

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

MARCELO GERVAZONI CARBONERA

**NAVEGAÇÃO EM ROBÔS MÓVEIS POR ARBITRAGEM E FUSÃO
EM ARQUITETURAS COMPORTAMENTAIS**

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2021

MARCELO GERVAZONI CARBONERA

**NAVEGAÇÃO EM ROBÔS MÓVEIS POR ARBITRAGEM E FUSÃO
EM ARQUITETURAS COMPORTAMENTAIS**

Trabalho de Conclusão de Curso apresentado na disciplina de TCC1 como requisito parcial à obtenção do título de Bacharel em Engenharia de Computação, do Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná.

Orientadora: Dra. Kathya Silvia Collazos Linares

PATO BRANCO

2021

Robots of the world! The power of man has fallen! A new world has arisen: the Rule of the Robots! March! (Radius, Rossum Universal Robots, ato III)

He move in space with minimum waste and maximum joy (Sade, Smooth Operator)

LISTA DE FIGURAS

FIGURA 1	– Arbitragem e fusão de comportamentos	10
FIGURA 2	– Comportamentos definidos por campos potenciais	11
FIGURA 3	– Modelos uniciclo e diferencial	12
FIGURA 4	– Diagrama de Transição de Estado	16
FIGURA 5	– SED em malha fechada	16
FIGURA 6	– Exemplos de diagramas de transição para autômatos temporizados com guarda	17
FIGURA 7	– Controlador Híbrido	18
FIGURA 8	– Modo deslizante e diagramas de transição	19
FIGURA 9	– Representação gráfica da ação de um controle PID	20
FIGURA 10	– Diagrama de blocos para um sistema com controlador <i>Fuzzy</i>	21
FIGURA 11	– Conjuntos <i>Fuzzy</i> possíveis para a variável “altura”	22
FIGURA 12	– Operações união e intersecção em conjuntos <i>Fuzzy</i>	24
FIGURA 13	– Inferência em Lógica <i>Fuzzy</i>	25
FIGURA 14	– Defuzzificação COG	25
FIGURA 15	– Sinais de saída dos canais dos <i>encoders</i> para sentido de rotação	27
FIGURA 16	– Robôs Khepera 3 e QuickBot	30
FIGURA 17	– Robôs Khepera 3 e QuickBot em simulação	31
FIGURA 18	– Materiais e robô após montagem	33
FIGURA 19	– Resposta do sensor infravermelho	34
FIGURA 20	– Comportamento Ir para Objetivo	35
FIGURA 21	– Comportamento Evitar Obstáculo	36
FIGURA 22	– Tipos de obstáculos	38
FIGURA 23	– Comportamento Seguir Parede	39
FIGURA 24	– Recomendação vetorial	40
FIGURA 25	– Autômato Híbrido que soluciona o problema de navegação	41
FIGURA 26	– Diagrama do sistema <i>Fuzzy</i> “Evitar Obstáculo”	42
FIGURA 27	– Conjuntos <i>Fuzzy</i> para as entradas dos sensores	42
FIGURA 28	– Conjuntos <i>Fuzzy</i> para as saídas vetoriais	43
FIGURA 29	– Conjuntos <i>Fuzzy</i> para a saída de velocidade	43
FIGURA 30	– Diagrama do sistema <i>Fuzzy</i> “Seguir Parede”	44
FIGURA 31	– Curva de Ativação para comportamento Seguir Parede	45
FIGURA 32	– Conjuntos <i>Fuzzy</i> para as saídas vetoriais	45
FIGURA 33	– Diagrama do sistema <i>Fuzzy</i> “Seguir Recomendação”	45
FIGURA 34	– Conjuntos <i>Fuzzy</i> para as entradas vetoriais	46
FIGURA 35	– Conjuntos <i>Fuzzy</i> para a entrada de velocidade	47
FIGURA 36	– Conjuntos <i>Fuzzy</i> para as velocidades angulares de saída	47

LISTA DE TABELAS

TABELA 1	– Tabela de custos	28
TABELA 2	– Estados do autômato	38
TABELA 3	– Condições utilizadas nas transições do autômato	39
TABELA 4	– Regras <i>Fuzzy</i> para sistema “Evitar Obstáculo”	44
TABELA 5	– Regras <i>Fuzzy</i> para sistema “Seguir Parede”	46
TABELA 6	– Recomendações para velocidades	48

SUMÁRIO

1 INTRODUÇÃO	6
1.1 CONSIDERAÇÕES INICIAIS	6
1.1.1 Aspectos qualitativos em robótica móvel	7
1.2 OBJETIVOS	8
1.2.1 Objetivo Geral	8
1.2.2 Objetivos Específicos	8
1.3 MOTIVAÇÃO E JUSTIFICATIVA	8
1.4 ESTRUTURA	9
2 REFERENCIAL TEÓRICO	10
2.1 ARQUITETURA BASEADA EM COMPORTAMENTO	10
2.1.1 Comportamentos como campos potenciais	10
2.2 MODELAGEM	12
2.2.1 Modelo cinemático do uniciclo	12
2.2.2 Modelo cinemático do robô móvel de acionamento diferencial	13
2.3 CONSIDERAÇÕES PARA O CONTROLE DE ROBÔS MÓVEIS	13
2.4 SISTEMAS A EVENTOS DISCRETOS	15
2.4.1 Autômatos temporizados com guardas	16
2.5 SISTEMAS HÍBRIDOS	17
2.6 CONTROLADOR PID	19
2.7 LÓGICA FUZZY EM CONTROLE	21
2.7.1 Conjuntos Fuzzy	22
2.7.2 Componentes de um Sistema <i>Fuzzy</i>	23
2.7.2.1 Fuzzificação	23
2.7.2.2 Mecanismo de Inferência	24
2.7.2.3 Defuzzificação	24
2.8 ODOMETRIA	26
3 MATERIAIS E MÉTODO	28
3.1 MATERIAIS	28
3.2 METODOLOGIA	28
4 DESENVOLVIMENTO	30
4.1 O SIMULADOR SIMIAM E ALTERAÇÕES NECESSÁRIAS	30
4.1.1 Pacote “ui”	31
4.1.2 Pacote “simulator”	31
4.1.3 Pacote “robot”	32
4.1.4 Pacote “controller”	32
4.2 MONTAGEM FÍSICA DO ROBÔ	33
4.2.1 Especificações	33
4.3 DESENVOLVIMENTO DOS CONTROLADORES	33
4.3.1 Por Controlador Híbrido	34
4.3.1.1 Comportamento ”Ir Para Objetivo”	35
4.3.1.2 Comportamento ”Evitar Obstáculo”	35

4.3.1.3 Comportamento mesclado "Ir Para Objetivo e Evitar Obstáculo"	37
4.3.1.4 Mínimos locais	37
4.3.1.5 Comportamento "Seguir parede"	37
4.3.1.6 O autômato para arbitragem de comportamentos	38
4.3.2 Por Controlador <i>Fuzzy</i>	38
4.3.2.1 Comportamento "Evitar Obstáculo"	39
4.3.2.2 Comportamento "Seguir parede"	39
4.3.2.3 Comportamento "Seguir Recomendação"	39
4.3.2.4 A estratégia para fusão de comportamentos	39
4.4 SIMULAÇÃO	39
4.4.1 Utilizando controlador Híbrido	39
4.4.2 Utilizando controlador <i>Fuzzy</i>	39
4.5 MONTAGEM FÍSICA	40
4.5.1 Processo de montagem	40
4.5.2 Curvas dos motores	40
4.6 IMPLEMENTAÇÃO DO SISTEMA EMBARCADO	40
4.6.1 Considerações práticas, limitações do modelo e da simulação	40
4.6.2 Esquema lógico do sistema embarcado	40
4.6.3 Projeto da placa	40
4.6.4 Implementação da abordagem Híbrida	41
4.6.5 Implementação da solução <i>Fuzzy</i>	41
4.7 RESULTADO E DISCUSSÃO	41
4.7.1 Resultado para contrador híbrido	41
4.7.2 Resultado para contrador <i>fuzzy</i>	41
5 CONSIDERAÇÕES FINAIS	49
REFERÊNCIAS	50

1 INTRODUÇÃO

Este Capítulo apresenta o que se pretende realizar com o desenvolvimento do presente trabalho.

1.1 CONSIDERAÇÕES INICIAIS

A robótica móvel mescla aspectos das engenharias mecânica, elétrica e eletrônica, além das ciências cognitiva, social e de computação. Os objetivos da robótica móvel são voltados para aplicações diversas, como sensoriamento remoto, exploração ou operação em ambientes inóspitos, carros autônomos, além de relacionamento com humanos, em tarefas assistivas (SIEGWART; NOURBAKHS, 2004).

Na UTFPR câmpus Pato Branco, entre os trabalhos que tratam de robôs móveis, destacam-se:

- a) Bieseck (2016), que desenvolveu um robô para estacionamento automático.
- b) Petry (2016), que desenvolveu um robô seguidor de linha, comparando uma arquitetura de controle híbrida (sistema dinâmico combinado a um sistema a eventos discretos) com o controle por lógica *Fuzzy*.
- c) Lopes (2017), que desenvolveu um robô seguidor de linha, também de arquitetura híbrida.
- d) Marinho (2017), cujo trabalho aprimorou um robô de sumô (robô de competição) previamente disponível.

A execução de tarefas em robôs autônomos envolve raciocínio automatizado, percepção e controle, que incluem problemas como o de planejamento de trajetórias. De uma forma geral, esse problema consiste em encontrar um trajeto que transporte o robô de uma configuração (posição e direção) inicial até uma configuração final. A navegação é, portanto, uma importante área de estudo em robôs autônomos (MARCHI, 2001; NETO, 2007; OTTONI, 2000).

No intuito de aumentar robustez na tarefa de navegação, pode-se usar *lógica fuzzy*, que permite lidar com incertezas do mundo real (YAHMEDI; FATMI, 2011), já que ruídos podem provocar comportamentos erráticos.

A lógica *fuzzy* é um mapeamento não linear de dados de entrada em saídas escalares que permite traduzir conhecimento qualitativo (linguístico) em uma solução numérica, de modo relativamente direto (MENDEL, 1995).

1.1.1 ASPECTOS QUALITATIVOS EM ROBÓTICA MÓVEL

Conforme Mataric (2014), controle de robôs móveis pode ser dividido em controle deliberativo, reativo, híbrido (no sentido de combinar os aspectos deliberativo e reativo) e baseado em comportamento.

No controle deliberativo, deve-se planejar antes de executar qualquer ação. Essa abordagem remonta aos primeiros estudos voltados à aplicação de conceitos da Inteligência Artificial (IA) clássica ao campo da robótica. O foco desta abordagem é atender metas de longo prazo. Porém o alto custo computacional (manter representação interna do ambiente e operar buscas no modelo) traz dificuldades em alcançar requisitos de curto prazo (possibilidade iminente de colisão).

O controle reativo, por sua vez, tem como foco uma curta escala de tempo, que pode ser obtida pelo estabelecimento de um mapeamento direto entre entradas e saídas, sem representação interna do ambiente. Pode ser visto como reflexo ou instinto.

Na arquitetura híbrida, no sentido encontrado no livro de Mataric (2014), o controle deliberativo (“inteligente, porém lento”) e reativo (“rápido, porém inflexível”) são combinados de modo a extrair os pontos fortes de cada abordagem. Há a necessidade de implementar uma camada intermediária entre os dois tipos, a fim de balancear objetivos conflitantes (como escala de tempo). No restante deste texto, o termo “arquitetura híbrida” será usado apenas para designar sistemas que mesclam aspectos de controle contínuo e a eventos discretos.

Por fim, o modo baseado em comportamento surgiu do controle reativo, diferindo deste pelo nível de abstração. Enquanto comportamentos descrevem funções como “seguir objeto”, “encontrar objeto”, “seguir parede”, reações são simples ações como “andar reto” ou “virar”. Há também uma distinção na escala de tempo das ações, já que comportamentos não são instantâneos e operam ao longo do tempo. Tendo em vista que esta arquitetura será utilizada neste trabalho, maiores considerações serão tecidas no Capítulo 2.

Sobre as arquiteturas reativas e comportamentais, existe a arquitetura de subsunção, que organiza comportamentos em camadas, de modo a estabelecer uma hierarquia na qual módulos de maior importância podem inibir modulos de baixa importância. Para exemplificar, um comportamento responsável por parar o robô deve inibir modulos responsáveis por seu

movimento (MATARIC, 2014).

1.2 OBJETIVOS

O presente trabalho tem por objetivo assimilar aspectos relativos ao campo da robótica móvel. Nessa seção, pretende-se delimitar quais objetivos devem ser atendidos.

1.2.1 OBJETIVO GERAL

Construir um robô móvel de acionamento diferencial autônomo, sem armazenamento de mapa, capaz de se locomover entre duas referências, desviando de obstáculos, se necessário.

Pretende-se comparar qualitativamente soluções por controlador híbrido (sistema a eventos discretos combinado a controladores contínuos) e controlador *fuzzy*, de modo a identificar aspectos positivos e negativos de cada abordagem.

1.2.2 OBJETIVOS ESPECÍFICOS

- Levantar o modelo matemático para o robô móvel de acionamento diferencial.
- Definir uma arquitetura de controle (utilizando controle híbrido e lógica *fuzzy*).
- Construir o robô.
- Construir o sistema embarcado com a lógica de controle.
- Definir um ambiente de teste e validação para o robô.

1.3 MOTIVAÇÃO E JUSTIFICATIVA

Com os objetivos estipulados, pretende-se explorar o campo da robótica móvel em seus aspectos cruciais: navegação e inteligência.

A utilização de lógica *fuzzy* tem por intuito simplificar o processo de projeto, já que a introdução de variáveis linguísticas permite resolver o problema em questão utilizando predominantemente conhecimentos qualitativos. Deseja-se investigar tal abordagem de forma crítica, verificando qualidades e deficiências frente à contraparte híbrida, que utiliza controle baseado em comportamento associado a sistemas a eventos discretos.

1.4 ESTRUTURA

Este trabalho está organizado em cinco capítulos: Introdução, Referencial Teórico, Materiais e Método e Desenvolvimento.

As etapas do trabalho referentes a teoria de controle e lógica *fuzzy* serão discutidas no Capítulo 2 e desenvolvidas para o robô móvel no Capítulo 4.

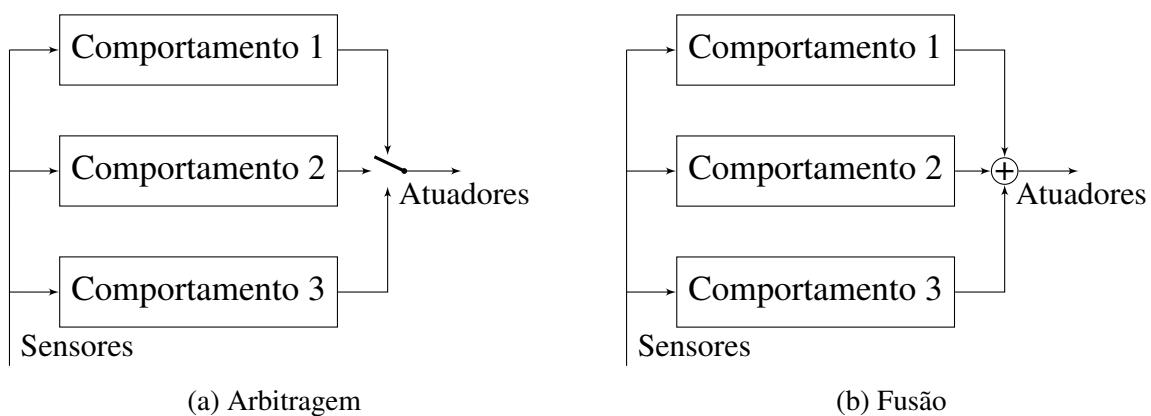
2 REFERENCIAL TEÓRICO

Este Capítulo busca explanar os principais conceitos relacionados ao tema.

2.1 ARQUITETURA BASEADA EM COMPORTAMENTO

Um dos desafios que surgem em arquiteturas comportamentais é justamente a seleção de ações tendo como base um conjunto de comportamentos (MATARIC, 2014). Duas maneiras tradicionais de fazer isso são: arbitragem e fusão. Na arbitragem, seleciona-se apenas um dos comportamentos por vez, enquanto que na fusão, os comportamentos operam em paralelo e suas saídas são combinadas. Um esquema dos dois modelos pode ser visto na Figura 1.

Figura 1 – Arbitragem e fusão de comportamentos



Fonte: baseado em Mataric (2014), p. 167.

2.1.1 COMPORTAMENTOS COMO CAMPOS POTENCIAIS

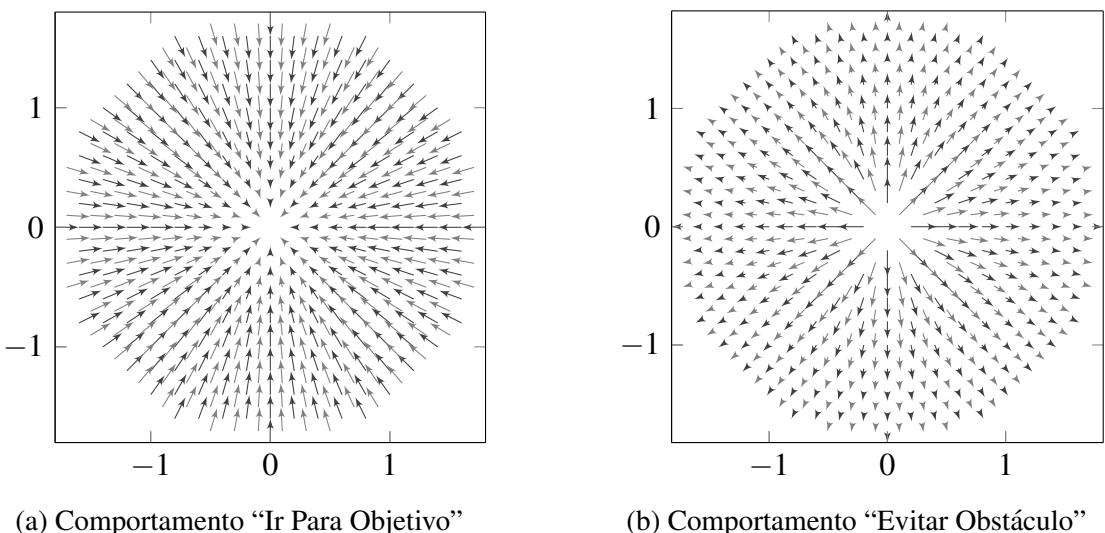
Uma forma de definir comportamentos é por meio de campos potenciais, onde o vetor gradiente determina uma direção de referência para o sistema de controle (ARKIN, 1998; YUN; TAN, 1997). Como exemplo, uma possível escolha para um comportamento de “Ir Para Objetivo” é dado na Equação 1, retratada na Figura 2.a ($R = 1$). Já o comportamento “Evitar Obstáculo” (pontual) pode ser descrito pela Equação 2, retratada na Figura 2.b ($R = 0$ e $S = 1$). Nas equações, S é uma esfera de influência, R é uma distância crítica, $\hat{\mathbf{u}}_{ipo}$ e $\hat{\mathbf{u}}_{eo}$ são vetores

unitários que apontam, respectivamente, para o objetivo e na direção oposta ao obstáculo.

$$V(x, y) = \begin{cases} \hat{\mathbf{u}}_{ipo} & , \text{para } d \geq R \\ \frac{d}{R} \hat{\mathbf{u}}_{ipo} & , \text{para } d < R \end{cases} \quad (1)$$

$$V(x, y) = \begin{cases} [0 \ 0]^T & , \text{para } d > S \\ \frac{S-d}{S-R} \hat{\mathbf{u}}_{eo} & , \text{para } R < d \leq S \\ \hat{\mathbf{u}}_{eo} & , \text{para } d \leq R \end{cases} \quad (2)$$

Figura 2 – Comportamentos definidos por campos potenciais



Fonte: baseado em Arkin (1998), p. 146 e 148

Em um robô real, cada sensor é associado a uma distância até um obstáculo (mesmo que infinitamente distante) e, consequentemente, a um vetor definido pelo comportamento “Evitar Obstáculo”. Uma combinação linear entre estes vetores cria um obstáculo virtual pontual a partir de obstáculos não pontuais (YUN; TAN, 1997).

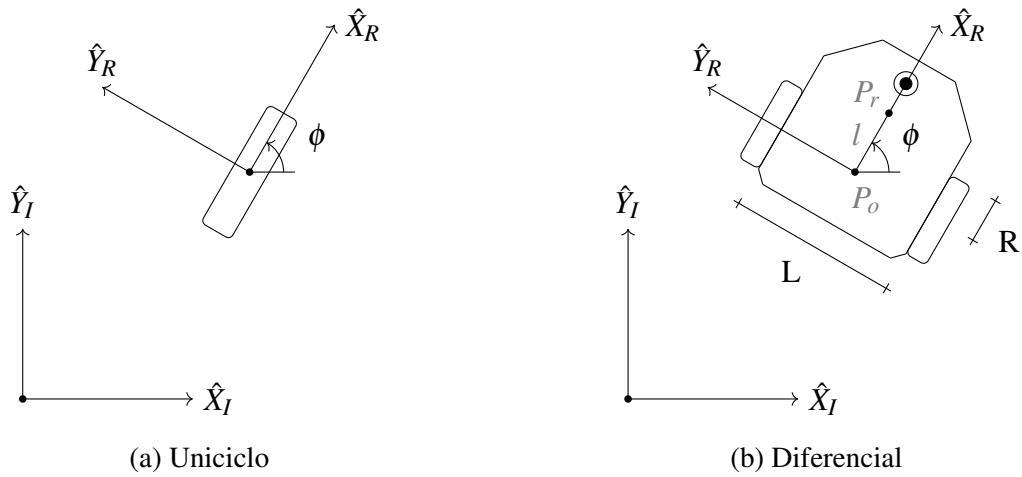
Vetores associados a comportamentos distintos podem ser combinados (linearmente) a fim de obter uma ação intermediária (Fusão, retratada na Figura 1.b), ou suas magnitudes são comparadas de modo que o “vencedor” assume controle pleno (Arbitragem, retratada na Figura 1.a).

A estratégia de utilizar campos potenciais tem suas particularidades, já que podem existir mínimos locais, o que leva o robô a parar antes de chegar ao objetivo. Para resolver este problema, deve-se, ou estabelecer um campo potencial sem mínimos locais, ou desenvolver métodos para escapar deles (YUN; TAN, 1997). Yun e Tan (1997) apresentam uma solução do segundo tipo, ao definir um comportamento “Seguidor de Parede”.

2.2 MODELAGEM

Nesta seção serão levantados os modelos cinemáticos para o robô móvel de acionamento diferencial e uniciclo. A seguir, será feita uma relação entre os dois, que permitirá realizar o controle do robô diferencial a partir do uniciclo, tornando mais simples o projeto. Os esquemas dos dois modelos pode ser visto na Figura 3.

Figura 3 – Modelos uniciclo e diferencial



Fonte: baseado em Siegwart e Nourbakhsh (2004), p. 49 e 54.

2.2.1 MODELO CINEMÁTICO DO UNICICLO

O modelo uniciclo representa uma única roda que se movimenta em uma superfície sem deslizamento. Considera-se que existe acesso direto às velocidades linear e angular.

O modelo cinemático pode ser visto na Equação 3 (LAVALLE, 2006). A Equação 4 representa o uniciclo na forma matricial (SIEGWART; NOURBAKHSH, 2004).

$$\begin{cases} \dot{x} = v \cos \phi \\ \dot{y} = v \sin \phi \\ \dot{\phi} = \omega \end{cases} \quad (3)$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos \phi & 0 \\ \sin \phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (4)$$

2.2.2 MODELO CINEMÁTICO DO ROBÔ MÓVEL DE ACIONAMENTO DIFERENCIAL

Neste modelo (Equação 5), são consideradas como entradas as velocidades angulares das rodas esquerda e direita (LAVALLE, 2006).

$$\begin{cases} \dot{x} = \frac{R}{2}(\omega_l + \omega_r) \cos \phi \\ \dot{y} = \frac{R}{2}(\omega_l + \omega_r) \sin \phi \\ \dot{\phi} = \frac{R}{L}(\omega_r - \omega_l) \end{cases} \quad (5)$$

O robô móvel de acionamento diferencial, arquitetura usada neste trabalho, é cinematicamente equivalente ao uniciclo (DIB, 2011). Isso pode ser mostrado igualando \dot{x} , \dot{y} e $\dot{\phi}$ das Equações 3 e 5. Resolvendo para ω_l e ω_r , tem-se a igualdade da Equação 6.

$$\begin{aligned} \omega_l &= \frac{2v - L\omega}{2R} \\ \omega_r &= \frac{2v + L\omega}{2R} \end{aligned} \quad (6)$$

Essa equivalência entre modelos é possível pois eles não consideram a dinâmica do sistema (LAVALLE, 2006).

2.3 CONSIDERAÇÕES PARA O CONTROLE DE ROBÔS MÓVEIS

O sistema uniciclo tem duas restrições ditas não holonômicas, o que significa que não podem ser integradas para obter uma restrição geométrica (ORIOLO, 2013). Na Equação 7, a primeira se refere à ausência de deslizamento lateral, enquanto a segunda é relativa à condição de rotação pura (ausência de deslizamento no sentido do vetor velocidade) (JAZAR, 2011, pg. 955).

$$\begin{aligned} \dot{x} \sin \phi - \dot{y} \cos \phi &= 0 \\ \dot{x} \cos \phi + \dot{y} \sin \phi &= v \end{aligned} \quad (7)$$

Dois tipos de problemas em controle de robôs móveis podem ser explicitados: estabilização em um determinado estado (configuração) e seguir trajetória (*tracking*). Pela própria natureza do sistema não holonômico, o primeiro problema é considerado mais difícil (DIB, 2011).

No caso da estabilização, o teorema de Brockett afirma que, para sistemas subatuados, com menos entradas que variáveis de estado, o sistema não pode ser estabilizado assintoticamente (exponencialmente) usando leis de controle por realimentação de estado contínuas e invariantes no tempo (ASTOLFI, 1995; BANAVAR; SANKARANARAYANAN, 2006). Para solucionar esse problema, técnicas de controle não linear têm sido exploradas,

envolvendo controladores descontínuos e/ou variantes no tempo (DIB, 2011).

O fato do robô movel ser de estado completamente controlável apesar de subatuado é devido à presença de restrições não holonômicas (ORIOLO, 2013).

Para o problema de seguir trajetória (*tracking*), o objetivo é seguir uma curva parametrizada em função do tempo (CARONA et al., 2008).

Em Glotfelter e Egerstedt (2018), uma técnica é explorada a fim de tratar o modelo diferencial pelo modelo de integrador-único (ponto controlado por velocidade, ou, $\dot{\mathbf{x}} = \mathbf{u}$). Os autores utilizam como referência um ponto deslocado em relação ao ponto no centro entre as duas rodas. A Figura 3.b indica a posição deste ponto (P_r) e a Equação 8 mostra os valores de x e y para a nova referência. Derivando e substituindo \dot{x} , \dot{y} e $\dot{\phi}$ da Equação 3, obtém-se a Equação 9.

$$\begin{cases} x' = x_o + l \cos \phi \\ y' = y_o + l \sin \phi \end{cases} \quad (8)$$

$$\begin{cases} \dot{x}' = \dot{x}_o - \dot{\phi}l \sin \phi = v \cos \phi - l \omega \sin \phi \\ \dot{y}' = \dot{y}_o + \dot{\phi}l \cos \phi = v \sin \phi + l \omega \cos \phi \end{cases} \quad (9)$$

A Equação 10 é idêntica à 9, porém, na forma matricial. Sua inversa estabelece um mapeamento direto entre as entradas $[\dot{x} \ \dot{y}]^T$ do modelo de integrador-único para as entradas do modelo uniciclo (Equação 11), ou modelo diferencial (Equação 12).

$$\begin{bmatrix} \dot{x}' \\ \dot{y}' \end{bmatrix} = \begin{bmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & l \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (10)$$

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{l} \end{bmatrix} \begin{bmatrix} \cos \phi & \sin \phi \\ -\sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} \dot{x}' \\ \dot{y}' \end{bmatrix} \quad (11)$$

$$\begin{bmatrix} \omega_l \\ \omega_r \end{bmatrix} = \begin{bmatrix} \frac{1}{R} & \frac{-L}{2R} \\ \frac{1}{R} & \frac{L}{2R} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{l} \end{bmatrix} \begin{bmatrix} \cos \phi & \sin \phi \\ -\sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} \dot{x}' \\ \dot{y}' \end{bmatrix} \quad (12)$$

Pelo que pode ser verificado em Glotfelter e Egerstedt (2018), os modelos uniciclo e de integrador-único não são equivalentes, já que l não pode tender a zero. Um controle de modelo preditivo é estabelecido para ajustar l a fim de obter uma resposta que sacrifique o mínimo possível em precisão e manobrabilidade, comparando com uma escolha estática de parâmetro ($l = \sqrt{\frac{1-\alpha}{\alpha} \frac{Lv_{max}}{R}}$, com $\alpha = 0,99$).

Esta técnica, também chamada “*look-ahead*”, é utilizada, com modificações, em alguns trabalhos como passo intermediário antes de realizar uma linearização por realimentação

(YUN; YAMAMOTO, 1992; NOVEL et al., 1995). Em Yun e Yamamoto (1992) é mostrado que o modelo dinâmico não pode ser linearizado por realimentação entrada-estado. A linearização por realimentação entrada-saída estática só é possível adotando esta nova referência.

Apesar de em Yun e Yamamoto (1992) a dinâmica do sistema ser considerada, em Novel et al. (1995) é dito que um modelo cinemático pode ser usado. Ao considerar a dinâmica (primeiro trabalho) uma realimentação é definida de forma a compensar não linearidades do sistema, a fim de tratá-lo como um modelo de duplo integrador ($\ddot{\mathbf{x}} = \mathbf{u}$). A partir disso, o sistema pode ser controlado por técnicas de controle linear.

Para o modelo cinemático deste trabalho, a definição de um campo potencial não é tão conveniente, já que o negativo do gradiente associado especifica força, de modo que a aceleração é obtida considerando massa (YUN; TAN, 1997; HALLIDAY et al., 2012). Em Kwon et al. (2009), o campo vetorial definido indica velocidades. Os vetores velocidade calculados podem ser usados como entrada do modelo de integrador único.

Apesar do modelo não ser controlável por técnicas lineares, é possível utilizar controle PID para seguir ângulo, com a limitação de não ter controle sobre velocidades linear ou angular. Para aplicações onde a convergência não tem restrições temporais significativas, controle angular é suficiente e será utilizado no caso deste trabalho.

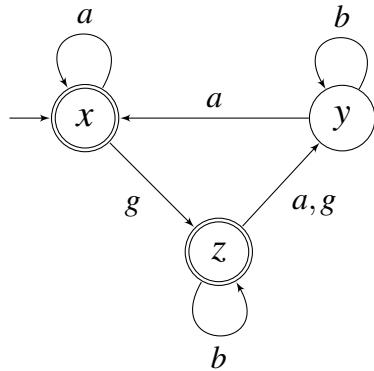
2.4 SISTEMAS A EVENTOS DISCRETOS

Um Sistema a Eventos Discretos (SED) pode ser caracterizado como um sistema dinâmico, cujos estados são discretos, sendo que transições entre estes estados ocorrem a partir da detecção de eventos assíncronos (CASSANDRAS; LAFORTUNE, 2008). Um evento pode ser definido como um estímulo que provoca transições instantâneas de estado. Na ausência de eventos, o sistema permanece no mesmo estado (CURY, 2001).

Um SED pode ser modelado usando o formalismo presente em teoria de computação: linguagens regulares e autômatos finitos. O autômato finito determinístico é uma tupla $M = (Q, \Sigma, \delta, q, F)$ formada por um conjunto de estados (Q), um conjunto de símbolos (Σ), uma função de transição (δ), um estado inicial (q), e um conjunto de estados finais (F , que é subconjunto de Q). (CASSANDRAS; LAFORTUNE, 2008; MAHESHWARI; SMID, 2019).

O conjunto de eventos admissíveis em um SED é o alfabeto de uma linguagem e um conjunto de eventos sucessivos seriam palavras desta linguagem. O autômato representa uma linguagem formal. Uma forma de representar o autômato é por meio de um diagrama de transição de estado. Um exemplo pode ser visto na Figura 4.

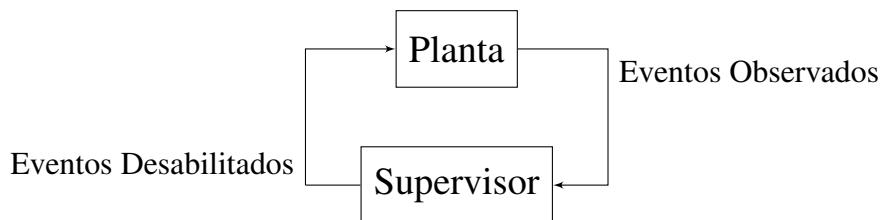
Figura 4 – Diagrama de Transição de Estado



Fonte: baseado em Cassandras e Lafortune (2008), p. 60.

O SED possui um comportamento, ou dinâmica, que pretende-se controlar, já que algumas cadeias de eventos aceitas pela linguagem podem não ser desejadas. A estrutura ou agente responsável pelo controle é chamado supervisor, que interage com a planta capturando eventos ocorridos (apenas aqueles que são observáveis) e atuando de forma a permitir ou suprimir eventos passíveis de ocorrência (CASSANDRAS; LAFORTUNE, 2008). Um esquema desta interação pode ser visto na Figura 5.

Figura 5 – SED em malha fechada



Fonte: baseado em Cury (2001), p. 49.

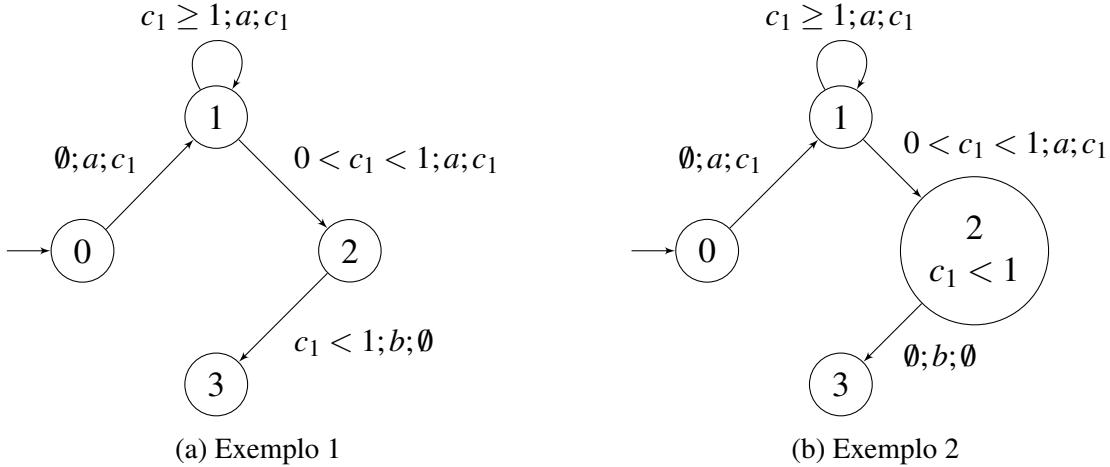
2.4.1 AUTÔMATOS TEMPORIZADOS COM GUARDAS

Uma representação mais completa é o autômato temporizado com guardas, que possui um conjunto de variáveis contínuas e em função do tempo denominadas *clocks*, cuja dinâmica é linear. O temporizador pode ser usado para estabelecer condições que provocam transições. Durante essas transições, a variável temporizada pode sofrer *reset* (onde o valor 0 é atribuído) (CASSANDRAS; LAFORTUNE, 2008).

Transições temporizadas são tuplas da forma (guardas; eventos; *reset*), onde “guardas” é um conjunto de condições em variáveis do tipo *clock* que devem ser atendidas para ocorrência de transição, os eventos são componentes já definidos e *reset* especifica um conjunto de variáveis do tipo *clock* às quais deve ser atribuído valor 0. Quando o conjunto “guardas” é vazio

\emptyset , considera-se valor lógico verdadeiro e quando o conjunto *reset* é vazio, nenhuma variável *clock* sofre atribuição de valor 0 (CASSANDRAS; LAFORTUNE, 2008). Dois exemplo de diagramas de transição são mostrados na Figura 6.

Figura 6 – Exemplos de diagramas de transição para autômatos temporizados com guarda



Fonte: baseado em Cassandras e Lafourture (2008), p. 301.

Em casos nos quais a condição de guarda deixa de ser válida, como, por exemplo, quando no estado 2 da Figura 6.a, c_1 se torna maior ou igual a 1, ocorre *deadlock* por tempo (*timelock*). Quando é necessário forçar a ocorrência de um evento para evitar *timelock*, modela-se como na Figura 6.b. A condição associada a um estado é chamada Invariante (CASSANDRAS; LAFORTUNE, 2008).

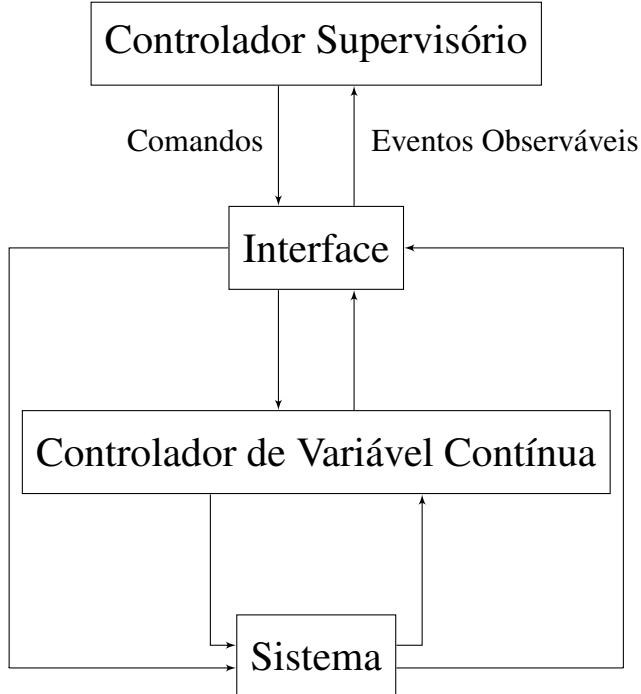
Por fim, o autômato pode ser representado como uma tupla (Q, E, C, Tra, Inv, q_0) , onde Q é um conjunto de estados, E é um conjunto de eventos, C é um conjunto de temporizadores (variáveis contínuas), “*Tra*” é um conjunto de transições temporizadas, “*Inv*” é um conjunto de invariantes e q_0 é o estado inicial (CASSANDRAS; LAFORTUNE, 2008).

2.5 SISTEMAS HÍBRIDOS

Sistemas híbridos são formados quando Sistemas Dinâmicos de Variável Contínua (SDVC) e Sistemas a Eventos Discretos (SED) funcionam de maneira conjunta. Neste caso, controladores de variável contínua são abstraídos por um SED. Transições de estado provocam transições ou “chaveamento” entre controladores (CASSANDRAS; LAFORTUNE, 2008). A Figura 7 mostra o esquema da arquitetura de controle de um sistema híbrido.

Neste sentido, é necessário introduzir o conceito de autômatos híbridos. Ao invés de estados discretos, os estados do sistema são dados por (q, \mathbf{x}) , com $q \in Q$, onde Q é um

Figura 7 – Controlador Híbrido



Fonte: baseado em Cassandras e Lafortune (2008), p. 43.

conjunto de estados discretos e $\mathbf{x} \in X$, onde X é um conjunto de estados contínuos. O estado q determina o modo de operação do sistema. O autômato híbrido é, portanto, uma extensão do autômato temporizado com guardas, já que as funções dos temporizadores (lineares) podem ser substituídas por funções arbitrárias (CASSANDRAS; LAFORTUNE, 2008).

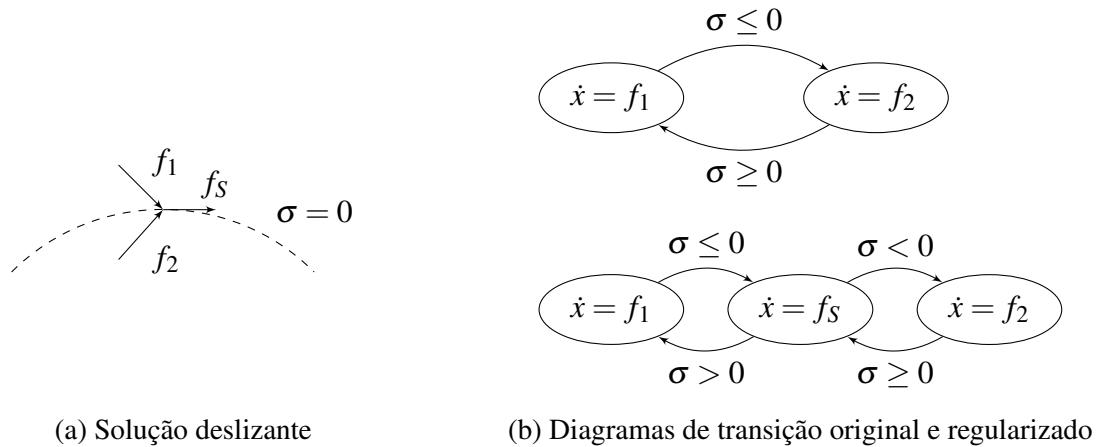
Esse novo autômato pode ser representado pela tupla $G_h = (Q, X, E, U, f, \phi, Inv, guard, \rho, q_0, \mathbf{x}_0)$, onde Q é um conjunto de estados discretos, X é um conjunto de variáveis contínuas (o espaço de estados $X \subseteq \mathbb{R}^n$), E é um conjunto de eventos, U é um conjunto de entradas controláveis ($U \subseteq \mathbb{R}^m$), f é um campo vetorial ($f : Q \times X \times U \rightarrow X$), ϕ é uma função de transição de estados discretos ($\phi : Q \times X \times E \rightarrow Q$), “Inv” é um conjunto de invariantes, ou domínio ($Inv \subseteq Q \times X$), “guard” é um conjunto de condições de guarda ($guard \subseteq Q \times Q \times X$), ρ é uma função *reset* ($\rho : Q \times Q \times X \times E \rightarrow X$) cujo objetivo é alterar \mathbf{x} quando ocorre mudança de modo (q), q_0 é o estado discreto inicial (modo de operação) e \mathbf{x}_0 é o estado contínuo inicial (CASSANDRAS; LAFORTUNE, 2008, pg. 316).

Em robótica móvel, autômatos fornecem um meio de implementar a arbitragem requisitada pela arquitetura baseada em comportamentos, onde cada modo de operação corresponde a um comportamento. Porém, o uso de autômatos híbridos traz uma particularidade potencialmente problemática: o comportamento de Zenão, no qual, teoricamente, transições infinitas de modo ocorrem em tempo finito, o que bloqueia a passagem de tempo (EGERSTEDT,

2000).

Isso ocorre quando o campo vetorial descontínuo subjacente a um sistema híbrido converge para uma superfície de separação entre modos, o que é chamado de deslize. Essa limitação do autômato, no ambiente real, provoca oscilações, já que mudanças de controladores ocorrem em curto espaço de tempo. Uma possível solução é por meio do processo de regularização, que na prática adiciona um estado de forma a implementar uma “dinâmica de deslize” sobre a fronteira que separa comportamentos (EGERSTEDT, 2000). Este conceito está ilustrado na Figura 8.

Figura 8 – Modo deslizante e diagramas de transição



Fonte: baseado em Egerstedt (2000)

O termo “comportamento de Zenão” faz alusão ao filósofo grego Zenão de Eleia, conhecido pelos seus famosos paradoxos (EGERSTEDT, 2000).

2.6 CONTROLADOR PID

O controlador PID é um tipo de controle por realimentação no qual a ação de controle é constituída por três termos: proporcional (P), integral (I) e derivativo (D) (JOHAN; MURRAY, 2021).

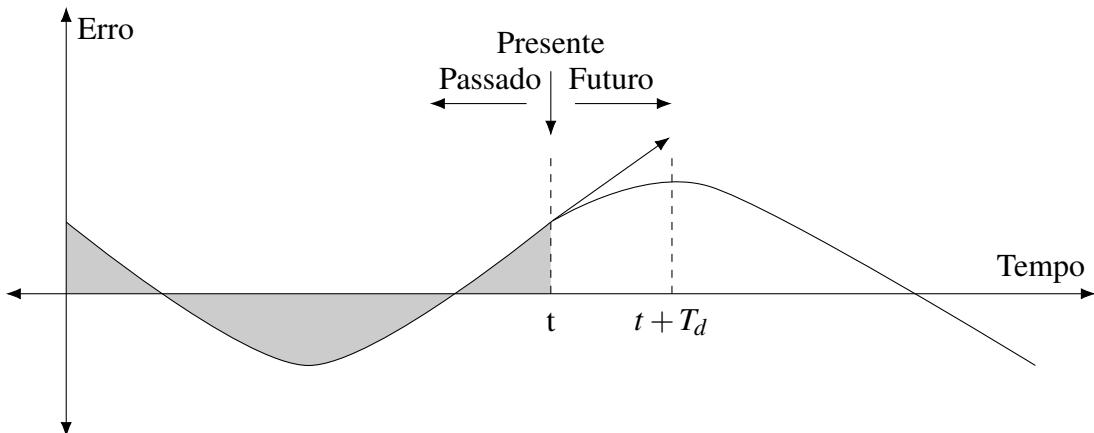
A Equação 13 mostra o cálculo da ação de controle, onde k_p , k_i e k_d são constantes dos termos proporcional, integral e derivativo, respectivamente. O termo $e(t)$ é o erro da variável a ser controlada (JOHAN; MURRAY, 2021).

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau) d\tau + k_d \frac{de(t)}{dt} \quad (13)$$

A Figura 9 mostra uma representação gráfica da ação do controlador. O termo

proporcional considera o erro atual e, portanto, leva em conta o presente. O termo integral considera a soma dos erros e por isso leva em conta o passado. O termo derivativo projeta a tendência do sinal de controle e, consequentemente, leva em conta uma espécie de previsão do futuro. As constantes retratam o grau de importância do termo para a composição do sinal de controle (JOHAN; MURRAY, 2021).

Figura 9 – Representação gráfica da ação de um controle PID



Fonte: baseado em Johan e Murray (2021), pg. 20

A Equação 13 representa a ação de controle em tempo contínuo. A equação em tempo discreto é necessária para implementação computacional. O termo integral é calculado por integração numérica e o termo derivativo por equação de diferenças. A Equação 14 representa a discretização do controlador PID, onde T_d é o período da discretização (JOHAN; MURRAY, 2021).

$$u(k) = k_p e(k) + k_i T_d \sum_0^k e(k) + k_d \frac{e(k) - e(k-1)}{T_d} \quad (14)$$

Duas considerações práticas são muito importantes em robôs móveis. O *Windup* é uma não-linearidade que ocorre quando a saída do controlador é maior que a capacidade dos atuadores, o que leva à saturação e, consequentemente, degradação de performance. A zona morta é outra não-linearidade que ocorre quando a saída do controlador é muito pequena e o atuador, por conta de forças de atrito, é incapaz de responder ao sinal de controle (JOHAN; MURRAY, 2021).

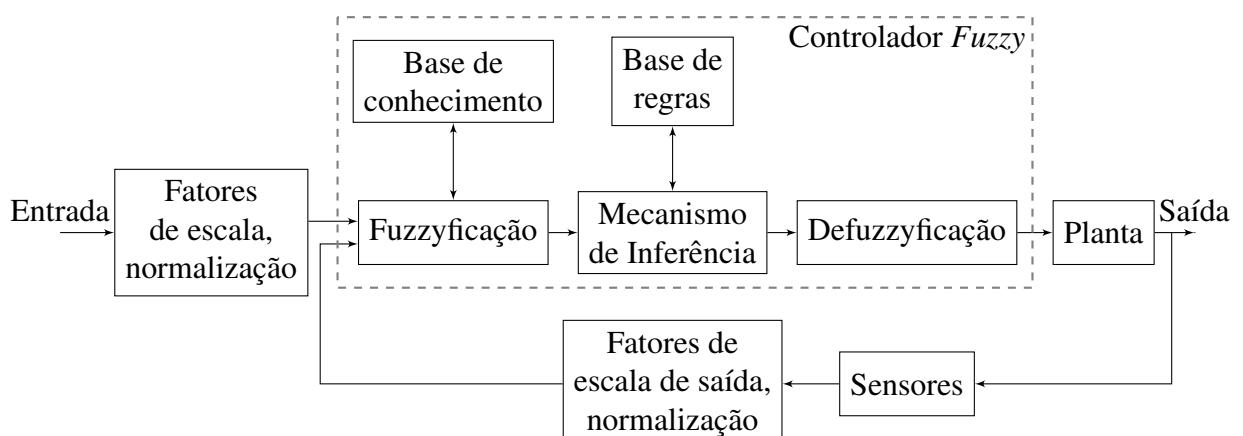
No caso do *Windup*, o robô perde capacidade de cumprir requisitos de velocidade angular. É importante, após o cálculo da atuação, verificar a ocorrência de saturação e, se necessário, sacrificar o requisito de velocidade linear a fim de cumprir o requisito de velocidade angular.

2.7 LÓGICA FUZZY EM CONTROLE

Em certas circunstâncias, a lógica Booleana pode não ser suficiente para descrever conhecimentos qualitativos com precisão. Como exemplo, ao determinar como “alto” todo indivíduo com altura maior que 1,70 metros, pessoas com 1,71 ou 2,00 metros seriam consideradas igualmente altas. A lógica *Fuzzy* acrescenta a informação de “pertinência”, que atribui um “grau de certeza” de um valor (neste exemplo, altura) em relação à classe avaliada (categoria “alto”) (LILLY, 2011).

Sistemas *Fuzzy* tem por objetivo mesclar aspectos exatos com conhecimentos imprecisos que caracterizam o pensamento humano. Deste modo, aspectos qualitativos de um problema, ou conhecimento especializado, são mapeados e utilizados em um sistema *Fuzzy* (LILLY, 2011). Um diagrama de blocos de um controlador *Fuzzy* e seus componentes atuando em um sistema pode ser visto na Figura 10.

Figura 10 – Diagrama de blocos para um sistema com controlador *Fuzzy*



Fonte: baseado em Ross (2009), p. 442 e Lilly (2011), p. 29.

Os sistemas *Fuzzy*, de acordo com Ross (2009), são úteis em contextos onde é necessário lidar com sistemas muito complexos, compreendidos parcialmente, além de casos nos quais soluções rápidas são desejadas, mesmo que aproximadas.

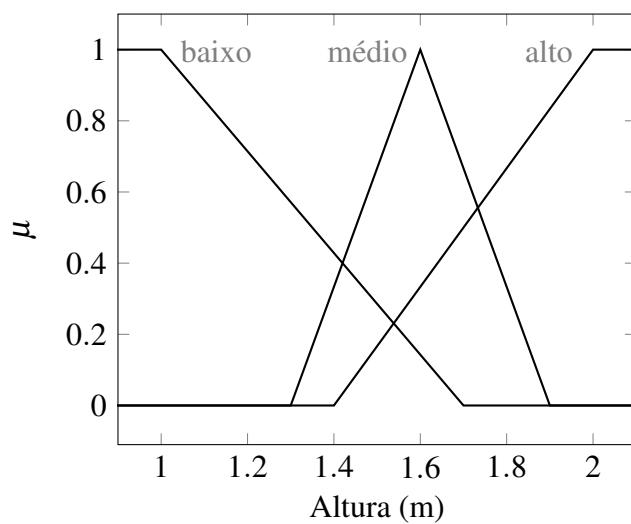
Já Lilly (2011) afirma que problemas não lineares, de difícil solução por controle clássico, são exemplos de situações nas quais sistemas *Fuzzy* podem ser utilizados. Entretanto, o autor alerta que, por conta do alto custo computacional desta solução, ela não é indicada para casos envolvendo problemas lineares e invariantes no tempo, nos quais soluções mais diretas estão disponíveis, por exemplo, controle PID e técnica de alocação de pólos.

2.7.1 CONJUNTOS FUZZY

No exemplo dado anteriormente, uma proposição da lógica booleana foi usada para definir um conjunto dos indivíduos “altos”. Este conjunto (convencional) é chamado “crisp”. Em conjuntos *Fuzzy*, associa-se a cada membro de um conjunto um valor real entre 0 e 1 chamado “grau de pertinência”, onde 0 representa exclusão absoluta, 1 representa pertinência total e qualquer valor intermediário representa pertinência parcial (LILLY, 2011).

Os valores numéricos possíveis para altura são chamados “universo de discurso”, o valor “alto” é dito ser um “valor linguístico” associado a “comprimento”, que é a variável linguística no exemplo dado. Para ilustrar o conceito, funções de pertinência associadas à variável linguística “altura” podem ser vistas na Figura 11, onde μ é o grau de pertinência. Assim, valores de altura podem pertencer a mais de uma classe, porém com diferentes graus de pertinência (LILLY, 2011).

Figura 11 – Conjuntos *Fuzzy* possíveis para a variável “altura”



Fonte: baseado em Lilly (2011), p. 32.

As definições conceituais em teoria de conjuntos são diferentes para conjuntos *Fuzzy*. As mais importantes, explicitadas em Lilly (2011), estão reunidas a seguir, onde $\mu_1(x)$ e $\mu_2(x)$ são graus de pertinência relacionados aos conjuntos *Fuzzy* M_1 e M_2 , respectivamente, definidas para uma mesma variável x no universo de discurso χ .

- Subconjunto: M_1 é subconjunto *Fuzzy* de M_2 ($M_1 \subseteq M_2$) se $\mu_1 \leq \mu_2$, $\forall x \in \chi$.
- Complemento: o complemento *Fuzzy* de M (\overline{M}) é dado por $\mu_{\overline{M}}(x) = 1 - \mu_M(x)$.
- Intersecção (AND):

Pode ser usada qualquer função que cumpra os requisitos:

1. Um elemento do universo não pode pertencer a dois conjuntos *Fuzzy* com maior grau de pertinência que a cada conjunto individualmente.
2. Se um elemento não pertence a um dos conjuntos, não pode pertencer à intersecção.
3. Se um elemento pertence aos dois conjuntos *Fuzzy* com pertinência total, então pertence à intersecção com pertinência total.

Duas funções que cumprem os requisitos são: $\mu_{M_1 \cap M_2}(x) = \min\{\mu_1(x), \mu_2(x) : x \in \chi\}$ (função mínimo) e $\mu_{M_1 \cap M_2}(x) = \{\mu_1(x)\mu_2(x) : x \in \chi\}$ (produto algébrico).

- União (OR):

Pode ser usada qualquer função que cumpra os requisitos:

1. Um elemento do universo não pode pertencer à união de dois conjuntos *Fuzzy* com menor grau de pertinência que a cada conjunto individualmente.
2. Se um elemento pertence a um dos conjuntos, então deve pertencer à união.
3. Se um elemento não pertence a nenhum dos dois conjuntos *Fuzzy*, então não pertence à união.

Duas funções que cumprem o requisito são: $\mu_{M_1 \cup M_2}(x) = \max\{\mu_1(x), \mu_2(x) : x \in \chi\}$ (função máximo) e $\mu_{M_1 \cup M_2}(x) = \{\mu_1(x) + \mu_2(x) - \mu_1(x)\mu_2(x) : x \in \chi\}$ (soma algébrica).

Um exemplo das operações união e intersecção pode ser visto na figura 12.

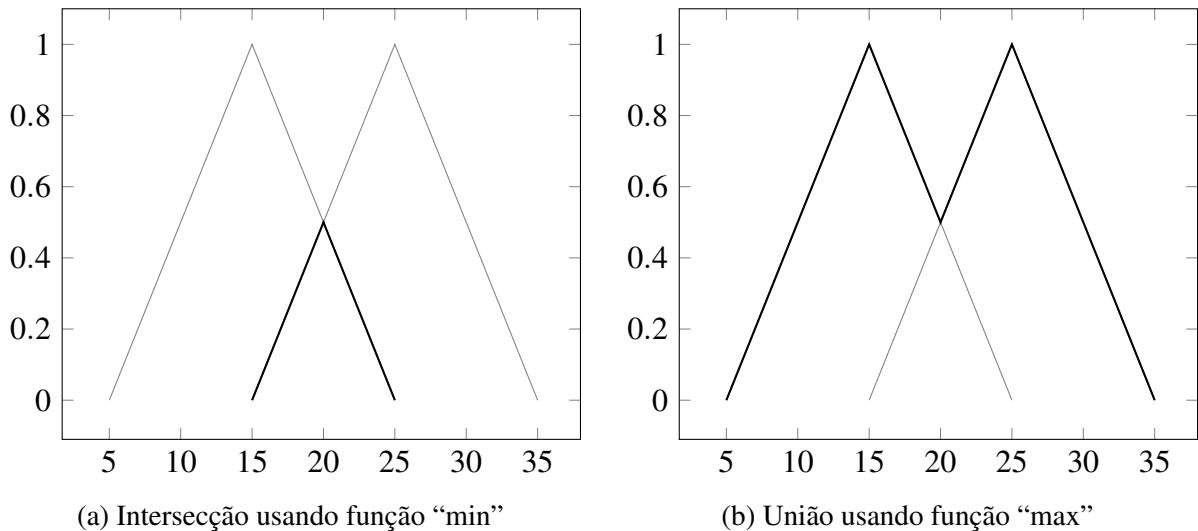
2.7.2 COMPONENTES DE UM SISTEMA *FUZZY*

Conhecimento qualitativo é utilizado para gerar regras do tipo “Se P então Q”, que relacionam valores linguísticos de entrada com valores linguísticos de saída. O conjunto de regras forma a “Base de Regras”, que pode ser vista como uma tabela de consulta, com número de dimensões igual ao número de variáveis (PASSINO; YURKOVICH, 1998).

2.7.2.1 FUZZIFICAÇÃO

O processo de associar variáveis numéricas a conjuntos fuzzy, a fim de obter valores linguísticos com seus respectivos valores de pertinência, é denominado Fuzzificação. A saída deste processo é o conjunto de graus de pertinência vinculados aos valores linguísticos definidos (PASSINO; YURKOVICH, 1998).

Figura 12 – Operações união e intersecção em conjuntos Fuzzy



Fonte: baseado em Lilly (2011), p. 20 e 21.

2.7.2.2 MECANISMO DE INFERÊNCIA

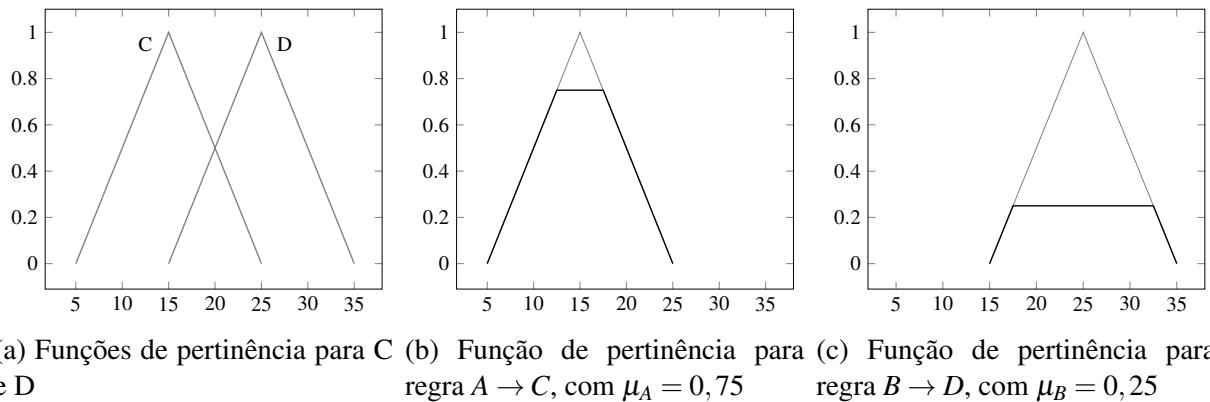
Ao final do processo de fuzzificação, a Base de Regras deve ser utilizada para encontrar saídas definidas por variáveis linguísticas. Contudo, apenas entradas associadas a graus de pertinência maiores que zero devem ser levados em consideração, a fim de obter um subconjunto de regras (ativas) a serem avaliadas (PASSINO; YURKOVICH, 1998).

Um conjunto de regras ativas e os respectivos graus de pertinência de suas premissas são utilizados para calcular funções de pertinência para cada regra. Como exemplo, dadas as regras $A \rightarrow C$ e $B \rightarrow D$, onde as premissas A e B possuem pertinências $\mu_A = 0,75$ e $\mu_B = 0,25$, e os consequentes C e D possuem funções de pertinência mostradas na Figura 13.a. As Figuras 13.b e 13.c mostram funções de pertinência obtidas para as regras. As saídas destas regras são vistas como recomendações que, apesar de conflitantes, podem ser combinadas de acordo com sua importância (pertinência) (PASSINO; YURKOVICH, 1998).

2.7.2.3 DEFUZZIFICAÇÃO

O processo de associar recomendações distintas dadas pelo conjunto de regras ativas a variáveis numéricas de saída é denominado Defuzzificação. Existem vários meios de cumprir tal objetivo, sendo que um método bastante utilizado é o cálculo do “centro de gravidade” (COG) das funções de pertinência das regras. A Equação 15 mostra este cálculo, no qual b_i é o

Figura 13 – Inferência em Lógica Fuzzy



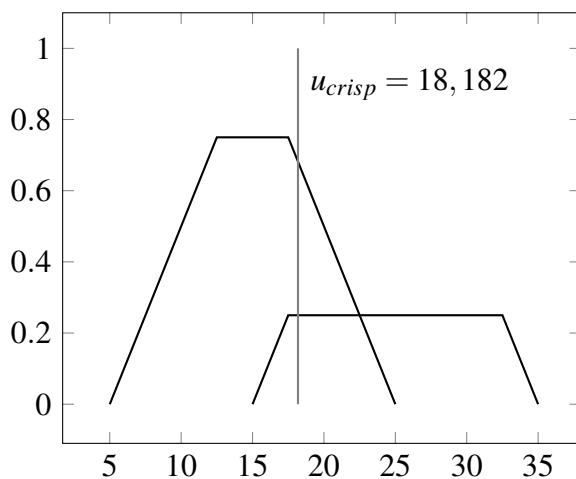
Fonte: baseado em Passino e Yurkovich (1998), p. 43 e 44.

centro da função de pertinência (PASSINO; YURKOVICH, 1998).

$$u_{crisp} = \frac{\sum_i b_i \int \mu_i}{\sum_i \int \mu_i} \quad (15)$$

O valor $\int \mu_i$ pode ser obtido analiticamente para uma função de pertinência triangular. Dada uma base “b” e altura de corte “h”, pode-se mostrar que a área se iguala a $b(h - \frac{h^2}{2})$ (PASSINO; YURKOVICH, 1998). Para o exemplo anterior, o processo de defuzzificação foi ilustrado na Figura 14.

Figura 14 – Defuzzificação COG



Fonte: baseado em Passino e Yurkovich (1998), p. 47.

As duas curvas da Figura 14 possuem áreas sobrepostas que estão sendo somadas duas vezes no cálculo de defuzzificação. Algumas bibliografias (por exemplo, Passino e Yurkovich (1998)) citam a existência de um cálculo global, que utiliza todas as funções para calcular a chamada “função de agregação” e o centro de massa é calculado sobre ela.

Nem todas as bibliografias mostram exemplos utilizando este método, por conta do custo computacional. Porém, ao longo do desenvolvimento, observou-se que o simulador (Matlab) estava computando a função de agregação por integração numérica, o que gerava diferenças de resultado no cálculo do microcontrolador. A função de agregação em simulação está sendo calculada aplicando a função *max()* sobre todas as funções parciais (verificado empiricamente). A Equação 16, encontrada em Ross (2009) mostra a saída *fuzzy* sobre a função de agregação, pelo método do centroide.

$$u_{crisp} = \frac{\int \mu^*(x) x dx}{\int \mu^*(x) dx} \quad (16)$$

Para manter o resultado da implementação fiel à simulação, mas ao mesmo tempo evitar ter de realizar um método de integração numérica no microcontrolador, foi desenvolvida uma forma de calcular a integral analítica da curva, o que reduz complexidade computacional, ao custo de aumentar complexidade de espaço para o armazenamento de uma estrutura de dados.

Foi utilizada uma lista encadeada para armazenar parâmetros das funções de pertinência. A lista é ordenada pela posição do centro de massa da função de pertinência no eixo X. Como essas funções são todas triangulares, a função de agregação é descontínua, porém formada por retas. O algoritmo para cálculo do centro de massa, ao percorrer a lista, calcula e soma as integrais definidas parciais.

2.8 ODOMETRIA

Odometria em robótica móvel é o processo de cálculo da posição do robô utilizando sensores. No caso específico de um robô móvel de acionamento diferencial, deseja-se calcular a coordenada em um plano (x e y) e a orientação (ϕ) do robô a partir de sensores de efeito hall acoplados aos mostores. (OLSON, 2009)

Cada motor possui dois sensores, de modo a produzir sinais em canais A e B, que se comportam conforme a Figura 15. Para medir velocidade, deve-se calcular a quantidade de pulsos gerados em um período de tempo, utilizando a especificação de pulsos por revolução (PPR), bem como a medida de circunferência do pneu (DYNAPAR, 2020). A Equação 17 mostra o cálculo de velocidade, onde N_{ticks} é o período de amostragem, N_{PPR} é o número de partes por revolução, R é o raio do pneu e T é o período de amostragem.

$$V_{pneu} = \frac{N_{ticks}}{N_{PPR}} \frac{2\pi R}{T} \quad (17)$$

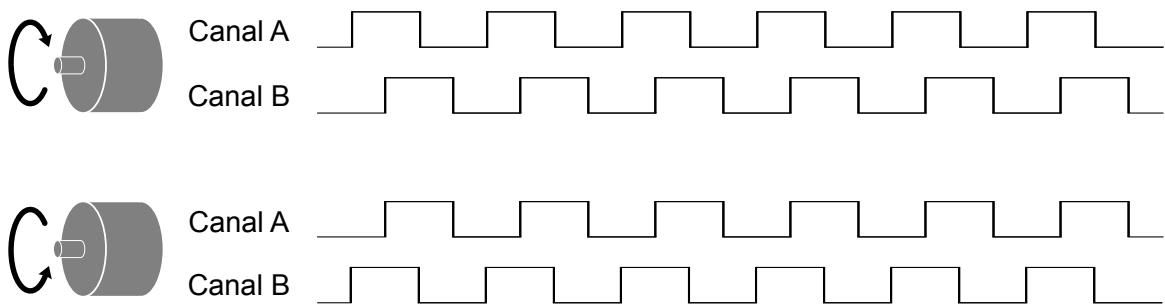


Figura 15 – Sinais de saída dos canais dos *encoders* para sentido de rotação

Fonte: baseado em Dynapar (2020)

A utilização de uma caixa de redução entre o motor e o pneu adiciona uma particularidade interessante, pois apesar de reduzir a velocidade máxima, aumenta a precisão dos sensores pela razão do sistema de engrenagens.

Para descobrir a direção de rotação, é necessário analisar a diferença de fase entre os sinais. Ao detectar borda de subida e descida em um dos canais e a seguir verificar se os dois possuem sinais lógicos iguais ou diferentes, um contador é incrementado ou decrementado.

Decidir se o contador será incrementado ou decrementado na presença de sinais iguais é um processo empírico, pois o sentido de movimento depende da fixação dos motores no chassi do robô. Como há diferença de 180 graus na fixação, um sentido horário de rotação pode ser "para frente" ou "para trás" dependendo de qual motor for analisado.

Pode-se utilizar apenas borda de subida para esse cálculo, porém, ao levar em consideração as duas bordas do sinal, duplica-se a contagem de pulsos por revolução.

O cálculo do estado atual a partir de estados anteriores e medições dos encoders pode ser obtido a partir da Equação 18, onde $[x, y, \phi]^T$ é o estado anterior, $[x' y' \phi']^T$ é o estado a ser calculado, L é a distância entre os pneus e d_{esq} e d_{dir} são as distâncias percorridas pelos pneus esquerdo e direito, respectivamente. (OLSON, 2009).

$$\begin{cases} x' = x + \frac{d_{esq} + d_{dir}}{2} \cos \phi \\ y' = y + \frac{d_{esq} + d_{dir}}{2} \sin \phi \\ \phi' = \phi + \frac{d_{dir} - d_{esq}}{L} \end{cases} \quad (18)$$

3 MATERIAIS E MÉTODO

Este Capítulo descreve os requisitos materiais para o desenvolvimento do trabalho e os métodos que serão adotados para atingir os objetivos.

3.1 MATERIAIS

Os materiais a serem utilizados dividem-se em componentes eletrônicos e componentes físicos. Os componentes eletrônicos podem ser vistos na tabela 1.

Tabela 1 – Tabela de custos

Item	Qty	Custo Unitário	Custo Total
Bateria LiPo Limskey 11,1v 3S 4200mAh	1	112,05	112,05
Módulo para cartão de memória SD WAVGAT, comunicação SPI	1	2,79	2,79
Ponte H dupla L298N	1	6,12	6,12
Sensor infravermelho GP2Y0A21YK0F, alcance de 10 a 80 cm	5	13,07	65,35
Módulo conversor Buck DC-DC, saída 5v	1	9,14	9,14
Sensor inercial IMU GY-87, com 10 graus de liberdade (MPU6050, HCM5883L, BMP180)	1	25,9	25,9
Módulo wireless NRF24L01 com antena de alcance de 1000 metros	1	7,69	7,69
Conjuntos de motores, caixa de redução 1:34 e sensores de efeito hall (341,2 PPR)	2	47,015	94,03
Esfera de rolagem	1	2,002	2,002
Tiva Connected LaunchPad TM4C1294	1	80,00	80,00
Total:			405,07

Materiais para o corpo do robô, como chapa de MDF para o chassi, parafusos de rosca, abraçadeiras não tiveram seus custos contabilizados.

Entre as ferramentas, será utilizado *software* Matlab, com o simulador Simiam (GEORGIA ROBOTICS AND INTELLIGENT SYSTEMS LABORATORY, 2013), que é implementado como um aplicativo Matlab.

3.2 METODOLOGIA

Neste trabalho o protótipo será desenvolvido tendo como base o robô móvel de acionamento diferencial Quickbot (DUKOR et al., 2017). Porém, esse robô se encontra com alguns componentes de hardware obsoletos. Componentes mais atuais serão utilizados.

O simulador Simiam será estudado de modo a permitir a inserção de novas funcionalidades, como o módulo *fuzzy*, pois este simulador foi implementado visando a utilização de uma arquitetura de controle híbrida.

Em seguida, deve-se projetar os comportamentos do robô que solucionem o problema proposto de duas maneiras separadas: utilizando o sistema híbrido e por meio de regras *fuzzy*.

Utilizar o simulador Simiam, desenvolvido pela Universidade Georgia Tech, para validar o algoritmo de controle.

Num próximo passo, a montagem física deve tomar espaço no cronograma, concomitante à implementação do sistema embarcado.

Os testes serão realizados em ambiente pré-definido. Contudo, os obstáculos serão colocados em posições distintas para verificar a robustez do algoritmo.

Para finalizar, o robô móvel e os algoritmos implementados no sistema embarcado serão submetidos a testes em protótipo.

4 DESENVOLVIMENTO

Este Capítulo apresenta o que foi desenvolvido como resultado deste trabalho, visando alcançar os objetivos propostos.

4.1 O SIMULADOR SIMIAM E ALTERAÇÕES NECESSÁRIAS

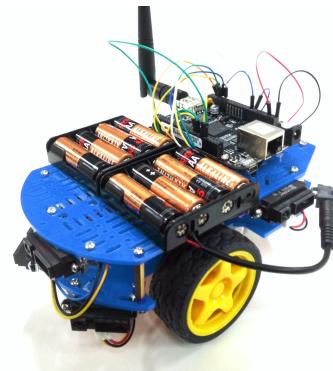
O Simulador Simiam foi implementado pela Universidade Georgia Tech, inicialmente oferecendo apenas suporte ao robô Khepera 3. Posteriormente, para atender necessidades do curso *Control of Mobile Robots*, hospedado na plataforma “Coursera.org”, o robô de baixo custo QuickBot foi adicionado. Os robôs Khepera e QuickBot reais podem ser vistos na Figura 16. Suas contrapartes simuladas estão retratadas na Figura 17.

O curso mencionado foi removido da plataforma no dia 17 de Agosto de 2020 e apenas os alunos concluíntes possuem acesso. De qualquer forma, o simulador ainda pode ser obtido gratuitamente.

Figura 16 – Robôs Khepera 3 e QuickBot



(a) Robô Khepera 3

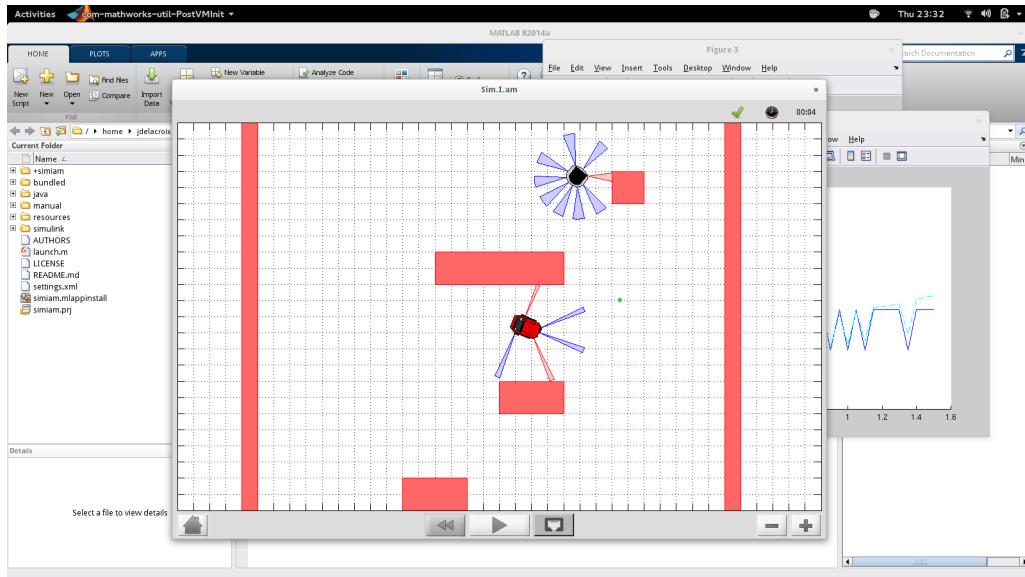


(b) Robô QuickBot

Fonte: Wang (2008), Fuchs et al. (2013)

O Simiam é implementado como um aplicativo Matlab. Possui uma interface gráfica intuitiva e uma arquitetura interna simples, facilmente customizável. As subseções a seguir apresentam os pacotes mais importantes, bem como as principais classes que os compõem e chama atenção para as alterações necessárias a fim de tornar a simulação coerente com o presente trabalho.

Figura 17 – Robôs Khepera 3 e QuickBot em simulação



Fonte: Croix (2013)

4.1.1 PACOTE “UI”

O pacote “ui” é responsável pela interface gráfica do simulador. A classe “AppWindow” possui um método construtor que inicializa atributos e o método “load_ui” que invoca o método “create_layout”, responsável por criar o leiaute da interface.

O botão start, criado pelo método anterior é tratado por “ui_button_start”. Esta é a função principal responsável por criar o ambiente de simulação (definido no arquivo “settings.xml”), inserir um ou mais robôs e iniciar a simulação.

Alterações mínimas foram efetuadas nesta etapa. Foram adicionados comandos para maximizar a janela do simulador e para ajustar o “zoom” de modo a permitir uma visão panorâmica de todo o ambiente de simulação. O arquivo de configuração “settings.xml” foi alterado a fim de ajustar o “tamanho do mundo” ao tamanho da tela.

4.1.2 PACOTE “SIMULATOR”

O pacote “simulator” realiza de fato a simulação. A classe “World” é responsável por extrair informações do arquivo “.xml”, como quantidade e localização inicial de robôs e obstáculos, armazenado em estruturas de dados. A classe “Simulator” atualiza a simulação na janela, enquanto a classe “Physics” é responsável por detectar colisões de robôs com obstáculos e entre si.

As alterações neste pacote foram todas na classe “Simulator”, no intuito de possibilitar

a captura da simulação em video “.mp4” ou “.gif”. Apesar de não ser uma alteração fundamental para a etapa de execução do trabalho, é necessária para a apresentação. Não foi criado um botão na interface para especificar ao simulador a captura em video. Assim, é necessário, no construtor da classe Simulator, atribuir valor verdadeiro aos atributos “gravarvideo” e/ou “gravargif”.

4.1.3 PACOTE “ROBOT”

O pacote “robot” define um ou mais robôs passíveis de serem instanciados. A classe “QuickBot” estabelece parâmetros físicos, tais como informações geométricas, posicionamento dos sensores no corpo do robô e estabelece limitações, como velocidades de saturação e zona morta dos motores. Além de instanciar objetos das classes “DifferentialDrive”, “ProximitySensor” e “WheelEncoder”, “QuickBot” extende a classe “Robot”.

A classe “DifferentialDrive” implementa a “dinâmica” dos modelos de acionamento diferencial e uniciclo. Com os métodos “uni_to_diff” e “diff_to_uni”, a equivalência entre modelo apresentada na Equação 6 é estabelecida no espaço de simulação.

A classe “ProximitySensor” estabelece características do sensor infravermelho usado no QuickBot, tais como distâncias mínimas e máximas, espalhamento, localização e direção em relação ao corpo do robô e adiciona um ruído gaussiano. “WheelEncoder”, similarmente, estabelece características do *encoder*.

As alterações nesta etapa foram no intuito de incluir o robô deste trabalho no Simiam. O arquivo “settings.xml” deve determinar que um objeto (robô) da nova classe criada seja instanciado, ao invés do objeto da classe QuickBot.

4.1.4 PACOTE “CONTROLLER”

O pacote “controller” é responsável pela implementação de todos os controladores e supervisores dos robôs implementados. A classe “Supervisor” é extendida para obter o controlador supervisório de cada robô a ser simulado. Para o QuickBot e Khepera 3, os respectivos supervisores são definidos pelas classes “QBSupervisor” e “K3Supervisor”. Essas classes definem máquinas de estado, onde cada estado é associado a um controlador de variável contínua. Esses controladores, por sua vez, extendem a classe “Controller” e implementam os “comportamentos”, ou modos, dos robôs.

As alterações neste pacote foram responsáveis pela inclusão de suporte a controladores Fuzzy, além de adicionar o supervisório específico para o robô desenvolvido.

4.2 MONTAGEM FÍSICA DO ROBÔ

Os componentes físicos do robô podem ser vistos na Figura 18.a e o resultado após montagem está retratado na Figura 18.b.

Figura 18 – Materiais e robô após montagem



(a) Materiais do robô

(b) Robô montado

Fonte: autoria própria

4.2.1 ESPECIFICAÇÕES

As variáveis L e R na Equação 5, para o robô deste trabalho, valem respectivamente 18 cm e 3,4 cm.

O sensor infravermelho GP2Y0A21YK0F utilizado neste trabalho possui a curva de tensão por distância representada na Figura 19. A faixa de distância está entre 10 e 80 cm. No QuickBot, é utilizado o sensor GP2Y0A41SK0F, com distância de medição entre 4 e 30 cm.

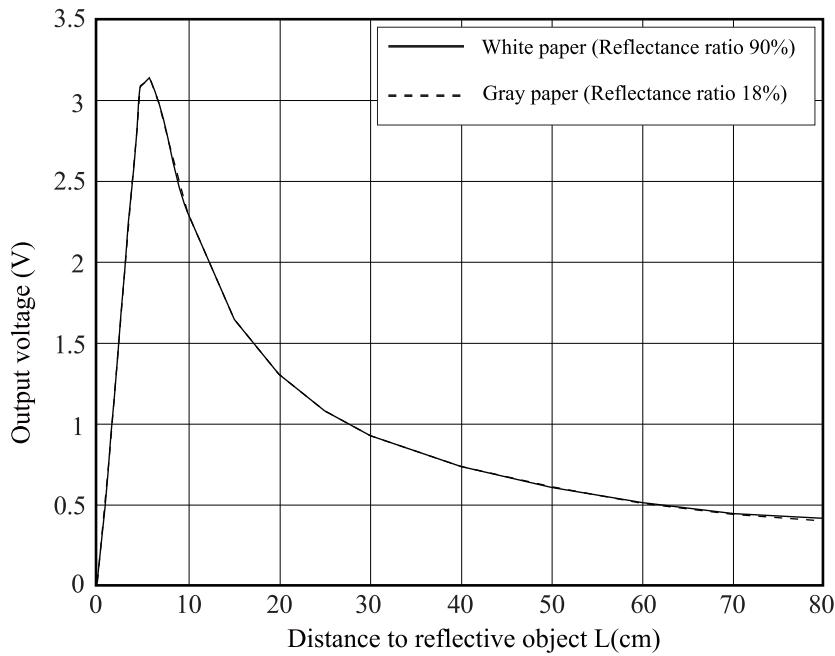
Para a região da curva onde $d > 5,5\text{cm}$, o polinômio que a aproxima, associando o inverso da distância pela tensão ($d(v) = \frac{1}{d'(v)}$), obtido utilizando a função polyfit do Matlab pode ser visto na Equação 19.

$$d'(v) = 21.49298v^5 + 21.86894v^4 + 22.25754v^3 + 22.65912v^2 + 23.07406v + 23.50272 \quad (19)$$

4.3 DESENVOLVIMENTO DOS CONTROLADORES

Nesta seção pretende-se investigar a arquitetura comportamental em robótica móvel para as duas estratégias clássicas de implementação: arbitragem e fusão de comportamentos.

Figura 19 – Resposta do sensor infravermelho



Fonte: SHARP Corporation (2006)

Para isso, controladores híbrido e fuzzy serão criados, o primeiro para arbitragem e o segundo para fusão.

4.3.1 POR CONTROLADOR HÍBRIDO

Nesta subseção, pretende-se demonstrar um controlador simples capaz de solucionar o problema de navegação, usando autômato híbrido. Para cada estado no autômato, um controlador é selecionado e uma recomendação de saída é calculada.

A recomendação para todos os comportamentos é dada por um vetor. Seu ângulo é calculado para o sistema de coordenadas global e, subtraindo o ângulo do robô, o erro é calculado. A partir do erro, um controlador PID calcula a velocidade angular.

A velocidade linear é definida inicialmente como fixa, sendo que cada comportamento pode definir sua velocidade máxima. Após o cálculo de velocidade angular pelo controlador, a Equação 6 é utilizada para calcular velocidades angulares dos motores que atendam aos requisitos de velocidade linear e angular do modelo uniciclo. Contudo, w_l e w_r podem estar saturadas ou em região de zona morta.

Em condição de saturação, a velocidade linear é sacrificada para atender o requisito de velocidade angular. A diferença entre a velocidade sugerida e a velocidade máxima permitida

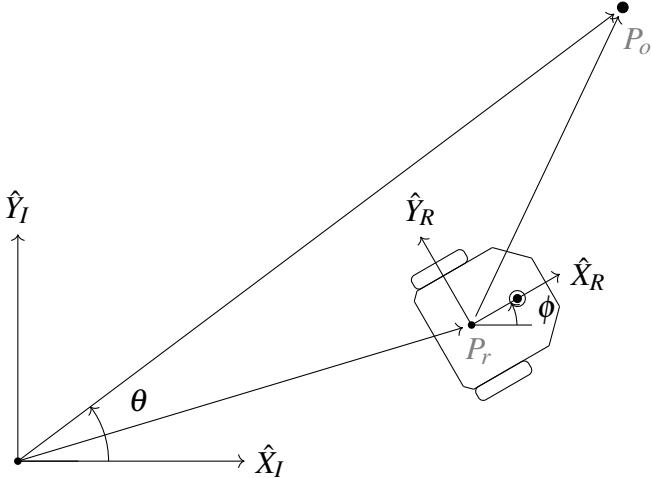
pelos motores é subtraída de w_l e w_r .

Para evitar zona morta, a diferença entre velocidade mínima permitida e velocidade sugerida é adicionada a w_l e w_r para evitar travamento das rodas devido ao atrito.

4.3.1.1 COMPORTAMENTO "IR PARA OBJETIVO"

O comportamento “Ir Para Objetivo”, exemplificado na Figura 20, calcula um vetor normalizado que aponta para o objetivo a partir do centro de massa do robô. O vetor é calculado operando $P_o - P_r$, onde P_o e P_r são pontos que podem ser vistos na Figura. A recomendação para velocidade linear é a máxima, já que o robô está livre de obstáculos.

Figura 20 – Comportamento Ir para Objetivo



Fonte: autoria própria

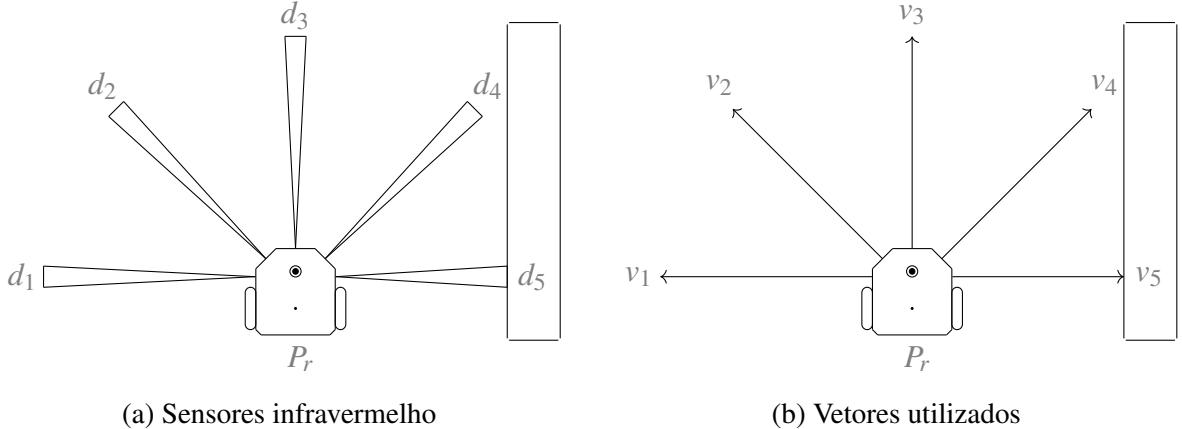
O controlador PID para comportamento “Ir Para Objetivo” possui parâmetros $k_p = 4$ e $k_i = k_d = 0,01$. O erro no ângulo, entrada do controlador, é calculado pela Equação 20, onde ϕ é o ângulo do robô e a função $atan2()$ é o arco tangente com dois parâmetros, presente em muitas linguagens de programação, tendo a vantagem de calcular o ângulo sem dúvidas quanto ao quadrante.

$$e_\theta = atan2(v_y, v_x) - \phi \quad (20)$$

4.3.1.2 COMPORTAMENTO "EVITAR OBSTÁCULO"

O comportamento “Evitar Obstáculo” calcula um vetor cujo objetivo é afastar o robô de barreiras encontradas. A Figura 21.a mostra a disposição dos sensores enquanto a Figura 21.b mostra vetores com origem nas coordenadas dos sensores e cujos módulos são iguais às distâncias medidas.

Figura 21 – Comportamento Evitar Obstáculo



Fonte: autoria própria

É interessante definir sistemas de coordenadas para cada sensor de modo que a coordenada do obstáculo possa ser representada como um vetor da forma $[d_i \ 0]^T$, onde d_i é a distância detectada pelo i-ésimo sensor. Assim, a representação é extremamente simples e, com matrizes de rotação, pode-se encontrar suas posições no sistemas de coordenadas do robô.

A representação dos obstáculos no sistema de coordenadas do robô está retratada na Equação 21, onde θ_i é o ângulo do i-ésimo sensor, d_i é a distância ao obstáculo (pode estar saturada) e $[o_x \ o_y]_i^T$ é um vetor na coordenada do robô que define um *offset*, apontando para a origem do sistema de coordenadas do sensor.

$$\begin{bmatrix} x \\ y \end{bmatrix}_i^R = \begin{bmatrix} \cos \theta_i & -\sin \theta_i \\ \sin \theta_i & \cos \theta_i \end{bmatrix} \begin{bmatrix} d_i \\ 0 \end{bmatrix} + \begin{bmatrix} o_x \\ o_y \end{bmatrix}_i^R \quad (21)$$

Os cinco sensores foram posicionados de modo a formar ângulos de $-\pi/2$, $-\pi/4$, 0 , $\pi/4$ e $\pi/2$, respectivamente. Os cinco vetores no sistema de coordenadas do robô são combinados linearmente para formar uma única recomendação. A recomendação é dada pela Equação 22.

$$\mathbf{u}_{eo} = k_1 \mathbf{v}_1 + k_2 \mathbf{v}_2 + k_3 \mathbf{v}_3 + k_4 \mathbf{v}_4 + k_5 \mathbf{v}_5 + \mathbf{v}_{eq} \quad (22)$$

As constantes k_1 a k_5 valem respectivamente, 0.7, 2, 1.2, 2 e 0.7. Elas foram definidas a partir de simulação. A razão de considerar peso maior para os sensores diagonais ($-\pi/4$ e $\pi/4$) é causar uma deflexão ao encontrar obstáculos frontais. O peso pequeno nos sensores laterais ($-\pi/2$ e $\pi/2$) é devido ao baixo risco de colisão, já que o sentido de movimento do

robô é perpendicular a estes obstáculos.

Em condição de ausência de obstáculos, os módulos dos vetores v_1 a v_5 estarão saturados em 80 (calculo dos sensores em centímetros). Neste caso, a soma dos cinco primeiros termos tem como resultante um vetor ao longo do eixo X no sistema de coordenadas do robô. Ao encontrar uma parede posicionada na perpendicular em relação ao sentido de movimento, a resultante continuaria no eixo X, provocando colisão. O vetor v_{eq} tem por objetivo reduzir a magnitude da resultante. Seu módulo poderia ser definido de modo que a resultante se torne zero a uma distância segura do obstáculo.

A condição de equilíbrio é indesejável, pois o robô não atinge o objetivo, mas este caso é um excelente ponto de partida para compreender o comportamento de evitar obstáculo. A partir do equilíbrio, se for adicionado um obstáculo em qualquer sensor (ou aproximado um obstáculo já existente), a magnitude do vetor correspondente diminui, e a resultante será no sentido contrário.

O valor de v_{eq} foi definido como $[-240 \ 0]^T$, de modo que o robô não para até se chocar com o obstáculo. Portanto, o efeito deste vetor é apenas reduzir a aparente “inercia” e aumentar o efeito de deflexão ao encontrar obstáculos em ângulos oblíquos. Para aproximações com angulos retos, é necessário um cálculo diferente.

4.3.1.3 COMPORTAMENTO MESCLADO ”IR PARA OBJETIVO E EVITAR OBSTÁCULO”

deflexão!

4.3.1.4 MÍNIMOS LOCAIS

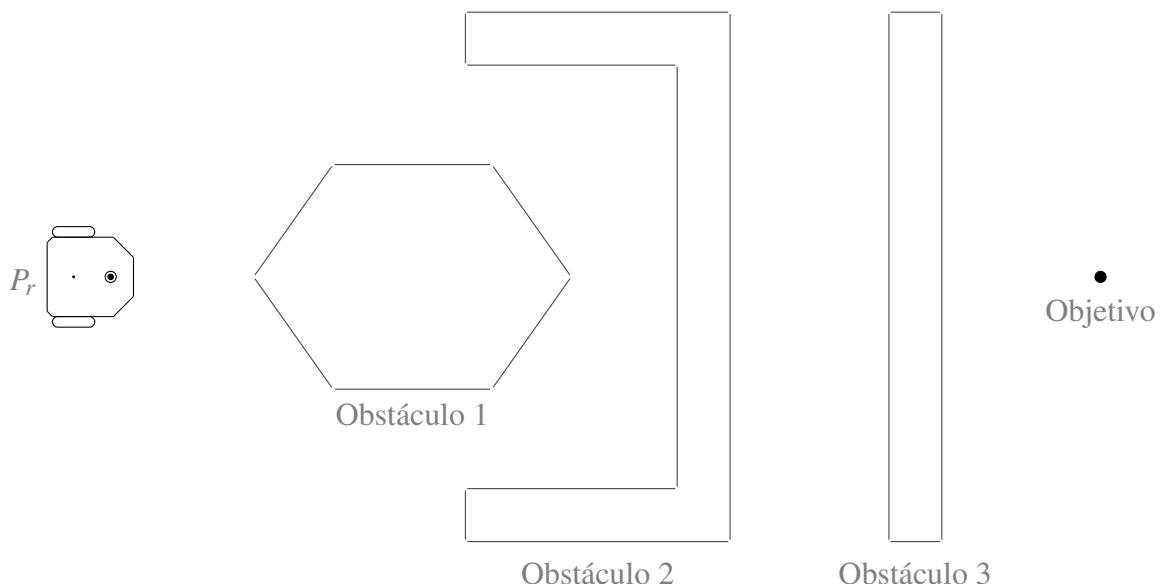
4.3.1.5 COMPORTAMENTO ”SEGUIR PAREDE”

$$\mathbf{u}_a \cdot \mathbf{u}_t = |\mathbf{u}_a| \cdot |\mathbf{u}_t| \cos(\theta) \quad (23)$$

$$\mathbf{u}_a \cdot \mathbf{u}_t = |\mathbf{u}_a| \cos(\theta) = d \quad (24)$$

$$\mathbf{u}_p = \mathbf{u}_a - d \cdot \mathbf{u}_t \quad (25)$$

Figura 22 – Tipos de obstáculos



Fonte: autoria própria

4.3.1.6 O AUTÔMATO PARA ARBITRAGEM DE COMPORTAMENTOS

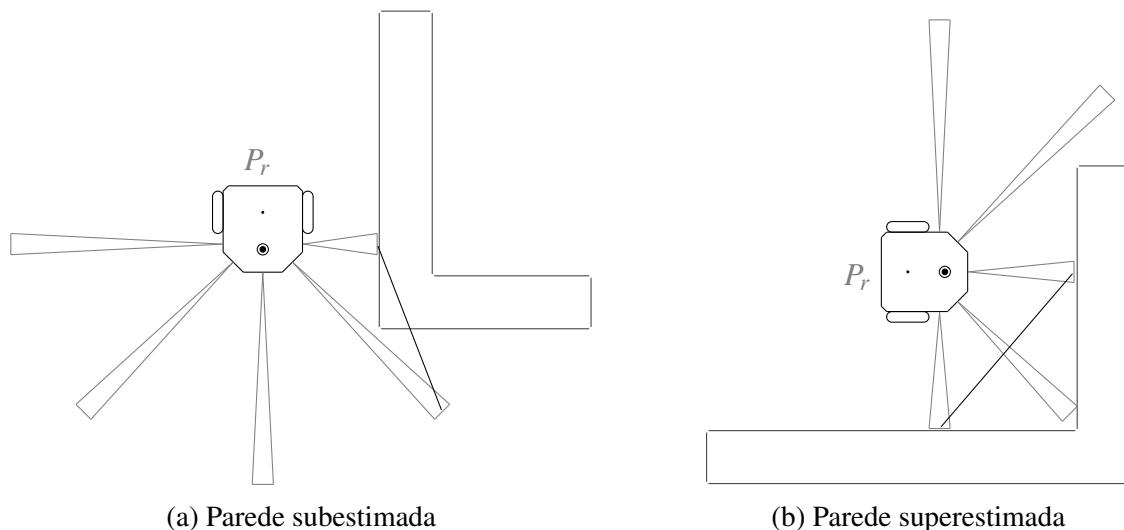
Tabela 2 – Estados do autômato

Índice	Comportamento	Descrição
0	P	Parar
1	IPO	Ir para Objetivo
2	EO	Eitar Obstáculos
3	IPO_E.EO	Ir para Objetivo e Eitar Obstáculos (combinados)
4	SP AH	Seguir Parede sentido Anti-horário
5	SP H	Seguir Parede sentido Horário

4.3.2 POR CONTROLADOR FUZZY

Nesta seção, pretende-se demonstrar um controlador simples capaz de solucionar o problema de navegação, usando sistema *fuzzy*.

Figura 23 – Comportamento Seguir Parede



Fonte: autoria própria

Tabela 3 – Condições utilizadas nas transições do autômato

Legenda	Condições	Cálculo da condição
a	No objetivo	Equacao1
b	Fez progresso	Equacao1
c	Tem Obstáculo	Equacao1
d	Está inseguro	Equacao1
e	Livre de obstáculo	Equacao1
f	Contornando pela esquerda	Equacao1
g	Contornando pela direita	Equacao1

4.3.2.1 COMPORTAMENTO "EVITAR OBSTÁCULO"

4.3.2.2 COMPORTAMENTO "SEGUIR PAREDE"

4.3.2.3 COMPORTAMENTO "SEGUIR RECOMENDAÇÃO"

4.3.2.4 A ESTRATÉGIA PARA FUSÃO DE COMPORTAMENTOS

4.4 SIMULAÇÃO

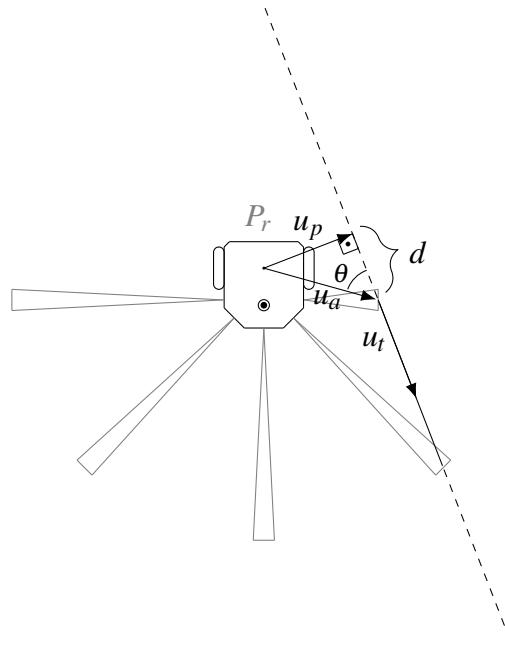
4.4.1 UTILIZANDO CONTROLADOR HÍBRIDO

a

4.4.2 UTILIZANDO CONTROLADOR FUZZY

a

Figura 24 – Recomendação vetorial



Fonte: autoria própria

4.5 MONTAGEM FÍSICA

4.5.1 PROCESSO DE MONTAGEM

4.5.2 CURVAS DOS MOTORES

a

4.6 IMPLEMENTAÇÃO DO SISTEMA EMBARCADO

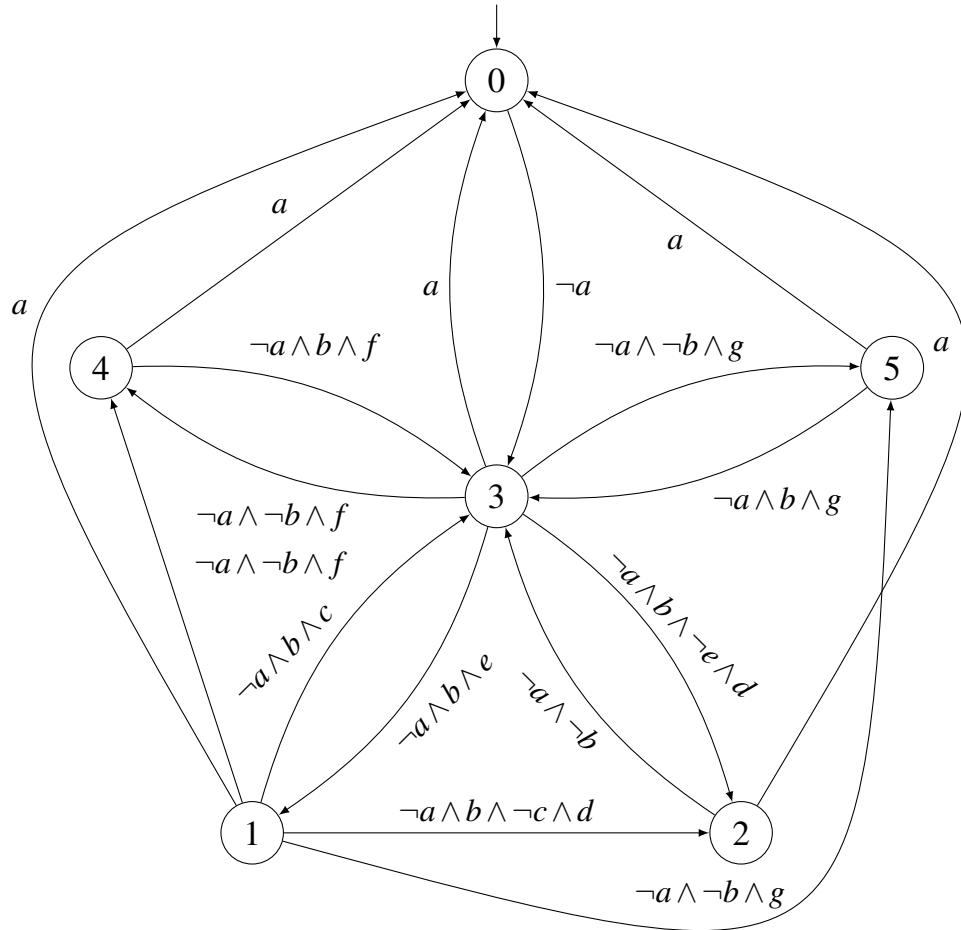
4.6.1 CONSIDERAÇÕES PRÁTICAS, LIMITAÇÕES DO MODELO E DA SIMULAÇÃO

4.6.2 ESQUEMA LÓGICO DO SISTEMA EMBARCADO

4.6.3 PROJETO DA PLACA

a

Figura 25 – Autômato Híbrido que soluciona o problema de navegação



Fonte: autoria própria

4.6.4 IMPLEMENTAÇÃO DA ABORDAGEM HÍBRIDA

4.6.5 IMPLEMENTAÇÃO DA SOLUÇÃO FUZZY

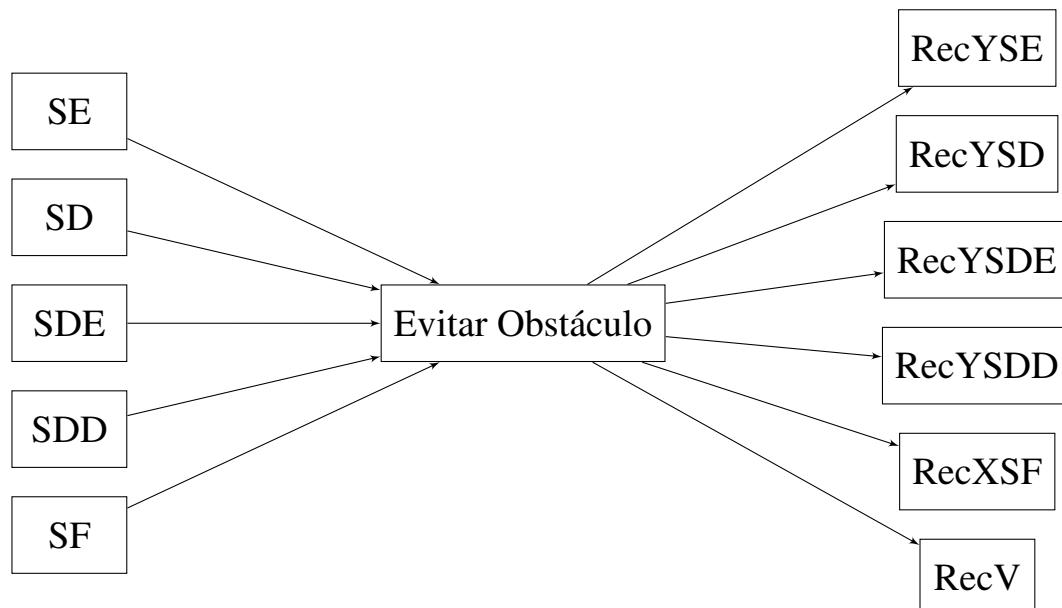
4.7 RESULTADO E DISCUSSÃO

4.7.1 RESULTADO PARA CONTRADOR HÍBRIDO

a

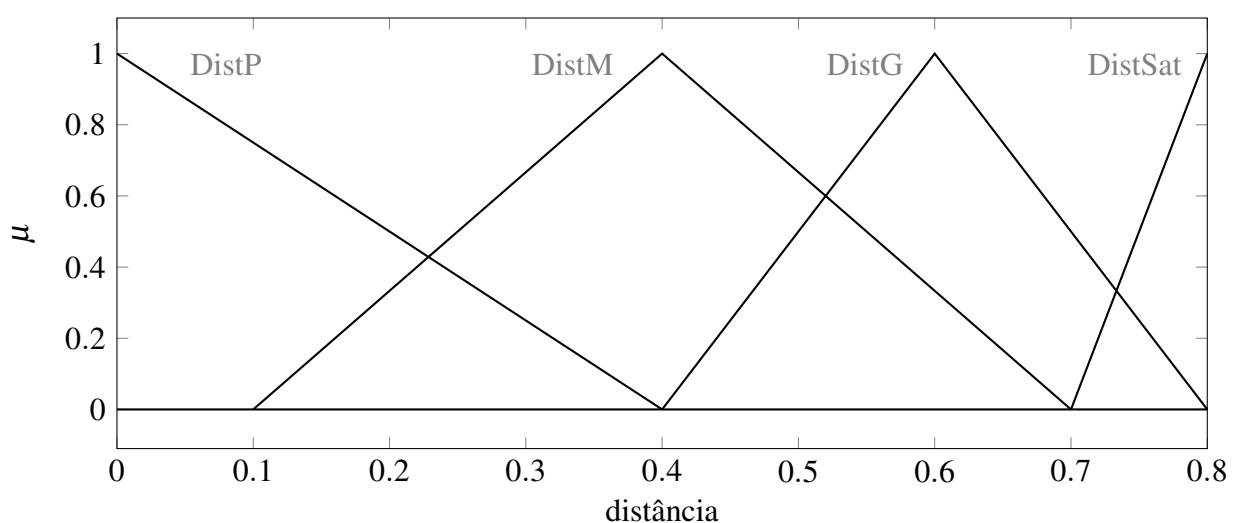
4.7.2 RESULTADO PARA CONTRADOR FUZZY

Figura 26 – Diagrama do sistema Fuzzy “Evitar Obstáculo”



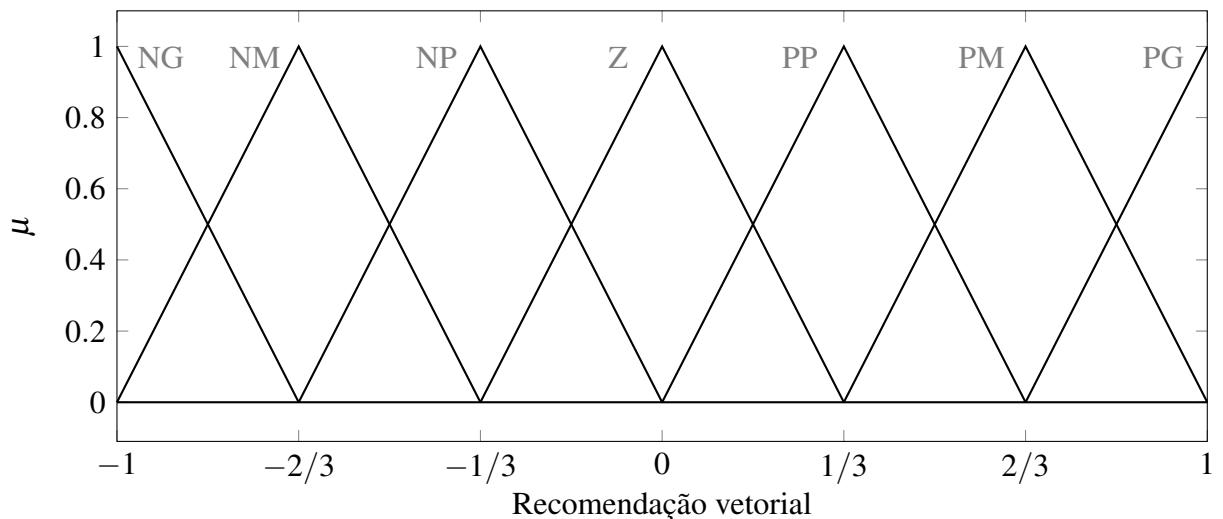
Fonte: autoria própria

Figura 27 – Conjuntos Fuzzy para as entradas dos sensores



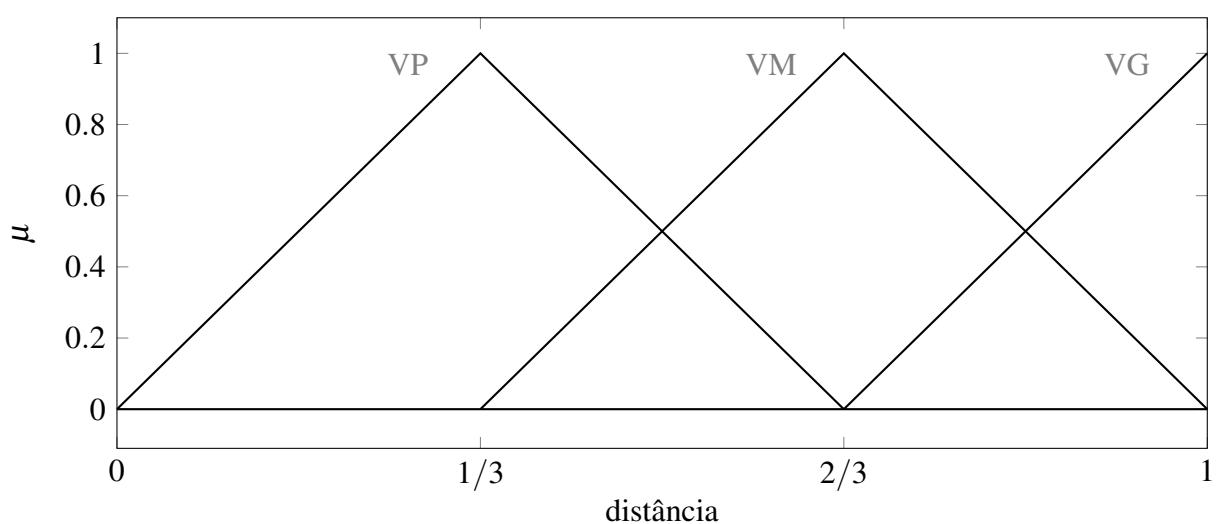
Fonte: autoria própria

Figura 28 – Conjuntos Fuzzy para as saídas vetoriais



Fonte: autoria própria

Figura 29 – Conjuntos Fuzzy para a saída de velocidade



Fonte: autoria própria

Tabela 4 – Regras Fuzzy para sistema “Evitar Obstáculo”

	Entradas					Saídas					
	SE	SD	SDE	SDD	SF	RecYSE	RecYSD	RecYSDE	RecYSDD	RecXSF	RecV
1	DistP	-	-	-	-	NG	-	-	-	-	-
2	DistM	-	-	-	-	NM	-	-	-	-	-
3	DistG	-	-	-	-	NP	-	-	-	-	-
4	DistSat	-	-	-	-	Z	-	-	-	-	VG
5	-	DistP	-	-	-	-	PG	-	-	-	-
6	-	DistM	-	-	-	-	PM	-	-	-	-
7	-	DistG	-	-	-	-	PP	-	-	-	-
8	-	DistSat	-	-	-	-	Z	-	-	-	VG
9	-	-	DistP	-	-	-	-	NG	-	-	VP
10	-	-	DistM	-	-	-	-	NM	-	-	VM
11	-	-	DistG	-	-	-	-	NP	-	-	VM
12	-	-	DistSat	-	-	-	-	Z	-	-	VG
13	-	-	-	DistP	-	-	-	-	PG	-	VP
14	-	-	-	DistM	-	-	-	-	PM	-	VM
15	-	-	-	DistG	-	-	-	-	PP	-	VM
16	-	-	-	DistSat	-	-	-	-	Z	-	VG
17	-	-	-	-	DistP	-	-	-	-	NG	VP
18	-	-	-	-	DistM	-	-	-	-	NM	VM
19	-	-	-	-	DistG	-	-	-	-	NP	VM
20	-	-	-	-	DistSat	-	-	-	-	Z	VG

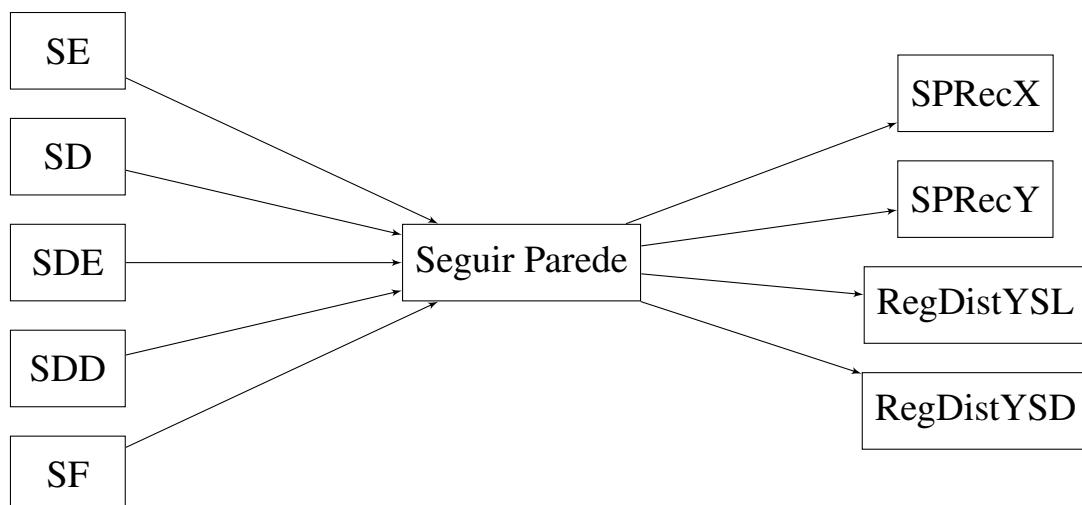
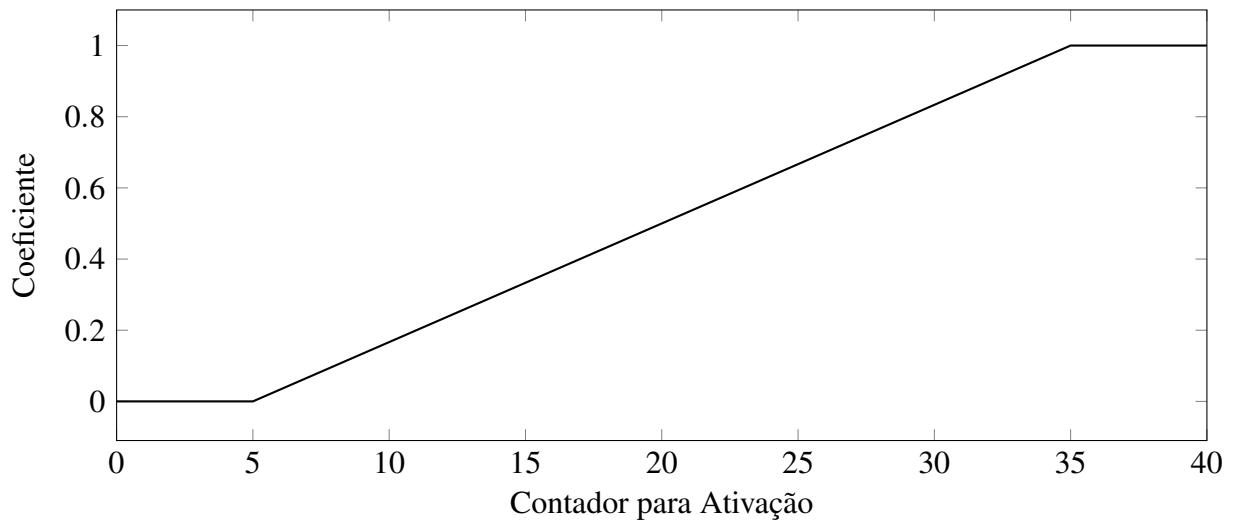
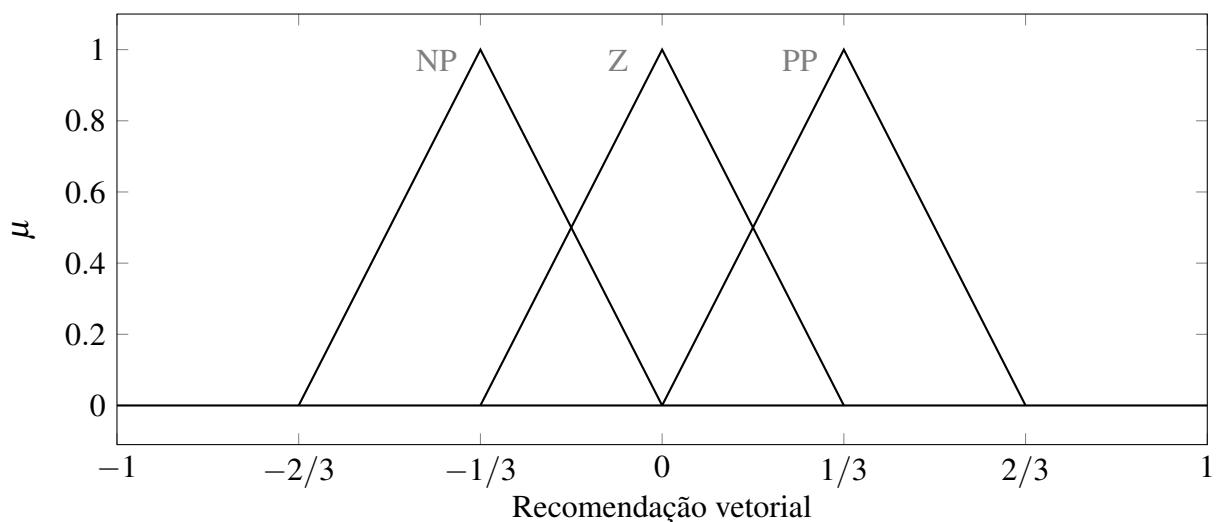
Figura 30 – Diagrama do sistema Fuzzy “Seguir Parede”**Fonte:** autoria própria

Figura 31 – Curva de Ativação para comportamento Seguir Parede



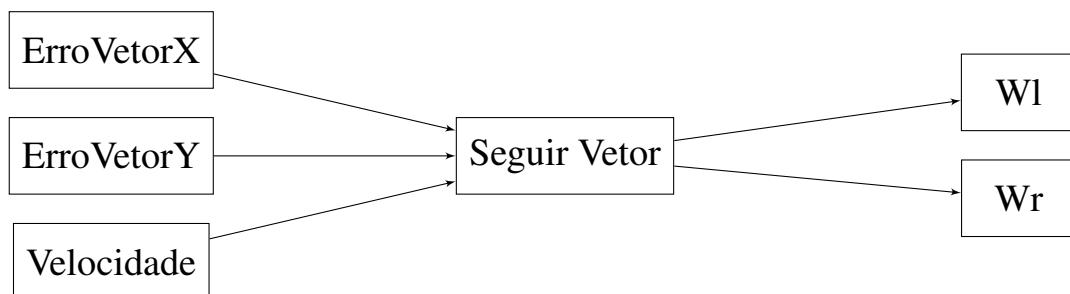
Fonte: autoria própria

Figura 32 – Conjuntos Fuzzy para as saídas vetoriais



Fonte: autoria própria

Figura 33 – Diagrama do sistema Fuzzy “Seguir Recomendação”



Fonte: autoria própria

Tabela 5 – Regras Fuzzy para sistema “Seguir Parede”

Relação das Entradas		Entradas					Saídas			
		SE	SD	SDE	SDD	SF	SPRecX	SPRecY	RegDistYSL	RegDistYSR
AND	1	DistSat	DistSat	DistSat	DistSat	-	Z	Z	-	-
OR	2	\neg DistSat	\neg DistSat	\neg DistSat	\neg DistSat	-	PP	-	-	-
AND	3	\neg DistSat	-	DistSat	-	-	-	PP	Z	-
AND	4	-	\neg DistSat	-	DistSat	-	-	NP	Z	-
AND	5	\neg DistSat	-	-	-	\neg DistSat	-	NP	Z	-
AND	6	-	\neg DistSat	-	-	\neg DistSat	-	PP	Z	-
AND	7	DistP	-	-	-	-	-	-	Z	-
AND	8	DistM	-	-	-	-	-	-	PP	-
AND	9	DistG	-	-	-	-	-	-	PP	-
AND	10	-	DistP	-	-	-	-	-	Z	-
AND	11	-	DistM	-	-	-	-	-	NP	-
AND	12	-	DistG	-	-	-	-	-	NP	-
AND	13	-	-	DistP	-	-	-	-	-	Z
AND	14	-	-	DistM	-	-	-	-	-	PP
AND	15	-	-	DistG	-	-	-	-	-	PP
AND	16	-	-	-	DistP	-	-	-	-	Z
AND	17	-	-	-	DistM	-	-	-	-	NP
AND	18	-	-	-	DistG	-	-	-	-	NP
AND	19	\neg DistSat	-	\neg DistSat	-	\neg DistSat	-	NP	Z	-
AND	20	-	\neg DistSat	-	\neg DistSat	-	PP	Z	-	-

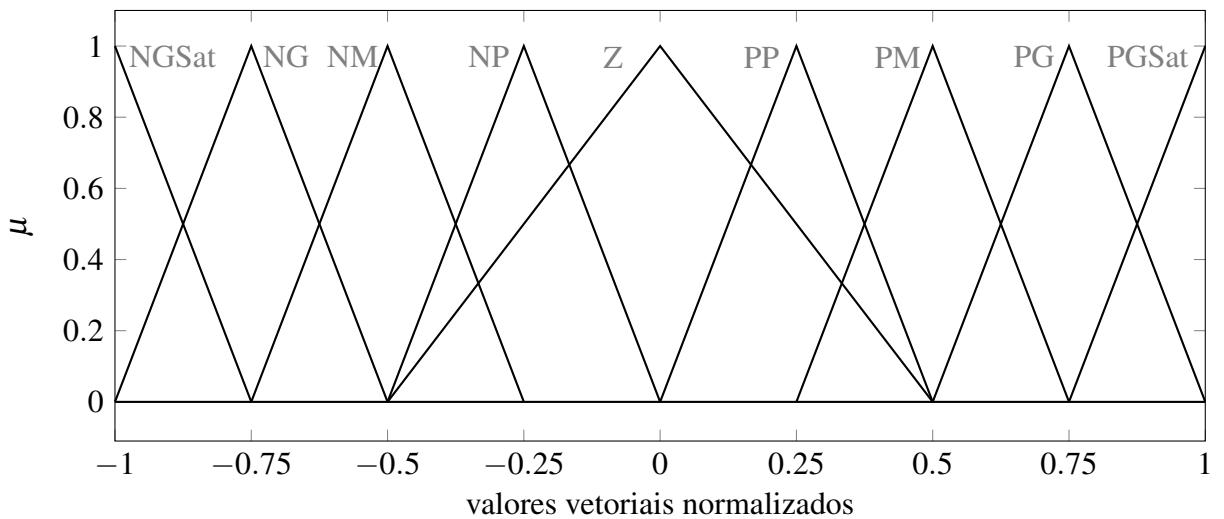
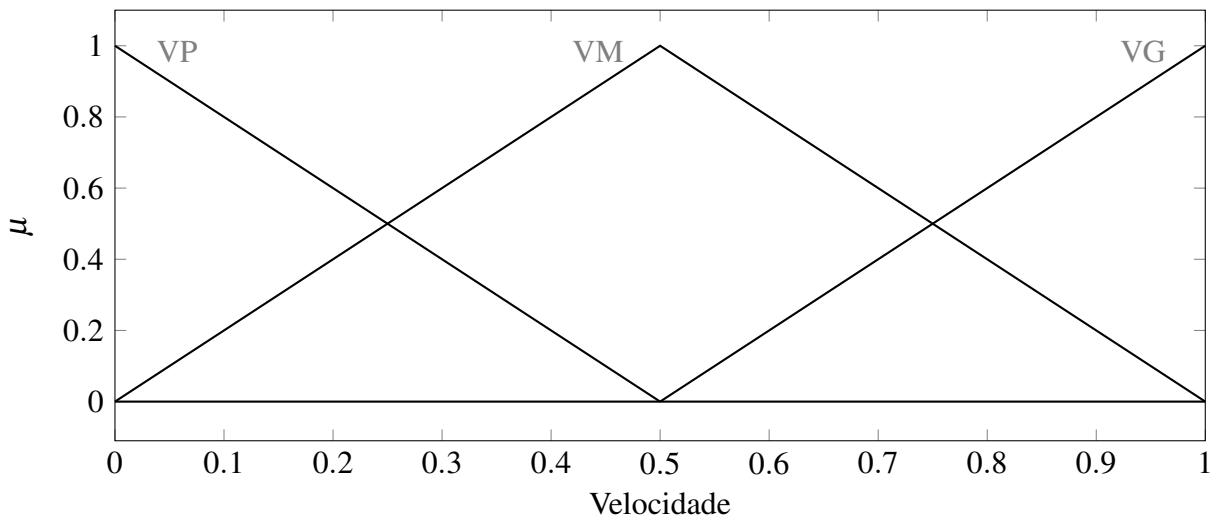
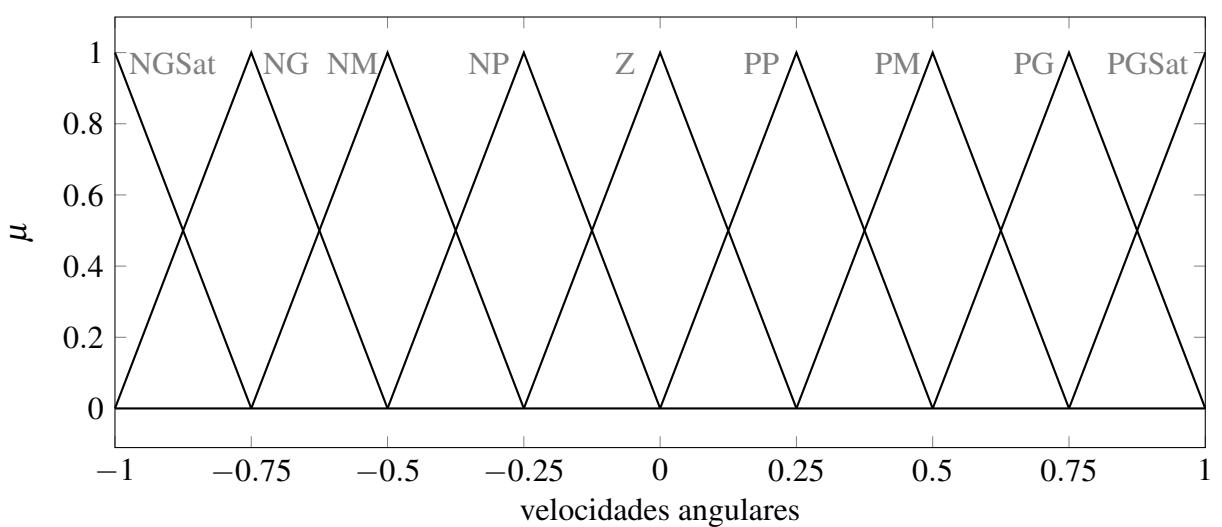
Figura 34 – Conjuntos Fuzzy para as entradas vetoriais**Fonte: autoria própria**

Figura 35 – Conjuntos Fuzzy para a entrada de velocidade



Fonte: autoria própria

Figura 36 – Conjuntos Fuzzy para as velocidades angulares de saída



Fonte: autoria própria

Tabela 6 – Recomendações para velocidades

X / Y	-4	-3	-2	-1	0	1	2	3	4
-4	1	1	1	1	1	1	1	1	1
-3	1	1	1	1	1	1	1	1	1
-2	1	1	1	1	1	1	1	1	1
-1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1

(a) Recomendacao para velocidade VP

X / Y	-4	-3	-2	-1	0	1	2	3	4
-4	2	2	2	2	2	2	2	2	2
-3	2	1	1	1	1	1	1	1	2
-2	2	1	1	1	1	1	1	1	2
-1	2	1	1	1	1	1	1	1	2
0	2	1	1	1	0	1	1	1	2
1	2	1	1	1	1	1	1	1	2
2	2	1	1	1	1	1	1	1	2
3	2	1	1	1	1	1	1	1	2
4	2	2	2	2	2	2	2	2	2

(b) Recomendacao para velocidade VM

X / Y	-4	-3	-2	-1	0	1	2	3	4
-4	3	3	3	3	3	3	3	3	3
-3	3	2	2	2	2	2	2	2	3
-2	3	2	2	2	2	2	2	2	3
-1	3	2	2	1	1	1	2	2	3
0	3	2	2	1	0	1	2	2	3
1	3	2	2	1	1	1	2	2	3
2	3	2	2	2	2	2	2	2	3
3	3	2	2	2	2	2	2	2	3
4	3	3	3	3	3	3	3	3	3

(c) Recomendacao para velocidade VG

X / Y	-4	-3	-2	-1	0	1	2	3	4
-4	4	4	4	4	4	4	4	4	4
-3	4	4	4	3	3	3	3	3	3
-2	4	4	3	2	2	2	2	2	2
-1	4	3	2	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0
1	-4	-3	-2	-1	-1	-1	-1	-1	-1
2	-4	-4	-3	-2	-2	-2	-2	-2	-2
3	-4	-4	-4	-3	-3	-3	-3	-3	-3
4	-4	-4	-4	-4	-4	-4	-4	-4	-4

(d) Recomendação para velocidade angular

5 CONSIDERAÇÕES FINAIS

Para concluir esta etapa do trabalho, algumas deliberações precisam ser feitas. Por enquanto, uma parte do trabalho foi completamente desconsiderada: estimativa do estado do sistema. Para simulação, muitos trabalhos acadêmicos consideram o estado atual do robô como um valor conhecido. Para um robô real, autônomo, este estado pode ser estimado usando técnicas como observadores, filtro de Kalman, odometria, entre outras.

É importante salientar que há um conflito entre os objetivos deste trabalho, já que um robô ao mesmo tempo autônomo e sem armazenamento de mapa deve possuir a capacidade de estimar com perfeição seu estado atual. Na prática, a estimativa de estado carrega erros que são somados no decorrer do tempo de execução. Para corrigir tais erros, ou o robô deve armazenar mapa (representação interna do mundo à sua volta) e reconhecer locais já visitados, ou o robô precisa de um sistema externo (computador com câmera, sistema GPS) que envie correções (isso reduz autonomia).

A segunda solução pode ser adotada neste trabalho, a fim de torná-lo mais focado no aspecto de controle e navegação, o que possibilitará explorar as diferentes abordagens estipuladas (controle híbrido e *fuzzy*) com maior aproveitamento.

Desta forma, uma comparação qualitativa será efetuada. Esta análise deve ser qualitativa pois não existe um padrão de teste (*benchmark*) amplamente adotado em robótica móvel. Logo, qualquer análise comparativa só pode ser qualitativa.

REFERÊNCIAS

- ARKIN, R. C. **Behavior-Based Robotics**. [S.l.]: MIT Press, 1998. (Intelligent Robotics and Autonomous Agents Series). ISBN 9780262529204.
- ASTOLFI, A. Exponential stabilization of nonholonomic systems via discontinuous control. **IFAC Proceedings Volumes**, v. 28, n. 14, p. 661 – 666, 1995. ISSN 1474-6670. 3rd IFAC Symposium on Nonlinear Control Systems Design 1995, Tahoe City, CA, USA, 25-28 June 1995. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1474667017469047>>. Acesso em: 1 maio 2019.
- BANAVAR, R. N.; SANKARANARAYANAN, V. **Switched Finite Time Control of a Class of Underactuated Systems**. [S.l.]: Springer-Verlag Berlin Heidelberg, 2006. (Lecture Notes in Control and Information Sciences, v. 333).
- BIESEK, L. J. **Veículo autônomo: uma contribuição para estacionamento**. 50 f. Monografia (Trabalho de Conclusão de Curso (Graduação)) — Universidade Tecnológica Federal do Paraná, Pato Branco, 2016.
- CARONA, R.; AGUIAR, A. P.; GASPAR, J. Control of unicycle type robots tracking, path following and point stabilization. In: **Proc. of IV Jornadas de Engenharia Electrónica e Telecomunicações e de Computadores**. [S.l.: s.n.], 2008.
- CASSANDRAS, C. G.; LAFORTUNE, S. **Introduction to Discrete Event Systems**. 2. ed. [S.l.]: Springer US, 2008. ISBN 978-0-387-33332-8.
- CROIX, J. P. de la. **Quickbot project**. 2013. Disponível em: <<http://jpdelacroix.com/software/simiam.html>>. Acesso em: 21 maio 2019.
- CURY, J. E. R. **Teoria de Controle Supervisório de Sistemas a Eventos Discretos**. [S.l.], nov. 2001. In: V Simpósio Brasileiro de Automação Inteligente.
- DIB, A. **Commande non linéaire et asservissement visuel de robots autonomes**. Tese (Theses) — Supélec, out. 2011. Disponível em: <<https://tel.archives-ouvertes.fr/tel-00657022>>. Acesso em: 25 abr. 2019.
- DUKOR, K. F.; IGWEGBE, G.; KADIRI, D. **Quickbot project**. 2017. Disponível em: <<https://www.hackster.io/quickbot-team/quickbot-1f93e3>>. Acesso em: 8 abr. 2019.
- DYNAPAR. **Quadrature Encoder Overview**. 2020. Disponível em: <https://www.dynapar.com/technology/encoder_basics/quadrature_encoder/>. Acesso em: 2 nov. 2020.
- GERSTEDT, M. Behavior based robotics using hybrid automata. In: **Proceedings of the Third International Workshop on Hybrid Systems: Computation and Control**. [S.l.: s.n.], 2000.

- FUCHS, T. et al. **Using PySimiam in Coursera ‘Control of mobile robots’ course**. 2013. Disponível em: <<http://pysimiam.sourceforge.net/coursera.html>>. Acesso em: 21 maio 2019.
- GEORGIA ROBOTICS AND INTELLIGENT SYSTEMS LABORATORY. **Sim.I.am**. 2013. Disponível em: <<http://gritslab.gatech.edu/home/2013/10/sim-i-am/>>. Acesso em: 8 abr. 2019.
- GLOTFELTER, P.; EGERSTEDT, M. A parametric mpc approach to balancing the cost of abstraction for differential-drive mobile robots. In: **2018 IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2018. p. 732–737.
- HALLIDAY, D.; RESNICK, R.; WALKER, J. **Fundamentos De Física - Volume 1 - Mecânica**. 9^a. ed. [S.l.]: LTC, 2012. ISBN 9788521630357.
- JAZAR, R. N. **Advanced Dynamics: Rigid Body, Multibody, and Aerospace Applications**. 1^a. ed. [S.l.]: John Wiley, 2011. ISBN 9780470398357.
- JOHAN, K. A.; MURRAY, R. M. **Feedback Systems An Introduction for Scientists and Engineers**. 2^a. ed. [S.l.]: Princeton University Press, 2021. ISBN 0691193983.
- KWON, J.-W.; KIM, C.-J.; CHWA, D. Vector field trajectory tracking control for wheeled mobile robots. **2009 IEEE International Conference on Industrial Technology**, p. 1–6, fev. 2009.
- LAVALLE, S. **Planning Algorithms**. [S.l.]: Cambridge University Press, 2006. ISBN 9781139455176.
- LILLY, J. **Fuzzy Control and Identification**. [S.l.]: Wiley, 2011. ISBN 9781118097816.
- LOPES, W. A. **Projeto e implementação de robô autônomo seguidor de linha**. 114 f. Monografia (Trabalho de Conclusão de Curso (Graduação)) — Universidade Tecnológica Federal do Paraná, Pato Branco, 2017.
- MAHESHWARI, A.; SMID, M. **Introduction to Theory of Computation**. School of Computer Science Carleton University, 2019. Disponível em: <<https://cglab.ca/michiel/TheoryOfComputation/TheoryOfComputation.pdf>>. Acesso em: 7 maio 2019.
- MARCHI, J. Dissertação (mestrado), **Navegação de robôs móveis autônomos: estudo implementação de abordagens**. Florianópolis: [s.n.], 2001. 119 f. Centro Tecnológico. Programa de Pós-Graduação em Engenharia Elétrica.
- MARINHO, D. L. **Aperfeiçoamento de um robô de sumô autônomo**. 67 f. Monografia (Trabalho de Conclusão de Curso (Graduação)) — Universidade Tecnológica Federal do Paraná, Pato Branco, 2017.
- MATARIC, M. J. **Introdução à Robótica**. 1^a. ed. [S.l.]: Editora Unesp, 2014. ISBN 9788539304905.
- MENDEL, J. M. Fuzzy logic systems for engineering: a tutorial. **Proceedings of the IEEE**, v. 83, n. 3, p. 345–377, mar. 1995. ISSN 0018-9219.
- NETO, A. M. Dissertação (mestrado), **Navegação de robôs autônomos baseada em monovisão**. Campinas: [s.n.], 2007. 110 f. Faculdade de Engenharia Mecânica. Programa de Pós-Graduação em Engenharia Mecânica.

NOVEL, B.; CAMPION, G.; BASTIN, G. Control of nonholonomic wheeled mobile robots by state feedback linearization. **I. J. Robotic Res.**, v. 14, p. 543–559, dez. 1995.

OLSON, E. A primer on odometry and motor control. MIT, p. 1–15, jun. 2009.

ORIOLO, G. Wheeled robots. In: BAILLIEUL, J.; SAMAD, T. (Ed.). **Encyclopedia of Systems and Control**. London: Springer London, 2013. p. 1–9. ISBN 978-1-4471-5102-9. Disponível em: <https://doi.org/10.1007/978-1-4471-5102-9_178-1>. Acesso em: 1 maio 2019.

OTTONI, G. de L. **Planejamento de Trajetórias para Robôs Móveis**. 82 f. Monografia (Trabalho de Conclusão de Curso (Graduação)) — Universidade Federal do Rio Grande, Rio Grande, 2000.

PASSINO, K.; YURKOVICH, S. **Fuzzy Control**. [S.l.]: Addison-Wesley, 1998. ISBN 9780201180749.

PETRY, M. L. **Controle híbrido de um robô autônomo seguidor de linha**. 82 f. Monografia (Trabalho de Conclusão de Curso (Graduação)) — Universidade Tecnológica Federal do Paraná, Pato Branco, 2016.

ROSS, T. J. **Fuzzy Logic with Engineering Applications**. [S.l.]: Wiley, 2009. ISBN 9780470748510.

SHARP CORPORATION. **GP2U0A21YK0F**. [S.l.], Dezembro de 2006.

SIEGWART, R.; NOURBAKHSH, I. R. **Introduction to Autonomous Mobile Robots**. [S.l.]: Bradford Company, 2004. ISBN 026219502X.

WANG, J. **A Khepera III robot at the Georgia Institute of Technology**. 2008. Disponível em: <https://en.wikipedia.org/wiki/Khepera_mobile_robot#/media/File:Khepera_III_robot.jpg>. Acesso em: 21 maio 2019.

YAHMEDI, A. A.; FATMI, M. A. Fuzzy logic based navigation of mobile robots. In: TOPALOV, A. E. (Ed.). **Recent Advances in Mobile Robotics**. InTech, 2011. p. 287–310. ISBN 978-953-307-909-7. Disponível em: <<http://www.intechopen.com/books/recent-advances-in-mobile-robotics/fuzzy-logic-based-navigation-ofmobile-robots>>. Acesso em: 28 abr. 2019.

YUN, X.; TAN, K.-C. A wall-following method for escaping local minima in potential field based motion planning. In: **1997 8th International Conference on Advanced Robotics. Proceedings. ICAR'97**. [S.l.: s.n.], 1997. p. 421–426. ISBN 0-7803-4160-0.

YUN, X.; YAMAMOTO, Y. **On Feedback Linearization of Mobile Robots**. 1992. 18 p. Disponível em: <https://repository.upenn.edu/cgi/viewcontent.cgi?article=1524&context=cis_reports>. Acesso em: 20 maio 2019.