i> de <i>Student</i> comparando o experimento de referência e o experimento que se recupera de erros de estimativa	79
Figura 41 – Comparação entre os resultados obtidos no experimento sem recuperação de erro e no primeiro trecho do experimento com recuperação de erro	79
Figura 42 – Erros de orientação e posição em função do tempo do último experimento totalmente simulado	81
Figura 43 – Valores-p ao longo do tempo, obtidos pela aplicação do teste <i>T</i> de <i>Student</i> comparando o experimento de referência e o experimento simulado com todas as contribuições propostas	82
Figura 44 – Comparação entre os resultados obtidos no experimento de referência e no primeiro trecho do experimento com a implementação completa do MCL	82
Figura 45 – Quantidade de partículas utilizadas ao longo do tempo no último experimento totalmente simulado	83
Figura 46 – Sequência de imagens do processo de classificação dos pontos	84
Figura 47 – Posições do robô para o experimento	85
Figura 48 – Caminho percorrido pelo robô real	87

LISTA DE TABELAS

Tabela 1 – Resumo da configuração do primeiro experimento	63
Tabela 2 – Resumo de configuração do experimento utilizando <i>landmarks</i>	66
Tabela 3 – Resumo de configuração do experimento utilizando pontos-chaves	68
Tabela 4 – Resumo de configuração do experimento com adaptação da quantidade de partículas	71
Tabela 5 – Resumo de configuração do experimento do método de melhor observação .	75
Tabela 6 – Resumo de configuração do experimento de recuperação de erro de estimativa	78
Tabela 7 – Resumo de configuração do último experimento totalmente simulado	81
Tabela 8 – Resumo de configuração dos experimentos com robôs reais	86
Tabela 9 – Resultados do experimento de localização global com robô real	86
Tabela 10 – Resultados do experimento de rastreamento de posição com robô real	88

SUMÁRIO

1	INTRODUÇÃO	21
1.1	MOTIVAÇÃO	21
1.2	OBJETIVO	22
1.3	JUSTIFICATIVA	22
1.4	ORGANIZAÇÃO DO TRABALHO	23
2	TEORIA	25
2.1	DEFINIÇÃO	25
2.2	A PARTÍCULA	26
2.3	FUNCIONAMENTO	27
2.4	MODIFICAÇÕES	28
3	TRABALHOS RELACIONADOS	31
3.1	QUANTIDADE DE PARTÍCULAS	31
3.2	RECUPERAÇÃO DE ERROS DE ESTIMATIVA	32
3.3	REAMOSTRAGEM DE PARTÍCULAS	33
3.4	ROBOCUP HUMANOID SOCCER KIDSIZE LEAGUE	33
4	PROPOSTA	37
4.1	MODELO DE MOVIMENTO	37
4.2	MODELO DE OBSERVAÇÃO	41
4.2.1	Utilização de <i>Landmarks</i>	42
4.2.2	Utilização de Pontos-Chaves	46
4.2.3	Utilizando a Distância da Borda	49
4.2.4	Utilizando a Orientação do Sensor	51
4.3	MÉTODO DE REAMOSTRAGEM	51
4.4	QUANTIDADE DE PARTÍCULAS ADAPTÁVEL	54
4.5	MELHOR OBSERVAÇÃO	57
4.6	ALGORITMO FINAL	58
5	EXPERIMENTOS	61
5.1	EXPERIMENTO SIMULADO DE REFERÊNCIA	61
5.1.1	Configuração	61
5.1.2	Resultados	64
5.2	EXPERIMENTO SIMULADO UTILIZANDO <i>LANDMARKS</i>	65
5.2.1	Configuração	65
5.2.2	Resultados	67
5.3	EXPERIMENTO SIMULADO UTILIZANDO PONTOS-CHAVES	67
5.3.1	Configuração	68
5.3.2	Resultados	70

5.4	EXPERIMENTO SIMULADO ADAPTANDO A QUANTIDADE DE PARTÍCULAS	70
5.4.1	Configuração	70
5.4.2	Resultados	73
5.5	EXPERIMENTO SIMULADO ESTIMANDO A MELHOR OBSERVAÇÃO .	74
5.5.1	Configuração	74
5.5.2	Resultados	75
5.6	EXPERIMENTO SIMULADO COM RECUPERAÇÃO DE ERROS DE ESTIMATIVA	77
5.6.1	Configuração	77
5.6.2	Resultados	80
5.7	EXPERIMENTO SIMULADO COM A IMPLEMENTAÇÃO COMPLETA . .	80
5.7.1	Configuração	80
5.7.2	Resultados	82
5.8	EXPERIMENTO COM ROBÔ REAL DE LOCALIZAÇÃO GLOBAL	84
5.8.1	Configuração	84
5.8.2	Resultados	86
5.9	EXPERIMENTO COM ROBÔ REAL DE RASTREAMENTO DE POSIÇÃO	87
5.9.1	Configuração	87
5.9.2	Resultados	88
5.10	DISCUSSÃO DOS RESULTADOS	88
5.10.1	Dispersão de Partículas	88
5.10.2	Adaptação da Quantidade das Partículas	89
5.10.3	Estimativa de Melhor Observação	90
6	CONCLUSÃO	93
	REFERÊNCIAS	95

1 INTRODUÇÃO

O problema da auto-localização, ou apenas localização, tem se apresentado como um problema recorrente para a robótica móvel. Segundo Thrun, Burgard e Fox (2005), existem diversas formas de resolver este problema, mas cada domínio exige sua própria solução. Ainda assim, existem poucos trabalhos que envolvem o domínio da *RoboCup Soccer Humanoid League*, descrito por Gerndt et al. (2015), pois cada robô exige uma solução própria, devido as suas características únicas. Assim, este trabalho apresenta uma solução para o problema da localização para robôs humanoides autônomos.

1.1 MOTIVAÇÃO

O processo de localização é definido neste trabalho como o ato do robô de estimar sua própria posição no domínio. É possível inferir a posição do robô no campo a partir de observações de características discrepantes do mundo e conhecendo as ações realizadas pelo robô. Por sua vez, a posição é um conjunto de informações que indicam onde um robô se encontra no mundo, que deve ser capaz de descrever, sem ambiguidade, qualquer ponto do mundo no qual o robô está inserido.

Conforme Gerndt et al. (2015), o domínio em questão exige que os robôs possuam características humanas. Desta forma, eles devem se mover como pessoas, e seus sensores são limitados a sensores que imitem os sentidos humanos, como por exemplo câmeras e sensores inerciais. Além disso, os robôs precisam ser totalmente autônomos, ou seja, tomar decisões e coordenar estratégias sem auxílios externos. Para que as decisões e estratégias sejam eficientes, os robôs precisam conhecer suas posições no campo.

Figura 1 – Imagem vista pela câmera do robô em movimento

Fonte: Autor

Conforme o robô se move ele deve manter um ritmo de balanço, de maneira que ele consiga manter o equilíbrio enquanto anda. Esse balanço faz com que a câmera esteja em constante movimento, gerando observações ruidosas. A figura 1 apresenta uma imagem obtida pela câmera do robô em movimento. Ainda que alguma característica seja identificada, é praticamente impossível determinar qual a distância exata entre o robô e a característica observada. Além disso, as características podem ser ocluídas em situações diversas, ou ainda, o robô pode não conseguir identificar a característica por ela estar muito distante.

Além da câmera os robôs podem usar sensores inerciais e retroalimentação do posicionamento dos servo-motores como informação para reconstituir a trajetória feita pelo robô, no entanto esses sensores acumulam erros ao longo do tempo. Além disso, o movimento dos robôs é errático, pois eles estão sujeitos à quedas, folgas nas juntas, escorregamento e imprecisão no posicionamento dos servo-motores. Por fim, cada robô tem um comportamento único devido às suas características físicas.

Não é uma tarefa trivial fazer com que um robô autônomo se localize em um ambiente real. A incerteza gerada pelos movimentos imprecisos do robô e pelos sensores ruidosos impossibilitam determinar a posição exata do robô. No entanto, é possível inferir quais posições tem maior probabilidade de representar a posição atual do robô no mundo. Isso pode ser feito através da inferência em redes Bayesianas, conforme o apresentado por Russell et al. (2010).

1.2 OBJETIVO

Este trabalho apresenta um sistema de localização baseado no método probabilístico de inferência aproximada em redes Bayesianas dinâmicas conhecido como Localização de Monte-Carlo (MCL), proposto por Dellaert et al. (1999) e Fox et al. (1999). O sistema é capaz de estimar a posição do robô no mundo e se recuperar de erros de estimativa, alterando a quantidade de partículas de maneira que o equilíbrio entre a qualidade da estimativa e o tempo de execução seja mantido, além de apresentar um método que determina qual observação deve ser feita para melhorar a estimativa da posição.

1.3 JUSTIFICATIVA

Ainda que seja comum encontrar métodos baseados no MCL aplicados à diversos domínios, uma nova implementação se faz necessária, de maneira que todas as peculiaridades do domínio deste trabalho sejam consideradas. O método apresentado neste trabalho considera as falhas de movimento dos robôs, os ruídos contidos nas observações feitas pelos robôs e as limitações de hardware e software para apresentar um sistema de localização funcional para robôs humanoides autônomos.

O MCL modela os erros de movimento e observação, além de resolver os três principais problemas de localização: a localização global, o rastreamento de posição e o problema do

sequestro. A localização global é a situação na qual o robô desconhece sua posição inicial e deve estimar sua posição conforme ele se move e observa o ambiente. O rastreamento de posição é a situação na qual a posição do robô é atualizada conforme ele se move. Por fim, o problema do sequestro é quando o robô precisa recuperar-se de uma estimativa errada de posição. A posição estimada estará errada quando o robô é movido de uma posição conhecida para uma desconhecida, ou quando o robô sofre algum erro de movimento não modelado.

Os robôs utilizados para experimentos neste trabalho possuem limitações de *hardware* e *software*, que podem deixar o sistema de localização excessivamente lento. Um bom sistema de localização deve ser capaz de retornar a posição estimada do robô em tempo real, que no caso do domínio deste trabalho será considerado um período de 1 segundo, no máximo. Além disso, o sistema de localização é executado em paralelo com vários outros sistemas essenciais para o funcionamento do robô, de maneira que se o sistema de localização for computacionalmente custoso, isso pode prejudicar o funcionamento do robô como um todo.

Para implementar a localização de robôs humanoides autônomos este trabalho propõe a implementação de um algoritmo de localização baseado no MCL. O trabalho propõe a implementação de um modelo de movimento baseado em velocidade para robôs humanoides omnidirecionais, três modelos de observação e um método de reamostragem de partículas que trabalha em tempo linear em função das quantidades de partículas. Além de apresentar três contribuições para o estado da arte dos métodos de localização: a dispersão de partículas de baixa representatividade, com o objetivo de recuperar o filtro de erros de estimativa; o método que adapta a quantidade de partículas à distribuição probabilística, para economizar recursos computacionais; e um método que estima qual observação deve prover a melhor informação para o sistema de localização.

1.4 ORGANIZAÇÃO DO TRABALHO

Assim, o capítulo 2 apresenta a definição de MCL utilizada neste trabalho. Em seguida, o capítulo 3 apresenta trabalhos relacionados à este. Então, a proposta deste trabalho é apresentada no capítulo 4, seguido de sua implementação e testes no capítulo 5. As conclusões e trabalhos futuros estão no capítulo 6.

2 TEORIA

Thrun, Burgard e Fox (2005) afirmam que o MCL é um dos métodos mais populares para resolver o problema da localização de robôs autônomos móveis, o que se reflete na quantidade de material existente sobre o assunto, aplicando o método nos mais diversos domínios. Este capítulo define o MCL que será utilizado neste trabalho, que é baseado no método descrito por Thrun, Burgard e Fox (2005).

2.1 DEFINIÇÃO

O problema da localização consiste em estimar a posição do robô no domínio a partir do mapa do mundo obtido *a priori*, da ação u_k executada pelo robô e da observação z_k do ambiente. Por sua vez, a posição do robô sobre o campo é representada pelo vetor L , composto pelas coordenadas cartesianas X e Y e a orientação Θ .

No entanto, ainda que o mapa do mundo seja exato, a ação u_k e a observação z_k são ruidosas, e portanto métodos determinísticos não poderiam ser utilizados para determinar a posição a partir dos dados, tendo em vista que esses métodos devem gerar valores inconsistentes devido aos erros existentes nos dados. Assim, é necessário utilizar métodos probabilísticos para tratar esses ruídos e através dessas informações estimar a posição do robô no mundo. Este problema pode ser resolvido a partir da inferência aproximada em uma rede Bayesiana dinâmica.

Uma rede Bayesiana é um grafo acíclico no qual cada nó representa uma variável e cada nó possui uma distribuição de probabilidades condicionais. Além disso, a rede representa as relações de dependência condicionais entre os nós. De maneira semelhante, uma rede Bayesiana pode possuir variáveis aleatórias que variam em função do tempo (RUSSELL et al., 2010).

Figura 2 – Rede Bayesiana com variáveis dinâmicas

A figura 2 apresenta a representação do problema da localização como uma rede Bayesiana com variáveis dinâmicas. O filtro Bayesiano é um método utilizado para se fazer inferências de variáveis de consulta em uma rede Bayesiana com variáveis que possuem valores que variam ao longo do tempo. Ele consiste em calcular a distribuição probabilística sobre a variável de consulta dadas todas as variáveis de evidência até o instante atual.

O filtro Bayesiano é definido matematicamente como

$$bel(l_k) = \eta P(z_k|l_k) \int P(l_k|u_k, l_{k-1}) bel(l_{k-1}) dl_{k-1}, \quad (1)$$

onde $bel(l_k)$ é a representação da crença da posição l do robô no instante k , η é uma constante de normalização, a probabilidade $P(z_k|l_k)$ é chamada de modelo de observação e $P(l_k|u_k, l_{k-1})$ é o modelo de movimento, ou modelo de transição, que é aplicado sobre a crença $bel(l_{k-1})$ na posição l no instante $k - 1$.

O filtro é aplicado recursivamente sobre a crença inicial $bel(l_0)$, para obter a crença da posição no instante k . Essa equação assume a hipótese de Markov de primeiro grau, que afirma que a posição atual do robô l_k depende apenas da posição no instante anterior l_{k-1} e ação executada pelo robô u_k entre os instantes $k - 1$ e k , e que a observação z_k depende apenas da posição atual do robô l_k (RUSSELL et al., 2010).

2.2 A PARTÍCULA

O MCL é um filtro Bayesiano que resolve o problema de localização utilizando n partículas para representar a crença da posição do robô. A partícula representa uma possível posição do robô no mundo, onde cada partícula possui coordenadas cartesianas X e Y , orientação Θ e peso w , onde o peso é proporcional à probabilidade da partícula representar a posição do robô no domínio. Cada partícula pode ser considerada uma simulação do robô real, de maneira que movimentos e observações feitas pelo robô sejam simuladas em um mundo virtual, considerando a posição de cada partícula. Por fim, o peso de cada partícula é proporcional ao quão bem ela representa o robô real neste instante.

Os movimentos do robô real são imprecisos, e esta imprecisão é considerada pelas partículas como erros que alteram seus padrões de movimento ao longo do tempo. Esses erros são modelados de maneira que o movimento executado em média pelo robô seja o mesmo executado pela partícula, adicionado de um erro Gaussiano. O capítulo 4.1 apresenta o modelo de movimento utilizado pela partícula.

A observação do robô é utilizada para calcular o peso de cada partícula. O peso representa o quão semelhantes são a observação feita pelo robô real e a observação que deveria ser feita se o robô estivesse na posição dada pela partícula. Desta forma espera-se que uma partícula que representa a posição do robô receba um peso alto, enquanto uma partícula distante da posição do robô receba peso nulo. No capítulo 4.2 é apresentado o modelo de observação utilizado neste trabalho.

2.3 FUNCIONAMENTO

O MCL é iniciado com n partículas distribuídas uniformemente sobre todo o domínio. Então, a cada iteração do filtro, uma sequência de algoritmos auxiliares é executada, conforme apresentado no algoritmo da figura 3. Como entrada o algoritmo recebe o conjunto de partículas da iteração anterior H_{k-1} , a ação executada pelo robô u_k e a observação z_k . Cria-se um conjunto de partículas auxiliar vazio, em seguida para cada partícula são aplicados o modelo de movimento e o modelo de observação, então uma cópia da partícula na nova posição com o novo peso é armazenado no conjunto auxiliar. Por fim, as partículas são selecionadas em função de seus pesos, de maneira que partículas com pesos maiores terão mais cópias na próxima iteração do filtro que as demais.

O modelo de movimento tem a função de prever qual a posição da distribuição de probabilidades dado que o robô tenha executado um dado movimento. Ele pode ser feito através do conhecimento *a priori* do quanto o robô se move em função do tempo, por exemplo a velocidade e aceleração do robô, ou ainda pode-se utilizar um modelo de odometria, que calcula o deslocamento do robô a partir de informações obtidas pelos sensores do robô. Os possíveis erros aleatórios contidos nos movimentos do robô devem ser considerados por este modelo. Em um filtro de partículas, o modelo de movimento é aplicado à cada partícula, fazendo com que elas se espalhem, de maneira que em média o conjunto execute o movimento que o robô teria feito caso seus movimentos fossem livres de erros (THRUN; BURGARD; FOX, 2005).

A partir de observações feitas do mundo obtêm-se informações do estado atual do domínio. O modelo de observação consiste em utilizar essas informações para calcular qual a distribuição de probabilidades *a posteriori* da posição atual do robô. Em um filtro de partículas, para cada partícula calcula-se a probabilidade dela representar a posição do robô no mundo. Esse valor é calculado através da similaridade entre a observação feita pelo robô e a observação que deveria ter sido feita pelo robô caso o robô estivesse na posição dada pela partícula. Então as partículas são selecionadas e copiadas em função da probabilidade que cada partícula têm de representar a posição do robô, de maneira que partículas com maiores probabilidades de

Figura 3 – Algoritmo de Localização de Monte-Carlo

Localização de Monte-Carlo (H_{k-1}, u_k, z_k)

1. Cria um conjunto H'_k vazio
2. **para cada** elemento $i \in H_{k-1}$ **faça**
3. $l \leftarrow$ Modelo de Movimento (H_{k-1}^i, u_k)
4. $w \leftarrow$ Modelo de Observação (H_{k-1}^i, z_k)
5. $H'_k \leftarrow (l, w)$
6. **fim**
7. $H_k \leftarrow$ Reamostragem (H'_k, n)
8. **retorna** H_k

Fonte: Autor

representar a posição do robô são selecionadas e copiadas mais vezes para fazer parte da distribuição probabilística *a posteriori* do filtro. Isso é chamado de atualização, que consiste em calcular a distribuição de probabilidades *a posteriori* a partir da predição feita da distribuição de probabilidades *a priori*.

Durante as primeiras iterações do filtro, as partículas estão espalhadas por todo o domínio, nessa situação o filtro está realizando a localização global. Após algumas iterações, as partículas ficam aglomeradas em torno de um mesmo ponto, neste instante diz-se que o filtro convergiu. Daí em diante, conforme o robô se move, as partículas acompanham o movimento do robô, o que caracteriza o rastreamento de posição.

Eventualmente o robô é movido para outra posição, ou sofre um erro de movimento que não foi modelado, fazendo com que a posição estimada pelo sistema esteja errada. Esta situação é conhecida como o problema do sequestro, segundo Thrun, Burgard e Fox (2005). O algoritmo da figura 3 não é capaz de resolver este problema, no entanto ele pode ser modificado para resolver este e outros problemas.

2.4 MODIFICAÇÕES

Algumas modificações ao método original foram propostas, com o objetivo de solucionar o problema do sequestro, aumentar a robustez e melhorar o desempenho do MCL. Esta seção apresenta as modificações mais tradicionais do MCL.

Para resolver o problema do sequestro Thrun, Burgard e Fox (2005) argumentam que é possível adicionar partículas aleatórias a cada iteração. No caso do filtro estar rastreando a posição do robô, essas novas partículas são descartadas naturalmente durante a iteração do filtro. Se o robô estiver sofrendo o problema do sequestro, algumas dessas partículas podem ser colocadas em regiões que representam melhor a posição do robô no mundo real, fazendo com que a estimativa da posição volte a se tornar correta.

Lenser e Veloso (2000) apresentam um algoritmo que determina qual a posição em que o robô deveria estar dada a observação feita naquele instante. Esta posição é utilizada como heurística para espalhar novas partículas no domínio. No entanto, este método depende do tipo de observação feita. Em algumas situações pode ser muito custoso encontrar qual posição corresponde àquela observação, tornando o método ineficiente.

O problema do sequestro é eventual, por isso adicionar partículas aleatórias em todas as iterações é uma maneira ineficiente de lidar com o problema. Thrun, Burgard e Fox (2005) apresentam o algoritmo conhecido como *Augmented MCL* (MCL Melhorado), no qual novas partículas são adicionadas quando necessário. Para isso utiliza-se uma comparação entre médias móveis do peso médio das partículas à cada iteração, uma média de curto prazo, que é sensível às rápidas variações do peso médio, e uma de longo prazo, que é pouco afetada por essas variações. O peso médio é calculado a partir da média dos pesos do conjunto de partículas. Quando a média de curto prazo se torna menor que a de longo prazo, significa que houve uma

queda na qualidade da estimativa e, portanto, deve-se adicionar novas partículas para tentar se recuperar do erro. Caso contrário ele funciona normalmente. Essa comparação das médias serve para evitar que partículas aleatórias sejam geradas em situações em que as observações feitas são muito ruidosas.

No entanto, esses métodos podem ter efeitos indesejados no sistema. Um deles ocorre quando o filtro é muito sensível ao problema do sequestro e as observações são muito ruidosas. Quando o robô está sofrendo do problema do sequestro as observações feitas pelo robô não correspondem às observações esperadas pelo conjunto de partículas, fazendo com que o conjunto de partículas receba pesos muito baixos, que é a mesma situação que ocorre quando as observações são muito ruidosas. Se esta situação for recorrente, o filtro vai estar tentando se recuperar continuamente, se comportando como se estivesse resolvendo o problema do sequestro, o que implica em um filtro que não converge. A outra situação acontece quando o filtro não é sensível o bastante, o filtro de partículas precisa fazer diversas iterações até que ele reconheça a situação de sequestro, o que pode demorar muito dependendo do domínio. Isso implica em erros de posição que podem ser prejudiciais ao funcionamento do robô. Por isso, o método precisa estar bem calibrado de acordo com o domínio.

Outra alteração comum é a adição de algum algoritmo que altere a quantidade de partículas de acordo com a necessidade. O algoritmo conhecido como *Adaptive MCL* (MCL adaptativo), apresentado por Fox (2003), altera a quantidade de partículas do filtro de acordo com a entropia da informação contida nas posições das partículas. Se as partículas estão muito espalhadas, a entropia da informação é alta, e assim a quantidade de partículas é diretamente proporcional à este valor de entropia. Segundo Thrun, Burgard e Fox (2005) a ideia principal do algoritmo é que a cada iteração do filtro, determinar o número de amostras que, com probabilidade $1 - \delta$, faça o erro entre a verdadeira *posteriori* e a aproximação baseada na amostragem ser menor que ϵ . A partir disso é possível determinar quantas partículas podem ser usadas para descrever satisfatoriamente a crença do filtro.

Alterar a quantidade de partículas de acordo com a necessidade é importante, pois o tempo de execução de cada iteração do filtro de partículas é diretamente proporcional à quantidade de partículas. Assim, se houver uma quantidade muito grande de partículas, o custo computacional para executar cada iteração do filtro será alto. No entanto, reduzir a quantidade de partículas reduz a qualidade da estimativa feita pelo filtro de partículas, pois a qualidade da estimativa feita pelo filtro também é diretamente proporcional à quantidade de partículas. Dessa forma, o desejável é possuir uma grande quantidade de partículas de maneira que elas possam representar bem a distribuição de probabilidades, que implica em uma maior qualidade na estimativa feita. Então deve-se utilizar uma quantidade de partículas, tal que haja o equilíbrio entre o custo computacional e a qualidade da estimativa. Para tanto deve-se considerar que quando a incerteza da distribuição de probabilidades é baixa, uma pequena quantidade de partículas é suficiente para estimá-la, de maneira semelhante, quando a incerteza é alta são necessárias muitas

partículas para conseguir estimar a posição. Levando isso em consideração, pode-se traçar uma relação entre a incerteza do sistema e a quantidade de partículas.

O próximo capítulo apresenta outras modificações possíveis que melhoram o desempenho do MCL. Além de outras propostas relacionadas à este trabalho.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta trabalhos que utilizam o MCL, algoritmos que auxiliam o processo de localização e trabalhos desenvolvidos para domínios semelhantes ao deste trabalho.

3.1 QUANTIDADE DE PARTÍCULAS

Filtros de partículas têm tempo de execução proporcional à quantidade de partículas utilizadas. Portanto, quanto menos partículas mais rápida será a execução do filtro. No entanto, a qualidade da posição estimada pelo filtro é diretamente proporcional à quantidade de partículas (THRUN; BURGARD; FOX, 2005). Quanto mais partículas forem utilizadas para representar a distribuição de probabilidades, maior será a qualidade da estimativa feita pelo filtro. Desta forma, deve-se considerar utilizar uma quantidade de partículas que forneça o equilíbrio entre velocidade de execução e qualidade da posição estimada do robô.

Visando o melhor desempenho do MCL, Fox (2003) apresentou um método baseado em entropia da informação, onde a quantidade de partículas é proporcional à entropia do sistema, que já foi apresentada no capítulo 2.4. Para estimar a quantidade de partículas, o método proposto por Fox, divide o domínio em regiões, então verifica quantas dessas regiões possuem partículas, então utiliza-se a distância de Kullback-Leibler para determinar a quantidade de partículas que representam a dada distribuição de partículas.

Outro método, apresentado por Hong, Zhou e Tian (2009), utiliza as características da distribuição de partículas para alterar a quantidade de partículas que serão utilizadas para a próxima iteração do filtro. Entre essas características estão o quanto essas partículas estão aglomeradas e o quão bem elas representam a posição do robô. A primeira é a variável *focus*, ela recebe o valor de *Yes* ou *No* de acordo com a área que as partículas ocupam no mundo; se as partículas estão contidas em uma pequena área, a variável *focus* será *Yes*. Já a segunda, chamada *near*, pode assumir três valores -1 , 0 e $+1$. Sendo que a variável é 0 quando o filtro é iniciado, -1 se a observação esperada pela partícula com o maior peso for muito diferente da observação feita pelo robô e $+1$ se a observação feita pelo robô é semelhante a observação esperada pela partícula com maior peso. Então, o estado da distribuição de probabilidades pode ser identificada como uma das quatro classes *messy*, *approach*, *cluster* e *error*. O estado *messy* representa o estado inicial do filtro, neste estado utiliza-se uma grande quantidade de partículas para representar a distribuição de probabilidades. Quando a variável *near* se torna $+1$, o estado do sistema é alterado para *approach*. Então, quando a variável *focus* se torna *Yes*, o estado do sistema é alterado para *cluster*, e nesse estado a quantidade de partículas é alterada para uma quantidade pequena, isso significa que o filtro convergiu. Finalmente, eventualmente a variável *near* pode se tornar -1 enquanto a variável *focus* ainda for *Yes*, então o estado do filtro será alterado para *error*, nesse estado a quantidade de partículas é alterada para um valor grande novamente, este estado representa o problema do sequestro.

O trabalho de Soto (2005) utiliza o momento da função de distribuição de probabilidades para determinar a quantidade de partículas que melhor representa a distribuição de probabilidades. Ele afirma que é possível testar a precisão que um filtro de partículas tem de estimar o momento de uma densidade ao invés de testar a precisão da estimativa da distribuição de probabilidades *a posteriori*.

Straka e Šimandl (2011) propõem uma generalização da técnica chamada de SSA. O SSA adapta a quantidade de partículas de acordo com a qualidade da função de distribuição de probabilidades. O método é generalizado de maneira que possa permitir a adaptação da quantidade de partículas em função da qualidade de uma estimativa arbitrária. A intensão é adaptar a quantidade de partículas mantendo um nível de erro máximo predeterminado entre a estimativa aproximada e a estimativa real.

Desta forma, este trabalho apresenta um método que altera a quantidade de partículas. O método utiliza a dispersão das partículas para determinar quantas partículas devem ser utilizadas no MCL, sendo esta dispersão proporcional ao desvio padrão das posições das partículas. O capítulo 4.4 apresenta o método em detalhes.

3.2 RECUPERAÇÃO DE ERROS DE ESTIMATIVA

Um problema que deve ser considerado no domínio de robôs móveis é o problema do sequestro. Conforme o que já foi apresentado no capítulo 2.4, a maneira mais simples de resolver este problema é adicionar partículas aleatoriamente antes ou depois de cada iteração do filtro de partículas (THRUN; BURGARD; FOX, 2005), no entanto este método é falho, pois a distribuição de partículas *a posteriori* estará errada, além do desperdício de processamento. Assim, os trabalhos apresentados por Lenser e Veloso (2000) e Zhang, Zapata e Lépinay (2009), apresentam métodos que adicionam partículas em função de uma heurística. O primeiro método já foi comentado no capítulo 2.4, no entanto este método foi melhorado por Coltin e Veloso (2013) ao utilizar múltiplas observações para adicionar partículas nas regiões com maior probabilidade de representar a posição do robô, assim evitando que observações ruidosas ou ambíguas gerem partículas em regiões indesejadas.

Já o trabalho de Zhang, Zapata e Lépinay (2009) apresenta um método que determina quais as regiões do domínio têm maior probabilidade de representar a posição do robô naquele instante, então novas partículas são adicionadas nessas regiões quando necessário. O método funciona em três etapas. Na primeira, que deve ser feita previamente, é gerado o mapa do mundo no qual o robô deve se mover, e ao mesmo tempo é gerado um mapa de energia, onde a energia denota alguma informação especial observada. A segunda etapa, que é feita durante as iterações do filtro de partículas, consiste em calcular um subconjunto do mapa de energia, ele representa as prováveis posições do robô dada a observação das informações especiais previamente mapeadas. A última etapa é a localização, que utiliza o mapa do mundo para se localizar e o subconjunto do mapa de energia para gerar novas partículas aleatoriamente.

Thrun, Burgard e Fox (2005) utilizam duas médias móveis para determinar se um filtro de partículas convergiu em uma posição errada, essas médias representam a evolução da soma-tória dos pesos calculados das partículas ao longo do tempo. Se o peso cai muito com relação às iterações anteriores novas partículas são geradas, sendo que a quantidade de partículas geradas é proporcional ao quão ruim está a estimativa.

O problema da localização apresenta uma distribuição contínua de probabilidades sobre o domínio e a única forma de representar tal distribuição seria utilizando infinitas partículas, o que é inviável. Um erro de estimativa é característico da situação em que as partículas passam a representar um ponto de máximo local ao invés do máximo global. Os erros de estimativa podem ser causados por sequências de observações ruins ou por erros de movimentos não modelados. Os trabalhos demonstram que a inserção de novas partículas é necessário para que o MCL se recupere de erros de estimativa. Assim, este trabalho apresenta um método que ajude o MCL a se recuperar de erros na posição estimada no capítulo 4.1. A proposta deste trabalho é utilizar o erro da partícula, que por sua vez se reflete no seu peso, para dispersar partículas com pesos menores, de maneira que essas partículas eventualmente encontrem uma região que represente um ponto de máximo global da função de distribuição de probabilidades.

3.3 REAMOSTRAGEM DE PARTÍCULAS

O método de reamostragem apresentado neste trabalho foi inspirado pelos métodos apresentados por Douc e Cappé (2005), eles apresentam diversos métodos para amostrar partículas, descrevendo os métodos e suas características. Conforme o apresentado por Li, Bolic e Djuric (2015), a fase de reamostragem é muito importante em um filtro de partículas, sendo responsável pela qualidade da distribuição *a posteriori* de partículas. Devido a sua importância, métodos para reamostragem em filtro de partículas foram estudados por Liu, Chen e Logvinenko (2001), Bolić, Djurić e Hong (2004) e Hol, Schon e Gustafsson (2006).

Levando em consideração os textos citados, este trabalho propõe um método que atende aos seguintes requisitos: de que se todas as partículas tiverem o mesmo peso, todas devem ser selecionadas, de que qualquer partícula com o peso acima da média deve ser selecionada ao menos uma vez e de que o tempo de execução do algoritmo seja linearmente proporcional à quantidade de partículas.

3.4 ROBOCUP HUMANOID SOCCER KIDSIZE LEAGUE

O domínio considerado neste trabalho é excelente para testar métodos de localização, pois o ambiente dinâmico apresenta vários desafios para o processo de localização. Os robôs sofrem com diversos problemas, entre eles, quando um robô esbarra no outro a direção do seu movimento é alterada sem o conhecimento do robô, o robô pode sofrer quedas que o desorienta, ou ainda, em alguns momentos o robô não poderá observar completamente o mundo, por ele

estar sofrendo algum tipo de oclusão ou pelo fato do robô não conseguir fazer observações de objetos e características distantes. Os robôs são limitados computacionalmente e devem responder em tempo real, enquanto executam diversos processos responsáveis por funções essenciais para o funcionamento do robô.

Assim, parte da pesquisa foi feita com relação ao que já é utilizado por outras equipes que participam da *RoboCup* nas categorias *Humanoid Soccer League* e a *Standard Platform League*. Toda a documentação referente as equipes podem ser encontradas no site oficial da liga¹. Em ambas as ligas os robôs são humanoides autônomos, que utilizam como sensor principal câmeras e devem se mover com o auxílio de pernas. A maioria das equipes usam alguma variação de filtro de partículas para se localizar (RAMEZANI et al., 2017; HAYASHIBARA et al., 2015; JUÁREZ et al., 2015; YANIV et al., 2016; GAO et al., 2017; XINGJIE et al., 2016; ANURAG et al., 2017; ALLGEUER; MISSURA; BEHNKE, 2015; AMIRI; NADIRI, 2017; TASCH et al., 2015; HOSSEINI et al., 2016; MELLMANN et al., 2014; MORALES et al., 2012; AKIN et al., 2014; KOK et al., 2013), enquanto outras equipes usam variações do Filtro de *Kalman* (DRAEGERT et al., 2017; AMOS et al., 2017; KARGAS et al., 2013; ASHAR et al., 2014) e ainda existem algumas equipes que combinam as técnicas (DIJK et al., 2015). Note que os trabalhos citados são *Technical Description Papers* (TDP), onde são apresentadas pequenas descrições do funcionamento dos robôs utilizados na competição, nestes trabalhos existem apenas citações do método utilizado, sem uma descrição profunda do método em si.

No entanto algumas implementações se destacam: a primeira é utilizar um filtro de partículas inicialmente, enquanto o robô ainda não está localizado, então quando as partículas convergem utiliza-se um filtro de *Kalman* para rastrear a posição do robô (TEIMOURI et al., 2017; TEIMOURI; SALEHI; MEYBODI, 2016; IMRAN et al., 2015). Este sistema leva em consideração que o filtro de partículas consegue se sair melhor para localizar um robô inicialmente perdido, se aproveitando da propriedade do filtro de manter múltiplas hipóteses, então quando o filtro de partículas converge, o sistema é transformado em um filtro de *Kalman*, que é um filtro Bayesiano com a restrição de que todas as variáveis são representadas por Gaussianas e é de rápida execução.

Outras implementações utilizam a proposta de Lenser e Veloso (2000), de adicionar partículas de acordo com as observações feitas do ambiente (HERCULANO et al., 2017; RÖFFER et al., 2016). Alguns utilizam alguma forma de rastreamento de objetos que se movem no campo, com filtro de partículas, como no caso de Allali et al. (2017), ou filtro de *Kalman*, conforme o utilizado por Huan et al. (2016), desta forma é possível ter o conhecimento da disposição dos objetos no mundo. Por fim, Aly et al. (2017) aplicam um método para mapear o entorno do campo e utilizam essa informação para se localizar posteriormente.

O trabalho de Nagi, Adiprawita e Mutijarsa (2014) apresenta uma implementação completa do MCL, que usava os postes amarelos dos gols como *landmarks*. Primeiro eles encontram os pontos amarelos na imagem que estão sobre a linha imaginária do horizonte. A partir desses

¹<https://www.robocuphumanoid.org/>

pontos, o método encontra as bases das traves, conhecendo-se a posição da câmera com relação ao solo, calcula-se a distância das bases das traves ao robô.

Tian et al. (2013) apresentam um método intuitivo para verificar se a estimativa do filtro de partículas corresponde à posição real do robô. Após executar os passos convencionais do filtro de partículas, a quantidade de partículas com pesos baixos é contada. Se menos de 20% das partículas tiverem pesos baixos, as partículas com pesos baixos são descartadas e são feitas cópias aleatórias das partículas com pesos altos, até que a quantidade de cópias seja igual a quantidade de partículas descartadas, este caso considera que o filtro convergiu. Se mais que 80% das partículas tiverem peso baixo, essas partículas com peso baixo são descartadas e novas partículas são geradas aleatoriamente, este caso considera a situação de sequestro. Se a quantidade estiver entre 20% e 80% o filtro continua seu funcionamento convencional.

Perico, Santos e Bianchi (2015) propõem o uso de informações qualitativas para calcular os pesos das partículas. As informações obtidas pelo robô do mundo correspondem à regiões qualitativas do espaço, que representam as relações entre o robô e os *landmarks* observados. As regiões qualitativas se baseiam na distância e direção com relação ao robô, onde a distância pode assumir os valores *at*, *near*, *far* e *further*, enquanto a direção pode assumir os valores de *left*, *left-front*, *front*, *right-front* e *right*. Então, o peso da partícula é calculado comparando as distâncias e direções observadas pelo robô com as esperadas pela partícula. Para cada informação correta o peso é multiplicado por um valor tendendo à 1, no outro caso é multiplicado o complemento do valor.

Recentemente Chang, Wang e Wang (2016) utilizaram o filtro de *Kalman* para localização de robôs autônomos móveis em ambientes dinâmicos. Eles utilizam objetos em movimento como *landmarks*. Baseado nisso eles apresentam o problema da localização de múltiplos robôs em ambientes dinâmicos como um problema de localização e mapeamento simultâneos. Segundo eles, a maioria dos métodos consideram esses objetos como erros de observação, ou apenas rastreiam tais objetos. No entanto, o trabalho implementado por Chang, Wang e Wang (2016) utiliza os objetos em movimento para melhorar a capacidade do algoritmo de localização para um time de robôs. Ao compartilharem as informações sobre esses objetos, os robôs terão mais informações como referência para realizar a localização.

O próximo capítulo apresenta a proposta para resolver o problema da localização para robôs humanoides autônomos.

4 PROPOSTA

Este capítulo apresenta os métodos e algoritmos utilizados para implementar o sistema de localização para um robô humanoide autônomo. O capítulo está dividido em seis seções. Na seção 4.1 é apresentado o modelo de movimento, em seguida é apresentado o modelo de observação e suas variações na seção 4.2. O método utilizado para reamostrar as partículas de acordo com seus pesos é apresentado na seção 4.3. A seção 4.4 apresenta o método utilizado para determinar qual a quantidade de partículas é necessária para representar a distribuição de probabilidades. Um método que estima qual a observação oferece a melhor informação para aumentar a qualidade da estimativa da posição do robô é apresentado na seção 4.5. Finalmente, o algoritmo completo utilizado pelo sistema de localização é apresentado na seção 4.6.

4.1 MODELO DE MOVIMENTO

O modelo de movimento é a representação de como uma ação altera a distribuição de probabilidades. A partir dele pode-se estimar como o estado evolui em função do tempo e da ação executada. Este modelo deve calcular a probabilidade *a posteriori* $P(l_k | u_k, l_{k-1})$, onde l_k representa a posição atual do robô, u_k é o conjunto de ações executadas pelo robô desde a última atualização e l_{k-1} é a posição anterior do robô. O modelo de movimento também pode ser chamado de modelo de transição ou predição.

Os requisitos para o modelo de movimento são: que ele seja capaz de prever o deslocamento dado uma variação de tempo Δt e ser capaz de modelar os erros inerentes aos movimentos dos robôs. Para tanto, pode-se modelar o movimento do robô a partir da velocidade de seus movimentos.

Para o domínio deste trabalho, considera-se que o robô tem três velocidades: velocidade angular v_k^θ , velocidade escalar frontal v_k^f e velocidade escalar lateral v_k^s . Ou seja, o robô pode girar em torno do próprio eixo, andar para frente, para trás, para a esquerda, para a direita, ou qualquer combinação desses movimentos. Assim, têm-se que o comando de movimento é dado por

$$u_k = \begin{pmatrix} v_k^f \\ v_k^s \\ v_k^\theta \end{pmatrix}. \quad (2)$$

A partir destas velocidades pode-se gerar as equações que regem os movimentos do robô, desta forma

$$\begin{aligned} x_k &= x_{k-1} + \int_{t_i}^{t_j} v_k^x(v_k^f, v_k^s, \theta_k, \Delta t) dt, \\ y_k &= y_{k-1} + \int_{t_i}^{t_j} v_k^y(v_k^f, v_k^s, \theta_k, \Delta t) dt, \\ \theta_k &= \theta_{k-1} + v_k^\theta \Delta t, \end{aligned} \quad (3)$$

Figura 4 – Composição do movimento do robô

Fonte: Autor

notando que não se pode adicionar u_k diretamente à posição anterior l_{k-1} , pois é necessário converter as velocidades escalares relativas à orientação do robô para velocidades vetoriais referentes ao plano de movimento.

A figura 4 apresenta graficamente as relações trigonométricas utilizadas para determinar as funções $v_k^x(v_k^f, v_k^s, \theta_k, \Delta t)$ e $v_k^y(v_k^f, v_k^s, \theta_k, \Delta t)$, as quais serão representadas resumidamente como v_k^x e v_k^y respectivamente. O robô é representado por um círculo, onde a reta que vai do centro à borda do círculo representa a orientação do robô no mundo. Da figura 4 tem-se que

$$v_k^x = v_k^f \cos(\theta_k) + v_k^s \sin(\theta_k), \quad (4)$$

onde

$$\theta_k = \theta_{k-1} + v_k^\theta \Delta t. \quad (5)$$

Então, substituindo a equação (5) na equação (4) obtém-se:

$$v_k^x = v_k^f \cos(\theta_{k-1} + v_k^\theta \Delta t) + v_k^s \sin(\theta_{k-1} + v_k^\theta \Delta t). \quad (6)$$

A equação (6) apresenta a variação da velocidade x em função da orientação θ e das velocidades frontal v_k^f e lateral v_k^s . Agora é necessário integrar esta função no tempo de maneira que seja possível obter a variação da posição em função da orientação θ , das velocidades v_k^f e v_k^s e do tempo desde a última iteração k . Então

$$\begin{aligned} \int_{t_i}^{t_j} v_k^x dt &= \int_{t_i}^{t_j} v_k^f \cos(\theta_{k-1} + v_k^\theta \Delta t) dt \\ &\quad + \int_{t_i}^{t_j} v_k^s \sin(\theta_{k-1} + v_k^\theta \Delta t) dt, \\ \int_{t_i}^{t_j} v_k^x dt &= \frac{v_k^f}{v_k^\theta} [\sin(\theta_{k-1} + v_k^\theta t_j) - \sin(\theta_{k-1} + v_k^\theta t_i)] \\ &\quad + \frac{v_k^s}{v_k^\theta} [\cos(\theta_{k-1} + v_k^\theta t_i) - \cos(\theta_{k-1} + v_k^\theta t_j)]. \end{aligned} \quad (7)$$

Considerando que $t_i = 0$ e que $t_j = \Delta t$, tem-se que a equação (7) se torna:

$$\int_{t_i}^{t_j} v_k^x dt = \frac{v_k^f}{v_k^\theta} [\sin(\theta_{k-1} + v_k^\theta \Delta t) - \sin(\theta_{k-1})] + \frac{v_k^s}{v_k^\theta} [\cos(\theta_{k-1}) - \cos(\theta_{k-1} + v_k^\theta \Delta t)]. \quad (8)$$

No caso especial onde $v_k^\theta = 0$ tem-se que

$$\begin{aligned} \int_{t_i}^{t_j} v_k^x dt &= \int_{t_i}^{t_j} v_k^f \cos(\theta_{k-1}) dt + \int_{t_i}^{t_j} v_k^s \sin(\theta_{k-1}) dt, \\ &= v_k^f \cos(\theta_{k-1}) (t_j - t_i) + v_k^s \sin(\theta_{k-1}) (t_j - t_i), \\ &= v_k^f \cos(\theta_{k-1}) \Delta t + v_k^s \sin(\theta_{k-1}) \Delta t. \end{aligned} \quad (9)$$

Seguindo o mesmo raciocínio para v_k^y , tem-se

$$\int_{t_i}^{t_j} v_k^y dt = \frac{v_k^f}{v_k^\theta} [\cos(\theta_{k-1}) - \cos(\theta_{k-1} + v_k^\theta \Delta t)] + \frac{v_k^s}{v_k^\theta} [\sin(\theta_{k-1}) - \sin(\theta_{k-1} + v_k^\theta \Delta t)] \quad (10)$$

e quando $v_k^\theta = 0$ tem-se que

$$\int_{t_i}^{t_j} v_k^y dt = v_k^f \sin(\theta_{k-1}) \Delta t - v_k^s \cos(\theta_{k-1}) \Delta t. \quad (11)$$

Por fim, substituindo as equações (7) e (10) no conjunto de equações (3) obtém-se o conjunto de equações

$$\begin{aligned} x_k &= x_{k-1} + \frac{v_k^f}{v_k^\theta} [\sin(\theta_{k-1} + v_k^\theta \Delta t) - \sin(\theta_{k-1})] + \frac{v_k^s}{v_k^\theta} [\cos(\theta_{k-1}) - \cos(\theta_{k-1} + v_k^\theta \Delta t)], \\ y_k &= y_{k-1} + \frac{v_k^f}{v_k^\theta} [\cos(\theta_{k-1}) - \cos(\theta_{k-1} + v_k^\theta \Delta t)] + \frac{v_k^s}{v_k^\theta} [\sin(\theta_{k-1}) - \sin(\theta_{k-1} + v_k^\theta \Delta t)], \\ \theta_k &= \theta_{k-1} + v_k^\theta \Delta t. \end{aligned} \quad (12)$$

No entanto, o conjunto de equações (12) representa um domínio determinístico, para que estas equações sejam úteis em um domínio estocástico deve-se inserir erros no cálculo. Esses erros são gerados a partir de distribuições normais $\mathcal{N}(\sigma)$ com média zero, onde σ é o desvio padrão, que serão aplicados sobre os valores de velocidades dados pelo vetor u_k .

Além disso deve-se considerar um erro não modelado que sempre afeta o movimento da partícula. Esse erro representa a perda da confiança no posicionamento do robô. Este erro é justificado pois enquanto as partículas não se movem e novas observações são feitas, o processo de reamostragem irá selecionar as melhores partículas e fazer cópias delas, após algumas iterações todas as partículas serão idênticas, por mais que a partícula seja uma boa representação da posição do robô no mundo ela pode não ser correta. Isso implica que a confiança que o filtro têm naquela posição é máxima, mas a posição não reflete a posição real do robô.

Além dos erros que as velocidades geram entre si, este trabalho propõe utilizar o peso das partículas para gerar erros de movimento, com o objetivo de espalhar partículas com pesos baixos para solucionar o problema do sequestro. A justificativa para utilizar este método é que: uma partícula que tenha recebido peso baixo não representa a posição do robô, e se ela for selecionada durante o processo de reamostragem na próxima iteração, ela deverá se mover mais que as demais partículas, de maneira que ela tenha chance de cair em uma região na qual ela receba um peso maior, ou ela é descartada no processo de reamostragem. Quando o robô está sofrendo o problema do sequestro todas as partículas devem receber pesos baixos, desta forma todas as partículas se espalham pelo domínio, e algumas dessas partículas deverão cair em regiões que representam melhor a posição do robô no mundo.

Considerando todos estes erros, as velocidades adicionadas de erros são calculadas por

$$\begin{aligned}\hat{v}_k^f &= v_k^f + \mathcal{N}(\sigma_1 v_k^f) + \mathcal{N}(\sigma_2 v_k^s) + \mathcal{N}(\sigma_3 v_k^\theta) + \mathcal{N}(\sigma_4 w_\epsilon) + \mathcal{N}(\sigma_5), \\ \hat{v}_k^s &= v_k^s + \mathcal{N}(\sigma_6 v_k^f) + \mathcal{N}(\sigma_7 v_k^s) + \mathcal{N}(\sigma_8 v_k^\theta) + \mathcal{N}(\sigma_9 w_\epsilon) + \mathcal{N}(\sigma_{10}), \\ \hat{v}_k^\theta &= v_k^\theta + \mathcal{N}(\sigma_{11} v_k^f) + \mathcal{N}(\sigma_{12} v_k^s) + \mathcal{N}(\sigma_{13} v_k^\theta) + \mathcal{N}(\sigma_{14} w_\epsilon) + \mathcal{N}(\sigma_{15}),\end{aligned}\tag{13}$$

onde σ_i são fatores que representam o quanto uma velocidade influencia a outra.

O fator w_ϵ é calculado após o peso da partícula ser determinado. Através das equações utilizadas para se obter o peso w de uma partícula, é possível determinar o peso máximo w_{MAX} que seria atribuído à partícula que representasse perfeitamente a observação feita pelo robô naquele instante. O método utilizado para determinar o peso da partícula é abordado no capítulo 4.2. O peso máximo w_{MAX} é multiplicado por um fator de redução f_w , de maneira que qualquer partícula com peso w maior que $f_w w_{MAX}$ terá um fator w_ϵ nulo, enquanto uma partícula de peso baixo terá um fator w_ϵ alto, fazendo com que ela se mova mais durante a predição. O fator w_ϵ é calculado pela equação

$$w_\epsilon = \max\{0, \ln f_w w_{MAX} - \ln w\},\tag{14}$$

que retorna um valor maior ou igual a 0. Utiliza-se uma função logarítmica para calcular o fator w_ϵ , pois o peso da partícula varia exponencialmente em função do erro de posição, desta forma, se fosse utilizada apenas a razão entre o peso máximo e o peso da partícula, o fator w_ϵ seria muito alto e iria variar muito de uma partícula para a outra. A função logarítmica faz com que w_ϵ seja linearmente proporcional ao erro da partícula.

Finalmente, deve-se utilizar as velocidades calculadas pelo conjunto de equações (13) no conjunto de equações (12), de maneira que

$$\begin{aligned}x_k &= x_{k-1} + \frac{\hat{v}_k^f}{\hat{v}_k^\theta} [\sin(\theta_{k-1} + \hat{v}_k^\theta \Delta t) - \sin(\theta_{k-1})] + \frac{\hat{v}_k^s}{\hat{v}_k^\theta} [\cos(\theta_{k-1}) - \cos(\theta_{k-1} + \hat{v}_k^\theta \Delta t)], \\ y_k &= y_{k-1} + \frac{\hat{v}_k^f}{\hat{v}_k^\theta} [\cos(\theta_{k-1}) - \cos(\theta_{k-1} + \hat{v}_k^\theta \Delta t)] + \frac{\hat{v}_k^s}{\hat{v}_k^\theta} [\sin(\theta_{k-1}) - \sin(\theta_{k-1} + \hat{v}_k^\theta \Delta t)], \\ \theta_k &= \theta_{k-1} + \hat{v}_k^\theta \Delta t.\end{aligned}\tag{15}$$

A distribuição de partículas após a aplicação do modelo de movimento representa a predição da posição do robô no instante atual, que é o termo $P(l_k | l_{k-1}, u_k)$ do filtro Bayesiano. O modelo de movimento aplica as equações (13) e (15), nessa ordem, sobre cada partícula, conforme o apresentado no algoritmo 5. Esse recebe como entrada a posição da partícula no instante anterior l_{k-1} , o comando de movimento u_k e o tempo desde a última iteração Δt , retornando como resposta a nova posição da partícula l_k .

Assim, foi apresentado o algoritmo que corresponde ao modelo de movimento implementado para o sistema de localização deste trabalho, que se baseia nas velocidades determinadas empiricamente de cada movimento executado pelo robô. Esses valores são obtidos medindo as distâncias percorridas pelo robô real ao executar cada movimento. Por exemplo, se o robô recebe o comando de andar para frente por um minuto, mede-se a distância que este andou, além

Figura 5 – Modelo de Movimento

Modelo de Movimento ($l_{k-1}, u_k, \Delta t$)

1. $\hat{v}_k^f \leftarrow v_k^f + \mathcal{N}(\sigma_1 v_k^f) + \mathcal{N}(\sigma_2 v_k^s) + \mathcal{N}(\sigma_3 v_k^\theta) + \mathcal{N}(\sigma_4 w_\epsilon) + \mathcal{N}(\sigma_5)$
2. $\hat{v}_k^s \leftarrow v_k^s + \mathcal{N}(\sigma_6 v_k^f) + \mathcal{N}(\sigma_7 v_k^s) + \mathcal{N}(\sigma_8 v_k^\theta) + \mathcal{N}(\sigma_9 w_\epsilon) + \mathcal{N}(\sigma_{10})$
3. $\hat{v}_k^\theta \leftarrow v_k^\theta + \mathcal{N}(\sigma_{11} v_k^f) + \mathcal{N}(\sigma_{12} v_k^s) + \mathcal{N}(\sigma_{13} v_k^\theta) + \mathcal{N}(\sigma_{14} w_\epsilon) + \mathcal{N}(\sigma_{15})$
4. **se** $\hat{v}_k^\theta = 0$ **então**
5. $x_k \leftarrow x_{k-1} + \hat{v}_k^f \cos(\theta_{k-1}) \Delta t + \hat{v}_k^s \sin(\theta_{k-1}) \Delta t$
6. $y_k \leftarrow y_{k-1} + \hat{v}_k^f \sin(\theta_{k-1}) \Delta t - \hat{v}_k^s \cos(\theta_{k-1}) \Delta t$
7. $\theta_k \leftarrow \theta_{k-1}$
8. **senão**
9. $x_k \leftarrow x_{k-1} + \frac{\hat{v}_k^f}{\hat{v}_k^\theta} [\sin(\theta_{k-1} + \hat{v}_k^\theta \Delta t) - \sin(\theta_{k-1})]$
10. $y_k \leftarrow y_{k-1} + \frac{\hat{v}_k^s}{\hat{v}_k^\theta} [\cos(\theta_{k-1}) - \cos(\theta_{k-1} + \hat{v}_k^\theta \Delta t)]$
11. $\theta_k \leftarrow \theta_{k-1} + \hat{v}_k^\theta \Delta t$
12. **retorna** l_k

Fonte: Autor

dos erros lateral e angular, repetindo o mesmo procedimento várias vezes, é possível calcular os valores médios dessas grandezas e seus desvios padrões. Essas médias e desvios padrões das velocidades do robô modelam o movimento do robô, sendo que as médias das velocidades são utilizadas para determinar o vetor u_k enquanto os desvios padrões são utilizados nos termos da equação (13).

Em seguida é apresentado o modelo de observação utilizado neste trabalho.

4.2 MODELO DE OBSERVAÇÃO

O mundo em que o robô está inserido pode ser descrito por características que podem ser observadas por sensores apropriados. Essas características podem ser mutuamente distinguíveis ou não, e suas posições no mundo devem ser conhecidas. Este trabalho considera que o sensor usado é uma câmera e que as informações de tais características são relativas à posição do robô. Outro sensor que será utilizado será o sensor inercial, que é capaz de fornecer a orientação do robô.

O modelo de observação representa o cálculo de $P(z_k | l_k)$ de um filtro Bayesiano, que é o cálculo da probabilidade da observação z_k ter sido feita pelo robô na posição l_k . No entanto, ao invés de calcular esta probabilidade, o modelo apresentado calcula um peso $W(z_k | l_k)$ que é proporcional à probabilidade $P(z_k | l_k)$ de uma dada partícula representar a posição do robô no domínio. Este trabalho apresenta três métodos para implementar o modelo de observação. O primeiro é o método tradicional de observação de *landmarks*, o segundo é um método que

utiliza pontos-chaves da imagem, observando se estes pontos estão dentro ou fora do campo, e o terceiro método foi implementado para ser usado em simulação.

4.2.1 Utilização de *Landmarks*

O *landmark* é uma característica do ambiente que pode ser identificada e classificada por sensores. Em geral, as informações obtidas por sensores como câmeras são tratadas antes que possam ser usadas para o processo de localização. Esse tratamento consiste em extrair da imagem as características do ambiente e estimar o posicionamento destes itens com relação ao robô. No caso deste trabalho, considera-se que a câmera ao observar o *landmark* retorna apenas a direção relativa em que o *landmark* foi observado, aqui representado pelo ângulo ϕ , sendo que os *landmarks* considerados foram apenas os postes dos gols.

Este método considera que o cálculo do peso W da partícula é feito utilizando dois termos: um que utiliza a distância d da partícula ao *landmark* ($W_\delta(z_k|l_k)$) e uma comparação do ângulo medido ϕ e do ângulo calculado φ ($W_\varphi(z_k|l_k)$).

A figura 6 apresenta como são obtidos os valores de distância d e direção φ do *landmark* relativos à posição da partícula. Da figura 6, a posição (l_x, l_y) é a posição da partícula no domínio, enquanto (q_x, q_y) é a posição do *landmark* no mundo. Considerando que as posições dos *landmarks* são previamente conhecidas pelo robô, e que o robô é capaz de reconhecer estes *landmarks*. A distância d é a distância Euclidiana entre a posição da partícula e a posição do *landmark*, por sua vez φ é a direção relativa que o robô observaria o *landmark* dado que o robô estivesse na posição representada pela partícula.

A utilização da distância relativa entre partícula e *landmark* é justificada pelas limitações do domínio. A primeira limitação é que *landmarks* distantes podem não ser observados por se tornarem irreconhecíveis, conforme a figura 7. Outro problema é a oclusão, o domínio deste trabalho apresenta um ambiente dinâmico, que contém objetos e agentes que se movem pelo

Figura 6 – Distância d e direção φ relativas entre partícula e *landmark*

Fonte: Autor

Figura 7 – *Landmark* distante 5.5m do robô

Fonte: Autor

ambiente sem o controle ou conhecimento do robô. Esses objetos e agentes eventualmente podem ocluir um *landmark*, e quanto maior a distância entre robô e *landmark* maior as chances de oclusão. Por fim, deve-se considerar que o próprio corpo de robô pode ocluir *landmarks* próximos à ele.

Assim, a equação que considera as três limitações citadas para o cálculo do peso em função da distância d é dada por

$$W_{\delta}(z_k|l_k) = \left(\psi + (1 - \psi) e^{-\frac{d^2}{2\sigma_o^2}} \right) \cdot \left(\frac{1 + \operatorname{erf}\left(\frac{\mu_f - d}{\sqrt{2}\sigma_f}\right)}{2} \right) \cdot \left(\frac{1 - \operatorname{erf}\left(\frac{\mu_n - d}{\sqrt{2}\sigma_n}\right)}{2} \right), \quad (16)$$

onde ψ é a probabilidade de um *landmark* não estar ocluído quando a distância d é muito grande, σ_o é a taxa de decaimento deste fator, onde para uma distância 0 não deve haver oclusão, este termo surge da premissa de que o aumento da probabilidade de um objeto ser ocluído por outro é proporcional à distância; μ_f e σ_f são a média e o desvio padrão da distância máxima na qual o robô consegue observar um *landmark*, μ_n e σ_n são a média e o desvio padrão da distância mínima a partir da qual o robô começa a ser capaz de observar um *landmark* e erf é a função erro, que foi escolhida nesta situação por representar os limiares das distâncias mínimas e máximas como regiões em que um objeto pode ser ou não detectado.

A figura 8 apresenta a curva característica desta equação, onde a curva tracejada representa apenas o primeiro termo da equação (16) e a outra representa a variação do peso W_{δ} em função da distância d . Na figura é possível observar que partículas muito próximas ou muito distantes do *landmark* devem receber um peso W_{δ} nulo.

O segundo termo do cálculo do peso, $W_{\varphi}(z_k|l_k)$, compara a observação feita pelo robô com a observação esperada pela partícula. Para tanto considere dois ângulos quaisquer α e β . Para calcular o quão semelhantes são esses ângulos pode-se utilizar uma função Gaussiana,

Figura 8 – Gráfico do valor de $W_\delta(z_k|l_k)$ em função da distância

Fonte: Autor

como

$$G(\alpha, \beta) = \frac{e^{-\frac{(\alpha-\beta)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}, \quad (17)$$

onde σ é o desvio padrão esperado entre os valores de α e β . Uma característica desta função é que ela possui um fator de normalização, de maneira que

$$\int_{\beta=-\infty}^{\beta=+\infty} G(\alpha, \beta) d\beta = 1, \quad (18)$$

onde foi considerado que α é o valor esperado em torno do qual β pode ser obtido dentro de um desvio padrão σ com probabilidade $G(\alpha, \beta)$.

No entanto, ângulos representam um domínio não contínuo, o que difere de um domínio mais simples como o da distância. Por exemplo, considere os ângulos $\alpha = +180^\circ$ e $\beta = -180^\circ$, eles são o mesmo ângulo, e quando aplicados sobre a função (17) deveriam retornar o valor máximo, no entanto, dependendo do valor de σ , o valor retornado pela função tende à 0.

Para contornar este problema, este trabalho propõe que, dados os ângulos α e β , eles sejam convertidos em números complexos pela função

$$Cpx(\gamma) = \cos \gamma + j \sin \gamma, \quad (19)$$

onde o valor retornado é um número complexo, com módulo 1. Então esses números complexos são aplicados na função (17).

Espera-se que o peso w calculado seja $0 \leq w \leq 1$. Para isso, usa-se uma versão não normalizada da função (17), usando como entrada os valores calculados a partir da utilização da função (19).

$$f(\alpha, \beta) = e^{-\frac{|Cpx(\alpha) - Cpx(\beta)|^2}{2\sigma_j^2}}, \quad (20)$$

Figura 9 – Comparação entre ângulos

Fonte: Autor

onde σ_j é o desvio padrão entre os números complexos que representam os ângulos α e β . Para facilitar o trabalho com o valor de σ_j , pode-se calcular o valor correspondente para números complexos do valor do σ usado para ângulos a partir de

$$\sigma_j = |Cpx(\sigma) - 1|. \quad (21)$$

onde σ é o valor desejado de desvio padrão para os ângulos.

Este método é apresentado graficamente na figura 9, onde d é a distância calculada entre os números complexos correspondentes à cada ângulo. Note que a distância $|Cpx(\alpha) - Cpx(\beta)|$ não é linear com relação à diferença de ângulos $|\alpha - \beta|$, entretanto a relação continua monotônica para ângulos que respeitam o domínio $0^\circ \leq |\gamma| \leq 180^\circ$.

Então, substituindo as equações (21) e (19) na função (20) tem-se que

$$W_\varphi(z_k|l_k) = e^{-\frac{|(\cos \phi + j \sin \phi) - (\cos \varphi + j \sin \varphi)|^2}{2|(\cos \sigma_\varphi + j \sin \sigma_\varphi) - 1|^2}}, \quad (22)$$

onde ϕ é o ângulo medido pelo robô, φ é o ângulo calculado em função da posição da partícula l_k e σ_φ é o desvio padrão usado para calcular o peso.

Finalmente, o cálculo de $W(z_k|l_k)$ é

$$W(z_k|l_k) = W_\delta(z_k|l_k) W_\varphi(z_k|l_k). \quad (23)$$

Até aqui foi considerada a existência de apenas um *landmark*. Quando existirem q *landmarks* e eles forem mutuamente distinguíveis, utiliza-se

$$W(z_k|l_k) = \prod_{i=1}^q W(z_k^i|l_k), \quad (24)$$

onde z_k^i se refere à direção relativa medida pelo robô do *landmark* i .

Entretanto, quando existem q *landmarks* e estes não são diferenciáveis, existe o problema da associação de informação, que é o problema de descobrir qual informação pertence a

Figura 10 – Algoritmo de cálculo de peso utilizando *landmarks*

Peso ($z_k|l_k$)

1. $w \leftarrow 1$
2. $w_\varphi \leftarrow 1$
3. $w_e \leftarrow 1$
4. **para cada elemento** $i \in z_k$
5. $d \leftarrow |l_k - q_{x,y}^i|$
6. $\varphi \leftarrow$ direção relativa do robô ao *landmark*
7. $w \leftarrow w \cdot \left(\psi + (1 - \psi) e^{-\frac{d^2}{2\sigma_o^2}} \right) \cdot \left(\frac{1 + \operatorname{erf}\left(\frac{\mu_f - d}{\sqrt{2}\sigma_f}\right)}{2} \right) \cdot \left(\frac{1 - \operatorname{erf}\left(\frac{\mu_n - d}{\sqrt{2}\sigma_n}\right)}{2} \right)$
8. $aux \leftarrow e^{-\frac{|(\cos \phi + j \sin \phi) - (\cos \varphi + j \sin \varphi)|^2}{2|(\cos \sigma_\varphi + j \sin \sigma_\varphi) - 1|^2}}$
9. $w \leftarrow w \cdot aux$
10. $w_\varphi \leftarrow w_\varphi \cdot aux$
11. $w_e \leftarrow w_e \cdot f_w$
12. **fim**
13. $w_\epsilon \leftarrow \max\{0, \ln w_e - \ln w_\varphi\}$
14. **retorna** w, w_ϵ

Fonte: Autor

qual *landmark*. Considere que o vetor z_k possui q valores, e que o vetor z'_k contém os q valores de direção dos *landmarks* que foram calculados em função da posição da partícula l_k , deve-se ordenar os valores de maneira que a equação (24) seja maximizada.

O valor w_ϵ é calculado a partir da comparação entre os ângulos esperado ϕ e medido φ . Esse valor é no máximo 1, que é obtido quando ambos os ângulos são iguais. Quando q *landmarks* são observados a equação (14) se torna

$$w_\epsilon = \max\{0, \ln \prod_{i=1}^q f_w - \ln \prod_{i=1}^q W_\varphi(z_k|l_k)\}. \quad (25)$$

A implementação do método é apresentada pelo algoritmo da figura 10.

4.2.2 Utilização de Pontos-Chaves

Uma outra forma de se fazer observações do ambiente é observar pontos-chaves na imagem. Este método consiste na comparação entre pontos da imagem que representam pontos com distância e direção conhecidas em relação ao robô, com os mesmos pontos observados a partir do ponto de vista das partículas.

Considere a figura 11, sempre que uma câmera com inclinação γ à uma altura h observa um ponto p da imagem I , esse ponto p será uma representação do ponto P no mundo real à uma distância D . Assim, se nem a altura h e nem a inclinação γ se alterar, o ponto p na imagem sempre irá representar o ponto P no mundo real, cuja distância e orientação são conhecidas. Na figura 11 λ representa o campo de visão vertical da câmera, e d é a distância do ponto p ao limite inferior da imagem.

Figura 11 – Correlação entre um ponto P no mundo real e um ponto p na imagem

Fonte: Autor

O método propõe que pontos-chaves determinados empiricamente representarão pontos com distância e direção conhecidos no mundo. A premissa é que se forem utilizados infinitos pontos para prever a observação que o robô faria se ele estivesse na posição representada pela partícula, seria possível calcular o peso de dada partícula a partir da comparação entre a imagem observada pelo robô e a imagem prevista por este método. A simplificação feita da ideia inicial é que, ao processar a imagem obtida pela câmera, é possível determinar se o ponto no mundo real está dentro ou fora do campo de futebol. Então essa informação é comparada ao esperado pelas partículas.

No entanto, a afirmação de que a câmera do robô tem altura e inclinação fixas é falsa. Conforme o robô se move, a câmera adquire um movimento de balanço durante todo o tempo. Fazendo com que o ponto P representado pelo ponto p da imagem esteja sempre em movimento. Quando esse ponto está próximo do limiar entre ser classificado como dentro ou fora do campo a informação obtida pelo sistema de visão não é confiável. Essa situação piora com a distância, de maneira que um ponto P muito distante varia muito sua distância D com o movimento.

Entretanto, isso não invalida o método. O movimento do robô gera um balanço senoidal, fazendo com que haja uma altura média h_m e uma inclinação média γ_m , assim espera-se que em média os pontos sejam classificados corretamente.

Para se calcular o peso a partir da observação dos pontos-chaves é preciso considerar quatro situações para cada ponto:

- o robô observa o ponto-chave p dentro do campo e a partícula espera que o ponto-chave q esteja dentro do campo;
- o robô observa o ponto-chave p fora do campo e a partícula espera que o ponto-chave q esteja fora do campo;

- c) o robô observa o ponto-chave p dentro do campo e a partícula espera que o ponto-chave q esteja fora do campo;
- d) o robô observa o ponto-chave p fora do campo e a partícula espera que o ponto-chave q esteja dentro do campo.

A primeira e a segunda situação representam a concordância da informação entre o observado e o esperado. A terceira e a quarta situação representam a discordância entre essas informações. Em todas essas situações o sistema de visão pode ter errado a classificação do ponto chave.

Desta forma o cálculo do peso é dado por

$$W(z_t|l_t) = \kappa_d \kappa_{p,q} \quad (26)$$

onde p e q podem estar dentro ou fora do campo, $\kappa_{p,q}$ é o fator que quantifica a importância de cada uma das situações descritas e κ_d é um fator que varia em função da distância dos pontos-chaves.

Os valores dos coeficientes são obtidos da seguinte forma: diversas imagens são obtidas a partir de posições conhecidas, de forma que sabe-se como cada ponto deveria ser classificado; então a primeira situação é representada pela razão entre a quantidade de pontos classificados como dentro do campo pelo sistema de visão e a quantidade de pontos classificados, os demais valores são obtidos da mesma forma; e o valor de κ_d representa a confiabilidade de um ponto dada a sua distância, de maneira que este valor é a razão entre a quantidade de pontos classificados corretamente e a quantidade de pontos classificados.

Para que o método seja efetivo, é necessário usar vários pontos-chaves para fazer o cálculo do peso. Assim, o peso final é calculado usando a fórmula (24).

O peso máximo obtido por este método é obtido quando os pontos-chaves observados pelo robô são classificados de acordo com o esperado pela partícula. Assim, a equação (14) é dada por

$$w_\epsilon = \max\{0, \ln \prod_{i=1}^q f_w \kappa_d \kappa_{p,p} - \ln \prod_{i=1}^q \kappa_d \kappa_{p,q}\} \quad (27)$$

Figura 12 – Algoritmo de cálculo de peso usando pontos-chaves

```

Peso ( $z_k|l_k$ )
1.   $w \leftarrow 1$ 
2.   $w_e \leftarrow 1$ 
3.  para cada elemento  $i \in z_k$  faça
4.       $q \leftarrow [dentro, fora]$  de acordo com sua posição no domínio
5.       $w \leftarrow w \cdot \kappa_d \kappa_{p,q}$ 
6.       $w_e \leftarrow w_e \cdot f_w \kappa_d \kappa_{p,q}$ 
7.  fim
8.   $w_\epsilon \leftarrow \max\{0, \ln w_e - \ln w\}$ 
9. retorna  $w, w_\epsilon$ 

```

Fonte: Autor

onde o peso máximo resulta da multiplicação dos coeficientes das situações que concordam com a observação p do ponto-chave feita pelo robô.

Finalmente, o algoritmo da figura 12 apresenta a implementação do método de cálculo de peso descrito.

4.2.3 Utilizando a Distância da Borda

O terceiro método é o mais simples, que utiliza a distância linear do robô à borda em uma determinada direção. Este método foi usado nos experimentos com o objetivo de reduzir o custo computacional da simulação da visão. Utilizar *landmarks* ou os pontos-chaves possuem como problema principal o peso computacional. No caso dos *landmarks* esse problema não é tão acentuado, mas ainda assim, isso implica que para cada partícula é preciso calcular as orientações e distâncias relativas para q os *landmarks* para cada uma das n partículas. Para o método de pontos-chaves, quando são utilizados vários pontos, o método deve calcular as posições de cada ponto para cada partícula, verificar se esses pontos estão dentro ou fora do campo e comparar com a observação feita pela partícula (por exemplo em testes preliminares foram utilizados 29 pontos (ALMEIDA; COSTA; BIANCHI, 2017)). Em ambos os casos, o tempo de execução de cada iteração do algoritmo pode se tornar tão longo que tornaria o método inviável.

No caso da simulação, seu objetivo é avaliar o funcionamento das contribuições feitas à implementação do método de localização de Monte-Carlo. Assim, para testar as influências

Figura 13 – Cálculo da distância até a borda

das contribuições sobre o método de localização de Monte-Carlo foi feita uma implementação simples de um modelo de observação que considera que o robô tem um campo de visão que varia em função da posição da cabeça, e que as informações obtidas pelo sistema de visão são dependentes da posição do campo de visão.

A distância até a borda d é calculada em função da distância em linha reta até a borda d_x , da orientação do robô θ e da orientação da cabeça do robô λ da seguinte forma

$$d = \frac{d_x}{\cos(\lambda - \theta)}. \quad (28)$$

A figura 13 apresenta as relações descritas. Os pontos A, B, C e D representam os vértices do campo e α , β , γ e δ são os ângulos formados entre os vértices e o ângulo 0 tendo o robô como centro.

Para se calcular o peso da partícula, é preciso saber qual é a direção absoluta ($\lambda - \theta$) que o robô simulado está observando e qual a distância d observada pelo robô. Então utilizando a posição l_k da partícula é possível calcular a distância d_p que a partícula espera observar a borda. Isso é feito seguindo o processo apresentado no algoritmo da figura 14.

Primeiro, a borda que está sendo observada é identificada, então a equação (28) determina a distância d_p até a borda esperada em função da posição da partícula. Finalmente as distâncias são comparadas pela equação de similaridade

$$w = e^{-\frac{(d-d_p)^2}{2\sigma^2}}, \quad (29)$$

onde σ representa o desvio padrão utilizado na comparação. O peso máximo neste caso será igual à 1, que ocorre quando a distância observada d é igual a distância calculada d_p .

Figura 14 – Algoritmo de cálculo de peso usando distância para a borda

```

Peso ( $z_k|l_k$ )
1.  se  $\alpha \leq \lambda - \theta < \beta$  então
2.     $d_p \leftarrow$  distância à borda  $\overline{AB}$ 
3.  senão se  $\beta \leq \lambda - \theta < \gamma$  então
4.     $d_p \leftarrow$  distância à borda  $\overline{BC}$ 
5.  senão se  $\delta \leq \lambda - \theta < \delta$  então
6.     $d_p \leftarrow$  distância à borda  $\overline{CD}$ 
7.  senão
8.     $d_p \leftarrow$  distância à borda  $\overline{AD}$ 
9.  fim
10.  $w \leftarrow e^{-\frac{(d-d_p)^2}{2\sigma^2}}$ 
11.  $w_\epsilon \leftarrow \max\{0, \ln f_w - \ln w\}$ 
12. retorna  $w, w_\epsilon$ 

```

Fonte: Autor

Figura 15 – Algoritmo de cálculo de peso usando a orientação do robô

Peso ($z_k | l_k$)

1. $w \leftarrow e^{-\frac{|(\cos \phi + j \sin \phi) - (\cos \varphi + j \sin \varphi)|^2}{2|(\cos \sigma_\varphi + j \sin \sigma_\varphi) - 1|^2}}$
2. $w_\epsilon \leftarrow \max\{0, \ln f_w - \ln w\}$
3. **retorna** w, w_ϵ

Fonte: Autor

4.2.4 Utilizando a Orientação do Sensor

Todos os métodos descritos até o momento utilizaram apenas a visão do robô como sensor. No entanto, todos os métodos apresentados possuem o problema da ambiguidade. A ambiguidade surge da falta de informações sobre características únicas do ambiente, que pode ser ocasionada pela simetria do ambiente, por exemplo. O domínio do futebol é um domínio totalmente simétrico, e por isso existe a dificuldade de associar a informação obtida ao *landmark* correto.

Para contornar esse problema este trabalho utiliza o sensor inercial para determinar qual é a orientação do robô. O sensor inercial possui acelerômetros e giroscópios, que retornam a orientação do robô ao longo do tempo. Esta informação de orientação é usada para calcular os pesos das partículas.

O peso é calculado pela equação (22), onde ϕ é substituído pelo ângulo obtido pelo sensor inercial, φ é substituído pela orientação da partícula e σ_φ é o desvio padrão considerado. O algoritmo 15 implementa esta função. O peso máximo neste caso é 1.

Quando o MCL executa o modelo de observação, dependendo do que foi observado pelo robô, é utilizado um dos algoritmos 10, 12 ou 14. O resultado obtido por estes algoritmos será multiplicado pelo resultado obtido pelo algoritmo 15, que representa o peso e o fator de erro da partícula. O próximo passo do filtro é selecionar as partículas de acordo com seus pesos.

4.3 MÉTODO DE REAMOSTRAGEM

A fase de reamostragem é característica do filtro de partículas, sendo esta a fase que finaliza o cálculo da representação da distribuição de probabilidades *a posteriori*, que será usada como a distribuição *a priori* na próxima iteração do filtro. Nesta fase as partículas são selecionadas de acordo com seus pesos, de maneira que partículas com pesos maiores tenham mais cópias ao fim da iteração, enquanto partículas com pesos baixos serão descartadas.

Deve-se considerar para o processo de reamostragem que quando se inicia uma nova iteração do algoritmo e não foi feita qualquer observação, então todas as partículas recebem peso igual à 1, neste caso, quando todas as partículas recebem pesos iguais, todas devem ser selecionadas para a próxima iteração do filtro. Isso evita que o filtro perca informações apenas

por ter executado uma iteração na qual não houveram informações para ponderar os pesos das partículas. Considera-se também que qualquer partícula com peso maior que o peso médio das partículas deve ser selecionada ao menos uma vez para a próxima iteração, de maneira que dadas novas observações, a certeza gerada por essa partícula seja confirmada. Por fim, a última consideração que deve ser feita é que este método não deve ser computacionalmente custoso, de maneira que seu funcionamento deve ter complexidade temporal de $O(n)$.

A partir das considerações feitas, este trabalho propõe um método de reamostragem para o filtro de partículas. Considere o vetor H_k que contém n partículas, cada partícula $h_k^i \in H_k$ possui um peso dado por w_i . O primeiro passo é reordenar o vetor H_k aleatoriamente. Isso faz com que conjuntos de partículas semelhantes que estejam contíguas no vetor sejam dispersas. Em seguida os pesos são enfileirados, conforme a figura 16, onde m_i é o peso acumulado. Então o tamanho do passo s é calculado pela equação

$$s = \frac{m_n}{n' + 1}, \quad (30)$$

onde o n' é o número de partículas que serão amostradas e m_n é o valor do somatório de todos os pesos. Quando os pesos são normalizados m_n deve ser igual a 1.

Figura 16 – Método de Reamostragem de Partículas

(a) Quando todas as partículas têm o mesmo peso e $n' = n = 10$.

(b) Quando as partículas têm pesos diferentes e $n' = n = 10$.

(c) Quando as partículas têm pesos diferentes e $n' = 5$ e $n = 10$.

(d) Quando as partículas têm pesos diferentes e $n' = 15$ e $n = 10$.

A figura 16a apresenta uma situação em que todas as partículas têm o mesmo peso, e todas elas são selecionadas para a próxima iteração. Na figura 16b cada partícula tem um peso diferente, ao fim da iteração a primeira, a terceira e a sétima partícula serão descartadas, enquanto a oitava partícula terá 3 cópias para a próxima iteração. Quando o número de partículas que serão amostradas é menor que o número de partículas existentes no conjunto, conforme a figura 16c, algumas partículas relativamente boas são perdidas, como foram o caso da sexta e décima partículas, mas partículas muito boas passam para a próxima iteração, como é o caso da oitava partícula. Na situação inversa, quando aumenta-se a quantidade de partículas, as chances de partículas ruins serem selecionadas é maior, mas a figura 16d mostra que partículas ruins ainda são descartadas do processo, como foi o caso da primeira e da terceira partícula.

Este algoritmo irá selecionar partículas proporcionalmente aos seus pesos, sendo que a quantidade de partículas selecionadas pode ser variada entre uma iteração e outra. No caso excepcional onde todas as partículas têm o mesmo peso, todas elas serão selecionadas para a próxima iteração, dado que $n' = n$. Além disso, nem a primeira e nem a última partícula na fila serão forçadamente selecionadas.

Esta etapa do filtro é importante, pois é nela que a qualidade da estimativa do filtro pode ser observada, visto que ao final desta fase é possível descobrir se o filtro é tendencioso. Este algoritmo apresenta as seguintes características:

Figura 17 – Método de reamostragem de partículas

```

Reamostragem ( $H_k, n'$ )
1.  ordena o vetor  $H_k$  aleatoriamente
2.  cria os vetores  $m$  e  $R$  vazios
3.  para cada elemento  $i \in H_k$  faça
4.       $m_i \leftarrow \sum_{j=1}^i w_j$ 
5.  fim
6.   $s \leftarrow \frac{m_n}{n'+1}$ 
7.   $i \leftarrow 1$ 
8.  para  $C$  de 1 até  $n'$  faça
9.       $Proximo \leftarrow Falso$ 
10.     enquanto  $Proximo = Falso$  faça
11.         se  $m_i > s \times C$  então
12.              $R \leftarrow H_k^i$ 
13.              $Proximo \leftarrow Verdadeiro$ 
14.         senão
15.              $i \leftarrow i + 1$ 
16.         fim
17.     fim
18. fim
19. retorna  $R$ 

```

Fonte: Autor

- a) se todas as partículas tiverem o mesmo peso, todas elas devem ser selecionadas para a próxima iteração do MCL;
- b) toda partícula que tenha um peso acima da média, deve ser selecionada ao menos uma vez;
- c) se uma partícula tiver um peso muito maior que as demais, esta deve ser selecionada mais vezes;
- d) o tempo de execução deste algoritmo aumenta linearmente com o aumento da quantidade de partículas.

Observe que o denominador da equação (30) é $n' + 1$ para que nem a última e nem a primeira partícula sejam selecionadas forçadamente. Se este denominador fosse apenas n' , durante a iteração final do algoritmo de reamostragem da linha 8 do algoritmo da figura 17, o peso acumulado seria igual ao peso total das partículas, desta forma a última partícula seria selecionada, apenas por estar no fim do vetor. Outra consideração é que se o valor de C fosse iniciado em 0 e variasse até $n' - 1$, para evitar a última partícula, a primeira partícula sempre seria selecionada. Isso aconteceria pois o valor inicial do peso acumulado seria 0, forçando que a primeira partícula sempre fosse selecionada. Ao utilizar $n' + 1$ para calcular o valor do passo s e iniciar a contagem com $C = 1$, implica que nem a primeira e nem a última partícula será selecionada apenas por estar alocada em uma posição que pode ser considerada mais privilegiada que as demais posições do vetor.

O algoritmo da figura 17 apresenta o funcionamento do método de reamostragem descrito.

4.4 QUANTIDADE DE PARTÍCULAS ADAPTÁVEL

Após a reamostragem das partículas, é possível verificar quantas partículas são necessárias para representar a distribuição de probabilidades. Este trabalho considera que a dispersão das partículas pode ser utilizada para determinar a quantidade de partículas. Quando o filtro executa o modelo de movimento as partículas se espalham, então o modelo de observação pondera essas partículas, em alguns casos, partículas distantes vão apresentar pesos semelhantes, e quando o filtro reamostra essas partículas várias delas são selecionadas. Isso implica que o filtro tem muita incerteza da posição do robô, que pode ter sido causada por ruídos na observação feita pelo robô.

Então, quando existe muita incerteza na observação do robô são necessárias muitas partículas para representar todas as possíveis posições que o robô real pode estar. Por outro lado, quando a incerteza é baixa, poucas partículas são o suficiente para representar a distribuição de probabilidades. A figura 18 apresenta quatro situações distintas, na primeira 30 partículas tentam representar uma distribuição probabilística com muita incerteza e a segunda apresenta 1000 partículas representando a mesma distribuição de probabilidades, já na terceira 30 partículas são usadas para representar uma distribuição probabilística com incerteza baixa e a quarta

Figura 18 – Distribuição de probabilidades e partículas

(a) Utiliza 30 partículas.

(b) Utiliza 1000 partículas.

(c) Utiliza 30 partículas.

(d) Utiliza 1000 partículas.

Fonte: Autor

imagem apresenta uma representação da mesma distribuição feita por 1000 partículas. As figuras apresentam um plano unidimensional.

Este trabalho utiliza o desvio padrão das partículas para representar o espalhamento das partículas, que indiretamente representa a incerteza da distribuição de probabilidades. Para se obter o desvio padrão, o primeiro passo é obter a matriz de covariância das posições

$$\Sigma = \frac{1}{n-1} \sum_{i=1}^n S_i^T S_i. \quad (31)$$

onde S é uma matriz 1×3 representando a diferença entre a posição da partícula i e a média das posições do conjunto de partículas e n é a quantidade de partículas.

Após o cálculo da matriz de covariância Σ , calcula-se o determinante desta matriz $|\Sigma|$, de maneira que ele se torne um número. Então obtém-se a raiz quadrada do número, que resulta no determinante da matriz de desvios padrões $|\sigma|$.

Figura 19 – Quantidade de partículas em função do desvio padrão de suas posições

Fonte: Autor

O valor de $|\sigma|$ é convertido em quantidade de partículas por

$$n' = |\sigma| \frac{n_{max} - n_{min}}{4\varepsilon} + n_{min} + \frac{n_{max} - n_{min}}{2} - \mu \frac{n_{max} - n_{min}}{4\varepsilon} \quad (32)$$

onde n' é a quantidade de partículas necessárias para representar a distribuição de probabilidades, n_{max} e n_{min} são os limites superior e inferior da quantidade de partículas, respectivamente, ε representa o quão rápido a quantidade de partículas aumenta com o aumento do valor de $|\sigma|$ e μ é o valor de $|\sigma|$ em que a quantidade de partículas n' é igual à média entre n_{max} e n_{min} . A figura 19 apresenta a curva característica da equação (32), onde n_{med} é o valor médio entre n_{min} e n_{max} .

O algoritmo 20 apresenta o método implementado, que consiste na aplicação do cálculo do desvio padrão das posições das partículas, seguido da conversão deste valor para quantidade de partículas. Ele recebe como entrada o vetor de partículas e retorna a quantidade de partículas.

Figura 20 – Método que varia quantidade de partículas

Quantidade (H_k)

1. $M \leftarrow$ posição média de H_k
2. $S \leftarrow H_k - M$
3. $\Sigma \leftarrow \frac{1}{n-1} \sum_{i=1}^n S_i^T S_i$
4. $|\sigma| \leftarrow \sqrt{|\Sigma|}$
5. $n' \leftarrow |\sigma| \frac{n_{max}-n_{min}}{4\varepsilon} + n_{min} + \frac{n_{max}-n_{min}}{2} - \mu \frac{n_{max}-n_{min}}{4\varepsilon}$
6. **retorna** n'

Fonte: Autor

4.5 MELHOR OBSERVAÇÃO

O domínio da *RoboCup Soccer Humanoid League* restringe o principal sensor do robô, a câmera. Este sensor não pode garantir ao robô um campo de visão maior que 180° (ROBOCUP SOCCER HUMANOID LEAGUE, 2016), impedindo que o robô tenha visão global do mundo. Com isso, para que o robô seja capaz de observar todo o mundo a sua volta ele precisa fazer várias observações parciais. No entanto, algumas observações podem não conter informações úteis, e o tempo utilizado para processar tal imagem é desperdiçado.

Uma informação útil, é definida como aquela que auxilia no processo de localização, por exemplo, uma imagem em que se observa algum *landmark*. A utilidade de uma informação representa o quanto dada informação melhorou o estado atual com relação ao estado anterior. Em um filtro de partículas isso se reflete no quanto a informação auxiliou no processo de convergência das partículas.

Antes de se obter qualquer informação, é possível verificar o quão útil cada observação pode ser, analisando a informação esperada de cada observação possível, esse método é conhecido como Valor da Informação Perfeita (RUSSELL et al., 2010). Em um filtro de partículas, considerando a hipótese de que o mundo é um sistema de Markov de primeiro grau, pode se afirmar que a observação feita do mundo depende apenas da posição atual do robô. A crença nessa posição é representada pelas partículas, onde cada partícula tem probabilidade de representar a posição do robô no mundo real. A partir das posições das partículas é possível determinar qual será a observação feita pelo robô, e assim determinar qual a informação esperada de tal observação.

A crença do filtro de partículas é representada pelo desvio padrão das partículas, que é uma métrica de espalhamento. Desta forma, a utilidade de uma informação pode ser calculada pelo quanto esta informação reduz o desvio padrão do conjunto de partículas.

A utilidade de uma informação também está atrelada ao esforço em obtê-la. Por exemplo, se duas informações contribuem igualmente para o processo de localização, o robô deverá obter a informação que exige o menor esforço para ser observada. O esforço pode considerar fatores físicos e computacionais, como por exemplo, o quanto um robô precisa se mover para observar algo, ou quanto tempo de processamento é necessário para se obter dada informação. Para o domínio em questão, a posição da cabeça representa um fator físico de esforço, pois a cabeça pode se mover horizontalmente de -90° até 90° , assim o método apresentado considera o movimento da cabeça como parte da função de utilidade.

Considerando que o movimento do robô não se altera em instantes futuros, é possível prever qual será a distribuição de partículas em um instante futuro e, a partir desta distribuição, estimar qual das possíveis observações contém a informação mais útil para o processo de localização. Desta forma espera-se que o robô evite fazer observações que não auxiliem o processo de localização.

Figura 21 – Método que estima a melhor observação

Melhor Observação ($H_k, u_k, \alpha, \Delta t, m$)

1. $P \leftarrow$ cópias de m partículas aleatórias de H_k
2. **para cada** elemento $\rho \in P$ **faça**
3. $\rho \leftarrow$ **Modelo de Movimento**(ρ, u_k, Δ)
4. **fim**
5. $desvio_i \leftarrow$ desvio padrão das posições das partículas
6. **para cada** observação possível β **faça**
7. $z_k \leftarrow$ informação provável considerando a posição P_0 e β
8. **para cada** elemento $\rho \in P$ **faça**
9. $peso(\rho) \leftarrow$ **Peso**($z_k|\rho$)
10. **fim**
11. $P \leftarrow$ **Reamostragem**(P, m)
12. $desvio_f \leftarrow$ desvio padrão das posições após a reamostragem
13. $utilidade(\beta) \leftarrow \eta_1(desvio_i - desvio_f) - \eta_2|\alpha - \beta|$
14. **fim**
15. $\gamma \leftarrow \beta \propto utilidade(\beta)$
16. **retorna** γ

Fonte: Autor

O método implementado é descrito no algoritmo da figura21. Ele recebe o conjunto de partículas H_k , o vetor de movimento u_k , a direção atual observada α , o tempo de predição Δt e a quantidade de partículas m que será utilizada para o método. No começo m partículas são copiadas aleatoriamente do conjunto H_k , então é aplicado o modelo de movimento u_k sobre as posições das partículas por um dado tempo Δt , com o objetivo de prever como estará a distribuição de probabilidades em instantes futuros. Em seguida calcula-se o desvio padrão das partículas após a predição. Então para cada observação possível é gerada uma informação, considerando que o robô esteja na posição dada pela partícula P_0 e direção de observação β . Calcula-se os pesos das partículas e elas passam pela fase de reamostragem. A partir das partículas é calculado o novo desvio padrão. E então calcula-se a utilidade daquela observação, levando em consideração a diferença entre os desvios e o quanto é necessário o robô se mover para obter aquela informação. Finalmente, cada observação tem probabilidade proporcional à sua utilidade de ser selecionada.

4.6 ALGORITMO FINAL

Finalmente, o filtro de partículas implementado neste trabalho está descrito no algoritmo da figura 22. Ele recebe o conjunto de partículas H_{k-1} , que foram geradas na iteração anterior, o vetor de movimento u_k , o vetor de observações z_k , o tempo desde a última iteração Δt , a quantidade de partículas n que deverá ser utilizada para representar a distribuição probabilística, a posição atual da cabeça α e o tempo de predição $\Delta' t$ para estimar qual a melhor observação à ser feita em instantes futuros.

Figura 22 – Algoritmo de Localização de Monte-Carlo

Localização de Monte-Carlo ($H_{k-1}, u_k, z_k, \Delta t, n, \alpha, \Delta' t$)

1. cria o conjunto H_k de partículas vazio
2. **para cada** elemento $\rho \in H_{k-1}$ **faça**
3. $\rho \leftarrow \text{Modelo de Movimento}(\rho, u_k, \Delta t)$
4. $w_\rho \leftarrow \text{Peso}(z_k | \rho)$
5. $H_k \leftarrow \rho$
6. **fim**
7. $H_k \leftarrow \text{Reamostragem}(H_k, n)$
8. $\gamma \leftarrow \text{Melhor Observação}(H_k, u_k, \alpha, \Delta' t)$
9. **retorna** H_k, n', γ

Fonte: Autor

Primeiro cria-se um vetor auxiliar H_k , que irá receber as partículas modificadas nesta iteração. Então todas as partículas são movidas de acordo com o modelo de movimento e seus pesos calculados pelo modelo de observação, em seguida cópias das partículas são feitas proporcionalmente aos seus pesos. Então calcula-se qual a quantidade de partículas podem ser utilizadas para representar a distribuição de probabilidades. Finalmente, estima-se qual observação deve obter a melhor informação para o filtro de partículas no futuro.

O filtro é iniciado a partir de uma distribuição uniforme de partículas sobre todo o domínio. Ao longo do tempo, essas partículas devem convergir em torno da posição que representa o robô no mundo real.

Assim, este capítulo apresentou toda a implementação do sistema de localização. O próximo capítulo irá apresentar o conjunto de experimentos realizados para avaliar a implementação apresentada.

5 EXPERIMENTOS

Este capítulo apresenta os experimentos realizados para avaliação dos métodos. O capítulo 5.1 apresenta um experimento utilizado como referência para comparar os demais métodos. Os capítulos 5.2 e 5.3 apresentam os métodos de observação implementados no capítulo 4.2. O método que altera a quantidade de partículas de acordo com a necessidade é apresentado no capítulo 5.4, o capítulo 5.5 apresenta o método que estima de qual observação deve se obter a melhor informação e o capítulo 5.6 apresenta o experimento de recuperação de erro de estimativa, então o experimento do capítulo 5.7 testa os três métodos em conjunto. Finalmente, os capítulos 5.8 e 5.9 apresentam experimentos no robô real, sendo que os mesmos experimentos foram feitos em simulação. Cada um desses capítulos apresentam como o experimento foi configurado e seus resultados.

Note que em todos os experimentos feitos, a cabeça do robô pode assumir as orientações de -90° , 0° ou 90° com relação à orientação do robô, isso vale tanto para o robô real quanto para o robô simulado.

5.1 EXPERIMENTO SIMULADO DE REFERÊNCIA

Este experimento tem o objetivo de gerar uma base de comparação para os experimentos simulados, de maneira que ele serve para se obter o comportamento característico esperado do sistema de localização ao se utilizar as contribuições propostas por este trabalho. Neste experimento foi utilizado uma configuração do MCL, que não é capaz de se recuperar de erros de estimativa, a quantidade de partículas é constante e não utiliza o método da melhor observação. Além disso, como modelo de observação foram utilizados a distância à borda e a orientação do robô obtida pelo sensor inercial. Primeiro será apresentada a configuração utilizada para este experimento e em seguida os resultados obtidos.

5.1.1 Configuração

Para todos os experimentos simulados foi utilizado o simulador desenvolvido pelo time *RoboFEI-HT* (PERICO et al., 2016), a figura 23 apresenta uma imagem do simulador. Este simulador foi escolhido devido à portabilidade de código para o robô real usado pelo time.

Para este primeiro experimento foi feito o caminho apresentado na figura 24. Nele o robô inicia ao sul do campo, orientado para o norte, anda reto por 16 segundos e para. Então ele gira para a direita por 10 segundos, e anda por mais 15 segundos. Quando ele para novamente, ele gira para a esquerda por 4 segundos e anda 15 segundos e finaliza o seu movimento. No total este percurso leva 1 minuto para ser concluído.

O robô simulado se move idealmente neste experimento. No entanto seu sistema de visão é simulado com a adição de erros. Neste primeiro experimento foi usado o método de

Figura 23 – Simulador RoboFEI-HT

Fonte: Autor

observação baseado na distância até a borda, na direção em que o robô faz sua observação, além de utilizar a orientação do robô com o objetivo de filtrar ambiguidades.

O sistema de localização implementado considera o algoritmo apresentado no capítulo 4.6. Para o modelo de movimento, foi adotado que o conjunto de fatores σ_i receberam os seguintes valores:

Figura 24 – Caminho feito pelo robô em simulação

Fonte: Autor

- a) para σ_1 , σ_6 e σ_{11} , que representam o quanto a velocidade frontal v_k^f influencia o vetor de velocidades, adotou-se o valor de 1, o desvio padrão gerado por este termo será igual à velocidade;
- b) para σ_2 , σ_7 e σ_{12} , que representam o quanto a velocidade lateral v_k^s influencia o vetor de velocidades, adotou-se o valor de 2, o desvio padrão gerado por este termo será o dobro da velocidade;
- c) para σ_3 , σ_8 e σ_{13} , que representam o quanto a velocidade angular v_k^θ influencia o vetor de velocidades, adotou-se o valor de 1, o desvio padrão gerado por este termo será igual à velocidade;
- d) para σ_4 , σ_9 e σ_{14} , que representam o quanto o erro do peso w_ϵ influencia o vetor de velocidades, adotou-se o valor de 0, o que implica na não utilização do método para se recuperar de erros de estimativa;
- e) para σ_5 , σ_{10} e σ_{15} , que representam o fator de espalhamento das partículas para cada velocidade, adotou-se o valor de 5, 7 e 5, respectivamente.

Assim, este experimento não testa a capacidade do filtro de se recuperar do problema do sequestro. Esses valores são obtidos de maneira que para qualquer movimento que ele execute exista a probabilidade de pelo menos 1% do robô permanecer parado, assim modelando o erro existente no robô real, que apresenta variação em sua aceleração inicial, de modo que eventualmente ao enviar o comando o robô pode demorar alguns instantes para iniciar o movimento efetivamente. Além disso, as velocidades do vetor u_k adotadas para cada movimento executado foram:

- a) quando o robô anda, a velocidade frontal v_k^f foi de 20 cm/s e as demais eram nula;
- b) quando o robô girava, a velocidade de rotação v_k^θ foi de 10°/s e as demais eram nulas.

Para o cálculo dos pesos das partículas pelo modelo de observação foram utilizados os algoritmos 14 e 15, que retornam o peso em função da distância da borda observada e em função

Tabela 1 – Resumo da configuração do primeiro experimento

Quantidade de Partículas	1000				
Modelo de Movimento	$\sigma_1 \leftarrow 1$	$\sigma_2 \leftarrow 2$	$\sigma_3 \leftarrow 1$	$\sigma_4 \leftarrow 0$	$\sigma_5 \leftarrow 5$
	$\sigma_6 \leftarrow 1$	$\sigma_7 \leftarrow 2$	$\sigma_8 \leftarrow 1$	$\sigma_9 \leftarrow 0$	$\sigma_{10} \leftarrow 7$
	$\sigma_{11} \leftarrow 1$	$\sigma_{12} \leftarrow 2$	$\sigma_{13} \leftarrow 1$	$\sigma_{14} \leftarrow 0$	$\sigma_{15} \leftarrow 5$
Modelo de Observação	Landmarks	N/A			
	Pontos-Chaves	N/A			
	Distâncias	$\sigma \leftarrow 50\text{ cm}$ $f_w \leftarrow e^{-1}$			
	Orientação	$\sigma_\varphi \leftarrow 20^\circ$ $f_w \leftarrow e^{-1}$			
Adaptação da Quantidade de Partículas	N/A				
Melhor Observação	N/A				

Fonte: Autor

da orientação obtida do sensor inercial, respectivamente. Os valores utilizados para os fatores desses algoritmos foram:

- a) para o fator σ do algoritmo 14 foi utilizado o valor de 50 cm ;
- b) para o fator σ_φ do algoritmo 15 foi utilizado o valor de 20° ;
- c) para o fator f_w de ambos os algoritmos foi utilizado o valor de e^{-1} .

Foram utilizadas 1000 partículas durante o experimento. Os algoritmos que calculam a quantidade de partículas necessárias para representar a distribuição de probabilidades e o algoritmo que estima qual a observação que deve gerar a melhor informação não foram utilizados neste experimento. Cada experimento foi executado 30 vezes. A tabela 1 apresenta a configuração usada no MCL.

Assim, o robô deve se mover conforme o descrito no início desta seção, enquanto o filtro de partículas, utilizando os valores adotados na tabela 1, deve estimar a posição do robô enquanto se move pelo domínio. O erro de posição é calculado com relação a posição do robô obtida diretamente do simulador, esse erro de posição é utilizado para avaliar os demais experimentos.

5.1.2 Resultados

O gráfico, apresentado pela figura 25, mostra os erros de orientação, em graus, e posição, em cm , da posição estimada pelo sistema de localização e a posição real do robô simulado. Nele pode-se perceber a queda do erro ao longo do tempo, conforme as partículas convergem, este é o comportamento esperado dos demais experimentos.

Figura 25 – Erros de orientação e posição em função do tempo do experimento de referência

5.2 EXPERIMENTO SIMULADO UTILIZANDO *LANDMARKS*

Neste experimento foi utilizado o método de observação de *landmarks* para realizar o cálculo dos pesos das partículas. Este experimento é necessário pois a observação de *landmarks* é tradicionalmente utilizada para aquisição de informações para robôs que usam sensores visuais como método principal de observação do mundo.

5.2.1 Configuração

Este experimento foi simulado, utilizando como *landmarks* as traves verticais que formam os gols de ambos os lados do campo. Eles são representados na figura 24 pelos 4 Xs. O robô executa o mesmo caminho do experimento de referência, já descrito e apresentado na figura 24.

A visão simulada retorna as direções relativas das traves observadas em graus. Se as traves estão contidas no campo de visão do robô simulado, que corresponde à distância mínima de 80 cm , à distância máxima de 7 m e entre os ângulos de 26° e -26° em relação a direção em que a câmera do robô simulado está observando, conforme a figura 26, então o sistema de visão retorna a direção relativa entre o robô e o *landmark*. Estes valores são aproximadamente os mesmos valores empíricos obtidos de robôs reais. Cada ângulo obtido é adicionado de um erro com variância de 1° e então os ângulos obtidos são ordenados em ordem decrescente de valor.

Para representar a distribuição de probabilidades foram utilizadas 1000 partículas em todo o experimento. Os fatores do modelo de movimento são os mesmos utilizados no último experimento.

Figura 26 – Campo de visão do robô simulado

Fonte: Autor

Para o modelo de observação, foram utilizados os modelos apresentados pelos algoritmos 10 e 15. Como fatores para este modelo foram utilizados:

- para o fator ψ do algoritmo 10 foi utilizado o valor de 0.7;
- para o fator σ_o do algoritmo 10 foi utilizado o valor de 70 cm;
- para o fator μ_f do algoritmo 10 foi utilizado o valor de 700 cm, que implica que em média a distância máxima na qual o robô consegue identificar um *landmark* do tipo trave de gol é de 7 metros;
- para o fator σ_f do algoritmo 10 foi utilizado o valor de 10 cm;
- para o fator μ_n do algoritmo 10 foi utilizado o valor de 10 cm, que implica que a trave deve estar à 10 centímetros para ser identificada;
- para o fator σ_n do algoritmo 10 foi utilizado o valor de 1 cm;
- para o fator σ_φ do algoritmo 10 foi utilizado o valor de 5°;
- para o fator σ_φ do algoritmo 15 foi utilizado o valor de 20°;
- para o fator f_w de ambos os algoritmos foi utilizado o valor de e^{-1} .

Assim, esses valores representam que se o *landmark* está 2.1 m ($3 \times \sigma_o$) distante da posição representada pela partícula, deve haver uma probabilidade de 70% de chances de dado *landmark* ter sido observado. Além disso, após 7.3 m um *landmark* é impossível de ser observado, e finalmente que nenhum *landmark* pode ser observado à menos de 7 cm de distância do robô.

Os algoritmos que calculam a quantidade de partículas necessárias para representar a distribuição de partículas e o algoritmo que estima qual a observação que deve gerar a melhor informação não foram utilizados neste experimento. Cada experimento foi executado 30 vezes. A tabela 2 apresenta a configuração usada no MCL.

Tabela 2 – Resumo de configuração do experimento utilizando *landmarks*

Quantidade de Partículas	1000					
Modelo de Movimento	$\sigma_1 \leftarrow 1 \quad \sigma_2 \leftarrow 2 \quad \sigma_3 \leftarrow 1 \quad \sigma_4 \leftarrow 0 \quad \sigma_5 \leftarrow 5$ $\sigma_6 \leftarrow 1 \quad \sigma_7 \leftarrow 2 \quad \sigma_8 \leftarrow 1 \quad \sigma_9 \leftarrow 0 \quad \sigma_{10} \leftarrow 7$ $\sigma_{11} \leftarrow 1 \quad \sigma_{12} \leftarrow 2 \quad \sigma_{13} \leftarrow 1 \quad \sigma_{14} \leftarrow 0 \quad \sigma_{15} \leftarrow 5$					
Modelo de Observação	<i>Landmarks</i>	$\psi \leftarrow 0.7$ $\mu_n \leftarrow 10\text{ cm}$		$\sigma_o \leftarrow 70\text{ cm}$ $\sigma_n \leftarrow 1\text{ cm}$	$\mu_f \leftarrow 700\text{ cm}$ $\sigma_\varphi \leftarrow 5^\circ$	$\sigma_f \leftarrow 10\text{ cm}$ $f_w \leftarrow e^{-1}$
	<i>Pontos-Chaves</i>	N/A				
	Distâncias	N/A				
	Orientação	$\sigma_\varphi \leftarrow 20^\circ \quad f_w \leftarrow e^{-1}$				
Adaptação da Quantidade de Partículas	N/A					
Melhor Observação	N/A					

Fonte: Autor

Figura 27 – Erros de orientação e posição em função do tempo do experimento utilizando *landmarks*

Fonte: Autor

5.2.2 Resultados

A figura 27 apresenta os erros de orientação e posição do sistema de localização. É possível observar o comportamento esperado, parecido com o observado no gráfico da figura 25, em ambos foi observado que o tempo necessário para que as partículas convergissem na posição correta foi de mais ou menos 10 segundos. Além disso, é possível perceber que este método se comporta da maneira esperada, onde inicialmente quando o robô está perdido, o erro de posição é alto, mas com o tempo a posição estimada pelo filtro de partículas converge em torno da posição real do robô.

No entanto, existe um momento próximo aos 30 segundos em que a posição começa a ser perdida, neste instante o robô está andando para frente pela segunda vez. Nesta posição o robô está à 8 metros das traves ao leste do campo, que estão fora do campo de visão do robô. Assim, o filtro está sem informações de atualização, fazendo apenas a predição da posição do robô. Aos 40 segundos a posição começa a ser recuperada, levando ainda algum tempo para convergir novamente.

5.3 EXPERIMENTO SIMULADO UTILIZANDO PONTOS-CHAVES

Este experimento usa o método de pontos-chaves da imagem, cuja posição no mundo real são conhecidas para calcular o peso das partículas do filtro. Os pontos-chaves foram determinados de maneira que eles estivessem em ângulos e distâncias conhecidas no domínio.

Figura 28 – Posições dos pontos-chaves

Fonte: Autor

O objetivo deste experimento é avaliar o funcionamento do modelo de observação que utiliza pontos-chaves, que é parte da proposta deste trabalho. Note que este método é utilizado nos experimentos com robôs reais para o cálculo do peso das partículas dos capítulos 5.8 e 5.9.

5.3.1 Configuração

Este experimento foi simulado utilizando 10 pontos-chaves como informação do sistema de visão para o cálculo do peso. 6 desses pontos estão dispostos à 1 m, 2 m e 3 m, nas direções de 45° e -45°. Os outros 4 pontos estão dispostos em 0° à 50 cm, 1 m, 2 m e 4 m. Os pontos são apresentados na figura 28 com o símbolo $\times$.

Tabela 3 – Resumo de configuração do experimento utilizando pontos-chaves

Quantidade de Partículas	1000				
Modelo de Movimento	$\sigma_1 \leftarrow 1$	$\sigma_2 \leftarrow 2$	$\sigma_3 \leftarrow 1$	$\sigma_4 \leftarrow 0$	$\sigma_5 \leftarrow 5$
	$\sigma_6 \leftarrow 1$	$\sigma_7 \leftarrow 2$	$\sigma_8 \leftarrow 1$	$\sigma_9 \leftarrow 0$	$\sigma_{10} \leftarrow 7$
	$\sigma_{11} \leftarrow 1$	$\sigma_{12} \leftarrow 2$	$\sigma_{13} \leftarrow 1$	$\sigma_{14} \leftarrow 0$	$\sigma_{15} \leftarrow 5$
Modelo de Observação	Landmarks	N/A			
	Pontos-Chaves	$\kappa_d \leftarrow 1$	$\kappa_{00} \leftarrow 1$	$\kappa_{11} \leftarrow 1$	
		$f_w \leftarrow 0.25$	$\kappa_{10} \leftarrow 0.1$	$\kappa_{01} \leftarrow 0.3$	
	Distâncias	N/A			
	Orientação	$\sigma_\varphi \leftarrow 20^\circ$ $f_w \leftarrow e^{-1}$			
Adaptação da Quantidade de Partículas	N/A				
Melhor Observação	N/A				

Fonte: Autor

No simulador esses pontos recebem o valor de 1 se eles estiverem dentro dos limites do campo, e 0 do contrário. Para representar os erros da visão real, a distância em que esses pontos se encontram do robô podem variar a partir de uma distribuição normal com média 0 e desvio padrão de $1/10$ da própria distância, isso serve para tentar simular o erro causado pela movimentação do robô. Os ângulos dos pontos também sofrem com um desvio padrão de mais ou menos 1° . Além disso, cada ponto tem 30% de chances de ser aleatoriamente classificado como 0 ou 1, que serve para simular erros de classificação causados por falhas no processo de visão.

Neste experimento foram utilizadas 1000 partículas para representar a distribuição de probabilidades. Para o modelo de movimento foram utilizados os mesmos valores dos experimentos anteriores.

Para o modelo de observação foram utilizados os algoritmos 12 e 15. Os fatores considerados para estes algoritmos foram:

- para o fator κ_d foi utilizado o valor 1;
- para os fatores κ_{00} , κ_{11} , κ_{10} e κ_{01} do algoritmo 12 foram utilizados os valores 1, 1, 0.1 e 0.3, respectivamente;
- para o fator f_w do algoritmo 12 foi utilizado o valor 0.25;
- para o fator σ_φ do algoritmo 15 foi utilizado o valor de 20° ;
- para o fator f_w do algoritmo 15 foi utilizado o valor de e^{-1} .

Os algoritmos que calculam a quantidade de partículas necessárias para representar a distribuição de partículas e o algoritmo que estima qual a observação que deve gerar a melhor

Figura 29 – Erros de orientação e posição em função do tempo do experimento utilizando pontos-chaves

informação não foram utilizados neste experimento. Cada experimento foi executado 30 vezes. A tabela 3 apresenta a configuração usada no MCL.

5.3.2 Resultados

A figura 29 apresenta os resultados obtidos para este método. Pode-se observar que o erro obtido nele é muito maior do que aqueles obtidos nos outros dois métodos. No entanto, este método também apresenta o comportamento esperado para o sistema de localização.

Mesmo com esse erro o método é útil para o domínio. É desejável ter a informação da posição do robô com o mínimo de erro possível, mas apenas conhecendo a região na qual o robô se encontra já é possível definir estratégias. Por exemplo, se a bola está no campo de ataque e o robô no campo de defesa, o robô deve se mover na direção do campo de ataque até que a bola seja vista, quando a bola é vista um método reativo, que utiliza informações diretamente do sistema de visão, irá se alinhar com a bola e irá chutá-la na direção do gol.

Além disso, a estimativa melhora conforme o robô se aproxima das bordas do campo, que são regiões onde o interesse em se ter mais precisão é maior. Essa situação é apresentada no gráfico da figura 29 pela queda do erro próximo ao final do experimento, no qual o robô se aproxima do canto nordeste do campo, conforme a figura 24.

5.4 EXPERIMENTO SIMULADO ADAPTANDO A QUANTIDADE DE PARTÍCULAS

Este experimento utiliza as distâncias das bordas como informação para o cálculo dos pesos das partículas. Além disso, a quantidade de partículas varia em função do desvio padrão das posições das partículas.

O objetivo deste experimento é avaliar o funcionamento do método que varia a quantidade de partículas em função do desvio padrão das posições das partículas. Ao final do experimento, os resultados obtidos serão comparados aos obtidos pelo experimento de referência.

5.4.1 Configuração

Este experimento foi feito nas mesmas condições do experimento apresentado no capítulo 5.1. Assim, o caminho feito pelo robô é o caminho apresentado pela figura 24 e os modelos de movimento e observação são os mesmos apresentados no capítulo 5.1.

O método utilizado para variar a quantidade de partículas é feito pelo algoritmo 20, onde os fatores utilizados são:

- a) para o fator n_{max} e n_{min} foram utilizados os valores 1000 e 30, ou seja, a quantidade de partículas variam entre 30 e 1000;
- b) para o fator μ o valor utilizado foi 6;
- c) para o fator ε foi utilizado o valor 2.

Tabela 4 – Resumo de configuração do experimento com adaptação da quantidade de partículas

Quantidade de Partículas	de 30 à 1000				
Modelo de Movimento	$\sigma_1 \leftarrow 1$	$\sigma_2 \leftarrow 2$	$\sigma_3 \leftarrow 1$	$\sigma_4 \leftarrow 0$	$\sigma_5 \leftarrow 5$
	$\sigma_6 \leftarrow 1$	$\sigma_7 \leftarrow 2$	$\sigma_8 \leftarrow 1$	$\sigma_9 \leftarrow 0$	$\sigma_{10} \leftarrow 7$
	$\sigma_{11} \leftarrow 1$	$\sigma_{12} \leftarrow 2$	$\sigma_{13} \leftarrow 1$	$\sigma_{14} \leftarrow 0$	$\sigma_{15} \leftarrow 5$
Modelo de Observação	<i>Landmarks</i>	N/A			
	<i>Pontos-Chaves</i>	N/A			
	Distâncias	$\sigma \leftarrow 50cm$ $f_w \leftarrow e^{-1}$			
	Orientação	$\sigma_\varphi \leftarrow 20^\circ$ $f_w \leftarrow e^{-1}$			
Adaptação da Quantidade de Partículas	$n_{max} \leftarrow 1000$ $n_{min} \leftarrow 30$ $\mu \leftarrow 6$ $\varepsilon \leftarrow 2$				
Melhor Observação	N/A				

Fonte: Autor

Assim, quando o desvio padrão das posições das partículas for igual à 6 a quantidade de partículas será de 515. Além disso, quando o desvio padrão for menor ou igual à 2 serão utilizadas apenas 30 partículas para representar a distribuição de probabilidades, e por sua vez se o desvio padrão for maior ou igual à 10, então serão utilizadas 1000 partículas para representar a distribuição de probabilidades.

Figura 30 – Erros de orientação e posição em função do tempo do experimento com adaptação da quantidade de partículas

Fonte: Autor

O algoritmo que estima qual a observação deve gerar a melhor observação não foi utilizado neste experimento. O experimento foi executado 30 vezes. A tabela 4 apresenta a configuração usada no MCL.

Figura 31 – Valores-p ao longo do tempo, obtidos pela aplicação do teste T de *Student* comparando o experimento de referência e o experimento que varia a quantidade de partículas

Fonte: Autor

Figura 32 – Comparação entre os resultados obtidos com e sem a adaptação da quantidade de partículas

Fonte: Autor

5.4.2 Resultados

A figura 30 apresenta os erros da posição e da orientação ao longo do tempo.

Estes resultados são muito semelhantes aos obtidos no capítulo 5.1. O gráfico da figura 31 apresenta os valores-p ao longo do tempo, obtidos do teste *T* de *Student* aplicado às médias das posições estimadas pelo sistema de localização durante os experimentos de referência e este experimento. A partir dele verifica-se que não é possível rejeitar a hipótese nula de que as duas implementações apresentam o mesmo resultado. A figura 32 apresenta essa semelhança, onde a média do erro de posição da figura 25 está representada pela linha tracejada, enquanto a média do erro de posição da figura 30 está representada pela linha. Pode-se observar que a diferença ao longo do tempo é muito pequena, o que valida a utilização do desvio padrão das posições das partículas como métrica para alterar a quantidade de partículas.

Por fim, o gráfico da figura 33 apresenta a variação da quantidade de partículas em função do tempo para este experimento. Este gráfico representa os valores médios, desvios padrões, valores máximos e valores mínimos da quantidade de partículas ao longo do tempo para o conjunto de experimentos. Em média a quantidade de partículas utilizada durante o experimento foi de 450 partículas, tendo em vista que a velocidade de execução das iterações dos filtros é diretamente proporcional à quantidade de partículas este experimento utilizou em torno de 50% menos tempo para executar uma iteração do que o filtro de partículas apresentado no experimento do capítulo 5.1. O gráfico apresenta indiretamente a variação do desvio padrão

Figura 33 – Adaptação da quantidade de partículas em função do tempo

Fonte: Autor

das posições das partículas, e dessa forma pode-se observar que, ainda que as partículas estejam divergindo ao longo do tempo, o erro de posição estimada aumenta pouco.

5.5 EXPERIMENTO SIMULADO ESTIMANDO A MELHOR OBSERVAÇÃO

O objetivo deste experimento é avaliar o funcionamento do algoritmo apresentado no capítulo 4.5. Este método escolhe qual a direção que o robô deve olhar para obter a melhor informação para o processo de localização. Os resultados obtidos por este método serão comparados aos obtidos pelo experimento de referência do capítulo 5.1.

5.5.1 Configuração

As condições deste experimento são as mesmas do experimento do capítulo 5.1. Portanto, os modelos de movimento e observação são os mesmos utilizados naquele experimento. Além disso, neste experimento a quantidade de partículas é fixa.

A observação feita pelo robô, dada uma posição fixa, é relativa apenas à posição de sua câmera. Essa posição pode assumir os valores de -90° , 0° e 90° , onde 0° indica que a cabeça do robô está direcionada com a mesma orientação do robô, conforme o apresentado na figura 34. Quando o robô não utiliza o algoritmo da melhor observação para determinar qual a melhor direção para se observar, o robô faz observações nos ângulos de -90° , 0° e 90° sequencialmente e constantemente. No algoritmo 21, esses ângulos são os valores que α , β e γ podem assumir. Dessa forma, o algoritmo deve escolher entre um desses ângulos para o robô observar.

Para estimar qual direção deve ser observada para se obter a melhor observação foi utilizado o algoritmo 21. Os valores utilizados para os fatores foram:

- a) para o fator η_1 foi utilizado o valor de 1;
- b) para o fator η_2 foi utilizado o valor de $1/18$;
- c) para o fator m foi utilizado o valor de 30;
- d) para o fator Δt foi utilizado o valor de 5 segundos.

O algoritmo para adaptar a quantidade de partículas à distribuição de probabilidades não foi utilizado. O experimento utiliza 1000 partículas e foi executado 30 vezes. A tabela 5 resume a configuração do MCL.

Figura 34 – Direções que o robô pode observar

Fonte: Autor

Tabela 5 – Resumo de configuração do experimento do método de melhor observação

Quantidade de Partículas	1000				
Modelo de Movimento	$\sigma_1 \leftarrow 1$	$\sigma_2 \leftarrow 2$	$\sigma_3 \leftarrow 1$	$\sigma_4 \leftarrow 0$	$\sigma_5 \leftarrow 5$
	$\sigma_6 \leftarrow 1$	$\sigma_7 \leftarrow 2$	$\sigma_8 \leftarrow 1$	$\sigma_9 \leftarrow 0$	$\sigma_{10} \leftarrow 7$
	$\sigma_{11} \leftarrow 1$	$\sigma_{12} \leftarrow 2$	$\sigma_{13} \leftarrow 1$	$\sigma_{14} \leftarrow 0$	$\sigma_{15} \leftarrow 5$
Modelo de Observação	<i>Landmarks</i>	N/A			
	<i>Pontos-Chaves</i>	N/A			
	Distâncias	$\sigma \leftarrow 50\text{ cm}$ $f_w \leftarrow e^{-1}$			
	Orientação	$\sigma_\varphi \leftarrow 20^\circ$ $f_w \leftarrow e^{-1}$			
Adaptação da Quantidade de Partículas	N/A				
Melhor Observação	$\eta_1 \leftarrow 1$ $\eta_2 \leftarrow \frac{1}{18}$ $m \leftarrow 30$ $\Delta t \leftarrow 5s$				

Fonte: Autor

5.5.2 Resultados

Os resultados deste experimento são apresentados no gráfico da figura 35. Ele mostra que o filtro leva em torno de 5 segundos para alcançar um erro menor que 50 centímetros. Aos 10 segundos, pode-se dizer que as partículas convergiram.

Figura 35 – Erros de orientação e posição em função do tempo do experimento com o método da melhor observação

Fonte: Autor

Comparando esses resultados com os obtidos pelo experimento de referência do capítulo 5.1, pode-se perceber uma pequena melhoria na estimativa da posição do MCL. Isso é constatado ao se observar os gráficos das figuras 36 e 37, que apresenta respectivamente os valores-p ao longo do tempo, obtidos a partir da aplicação do teste T de *Student* sobre os resultados obtidos durante o experimento de referência e do experimento que utiliza a melhor observação, e os erros médios da posição estimada em ambos os experimentos. O teste T mos-

Figura 36 – Valores-p ao longo do tempo, obtidos pela aplicação do teste T de *Student* comparando o experimento de referência e o experimento que utiliza o algoritmo de melhor observação

Fonte: Autor

Figura 37 – Comparação entre os resultados obtidos utilizando o método da melhor observação e não utilizando

Fonte: Autor

tra que até os 5 segundos de execução e entre os 25 e 50 segundos é possível rejeitar a hipótese nula de que os dois experimentos são iguais, com uma certeza maior que 95%. Analisando o gráfico da figura 37 pode-se afirmar que o erro médio de posição utilizando o método da melhor observação é menor que o erro médio obtido pelo experimento de referência, nos trechos em que o teste T permite rejeitar igualdade dos erros médios.

5.6 EXPERIMENTO SIMULADO COM RECUPERAÇÃO DE ERROS DE ESTIMATIVA

O objetivo deste experimento é testar a capacidade do filtro de partículas de se recuperar do problema do sequestro. Ele utiliza a diferença entre o peso esperado e o peso da partícula para mover essa partícula de forma aleatória, conforme o apresentado no capítulo 4.1.

5.6.1 Configuração

O experimento pode ser dividido em dois trechos. No primeiro, o robô faz o caminho descrito no primeiro experimento, apresentado na figura 24. Isso é feito para observar a influencia deste método sobre o processo de localização global e rastreamento de posição.

Quando o primeiro trecho é concluído, o robô é movido para uma nova posição e executa o movimento apresentado na figura 38. Nele o robô inicia ao sul do campo, orientado para o norte. Ele anda durante 25 segundos, gira para a esquerda durante 10 segundos, volta a andar durante 25 segundos e para. Este percurso demora mais 1 minuto, que somado ao percurso anterior implica em 2 minutos de experimento.

Conforme o apresentado no capítulo 4.1, para que o filtro seja capaz de se recuperar de erros de estimativa, o modelo de movimento foi modificado, com a adição de um termo

Figura 38 – Caminho feito pelo robô após ele ser movido

Tabela 6 – Resumo de configuração do experimento de recuperação de erro de estimativa

Quantidade de Partículas	1000				
Modelo de Movimento	$\sigma_1 \leftarrow 1$	$\sigma_2 \leftarrow 2$	$\sigma_3 \leftarrow 1$	$\sigma_4 \leftarrow 500$	$\sigma_5 \leftarrow 5$
	$\sigma_6 \leftarrow 1$	$\sigma_7 \leftarrow 2$	$\sigma_8 \leftarrow 1$	$\sigma_9 \leftarrow 500$	$\sigma_{10} \leftarrow 7$
	$\sigma_{11} \leftarrow 1$	$\sigma_{12} \leftarrow 2$	$\sigma_{13} \leftarrow 1$	$\sigma_{14} \leftarrow 100$	$\sigma_{15} \leftarrow 5$
Modelo de Observação	Landmarks	N/A			
	Pontos-Chaves	N/A			
	Distâncias	$\sigma \leftarrow 50\text{ cm}$ $f_w \leftarrow e^{-1}$			
	Orientação	$\sigma_\varphi \leftarrow 20^\circ$ $f_w \leftarrow e^{-1}$			
Adaptação da Quantidade de Partículas	N/A				
Melhor Observação	N/A				

Fonte: Autor

de espalhamento que varia em função do peso da partícula. Neste caso, os fatores do modelo de movimento são idênticos aos utilizados pelos demais experimentos, exceto os fatores de espalhamento. Assim,

- para o fator σ_4 foi utilizado o valor de 500;
- para o fator σ_9 foi utilizado o valor de 500;
- e para o fator de σ_{14} foi utilizado o valor de 100.

Figura 39 – Erros de orientação e posição em função do tempo do experimento com o método de recuperação de erros de estimativa

Fonte: Autor

Esses valores foram escolhidos de maneira que quando a partícula obter fator de erro w_ϵ de 1, a partícula deverá receber um erro de movimento auto o bastante para fazer com que ela se mova bem mais que as demais partículas.

O modelo de observação utilizado foi o mesmo utilizado no primeiro experimento, que considera a distância até a borda observada para realizar a localização. O método para adaptar a quantidade de partículas em função do desvio padrão não foi utilizado neste experimento,

Figura 40 – Valores-p ao longo do tempo, obtidos pela aplicação do teste T de *Student* comparando o experimento de referência e o experimento que se recupera de erros de estimativa

Fonte: Autor

Figura 41 – Comparação entre os resultados obtidos no experimento sem recuperação de erro e no primeiro trecho do experimento com recuperação de erro

Fonte: Autor

da mesma forma, o método para estimar qual a observação deve conter a informação mais valiosa para o processo de localização não foi utilizado. Isso faz com que este experimento seja semelhante ao primeiro experimento, com a única diferença de que este é capaz de se recuperar de erros de estimativa enquanto o de referência não seria capaz.

A distribuição de probabilidades foi feita utilizando 1000 partículas. O experimento foi repetido 30 vezes. A tabela 6 apresenta um resumo da configuração deste experimento.

5.6.2 Resultados

Os resultados obtidos são apresentados na figura 39. É possível observar o instante em que a posição do robô é alterada pelo pico no gráfico aos 60 segundos. Após o robô ser movido, o filtro leva em torno de 10 segundos para convergir novamente na posição correta do robô.

Em comparação ao primeiro experimento, o método testado para recuperação de erros não afeta os processos normais de localização, no caso a localização global e o rastreamento de posição. Isso pode ser observado nas figuras 40 e 41. O primeiro apresenta os valores-p obtidos a partir da aplicação do teste T de *Student* sobre os erros médios das posições estimadas pelos experimentos, o teste T mostra que não é possível rejeitar a hipótese nula de que ambos os resultados são iguais. O segundo compara os erros médios obtidos na primeira parte deste experimento, com os obtidos no experimento do capítulo 5.1, onde a linha tracejada são os resultados obtidos no experimento de referência e a outra linha representa os resultados obtidos neste experimento.

Assim, este método não afeta o comportamento esperado do filtro de partículas, e ainda é capaz de se recuperar de erros de estimativa.

5.7 EXPERIMENTO SIMULADO COM A IMPLEMENTAÇÃO COMPLETA

O objetivo é utilizar o algoritmo 22 em sua totalidade. Desta forma, serão utilizados os métodos que foram testados separadamente nos capítulos 5.4, 5.5 e 5.6. Como modelo de observação utilizado foi o mesmo do primeiro experimento.

5.7.1 Configuração

O caminho que o robô fez durante este experimento foi o mesmo executado pelo robô no experimento anterior, que consiste em percorrer o caminho apresentado na figura 24 seguido do caminho apresentado na figura 38, desta forma testando a situação do sequestro.

Além disso a quantidade de partículas varia ao longo do tempo em função do desvio padrão das posições das partículas, conforme o experimento do capítulo 5.4. Também foi utilizado o método que estima de qual observação deve se obter a melhor informação para o processo de localização, conforme o apresentado no experimento do capítulo 5.5.

Tabela 7 – Resumo de configuração do último experimento totalmente simulado

Quantidade de Partículas	de 30 à 1000				
Modelo de Movimento	$\sigma_1 \leftarrow 1$	$\sigma_2 \leftarrow 2$	$\sigma_3 \leftarrow 1$	$\sigma_4 \leftarrow 500$	$\sigma_5 \leftarrow 5$
	$\sigma_6 \leftarrow 1$	$\sigma_7 \leftarrow 2$	$\sigma_8 \leftarrow 1$	$\sigma_9 \leftarrow 500$	$\sigma_{10} \leftarrow 7$
	$\sigma_{11} \leftarrow 1$	$\sigma_{12} \leftarrow 2$	$\sigma_{13} \leftarrow 1$	$\sigma_{14} \leftarrow 100$	$\sigma_{15} \leftarrow 5$
Modelo de Observação	<i>Landmarks</i>		N/A		
	<i>Pontos-Chaves</i>		N/A		
	Distâncias		$\sigma \leftarrow 50cm$ $f_w \leftarrow e^{-1}$		
	Orientação		$\sigma_\varphi \leftarrow 20^\circ$ $f_w \leftarrow e^{-1}$		
Adaptação da Quantidade de Partículas	$n_{max} \leftarrow 1000$ $n_{min} \leftarrow 30$ $\mu \leftarrow 6$ $\varepsilon \leftarrow 2$				
Melhor Observação	$\eta_1 \leftarrow 1$ $\eta_2 \leftarrow \frac{1}{18}$ $m \leftarrow 30$ $\Delta_t \leftarrow 5s$				

Fonte: Autor

Assim, a tabela 7 apresenta a configuração do filtro de partículas para este experimento.

Figura 42 – Erros de orientação e posição em função do tempo do último experimento totalmente simulado

Fonte: Autor

5.7.2 Resultados

Os resultados obtidos neste experimento são apresentados na figura 42. No gráfico é possível observar que o sistema se recupera do erro gerado com a mudança da posição do robô, além de apresentar o comportamento esperado do sistema de localização.

Figura 43 – Valores-p ao longo do tempo, obtidos pela aplicação do teste *T* de *Student* comparando o experimento de referência e o experimento simulado com todas as contribuições propostas

Fonte: Autor

Figura 44 – Comparação entre os resultados obtidos no experimento de referência e no primeiro trecho do experimento com a implementação completa do MCL

Fonte: Autor

Figura 45 – Quantidade de partículas utilizadas ao longo do tempo no último experimento totalmente simulado

Fonte: Autor

As figuras 43 e 44 apresentam as comparações entre a implementação do experimento de referência e este experimento, que faz a implementação completa do algoritmo proposto. A partir do teste T apresentado é possível observar que sempre que é possível rejeitar a hipótese nula de que as implementações apresentam resultados iguais. Pode-se observar que em ambos os trechos em que rejeitar a igualdade é possível, a implementação deste experimento apresenta um erro médio inferior ao apresentado no experimento de referência.

Por fim, o gráfico da figura 45 apresenta a média, o desvio padrão, a máxima e a mínima quantidade de partículas utilizada pelo MCL durante este experimento. No ponto em que ocorre a mudança de posição do robô pode-se observar que a quantidade de partículas varia muito, em alguns casos ela está em seu valor mínimo, e por vezes em seu valor máximo. Quando as partículas recebem pesos baixos elas são dispersas no domínio, fazendo com que a quantidade de partículas aumente até ao máximo. Caso alguma dessas partículas, ainda que com peso muito baixo, receba um peso muito maior que as demais, apenas ela será selecionada, fazendo com que o desvio padrão das posições tenda à 0, consequentemente a quantidade de partículas cai. Essa situação se repete até que algumas partículas se aproximem da região correspondente à posição do robô, fazendo as partículas convergirem naquela região.

A quantidade média de partículas utilizadas durante este experimento foi de 307, isso significa que em média esta implementação do MCL deve funcionar em torno de 3 vezes mais rápido do que o apresentado no capítulo 5.1.

5.8 EXPERIMENTO COM ROBÔ REAL DE LOCALIZAÇÃO GLOBAL

O objetivo deste experimento e do próximo é apresentar uma implementação em robô real do sistema de localização, além disso, pretende-se confirmar que o simulador utilizado é capaz de representar o robô real, desta forma fazendo com que os resultados obtidos nos experimentos anteriores também possam ser estendidos aos robôs reais. Este primeiro testa a capacidade do sistema de localização de estimar a posição do robô no domínio a partir de nenhuma informação *a priori*, conhecido como processo de localização global.

5.8.1 Configuração

Este experimento utiliza um robô B1, do time de futebol de robôs RoboFEI. A mecânica desse robô é baseada na mecânica do *DARwIn-OP* (ROMELA, 2017). O robô possui 49 *cm* de altura e pesa 3 *kg*. Ele possui 20 servo-motores *Dynamixel RX-28*, que geram 20 graus de liberdade para os movimentos do robô. O sensor inercial utilizado é a *UM7 Ultra-Miniature Orientation Sensor* e a câmera é uma *Logitech HD Pro Webcam C920*. O computador utilizado é um *Intel NUC Core i5-4250U, 8GB SDRAM, 120GB SSD*. E a arquitetura de software utilizada é a arquitetura em cruz (PERICO et al., 2014).

Figura 46 – Sequência de imagens do processo de classificação dos pontos

(a) Imagem original.

(b) Máscara de segmentação.

(c) Máscara da primeira transformada.

(d) Máscara da segunda transformada.

Fonte: Autor

Figura 47 – Posições do robô para o experimento

Fonte: Autor

Para este experimento considerou-se que o robô permaneceu parado, o objetivo de manter o robô parado é avaliar o funcionamento do modelo de observação proposto no capítulo 4.2, que utiliza pontos-chaves. O sistema de visão implementado utiliza pontos-chave da imagem, conforme o apresentado no capítulo 5.3. Para determinar se o ponto está dentro ou fora do campo, foi feito o seguinte método, dada uma imagem capturada pelo sistema de visão, como na figura 46a, a cor verde é segmentada, criando uma máscara sobre os pontos de cor verde, a figura 46b apresenta a sobreposição da máscara sobre a imagem original, então é aplicado uma transformação morfológica para eliminar pontos verdes fora do campo, conforme apresentado na figura 46c e, finalmente, é aplicado sobre esta máscara uma nova transformação morfológica que aumenta o tamanho das áreas pertencentes à máscara, a imagem final é apresentada na figura 46d. Note que o robô está usando uma lente olha de peixe para aumentar o campo de visão da câmera. Nessas imagens, os pontos vermelhos são os pontos-chaves escolhidos, que têm as distâncias definidas na figura 28.

Para este experimento o robô foi posicionado em 3 posições diferentes, conforme o apresentado pela figura 47. Na primeira o robô se encontra sobre a marca do pênalti, à 1.8 m da borda oeste do campo e 3 m da borda sul. Na segunda posição o robô está à 60 cm da borda oeste e 1.3 m da borda sul. E na terceira posição o robô está à 1.8 m da borda oeste e 4.7 m da borda sul.

Para este experimento, o MCL foi configurado conforme a tabela 8. O experimento foi executado utilizando o método de recuperação de erros de estimativa, o método que estima de qual observação deve ser obtida a melhor informação e a adaptação da quantidade de partículas para representar a distribuição de probabilidades.

Além disso, para as mesmas situações foram feitos experimentos em simulador, para verificar a confiabilidade dos dados, utilizando as mesmas configurações deste experimento. Cada execução foi finalizada após 15 segundos de sua inicialização.

Tabela 8 – Resumo de configuração dos experimentos com robôs reais

Quantidade de Partículas	de 30 à 1000				
Modelo de Movimento	$\sigma_1 \leftarrow 1$	$\sigma_2 \leftarrow 2$	$\sigma_3 \leftarrow 1$	$\sigma_4 \leftarrow 500$	$\sigma_5 \leftarrow 5$
	$\sigma_6 \leftarrow 1$	$\sigma_7 \leftarrow 2$	$\sigma_8 \leftarrow 1$	$\sigma_9 \leftarrow 500$	$\sigma_{10} \leftarrow 7$
	$\sigma_{11} \leftarrow 1$	$\sigma_{12} \leftarrow 2$	$\sigma_{13} \leftarrow 1$	$\sigma_{14} \leftarrow 100$	$\sigma_{15} \leftarrow 5$
Modelo de Observação	Landmarks	N/A			
	Pontos-Chaves	$\kappa_d \leftarrow 1$	$\kappa_{00} \leftarrow 1$	$\kappa_{11} \leftarrow 1$	
		$f_w \leftarrow 0.25$	$\kappa_{10} \leftarrow 0.1$	$\kappa_{01} \leftarrow 0.3$	
	Distâncias	N/A			
	Orientação	$\sigma_\varphi \leftarrow 20^\circ \quad f_w \leftarrow e^{-1}$			
Adaptação da Quantidade de Partículas	$n_{max} \leftarrow 1000 \quad n_{min} \leftarrow 30 \quad \mu \leftarrow 6 \quad \varepsilon \leftarrow 2$				
Melhor Observação	$\eta_1 \leftarrow 1 \quad \eta_2 \leftarrow \frac{1}{18} \quad m \leftarrow 30 \quad \Delta_t \leftarrow 5s$				

Fonte: Autor

5.8.2 Resultados

Os resultados são apresentados na tabela 9.

No caso do erro de posição, os valores obtidos no experimento com robôs reais e com robôs simulados são bem próximos, utilizando o teste T de *Student* para determinar se a hipótese nula de que os resultados são iguais pode ser rejeitada, obteve-se o valor-p de 0.76, ou seja, não é possível rejeitar a hipótese nula. No entanto, a orientação apresentou erros muito diferentes, ao aplicar o teste T sobre os resultados foi obtido o valor-p de 6×10^{-11} , ou seja, é possível rejeitar a hipótese nula de que os resultados obtidos quanto aos ângulos são os mesmos. Assim, o robô real apresentou um erro quase 4 vezes maior de orientação do que o robô simulado. Isso se deve ao fato do sensor de orientação real possuir erros, que não foram modelados pelo sistema simulado.

O erro do sensor de orientação é um erro acumulativo, que ocorre com uma certa frequência ao longo do tempo. Para minimizar este erro, o sensor foi reiniciado antes de cada execução do experimento.

Tabela 9 – Resultados do experimento de localização global com robô real

Experimento	Erro de Posição		Erro de Orientação	
	Média [cm]	Desvio Padrão [cm]	Média [°]	Desvio Padrão [°]
Robô Real	57.95	33.11	10.53	5.23
Robô Simulado	55.70	28.63	2.50	2.63

Fonte: Autor

5.9 EXPERIMENTO COM ROBÔ REAL DE RASTREAMENTO DE POSIÇÃO

Este experimento tem o objetivo de observar o comportamento da localização durante o rastreamento de posição de um robô real, e comparar os resultados aos obtidos pelo robô simulado.

5.9.1 Configuração

Para observar o comportamento do sistema de localização durante o rastreamento de posição do robô, o robô se move da marca do pênalti até a linha da área do gol, conforme a figura 48. Ao todo, o percurso possui 1.2 m de distância em linha reta. O experimento terminava quando o robô estava sobre a linha da área do gol. Para o robô simulado, que em média o erro de movimento é 0, a posição que ele deveria parar considerando o eixo Y deveria ser igual a inicial. Como o robô real possui um erro lateral tendencioso, para determinar a posição final do robô deve-se levar em consideração o quanto ele se deslocou em Y . Assim, ao final de cada execução do experimento foram coletados os dados das posições do robô real e do robô simulado, bem como das estimativas feitas pelo filtro de partículas em cada situação. Por fim, ao invés de iniciar o filtro com partículas uniformemente distribuídas, o filtro será iniciado a partir de uma distribuição normal com média igual a posição do robô e desvio padrão de 10 cm nos eixos X e Y , e 3° na orientação das partículas.

As demais configurações são as mesmas utilizadas no experimento anterior, e portanto a tabela 8 também vale para este experimento.

Figura 48 – Caminho percorrido pelo robô real

Tabela 10 – Resultados do experimento de rastreamento de posição com robô real

Experimento	Erro de Posição		Erro de Orientação	
	Média [<i>cm</i>]	Desvio Padrão [<i>cm</i>]	Média [°]	Desvio Padrão [°]
Robô Real	25.58	15.91	9.80	6.68
Robô Simulado	13.23	7.2	4.17	2.76

Fonte: Autor

5.9.2 Resultados

Os resultados obtidos são apresentados na tabela 10. Ao aplicar o teste T sobre os dados obtidos pelo robô real e pelo robô simulado, têm-se que o valor-p para o erro de posição foi de 4×10^{-4} , ou seja, é possível rejeitar a hipótese nula de que esses resultados são iguais. Ao aplicar o teste T sobre o erro de orientação obtêm-se o valor-p de 1×10^{-4} , desta forma também pode-se rejeitar a hipótese nula de que os resultados são iguais. Com isso, pode-se afirmar que os resultados obtidos com o robô real em movimento, são inferiores aos resultados obtidos pelo robô simulado.

Assim, ao comparar os resultados obtidos neste experimento com os obtidos no experimento anterior, conforme apresentados na tabela 9, têm-se que quando o robô está em movimento a diferença entre o robô real e o robô simulado é maior, isso é ocasionado pelos erros do robô real não modelados pelo robô simulado. Entre os erros não modelados estão o eventual erro cumulativo do sensor inercial do robô real e o erro de movimento lateral tendencioso. No entanto, os erros obtidos nesses dois experimentos para os robôs reais mostram que o sistema de localização proposto pode ser utilizado para estimar a posição de um robô real no campo.

5.10 DISCUSSÃO DOS RESULTADOS

Finalmente, a partir dos experimentos realizados pode-se concluir que este trabalho possui três contribuições para a área que são: o método para dispersão de partículas de baixa representatividade, o método que adapta a quantidade de partículas de acordo com a distribuição de probabilidades que deve ser representada e o método que estima a observação que deve gerar o melhor conjunto de informações para se realizar o processo de localização.

5.10.1 Dispersão de Partículas

O experimento do capítulo 5.6 avalia a utilização do método ao mostrar que em uma situação na qual o robô está perdido, o filtro se recupera sendo capaz de estimar novamente a posição do robô no mundo. O método implementado compara o peso da partícula com o peso máximo que seria obtido por ela. Se o peso for muito menor que o peso máximo, significa que a partícula está errada.

Se a partícula está muito errada, o erro no vetor de velocidades é tão alto que é como se uma nova posição fosse gerada aleatoriamente para a partícula. Mas se o erro é pequeno, a partícula receberá apenas os erros que ela normalmente receberia do modelo de movimento.

No entanto, o erro da partícula no instante t só influencia no movimento da partícula em $t + 1$, e entre o cálculo do peso atual e o movimento no instante futuro as partículas são selecionadas proporcionalmente aos seus pesos. Isso implica que partículas com pesos muito pequenos provavelmente não estarão em iterações futuras do filtro.

Este método começa a fazer sentido quando conjuntos de partículas recebem pesos baixos, pois a probabilidade de algumas partículas do conjunto serem selecionadas é maior. No caso extremo em que nenhuma partícula recebe um peso alto, por exemplo a situação do sequestro, onde nenhuma partícula é uma boa representação da posição do robô no mundo, todas elas serão movidas para novas posição, quase aleatórias. Cada vez que isso ocorre, existe a chance de que algumas se movam para a posição correta, fazendo o filtro convergir naquela posição.

Diferente da adição de novas partículas em posições aleatórias em toda iteração, esse método adiciona erro de acordo com a necessidade. Além disso, as restrições do domínio dificultam a utilização de heurísticas para adição de partículas em locais de maior probabilidade. Por fim, este método ainda considera que o peso máximo pode ser variável ao longo do tempo. O peso máximo deste domínio varia, pois a cada instante a quantidade de informações obtidas varia, afetando diretamente a forma como se calcula o peso.

5.10.2 Adaptação da Quantidade das Partículas

Quando se trabalha com robôs reais, deve-se considerar que os recursos computacionais devem ser divididos entre todos os módulos que mantêm o funcionamento do robô. O custo computacional de qualquer filtro de partículas está diretamente ligado à quantidade de partículas utilizadas. Desta forma, é comum adaptar a quantidade de partículas de acordo com a distribuição de probabilidades para reduzir custo computacional.

Este trabalho apresentou um método simples, que utiliza o desvio padrão das posições das partículas para determinar a quantidade de partículas. O experimento do capítulo 5.4 avaliou positivamente o método ao apresentar uma redução na quantidade de partículas utilizadas de mais de 50% durante a execução do experimento. Isso significa uma redução de custo computacional equivalente.

Este método funciona pois quanto mais as partículas estão dispersas, maior será o desvio padrão dessas partículas. Consequentemente, mais partículas são necessárias para representar a distribuição. Por outro lado, quando as partículas estão aglomeradas em torno da mesma posição, menor é a quantidade de partículas necessárias para representar a distribuição de probabilidades.

Depois que as partículas convergiram, e se tornaram poucas, pode ocorrer a situação em que o conjunto de partículas se encontre na posição errada, que caracteriza o problema do

sequestro. Os pesos que essas partículas recebem são baixos, então elas são dispersas durante a aplicação do modelo de movimento. Com isso, o desvio padrão cresce e por sua vez isso se reflete na quantidade de partículas.

As principais características deste método são: este método é eficiente, por ter custo computacional linearmente proporcional à quantidade de partículas, ele é de fácil implementação, o método necessita apenas do cálculo do desvio padrão das partículas, ou um número que o represente, e uma função que converta esse número para quantidade de partículas, ele é escalonável, de maneira que o método permanece o mesmo independente das dimensões do domínio, por exemplo, no caso deste trabalho o domínio exigia 3 dimensões, duas de posição e uma de orientação, por exemplo, o domínio de um *drone* pode utilizar domínios de 6 dimensões, tornando o método genérico.

5.10.3 Estimativa de Melhor Observação

O robô real possui um problema que não foi tratado ao longo do trabalho, que é o fato do sistema de visão não poder estar totalmente focado em obter informações para a localização. O sistema de visão precisa encontrar coisas como bola e outros robôs, enquanto busca por informações para localização. Todas essas informações são importantes para outros sistemas do robô, e por isso não é uma boa prática utilizar tempo de processamento com observações que não possuem informações que possam ser utilizadas pelo sistema de localização para melhorar a estimativa da posição.

Foi dessa necessidade que nasceu a ideia de se utilizar o valor de cada informação possível, ponderando a probabilidade de obtê-la com o quanto ela ajuda no processo de localização. Na situação abordada, os pesos das partículas foram calculados com relação à distância entre o robô e a borda do campo observada. Então para simular o valor que seria obtido pelo robô, uma partícula foi escolhida ao acaso, e sua posição foi dada como exata, então os pesos das demais partículas também escolhidas ao acaso foram calculados. Finalmente, a utilidade de cada observação era calculada a partir da quantidade de partículas perdidas no processo de reamostragem, e do quanto a câmera do robô seria deslocada para fazer a observação.

O experimento do capítulo 5.5 mostra que o método aumentou a eficiência do processo de localização, mesmo utilizando uma pequena quantidade de partículas no teste e poucas opções de observação, por apresentar que o erro de posição quando se utiliza o método é menor que o erro de posição quando não se utiliza, nas regiões onde o teste T permite rejeitar a hipótese de igualdade dos resultados. O problema do método é que ele se baseia em estimativas, que podem ser estimativas realistas ou não. Dependendo do quão fiel o modelo é do domínio real, essa estimativa se torna mais realista. O método implementado neste trabalho foi simplificado, para que fosse possível utilizá-lo, utilizando uma estimativa de qual seria a resposta do sistema de visão ao observar o domínio a partir da posição descrita pela partícula. Outro problema é que a fidelidade do modelo ao domínio pode aumentar muito o custo computacional, tornando o

método ineficiente, por exemplo, no caso dos pontos-chaves se fossem utilizados muitos pontos de observação do domínio, seria praticamente impossível executar uma iteração do método.

Esse método melhora o desempenho do sistema ao reduzir o tempo gasto no processamento de observações para obter informações que pouco auxiliam no processo de localização. A mesma ideia pode ser estendida aos demais processos, de maneira que seja possível decidir qual processo tem prioridade sobre o que o sistema de visão deve observar. Esse gerenciamento viabiliza a utilização das informações provenientes do sistema de visão por todos os módulos paralelamente, de maneira eficiente.

Assim, os métodos apresentados no capítulo 4 foram implementados e testados, e este capítulo apresentou os resultados obtidos. O próximo capítulo conclui o trabalho.

6 CONCLUSÃO

O problema da localização de robôs autônomos móveis é recorrente na literatura pois, ainda que o algoritmo seja genérico, sua implementação é dependente do domínio. Para o domínio deste trabalho, a *RoboCup Humanoid Soccer League*, as dificuldades de implementação de tal sistema surgem da maneira como o robô se move e das restrições impostas aos sensores. Para que o robô consiga se mover sem cair, ele precisa manter um ritmo de balanço, isso faz com que as informações obtidas pelos sensores sejam excessivamente ruidosas. A restrição de se utilizar apenas sensores que imitem as condições humanas trazem consigo o problema de não ser possível a obtenção precisa de distâncias.

Essas são algumas das dificuldades em lidar com a implementação em robôs reais. Outras dificuldades incluem: as limitações computacionais do robô real que deve executar paralelamente diversos processos essenciais para o seu funcionamento; os problemas de *hardware*, como folgas, peças tortas ou quebradas, superaquecimento de motores, falhas de comunicação entre o computador e os motores, entre outros. O somatório de todos os problemas tornam a implementação de um sistema de localização uma tarefa não trivial.

Além disso, a localização é dependente de sistemas adjacentes para funcionar. Estes não fazem parte do sistema de localização, mas desempenham uma função essencial para que o sistema possa funcionar. O primeiro deles é a odometria, com ela seria possível conhecer qual o deslocamento aproximado do robô e outras informações sobre o próprio robô, como por exemplo o posicionamento da câmera em relação à superfície do solo. A falta do sistema de odometria motivou a criação do modelo de movimento conforme o apresentado no capítulo 4.1. Esse sistema poderia ser usado não só como fonte de informação para a localização, mas também para fechar a malha de controle do robô.

O outro é o sistema de visão, que seria responsável por obter as informações necessárias para a localização. Entre elas, a detecção de *landmarks*. Os experimentos mostraram que para o domínio em questão é possível utilizar a detecção de *landmarks* para se obter informações do domínio. No entanto, o problema de detecção de *landmarks* neste domínio é complexo o bastante para motivar a implementação do método dos pontos-chaves do capítulo 4.2, ideia que surgiu da necessidade de conseguir informações de distância e orientação, sem a necessidade de utilizar métodos computacionalmente complexos e custosos.

Todos esses problemas tiveram de ser contornados para implementar o MCL de maneira que funcionasse para os robôs real e simulado utilizados nos experimentos. Os experimentos mostram que o sistema é funcional e que as contribuições deste trabalho são válidas.

As contribuições deste trabalho para o estado da arte do método de localização de Monte-Carlo são: a utilização dos pesos das partículas como fator para dispersar partículas com baixa representatividade, a utilização do desvio padrão das posições das partículas para determinar a quantidade de partículas necessárias para representar a distribuição de probabilidades e o método que estima qual observação deve fornecer a informação que melhora o desempenho

do sistema de localização. As duas primeiras contribuições se tornaram um artigo do *Latin American Robotics Symposium* de 2017 (ALMEIDA; COSTA; BIANCHI, 2017).

Finalmente, implementar um módulo completamente novo no sistema de um robô real é um desafio que ensina muito sobre como todos os módulos que compõem o *software* do robô devem trabalhar em conjunto para gerar um sistema funcional. Ter um módulo de localização é muito importante para o robô, é a localização que permite desenvolver estratégias em domínios com múltiplos agentes e planejar trajetórias. A localização de Monte-Carlo é um algoritmo simples, que pode ser resumido em poucas linhas. No entanto, o algoritmo é genérico o bastante para ser aplicado aos diversos domínios que tratam de robôs móveis autônomos, pois ele funciona para qualquer domínio, dado que o domínio tenha sido modelado corretamente. Por isso, o MCL é um dos métodos mais utilizados para resolver o problema da localização.

Este trabalho pode ser estendido de diversas formas em trabalhos futuros: a primeira, é desenvolver uma localização multi-agentes, que utilize as informações dos diversos robôs em campo para melhorar o processo de localização de todos; com as informações de vários robôs é possível implementar um sistema de *landmarks* dinâmicos, por exemplo, se um robô observar a bola em dada posição, um robô que não está localizado mas que está observando a bola pode utilizar essa informação para se localizar; implementar e aplicar os algoritmos apresentados em situação de jogo, de maneira que se possa observar o funcionamento do sistema em situações reais; analisar o contexto da informação antes de utilizá-la para o processo de localização, por exemplo, uma informação dentro do contexto é quando o tamanho e a posição da bola são compatíveis; utilizar informações qualitativas das relações entre os *landmarks* para realizar a localização, por exemplo o *landmark* A está à direita de B e longe do robô, isso faria que o robô se localizasse em uma região onde a afirmação fosse verdadeira; expandir o método que estima a melhor observação para os demais processos do robô e até mesmo considerar outros tipos de informações que poderiam ser obtidas do mundo; e buscar novas formas para implementar o modelo de observação utilizando algum método de aprendizado de máquina, de maneira que o robô seja capaz de se localizar após mapear o mundo.

REFERÊNCIAS

AKIN, H. Levent et al. **Cerberus'14 Team Report**. Boğaziçi University, Istanbul, 2014.

Disponível em: <<http://robot.cmpe.boun.edu.tr/~cerberus/wiki/lib/exes/fetch.php/wiki:cerberus14report.pdf>>. Acesso em: 6 mar. 2017.

ALLALI, J. et al. **Rhoban Football Club – Team Description Paper**: Humanoid Kid-Size

League, Robocup 2017 Nagoya. University of Bordeaux, Talence, 2017. Disponível em:

<https://www.robocuphumanoid.org/qualification/2017/22ee18648e39f3f656609d932ab6ccaa70a66929/Rhoban_Fooball_Club_Humanoid_KidSize_regularanddrop_in_2017_TDP.pdf>.

Acesso em: 6 mar. 2017.

ALLGEUER, Philipp; MISSURA, Marcell; BEHNKE, Sven. **NimbRo TeenSize Team**

Description 2015. Universität Bonn, Bonn, 2015. Disponível em: <https://www.robocuphumanoid.org/qualification/2015/7c2b44aadd91b0380f1d43997f0660778d13e696/NimbRo_TeenSize_Humanoid_TeenSize_2015_TDP.pdf>.

Acesso em: 6 mar. 2017.

ALMEIDA, Aislan C.; COSTA, Anna H. R.; BIANCHI, Reinaldo A. C. Vision-based

Monte-Carlo Localization for Humanoid Soccer Robots. In: IEEE. ROBOTICS Symposium (LARS) and 2017 5th Brazilian Symposium on Robotics (LARS-SBR), 2017 14th Latin American. [S.l.: s.n.], 2017.

ALY, Rami et al. **Application from Hamburg Bit-Bots for RoboCup 2017**. Universität

Hamburg, Hamburg, 2017. Disponível em: <https://www.robocuphumanoid.org/qualification/2017/0797440c3361087f5f4c382116d4fa693c02060b/Hamburg_Bit_Bots_Humanoid_KidSize_regularanddrop_in_2017_TDP.pdf>.

Acesso em: 6 mar. 2017.

AMIRI, Ahmad; NADIRI, Farzad. **Team Parand Teen Humanoid**: Team Description Paper.

Parand Islamic Azad University, Tehran, 2017. Disponível em: <https://www.robocuphumanoid.org/qualification/2017/1623408e22d5289c1f2ed49f222a362fd87ed021/Team_Parand_Humanoid_TeenSize_regularanddrop_in_2017_TDP.pdf>.

Acesso em: 6 mar. 2017.

AMOS, Matthew et al. **The NUbots Team Description Paper 2017**. The University of

Newcastle, Callaghan, 2017. Disponível em: <https://www.robocuphumanoid.org/qualification/2017/c393b713450a2537173e11cbccf2e142815a5923/NUbots_Humanoid_KidSize_regularanddrop_in_2017_TDP.pdf>.

Acesso em: 6 mar. 2017.

ANURAG, Anant et al. **AcYut TeenSize Team Description Paper 2017**. Birla Institute of

Technology and Science, Pilani, 2017. Disponível em: <<https://www.robocuphumanoid.org/>>

qualification/2017/7877fd463a6dd631006952c2cdbbf297bbd6f267/AcYut_Humanoid_TeenSize_regularanddrop_in_2017_TDP.pdf>. Acesso em: 6 mar. 2017.

ASHAR, Jayen et al. **RoboCup SPL 2014 Champion Team Paper**. University of New South Wales, Sydney, 2014. Disponível em: <<http://cgi.cse.unsw.edu.au/~robocup/2014ChampionTeamPaperReports/20141221-SPL2014ChampionTeamPaper.pdf>>. Acesso em: 6 mar. 2017.

BOLIĆ, Miodrag; DJURIĆ, Petar M; HONG, Sangjin. Resampling algorithms for particle filters: A computational complexity perspective. **EURASIP Journal on Applied Signal Processing**, Hindawi Publishing Corp., v. 2004, p. 2267–2277, 2004.

CHANG, Chun-Hua; WANG, Shao-Chen; WANG, Chieh-Chih. Exploiting moving objects: Multi-robot simultaneous localization and tracking. **IEEE Transactions on Automation Science and Engineering**, IEEE, v. 13, n. 2, p. 810–827, 2016.

COLTIN, Brian; VELOSO, Manuela. Multi-observation sensor resetting localization with ambiguous landmarks. **Autonomous Robots**, Springer, v. 35, n. 2-3, p. 221–237, 2013.

DELLAERT, Frank et al. Monte carlo localization for mobile robots. In: IEEE. **ROBOTICS and Automation**, 1999. Proceedings. 1999 IEEE International Conference on. Piscataway, NJ, USA: IEEE Service Center, 1999. v. 2, p. 1322–1328.

DIJK, Sander G. van et al. **Bold Hearts Team Description RoboCup 2015 Kid Size**. University of Hertfordshire, Hatfield, 2015. Disponível em: <https://www.robocuphumanoid.org/qualification/2015/98feb7ebd10e74ce1759499e9cc09c9f860e9e35/Bold_Hearts_Humanoid_KidSize_2015_TDP.pdf>. Acesso em: 6 mar. 2017.

DOUC, Randal; CAPPÉ, Olivier. Comparison of resampling schemes for particle filtering. In: IEEE. **IMAGE and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on**. Zagreb, Croatia: Faculty of Electrical Engineering e Computing, University of Zagreb, 2005. p. 64–69.

DOYLE, Sir Arthur Conan. **The Case-Book of Sherlock Holmes**. London, UK: John Murray, 1927. v. 1.

DRAEGERT, Jan et al. **Berlin United - FHumanoids Team Description Paper for RoboCup 2017**. Freie Universität Berlin, Berlin, 2017. Disponível em: <https://www.robocuphumanoid.org/qualification/2017/f3d45a44f4b79ff149d458ede393e5a6eb3a9560/Berlin_United_FHumanoids_Humanoid_KidSize_regularanddrop_in_2017_TDP.pdf>. Acesso em: 6 mar. 2017.

FOX, Dieter. Adapting the sample size in particle filters through KLD-sampling. **The international Journal of robotics research**, SAGE Publications, v. 22, n. 12, p. 985–1003, 2003.

FOX, Dieter et al. Monte carlo localization: Efficient position estimation for mobile robots. **AAAI/IAAI**, v. 1999, n. 343-349, p. 2–2, 1999.

GAO, Haidan et al. **SEU - UniRobot Team Description Paper**. Southeast University, Nanjing, 2017. Disponível em: <https://www.robocuphumanoid.org/qualification/2017/e3c0bf8619a019668f3330f3c4de552995174573/SEU_UniRobot_Humanoid_KidSize_regularanddrop_in_2017_TDP.pdf>. Acesso em: 6 mar. 2017.

GERNDT, Reinhard et al. Humanoid robots in soccer: Robots versus humans in RoboCup 2050. **IEEE Robotics & Automation Magazine**, IEEE, v. 22, n. 3, p. 147–154, 2015.

HAYASHIBARA, Yasuo et al. **CIT Brains (Kid Size League)**. Chiba Institute of Technology, Chiba, 2015. Disponível em: <https://www.robocuphumanoid.org/qualification/2015/548b9e3d6c18ba854563cc1b314971da53354692/CIT_Brains_Kid_Humanoid_KidSize_2015_TDP.pdf>. Acesso em: 6 mar. 2017.

HERCULANO, Davi et al. **ITAndroids Humanoid Team Description Paper for RoboCup 2017**. Instituto Tecnológico de Aeronáutica, São José dos Campos, 2017. Disponível em: <https://www.robocuphumanoid.org/qualification/2017/53f4ef65b44546bcb4711eb446f0caa402d7da16/ITAndroids_Humanoid_KidSize_regularanddrop_in_2017_TDP.pdf>. Acesso em: 6 mar. 2017.

HOL, Jeroen D; SCHON, Thomas B; GUSTAFSSON, Fredrik. On resampling algorithms for particle filters. In: IEEE. **NONLINEAR Statistical Signal Processing Workshop**, 2006 IEEE. Red Hook, NY, USA: Curran Associates, Inc., 2006. p. 79–82.

HONG, Wei; ZHOU, Changjiu; TIAN, Yantao. Robust Monte Carlo Localization for humanoid soccer robot. In: IEEE. **ADVANCED Intelligent Mechatronics**, 2009. AIM 2009. IEEE/ASME International Conference on. Red Hook, NY, USA: Curran Associates, Inc., 2009. p. 934–939.

HOSSEINI, Mojtaba et al. **Baset Adult - Size 2016 Team Description Paper**. Baset Pazhuh Tehran Company, Tehran, 2016. Disponível em: <https://www.robocuphumanoid.org/qualification/2016/18d73439714a882938e06b64fae741c06d3effb4/Baset_Adult_Size_Humanoid_AdultSize_2016_TDP.pdf>. Acesso em: 6 mar. 2017.

HUAN, Yin et al. **ZJUDancer Team Description Paper**. Zhejiang University, Hangzhou, 2016. Disponível em: <<https://www.robocuphumanoid.org/qualification/2016/10ab3f6f>

63b8a2264c8690404cfc83ad5988e9ca/ZJUDancer_Humanoid_KidSize_2016_TDP.pdf>.

Acesso em: 6 mar. 2017.

IMRAN, Muhammad Talha et al. **Team - NUST Description for RoboCup - SPL 2015 in Hefei, China**. National University of Sciences e Technology, Islamabad, 2015. Disponível em: <<http://rise.smme.nust.edu.pk/projects/RoboCupSPL/TeamNUST-TDP-2015.pdf>>. Acesso em: 6 mar. 2017.

JUÁREZ, Julio C. et al. **FALCONBOTS TEAM: Team Description Paper for Humanoid KidSize League of Robocup 2015**. Instituto Tecnológico Superior de San Martín Texmelucan, Puebla, 2015. Disponível em: <https://www.robocuphumanoid.org/qualification/2015/aeeb8a31a17fb35ad0694da822bfe5b68b9bad78/Falconbots_Humanoid_KidSize_2015_TDP.pdf>. Acesso em: 6 mar. 2017.

KARGAS, Nikolaos et al. **Kouretes 2013 SPL Team Description Paper**. Technical University of Crete, Chania, 2013. Disponível em: <<http://www.intelligence.tuc.gr/kouretes/docs/2013-kouretes-tdp.pdf>>. Acesso em: 6 mar. 2017.

KOK, Patrick de et al. **Dutch Nao Team**. Universiteit van Amsterdam, Amsterdam, 2013. Disponível em: <http://www.dutchnaoteam.nl/wp-content/uploads/2013/11/DNT_TDP_2014.pdf>. Acesso em: 6 mar. 2017.

LENSER, Scott; VELOSO, Manuela. Sensor resetting localization for poorly modelled mobile robots. In: IEEE. ROBOTICS and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on. Piscataway, NJ, USA: IEEE Service Center, 2000. v. 2, p. 1225–1232.

LI, Tiancheng; BOLIC, Miodrag; DJURIC, Petar M. Resampling methods for particle filtering: classification, implementation, and strategies. **IEEE Signal processing magazine**, IEEE, v. 32, n. 3, p. 70–86, 2015.

LIU, Jun S; CHEN, Rong; LOGVINENKO, Tanya. A theoretical framework for sequential importance sampling with resampling. In: SEQUENTIAL Monte Carlo methods in practice. New York: Springer, 2001. p. 225–246.

MELLMANN, Heinrich et al. **Berlin United - Nao Team Humboldt Team Report 2014**. Humboldt-Universität zu Berlin, Berlin, 2014. Disponível em: <<http://www.naoth.de/papers/NaoTH14Report.pdf>>. Acesso em: 6 mar. 2017.

MORALES, Marco et al. **Cuauhpiiltin 2012: Standard Platform League Team Description Paper**. Instituto Tecnológico Autónomo de México, Ciudad de México, 2012. Disponível em: <http://www.robotica.itam.mx/spl/tdp_spl_ek_robocup2012.pdf>. Acesso em: 6 mar. 2017.

NAGI, Imre; ADIPRAWITA, Widyawardana; MUTIJARSA, Kusprasapta. Vision-based Monte Carlo localization for RoboCup Humanoid Kid-Size League. In: IEEE. CONTROL Automation Robotics & Vision (ICARCV), 2014 13th International Conference on. [S.l.: s.n.], 2014. p. 1433–1438.

PERICO, Danilo H; SANTOS, Paulo E; BIANCHI, Reinaldo AC. Vision-based monte carlo localization without measurement: A qualitative approach during update phase. In: IEEE. ROBOTICS Symposium (LARS) and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR), 2015 12th Latin American. [S.l.: s.n.], 2015. p. 216–221.

PERICO, Danilo H. et al. A Robot Simulator Based on the Cross Architecture for the Development of Cognitive Robotics. In: IEEE. ROBOTICS Symposium and IV Brazilian Robotics Symposium (LARS/SBR), 2016 XIII Latin American. Red Hook, NY, USA: Curran Associates, Inc., 2016. p. 317–322.

PERICO, Danilo H. et al. Hardware and Software Aspects of the Design and Assembly of a New Humanoid Robot for RoboCup Soccer. In: ROBOTICS: SBR-LARS Robotics Symposium and Robocontrol (SBR LARS Robocontrol), 2014 Joint Conference on. [S.l.: s.n.], out. 2014. p. 73–78.

RAMEZANI, Sepehr et al. **AUTMan Humanoid Kid Size Team Description Paper:** RoboCup 2017 Humanoid Robot League, Nagoya, Japan. Amirkabir University of Technology, Tehran, 2017. Disponível em: <https://www.robocuphumanoid.org/qualification/2017/ea5f8b6675318838a231c2c930032d9b957fd29b/AUTMan_Humanoid_KidSize_regularanddrop_in_2017_TDP.pdf>. Acesso em: 6 mar. 2017.

ROBOCUP SOCCER HUMANOID LEAGUE, The Technical Comittee of. **RoboCup Soccer Humanoid League Laws of the Game 2016/2017**. Acesso em: 21 ago. 2017. 2016.

Disponível em:

<<http://www.robocuphumanoid.org/wp-content/uploads/RCHL-2017-final-2.pdf>>.

RÖFER, Thomas et al. **B-Human:** Team Report and Code Release 2016. Universität Bremen, Bremen, 2016. Disponível em: <<https://github.com/bhuman/BHumanCodeRelease/raw/master/CodeRelease2016.pdf>>. Acesso em: 6 mar. 2017.

ROMELA. **DARwIn OP:** Open Platform Humanoid Robot for Research and Education, 2017. Disponível em: <<http://www.romela.org/darwin-op-open-platform-humanoid-robot-for-research-and-education/>>. Acesso em: 21 ago. 2017.

RUSSELL, Stuart Jonathan et al. **Artificial intelligence: a modern approach**. Upper Saddle River, NJ, USA: Prentice Hall, 2010.

SOTO, Alvaro. Self adaptive particle filter. In: IJCAI, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence. [S.l.: s.n.], 2005. p. 1398–1406.

STRAKA, Ondřej; ŠIMANDL, Miroslav. Particle filter with adaptive sample size. **Kybernetika**, Institute of Information Theory e Automation AS CR, v. 47, n. 3, p. 385–400, 2011.

TASCH, Christopher et al. **WF Wolves & Taura Bots Teen Size 2015**. Ostfalia University of Applied Sciences, Wolfenbttel, 2015. Disponível em: <https://www.robocuphumanoid.org/qualification/2015/79ffabbb46e86d693e51313e6114245cb530041b/WF_Wolves_Taura_Bots_Teen_Humanoid_TeenSize_2015_TDP.pdf>. Acesso em: 6 mar. 2017.

TEIMOURI, Meisam; SALEHI, Mostafa E; MEYBODI, Mohammad Reza. A hybrid localization method for a soccer playing robot. In: IEEE. ARTIFICIAL Intelligence and Robotics (IRANOPEN), 2016. [S.l.: s.n.], 2016. p. 127–132.

TEIMOURI, Meisam et al. **MRL Team Description Paper for Humanoid KidSize League of RoboCup 2017**. Qazvin Islamic Azad University, Qazvin, 2017. Disponível em: <https://www.robocuphumanoid.org/qualification/2017/5c62316fb2be98fd90af513e68d3493d1f17b1e6/MRL_HSL_Humanoid_KidSize_regularanddrop_in_2017_TDP.pdf>. Acesso em: 6 mar. 2017.

THRUN, Sebastian; BURGARD, Wolfram; FOX, Dieter. **Probabilistic robotics**. Cambridge, MA, USA: MIT press, 2005.

TIAN, Bo et al. An intuitive and efficient switching particle filter for real-time vision-based localization. In: IEEE. CYBERNETICS and Intelligent Systems (CIS), IEEE Conference on. [S.l.: s.n.], 2013. p. 7–11.

XINGJIE, Liu et al. **Team TH - MOS**. Tsinghua University, Beijing, 2016. Disponível em: <https://www.robocuphumanoid.org/qualification/2016/e6c2a7c671b5de57ede8628b05af4f56950199a9/TH_MOS_Humanoid_KidSize_2016_TDP.pdf>. Acesso em: 6 mar. 2017.

YANIV, Shachor et al. **Team RoBIU**: Team Description Paper for Humanoid KidSize League of RoboCup 2016. Bar-Ilan University, Ramat Gan, 2016. Disponível em: <https://www.robocuphumanoid.org/qualification/2016/4229404f7c0a11a3ed8a1bb499694e2230ea685d/RoBIU_Humanoid_KidSize_2016_TDP.pdf>. Acesso em: 6 mar. 2017.

ZHANG, Lei; ZAPATA, René; LÉPINAY, Pascal. Self-adaptive Monte Carlo localization for mobile robots using range sensors. In: IEEE. INTELLIGENT Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on. Red Hook, NY, USA: Curran Associates, Inc., 2009. p. 1541–1546.