Universidade Federal de Sergipe
Programa de Pós-graduação em Engenharia Elétrica

ARQUITETURA DE SUBSUNÇÃO BASEADA EM OBJETIVO DE CONTROLE PRINCIPAL

PHILLIPE CARDOSO SANTOS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica (PROEE) da Universidade Federal de Sergipe, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientadores: Elyson Ádan Nunes Carvalho
Lucas Molina

São Cristóvão
Fevereiro de 2017

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
COORDENAÇÃO DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Ata da Sessão Solene de Defesa da 33ª Dissertação do Curso de Mestrado em Engenharia Elétrica – PROEE/UFS. Candidato: Phillipe Cardoso Santos.

Aos dezessete dias do mês de fevereiro do ano de dois mil e dezessete, com início às 15h00min, realizou-se na Sala de Videoconferência do PROEE da Universidade Federal de Sergipe, na Cidade Universitária Prof. José Aloísio de Campos, a Sessão Pública de Defesa de Dissertação de Mestrado do candidato **Phillipe Cardoso Santos**, que desenvolveu o trabalho intitulado: **"Arquitetura de Subsunção Baseada em Objetivo de Controle Principal"**, sob a orientação do Prof. Dr. Elyson Ádan Nunes Carvalho e coorientação do Prof. Dr. Lucas Molina (PROEE/UFS). A Sessão foi presidida pelo Prof. Dr. Eduardo Oliveira Freire (PROEE/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, Prof. Dr. Jânio Coutinho Canuto (PROEE/UFS), e em seguida ao Prof. Dr. José Gilmar Nunes de Carvalho Filho (UFS). Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando *(aprovado/reprovado)* "APROVADO" *(com/sem)* "SEM" ressalvas. Atendidas as exigências da Instrução Normativa 07/2014/PROEE, do Regimento Interno do PROEE (Resolução 76/2015/CONEPE), e da Resolução nº 25/2014/CONEPE que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária *"Prof. José Aloísio de Campos"*, 17 de fevereiro de 2017.

Prof. Dr. Eduardo Oliveira Freire (PROEE/UFS)
Presidente

Prof. Dr. Jânio Coutinho Canuto (PROEE/UFS)
Examinador Interno

Prof. Dr. José Gilmar Nunes de Carvalho Filho (UFS)
Examinador Externo

Phillipe Cardoso Santos
Candidato

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

Santos, Phillipe Cardoso
S237a Arquitetura de subsunção baseada em objetivo de controle principal / Phillipe Cardoso Santos ; orientador Elyson Ádan Nunes Carvalho. – São Cristóvão, 2017.
76 f. ; il.

Dissertação (mestrado em Engenharia elétrica) – Universidade Federal de Sergipe, 2017.

1. Engenharia elétrica. 2. Robótica. 3. Robôs móveis. 4. Estabilidade. 5. Teoria dos autômatos. I. Carvalho, Elyson Ádan Nunes, orient. II. Título.

CDU: 621.3:007.52

A meu pai Ronaldo dos Santos
e a minha mãe Siziana
Alcântara Cardoso.

Agradecimentos

Gostaria de agradecer primeiramente a Deus por sempre ter guiado meus passos, iluminado meu caminho e por ter colocado em minha vida todas as pessoas que contribuiram e/ou torceram para eu que pudesse chegar até aqui.

Aos meus pais Ronaldo e Sizi, que sempre me deram todo o apoio que precisei e nunca mediram esforços para que eu pudesse alcançar meus objetivos. Meu pai, que mudou nossas vidas através do estudo, sempre foi um espelho pra mim. Minha mãe, com seu coração gigante e jeito especial de dar bronca (rsrs), me ensinou a ser uma pessoa boa, respeitando e ajudando o próximo. Juntos, vocês me ensinaram a ser o que eu sou hoje. Sem vocês eu nunca teria chegado até aqui. Amo vocês!

Aos meus avós José de Sá e Dejenalva que estão sempre por perto, dando total apoio e torcendo muito por mim. A minha avó Marinita (*in memoriam*), uma pessoa muito batalhadora e que foi muito importante na minha infância.

Ao meu irmão Matheus, que tem dado muito orgulho para nossa família, sendo um menino bom e com um futuro brilhante pela frente. É sempre bom ter um irmão com o qual você pode contar, compartilhar tudo, sem brigas ou qualquer tipo de desentendimento.

Ao meu grande amigo e irmão Thiago, por todas as conversas, conselhos, apoio, torcida e parceria desde os tempos do infantil. Valeu mesmo man!

A Aline, que me ajudou a amadurecer muito durante essa caminhada, a descobrir que eu posso sonhar mais alto e a ser uma pessoa melhor. Muito obrigado por tudo.

A todos os meus tios e tias, em especial a Júnior Sá (o Doido) por ser praticamente um irmão mais velho, a todos os primos, primas, minhas duas madrinhas, meus afilhados... enfim, a toda a minha família, que sempre torceu por mim e vibrou a cada conquista. Uma família abençoada por Deus e da qual me orgulho de fazer parte.

Aos meus orientadores Elyson, Lucas e Eduardo que me apresentaram a robótica e me fizeram ter a certeza de que meu sonho é seguir a carreira acadêmica. Muito obrigado por tudo, desde todo o conhecimento transmitido, até os conselhos, broncas e elogios que com certeza me farão um profissional mais capacitado e também uma pessoa melhor.

À todos que fizeram ou fazem parte da família GPRUFS desde 2010, a convivência com vocês foi de fundamental importância para a realização não só desta dissertação, como de muitos outros trabalhos. Em especial, gostaria de agradecer à Stephanie, que acabou se tornando uma grande amiga com a qual pude dividir toda pressão da graduação

e do mestrado, filosofar sobre a vida e dividir alegrias e conquistas, sendo este mestrado a mais recente delas, mas que, com fé em Deus, será apenas mais uma de muitas que ainda virão. Gostaria de agradecer também mais especificamente à Lívia, Renato, Marcelo e Arthur.

Ao pessoal que entrou comigo na UFS na turma de 2009 e que me ajudou a superar as dificuldades do curso, principalmente a Lewis, Dami, Tarciana, Isla e, claro, a Mateus (o Galo), parceiro na maioria dos projetos na graduação e um amigo que levarei pra toda a vida.

À galera do Irradiar, aos colegas que fiz no Ciência sem Fronteiras e aos demais amigos, dentro os quais gostaria de citar Marcos, Bruno, Makson, Roberta, Rafael e Judá.

Por fim, gostaria de agradecer à CAPES pelo apoio financeiro que foi de fundamental importância para a realização deste trabalho.

Resumo

ARQUITETURA DE SUBSUNÇÃO BASEADA EM OBJETIVO DE CONTROLE PRINCIPAL

PHILLIPE CARDOSO SANTOS

Fevereiro/2017

Orientadores: Elyson Ádan Nunes Carvalho
Lucas Molina

Departamento: Engenharia Elétrica (DEL/PROEE/UFS)

Um aspecto muito importante na robótica é a tomada de decisão e execução que o sistema utiliza para alcançar seus objetivos. Na literatura, existem vários trabalhos diferentes para abordar como o robô deve se comportar diante de várias situações diferentes a fim de trazer uma maior robustez ao sistema, sendo a arquitetura de subsunção uma das mais utilizadas e referenciadas na área. Nesta arquitetura, a tarefa global é dividida em subtarefas que são executadas por comportamentos organizados em camadas de forma hierárquica. No entanto, pouco se pesquisa no que diz respeito a análise de estabilidade desta arquitetura, sendo que as mudanças de comportamento implicam em chaveamento de controladores, que por sua vez podem levar o sistema a instabilidade mesmo em casos em que todos os controladores sejam estáveis. Desta forma, neste trabalho é apresentada uma arquitetura de subsunção com prova de estabilidade garantida com base na teoria de controle chaveado com objetivo de controle principal. Além disso, um formalismo capaz de permitir a modelagem dos comportamentos de forma simples e rápida é proposto com base na teoria de sistemas a eventos discretos. Testes em ambientes reais foram realizados com o robô Pioneer P3-DX e os resultados obtidos comprovam a eficácia da abordagem proposta.

Palavras-chave: Navegação de robôs móveis; Arquitetura de subsunção; Estabilidade; Controle chaveado; Sistemas a eventos discretos.

Abstract

SUBSUMPTION ARCHITECTURE BASED ON MAIN CONTROL OBJECTIVE

PHILLIPE CARDOSO SANTOS

February/2017

Advisors: Elyson Ádan Nunes Carvalho
Lucas Molina

Department: Electrical Engineering (DEL/PROEE/UFS)

A very important aspect in robotics is the decision making and execution the system uses to achieve its goals. In literature, many different approaches can be found about how the robot must behave in different situations in order to have a more robust system. Subsumption architecture is one of the most used and referenced in the area. In this architecture, the global task is divided into subtasks which are performed by behaviors organized in hierarchical layers. However, little research has been done regarding the stability analysis of this architecture. Behavioral changes imply in controller switching, which can lead the system to instability even in cases where all controllers are stable. In this work, a subsumption architecture with guaranteed stability is presented based on the theory of switched systems with main control objective. In addition, a formalism capable of allowing behaviors modeling in a simple and fast way is proposed based on the theory of discrete events systems. Tests in real environments were performed with the Pioneer P3-DX robot and obtained results demonstrate the proposed approach effectiveness.

Keywords: Mobile robots navigation ; Subsumption architecture; Stability; Control switching; Discrete event systems.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Na robótica móvel são estudados os dispositivos eletromecânicos que são capazes de se movimentar e interagir com o ambiente estabelecido. Os robôs costumam ser utilizados com várias finalidades diferentes, desde entretenimento até a realização de processos de automação industrial, ou mesmo na exploração de ambientes inóspitos ou que apresentam algum tipo de perigo para a integridade humana.

Um aspecto muito importante na robótica é a tomada de decisão e execução que o sistema utiliza para alcançar seus objetivos. No caso da robótica móvel, o objetivo específico que está diretamente ligado à mobilidade é a competência de navegação [6]. Nesse sentido, navegação pode ser descrita como o processo de determinar um caminho adequado e seguro entre um ponto de partida e um ponto de destino, levando em consideração as informações disponíveis para o robô [7].

Nesse contexto, são pesquisadas as arquiteturas de navegação com o propósito de solucionar o problema da tomada de decisões pelo robô móvel. A arquitetura de um robô define como está organizada a tarefa de gerar ações a partir das percepções e das informações conhecidas a priori [8].

Segundo Ronald C. Arkin [9], em relação às suas metodologias, as arquiteturas de navegação podem ser qualitativamente classificadas de acordo com o modo que o sistema age mediante os estímulos do ambiente. Nesse sentido, as arquiteturas podem ser separadas em três tipos, quais sejam: (*i*) deliberativas (ou cognitivas), (*ii*) reativas (ou reflexivas) e (*iii*) híbridas.

O caráter deliberativo de um sistema de navegação permite ao robô móvel construir um modelo do ambiente onde são consideradas, principalmente, as posições e formas dos obstáculos, além do espaço livre navegável. Dessa maneira, sendo conhecido o ambiente de navegação, o robô pode gerar a trajetória de locomoção através de uma técnica de planejamento de movimento, escolhida a critério do projetista na implementação do sistema.

Já o caráter reativo de um sistema de navegação possibilita ao robô uma resposta em tempo real aos estímulos do ambiente, uma vez que esse tipo de arquitetura é caracterizada

por mapear diretamente a leitura dos sensores em ações do robô [9]. Nesta abordagem, não é necessário o conhecimento prévio do ambiente, o que torna essas arquiteturas mais indicadas para a navegação em ambientes dinâmicos, pois o robô pode reagir rapidamente às mudanças que ocorrerem.

A partir das arquiteturas de navegação citadas anteriormente, surge a navegação híbrida [10] [11] [12], a qual une o planejamento e a eficiência da arquitetura deliberativa à velocidade e independência da arquitetura reativa, podendo tornar então, a navegação híbrida mais eficiente. Normalmente a parte reativa é mais utilizada em situações de emergência, enquanto a deliberativa é responsável pelo planejamento global da tarefa, para que a meta determinada possa ser alcançada.

Na área de navegação é comum decompor uma tarefa complexa (tarefa global) em diversas tarefas mais simples (sub-tarefas), sendo estas sub-tarefas realizadas através de diferentes sistemas de navegação independentes, chamados de comportamentos [9].

A coordenação desses comportamentos é um dos principais aspectos relacionados à navegação e um dos trabalhos mais importantes e referenciados nessa área é a arquitetura de subsunção proposta por Brooks em [3]. Neste trabalho, Brooks criou um conjunto de comportamentos direcionados a cumprir tarefas, organizados por camadas que são acionadas a partir de estímulos a resposta de um conjunto de sensores. Cada comportamento aceita sinais de supressão e inibição, sendo que um sinal de supressão sobrepõe o sinal normal de entrada e o sinal de inibição inibe completamente a saída. Estes sinais permitem que os comportamentos se sobreponham, de forma que o sistema possa produzir um comportamento coerente.

Além disso, muitos comportamentos podem ser disparados simultaneamente, tornando-se necessário um mecanismo de escolha que seleciona a ação a ser executada. Para isso, os módulos comportamentais são organizados em camadas em uma hierarquia chamada de subsunção, em que as camadas mais baixas são inibidas pelas camadas mais altas.

Com a arquitetura de subsunção, Brooks obteve bons resultados que fizeram com que seu trabalho continue sendo muito utilizado até os dias atuais, mesmo com a vasta pesquisa que é realizada na área de navegação e a consequente criação de novas técnicas. Na figura 1.1 pode ser vista a quantidade de trabalhos que citam o trabalho de Brooks por ano desde a sua publicação.

A arquitetura de subsunção de Brooks, assim como outras arquiteturas baseadas em comportamento, trabalham selecionando uma ação adequada para cada momento e isto é feito através do chaveamento de comportamentos, normalmente representados pelo chaveamento de controladores. A regra aplicada para esse chaveamento é de fundamental importância para o sistema, uma vez que a utilização de vários controladores não garante por si só a estabilidade do mesmo. Isto fica evidenciado no fato de que um sistema formado apenas por subsistemas globalmente assintoticamente estáveis pode se tornar

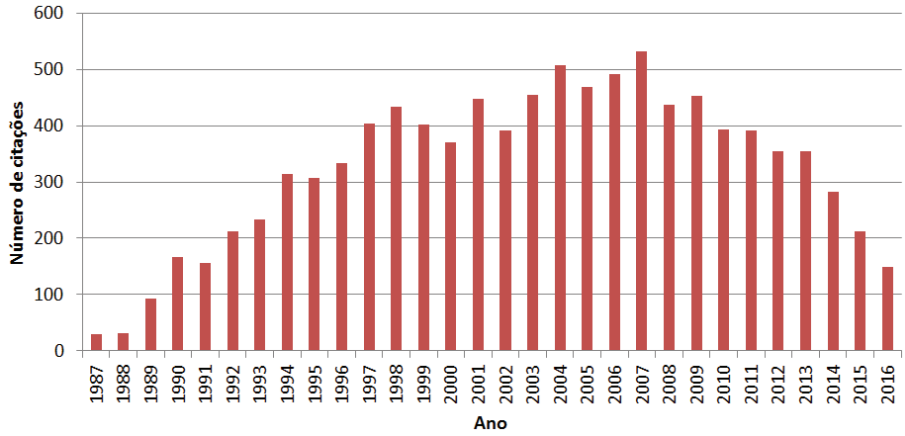


Figura 1.1: Número de citações de por ano (dados do Google Scholar^{TM}).

instável, mesmo no caso em que todos os subsistemas tenham o mesmo ponto de equilíbrio. Por outro lado, um sistema chaveado formado apenas por subsistemas instáveis pode se tornar globalmente assintoticamente estável a depender da regra de chaveamento empregada [1].

Portanto, faz-se necessário definir um conjunto de regras para o chaveamento para que a estabilidade do sistema possa ser garantida.

No entanto, mesmo com toda a importância da análise de estabilidade de sistemas chaveados, poucos trabalhos da área de navegação de robôs móveis abordam essa questão. Esse fato ocorre principalmente devido à complexidade da análise de estabilidade, principalmente nos sistemas não lineares. Brooks, inclusive, cita em [13] que a análise da estabilidade não depende de equações simples e por este motivo ele não realiza a análise de estabilidade da arquitetura de subsunção que ele desenvolveu.

Em 2006, Harper and Winfield [4] apresentaram o primeiro trabalho a desenvolver um formalismo matemático para prova de estabilidade da arquitetura de subsunção. Eles propuseram um formalismo baseado em duas extensões da teoria de estabilidade de Lyapunov, que eles denominaram de teoremas de estabilidade de segunda ordem.

Mesmo depois do trabalho publicado por Harper and Winfield [4], nenhum trabalho foi encontrado com tal análise. No entanto, um teorema proposto por Carvalho em [1] apresenta uma metodologia factível para a análise de estabilidade de um sistema baseado em subsunção. O teorema prova a estabilidade de sistemas chaveados onde o objetivo principal é governado por um controlador globalmente assintoticamente estável segundo Lyapunov, mas os outros controladores podem ser até mesmo instáveis. Essa metodologia garante, através de uma regra de chaveamento, que o objetivo principal do sistema será cumprido.

A ideia que o subsistema principal tem uma maior prioridade em relação aos demais subsistemas e que sua convergência é garantida pela regra de chaveamento imposta, torna possível a aplicação dessa técnica para prova de estabilidade da arquitetura de subsunção.

Portanto, neste trabalho é apresentada uma nova abordagem para implementação da arquitetura de subsunção com prova de estabilidade garantida com base na teoria de controle chaveado com objetivo de controle principal. Nesta abordagem, a estabilidade do sistema é garantida de forma mais abrangente do que na teoria proposta por Harper e Winfield.

Para que isso seja possível, o chaveamento dos controladores deve obedecer certas regras impostas pela teoria de objetivo de controle principal. O cumprimento das mesmas é garantido através da teoria de sistemas a eventos discretos, com a modelagem dos comportamentos do sistema desenvolvida na forma de autômatos.

Com essa metodologia, além de provar a estabilidade do sistema, é fornecido também um formalismo capaz de permitir a modelagem dos comportamentos de forma simples e rápida, mesmo para sistemas que possuem muitos comportamentos, o que não ocorre na maioria dos sistemas baseados na arquitetura de subsunção, onde o desenvolvimento da máquina de estados que governa o sistema passa a ser muito complicado para sistemas mais complexos. A metodologia proposta neste trabalho ainda permite a inclusão de novos comportamentos sem a necessidade de alterar os comportamentos já existentes, garantindo ainda a validade da análise de estabilidade.

Este trabalho está estruturado da seguinte forma. No capítulo 2 será apresentada a fundamentação teórica de sistemas chaveados e de sistemas a eventos discretos necessárias para o desenvolvimento deste trabalho. No capítulo 3 será apresentada a revisão bibliográfica com os principais trabalhos baseados na arquitetura de subsunção. No capítulo 4 será apresentado todo o desenvolvimento da metodologia proposta. Os resultados serão apresentados no capítulo 5. E por fim, no capítulo 6 serão apresentados as conclusões e os trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo são apresentadas as ferramentas teóricas que são úteis para analisar e garantir a estabilidade do sistema. Primeiramente, na seção 2.1, são apresentadas as definições de estabilidade e *quasi*-estabilidade, bem como o teorema de Lyapunov. Em seguida, na seção 2.2, é apresentada a representação matemática de um sistema chaveado. Na seção 2.3 é mostrada a ideia dos sistemas chaveados baseados em objetivo de controle principal. E por fim, na seção 2.4 são apresentadas as ferramentas de sistemas a evento discretos utilizadas para formalizar a regra de chaveamento proposta em [1], que garantirá a *quasi*-estabilidade do sistema.

2.1 Estabilidade e *quasi*-Estabilidade

Nesta seção será considerado o sistema autônomo

$$\dot{x} = f(x), \tag{2.1}$$

em que $f : D \to \Re^n$ é um mapa localmente Lipschitz em um domínio $D \subset \Re^n$ em $\Re^n$ e x o vetor de dimensão n de variáveis de estado do sistema.

Considerando $x = 0$ como o ponto de equilíbrio do sistema, uma vez que qualquer ponto de equilíbrio pode, sem perda de generalidade, ser transladado para a origem. Este ponto é definido em [14] como:

- Estável se, para todo $\varepsilon > 0$, existe um $\delta = \delta(\varepsilon) > 0$ tal que

$$||x(0)|| < \delta \Rightarrow ||x(t)|| < \varepsilon, \forall t \geq 0; \tag{2.2}$$

- Instável se não é estável;

- Assintoticamente estável se é estável e δ pode ser escolhido de tal forma que

$$||x(0)|| < \delta \Rightarrow lim_{t\to\infty} x(t) = 0. \tag{2.3}$$

Segundo [15], esta definição é muito restritiva e pode inviabilizar a demonstração de estabilidade de muitos sistemas encontrados na prática. Dessa forma, em [15] é proposta a definição de *quasi*-estabilidade assintótica e em [1] é definida a *quasi*-estabilidade. Ambos os termos são utilizados para definir sistemas que só atendem a definição clássica de estabilidade após um tempo $t \geq \tau$. Isso significa que um sistema pode se comportar de qualquer forma durante um tempo $t < \tau$, mas se a partir deste tempo τ o sistema atender os critérios apresentados em [14], ele será *quasi*-estável ou *quasi*-assintoticamente estável.

Uma maneira muito utilizada na literatura para analisar a estabilidade de um ponto de equilíbrio é através da "função de energia" do sistema. Desde que a função de energia seja de valor zero apenas no equilíbrio, quando a derivada desta função é menor ou igual a zero durante todo tempo, a energia do sistema tende a decair até que eventualmente chegue a zero. Desta maneira a trajetória tende para $x = 0$ quando t tende para ∞.

Em 1892, Lyapunov mostrou através do seguinte teorema que é possível determinar a estabilidade do sistema utilizando outras funções além da função de energia [14]:

Teorema 1 (Teorema de Lyapunov): *Seja $x = 0$ um ponto de equilíbrio de $\dot{x} = f(x)$ e $D \subset \Re^n$ um domínio contendo $x = 0$. Seja $V(x)$ uma função continuamente diferenciável que mapeia o domínio nos reais $(D \to \Re)$ de tal maneira que:*

$$V(0) = 0 \text{ e } V(x) > 0 \text{ em } D - \{0\}; \tag{2.4}$$

$$\dot{V}(x) \leq 0. \tag{2.5}$$

Então, $x = 0$ é estável. Adicionalmente, se

$$\dot{V}(x) < 0 \text{ em } D - \{0\}, \tag{2.6}$$

então, $x = 0$ é assintoticamente estável.

A aplicação do teorema de Lyapunov torna possível a análise da estabilidade do sistema sem a necessidade de resolver a equação diferencial $\dot{x} = f(x)$ e, além disso, a função de Lyapunov $V(x)$ pode ser muito simples, facilitando bastante a análise do sistema. Por outro lado, nem sempre existe um método pré-definido para encontrar a função $V(x)$. Na maior parte dos casos, as funções de energia do sistema são adotadas como funções candidatas de Lyapunov, porém, estas funções nem sempre satisfazem as condições do teorema de Lyapunov para estabilidade ou para estabilidade assintótica. Nesses casos, não se pode afirmar que o ponto de equilíbrio não é estável ou assintoticamente estável, respectivamente. Isto significa que a estabilidade do sistema não pode ser estabelecida

através desta função candidata.

Para a análise de estabilidade de sistemas *quasi*-estáveis foram desenvolvidas em [1] condições suficientes do tipo Lyapunov. Estas condições são apresentadas no teorema a seguir.

Teorema 2 (*Quasi-*estabilidade): *Considerando que $x = 0$ é o ponto de equilíbrio de $\dot{x} = f(x)$, e seja $V : \Re^n \to \Re$ uma função contínua, tal que:*

$$V(0) = 0; \tag{2.7}$$

$$V(x) > 0 \ em \ \Re^n - \{0\}; \tag{2.8}$$

$$||x(t)|| \to \infty \Rightarrow V[x(t)] \to \infty. \tag{2.9}$$

Se,

$$\forall t_0 \in \Re_+, \exists \tau \in [0, \infty] \mid \forall t \in [t_0 + \tau, \infty[\Rightarrow V[x(t)] \leq V[x(t_0)] \tag{2.10}$$

em que t e t_0 são medições de tempo, então $\dot{x} = f(x)$ é quasi-*estável.*

Em outras palavras, o teorema quer dizer que para qualquer tempo inicial t_0, se existir um tempo finito τ a partir do qual, para qualquer tempo superior a $t_0+\tau$, a função $V[x(t)]$ for sempre menor ou igual ao valor da função no tempo inicial $V[x(t_0)]$, então $\dot{x} = f(x)$ é *quasi*-estável.

Para os casos em que o valor de $V[x(t)]$ é sempre menor que o valor de $V[x(t_0)]$ a partir de um tempo $t_0 + \tau$, é provado em [1] que $\dot{x} = f(x)$ é *quasi*-estável e $V[x(t)]$ tende a um certo valor L que é limitado entre 0 e $V[x(t_0)]$. Este resultado é particularmente importante quando $L = 0$, pois assim é mostrado em [1] que o sistema é *quasi*-assintoticamente estável. O teorema a seguir apresenta a *quasi*-estabilidade assintótica.

Teorema 3 (*Quasi-*estabilidade assintótica): *Considerando que $x = 0$ é o ponto de equilíbrio de $\dot{x} = f(x)$, e seja $V : \Re^n \to \Re$ uma função contínua, que atenda as equações 2.7,2.8 e 2.9. Se,*

$$\forall t_0 \in \Re_+, \exists \tau \in (0, \infty) | \forall t \in [t_0 + \tau, \infty[\Rightarrow V[x(t)] < V[x(t_0)] \tag{2.11}$$

então $\dot{x} = f(x)$ é quasi-*estável e $V[x(t)] \to L, L \in [0, V[x(t_0)]]$.*

Na figura 2.1 é mostrado um exemplo de chaveamento que atende as condições que foram apresentas.

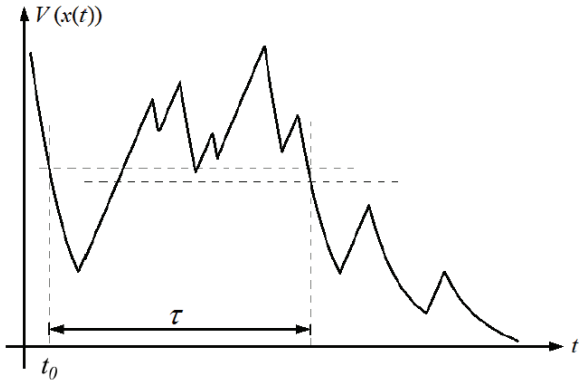


Figura 2.1: Exemplo de função $V[x(t)]$ de um sistema de chaveado que atende às proposições de *quasi*-Estabilidade [1]

2.2 Sistemas Chaveados

Um sistema chaveado é representado matematicamente na forma de equação diferencial,

$$\dot{x} = f_\sigma(x), \tag{2.12}$$

em que $\{f_\chi : \chi \in X\}$ é uma família de funções, de $\Re^n \to \Re^n$, suficientemente regulares, que é parametrizada por algum índice do conjunto X, x é o vetor de dimensão n de variáveis de estado do sistema e $\sigma : [0, \infty) \to X$ é uma função constante por partes chamada de sinal de chaveamento, que indica qual subsistema está ativo em cada instante de tempo. As funções f_χ representam cada subsistema do sistema chaveado.

Outra maneira de representar um sistema de controle chaveado é

$$\dot{x} = \begin{cases} f_1(x) \\ f_2(x) \\ \vdots \\ f_k(x) \end{cases}, \tag{2.13}$$

em que k representa a quantidade de elementos do conjunto f_χ. Esta representação é mais usada quando se tem um número de subsistemas pequeno e se pretende explicitar as funções de cada um deles.

Nos casos em que todos os subsistemas são lineares, o sistema pode ser apresentado na forma

$$\dot{x} = A_\sigma x, \tag{2.14}$$

em que $\{A_\chi : \chi \in X\}$ é uma família de matrizes que definem cada subsistema.

2.3 Sistemas chaveados baseados em objetivo de controle principal

Em [1], um sistema de controle chaveado é dado por

$$\dot{x} = \begin{cases} f_1(x) \\ f_2(x) \\ \vdots \\ f_i(x) \end{cases}, \tag{2.15}$$

sendo $f_i : \Re^n \rightarrow \Re^n$ definido pelo chaveamento das funções $f_{1...i}$, em que $f_1 : \Re^k \rightarrow \Re^k$ é um mapa de Lipschitz de um domínio $\Re^n$ em $\Re^n$, $f_{2...i} : \Re^m \rightarrow \Re^m$ são funções contínuas quaisquer e $x = \{x_1 \cup x_2 \cup \ldots\}$, onde x_1 e $x_{2...i}$ são vetores de dimensão k e m, respectivamente, de variáveis de estado contínuas no tempo.

A ideia de um sistema chaveado baseado em objetivo de controle principal, proposta em [1], consiste em considerar que $f_1(x)$ é o sistema principal, que leva o sistema chaveado a atingir o objetivo principal, ou seja, durante o tempo que o sistema estiver chaveado em $f_1(x)$, ele estará cumprindo o seu objetivo pré-estabelecido principal ou preponderante. Para isso, $f_1(x)$ deve ser globalmente assintoticamente estável segundo Lyapunov, ou seja:

$$V_{1x_1}(0) = 0 \text{ e } V_1(x_1) > 0 \text{ em } D - \{0\}; \tag{2.16}$$

$$\dot{V_1}(x_1) < 0 \text{ em } D - \{0\}; \tag{2.17}$$

$$||x_1(t)|| \rightarrow \infty \Rightarrow V[x_1(t)] \rightarrow \infty. \tag{2.18}$$

Durante o tempo em que o sistema estiver chaveado em $f_{2..i}(x)$, ele estará cumprindo outro objetivo, considerado como secundário. Assim, quando o sistema estiver chaveado em $f_{2..i}(x)$, que são funções que não possuem restrições quanto à estabilidade ou objetivo de controle, é considerado que há uma perturbação em $\dot{x} = f_1(x)$.

Em [1] é mostrado que esta abordagem permite que seja analisada a estabilidade do sistema chaveado em função apenas das k variáveis de estado de $f_1(x_1)$, que devem ser observáveis durante todo tempo. Nesse caso, o sistema terá como candidata SSLF (do inglês, *Switched Systems Lyapunov Function*) $V_1(x)$, e como V_1 é função somente das k variáveis de $\dot{x} = f_1(x)$, $V(x) = V(x_1) = V_1(x_1)$.

Para um sistema chaveado baseado em objetivo de controle principal não é considerado se os objetivos dos controladores secundários serão cumpridos, a proposta deste tipo de abordagem é possibilitar que haja chaveamento para os controladores secundários sem que o sistema deixe de cumprir o objetivo principal.

2.3.1 *Quasi*-estabilidade para sistemas com objetivo de controle principal baseado em chaveamento lento

Esta seção tem como base o teorema proposto em [1] que é classificado no grupo de chaveamento lento de controladores, uma vez que a ocorrência de um chaveamento para um controlador secundário é dependente do estado atual, e não pode ser realizado a qualquer momento [1], possuindo geralmente frequência de chaveamento baixa.

Nesse teorema é estabelecido o quanto a função de Lyapunov do sistema deverá ter caído (ρ) após o tempo τ_{SSLF} para que o sistema seja *quasi*-assintoticamente estável, mesmo quando há chaveamento para outros subsistemas. Esta proposição pode ser vista no teorema a seguir, cuja prova é apresentada em [1].

Teorema 4 (Teorema de objetivo de controle principal): *Considerando um sistema do tipo $\dot{x} = f(x)$, com função candidata SSLF $V_1(x)$, escolhida de acordo com 2.16, 2.17 e 2.18, que atende às condições de* quasi-*estabilidade que foram mostradas no teorema 2, sejam t_i e t_j dois instantes, se $\forall$ i e j, com $t_j - t_i > \tau_{SSLF}$ e $t_j > t_i$, $\exists \rho > 0$, tal que,*

$$V_1[x(t_j)] - V_1[x(t_i)] \leq -\rho||x(t_i)||, \tag{2.19}$$

então o sistema $\dot{x} = f(x)$ é quasi-assintoticamente estável.

Sendo assim, neste trabalho, a regra de chaveamento é modelada tendo como base a mensuração contínua dessa equação de forma que é permitido o chaveamento para outros controladores enquanto a mesma está sendo satisfeita. No momento em que ela não for satisfeita, o sistema é obrigado a voltar para o subsistema principal, fazendo com que a função de energia decresça novamente, uma vez que esse subsistema é globalmente assisntoticamente estável, até que a condição volte a ser satisfeita. Essa abordagem garante que, do ponto de vista global, a função de energia do subsistema principal sempre irá decrescer, mesmo que localmente ela cresça momentaneamente.

Para garantir que essas exigências sejam cumpridas e assim o robô alcance o seu objetivo principal, o sistema será modelado com base na teoria de sistemas a eventos discretos na forma de autômatos. Por esse motivo, na próxima seção serão apresendados os fundamentos de sistemas a eventos discretos necessários para o desenvolvimento deste trabalho.

2.4 Sistemas a Eventos Discretos

Um sistema a eventos discretos (SED) é definido em [16] como um sistema de estado discreto e dirigido por eventos, isto é, a evolução dos seus estados depende exclusivamente da ocorrência de eventos discretos e, em geral, assíncronos em relação à escala de tempo.

Nos sistemas em que o espaço de estados é discreto, as transições de estado ocorrem instantaneamente, sendo estas associadas a um evento. Este pode ser identificado como a

ocorrência de uma ação específica, por exemplo, a detecção de um obstáculo à frente do robô, o início de uma navegação dentro de um corredor ou uma variável atingir um dado *setpoint*.

A teoria de SED possui alguns formalismos que permitem uma melhor representação e modelagem da mesma. Nas subseções seguintes serão apresentadas as ferramentas que serão utilizadas neste trabalho.

2.4.1 Linguagem de um sistema

Um sistema a eventos discretos possui um conjunto de eventos $E = e_1, e_2, ..., e_n$ associados a ele, que recebe o nome de *alfabeto*.

A *linguagem* de um SED é definida como um conjunto de todos os possíveis comportamentos do sistema, sendo que cada comportamento pode ser descrito por uma sequência de eventos que recebe o nome de *palavra*.

Um exemplo de *linguagem* pode ser visto em [2], onde é mostrada uma linguagem L_1 contendo todas as possíveis palavras de comprimento três que comecem com o evento a. Para este exemplo é considerado um conjunto de eventos definidos por $E = a, b, c$. A linguagem L_1 obtida pode ser vista a seguir:

$$L_1 = aaa, aab, aac, aba, abb, abc, aca, acb, acc \tag{2.20}$$

Em [16] são feitas algumas definições para formalizar a linguagem de um sistema. Estas definições são:

1. ***Fechamento Kleene para eventos***: O Fecho de Kleene de um conjunto de eventos E, é o conjunto de todas as palavras de comprimento finito geradas a partir dos elementos de E, incluindo a palavra vazia ε. A notação E^* representa a operação Fecho de Kleene sobre E. Por exemplo, se $E = a, b, c$, então:

   $$E^* = \varepsilon, a, b, c, aa, ab, ac, bb, ba, bc, cc, ca, cb, aaa, ... \tag{2.21}$$

   Assim sendo, a maior linguagem que pode ser definida sobre um conjunto de eventos E é E^* e qualquer outra linguagem é um subconjunto de E^*.

2. ***Concatenação***: Seja $L_a, L_b \subseteq E^*$. Então uma palavra s pertence à concatenação das linguagens L_a e L_b, se ela puder ser escrita como a concatenação de uma palavra pertencente a L_a com outra pertencente a L_b, ou seja:

   $$L_a L_b := s \in E^* : (\exists s_a \in L_a)(\exists s_b \in L_b)[s = s_a s_b]. \tag{2.22}$$

   Por exemplo, considerando as linguagens $L_a = \{aa, ab\}$ e $L_b = \{ba, ab, cba\}$, a

concatenação de L_a com L_b é:

$$L_a L_b = aaba, aaab, aacba, abba, abab, abcba. \tag{2.23}$$

3. ***Prefixo Fechamento***: Seja $L \subseteq E^*$. Então o prefixo fechamento de L, denotado por $\bar{L}$, é a linguagem formada por todos os prefixos de todas as palavras pertencentes a L. Em geral, $L \subseteq \bar{L}$. Assim:

$$\bar{L} := s \in E^* : (\exists t \in E^*)[st \in L]. \tag{2.24}$$

Por exemplo, o prefixo fechamento da linguagem L_b é dado por:

$$\bar{L_b} = \varepsilon, b, ba, a, ab, c, cb, cba. \tag{2.25}$$

Obs.: L é dita de prefixo fechada quando $L = \bar{L}$, ou seja, se qualquer palavra pertencente a L também pertencer a $\bar{L}$.

4. ***Fechamento Kleene para linguagens***: Seja $L \subseteq E^*$. Então o fecho de Kleene de L, denotado por L^*, é o conjunto formado por todas as possíveis concatenações entre palavras pertencentes a L. Essa operação pode ser expressa da seguinte forma:

$$L := \varepsilon \cup L \cup LL \cup LLL \cup \ldots \tag{2.26}$$

Assim como no caso do fecho de Kleene para um conjunto de eventos E, o fecho de Kleene para linguagens deve incluir o elemento ε. Vale a pena ressaltar que o fecho de Kleene é uma operação idempotente, isto é $(L^*)^* = L^*$.

Considerando a linguagem L_a, seu fecho de Kleene é dado por:

$$L_a^* = \varepsilon, aa, ab, aaaa, aaab, abaa, abab, aaaaaa, aaaaab, \ldots. \tag{2.27}$$

2.4.2 Autômatos

Um autômato é um dispositivo capaz de representar uma sequência de eventos de acordo com regras bem definidas. Na figura 2.2 é mostrado a representação gráfica de um autômato. Nessa representação, cada nó está associado a um estado do sistema e as transições estão associadas aos eventos, que promovem a mudança de um estado para outro.

De acordo com [16], um autômato G é uma sêxtupla $G = (X, E, \Gamma, f, X_m, x_0)$ em que:

1. X é o espaço de estados do sistema;

2. E é o conjunto finito de eventos associados a G;

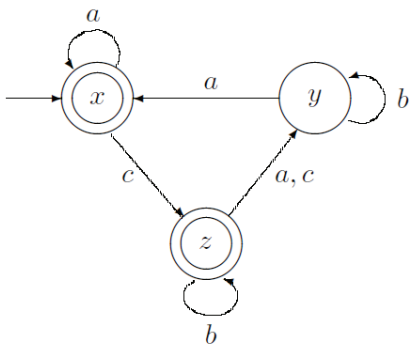


Figura 2.2: Diagrama de transição de estados do autômato G_1 [2].

3. f é a função de transição de estados definida como:

$$f : X \times E \to X \tag{2.28}$$

$$(x, e) \mapsto y = f(x, e), \tag{2.29}$$

que significa que existe uma transição associada a um evento e que, acontecendo no estado x, leva o sistema para o estado y;

4. Γ é a função que determina os eventos ativos em um estado, sendo definida por $\Gamma : X \mapsto 2^E$. Ou seja, $\Gamma(x)$ é o conjunto de todos os eventos $e \in E$ para os quais $f(x, e)$ é definida, $\forall x \in X$;

5. x_0 é o estado inicial ou estado de partida do sistema. No diagrama de transição de estados, o estado inicial é aquele que possui uma "seta livre" apontada para ele;

6. X_m é um subconjunto de X, denominado de conjunto dos estados marcados. Um estado pode ser marcado por diversos motivos como, por exemplo, para ressaltar um estado que indica o término de uma tarefa, ou para indicar que o sistema está em uma situação crítica.

Um autômato funciona sempre de maneira sequencial, sendo que o estado do sistema permanece o mesmo até que ocorra um evento que determine a sua mudança. Dentro de um autômato pode ocorrer de um evento levar à permanência no estado atual, o chamado "*self-looping*". Este pode ser visto na figura 2.2 quando um evento b leva os estados y e z a permanecerem em si. Assim como o evento a faz no estado x.

2.4.3 Linguagem de um autômato

A determinação da linguagem de um autômato é feita através da inspeção do diagrama de transição de estados do mesmo e pode ser dividida em linguagem gerada e linguagem

marcada. A linguagem gerada corresponde a todas as sequências de eventos que podem ser seguidas no diagrama de transição de estados, começando no estado inicial. Já a linguagem marcada é determinada por todas as sequências de eventos que levam o sistema do estado inicial até um estado marcado.

Por definição, a linguagem gerada por um autômato $G = (X, E, \Gamma, f, X_m, x_0)$ é dada por:

$$\mathcal{L}(G) := \{s \in E^* : f(x_0, s)\text{é definida}\}, \tag{2.30}$$

e a linguagem marcada é dada por:

$$\mathcal{L}_m(G) := \{s \in \mathcal{L}(G) : f(x_0, s) \in X_m\}. \tag{2.31}$$

Portanto, uma sequência s está em $\mathcal{L}(G)$ se e somente se corresponde a um caminho admissível no diagrama de transição de estado, equivalentemente, se e somente se f for definido em (x_0, s). $\mathcal{L}(G)$ é prefixo-fechado por definição, uma vez que um caminho só é possível se todos os seus prefixos também são possíveis. O mesmo não pode ser afirmado para a linguagem marcada, uma vez que nem todos os estados X precisam ser marcados.

Para exemplificar a determinação das linguagens gerada e marcada, considere o autômato da figura 2.3 que possui o conjunto de eventos $E = \{a, b\}$.

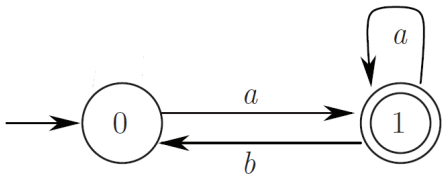


Figura 2.3: Exemplo de autômato.

A linguagem gerada deste autômato consiste de ε juntamente com todos os conjuntos de eventos de E^* que começam com o evento a e não possui nenhuma ocorrência de dois ou mais eventos b de forma consecutiva. Qualquer evento b ou é o último evento ou é imediatamente seguido pelo evento a. Já a linguagem marcada ($\mathcal{L}_m(G)$) é dada pelo subconjunto de $\mathcal{L}(G)$ que consiste nas sequências de eventos que terminam com o evento a, de modo que o sistema encerra o processo no estado marcado.

As linguagens gerada e marcada de um autômato permitem a análise da existência, ou não, de bloqueios no mesmo. A definição de bloqueio em autômatos, bem como suas características, são apresentadas a seguir.

2.4.4 Autômatos com bloqueio

Da definição de G, $\mathcal{L}(G)$ e $\mathcal{L}_m(G)$ tem-se que

$$\mathcal{L}_m(G) \subseteq \overline{\mathcal{L}_m(G)} \subseteq \mathcal{L}(G). \tag{2.32}$$

O primeiro conjunto de inclusões é devido a G e a definição de prefixo fechamento, e a segunda é consequência da definição de $\mathcal{L}_m(G)$ e do fato que $\mathcal{L}(G)$ é prefixo fechada por definição.

Um autômato G, no decorrer dos eventos, pode alcançar um estado $x_i \in X$, no qual $\Gamma(x) = \emptyset$, mas $x_i \notin X_m$. Neste caso, nenhum evento pode ser executado e o sistema fica preso no estado x_i não conseguindo alcançar mais o estado marcado. Este fenômento é conhecido denominado *deadlock*.

Quando um *deadlock* ocorre, necessariamente $\overline{\mathcal{L}_m(G)}$ será diferente de $\mathcal{L}(G)$, uma vez que qualquer palavra em $\mathcal{L}(G)$ que termina no estado x_i não pode ser prefixo de uma palavra de $\mathcal{L}_m(G)$.

Outra importante característica a se considerar é quando existe um conjunto de estados não marcados em G, que formam um conjunto no qual os estados podem ser alcançados entre si, porém, não há ocorrência de nenhum evento que leve o sistema a um outro estado fora deste conjunto. Esta característica é denominada de *livelock* e também implica no fato de que $\overline{\mathcal{L}_m(G)}$ será diferente de $\mathcal{L}(G)$.

Nesses dois casos é dito que o autômato possui um bloqueio e

$$\overline{\mathcal{L}_m(G)} \subset \mathcal{L}(G). \tag{2.33}$$

Quando não há bloqueio

$$\overline{\mathcal{L}_m(G)} = \mathcal{L}(G). \tag{2.34}$$

Um exemplo de autômato com bloqueio pode ser visto na figura 2.4, onde pode-se perceber que o estado 5 representa um *deadlock*, pois $5 \notin X_m$ e $\Gamma(5) = \emptyset$ e os estados 3 e 4 apresentam um *livelock*, pois é impossível alcançar o estado marcado a partir deles.

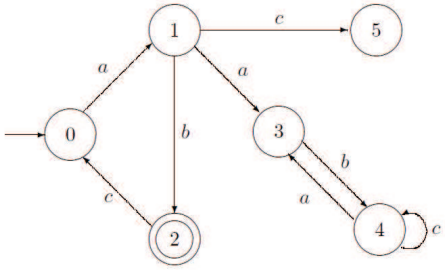


Figura 2.4: Autômato G_3: Exemplo de autômato com bloqueio.

Para este exemplo, as linguagens gerada e marcada são determinadas pelas expressões a seguir

$$\mathcal{L}(G_3) = \overline{\{abc\}^*} \cup \{a\}\{bca\}^*\{\{c\} \cup \{a\}\overline{\{b\}\{c\}^*\{a\}\}^*\}. \tag{2.35}$$

$$\mathcal{L}_m(G_3) = \{ab\}\{cab\}^*. \tag{2.36}$$

Pode-se perceber que $\overline{\mathcal{L}_m(G_3)} \subset \mathcal{L}(G_3)$, confirmando assim a existência de bloqueio no autômato G_3.

2.4.5 Operações com autômatos

Existem dois tipos básicos de operações com autômatos: as operações realizadas com um único autômato e as operações de composição, nas quais as manipulações ocorrem considerando as características de mais de um autômato. Neste trabalho serão apresentados os conceitos dos tipos de operações com um único autômato e será dada maior atenção às operações de composição entre autômatos.

Operações Unárias

As operações unárias são divididas em: *acessibilidade*, *co-acessibililade* e *trim*. Nessas operações, o diagrama de transição de estados de um autômato é alterado considerando apenas as propriedades deste autômato.

- *Acessibilidade:* Seja o autômato $G = (X, E, \Gamma, f, X_m, x_0)$, cuja linguagem marcada é $\mathcal{L}_m(G)$ e a linguagem gerada é $\mathcal{L}(G)$. Um estado $x_i \in X$ é acessível se existe uma palavra s pertencente a $\mathcal{L}(G)$ capaz de levar o sistema do estado x_0 até o estado x_i, ou seja $f(x_0, s) = x_i$. Do contrário, o estado é não acessível.

  A parte acessível de um autômato G, pode ser obtida apagando do diagrama de transição os estados não-acessíveis de G e as transições que partem destes estados. Vale salientar que essa operação não altera nem a linguagem marcada, nem a linguagem gerada pelo autômato G, já que essas linguagens partem, por definição, do estado inicial x_0 [2].

- *Co-acessibilidade:* Seja o autômato $G = (X, E, \Gamma, f, X_m, x_0)$, cuja linguagem marcada é $\mathcal{L}_m(G)$ e a linguagem gerada é $\mathcal{L}(G)$. Um estado de G definido por $x_i \in X$ e $x_i \notin X_m$ é co-acessível, se existe uma palavra $s \in E^*$, capaz de levar o sistema do estado x_i até um estado $x_j \in X_m$, ou seja, $f(x_i, s) = x_j \in X_m$. Do contrário o estado é não co-acessível.

  A parte co-acessível de um autômato G, pode ser obtida apagando do diagrama de transição os estados não co-acessíveis de G e as transições que partem destes estados. Ao contrário da acessibilidade, a obtenção da parte co-acessível de G pode alterar a linguagem gerada pelo autômato, já a linguagem marcada por G permanece sem sofrer alterações [2].

- *Trim:* A co-acessibilidade está estritamente relacionada com a ideia de bloqueio de um sistema. Dizer que um autômato G possui bloqueio significa dizer que $\mathcal{L}_m(G)$ é um subconjunto de $\mathcal{L}(G)$, onde $\mathcal{L}_m(G) \neq \mathcal{L}(G)$. Em outras palavras, um autômato

possui bloqueio quando nele existem estados acessíveis, que não são co-acessíveis. Equivalentemente, um autômato não possui bloqueio quando todos os estados acessíveis são também co-acessíveis, e vice-versa. O autômato que é tanto acessível, quanto co-acessível é chamado de *trim*. Dessa forma, a operação *trim* é obtida aplicando-se consecutivamente as operações de acessibilidade e co-acessibilidade ou vice-versa.

Composição Paralela

Para o desenvolvimento deste trabalho, a operação de composição paralela será utilizada. Esta é utilizada com o intuito de descrever um comportamento complexo em forma de uma composição de comportamentos mais simples, sendo capaz de criar o modelo de um sistema completo através da interação entre os comportamentos individuais do sistema, preservando as transições privadas de cada comportamento.

Considerando dois autômatos:

$$G_1 = (X_1, E_1, \Gamma_1, f_1, X_{m1}, x_{01}) \text{ e } G_2 = (X_2, E_2, \Gamma_2, f_2, X_{m2}, x_{02}) \tag{2.37}$$

a composição paralela de G_1 e G_2 é definida como sendo:

$$G_1 \| G_2 := \text{Ac}(X_1 \times X_2, E_1 \cup E_2, f_{1\|2}, \Gamma_{1\|2}, (x_{01}, x_{02}), X_{m1} \times X_{m2}) \tag{2.38}$$

em que "Ac" representa a acessibilidade do sistema. Um sistema é dito acessível quando todos os seus estados são acessíveis [16].

A função de transição de estados da equação 2.38, $f_{1||2}$, é definida por:

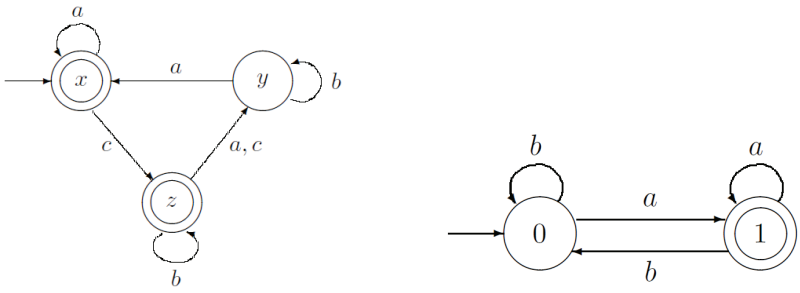
$$f_{1\|2}((x_1, x_2), e) = \begin{cases} (f_1(x_1, e), f_2(x_2, e)), \text{ se } e \in \Gamma_1(x_1) \cap \Gamma_2(x_2), \\ (f_1(x_1, e), x_2), \text{ se } e \in \Gamma_1(x_1) \backslash E_2, \\ (x_1, f_2(x_2, e)), \text{ se } e \in \Gamma_2(x_2) \backslash E_1, \\ \text{não definida, caso contrário.} \end{cases} \tag{2.39}$$

Os estados de $G_1||G_2$ são denotados em pares, sendo a primeira componente o estado atual de G_1 e a segunda componente o estado atual de G_2. O estado (x_1, x_2) é marcado somente se $x_1 \in X_{m1}$ e $x_2 \in X_{m2}$.

No resultado da composição paralela, um evento $e \in E_1 \cap E_2$ só pode ser executado se ele ocorrer simultaneamente em todos os modelos envolvidos na composição. Isso quer dizer que a composição é síncrona para *eventos comuns*. Por outro lado, os eventos privados, isto é, os eventos pertencentes ao conjunto $(E_1 \backslash E_2) \cup (E_2 \backslash E_1)$, não sofrem restrições e podem ser executados sempre que ocorrem em seus comportamentos de origem.

Para ilustrar o funcionamento da composição paralela, considere como exemplo os autômatos G_1 (mostrado anteriormente na figura 2.5a) e G_2 (mostrado na figura 2.5b).

A composição paralela entre estes autômatos resulta em um novo autômato $G_1||G_2$, em que:



(a) Diagrama de transição de estados do autômato G_1.

(b) Diagrama de transição de estados do autômato G_2.

Figura 2.5: Autômatos para realização da composição paralela.

$$X_{1||2} = (x,0),(x,1),(y,0),(y,1),(z,0),(z,1) \tag{2.40}$$

$$E_{1||2} = a,b,c, \tag{2.41}$$

sendo c o único evento privado, pertencente a G_1. O diagrama de transição de estado para $G_1||G_2$ é aquele representado na figura 2.6.

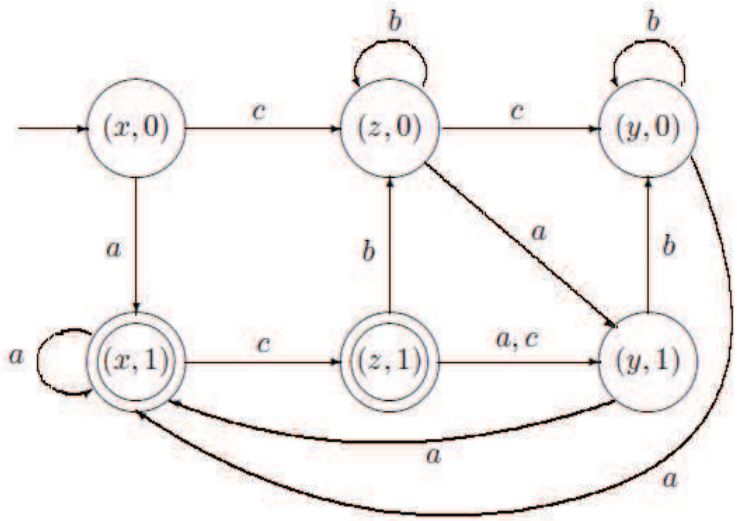


Figura 2.6: Composição paralela entre G_1 e G_2, $G_1||G_2$.

Capítulo 3

Revisão Bibliográfica

Uma das teorias mais importantes e referenciadas na área de navegação de robôs móveis é a arquitetura de subsunção desenvolvida por Brooks em [3]. Esta teoria deu início à chamada arquitetura de navegação reativa ao tentar solucionar as deficiências das abordagens deliberativas em ambientes dinâmicos e desconhecidos. A partir daí, várias teorias foram criadas a fim de habilitar o robô a navegar em ambientes desconhecidos e dinâmicos.

No entanto, mesmo com a vasta pesquisa que é realizada na área de navegação e a consequente criação de novas técnicas, a arquitetura de subsunção de Brooks vem sendo bastante utilizada desde a sua publicação. Na seção 3.1 é apresentada a arquitetura de subsunção, na seção 3.2 são apresentados vários trabalhos que fazem uso desta arquitetura desde sua publicação, na seção 3.3 são apresentadas as limitações da mesma e na seção 3.4 é apresentada a ideia proposta por Harper e Winfield em [4] para prova de estabilidade da arquitetura de subsunção.

3.1 Arquitetura de Subsunção

Existem várias abordagens diferentes para se construir um robô móvel autônomo e, como a maioria dos problemas de engenharia, elas geralmente são iniciadas pela decomposição do problema em pedaços menores, resolvendo cada um desses pedaços e então compondo as soluções encontradas [3].

Até a publicação do trabalho de Brooks [3], a maioria das abordagens eram voltadas para uma decomposição em módulos funcionais conectados em série, como apresentado a figura 3.1, de forma que a informação flui do ambiente para o robô através dos sensores e volta para o ambiente através das ações realizadas pelo robô fechando um ciclo realimentado.

Em 1986, Rodney Brooks introduziu uma nova arquitetura de navegação de robôs móveis que substituía essa ideia de composição em série por uma abordagem na qual a tarefa

Figura 3.1: Decomposição de um sistema de controle de um robô móvel em módulos funcionais [3].

é dividida em comportamentos organizados em paralelo e voltados para o cumprimento de subtarefas, como apresentado na figura 3.2.

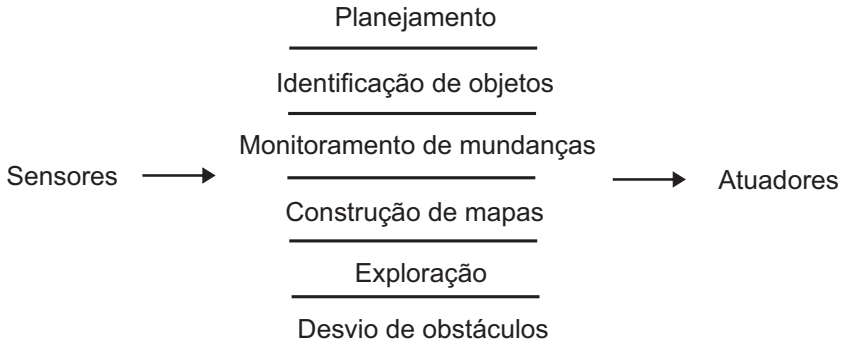


Figura 3.2: Decomposição de um sistema de controle de um robô móvel baseado em comportamentos [3].

A distribuição dos comportamentos de forma paralela pode levar o sistema a um impasse quando dois ou mais comportamentos precisam ser ativados ao mesmo tempo. Por este motivo, Brooks organizou os comportamentos em camadas de forma hierárquica, onde cada comportamento aceita sinais de supressão e inibição, sendo que um sinal de supressão sobrepõe o sinal normal de entrada e o sinal de inibição inibe completamente a saída. Estes sinais permitem que os comportamentos se sobreponham e os comportamentos das camadas mais altas tenham prioridade em relação às camadas mais baixas a fim de produzir um comportamento global satisfatório. Esta estrutura, denominada de arquitetura de subsunção, pode ser representada pela figura 3.3.

Esta arquitetura apresenta várias vantagens para a navegação de um robô móvel, como por exemplo a possibilidade de atribuição de múltiplos objetivos, existindo camadas individuais que trabalham de forma concorrente para o cumprimento dos mesmos com a possibilidade de tomada de decisões em tempo real; a utilização de múltiplos sensores sem a necessidade da fusão dos mesmos; e a aditividade, onde outras camadas podem ser facilmente adicionadas ao sistema uma vez que elas trabalham de forma paralela.

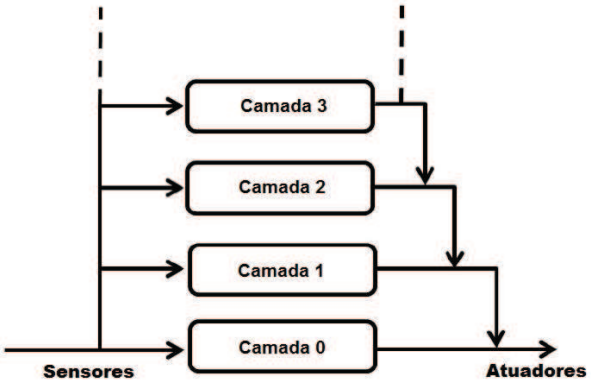


Figura 3.3: Arquitetura de subsunção [3].

3.2 Trabalhos Correlatos

Pouco tempo após sua publicação, o trabalho de Brooks passou a ser citado por muitos autores. Por exemplo, em 1990, Bellingham [17] utilizou a arquitetura de subsunção para robótica aquática.

Já Bonasso, em 1991 [18], utilizou a arquitetura de subsunção para produzir planejamento reativo que foi implementado utilizando uma programação com linguagem de autômatos. No mesmo ano, Hartley e Pipitone [19] realizaram experimentos para controle de um protótipo de avião a fim de testar a arquitetura de subsunção para um problema mais complexo.

Em 1992, Connell [20] desenvolveu uma arquitetura em três camadas para o controle de um robô. Ela combina uma camada de servo controle, uma camada de subsunção e uma camada simbólica de forma que permite que as vantagens de cada técnica possam ser inteiramente exploradas. Já em [21], foi utilizada programação genética para desenvolver um comportamento de seguir corredor utilizando a arquitetura de subsunção.

Em 1994, foi desenvolvido em [22] uma variação do controle por camadas para facilitar a configuração das tarefas a partir de uma arquitetura baseada em 3 camadas. Isto foi feito através de um controle de camadas configurado por estados para determinar as tarefas a serem executadas pelo robô.

A ideia de uma arquitetura baseada em camadas com prioridades também foi utilizada para o estudo de singularidades no que diz respeito à cinemática e algoritmos para manipuladores, no trabalho proposto por Chiaverini em [23].

Em 1998, Nakashima e Noda [24] aplicaram uma versão estendida da arquitetura de subsunção para futebol de robôs, que eles chamaram de arquitetura de subsunção dinâmica. Para tanto, a arquitetura foi aplicada em duas dimensões, em que a horizontal correspondia às camadas de execução de tarefas e a vertical que correspondia a tipos de comportamentos diferentes.

Togelius, em 2004 [25], combina a evolução incremental e modularizada com elementos

da arquitetura de subsunção. Basicamente, um robô em evolução em camadas é controlado por um controlador de subsunção onde cada camada é uma rede neural evolutiva de um dos vários tipos possíveis.

Yongjie, em 2006 [26], propôs uma arquitetura híbrida baseada em subsunção. A abordagem foi desenvolvida substituindo as camadas da arquitetura de Brooks por um grupo de processos independentes. Além disso, uma camada de comportamento de gestão é adicionada a fim de ajustar os parâmetros de comportamento do robô de acordo com a tarefa atual, estado do robô e ambiente. O modelo de comportamento decide como o sistema de controle irá agir após receber as informações. De um modo geral, o ajuste do modelo de comportamento inclui alterar o estado dos processos (ativados ou não), alterando os parâmetros de comportamento aritmético ou mudando as prioridades dos comportamentos.

Em 2008, Antonelli [27] apresentou uma técnica para lidar com a prioridade relativa de cada tarefa com base no trabalho desenvolvido em [23] com camadas de prioridade para singularidades em manipuladores. Neste trabalho, que foi chamado de comportamento baseado no espaço nulo (do inglês, *null-space-based behavioral* - NSB), o controle difere das outras arquiteturas devido ao fato de que as saídas de cada comportamento são combinadas a fim de gerar um comportamento mais complexo.

Em 2010, Godin *et. al.* [28] utilizaram uma abordagem inspirada no controle de camadas configurado por estados desenvolvido em [22], que é uma extensão do trabalho de Brooks, para aplicação em robótica aquática.

Em 2012, Beltran e Gomez [29] desenvolveram um trabalho baseado na arquitetura de subsunção usando algoritmo genético e redes neurais como métodos de aprendizagem. Neste trabalho o controlador da camada é composto por controladores de baixo nível para cada motor na estrutura do robô, sendo que cada motor tem um agente que realiza uma interpretação básica de instruções. Na camada de aprendizagem, o algoritmo genético é aplicado, a fim de encontrar as instruções corretas para alcançar o objetivo especificado. Além disso, esta camada é responsável pela formação de redes neurais na camada anterior.

Ainda em 2012, Liu [30] utilizou a arquitetura de subsunção para busca e resgate de vítimas por robôs autônomos baseado em tecnologia de sensoriamento biomimético. Foi desenvolvido também um controle hierárquico incluindo simultaneamente o sistema de sensoriamento, o processamento de dados, a camada deliberativa e o módulo de tomada de decisão por prioridade. Neste último, a prioridade é feita de modo a tornar o resgate mais objetivo.

Em 2013, Nagata *et. al.* [31] utilizaram a arquitetura de subsunção para controle de múltiplos robôs com comunicação *wireless*.

Já Turner [32] propôs a criação de comportamentos através de máquinas de estados finitos interconectadas assincronamente. Essas máquinas de estados foram desenvolvidas para manter estrutura de dados, monitorar as mensagens de entrada e a mensagens

enviadas através da utilização da teoria de MDD (do inglês, *Model Drive Development*).

Em 2014, Chunhua *et. al.* [33] implementou a arquitetura de subsunção para controle de um carro inteligente e Krijnen [34] para controle de um quadricóptero.

Ainda na área de robótica aérea, em 2016 Oland [35] aplicou a arquitetura de subsunção para controle de voo utilizando rotações compostas.

O fato de ser muito utilizada até os dias atuais mostra claramente a importância desta arquitetura de navegação para a robótica móvel e que através da mesma é possível obter resultados bastante satisfatórios. No entanto, este fato também é reflexo de alguns problemas que podem ser encontrados no trabalho de Brooks. Sendo assim, muitos autores ainda discutem a arquitetura de subsunção a fim de conseguir desenvolver uma metodologia cada vez mais robusta.

3.3 Limitações da arquitetura de subsunção

A arquitetura de subsunção apresenta várias vantagens, como o fato de decompor o sistema em uma série de unidades verticais e a utilização de dados sensoriais que são relevantes apenas para as suas necessidades de tomada de decisões específicas, fazendo com que a fusão de sensores seja substituída pela fusão de comandos, de forma a produzir ações mais coerentes [26]. Além disso, Turner, em [32], destaca que cada módulo funciona independentemente em relação ao outro, o que aumenta a velocidade de processamento e torna o sistema mais robusto no que diz respeito a falhas.

Conforme descrito por Brooks, o sistema de controle do robô deve obedecer multitarefas ao mesmo tempo, ainda que estas estejam em conflito entre si. Na arquitetura de subsunção é realizada uma hierarquia de camada em que o controlador de uma camada superior sobrepõe todas as camadas inferiores. A camada que suprime o controle do robô em qualquer ponto não precisa ter conhecimento de que está controlando o robô, e dessa mesma forma, a camada de controle não tem nenhuma informação sobre a camada suprimida [26]. Esta estrutura implica em algumas limitações que são apontadas em trabalhos da área e que foram resumidas em [26] nos seguintes itens:

- A fim de tomar uma decisão mais adequada, o controlador precisa pesar muitos fatores na seleção de comandos de controle. Às vezes, é inapto para o comportamento de alto nível conter o de nível mais baixo, perdendo assim as informações das camadas inferiores [36]. Por exemplo, caso a camada 2 decida controlar o robô, a saída da camada 0 é impedida e suas informações não são utilizadas;

- Um outro problema é encapsular comportamentos dentro de um conjunto de máquinas de estados finitos, que tornam o estado interno destes comportamentos inacessível para controlar camadas [37]. O projetista de cada camada pode não saber o que

as outras camadas vão fazer e ele não pode prever a relação entre a sua camada e outras camadas. Como resultado, uma vez que uma nova camada de comportamento é adicionada, há a necessidade de ajustar outras camadas;

- Embora a arquitetura de subsunção aumente a velocidade de resposta do sistema, o projeto do comportamento é relativamente simples e curto para a competência necessária do raciocínio e coordenação. Em média, é possível obter resultados satisfatórios para algumas tarefas simples, mas para tarefas mais complicadas, a limitação da falta de capacidade de planejamento emerge claramente.

Os problemas que surgem por conta do encapsulamento dos comportamentos em máquinas de estado finito tentam ser contornados na literatura através de formalismos que permitam uma maior simplicidade na modelagem da máquina de estados. Esses formalismos, visam permitir que vários comportamentos possam ser modelados por partes, a fim de cumprir tarefas mais complicadas, e também uma análise sobre a máquina de estados de forma a garantir que os estados levem o robô a cumprir o objetivo de navegação desejado. Alguns exemplos de tais formalismos podem ser vistos em [38], que apresenta uma estrutura lógica para modelar a arquitetura de subsunção e em [39], que apresenta uma abordagem voltada para a modelagem na forma de linguagens regulares.

Recentemente, [40] apresentou uma abordagem baseada em sistemas a eventos discretos com redes de Petri. Este trabalho possui características muito importantes para a modelagem de uma arquitetura de subsunção, como por exemplo o fato de que os módulos de comportamento podem ser compostos para criar módulos mais complexos de acordo com a aplicação e novas estratégias de controle podem ser adicionadas ao sistema. Além disso, o projeto do sistema é simples e de baixo custo computacional.

Nesta dissertação, todas essas características citadas são contempladas numa abordagem também baseada na teoria de sistemas a eventos discretos, mas com modelos representados na forma de autômatos e suas linguagens. Nesta abordagem, são propostos modelos de autômatos que facilitam a modelagem dos comportamentos e, através de suas propriedades, generalizam o comportamento global do sistema com um padrão que permite a análise do comportamento do sistema mais intuitiva.

Além dos problemas de modelagem, Turner [32] aponta que, para tarefas mais complexas, a incapacidade de planejar limita o sucesso da arquitetura de subsunção, por exemplo, com a aparição de situações em que ocorre um impasse entre as tarefas e dois ou mais processos ficam impedidos de continuar suas execuções.

Grande parte dos problemas apontados na arquitetura de subsunção de Brooks é proveniente da forma como é feito o encapsulamento em máquinas de estado finito e da incapacidade de planejar. No entanto, existe um ponto muito importante a se analisar na arquitetura de subsunção que é muito pouco trabalhado, a questão da estabilidade do próprio sistema. Uma vez que a mudança de comportamentos implica em trocar de

controlador, o chaveamento pode levar o sistema a instabilidade mesmo para casos em que todos os controladores sejam individualmente estáveis.

A única metodologia encontrada que desenvolveu uma teoria matemática para a análise de estabilidade da arquitetura de subsunção foi o trabalho de Harper e Winfield [4]. Neste trabalho, foi proposta uma base formal para a arquitetura de subsunção baseada em uma extensão da teoria de estabilidade de Lyapunov, o teorema de estabilidade de segunda ordem.

A metodologia proposta por Harper e Winfield segue uma abordagem similar a abordagem proposta nesta dissertação. Por isso, esse trabalho é aprensentado com mais detalhades a seguir.

3.4 Teorema de estabilidade de segunda ordem

A abordagem apresentada por Harper e Winfield é baseada na ideia de um campo gravitacional, no qual se um corpo se movimenta numa direção contrária ao centro de gravitação, a sua aceleração diminui progressivamente até que, no momento em que sua aceleração for zero, a força gravitacional fará o corpo retornar ao centro do campo.

Essa ideia é extrapolada para a função de energia do sistema, de modo que a regra de chaveamento só permite que a função de energia cresça enquanto a segunda derivada da função de Lyapunov for negativa, o que significa dizer, de modo análogo, uma desaceleração. Com isso, pode-se afirmar que a função de energia estará limitada por uma certa região e consequentemente o sistema será estável. O teorema proposto em [4] é apresentado a seguir.

***Teorema de estabilidade marginal de segunda ordem**: $x(t)$ define a evolução no tempo do sistema autônomo $\dot{x} = f(x)$ e $V(x)$ define uma função escalar definida na vizinhança de $x = 0$. $W(t) \equiv V(x,t)$ denota o valor de $V(x)$ ao longo da trajetória dos estados do sistema. Portanto, $\dot{W}(t) \equiv \dot{V}(x,t)$ e $\ddot{W}(t) \equiv \ddot{V}(x,t)$ denotam a taxa de variação e aceleração do valor de V(x) ao longo das trajetórias do sistema. A seguinte expressão define a estabilidade marginal de segunda ordem:*

$$\begin{aligned} &\dot{W}(t) \leq 0 \vee [0 < \dot{W}(t) < \dot{W}_{max} \wedge \ddot{W}(t) < \ddot{W}_{max} < 0] \\ &\rightarrow [\forall R, \exists r > 0, \|x(0)\| < r \rightarrow \|x(t)\| < R]. \end{aligned} \tag{3.1}$$

A estabilidade é possível na presença de condições iniciais positivas para $\dot{W}(t)$, desde que eles sejam limitados e que $\ddot{W}(t)$ tenha um limite superior negativo $\ddot{W}_{max}$. Na figura 3.4 é apresentada uma representação gráfica da estabilidade de segunda ordem.

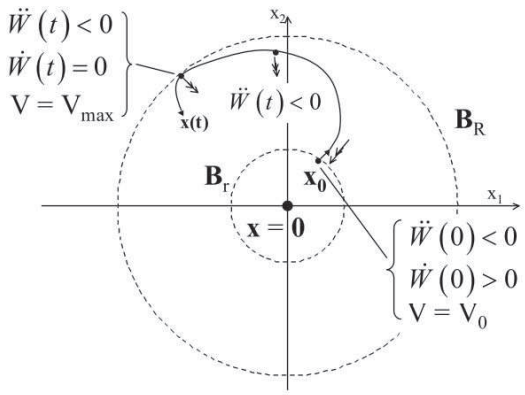


Figura 3.4: Representação gráfica da estabilidade de segunda ordem [4]

Para estabilidade assintótica na derivada de segunda ordem tem-se:

$$\begin{aligned} &\dot{W}(t) < 0 \vee [0 \leq \dot{W}(t) < \dot{W}_{max} \wedge \ddot{W}(t) < \ddot{W}_{max} < 0] \\ &\Rightarrow [\forall R, \exists r, 0 < r < R, \|x(t)\| < r \rightarrow [\|x(t)\| < R \\ &\wedge \|x(t \rightarrow \infty)\| \rightarrow 0]]. \end{aligned} \tag{3.2}$$

Essa teoria possibilitou a aplicação para prova de estabilidade da arquitetura de subsunção devido ao fato de que a primeira derivada da função de Lyapunov subsume a segunda. Sendo assim, a função de energia do comportamento de maior prioridade é analisada e o chaveamento para outros comportamentos só é permitido caso essa energia esteja decrescendo ou crescendo com aceleração negativa.

Na verdade, como esta teoria possibilita o crescimento da função de energia, a teoria proposta por Harper e Winfield prova a *quasi*-estabilidade do sistema.

No entanto, o fato de só permitir que a função de energia do sistema cresça enquanto a aceleração é negativa, gera uma grande restrição para o sistema, podendo fazer com que o mesmo não cumpra os objetivos dos comportamentos das camadas mais baixas. Na seção dos resultados são apresentadas situações nas quais essa restrição interfere de forma significativa no comportamento do robô.

Na abordagem aqui proposta também não será garantido que os comportamentos de menor ordem de prioridade serão cumpridos, porém, a utilização da teoria de sistemas chaveados com objetivo de controle principal permite uma maior flexibilidade com relação à restrição de crescimento da função de energia. Com isso, a estabilidade do sistema será garantida de forma mais abrangente que a abordagem de estabilidade de segunda ordem. A abordagem proposta é apresentada no próximo capítulo.

Capítulo 4

Arquitetura de subsunção baseada em objetivo de controle principal

A abordagem proposta para implementação de uma arquitetura de subsunção de modo que se possa obter uma garantia de *quasi*-estabilidade, é fundamentada na teoria de sistemas chaveados com objetivo de controle principal proposta por Carvalho em [1]. Neste trabalho, é apresentada uma regra de chaveamento que garante a *quasi*-estabilidade assintótica do sistema caso seu ponto de equilíbrio seja o mesmo de um subsistema principal, mesmo havendo chaveamento para outros subsistemas que possuam pontos de equilíbrio diferentes ou até mesmo que sejam instáveis. Para isso, o subsistema principal deve ser globalmente assintoticamente estável segundo Lyapunov e o chaveamento deve atender as regras que impõe o quanto e por quanto tempo a função de energia do sistema pode crescer para que a *quasi*-estabilidade assintótica seja garantida.

A ideia de que o subsistema principal tem uma maior prioridade em relação aos demais e a garantia de convergência do método, faz com que essa teoria se mostre factível para fundamentar a teoria de subsunção, uma vez que a regra de chaveamento pode ser aplicada de modo que cada camada seja considerada como um subsistema principal em relação às camadas inferiores.

Sendo assim, a abordagem proposta consiste na implementação de um controlador globalmente assintoticamente estável segundo Lyapunov (subsistema principal) e outros controladores (subsistemas secundários) nos quais a análise de estabilidade não é exigida. A regra de chaveamento é desenvolvida com base no teorema que foi apresentada na seção 2.3.1, que diz respeito ao chaveamento lento de controladores. Para isso, o chaveamento deve ser composto por uma função SSLF $V_1(x)$ que atenda à equação 2.19 e aos parâmetros de *quasi*-estabilidade apresentados na seção 2.1. É considerado também que as variáveis de estado do subsistema principal são conhecidas e observáveis durante todo o processo.

O chaveamento é feito com base na mensuração contínua de $||x(t)||$ e $V_1[x(t)]$, sendo escolhidos arbitrariamente o valor da constante ρ do teorema e um τ finito, como sendo a constante de chaveamento lento, que tem por objetivo determinar por quanto tempo a

SSLF do sistema pode crescer.

O modelo geral do sistema é apresentado na figura 4.1:

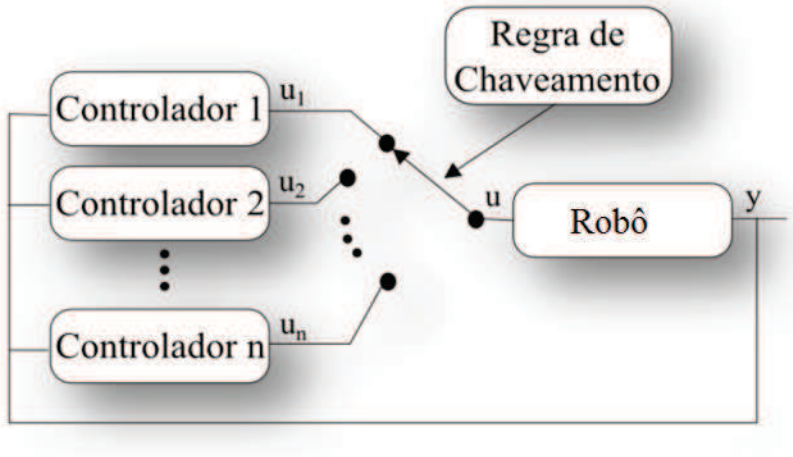


Figura 4.1: Modelo geral do sistema [1]

O sistema é iniciado no subsistema principal e permanece no mesmo até que as condições de *quasi*-estabilidade do teorema 2 e 3 sejam satisfeitas. A partir de então, os subsistemas de menor prioridade podem ser ativados durante o tempo τ, de acordo com a necessidade do sistema, devendo voltar ao subsistema principal após esse tempo e permanecendo no mesmo até que as condições sejam novamente satisfeitas. Como o controlador principal é globalmente assintoticamente estável segundo Lyapunov, tais condições são satisfeitas em um tempo τ_{SSLF} finito, atendendo assim ao teorema de chaveamento lento de controladores e portanto cumprindo o objetivo principal do sistema. Essa ideia é replicada para todos os subsistemas de forma hierárquica, de maneira que cada camada é considerada como principal em relação às que estejam em um nível de prioridade mais baixo.

Dessa forma, para cada subsistema é atribuída uma função que avalia se o objetivo do mesmo está sendo cumprido, a fim de permitir ou não o chaveamento para subsistemas de menor prioridade, e uma constante que indica por quanto tempo as camadas inferiores podem ser ativadas até que haja uma nova checagem do cumprimento do objetivo. Esse tempo deve ser sempre menor do que os tempos atribuídos para as camadas de maior prioridade de forma que possa haver um sincronismo em todo o sistema.

Uma vez definida a regra de chaveamento que governa todo o sistema e garante, através da teoria de chaveamento lento de controladores, o cumprimento do objetivo principal, faz-se necessário garantir que essas exigências serão cumpridas pelo robô durante a navegação. Por esse motivo, a modelagem dos subsistemas e da regra de chaveamento foram desenvolvidas na forma de autômatos com base na teoria de sistema discretos. Essa ferramenta permite a adição ou extração de subsistemas de forma fácil e rápida, além de possibilitar que a modelagem seja realizada por partes, ao invés de criar um

modelo complexo do comportamento de todo o sistema. Os detalhes da modelagem do sistema serão apresentados na próxima seção.

4.1 Modelagem do sistema em forma de autômatos

A modelagem do sistema foi feita com o desenvolvimento de um autômato para cada subsistema, sendo que para cada autômato é implementada a ativação do controlador referente àquele subsistema e a regra de chaveamento para que o subsistema da camada mais baixa possa ser ativado ou não.

Para isso, foi criado um modelo de autômato para o subsistema principal e um outro modelo para todas as camadas inferiores a ele, possibilitando assim a adição de mais subsistemas de forma simples e rápida através da composição paralela.

Para a modelagem do subsistema principal foram considerados os seguintes eventos definidos na tabela 4.1.

Tabela 4.1: Descrição dos eventos do autômato que modela o subsistema principal

Evento	Controlável	Observável	Descrição
A_{cp}	✓	✓	Ativar controlador principal
r_1	x	✓	Regra de chaveamento 1 satisfeita
H_{c2}	✓	✓	Habilitar controlador secundário
t	x	✓	Tempo τ atingido
n_1	x	✓	Necessidade de ativar o controlador 1
nn_1	x	✓	Não necessidade de ativar o controlador 1

Na tabela, o símbolo "✓" indica que o evento é controlável ou observável e o símbolo "x" indica que o mesmo é não controlável ou não observável.

A partir desses eventos foi possível desenvolver o autômato da figura 4.2 que modela o subsistema principal.

Este autômato pode ser representado em termos de linguagem através da seguinte expressão:

$$\mathcal{L}(P) = \overline{\{\{n_1 A_{cp} r_1 H_{c2} t\}^* \{n_1 A_{cp} nn_1 \cup nn_1\} \{n_1 A_{cp} nn_1\}^* n_1 A_{cp} r_1 H_{c2} t\}^*}. \quad (4.1)$$

O sistema é iniciado checando a necessidade de ativação do controlador principal. Se necessário (evento n_1), o controlador principal é ativado (evento A_{cp}) e o sistema analisa a regra de chaveamento que é baseada no teorema 3, ou seja, verifica se a energia atual do sistema ($V_1[x(t_j)]$) menos a energia $V_1[x(t_i)]$, onde t_i é o instante em que houve chaveamento para o controlador principal pela última vez, decresceu do valor de $\rho||x(t_i)||$. Se ela estiver sendo satisfeita (evento r_1), o sistema habilitará a ativação do controlador

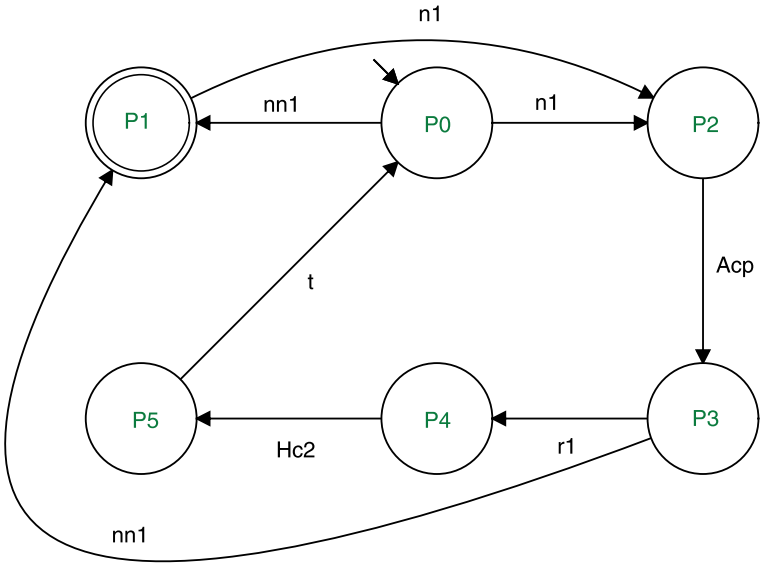


Figura 4.2: Autômato do subsistema principal

secundário (evento H_{c2}) por um tempo τ, devendo desabilitar o mesmo e retornar ao estado $P0$ assim que esse tempo é atingido (evento t) para uma nova verificação. Caso não haja necessidade de ativar o controlador principal (evento nn_1) o sistema irá para o estado marcado $P1$, que indica que o objetivo principal do robô foi cumprido. É importante destacar que o fato de habilitar o controlador secundário não significa ativá-lo, mas sim que o chaveamento para este será permitido caso necessário. Quem irá determinar se o chaveamento ocorrerá ou não é o autômato que modela o subsistema secundário. Esse subsistema é modelado a partir dos eventos mostrados na tabela 4.2.

Tabela 4.2: Descrição dos eventos do autômato que modela o subsistema secundário

Evento	Controlável	Observável	Descrição
H_{c2}	✓	✓	Habilitar controlador 2
r_2	x	✓	Regra de chaveamento 2 satisfeita
n_2	x	✓	Necessidade de ativar o controlador 2
nn_2	x	✓	Não necessidade de ativar o controlador 2
A_{c2}	✓	✓	Ativar controlador 2
H_{c3}	✓	✓	Habilitar controlador terciário
t	x	✓	Tempo τ atingido
t_2	x	✓	Tempo τ_2 atingido

A partir desses eventos foi possível desenvolver o autômato da figura 4.3 que modela o subsistema secundário e sua linguagem que é apresentada na expressão 4.2. É importante destacar que todas as linguagens que são apresentadas a partir daqui estão representadas em mais de uma linha por conta do tamanho das expressões, mas todas as linhas estão sob o mesmo prefixo fechamento.

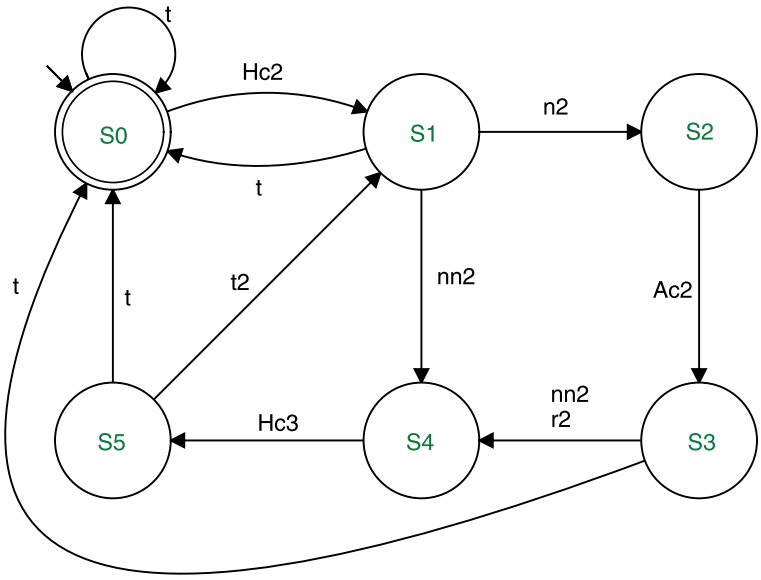


Figura 4.3: Autômato do subsistema secundário

$$\mathcal{L}(S) = \overline{\{t \cup H_{c2}t \cup H_{c2}n_2A_{c2}t \cup \{H_{c2}nn_2 \cup H_{c2}n_2A_{c2}\{r_2 \cup nn_2\}\}H_{c3}}} \overline{\{\{t_2nn_2 \cup t_2n_2A_{c2}\{nn_2 \cup r_2\}\}H_{c3}\}^*\{t \cup t_2t \cup t_2n_2A_{c2}t\}\}^*} \quad (4.2)$$

O subsistema secundário é iniciado no estado $S0$, indo para o estado $S1$ com a habilitação do controlador 2 (secundário), onde há a checagem sobre a necessidade de ativar o controlador ou não. Caso exista a necessidade (evento n_2), o controlador 2 é ativado (evento A_{c2}). No caso de ter sido ativado, o autômato permanecerá em S3 até que o tempo τ do subsistema principal seja atingido (evento t), fazendo com que o autômato passe do estado $S3$ para $S0$, ou que não haja mais necessidade de o controlador continuar ativado (evento nn_2) ou a regra de chaveamento for satisfeita (evento r_2), fazendo com a transição ocorra do estado $S3$ para o $S4$. Neste caso, o controlador 3 (terciário) será habilitado para atuar durante um tempo τ_2 (evento t_2) ou até que o tempo τ do subsistema principal seja atingido, devendo assim voltar para o estado inicial $S0$. É importante destacar que quando o tempo τ é atingido, o tempo τ_2 é resetado de forma a manter a sincronia entre os mesmos.

O subsistema terciário funciona da mesma maneira que o subsistema secundário, seguindo inclusive o mesmo modelo de autômato. Apenas alguns eventos são acrescentados em determinados estados de modo que o subsistema terciário dependa tanto do subsistema secundário quanto do primário.

Os eventos deste subsistema podem ser vistos na tabela 4.3 e o autômato na figura 4.4.

Tabela 4.3: Descrição dos eventos do autômato que modela o subsistema terciário

Evento	Controlável	Observável	Descrição
H_{c3}	✓	✓	Habilitar controlador 3
r_3	x	✓	Regra de chaveamento 3 satisfeita
n_3	x	✓	Necessidade de ativar o controlador 3
nn_3	x	✓	Não necessidade de ativar o controlador 3
A_{c3}	✓	✓	Ativar controlador 3
H_{c4}	✓	✓	Habilitar controlador 4
t	x	✓	Tempo τ atingido
t_2	x	✓	Tempo τ_2 atingido
t_3	x	✓	Tempo τ_3 atingido

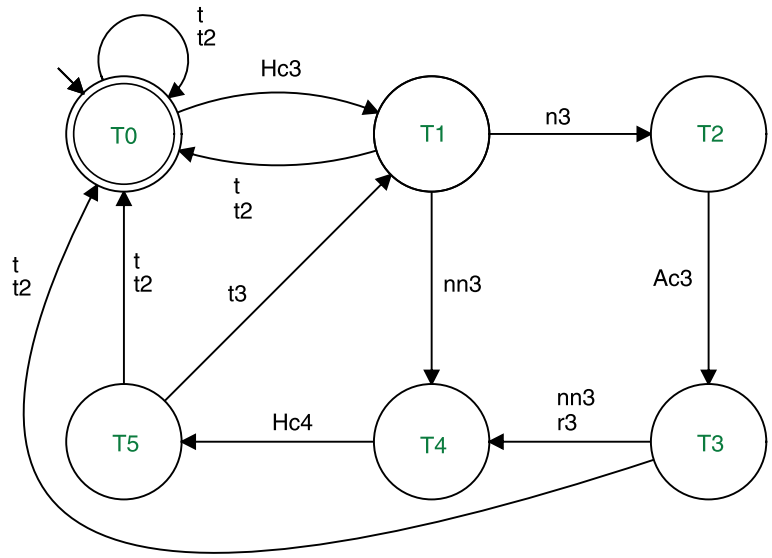


Figura 4.4: Autômato do subsistema terciário

A linguagem do subsistema terciário é representada por:

$$\mathcal{L}(T) = \overline{\{\{t \cup t_2\} \cup H_{c3}\{t \cup t_2\} \cup H_{c3}n_3A_{c3}\{t \cup t_2\} \cup \{H_{c3}nn_3 \cup H_{c3}n_3A_{c3}\{r_3 \cup nn_3\}\}H_{c4}} \overline{\{\{t_3nn_3 \cup t_3n_3A_{c3}\{nn_3 \cup r_3\}\}H_{c4}\}^*\{\{t \cup t_2\} \cup t_3\{t \cup t_2\} \cup t_3n_3A_{c3}\{t \cup t_2\}\}\}^*}, \tag{4.3}$$

onde pode-se perceber uma estrutura de linguagem muito semelhante à linguagem do subsistema secundário com a adição da dependência dos τs das camadas superiores que é dada pela união dos eventos $\{t \cup t_2\}$.

A partir da visualização da modelagem dos subsistemas secundários e terciários é possível notar um modelo padrão, sendo assim fácil prever como será a modelagem do subsistema quaternário ou qualquer outro que se queira adicionar ao sistema. Na figura 4.5 é possível ver a modelagem para o subsistema quaternário conforme o esperado e na figura 4.6 é apresentado o modelo geral pra qualquer comportamento que se queira

adicionar ao sistema.

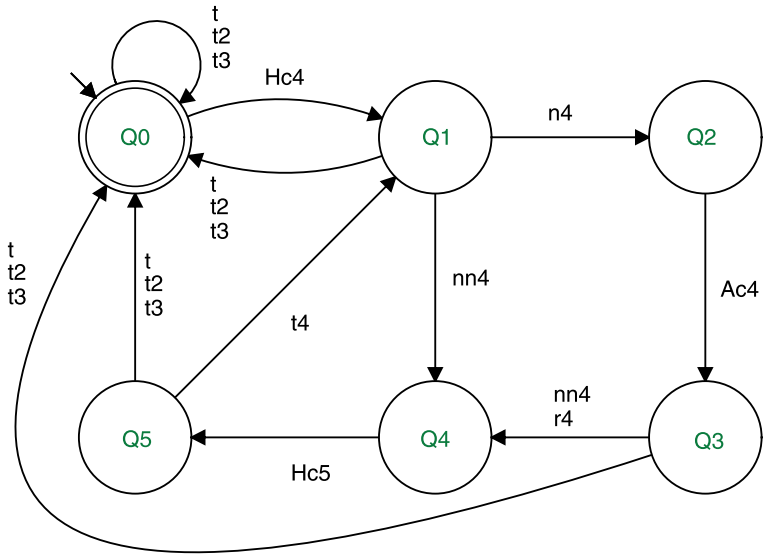


Figura 4.5: Autômato do subsistema quaternário

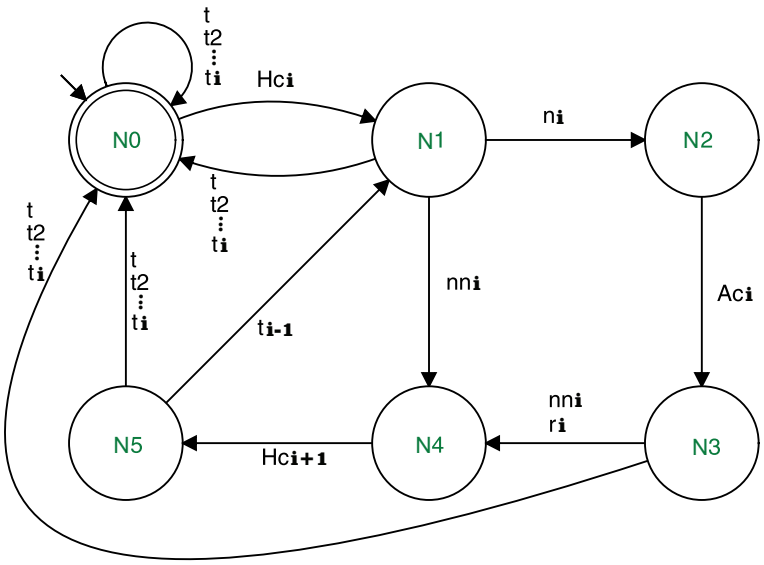


Figura 4.6: Autômato modelo para qualquer subsistema não principal

Desta forma, a linguagem para a modelagem de qualquer subsistema que não seja o principal pode ser generalizada pela expressão 4.4.

$$\begin{aligned}\mathcal{L}(N) = &\overline{\{\{t \cup t_2 \cup \ldots \cup t_{i-1}\} \cup H_{ci}\{t \cup t_2 \cup \ldots \cup t_{i-1}\} \cup H_{ci}n_iA_{ci}\{t \cup t_2 \cup \ldots \cup t_{i-1}\} \cup} \\ &\overline{\{H_{ci}nn_i \cup H_{ci}n_iA_{ci}\{r_i \cup nn_i\}\}H_{c(i+1)}\{\{t_inn_i \cup t_in_iA_{ci}\{nn_i \cup r_i\}\}H_{c(i+1)}\}^*} \\ &\overline{\{\{t \cup t_2 \cup \ldots \cup t_{i-1}\} \cup t_i\{t \cup t_2 \cup \ldots \cup t_{i-1}\} \cup t_in_iA_{ci}\{t \cup t_2 \cup \ldots \cup t_{i-1}\}\}\}^*}\end{aligned} \tag{4.4}$$

É importante destacar que quando o tempo τ_i é atingido (onde $\tau_1 = \tau$ e $t_1 = t$), todos os controladores de menor prioridade que "i" são automaticamente desabilitados e, conse-

quentemente, desativados. Por exemplo, se o tempo τ_2 é atingido então os controladores terciário, quaternário e assim por diante serão desabilitados e desativados. A partir daí, se a regra de chaveamento da camada de maior prioridade for satisfeita, os controladores inferiores voltam a ser habilitados, como observado no modelo.

Além disso, todos os tempos das camadas inferiores à "i" $(\tau_{i+1}, \tau_{i+2}...)$ são reiniciados. Por esse motivo, os tempos das camadas inferiores devem ser sempre menores que os da camadas superiores para que os ciclos possam ser completados e um sincronismo seja mantido em todo o sistema.

Uma vez modelados todos os subsistemas, o formalismo de sistemas a eventos discretos permite utilizar a operação de composição paralela de forma a gerar o autômato global que governa o sistema como um todo. Na figura 4.7 é mostrado o resultado da composição paralela para um sistema com 2 subsistemas.

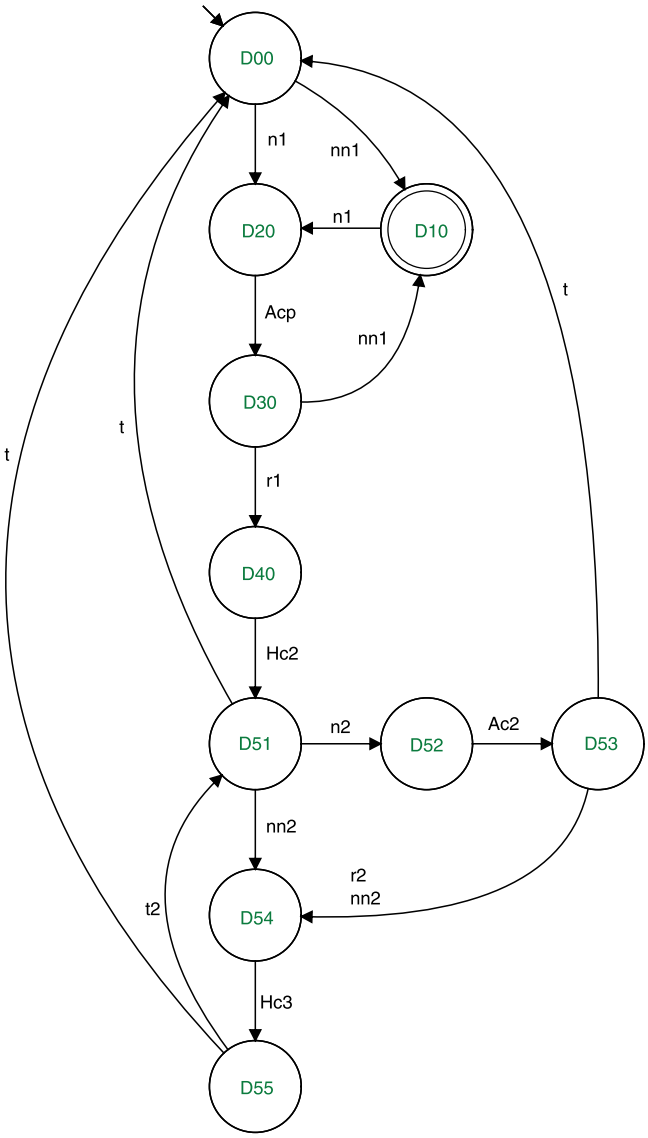


Figura 4.7: Autômato resultado da composição paralela de 2 subsistemas

Neste autômato pode-se perceber a hierarquia dada pela arquitetura de subsunção de

modo que a segunda camada só pode ser habilitada caso a regra de chaveamento imposta pela teoria de objetivo de controle principal seja satisfeita (evento r_1). Além disso, a cada intervalo de tempo τ o sistema deve voltar ao estado inicial para fazer uma nova checagem sobre o cumprimento da regra de chaveamento, não existindo nenhum estado, ou conjunto de estados, que impeçam o sistema de realizar o ciclo periodicamente. Com isso, pode-se afirmar que este autômato não possui *deadlock* ou *livelock*.

A linguagem gerada resultante da composição paralela entre os subsistemas principal e secundário é apresentada na equação 4.5. Através dela também é possível confirmar a inexistência de *deadlock* ou *livelock*.

$$\begin{aligned}\mathcal{L}(D) = &\overline{\{\{n_1A_{c1}r_1H_{c2}\{t \cup n_2A_{c2}t\} \cup \{n_1A_{c1}r_1H_{c2}\{nn_2 \cup n_2A_{c2}\{nn_2 \cup r_2\}\}H_{c3}t\}^*} \\ &\overline{\{nn_1 \cup n_1A_{c1}nn_1\}\{n_1A_{c1}nn_1\}^*} \\ &\overline{\{n_1A_{c1}r_1H_{c2}\{t \cup n_2A_{c2}t\} \cup \{n_1A_{c1}r_1H_{c2}\{nn_2 \cup n_2A_{c2}\{nn_2 \cup r_2\}\}H_{c3}t\}\}^*},\end{aligned} \tag{4.5}$$

Nesta linguagem, a primeira linha da expressão indica todas as combinações possíveis de eventos que levam o sistema a percorrer todos os estados, com exceção do estado marcado, e retornar para o estado inicial. A partir deste, o sistema pode alcançar o estado marcado através das combinações de eventos apresentados na segunda linha da expressão. Por fim, partindo do estado marcado, o sistema pode percorrer todo o autômato, voltando para o estado inicial e reiniciando o ciclo. Essa característica está representada na última linha da linguagem do autômato D.

A linguagem marcada do autômato D é a representação de todas combinações possíveis de eventos que levam o sistema a sair do estado inicial e ir até o estado marcado. Essa linguagem é apresentada na expressão a seguir, onde pode-se perceber uma grande semelhança com a linguagem gerada.

$$\begin{aligned}\mathcal{L}_m(D) = &\{\{n_1A_{c1}r_1H_{c2}\{t \cup n_2A_{c2}t\} \cup \{n_1A_{c1}r_1H_{c2}\{nn_2 \cup n_2A_{c2}\{nn_2 \cup r_2\}\}H_{c3}t\}^* \\ &\{nn_1 \cup n_1A_{c1}nn_1\}\{n_1A_{c1}nn_1\}^* \\ &\{n_1A_{c1}r_1H_{c2}\{t \cup n_2A_{c2}t\} \cup \{n_1A_{c1}r_1H_{c2}\{nn_2 \cup n_2A_{c2}\{nn_2 \cup r_2\}\}H_{c3}t\}\}^* \\ &\{nn_1 \cup n_1A_{c1}nn_1\}\{n_1A_{c1}nn_1\}^*\end{aligned} \tag{4.6}$$

A diferença entre esta linguagem e a linguagem gerada é a inexistência do fecho de prefixo e a adição do conjunto de eventos apresentados na última linha da expressão 4.6 de modo que o sistema sempre encerre o processo no estado marcado.

No entanto, pode-se perceber que o fecho de prefixo da linguagem marcada está contido na linguagem gerada, uma vez que o fecho de Kleene da mesma gera as combinações de eventos que fazem com que o sistema chegue ao estado marcado. Adicionalmente, como

a linguagem marcada possui a mesma expressão da linguagem gerada com a adição da sequência de eventos que levam o autômato ao estado marcado, pode-se afirmar que seu fecho de prefixo também contém a linguagem gerada. Portanto:

$$\overline{\mathcal{L}_m(D)} \subset \mathcal{L}(D) \tag{4.7}$$

e

$$\mathcal{L}(D) \subset \overline{\mathcal{L}_m(D)}, \tag{4.8}$$

logo,

$$\mathcal{L}(D) = \overline{\mathcal{L}_m(D)}. \tag{4.9}$$

Com isso, pode-se afirmar pela análise da linguagem que este autômato não possui *deadlock* ou *livelock*.

A seguir é apresentado o resultado da composição paralela para um sistema com 4 comportamentos realizado no *software* Supremica [41].

Neste exemplo pode-se perceber o quanto seria complicado modelar o sistema como um todo, dificultando inclusive o entendimento sobre o funcionamento da máquina de estados. Com a operação paralela e com os modelos criados para os autômatos dos comportamentos, esta modelagem pode ser feita por partes de forma simples e rápida.

Na figura 4.8, cada estado é representado por 4 dígitos que indicam a quais estados de cada camada ele pertence. Por exemplo, o estado $G5100$ indica que o autômato principal está no estado $P5$, o autômato secundário está no estado $S1$ e os autômatos terciário e quaternário estão nos estados iniciais $T0$ e $Q0$ respectivamente. Isso significa dizer que, de modo global, o sistema acabou de habilitar o controlador secundário, dado pelo evento H_{c2}, comum aos autômatos primário e secundário.

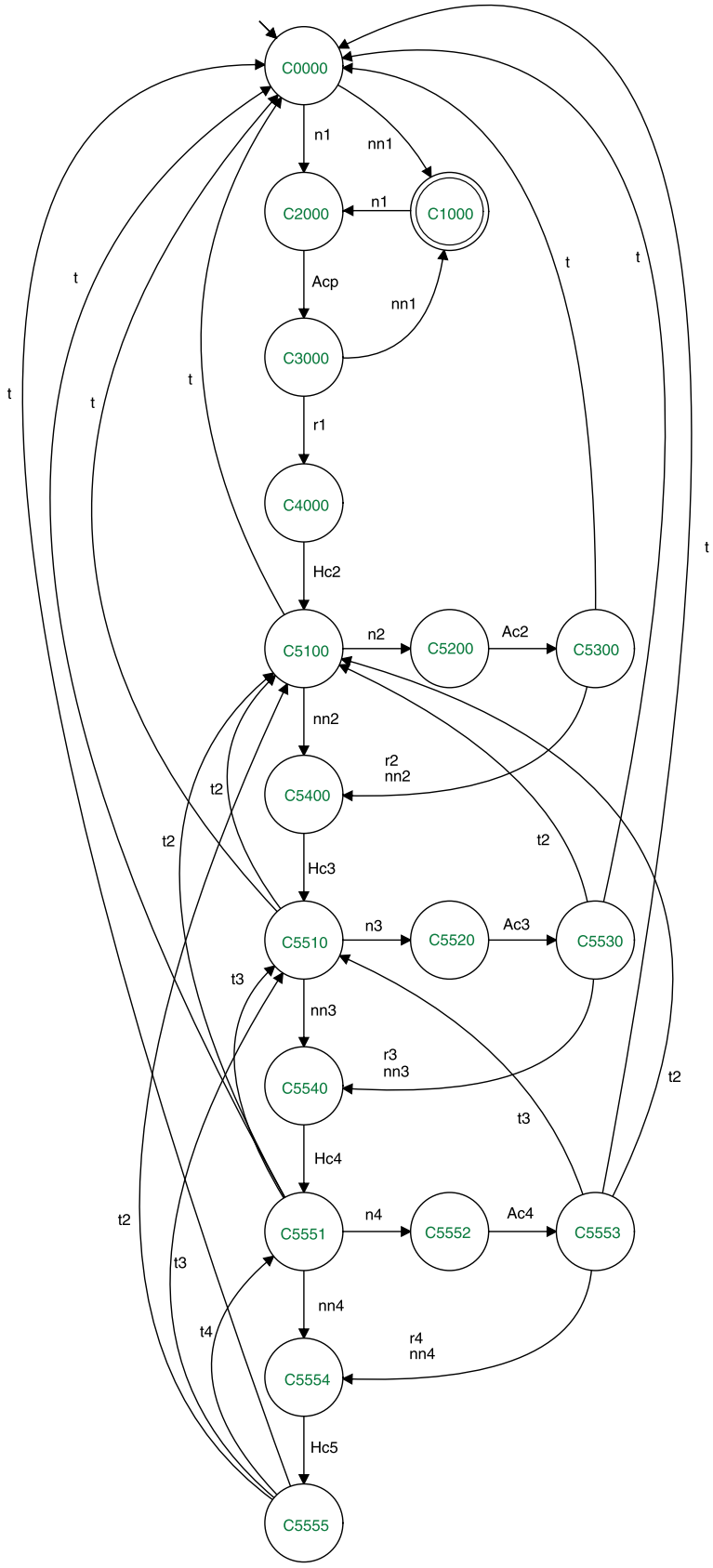


Figura 4.8: Autômato geral do modelo resultado da composição paralela de 4 subsistemas

A partir disso, é possível perceber que existe um padrão no desenvolvimento do autômato global que é dado pelo modelo criado para os autômatos de cada camada. A alteração

do estado inicial para a checagem da necessidade ou não de ativar o controlador de cada comportamento (mudança do estado "0" para o estado "1") é sempre iniciada após o evento de habilitação do respectivo controlador (H_{ci}). Sempre que isso ocorre, as camadas de maior prioridade se encontram no estado "5", que indica que a sua regra de chaveamento está sendo satisfeita e que os controladores das camadas mais baixas estão habilitados. Esses estados estão sempre esperando pelos comandos de finalização dos tempos de espera ($\tau_{i-1}, \tau_{i-2}..., \tau_1$) para que o ciclo possa ser reiniciado. Entre os estados "1" e "5" de cada autômato é onde ocorre a checagem da regra de chaveamento, da necessidade de ativar o controlador e a ativação do mesmo caso necessário.

Essa análise se repete para qualquer sistema com n comportamentos, podendo-se generalizar o resultado da aplicação da operação de composição paralela com o autômato apresentado na figura 4.9.

Com a generalização do autômato que governa todo o sistema pode-se perceber uma característica muito importante da metodologia proposta com base na teoria de sistemas a eventos discretos, que é a garantia da inexistência de *deadlock* ou *livelock*. Ou seja, o robô sempre retornará ao subsistema principal de tempos em tempos, não havendo nenhum estado ou conjunto de estados no autômato global que impeçam o sistema de retornar ao controlador principal. Com isso, todas as condições exigidas pelo teorema de chaveamento lento de controladores serão obedecidas e o sistema será *quasi*-assintoticamente estável.

Uma vez que o autômato resultante da composição de n subsistemas pode ser generalizado e sempre segue o mesmo padrão, sua linguagem também pode ser generalizada, permitindo assim a análise de inexistência de *deadlock* e *livelock* através das linguagens gerada e marcada, da mesma forma que foi feito para a combinação dos subsistemas primário e secundário. No entanto, para autômatos que englobam três ou mais subsistemas, existem muitas combinações de eventos possíveis para serem representadas e, consequentemente, as representações destas linguagens seriam muito extensas. Por este motivo esta análise não é apresentada neste trabalho.

A fim de demonstrar o funcionamento da metodologia proposta, foram executados testes com um robô real atribuindo três comportamentos diferentes que serão apresentados no próximo capítulo.

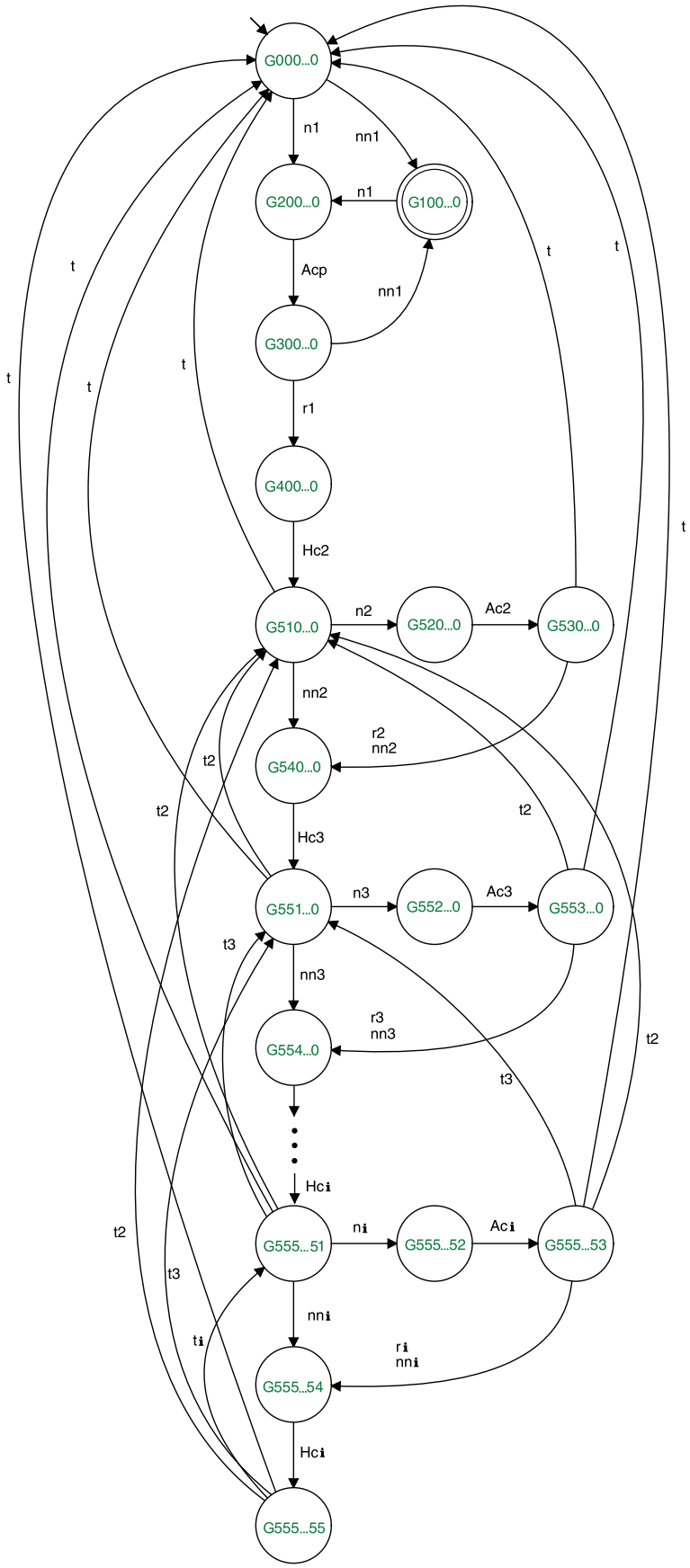


Figura 4.9: Autômato geral do modelo resultado da composição paralela de 4 subsistemas

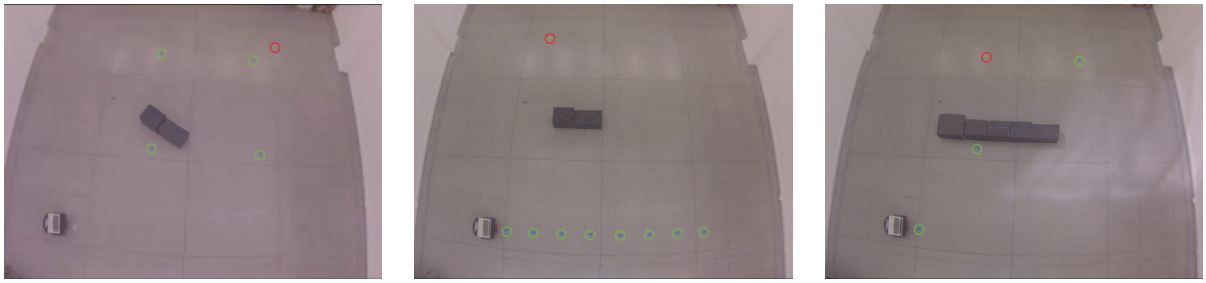
Capítulo 5

Resultados

Como forma de validar a abordagem proposta, foram realizados experimentos com o robô *Pioneer P3-DX* considerando a arquitetura de subsunção baseada na teoria de objetivo de controle principal e no teorema de estabilidade de segunda ordem, a fim de compará-las.

A posição e orientação do robô durante a navegação foram obtidas através da odometria do próprio *Pioneer* e os obstáculos foram detectados com base nos sensores ultrassônicos do mesmo. O robô é equipado com um conjunto de 16 sensores ultrassônicos dispostos em torno do robô. Eles são capazes de medir uma distância máxima de 300 cm, com um erro de medição de 5 cm.

Os experimentos consistiram na navegação do robô em três ambientes diferentes (apresentados na figura 5.1), nos quais o objetivo principal era chegar a um determinado ponto de destino. No entanto, obstáculos foram dispostos no ambiente e o robô, sem conhecimento prévio algum do ambiente, deveria desviar dos obstáculos desde que este processo não impedisse o cumprimento da tarefa principal de ir ao ponto de destino. Além disso, alguns marcos foram dispostos no ambiente, sendo considerado que o robô conseguia detectar marcos localizados à uma distância menor que 150 cm. O robô deveria capturá-los caso isso não implicasse em uma colisão com os obstáculos ou que o robô se distanciasse muito do ponto de destino.



(a) Ambiente 1 (b) Ambiente 2 (c) Ambiente 3

Figura 5.1: Ambientes utilizados para a navegação do robô. Nestas imagens, o círculo vermelho indica o ponto de destino, as caixas pretas são os obstáculos e os círculos verdes são os marcos

.

Os controladores desenvolvidos para demonstrar o funcionamento da metodologia proposta, bem como os experimentos realizados são apresentados na seção a seguir.

5.1 Controladores desenvolvidos para realização dos experimentos

Para fazer o robô chegar ao ponto de destino e para fazer o mesmo capturar os marcos, foi implementado um controlador de posição final baseado no desenvolvido em [42], sendo que para a captura dos marcos, um ponto de destino provisório é criado para o marco mais próximo do robô. Já para evitar obstáculos, foi implementado um controlador baseado no desenvolvido por [5].

É importante destacar que os controladores utilizados foram escolhidos de forma apenas a exemplificar e validar a aplicabilidade da abordagem proposta, não sendo objetivo deste trabalho o desenvolvimento de comportamentos com alto nível de performance.

Os controladores de posição final e de evitar obstáculos utilizados são detalhados nas subseções seguintes.

5.1.1 Controlador de posição final

O objetivo principal do sistema implementado é que o robô saia de um determinado ponto e chegue a outro. Dessa forma, o controlador de posição final foi definido como o controle principal do sistema, o que faz com que este deva ser assintoticamente estável segundo Lyapunov.

Para o desenvolvimento desse controlador, foram considerados as variáveis e sistemas de coordenadas apresentados na Figura 5.2.

De acordo com [42][5], as equações cinemáticas que descrevem o movimento do robô

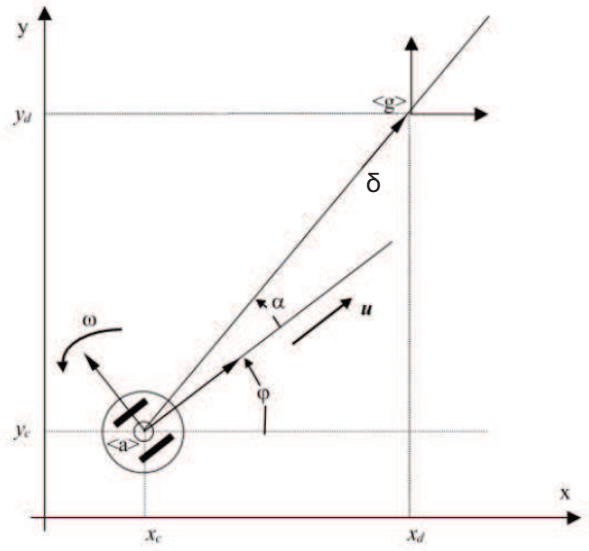


Figura 5.2: Variáveis e eixos de coordenadas relativos ao controlador de posição final [5].

no ambiente são:

$$\dot{x} = ucos(\varphi); \tag{5.1}$$

$$\dot{y} = usen(\varphi); \tag{5.2}$$

$$\dot{\varphi} = \omega; \tag{5.3}$$

em que u e ω representam, respectivamente, as velocidades linear e angular do robô e φ o ângulo entre os sistemas de coordenadas do robô e do ambiente (formado pelos eixos (x,y) da figura 5.2).

Para que o robô saia de sua posição e chegue à posição de destino, é necessário que o controlador determine as velocidades linear e angular do mesmo no sistema de coordenadas do ambiente. Para tanto, a velocidade linear será descrita por V_{pf}, enquanto que a velocidade angular por W_{pf}.

$$V_{pf} = kv_{pf}tanh(\delta)cos(\alpha) \tag{5.4}$$

$$W_{pf} = kw_{pf}\alpha + kv_{pf}\frac{tanh(\delta)}{\delta}tanh(\delta)cos(\alpha) \tag{5.5}$$

em que kv_{pf} e kw_{pf} são constantes positivas, ρ é a distância entre o sistema de coordenadas do robô e o sistema de coordenadas do destino e α o erro de orientação do robô com relação à linha que liga o robô ao destino, ambos mostrados na figura 5.2.

Segundo [42], estas equações levam o sistema a ser globalmente assintoticamente es-

tável, sendo a seguinte equação uma função de Lyapunov desse sistema:

$$V \equiv \frac{\delta^2 + \alpha^2}{2}. \tag{5.6}$$

5.1.2 Controlador de evitar obstáculos

Para o problema ilustrativo aqui abordado, o controlador de evitar obstáculos será um subsistema secundário.

Nesta abordagem será considerado um controlador inspirado no desenvolvido em [5], que consiste em manter uma velocidade linear constante e controlar a velocidade angular a partir de um ponto de destino virtual mantido a uma distância fixa do robô. Quando nenhum obstáculo é detectado, o ponto de destino virtual é posicionado no eixo x do sistema de coordenadas do robô. Caso contrário, é realizada uma rotação no ponto de destino virtual, de forma que o mesmo seja mantido oposto ao obstáculo, formando um ângulo de 180 graus entre o novo ponto de destino e o obstáculo, com relação ao robô.

As equações que regem este controle são:

$$V_{eo} = k_1; \tag{5.7}$$

$$W_{eo} = kw_{eo}\alpha' + kv_{eo}\frac{tanh(\rho')}{\rho'}sen(\alpha')cos(\alpha'); \tag{5.8}$$

em que k_1 é uma constante positiva, α' o erro de orientação do robô com relação ao ponto de destino virtual, ρ' uma distância, mantida constante, entre o robô e o ponto de destino virtual e kw_{eo} e kv_{eo} constantes positivas relativas a este controlador. Na Figura 5.3 estão ilustradas as principais variáveis envolvidas neste controlador.

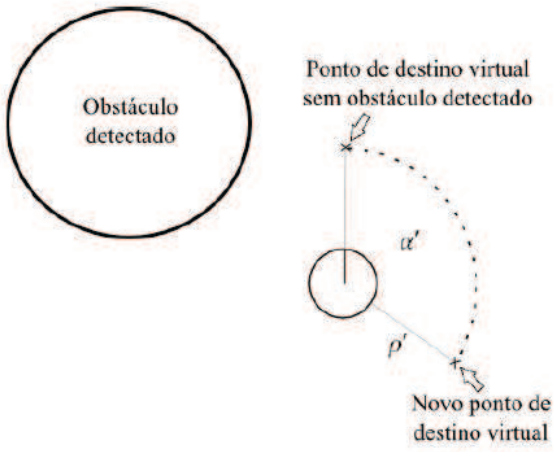


Figura 5.3: Ilustração no ambiente das principais variáveis do controlador de evitar obstáculos. [1]

É possível perceber que os controladores utilizados para demonstração do funciona-

mento da abordagem proposta são individualmente reativos, porém, nada impede que módulos deliberativos ou híbridos sejam adicionados e utilizados no sistema.

5.2 Experimentos

Para realização dos experimentos foram utilizados os mesmos controladores com os mesmos parâmetros tanto para a abordagem proposta quanto para a abordagem com regra de chaveamento baseada na derivada de segunda ordem.

Para os controladores, a velocidade linear máxima foi $V_{max} = 120cm/s$ e a velocidade angular máxima $W_{max} = 150graus/s$. Além disso, os valores das constantes utilizadas nos experimentos foram $k_1 = 30$, $kv_{eo} = 32{,}5$, $kw_{eo} = 75$, $kv_{pf} = 120$ e $kw_{pf} = 150$.

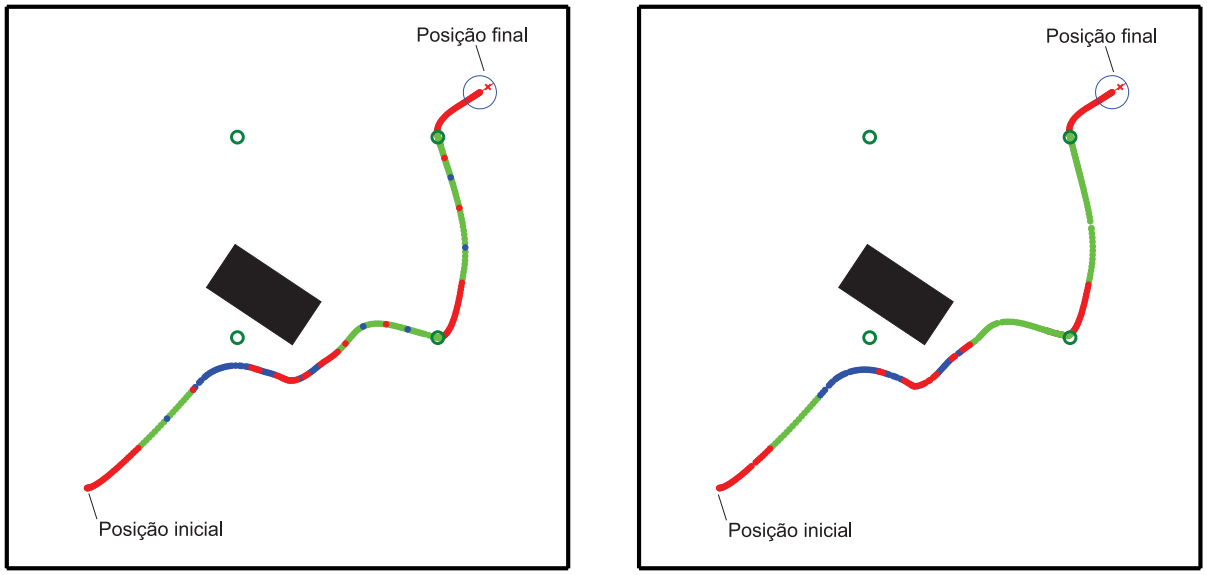
Com relação às regras de chaveamento, para a abordagem baseada na derivada de segunda ordem foi utilizado para $\ddot{W}_{max}$ um valor negativo aproximadamente igual a zero, possibilitando a maior flexibilidade possível ao sistema. Este valor indica que se houver crescimento na função SSLF, este deve ser sempre desacelerado, ou seja, $\ddot{W}(t) < 0$.

Para a abordagem baseada em objetivo de controle principal a constante ρ indica o quanto a função deve ter caído depois de um certo tempo τ. De forma a também dar uma maior flexibilidade ao sistema, para os experimentos apresentados a seguir $\rho = 0{,}1$, de maneira que seja necessário o decaimento da função SSLF, mas não necessariamente de forma acentuada. Para o tempo τ, foram utilizados alguns valores diferentes para fins de comparação.

Os experimentos realizados são apresentados nas seções a seguir. É importante destacar ainda que o robô não possuía nenhuma informação prévia do ambiente, sendo todos os controladores individualmente reativos.

5.2.1 Experimento 1

O primeiro experimento consistiu em um ambiente simples com apenas um obstáculo entre o robô e o ponto de destino e quatro marcos dispostos no ambiente. Os resultados obtidos podem ser vistos nas figuras 5.4a e 5.4b.



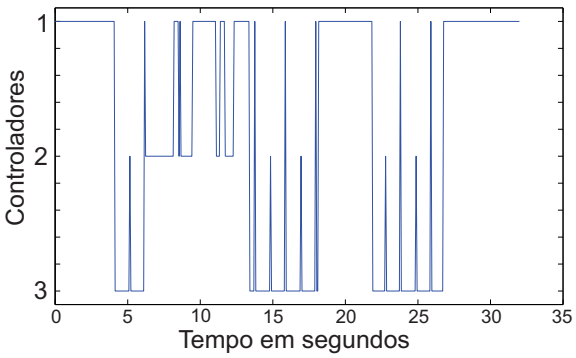
(a) Abordagem proposta com $\tau = 2s$ (b) Abordagem da derivada de segunda ordem

Figura 5.4: Trajetória realizada pelo robô durante o experimento. O vermelho indica a utilização do controlador de posição final, o azul o de evitar obstáculos e o verde o controlador responsável por levar o robô até os marcos.

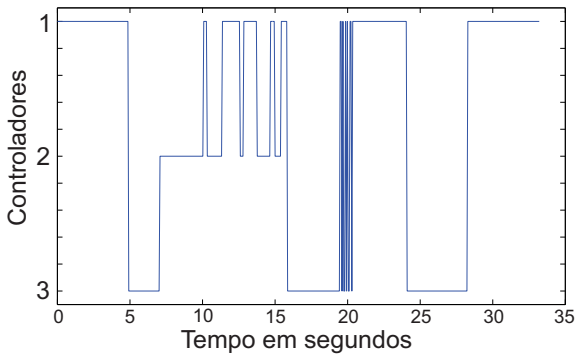
Pode-se perceber que o robô executou trajetórias bem semelhantes, cumprindo a tarefa com êxito ao alcançar o objetivo do controlador principal para ambas as abordagens. Além disso, o robô conseguiu cumprir o objetivo secundário de forma satisfatória, ao desviar do obstáculo, e ainda cumprir parte do objetivo terciário, capturando 2 marcos dos 4 dispostos no ambiente. Um dos marcos não foi capturado pelo fato de estar a uma distância maior do que a que o robô poderia detectar, já o segundo estava muito próximo do obstáculo, logo, foi dada prioridade ao desvio do mesmo.

Nessas figuras também é possível perceber quais controladores estavam ativos durante cada etapa da navegação do robô através das cores da trajetória, sendo que o vermelho indica a utilização do controlador de posição final, o azul o de evitar obstáculos e o verde o controlador responsável por levar o robô até os marcos. Na trajetória que o robô executou com base na abordagem proposta neste trabalho, pode-se perceber que em alguns pontos o robô chaveia momentaneamente para outro controlador. Isso ocorre pela característica da abordagem, na qual de tempos em tempos o sistema deve chavear para os controladores das camadas mais altas a fim de checar a regra de chaveamento. Os sinais de chaveamento dos controladores para estes experimentos são apresentados nas figuras 5.5a e 5.5b.

Comparando os dois sinais de chaveamento, pode-se perceber que para a abordagem proposta o sistema é chaveado com uma frequência bem maior do que com a abordagem baseada na derivada de segunda ordem. Isso ocorre devido à característica de periodicidade da teoria de objetivo de controle principal, na qual o sistema deve sempre voltar aos subsistemas de maior prioridade para checar se a regra de chaveamento imposta está sendo satisfeita ou não. Para este experimento, os valores τ e τ_2 foram escolhidos como

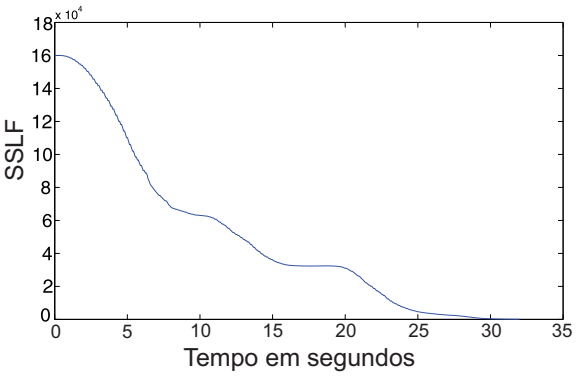


(a) Abordagem proposta com $\tau = 2s$ e com $\tau_2 = 1s$

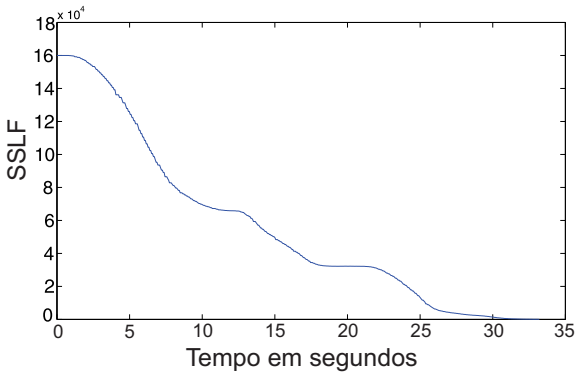


(b) Abordagem da derivada de segunda ordem

Figura 5.5: Sinais de chaveamento dos controladores durante a navegação do robô



(a) Abordagem proposta com $\tau = 2s$



(b) Abordagem da derivada de segunda ordem

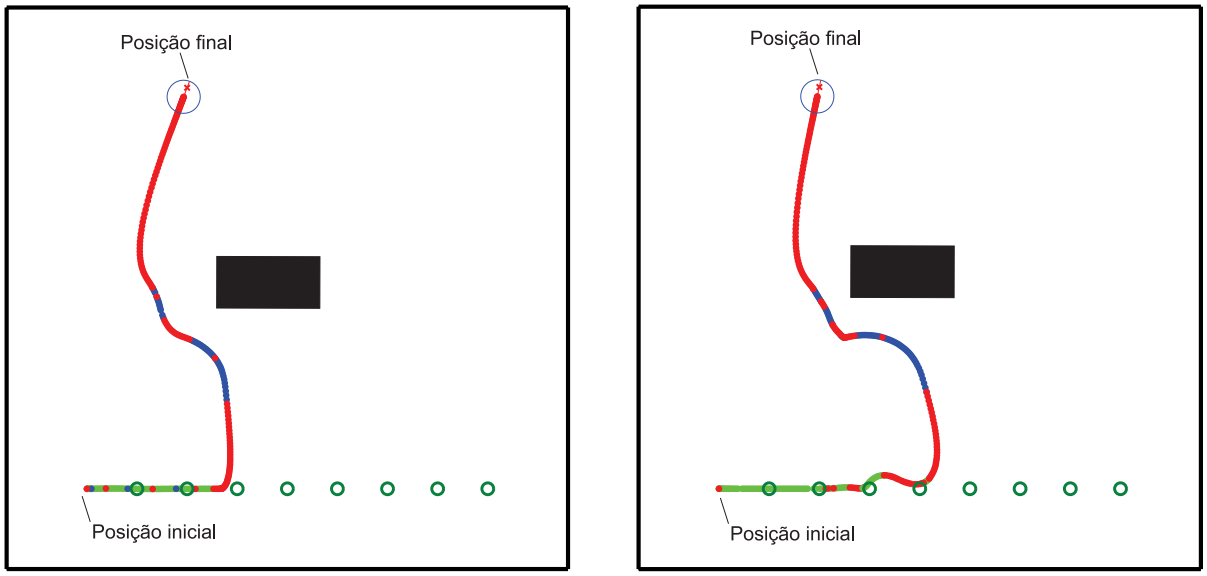
Figura 5.6: Comportamento da função SSLF do sistema para cada experimento.

sendo 2s e 1s respectivamente.

Nas figuras 5.6a e 5.6b são apresentados os comportamentos da função SSLF para cada experimento. Neste caso, pode-se perceber que para ambas as abordagens a função SSLF é sempre decrescente mesmo com o chaveamento para controladores com diferentes pontos de equilíbrio.

5.2.2 Experimento 2

Para o segundo experimento foi considerado basicamente o mesmo ambiente, no entanto, foram distribuídos oito marcos de forma a tentar induzir o robô a se afastar do ponto de destino. Inicialmente, os testes da abordagem proposta foram realizados mantendo-se os mesmos parâmetros de tempo do experimento anterior, ou seja, $\tau = 2s$ e $\tau_2 = 1s$. Os resultados obtidos com esses parâmetros e com a abordagem da derivada de segunda ordem são apresentadas nas figuras 5.7a e 5.7b respectivamente.



(a) Abordagem proposta com $\tau = 2s$ (b) Abordagem da derivada de segunda ordem

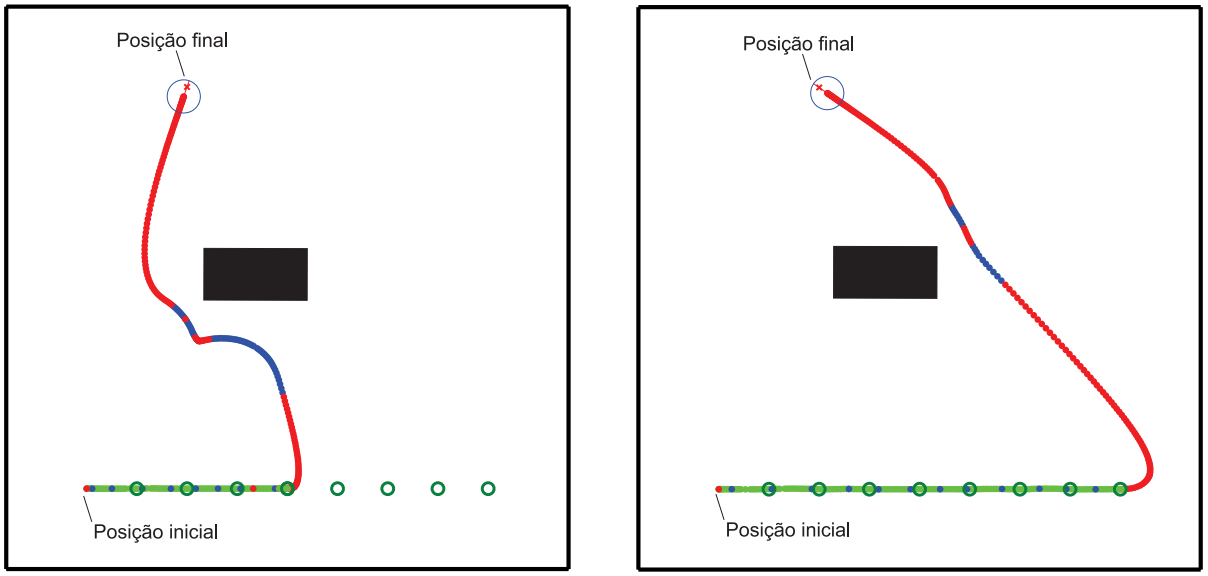
Figura 5.7: Trajetória realizada pelo robô durante o experimento.

As trajetórias novamente foram bem semelhantes, no entanto, a regra de chaveamento baseada na derivada de segunda ordem permitiu que o robô capturasse três marcos no ambiente, enquanto a regra de chaveamento proposta permitiu a captura de apenas dois marcos. A partir daí, o robô se afasta muito do ponto de destino e a função SSFL cresce mais que o limite permitido pelas regras da chaveamento. Com isso, nas duas abordagens o sistema é obrigado a chavear para o controlador principal de forma que o robô alcance o objetivo principal. Pode-se perceber ainda, que para o chaveamento baseado na derivada de segunda ordem, no momento em que o robô captura o terceiro marco ele é obrigado a chavear para o controlador principal e realiza uma curva passando muito próximo ao quarto marco, mas sem capturál-lo.

A grande diferença é que a abordagem da derivada de segunda ordem não possibilita uma maior flexibilização no que diz respeito a desviar do objetivo principal, de forma que para este experimento o robô não consegue capturar mais de três marcos antes que seja obrigado a voltar para o controlador de posição final. Já com a abordagem proposta, esta flexibilização pode ser dada através de um valor maior das constantes de tempo de chaveamento, permitindo que o robô se afaste por mais tempo até que também seja obrigado a retornar ao objetivo principal. Sendo assim, o experimento foi repetido mais duas vezes, o primeiro com valores de $\tau = 8s$ e $\tau_2 = 1s$ e o segundo com $\tau = 16s$ e $\tau_2 = 1s$. Os resultados obtidos são mostrados nas figuras 5.8a e 5.8b.

Com um tempo $\tau = 8s$, o robô passou a conseguir capturar quatro marcos antes de voltar ao objetivo principal. Já para um valor de $\tau = 16s$ todos os marcos puderam ser capturados.

Os sinais de chaveamento de todos esses experimentos são apresentados na figura 5.9,



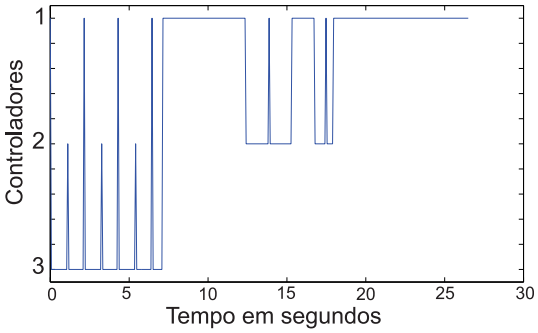
(a) Abordagem proposta com $\tau = 8s$

(b) Abordagem proposta com $\tau = 16s$

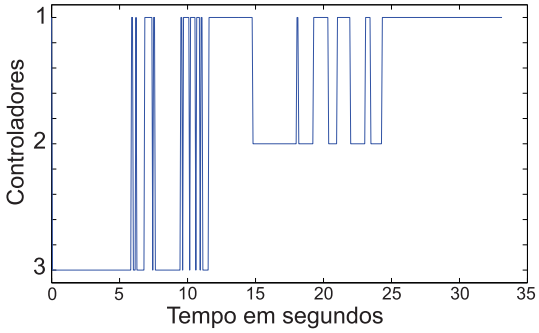
Figura 5.8: Trajetória realizada pelo robô durante o experimento 2 com valores de $\tau = 8s$ e $\tau = 16s$ respectivamente.

onde pode-se perceber claramente o aumento do tempo necessário até que o controlador principal volte a ser ativado.

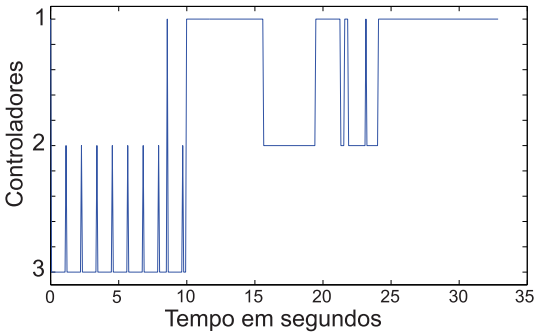
O aumento no valor de τ indica que a função de energia do sistema pode crescer por mais tempo até que haja a obrigatoriedade de chavear o sistema para o controlador principal. Na figura 5.10 são apresentados os gráficos da função SSLF dos 4 experimentos, onde pode-se perceber que para os tempos de 8 e 16 segnudos há um crescimento mais acentuado da função de energia.



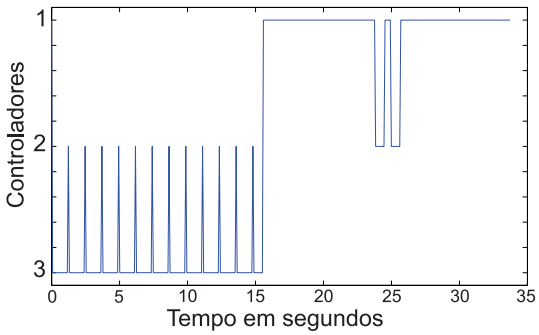
(a) Abordagem proposta com $\tau = 2s$ e $\tau_2 = 1s$



(b) Abordagem da derivada de segunda ordem

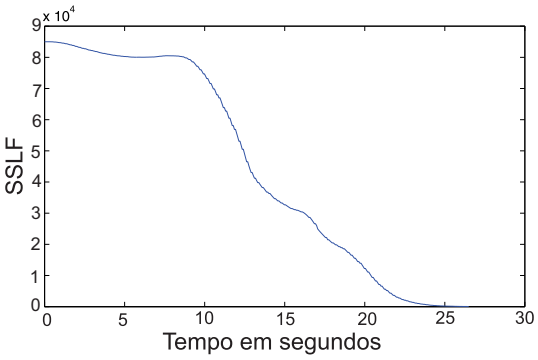


(c) Abordagem proposta com $\tau = 8s$ e $\tau_2 = 1s$

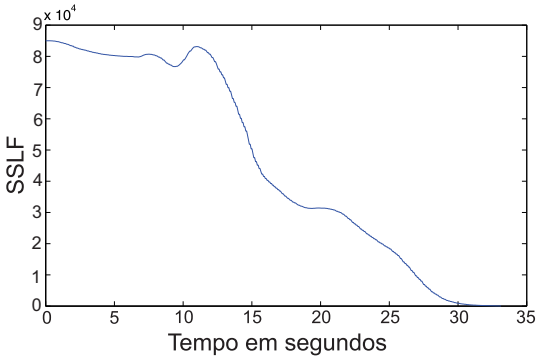


(d) Abordagem proposta com $\tau = 16s$ e $\tau_2 = 1s$

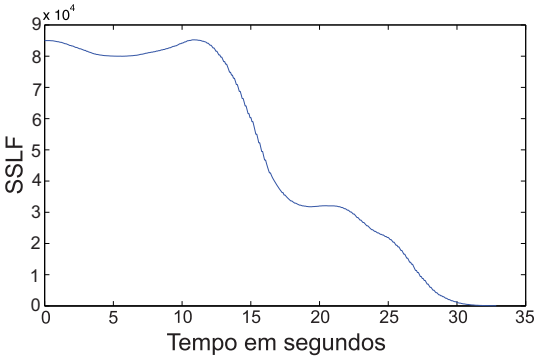
Figura 5.9: Sinais de chaveamento dos controladores durante a navegação do robô.



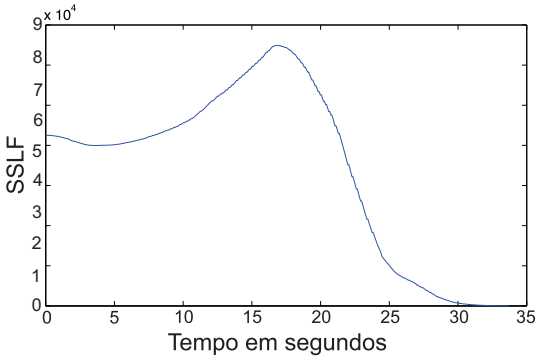
(a) Abordagem proposta com $\tau = 2s$



(b) Abordagem da derivada de segunda ordem



(c) Abordagem proposta com $\tau = 8s$



(d) Abordagem proposta com $\tau = 16s$

Figura 5.10: Comportamento da função SSLF do sistema para cada experimento.

5.2.3 Experimento 3

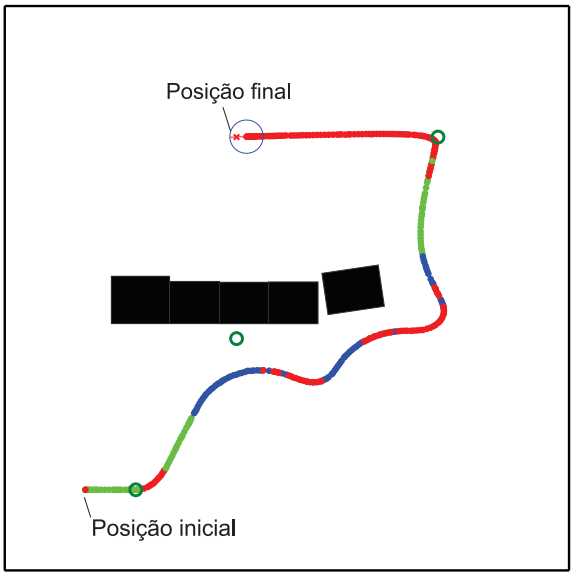
Para o experimento 3, os obstáculos foram dispostos no ambiente de forma que o desvio do mesmo leva o robô a se distanciar cada vez mais do ponto de destino, como apresentado na figura 5.11a. Os resultados obtidos com a abordagem proposta, para constantes de tempo $\tau = 2s$ e $\tau_2 = 1s$, e com a abordagem da derivada de segunda ordem são apresentadas nas figuras 5.11c e 5.11b respectivamente.

Em ambos os casos, a regra de chaveamento utilizada leva o robô a colidir com o obstáculo a fim de chegar ao ponto de destino. Este resultado já era esperado, uma vez que, de modo a ilustrar a abordagem proposta neste trabalho, foi atribuído ao robô o objetivo principal de chegar ao ponto de destino, mesmo que para isso ele deixe de capturar alguns marcos ou, inclusive, colida com obstáculos. Dessa forma, o robô arrastou os obstáculos do ambiente e chegou ao ponto de destino, conforme garantido tanto pela teoria de objetivo de controle principal quanto pela teoria da estabilidade de segunda ordem. Para casos em que os obstáculos sejam intransponíveis ou que evitar de obstáculos seja uma característica mais importante do que chegar ao ponto de destino, pode-se determinar como objetivo principal do sistema que o robô navegue pelo ambiente desviando de obstáculos. Neste caso, o robô não colidiria com os obstáculos, mas não seria possível garantir que ele chegaria ao ponto de destino.

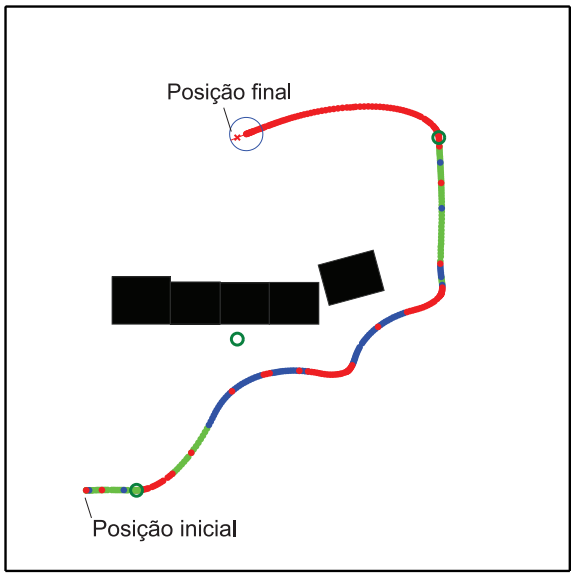
Mantendo o ponto de destino como o objetivo principal do robô, a colisão com o obstáculo pode ser evitada caso uma maior flexibilidade seja dada ao sistema. No entanto, apenas a abordagem proposta neste trabalho permite isso. Por esse motivo, um novo experimento foi realizado, agora com uma constante de tempo de chaveamento $\tau = 4s$. O resultado é apresentado na figura 5.11d, onde pode-se perceber que o robô foi capaz de desviar do obstáculo e ainda capturar um marco que estava posicionado longe do ponto de destino. No entanto, uma maior flexibilização não é garantia de melhor desempenho. A depender do ambiente, uma maior flexibilização pode levar o robô a não cumprir objetivos secundários. Portanto, a escolha dos parâmetros de flexibilização é muito importante. Uma estratégia para a escolha dos melhores valores para esses parâmetros ainda não foi desenvolvida e será investigada em trabalhos futuros.

Os sinais do chaveamento dos controladores para os três experimentos estão apresentados na figura 5.12, enquanto as funções SSLF estão apresentadas na figura 5.13.

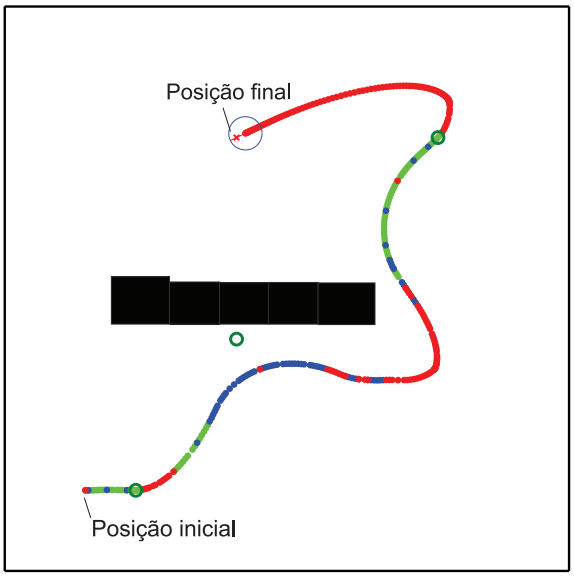
(a) Estado inicial para o experimento 3



(b) Abordagem da derivada de segunda ordem

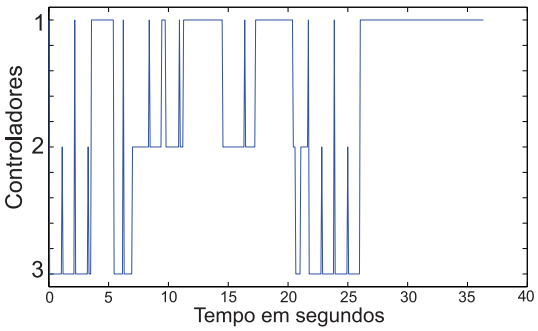


(c) Abordagem proposta com $\tau = 2s$

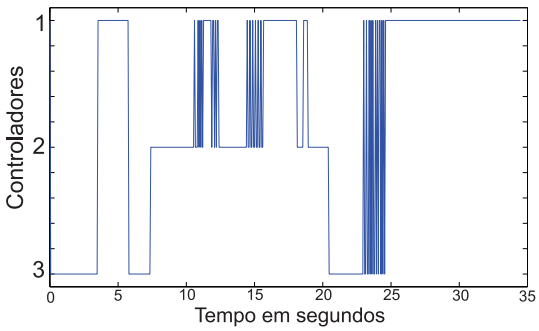


(d) Abordagem proposta com $\tau = 4s$

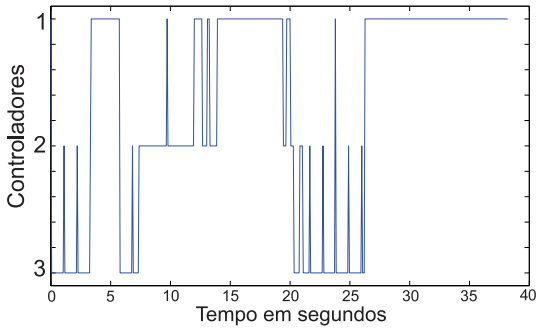
Figura 5.11: Trajetória realizada pelo robô durante o experimento.



(a) Abordagem proposta com $\tau = 2s$ e $\tau_2 = 1s$

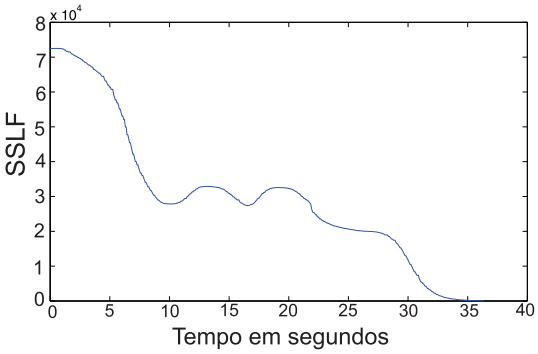


(b) Abordagem da derivada de segunda ordem

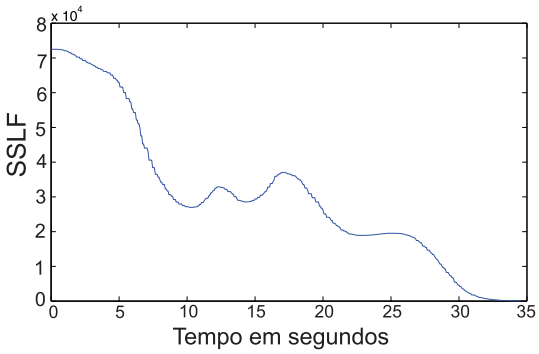


(c) Abordagem proposta com $\tau = 4s$ e $\tau_2 = 1s$

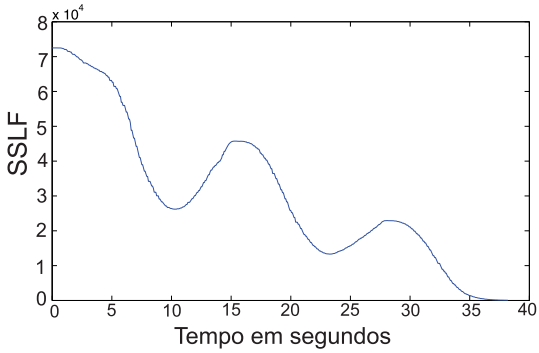
Figura 5.12: Sinais de chaveamento dos controladores durante a navegação do robô.



(a) Abordagem proposta com $\tau = 2s$ e $\tau_2 = 1s$



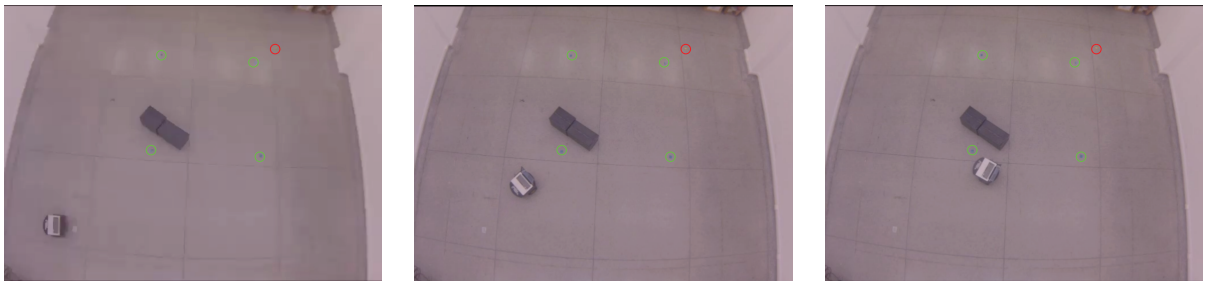
(b) Abordagem da derivada de segunda ordem



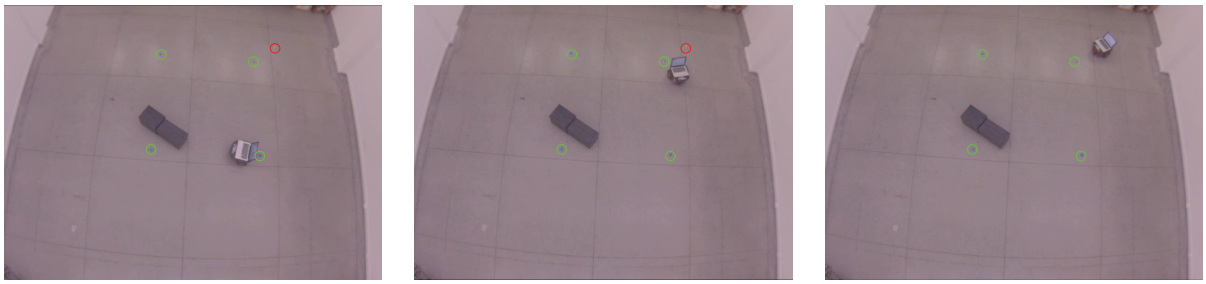
(c) Abordagem proposta com $\tau = 4s$ e $\tau_2 = 1s$

Figura 5.13: Sinais de chaveamento dos controladores durante a navegação do robô.

Nas figuras 5.14, 5.17 e 5.16 são apresentadas imagens de alguns dos experimentos a fim de ilustrar como os mesmos foram realizados.



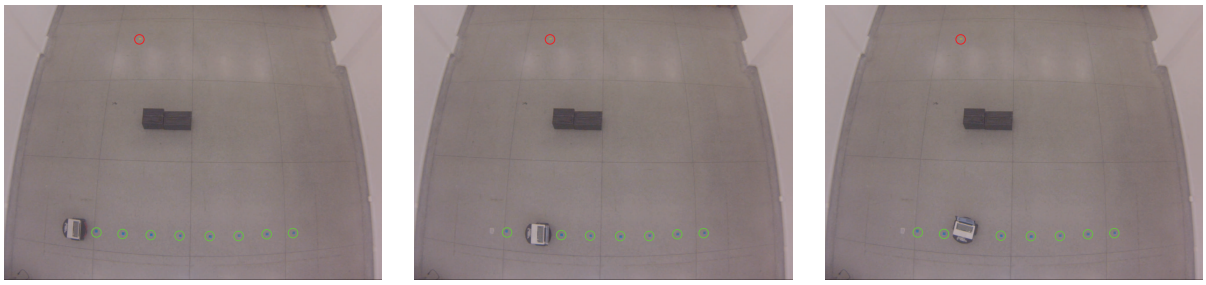
(a) Frame 1 (b) Frame 2 (c) Frame 3



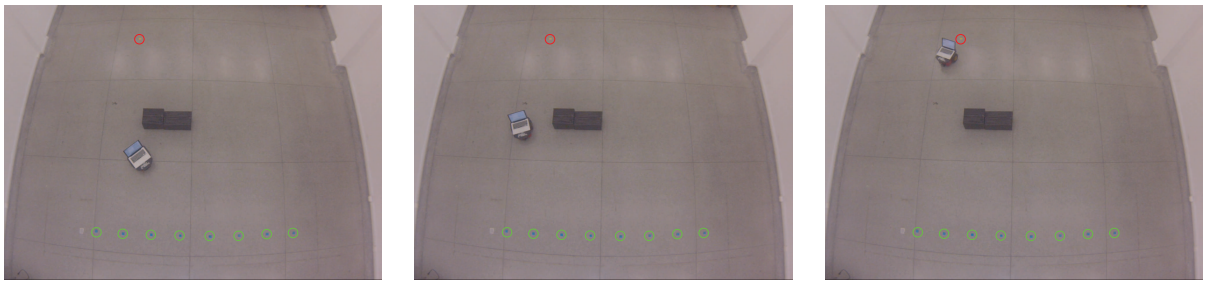
(d) Frame 4 (e) Frame 5 (f) Frame 6

Figura 5.14: Imagens do experimento real realizado com a abordagem proposta no primeiro ambiente com $\tau = 2s$, cuja trajetória foi apresentada na figura 5.4a

.



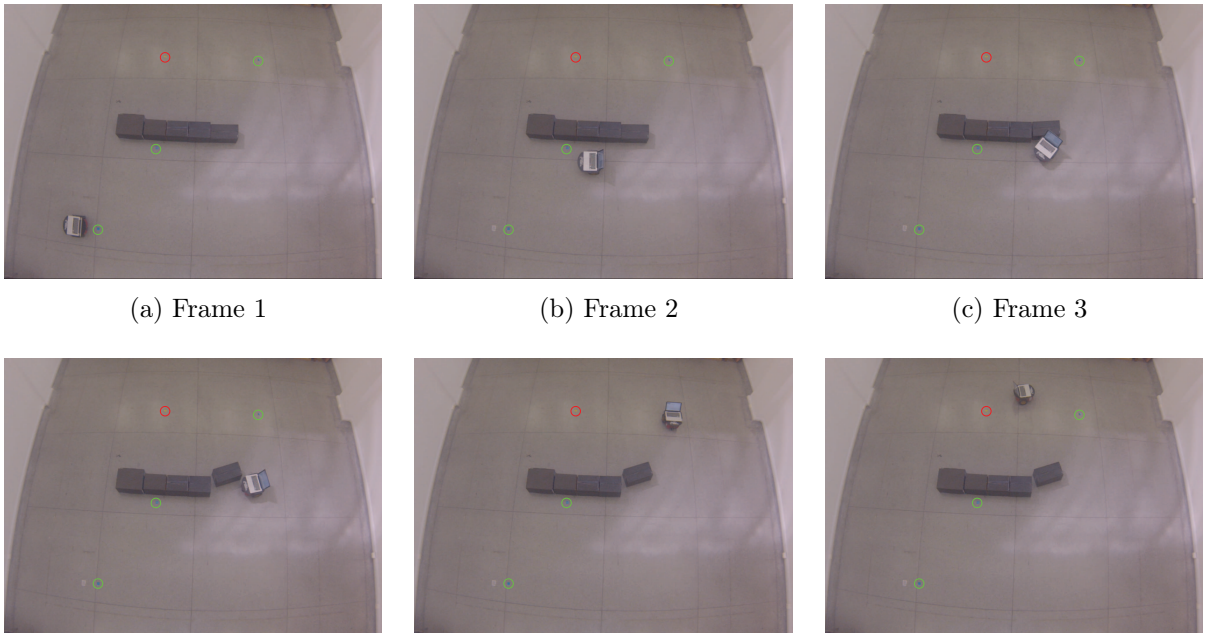
(a) Frame 1 (b) Frame 2 (c) Frame 3



(d) Frame 4 (e) Frame 5 (f) Frame 6

Figura 5.15: Imagens do experimento real realizado com a abordagem proposta no segundo ambiente com $\tau = 8s$, cuja trajetória foi apresentada na figura 5.8a

.

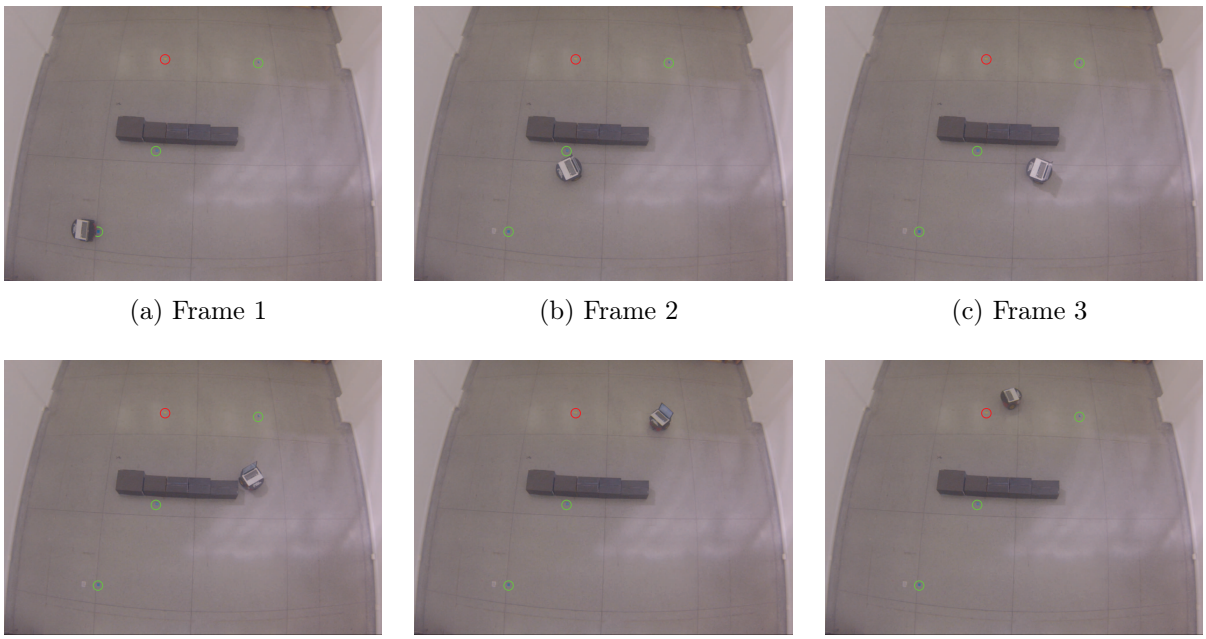


(a) Frame 1 (b) Frame 2 (c) Frame 3

(d) Frame 4 (e) Frame 5 (f) Frame 6

Figura 5.16: Imagens do experimento real realizado com a abordagem proposta no terceiro ambiente com $\tau = 2s$, cuja trajetória foi apresentada na figura 5.11c

.



(a) Frame 1 (b) Frame 2 (c) Frame 3

(d) Frame 4 (e) Frame 5 (f) Frame 6

Figura 5.17: Imagens do experimento real realizado com a abordagem proposta no terceiro ambiente com $\tau = 4s$, cuja trajetória foi apresentada na figura 5.11d

.

É importante destacar que o objetivo do trabalho não é o desenvolvimento de uma

arquitetura de navegação robusta, e sim de uma metodologia de implementação de arquiteturas de navegação de forma hierárquica, na qual o objetivo principal é garantido e os demais objetivos são buscados, desde que não atrapalhem os objetivos das camadas de maior prioridade. Por esse motivo não foram realizados testes com ambientes mais desafiadores com obstáculos clássico da robótica, como por exemplo os obstáculos em forma de "U". Para este caso um controlador de evitar obstáculos mais robusto, tal qual o desvio tangencial [43], poderia ser utilizado.

Na próxima seção é apresentada uma comparação dos resultados obtidos com as abordagens apresentadas.

5.3 Comparação dos resultados

A fim de comparar os resultados obtidos com a aplicação das abordagens baseadas em objetivo de controle principal e na derivada de segunda ordem, foram utilizadas as métricas apresentadas em [44], que fornecem uma indicação da qualidade da navegação, útil para comparar e analisar arquiteturas de controle de robôs móveis. Neste trabalho são apresentadas métricas de segurança e uma métrica dimensional.

As métricas de segurança são dividias em $SM1$, $SM2$ e SM_{min}, em que $SM1$ é a distância média entre o robô e os obstáculos durante toda a navegação; $SM2$ é a média dos menores valores de distância medidos pelos sensores, dando uma ideia do risco assumido durante a navegação, em termos da proximidade de um obstáculo; e SM_{min} é a menor distância em relação aos obstáculos que o robô esteve durante a tarefa, apresentando o risco máximo assumido.

A métrica dimensional é o comprimento total da trajetória realizada pelo robô (P_L) desde o ponto inicial até o ponto de destino. Para uma trajetória no plano $x-y$, composta por n pontos, onde $(x_1, f(x_1))$ é o ponto inicial e $(x_n, f(x_n))$ é o ponto de destino, P_L é calculado como [44]:

$$P_L = \sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (f(x_{i+1}) - f(x_i))^2}. \tag{5.9}$$

Para os experimentos realizados, tanto os valores das métricas de segurança quanto o valor do comprimento total da trajetória foram calculados em centímetros.

Além dessas métricas, as abordagens foram comparadas no que diz respeito ao número de marcos capturados (N_{MC}).

Na tabelas 5.1, 5.2 e 5.3 é apresentada a comparação feita para os experimentos realizados.

Tabela 5.1: Experimento 1

Abordagem	$SM1$	$SM2$	SM_{min}	P_L	N_{MC}
Objetivo principal ($\tau = 2s$)	265,5	193,4	30,7	699,3	2
Derivada de 2ª ordem	264,3	187,7	20,5	693,9	2

Tabela 5.2: Experimento 2

Abordagem	$SM1$	$SM2$	SM_{min}	P_L	N_{MC}
Objetivo principal ($\tau = 2s$)	266,3	185,1	34,9	568,3	2
Objetivo principal ($\tau = 8s$)	266,1	171,7	23,4	685,1	4
Objetivo principal ($\tau = 16s$)	270,2	194,2	37,3	941,3	8
Derivada de 2ª ordem	263,4	163,4	17,9	674,3	3

Tabela 5.3: Experimento 3

Abordagem	$SM1$	$SM2$	SM_{min}	P_L	N_{MC}
Objetivo principal ($\tau = 2s$)	255,5	136,3	0	800,2	1
Objetivo principal ($\tau = 4s$)	262,1	161,5	18,2	922,6	2
Derivada de 2ª ordem	260,8	142,3	0	807,2	1

Para o experimento 1, todos os parâmetros tiveram valores muito próximos, sendo que o robô executou uma trajetória segura, capturando dois marcos durante a navegação. Isso ocorreu pois as regras de chaveamento não impediram a execução dos controladores de menor prioridade. Sendo assim, como foram utilizados os mesmos parâmetros para os controladores de ambas abordagens, o resultado foi bem similar.

Para o experimento 2, o chaveamento baseado na derivada de segunda ordem gerou uma trajetória mais longa do que o chaveamento com a abordagem proposta para $\tau = 2s$, porém, menor do que os experimentos com $\tau = 8s$ e $\tau = 16s$. Isto é reflexo do número de marcos capturados em cada experimento, sendo que com a abordagem proposta foi possível capturar todos os marcos a partir da variação do valor de τ, enquanto para a abordagem da derivada de segunda ordem só foi possível capturar 3 marcos.

Em todos os testes realizados para o experimento 2, o robô navegou a uma distância segura em relação aos obstáculos, com média dos menores valores medidos pelos sensores em torno de $170cm$.

No terceiro experimento, tanto para a abordagem baseada na derivada de segunda ordem quanto para a abordagem proposta com $\tau = 2s$, houve colisão do robô com os obstáculos, sendo o valor de SM_{min} para os dois casos igual a zero, e apenas um marco foi capturado. No entanto, ao aumentar o valor de τ para $7s$ na abordagem proposta, o obstáculo pôde ser evitado e um outro marco ainda foi capturado.

Os resultados obtidos no experimento 2 e 3 deixam clara a importância da escolha do valor das constantes de tempo na abordagem proposta. Até o momento não foi desenvolvido nenhum método para determinar o melhor valor dessas constantes, ficando a cargo do projetista determinar o valor de acordo com a flexibilidade que se deseja dar ao sistema, lembrando que uma maior flexibilidade não significa um melhor desempenho. Essa flexibilidade não pode ser dada na abordagem baseada na derivada de segunda ordem, o que mostra a vantagem da aplicação da abordagem proposta.

Capítulo 6

Considerações Finais

Neste trabalho foi apresentada uma nova metodologia para provar a estabilidade de uma arquitetura de subsunção baseada na teoria de controle chaveado com objetivo de controle principal, além de um formalismo baseado na teoria de sistemas a eventos discretos capaz de permitir a modelagem dos comportamentos de forma simples e rápida, mesmo para sistemas que possuem muitos comportamentos.

A teoria de controle chaveado com objetivo de controle principal apresentada em [1] garante que, se existe um controlador principal globalmente assintoticamente estável, o sistema pode chavear para outros controladores, que podem ser estáveis ou não, e desde que o chaveamento siga as regras impostas ao crescimento da função candidata de Lyapunov, o sistema será *quasi*-assintoticamente estável. Esta ideia foi aplicada diretamente à arquitetura de subsunção apresentada por Brooks em [3] de modo que o cumprimento do objetivo do comportamento de maior prioridade é garantida. Além disso, o sistema foi modelado de maneira que para cada camada o controlador é considerado como principal em relação as camadas inferiores. No entanto, pela teoria de objetivo de controle principal, apenas o cumprimento do objetivo principal será garantido.

A garantia de atendimento da regra de chaveamento é dada pelo formalismo de sistemas a eventos discretos. Esta teoria permitiu a modelagem de diferentes comportamentos e suas interações de maneira formal e simples, de forma que o comportamento global do sistema pode ser desenvolvido por partes. Isto foi feito através dos dois modelos de autômatos apresentados, um para a modelagem do subsistema principal e outro para os demais subsistemas, inclusive outros que venham a ser adicionados ao sistema.

A adição de comportamentos é uma importante característica do formalismo proposto, uma vez que a teoria de sistemas a eventos discretos permite a modelagem do comportamento global do sistema através da composição paralela dos autômatos de cada camada. Sendo assim, adicionar um novo comportamento significa apenas realizar mais uma vez a operação de composição paralela entre o autômato que governa todo o sistema com o novo comportamento que deseja-se adicionar, sem a necessidade de alterar os comportamentos previamente modelados.

Através dos modelos de autômatos propostos e da operação de composição paralela foi possível ainda generalizar o autômato global do sistema para n comportamentos, garantindo a inexistência de *deadlocks* ou *livelocks* e, consequentemente, garantindo o funcionamento do modelo.

Os resultados obtidos demonstram a importância e funcionalidade da abordagem proposta, sendo que o objetivo principal do sistema foi alcançado em todos os experimentos realizados. Adicionalmente, a abordagem proposta se mostrou mais robusta em relação à única abordagem encontrada na literatura a fim de garantir a estabilidade de uma arquitetura de subsunção, pois possui mais flexibilidade no ajuste dos parâmetros de modo a conseguir cumprir objetivos secundários de forma mais abrangente.

A flexibilidade do sistema é determinada pelas constantes ρ, que impõe o quanto a função SSLF deve decrescer, e as constantes de tempo de chaveamento, que indicam por quanto tempo a função SSLF do sistema pode crescer e devem ser escolhidas de acordo com a flexibilidade que se deseja dar ao sistema. Lembrando que nem sempre uma maior flexibilidade significa um melhor desempenho.

Como trabalhos futuros propõe-se um estudo que permita escolher os valores das constantes de tempo automaticamente, de forma que mais objetivos secundários possam ser alcançados. Além disso, é proposta uma análise de estabilidade das demais camadas caso seja utilizado em todas elas apenas controladores globalmente assintoticamente estáveis. Para isso, é possível que algumas alterações no modelo sejam necessárias, tais como a mudança na forma de sincronismo dos tempos τ_i de chaveamento de cada camada. O objetivo é encontrar um conjunto de restrições que se forem atendidas possam garantir o cumprimento dos demais objetivos.

Referências Bibliográficas

[1] Elyson Ádan Nunes Carvalho. *Estabilidade de Sistemas Chaveados Baseados em Objetivo de Controle Principal*. Tese de doutorado, Universidade Federal de Campina Grande, UFCG, 2012.

[2] Lucas Molina. *Desenvolvimento de uma arquitetura de navegação deliberativa para robôs móveis utilizando a teoria de controle supervisório*. Universidade Federal do Rio de Janeiro, COPPE, 2010.

[3] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, 2(1):14–23, 1986.

[4] A. Harper, C.; Winfield. A methodology for provably stable behaviour-based intelligent control. *Robotics and Autonomous Systems, vol: 54 (1) pp: 52-73*, 2006.

[5] Eduardo Oliveira Freire. *Controle de Robô Móveis por Fusão de Sinais de Controle Usando Filtro de Informação Descentralizado*. Tese de Doutorado, Universidade Federal de do Espírito Santo, 2002.

[6] Roland Siegwart, Illah Reza Nourbakhsh e Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[7] Ananth Ranganathan e Sven Koenig. A reactive robot architecture with planning on demand. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1462–1468. IEEE, 2003.

[8] Stuart Russell, Peter Norvig e Artificial Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25:27, 1995.

[9] Ronald C Arkin. Behavior-based robotics (intelligent robotics and autonomous agents). 1998.

[10] Paolo Pirjanian. An overview of system architectures for action selection in mobile robotics. *Tech-report*, 1997.

[11] P. Pirjanian. Multiple objective action selection and behavior fusion using voting. *PhD thesis, Department of Medical Informatics and Image Analysis, Institute of Electronic Systems, Aalborg University*, 1998.

[12] P. Pirjanian. Behavior coordination mechanisms - state-of-the-art. *Institute of Robotics and Intelligent Systems, School of Engineering, University of Southern California.*, 1999.

[13] Rodney Allen Brooks. *Cambrian intelligence: The early history of the new AI*, volume 44. Mit Press Cambridge, MA, 1999.

[14] Hassan K Khalil. *Noninear Systems*. Prentice-Hall, New Jersey, 1996.

[15] Vangipuram Lakshmikantham, Srinivasa Leela e Anatoliui Andreevich Martynyuk. *Practical stability of nonlinear systems*. World Scientific, 1990.

[16] Christos G Cassandras e Stephane Lafortune. *Introduction to discrete event systems*. Springer Science & Business Media, 2009.

[17] James Bellingham, Thomas Consi, Robert Beaton e William Hall. Keeping layered control simple [autonomous underwater vehicles]. In *Autonomous Underwater Vehicle Technology, 1990. AUV'90., Proceedings of the (1990) Symposium on*, pages 3–8. IEEE, 1990.

[18] R Peter Bonasso. Integrating reaction plans and layered competences through synchronous control. In *IJCAI*, pages 1225–1233, 1991.

[19] Ralph Hartley e Frank Pipitone. Experiments with the subsumption architecture. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1652–1658. IEEE, 1991.

[20] Jonathan H Connell. Sss: A hybrid architecture applied to robot navigation. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 2719–2724. IEEE, 1992.

[21] John R Koza. Evolution of subsumption using genetic programming. In *Proceedings of the First European Conference on Artificial Life*, pages 110–119, 1992.

[22] J. J. Bellinghan, J. G.; Leonard. Task configuration with layered control. *International Advanced Robotics Programme, Mobile for Subsea Environments*, 1994.

[23] Stefano Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *Robotics and Automation, IEEE Transactions on*, 13(3):398–410, 1997.

[24] Hideyuki Nakashima e Itsuki Noda. Dynamic subsumption architecture for programming intelligent agents. In *icmas*, page 190. IEEE, 1998.

[25] Julian Togelius. Evolution of a subsumption architecture neurocontroller. *Journal of Intelligent & Fuzzy Systems*, 15(1):15–20, 2004.

[26] Yan Yongjie, Zhu Qidan e Cai Chengtao. Hybrid control architecture of mobile robot based on subsumption architecture. In *Mechatronics and Automation, Proceedings of the 2006 IEEE International Conference on*, pages 2168–2172. IEEE, 2006.

[27] Gianluca Antonelli, Filippo Arrichiello e Stefano Chiaverini. The null-space-based behavioral control for autonomous robotic systems. *Intelligent Service Robotics*, 1(1):27–39, 2008.

[28] MA Godin, JG Bellingham, B Kieft e R McEwen. Scripting language for state configured layered control of the tethys long range autonomous underwater vehicle. In *OCEANS 2010*, pages 1–7. IEEE, 2010.

[29] Jaime Beltran e Jonatan Gomez. Subsumption architecture for motion learning in robots. In *Computing Congress (CCC), 2012 7th Colombian*, pages 1–6. IEEE, 2012.

[30] Guanqun Liu, Haibo Tong e Rubo Zhang. An intelligent control architecture for search robot based on orthogonal perception information. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 2348–2352. IEEE, 2012.

[31] Fusaomi Nagata, Akimasa Otsuka, Keigo Watanabe e Maki K Habib. Multiple mobile robots system with network-based subsumption architecture. *International Journal of Mechatronics and Manufacturing Systems*, 6(1):57–71, 2013.

[32] Jay Thor Turner, Sidney N Givigi e Alain Beaulieu. Implementation of a subsumption based architecture using model-driven development. In *Systems Conference (SysCon), 2013 IEEE International*, pages 331–338. IEEE, 2013.

[33] Hu Chunhua, Pang Baobing e Si Feifei. Design of intelligent car for tracking trajectory based on subsumption architecture. *Shanxi Electronic Technology*, 1:001, 2014.

[34] Daan Krijnen e Coen Dekker. Ar drone 2.0 with subsumption architecture. In *Artificial intelligence research seminar*, 2014.

[35] Espen Oland, Tom Stian Andersen e Raymond Kristiansen. Subsumption architecture applied to flight control using composite rotations. *Automatica*, 69:195–200, 2016.

[36] Daniel Toal, Colin Flanagan, Caimin Jones e Bob Strunz. Subsumption architecture for the control of robots. In *Proc. of the IMC*, volume 13. Citeseer, 1996.

[37] J Kenneth Rosenblatt e David W Payton. A fine-grained alternative to the subsumption architecture for mobile robot control. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, pages 317–323. IEEE, 1989.

[38] Eyal Amir e Pedrito Maynard-Zhang. Logic-based subsumption architecture. *Artificial Intelligence*, 153(1):167–237, 2004.

[39] Franz Baader, Ralf Küsters e Ralf Molitor. Structural subsumption considered from an automata-theoretic point of view. In *Proceedings of the 1998 International Workshop on Description Logics*. Citeseer, 1998.

[40] Genichi Yasuda. Discrete event behavior-based distributed architecture design for autonomous intelligent control of mobile robots with embedded petri nets. In *Advances in Chaos Theory and Intelligent Control*, pages 805–827. Springer, 2016.

[41] Knut Akesson, Martin Fabian, Hugo Flordal e Robi Malik. Supremica-an integrated environment for verification, synthesis and simulation of discrete event systems. In *Discrete Event Systems, 2006 8th International Workshop on*, pages 384–385. IEEE, 2006.

[42] Secchi. *Control de Vehículos Autoguiados con Realimentación Sensorial*. Disertación de Maestría, Universidad Nacional de San Juan, 1999.

[43] Andre Ferreira, Flavio Garcia Pereira, Teodiano Freire Bastos-Filho, Mario Sarcinelli-Filho e Ricardo Carelli. Avoiding obstacles in mobile robot navigation: Implementing the tangential escape approach. In *2006 IEEE International Symposium on Industrial Electronics*, volume 4, pages 2732–2737. IEEE, 2006.

[44] ND Muñoz, JA Valencia e N Londono. Evaluation of navigation of an autonomous mobile robot. In *Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems*, pages 15–21. ACM, 2007.