TESE DE DOUTORADO Nº 277

PROGRAMAÇÃO DE ROBÔS POR DEMONSTRAÇÃO UTILIZANDO MODELOS NÃO LINEARES AUTORREGRESSIVOS

Rafael Francisco dos Santos

DATA DA DEFESA: 12/07/2018

Universidade Federal de Minas Gerais

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

PROGRAMAÇÃO DE ROBÔS POR DEMONSTRAÇÃO UTILIZANDO MODELOS NÃO LINEARES AUTORREGRESSIVOS

Rafael Francisco dos Santos

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Doutor em Engenharia Elétrica.

Orientador: Prof. Guilherme Augusto Silva Pereira

Coorientador: Prof. Luis Antonio Aguirre

Belo Horizonte - MG

Julho de 2018

Santos, Rafael Francisco dos.
S237p Programação de robôs por demonstração utilizando modelos não lineares autorregressivos [manuscrito] / Rafael Francisco dos Santos. – 2018.
xxiv, 106 f., enc.: il.

Orientador: Guilherme Augusto Silva Pereira.
Coorientador: Luis Antonio Aguirre.

Tese (doutorado) Universidade Federal de Minas Gerais, Escola de Engenharia.

Apêndice: f.105-106.
Bibliografia: f. 95-104.

1. Engenharia elétrica - Teses. 2. Robótica - Teses. 3.Programação por demonstração - Teses. I. Santos, Rafael Francisco dos. II. Pereira, Guilherme Augusto Silva. III.Aguirre, Luis Antonio. IV Universidade Federal de Minas Gerais. Escola de Engenharia. V. Título.

CDU: 621.3(043)

"Programação de Robôs por Demonstração Utilizando Modelos Não Lineares Autorregressivos"

Rafael Francisco dos Santos

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Doutor em Engenharia Elétrica.

Aprovada em 12 de julho de 2018.

Por:

Prof. Dr. Guilherme Augusto Silva Pereira
DEE (UFMG) - Orientador

Prof. Dr. Luis Antonio Aguirre
DELT (UFMG) - Coorientador

Leonardo A. B. Tôrres

Prof. Dr. Leonardo Antônio Borges Tôrres
DELT (UFMG)

Prof. Dr. Eduardo Mazoni Andrade Marçal Mendes
DELT (UFMG)

Prof. Dr. Fernando Cesar Lizarralde
COPPE (UFRJ)

André Luis Marq Marcato

Prof. Dr. André Luís Marques Marcato
Departamento de Energia (UFJF)

Dedico esta tese à minha família.

Resumo

Robôs autônomos são máquinas que operam sem intervenção humana, de forma totalmente automatizada, utilizando sensores para perceber o ambiente e tomar decisões. Devido à quantidade e à complexidade dos dados gerados pelos robôs atuais e à dificuldade em codificar movimentos complexos, torna-se necessária a criação de novas técnicas para a programação de políticas de controle satisfatórias. Uma das técnicas que tem mostrado ser promissora é a *programação por demonstração*, onde a ideia geral é tentar extrair uma lei de controle adequada de demonstrações de tarefas realizadas. A vantagem da programação por demonstração é a codificação direta de uma tarefa a partir de um conjunto de demonstrações, o que permite a programação do robô por pessoas "leigas". Este trabalho propõe uma metodologia para aprender movimentos de alcance de robôs a partir de um conjunto de demonstrações com o uso de modelos não lineares autorregressivos (NAR) polinomiais. Os movimentos de alcance são modelados como soluções para sistemas dinâmicos não lineares autônomos de tempo discreto, de modo que os movimentos iniciados próximos aos dados das demonstrações seguem as trajetórias treinadas e sempre alcançam e param no alvo. Como os modelos NAR, obtidos usando técnicas padrões de identificação de sistemas, nem sempre modelam adequadamente os movimentos de alcance, neste trabalho é apresentado um método que usa um estimador de mínimos quadrados com restrições para impor a localização de pontos fixos no modelo. Com a impo-

sição de novos pontos fixos é possível alterar a localização dos pontos fixos originais do modelo, permitindo assim o aprendizado de movimentos de alcance estáveis. O método foi comparado com o estado da arte utilizando uma biblioteca de movimentos de caligrafia. Os resultados mostram que o método proposto é uma boa alternativa para métodos propostos anteriormente quando é necessária uma boa precisão nas trajetórias aprendidas e, ao mesmo tempo, uma baixa demanda do controlador de baixo nível. O método também foi avaliado em um robô móvel real e em um manipulador industrial com sucesso.

Abstract

Autonomous robots are machines that operate without human intervention, fully automatic, using sensors to perceive the environment and make decisions employing that information. Due to the amount and complexity of the data generated by the robots and the difficulty in encoding complex movements, it is necessary to create new techniques to satisfactorily program control policies. One technique that has proven to be promising is *programming by demonstration*, where the general idea is to extract an adequate control law from demonstrations of the tasks to be performed. The main advantage of programming by demonstration is the direct encoding, from a set of demonstrations, of a given task, what allows robot to be programed by non-specialist people. This work proposes a methodology for learning robot reaching motions from a set of demonstrations using Nonlinear AutoRegressive (NAR) polynomial models. Reaching motions are modeled as solutions to autonomous discrete-time nonlinear dynamical systems so that the movements started near the data of the demonstrations follow the trained trajectories and always reach and stop at the target. Since NAR models obtained using standard system identification techniques do not always adequately model the reaching motions, in this work is presented a method that uses a least-squares estimator with constraints to impose the location of fixed points in the model. With the imposition of new fixed points it is possible to change the location of the original fixed points of the model, thus

allowing the learning of stable reaching motions. The method was compared with the state-of-the-art using a calligraphy movement library. The results show that the proposed method is a good alternative in the area when it is needed a good precision in the trajectories learned, and, at the same time, a low demand of the low level controller. The method was also evaluated in an actual mobile robot and an industrial manipulator with success.

Agradecimentos

Você não sabe o quanto eu caminhei
Pra chegar até aqui
Percorri milhas e milhas antes de dormir
Eu nem cochilei
Os mais belos montes escalei
Nas noites escuras de frio chorei, ei, ei, ei
Ei, ei, ei, ei, ei, ei, ei
(Cidade Negra, A estrada)

O caminho foi longo, com altos e baixos, mas "cheguei até aqui".

Agradeço a Deus, por mais uma etapa vencida, pela minha família e pela vida.

À minha esposa e companheira Lílian, por dividir toda esta jornada, por sempre me apoiar e incentivar. Às nossas filhas Rafaela e Bianca por serem a alegria das nossas vidas.

Aos meus irmãos Rodrigo e André, à minha irmã Laura, minhas tias Ângela e Salete, meu sogro Salim, minha sogra Zony, meus familiares e amigos por me apoiarem, vibrarem e por ajudarem de todas as formas.

Aos meus pais, onde quer que estejam, por me tornarem o que "sou" e por terem deixado a maior herança que poderiam deixar, a minha "educação".

Aos meus orientadores prof. Guilherme Pereira e prof. Luis Aguirre pela orientação, incentivo, paciência e pelos "puxões de orelha".

Aos colegas e amigos da UNIFEI por me apoiarem durante o doutorado. À UNIFEI pelo apoio e por permitir o meu afastamento.

Aos amigos do Laboratório de Sistemas de Computação e Robótica (CORO) pelo companheirismo, conversas e discussões.

Aos professores das bancas de qualificação e defesa, pelas sugestões e críticas que contribuíram para um trabalho melhor.

Aos professores e funcionários do Programa de Pós-Graduação em Engenharia Elétrica (PPGEE) da UFMG.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo suporte financeiro.

Sumário

Lista de Figuras xvii

Lista de Tabelas xxi

Lista de Siglas xxiii

1 Introdução 1
- 1.1 Motivação . . . 3
- 1.2 Objetivos . . . 3
- 1.3 Contribuições . . . 4
- 1.4 Organização do texto . . . 5

2 Estado da arte 7
- 2.1 Planejamento de movimento de robôs . . . 7
- 2.2 Programação por demonstração . . . 10
  - 2.2.1 Codificação de trajetórias baseadas em sistemas dinâmicos . . . 14
- 2.3 Comentários . . . 25

3 Ferramentas e técnicas adotadas 27
- 3.1 Sistemas dinâmicos . . . 27
  - 3.1.1 Espaço de estados . . . 28
  - 3.1.2 Estabilidade de pontos fixos . . . 29
- 3.2 Identificação de sistemas . . . 31
  - 3.2.1 Modelos NARMAX . . . 32
  - 3.2.2 Identificação de modelos lineares nos parâmetros . . . 33
  - 3.2.3 Seleção de Estrutura . . . 35
  - 3.2.4 Modelo MIMO polinomial . . . 38

4 Aprendizado de movimentos de alcance 41
- 4.1 Definição do problema . . . 41
- 4.2 Aprendizado utilizando modelos NAR . . . 43

4.2.1 Alocação de pontos fixos 45
4.3 Implementação Numérica 49
4.3.1 Obtendo um atrator no alvo 52
4.3.2 Analisando a bacia de atração do alvo 54
4.3.3 Melhorando um modelo RM 58
4.3.4 Procedimento completo 59

5 Experimentos e resultados **61**
5.1 Biblioteca LASA Handwriting Dataset 61
5.1.1 Comparação com trabalhos anteriores 67
5.1.2 Outras análises 75
5.2 Aplicação a um robô móvel 81
5.3 Aplicação a um manipulador 84
5.4 Discussão 87

6 Considerações finais **91**
6.1 Propostas de continuidade 92

Referências Bibliográficas **95**

A Estimador estendido de mínimos quadrados **105**

Lista de Figuras

2.1 Comparação entre funções potenciais e abordagens baseadas em sistemas dinâmicos para um ambiente sem obstáculos. O alvo dos exemplos é representado pelo asterisco preto. 9

2.2 Diagrama de blocos de uma arquitetura típica utilizada na PPD com codificação de trajetórias baseadas em sistemas dinâmicos. 13

4.1 Modelo bidimensional identificado utilizando o estimador MQ (a), e o estimador MQR (b). Em M_2 a *localização* de S_1 foi imposta por meio de uma estimação restrita, isso é $S_1 = \bar{\boldsymbol{y}}^* = [y_1^*, y_2^*]^{\mathrm{T}}$ é a nova posição de S_1, e a origem continua sendo um ponto fixo – porque o termo constante foi removido do conjunto de regressores candidatos. O tipo de estabilidade de S_1 e A não modifica, veja Conjectura 1. 47

4.2 Possível modelo bidimensional com a bacia de atração estimada na cor verde. A sela S_2 em (b) foi imposta à esquerda de sua contraparte em (a) a fim de aumentar a bacia de atração e incluir todos os dados de demonstração. M_3 é um modelo reaching motions. 48

4.3 Ilustra a varredura da área do erro entre uma demonstração e uma trajetória estimada. 53

4.4 Funcionamento do Algoritmo 2. As células coloridas de azul pertencem ao conjunto $Visited$ e as células de cor verde pertencem aos conjuntos $Visited$ e $Basin$. As setas pontilhadas em azul representam as células C que foram vinculadas as células C_{next}. 57

5.1 Três modelos iniciais obtidos com $\ell = 3$ utilizando o estimador MQ. Execução do método NAR-RM. As figuras mostram o retrato da fase dos modelos identificados em azul, os pontos fixos em amarelo (sela), verde (atrator) e vermelho (repulsor) e a bacia de atração em verde. As trajetórias das demonstrações são mostradas como trajetórias vermelhas. Todos os eixos são mostrados em milímetros. 64
5.2 Execução do método NAR-RM. As figuras mostram o retrato da fase dos modelos identificados em azul, os pontos fixos em amarelo (sela), verde (atrator) e vermelho (repulsor) e a bacia de atração em verde. As trajetórias das demonstrações são mostradas como trajetórias vermelhas e os movimentos executados em preto. Todos os eixos são mostrados em milímetros. 66
5.3 Os 30 modelos com os melhores valores de SEA e que tem pelo menos 10 mm de distância-α obtidos usando o NAR-RM. No lado esquerdo, a bacia de atração do alvo, os pontos fixos e as demonstrações são apresentadas. Do lado direito estão as reproduções obtidas pelos modelos com os mesmos pontos iniciais das demonstrações. 68
5.4 Reprodução de 4 modelos da Figura 5.3 iniciando de pontos diferentes dos pontos iniciais das demonstrações. As trajetórias das demonstrações são mostradas em vermelho e os movimentos reproduzidos em preto. Os eixos são mostrados em milímetros. 69
5.5 Quatro diferentes cenários de referência usados no benchmark. A trajetória vermelha (linha contínua) é o movimento demonstrado e a trajetória preta (linha tracejada) é o movimento perturbado gerado na simulação. A seta azul indica a direção e a amplitude da perturbação aplicada. 70
5.6 O desempenho do NAR-RM com os outros métodos para métrica trajectory-position-error do nível cinemático. Os resultados da métrica para os quatro benchmarks foram compilados e são mostrados juntos. 71
5.7 O desempenho do NAR-RM com os outros métodos para métricas de nível cinemático: trajectory-speed-errors, R2-speed, normalizedFinalTime. Os resultados de cada métrica para os quatro benchmarks foram compilados e são mostrados juntos. 72
5.8 O desempenho do NAR-RM com os outros métodos para métricas de nível cinemático & geométrico: R2, PL-R2 and mean-jerk. Os resultados de cada métrica para os quatro benchmarks foram compilados e são mostrados juntos. 73

5.9 Esquema da arquitetura utilizada nos experimentos com o robô móvel. 82
5.10 Robô móvel. 83
5.11 Retrato de fase do modelo e as reproduções realizadas pelo robô móvel. 84
5.12 Manipulador industrial. 85
5.13 Esquema da arquitetura utilizada nos experimentos com o manipulador. 85
5.14 Primeiro experimento usando o manipulador industrial. 86
5.15 Segundo experimento usando o manipulador industrial. 87
5.16 (a) Imagem contendo a configuração do segundo experimento e (b) o resultado das trajetórias inicializando o robô em partes distintas do espaço de trabalho. 87

Lista de Tabelas

2.1 Comparação dos métodos apresentados nesta seção. 24

3.1 Classificação de pontos fixos hiperbólicos para sistemas de n dimensões. 31

5.1 Porcentagem de pontos fixos que não mudam de tipo quando novos pontos são impostos usando a distância d. 62
5.2 Resultados obtidos utilizando o estimador MQ irrestrito. . . . 64
5.3 Numero de modelos (dos 41 modelos RM) com melhor desempenho após aplicar o método NAR-RM. 65
5.4 Comparação entre os modelos NAR-RM obtidos. 67
5.5 Diferentes métricas usadas pelo *benchmark* (Lemme et al., 2015). 70
5.6 Resultados das diferentes métricas usadas pelo benchmark (Lemme et al., 2015). A porcentagem em que cada valor é pior em relação ao melhor valor encontrado é representada na tabela pela notação w%. Os melhores valores são mostrados em negrito. . 74
5.7 Resultado da mediana do tempo gasto em minutos (t) e a mediana do número de modelos avaliados para os 30 padrões com as trajetórias sendo representadas com 21, 101, 201 e 1001 pontos. 76
5.8 Resultados das diferentes métricas usadas pelo benchmark (Lemme et al., 2015). A porcentagem em que cada valor é pior em relação ao melhor valor encontrado é representada na tabela pela notação w%. Os melhores valores são mostrados em negrito. . 77
5.9 Resultados de 3 execuções do método NAR-RM2 e uma do NAR-RM. São apresentados as medianas dos valores de SEA, distância-α, do tempo gasto em minutos (t) e do números de modelos avaliados para os 30 padrões de movimentos para os valores de $\ell = 3, 4, 5$ 80

Lista de Siglas

AIC	critério de informação de Akaike
ARMAX	modelo linear autorregressivo com média móvel e entradas exógenas
BM	fusão binária
C-GMR	Contração GMR
CLF-DM	movimentos dinâmicos baseados em funções de controle de Lyapunov
DMP	movimentos primitivos dinâmicos
EMQ	estendido de mínimos quadrados
ERR	taxa de redução de erro
FDM-DS	correspondência difeomórfica rápida para sistemas dinâmicos
GMM	modelo de mistura gaussiana
GMR	regressão de mistura gaussiana
GPR	processo de regressão gaussiana
LWPR	regressão de projeção localmente ponderada
MQ	mínimos quadrados
MQR	mínimos quadrados com restrições
NAR	modelo não linear autorregressivo
NARMAX	modelo não linear autorregressivo com média móvel e entradas exógenas

NAR-RM	movimento de alcance usando modelo não linear autorregressivo
NAR-RM2	movimento de alcance usando modelo não linear autorregressivo com fila de prioridade
NILC	candidata de Lyapunov moldada neuralmente
NIVF	campo vetorial estável moldado neuralmente
MIMO	multientradas, multisaídas
PPD	programação por demonstração
RNA	redes neurais artificiais
RM	movimento de alcance
SEA	varredura da área do erro
SEDS	estimador estável de sistemas dinâmicos
WSAQF	soma ponderada de funções quadráticas assimétricas

Capítulo 1

Introdução

A robótica tem sido utilizada na indústria (Hägele et al., 2008), para guiar carros de forma autônoma (Dias et al., 2015), para cuidar de pessoas (Broadbent et al., 2009), na execução de tarefas domésticas (Prassler et al., 2016) e em outras aplicações.

Robôs autônomos são máquinas que operam sem intervenção humana, de forma totalmente automatizada. Nesses robôs, uma política de controle utiliza informações de sensores para coordenar suas ações no mundo, mas a execução desta política em ambientes reais não é fácil devido a ruídos nos sensores, mapeamentos aproximados e incertezas na execução das tarefas. Além disso, a reprodução de movimentos complexos, como os inspirados em movimentos humanos, são de difícil codificação. Devido à quantidade e à complexidade dos dados gerados pelos robôs atuais e à necessidade de reproduzir movimentos mais complexos, novas técnicas vem sendo exigidas para programar políticas de controle satisfatórias.

Nas últimas décadas, os pesquisadores têm aplicado seus esforços em encontrar novas formas de ensinar tarefas aos robôs utilizando linguagem natural (Tellex et al., 2011), usando gestos (Yang et al., 2007) e usando outras tecnologias como em (Hwang et al., 2003), onde a trajetória de um robô móvel é ensinada movimentando o dedo em uma tela *touchscreen*.

Uma das técnicas que tem se mostrado promissora é a *programação por demonstração* (PPD) (Argall et al., 2009; Billard et al., 2008). Na PPD, as tarefas são aprendidas por meio de demonstrações, onde os componentes essenciais das tarefas são extraídos e codificados em uma lei de controle. Uma das vantagens da PPD é a codificação direta a partir de um conjunto de demonstrações, onde um usuário, sem conhecimento específico na área de robótica, pode realizar as demonstrações das tarefas a serem aprendidas.

Uma das subclasses de movimentos de robô que vêm sendo tratadas nas tarefas utilizando a PPD é a dos *movimentos de alcance* (do inglês, *reaching motions*). Basicamente, o movimento de alcance de robô consiste na execução de uma trajetória específica para alcançar e parar em um determinado alvo. A modelagem de movimentos de alcance fornece componentes básicos chamados movimentos primitivos (Schaal, 1999), que podem ser combinados para formar movimentos mais complexos. Movimentos de alcance codificados utilizando sistemas dinâmicos vem tendo grandes destaques na PPD, pois os sistemas dinâmicos podem ser robustos a perturbações e a mudanças no ambiente (Billard et al., 2008). Uma característica importante para se obter uma lei de controle útil para os movimentos de alcance, utilizando sistemas dinâmicos, é a garantia de estabilidade assintótica.

A PPD já foi utilizada para controlar manipuladores robóticos (Ijspeert et al., 2003; Khansari-Zadeh e Billard, 2011; Kim et al., 2014; Khansari-Zadeh e Billard, 2014), robôs humanoides (Nakanishi et al., 2004; Ott et al., 2008; Lemme et al., 2013), robôs móveis (Hayes e Demiris, 1994; Demiris e Hayes, 1996; Perk e Slotine, 2006; Narayanan et al., 2013), multi-robôs (Martins e Demiris, 2010), na robótica assistiva (Soh e Demiris, 2015) e na iteração humano-robô (Lawitzky et al., 2012).

1.1 Motivação

O uso intensivo de tecnologia computacional já faz parte do dia a dia das pessoas. O uso de *smartphones*, por exemplo, permite utilizar aplicativos que atendem especificamente às necessidades individuais de cada usuário. Apesar da robótica já estar presente em ambientes domésticos, os robôs utilizados geralmente são programados para realizar tarefas específicas, como: cortar a grama, limpar piscinas e limpar a casa. Como a sociedade está se acostumando a utilizar a tecnologia de acordo com as suas necessidades, o uso de robôs que possam ser reprogramados por um usuário, que não seja um especialista, se tornará um dos requisitos para a inserção da robótica no dia a dia das pessoas. A programação por demonstração tem mostrado ser uma ferramenta que pode permitir este avanço.

A aplicação da PPD na robótica tem apresentado bons resultados na modelagem de tarefas complexas usando robôs humanoides, manipuladores e robôs móveis. A maioria das técnicas aplicadas na literatura utilizam algoritmos iterativos que vão ajustando os resultados de acordo com uma função objetivo. Por isso, a relação entre obter uma solução adequada e o tempo de convergência dos algoritmos deve ser levada em consideração (Bautista-Ballester et al., 2014). Desta forma, desenvolver e testar novas técnicas aplicadas na PPD são importantes e representam uma contribuição para a área, pois, mesmo com a evolução dos *hardwares*, a quantidade de dados e as técnicas empregadas podem exigir um grande poder de computação.

1.2 Objetivos

Essa tese tem como objetivo principal propor uma metodologia para aprender movimentos de alcance de robôs a partir de um conjunto de demonstra-

ções com o uso de modelos não lineares autorregressivos (NAR) polinomiais. Como os modelos NAR, obtidos usando técnicas padrões de identificação de sistemas, nem sempre modelam adequadamente os movimentos de alcance, neste trabalho é apresentada uma metodologia que usa um estimador de mínimos quadrados com restrições para impor a localização de pontos fixos no modelo. Com a imposição de novos pontos fixos é possível alterar a localização dos pontos fixos originais do modelo, permitindo assim o aprendizado de movimentos de alcance estáveis.

Como objetivos específicos da metodologia, pode-se citar:

- Impor pontos fixos, utilizando restrições, durante a fase de estimação de parâmetros para alcançar alguma propriedade desejada do modelo;
- Estimar e avaliar numericamente a bacia de atração do alvo;
- Obter modelos com boa precisão na reprodução das demonstrações;
- Obter modelos robustos a perturbações temporais, causadas por possíveis atrasos de comunicação e pela dinâmica dos controladores de baixo nível; e
- Obter modelos robustos a perturbações espaciais, causadas por possíveis erros de atuação ou sensoriamento que possam introduzir pequenos desvios na execução das trajetórias.

1.3 Contribuições

A principal contribuição desta tese é a ideia inovadora de "modificar" a localização de alguns pontos fixos de um modelo com o objetivo de obter ou melhorar modelos de movimentos de alcance. A "modificação" é obtida através da reestimação dos parâmetros do modelo original com uma restrição

que impõe um ponto fixo próximo a um ponto fixo original do modelo. Uma outra contribuição é o procedimento denominado NAR-RM que utiliza a imposição de pontos fixos para obter modelos capazes de codificar movimentos de alcance com precisão e robustez a perturbações espaciais e temporais. Também foi proposta uma métrica chamada distância-α que é utilizada para estimar a qualidade da bacia de atração do alvo.

O método NAR-RM foi comparado com o estado da arte utilizando uma biblioteca de movimentos de caligrafia. Os resultados mostram que o NAR-RM é uma boa alternativa para os métodos já propostos na área quando é necessária uma boa precisão nas trajetórias aprendidas e, ao mesmo tempo, uma baixa demanda do controlador de baixo nível. O método também foi validado em um robô móvel e em um manipulador industrial.

Os resultados desta tese foram descritos no artigo *Learning robot reaching motions by demonstration using nonlinear autoregressive models* publicado na revista *Robotics and Autonomous Systems* em 2018 (Santos et al., 2018).

1.4 Organização do texto

O Capítulo 2 contextualiza a área de aplicação deste trabalho e o estado da arte envolvendo trabalhos na área de programação por demonstração de robôs. As ferramentas, os conceitos e técnicas utilizadas na tese são discutidas no Capítulo 3. A metodologia proposta e o procedimento NAR-RM são apresentadas no Capítulo 4. No Capítulo 5 são descritos os experimentos realizados e a análise dos resultados. Finalmente, as considerações finais sobre a tese e os trabalhos futuros são abordados no Capítulo 6.

Capítulo 2

Estado da arte

Este capítulo apresenta o estado da arte relacionado ao trabalho desenvolvido nesta tese. Nele é contextualizada a área de aplicação e os trabalhos relacionados à metodologia proposta. Na Seção 2.1 é apresentada previamente a área de planejamento de movimentos de robôs com o objetivo de identificar o tipo de planejamento abordado neste trabalho. Em seguida, na Seção 2.2, é apresentado o método conhecido como programação por demonstração, sua metodologia e os trabalhos relacionados a esta tese. Por fim, na Seção 2.3 é definida a área de atuação deste trabalho.

2.1 Planejamento de movimento de robôs

O problema mais básico de planejamento de movimento de robôs é guiar um robô de uma configuração inicial q_i para uma configuração final q_f sem colidir com obstáculos. Várias metodologias têm sido utilizadas para resolver este problema. Uma boa referência, que mostra de forma resumida algumas destas técnicas, pode ser encontrada em (Pereira e Chaimowicz, 2007) ou, para uma referência mais completa (Choset, 2005; Thrun et al., 2005). Nesta seção serão abordadas somente as técnicas de planejamento de movimento que mais se aproximam da metodologia proposta.

A área de controle tem mostrado em inúmeras aplicações reais que um

sinal de realimentação é importante para corrigir erros de modelagem e para se adaptar a situações não previstas. O sucesso destas aplicações inspiraram a utilização da realimentação no contexto de planejamento de movimentos (LaValle, 2006).

A técnicas de planejamento de movimentos com realimentação tem a vantagem de unificar as fases de planejamento e de controle do robô. Um dos primeiros trabalhos a utilizar planejamento de movimentos com realimentação foi (Khatib, 1986). A metodologia de Khatib (1986) é baseada na composição de uma função de potencial global por meio da soma de funções de potencial atrativas ao alvo e de forças repulsivas aos obstáculos, conhecida como método de campos potenciais. A entrada de controle do robô é obtida utilizando o gradiente da função. Mesmo sendo elegante e eficiente, a metodologia sofre com mínimos locais, que são pontos onde o gradiente se anula e, portanto, a velocidade do robô também é nula, o que pode impedir a convergência para o alvo.

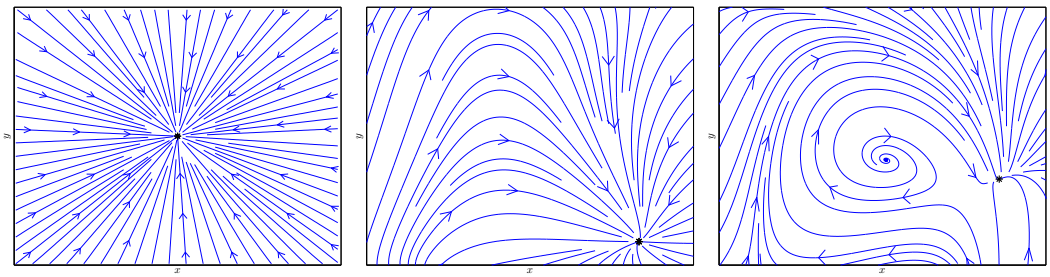
Posteriormente, o problema de mínimos locais foi resolvido por outros autores que propuseram técnicas para o cálculo de funções de potencial sem mínimo local, conhecidas como *funções de navegação* (Rimon e Koditschek, 1992). Diferentes tipos de funções de navegação foram sugeridas, como o uso de funções harmônicas (Connolly et al., 1990), campos eletrostáticos (Valavanis et al., 2000; Pimenta et al., 2006) e aproximações utilizando técnicas discretas de planejamento que utilizam buscas em grafos, como o algoritmo de Dijkstra's (Dijkstra, 1959) e o algoritmo A* (Hart et al., 1968).

Outro tipo de planejamento de movimentos com realimentação é baseado nos *movimentos primitivos* (Ijspeert et al., 2002; Billard et al., 2008), onde cada movimento codifica um comportamento[1] que é representado por um

[1]O comportamento são ações que um robô pode executar como: virar a direita; movi-

grupo de equações diferenciais autônomas ou não autônomas. Geralmente, as técnicas baseadas em movimentos primitivos são referidas como abordagens *baseadas em sistemas dinâmicos* por representarem diretamente o movimento do robô utilizando equações diferencias (Khansari-Zadeh, 2012).

Diferente das funções de potencial, as abordagens baseadas em sistemas dinâmicos permitem modelar a forma da trajetória para se alcançar o alvo. Na Figura 2.1 pode-se ver que a função de potencial só fornece uma forma de atingir o alvo (Figura 2.1(a)), enquanto as abordagens baseadas em sistemas dinâmicos permitem modelar a forma como o alvo será alcançado (Figuras 2.1(b) e 2.1(c)).



(a) Campo Potencial (b) Sistema Dinâmico 1 (c) Sistema Dinâmico 2

Figura 2.1: Comparação entre funções potenciais e abordagens baseadas em sistemas dinâmicos para um ambiente sem obstáculos. O alvo dos exemplos é representado pelo asterisco preto.

Quando se utiliza sistemas dinâmicos para codificar trajetórias, uma característica importante, para que se tenha uma política de controle útil, é a estabilidade assintótica local ou global ao alvo. Esta característica garante que toda trajetória, dentro da região com estabilidade assintótica, especifica a correta direção para se alcançar o alvo, dado a posição atual do robô. Desta forma, caso o robô sofra alguma perturbação espacial causada por algum erro de atuação ou sensoriamento que introduza pequenos desvios na execução de sua trajetória, ele ainda estará apto a alcançar o alvo se os desvios não o

mentar fazendo um trajetória em S e outros.

retirarem da região de estabilidade. Sistemas com estabilidade assintótica global definem uma função que representa um mapa de navegação global onde todas as trajetórias do espaço convergem para o alvo. Outra característica interessante, dependendo da aplicação, é que o sistema seja capaz de lidar com perturbações temporais causadas por atrasos de comunicação e pela dinâmica dos controladores de baixo nível. Uma alternativa para que o sistema seja robusto a estas perturbações temporais é que ele seja autônomo, ou seja, independente do tempo.

As abordagens baseadas em sistemas dinâmicos tem sido utilizadas para modelar movimentos de alcance (Ijspeert et al., 2002; Pastor et al., 2009; Ude et al., 2010; Khansari-Zadeh e Billard, 2011; Neumann et al., 2013; Kronander et al., 2015; Perrin e Schlehuber-Caissier, 2016), movimentos rítmicos[2] (Ijspeert et al., 2003, 2013; Schaal et al., 2005), movimentos de bater[3] (Calinon et al., 2010; Kronander et al., 2011), e outros.

Um método que tem sido utilizado para modelar movimentos primitivos com eficiência é a programação por demonstração (PPD), que será discutido na próxima seção.

2.2 Programação por demonstração

A Programação por Demonstração (do inglês, *Programming by Demonstration*), também conhecida como Aprendizado por Imitação (do inglês, *Learning by Imitation*) ou Aprendizado por Demonstração (do inglês, *Learning by Demonstration*), surgiu há quase 40 anos e tem apresentado resultados

[2]Movimentos rítmicos são movimentos periódicos, ou seja, que se repetem. Os movimentos rítmicos podem ser discretos ou contínuos.

[3]Movimentos de bater são movimentos que o robô deve executar para alcançar um ponto no espaço com uma certa configuração e uma certa velocidade. Estes movimentos são utilizados para jogar golf, jogar tênis de mesa e outras aplicações.

relevantes nas duas últimas décadas (Billard et al., 2008; Argall et al., 2009). Alguns aspectos que justificam a utilização da PPD para desenvolver uma interface flexível para ensinar robôs são:

- A PPD é um mecanismo para reduzir a complexidade do espaço de busca para o aprendizado. Ao observar bons e maus exemplos pode-se remover os maus exemplos do espaço de busca ou iniciar a busca por bons exemplos (ótimos locais).

- A PPD oferece um meio implícito de treinamento de um robô, onde as demonstrações de uma tarefa são codificadas diretamente. Como as demonstrações podem ser obtidas, por exemplo, através do movimento de uma pessoa na frente do robô ou o robô sendo guiado manualmente, isto torna a relação do ser humano com o robô algo natural, possibilitando a programação por pessoas "leigas".

Para desenvolver uma aplicação utilizando PPD, devem ser respondidas duas perguntas importantes: Como aprender uma habilidade (o que imitar) e como codificar uma habilidade (como imitar). Para aprender uma habilidade, é necessário definir como realizar a aquisição dos dados a serem imitados e qual métrica utilizar para quantitativamente avaliar a fidelidade de reprodução do robô. A aquisição de dados para o robô pode ser feita por meio da observação (Mühlig et al., 2009; Nehmzow et al., 2007), onde deve ser resolvido o problema de correspondência (Dautenhahn e Nehaniv, 2002), por meio de ambientes com realidade virtual, ou utilizando o próprio robô sendo dirigido de forma passiva por um usuário (Ijspeert et al., 2002; Khansari-Zadeh e Billard, 2011; Nehmzow et al., 2006). A métrica pode ser definida atribuindo pesos para a reprodução de cada um dos componentes da habilidade. Uma vez escolhida a forma de aquisição de dados e a métrica

a ser utilizada, é possível encontrar uma política de movimentação ótima minimizando a métrica.

Geralmente, a habilidade é aprendida utilizado um conjunto com várias demonstrações da mesma tarefa realizada em condições ligeiramente diferentes. A utilização deste conjunto permite extrair os componentes essenciais da tarefa, que geralmente são os que não se alteram durante as várias demonstrações.

Uma vez determinado como aprender uma habilidade, deve-se definir a forma como ela será codificada. Usando a codificação simbólica, a habilidade é segmentada e codificada em uma sequência de ações predefinidas descritas simbolicamente, podendo estas ações estarem organizadas de forma hierárquica. Na codificação de trajetória, as habilidades são codificadas utilizando *modelagem estatística* ou *sistemas dinâmicos*, onde é preciso definir bem quais variáveis utilizar para codificar o movimento, pois a escolha está relacionada ao problema "o que imitar".

Uma forma de representar os movimentos, na codificação de trajetória, é decompondo o mesmo em uma sequências de pontos, resultando em um movimento ponto a ponto, também conhecido como movimento discreto ou *movimentos de alcance* (do inglês, *reaching motions*). Esta abordagem geralmente é utilizada na PPD para modelar componentes básicos de um movimento que são chamados de movimentos primitivos que podem ser combinados para formar movimentos mais complexos.

Na codificação de trajetórias baseadas em modelagem estatística, pode-se citar algumas técnicas como uso de *modelo oculto de Markov* (Tso e Liu, 1996; Lee e Nakamura, 2006; Lee et al., 2011; Kulić et al., 2012) e *modelos de misturas gaussianas* (Calinon et al., 2007; Chatzis et al., 2012). As técnicas aplicadas são dependentes do tempo, o que as tornam sensíveis a

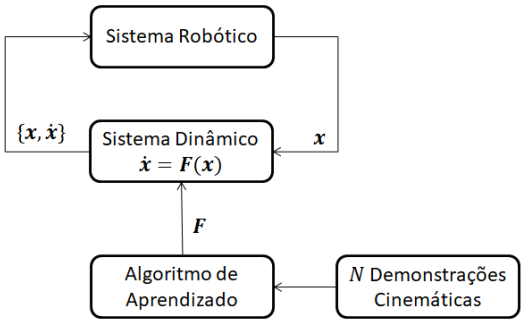


Figura 2.2: Diagrama de blocos de uma arquitetura típica utilizada na PPD com codificação de trajetórias baseadas em sistemas dinâmicos.

perturbações espaciais e temporais, sendo necessário a utilização de heurísticas para recalcular novas trajetórias no tempo, enquanto se otimiza a métrica que avalia a proximidade da nova trajetória com a trajetória desejada. As heurísticas dependem da aplicação e geralmente não são triviais, o que torna difícil a sua aplicação em espaço de estados de alta dimensão (Schaal et al., 2003).

Como discutido na Seção 2.1, a utilização de sistemas dinâmicos para codificar trajetórias apresenta a vantagem de ser um método robusto a perturbações espaciais e, caso o sistema dinâmico seja autônomo, ou seja, independente do tempo, ele também será robusto a perturbações temporais causadas por possíveis atrasos de comunicação e pela dinâmica dos controladores de baixo nível.

A Figura 2.2 ilustra a arquitetura típica utilizada na PPD com codificação de trajetórias baseadas em sistemas dinâmicos autônomos de primeira ordem, onde x representa a posição atual do robô, $\dot{x}$ as velocidades que o robô deve assumir na posição x e F as equações diferencias do sistema dinâmico. A arquitetura funciona da seguinte forma: Inicialmente os dados de N demonstrações cinemáticas são obtidas guiando-se o robô na *fase de treinamento*. Em seguida, na *fase de aprendizado*, as N demonstrações são utilizadas para

aprender um sistema dinâmico que modela movimentos compatíveis com os movimentos treinados. Por fim, na *fase de execução*, o sistema dinâmico aprendido é utilizado pelo bloco Sistema Robótico para controlar o robô. O bloco Sistema Robótico possui um sistema de controle que converte as variáveis obtidas pelo sistema dinâmico em comandos de motor e um sistema que fornece a posição do robô, após a aplicação dos comandos.

Este trabalho é restrito à PPD de robôs que utilizam a codificação de trajetórias baseadas em sistemas dinâmicos autônomos para modelar movimentos de alcance.

2.2.1 Codificação de trajetórias baseadas em sistemas dinâmicos

De acordo com Billard et al. (2008), o primeiro trabalho a enfatizar a abordagem de codificação de trajetórias utilizando sistemas dinâmicos foi Ijspeert et al. (2002, 2003) denominado *movimentos primitivos dinâmicos* (DMP, do inglês *Dynamic Movement Primitives*). O DMP combina sistemas dinâmicos não lineares de segunda ordem para codificar de forma robusta as trajetórias, e técnicas de regressão não paramétricas[4] para moldar a região do atrator de acordo com as demonstrações de trajetórias, garantindo estabilidade assintótica global ao alvo. Basicamente o DMP inicia com um sistema dinâmico composto por equações lineares, onde a condição de estabilidade é bem conhecida, e vai modulando o sistema com termos de forças não lineares até que alcance o comportamento desejado para o alvo. A estabilidade é garantida suprimindo os termos não lineares no fim do movimento, onde a mudança suave para um sistema dinâmico linear estável é controlada por uma variável

[4]Geralmente o DMP utiliza a regressão localmente ponderada (LWR, do inglês *Locally Weighted Regression*).

de fase. Como as trajetórias codificadas são de natureza cinemática, um controlador que transforma informações cinemáticas em comandos de controle é necessário.

O DMP se tornou uma técnica popular para codificar movimentos de robôs e foi utilizada em várias aplicações como caminhar (Nakanishi et al., 2004), Controle de vôo (Perk e Slotine, 2006), desvio de obstáculos (Hoffmann et al., 2009) e outras.

Apesar de popular, o DMP apresenta algumas restrições. A primeira é que o sistema dinâmico utilizado é não autônomo, ou seja, é necessário utilizar uma heurística para reiniciar a variável de fase caso aconteça uma perturbação temporal. A segunda é que cada dimensão do sistema dinâmico é codificada separadamente, não extraindo o efeito da combinação das dimensões no movimento, o que pode exigir uma heurística de sincronização caso ocorra alguma perturbação no movimento. A última restrição é que o DMP utiliza somente uma demonstração para codificar o movimento, o que reduz a sua capacidade de generalização.

Posteriormente, foram propostas outras técnicas estatísticas para modelar os sistemas dinâmicos como a *regressão de projeção localmente ponderada* (LWPR, do inglês *Locally Weighted Projection Regression*) (Schaal et al., 2002), *processo de regressão gaussiana* (GPR, do inglês *Gaussian Process Regression*) (Seeger, 2004) e *regressão de mistura gaussiana* (GMR, do inglês *Gaussian Mixture Regression*) (McLachlan e Peel, 2004). Apesar destas técnicas apresentarem algoritmos de regressão rápidos e serem projetadas para repetir trajetórias semelhantes a partir de uma única demonstração, elas nem sempre garantem a estabilidade assintótica ao alvo (Khansari-Zadeh e Billard, 2010, 2011).

Em 2010 foi proposto um algoritmo iterativo capaz de estimar sistemas

dinâmicos não lineares estáveis a partir de algumas demonstrações utilizando modelos de misturas gaussianas (GMM, do inglês *Gaussian Mixture Model*) denominado *fusão binária* (BM, do inglês *Binary Merging*) (Khansari-Zadeh e Billard, 2010). O algoritmo BM, utilizando a GMR, é capaz de estimar um sistema dinâmico autônomo, representado no espaço de estados, utilizando várias demonstrações e com estabilidade assintótica local ao alvo. A verificação da estabilidade local é feita numericamente, onde uma região local D^0, que contém o alvo e os pontos das demonstrações, é discretizada em m pontos e as condições de estabilidade, no sentido de Lyapunov, são verificadas. Assim, verifica-se se o alvo é um ponto de equilíbrio do sistema e se todos os m pontos atendem a uma função de Lyapunov pré-definida que garante a estabilidade assintótica ao alvo. Para garantir uma melhor precisão da região com estabilidade assintótica local, um conjunto de pontos S, próximos aos pontos iniciais das demonstrações, é escolhido para representar o limite da região inicial com estabilidade garantida. Os pontos contidos em S são simulados e a região limitada por suas trajetórias formam a região D que é retornada pelo BM e que possui estabilidade assintótica local ao alvo.

Comparado com o DMP, o BM possui as vantagens: de ser robusto a perturbações temporais, pois é modelado como um sistema dinâmico autônomo; de ser representado no espaço de estados, o que permite executar movimentos mais complexos do que o DMP (Ijspeert et al., 2003); de codificar várias dimensões do sistema dinâmicos em um único sistema; e de poder aprender movimentos utilizando várias demonstrações com pontos iniciais próximos. Entre as desvantagens do BM estão a ausência de estabilidade assintótica global ao alvo, limitando a sua área de atuação, e o cálculo de sua estabilidade local que é executado de forma discreta e numérica, dificultando a aplicação em sistema com várias dimensões devido ao alto custo computacional para

computar a região D.

Uma técnica chamada *estimador estável de sistemas dinâmicos* (SEDS, do inglês *Stable Estimator of Dynamical Systems*) (Khansari-Zadeh e Billard, 2011) foi publicada pelos mesmos autores do BM. Como no trabalho anterior, eles demonstraram que as técnicas de LWPR, GPR e GMR não conseguem garantir a estabilidade global do movimento. Os autores do SEDS justificaram que o motivo da não garantia de estabilidade ocorre porque as técnicas citadas não consideram, em suas formulações dos problemas de otimização associados, a restrição de tornar o sistema estável no atrator, o que pode convergir os movimentos para atratores falsos ou para regiões onde a velocidade é zero.

O SEDS é um algoritmo de aprendizado que garante a estabilidade de sistemas dinâmicos autônomos, representados no espaço de estados, modelados com GMM e utilizando a regressão GMR. A estabilidade é garantida resolvendo um problema de otimização não convexo que considera um conjunto de restrições que são obtidas utilizando uma função de Lyapunov, as quais garantem a estabilidade assintótica global ao alvo. A estabilidade assintótica global do sistema dinâmico estimado utilizando SEDS garante que todas as trajetórias convergem para o alvo. Isto assegura que o robô atingirá o alvo a partir de qualquer ponto do espaço de estados.

O SEDS apresenta algumas melhorias em relação ao BM: a primeira é a garantia de estabilidade assintótica global ao alvo e a segunda é a capacidade de aprender movimentos com pontos iniciais distantes, ou seja, é possível criar movimentos com diferentes comportamentos em regiões distintas do espaço de estados, o que não é possível utilizando BM. Apesar das melhorias, o BM apresenta maior precisão em relação ao SEDS, ao modelar alguns movimentos (Khansari-Zadeh e Billard, 2011). Isto ocorre porque a função quadrática

de Lyapunov, utilizada para definir o conjunto de restrições que são aplicadas durante o processo de otimização, limita a capacidade de representar alguns movimentos pelo SEDS (Khansari-Zadeh e Billard, 2014).

Em relação ao movimentos primitivos dinâmicos (DMP), além das vantagens já apontadas pelo BM como: a representação no espaço de estados, a robustez a perturbações temporais, a representação de várias dimensões em um único sistema e a possibilidade de utilizar várias demonstrações; pode-se destacar a capacidade de aprender movimentos com diferentes comportamentos em regiões distintas do espaço de estados, o que não é possível utilizando DMP, pois ele utiliza somente uma única demonstração.

A desvantagem do SEDS em relação ao DMP é que para robôs com muitas dimensões, como robôs humanoides, o número elevado de estados pode acarretar problemas computacionais e numéricos, o que não ocorre com o DMP, pois cada dimensão é modelada de forma independente. Além desse ponto, a fase de aprendizado do DMP é mais rápida, pois utiliza uma única demonstração.

Uma outra desvantagem na aplicação do SEDS é que a técnica depende da solução de um problema de otimização não convexo, onde não é possível garantir que a solução ótima do problema será encontrada. Além disso, o método utilizado para otimização é muito sensível aos valores iniciais da busca, o que pode fazer com que o processo de otimização convirja para um mínimo local. Os valores iniciais também devem ser valores válidos, ou seja, devem atender a todas as restrições do problema. Estes detalhes impactam na resposta final obtida pelo SEDS, que utiliza, para tentar amenizar estes problemas, uma função que a cada execução fornece valores iniciais diferentes e válidos. Os autores afirmaram que para os testes do trabalho (Khansari-Zadeh e Billard, 2011), três ou quatro execuções eram suficientes para obter

uma boa resposta.

O SEDS foi utilizado em robôs que desempenharam as tarefas de jogar mini-golf (Khansari-Zadeh et al., 2012), desviar de obstáculos (Khansari-Zadeh e Billard, 2012), pegar objetos arremessados (Kim et al., 2014), codificar comportamentos em um robô móvel (Narayanan et al., 2013), e outros.

Em 2013 os autores do DMP lançaram um artigo de revisão (Ijspeert et al., 2013), onde eles apresentaram a evolução da técnica nos últimos anos, destacando a possibilidade de codificar sistemas autônomos, de reconhecer movimentos e outras formas de utilização. Muitos trabalhos também foram inspirados no DMP como o *movimentos primitivos probabilísticos* (Neumann et al., 2014) e o *estilos adaptativos de movimentos primitivos dinâmicos* (Zhao et al., 2014) .

A metodologia proposta neste trabalho está relacionada com métodos que utilizam a codificação de trajetórias baseadas em sistemas dinâmicos autônomos, representados no espaço de estados e que possuam estabilidade assintótica global ou local ao alvo. Por isso, no restante desta seção, só serão abordados trabalhos que possuem estas especificações.

Uma técnica chamada *campo vetorial estável moldado neuralmente* (NIVF, do inglês *Neurally Imprinted stable Vector Fields*) (Lemme et al., 2013) foi proposta em 2013. Neste trabalho um sistema dinâmico autônomo, representado no espaço de estados e com garantia de estabilidade assintótica local, é obtido utilizando redes neurais artificiais aplicando um algoritmo de treinamento chamado *máquina de aprendizado extremo*[5]. A garantia da estabi-

[5]As máquinas de aprendizado extremo são redes neurais *feedforward* utilizadas para classificação ou regressão (Huang et al., 2006). Diferente dos outros algoritmos para treinamento de redes neurais, que fazem o ajuste dos parâmetros da rede por meio da apresentação iterativa dos padrões de treinamento, as máquinas de aprendizado extremo fazem uma projeção aleatória na camada escondida, onde os pesos são selecionados de forma aleatória.

lidade assintótica local é obtida restringindo o algoritmo de aprendizado da rede a atender uma função quadrática de Lyapunov. A estabilidade garantida é local, pois é necessário definir o conjunto finito de pontos para compor as restrições que irão delimitar a região contínua do espaço de estados que atenderá a função de Lyapunov.

Os resultados obtidos pelo NIVF são comparáveis ao SEDS, pois ambos utilizam uma função quadrática de Lyapunov. A diferença é que o NIVF garante somente a estabilidade assintótica local ao alvo e também necessita de um processo custoso, após a fase de aprendizado, para atestar a garantia de estabilidade (Neumann e Steil, 2015).

O dilema da estabilidade versus a precisão foi comentado em Khansari-Zadeh e Billard (2011) ao comparar os métodos SEDS e BM. Alguns resultados mostraram que o BM, que possui estabilidade local, conseguia modelar alguns movimentos com precisão bem maior que o SEDS, que possui estabilidade global, este resultado acabou inspirando novas técnicas.

Na tese de Khansari-Zadeh (2012) foi apresentado a técnica SEDS-II que posteriormente foi publicada com o nome *movimentos dinâmicos baseados em funções de controle de Lyapunov* (CLF-DM, do inglês *Control Lyapunov Function-based Dynamic Movements*) (Khansari-Zadeh e Billard, 2014). Esta técnica explora o poder das funções de controle de Lyapunov aplicadas na PPD. Além de identificar sistemas dinâmicos autônomos, representados no espaço de estados, com estabilidade assintótica global, a técnica permite ao usuário escolher qualquer método de regressão. O CLF-DM é aplicado utilizando um conjunto de demonstrações em 3 etapas: 1) uma função de Lyapunov válida é aprendida utilizando os dados das demonstrações resolvendo um problema restrito de otimização parametrizado pela *soma ponderada de funções quadráticas assimétricas* (WSAQF) ; 2) uma estimativa instável do

sistema dinâmico é obtida utilizando os dados das demonstrações e algum método de regressão; 3) a função de Lyapunov aprendida é utilizada para construir um comando de estabilização que é utilizado para garantir a estabilidade durante a fase de execução. O comando de estabilização é obtido resolvendo um problema convexo de otimização.

Os resultados apresentados pelos autores mostram que o CLF-DM apresentou melhores resultados que o SEDS, conseguindo modelar com maior precisão um número maior de movimentos. As desvantagens do CLF-DM são que a estabilidade assintótica global não é garantida na fase de aprendizado e a estabilização depende de um sinal de correção *on-line* que potencialmente interfere no sistema dinâmico, o que demanda uma seleção criteriosa de parâmetros (Neumann e Steil, 2015).

Outro trabalho que visa melhorar a relação entre estabilidade e precisão foi apresentado por Neumann et al. (2013). Neste trabalho os autores apresentaram uma técnica para aprender funções candidatas de Lyapunov mais flexíveis que as funções quadráticas a partir das demonstrações, denominada *candidata de Lyapunov moldada neuralmente* (NILC, do inglês *Neurally Imprinted Lyapunov Candidate*) e as incorporaram ao NIVF. O aprendizado foi dividido em duas etapas: 1) uma função candidata de Lyapunov flexível é aprendida a partir dos dados; 2) a função de Lyapunov aprendida é utilizada no NIVF substituindo a função quadrática de Lyapunov padrão. Os experimentos realizados mostraram que a utilização de funções candidatas de Lyapunov, aprendidas dos dados das demonstrações, permite aprender com precisão um conjunto maior de movimentos.

Diferente do CLF-DM, que utiliza as funções candidatas de Lyapunov na fase de execução, o NIVF junto com NILC utiliza a função na fase de aprendizado, resultando em um sistema dinâmico com estabilidade assintótica local

ao alvo.

No ano de 2015 um método chamado τ-SEDS foi proposto. O método utiliza transformações difeomórficas para melhorar o resultado do SEDS utilizando funções candidatas de Lyapunov (Neumann e Steil, 2015). O método funciona da seguinte forma: 1) uma função candidata de Lyapunov L é aprendida utilizando as demonstrações D, por exemplo, WSAQF ou NILC; 2) a função candidata de Lyapunov L é transformada para uma forma quadrática $\tilde{L}$ utilizando um difeomorfismo candidato τ; 3) as demonstrações D são transformadas para o novo espaço, o qual é consistente com a candidata de Lyapunov quadratica $\tilde{L}$, produzindo $\tilde{D}$; 4) o sistema dinâmico $\tilde{\mathbf{y}}$ é aprendido utilizando SEDS com $\tilde{D}$ e $\tilde{L}$; 5) a inversa do difeomorfismo τ^{-1} é usada para mapear o sistema $\tilde{\mathbf{y}}$ de volta ao espaço original obtendo $\mathbf{y}$. Neumann e Steil (2015) compararam os resultados do SEDS original, com os métodos que utilizam funções candidata de Lyapunov. Os métodos foram o CLF-DM, NIVF e τ-SEDS todos utilizando as duas funções candidatas obtidas dos dados das demonstrações WSAQF e NILC. Os resultados mostraram que a melhor resposta foi obtida utilizando τ-SEDS com a função candidata de Lyapunov aprendida utilizando a soma ponderada de funções quadráticas assimétricas (WSAQF). Os autores discutem no trabalho que mesmo obtendo o melhor resultado e conseguindo uma representação de movimentos maior que os métodos que utilizam funções candidatas de Lyapunov quadráticas, o método τ-SEDS também apresenta uma limitação que é imposta pelo difeomorfismo candidato, que requer uma função de Lyapunov candidata L que atenda a inequação $(x - x^*)^T \cdot \nabla L > 0$, onde x é um ponto no espaço original e x^* é o único ponto atrator assintoticamente estável.

Outro método que utiliza transformações difeomórficas, intitulado correspondência difeomórfica rápida para aprender globalmente sistemas dinâmicos

não lineares assintoticamente estáveis, foi proposto em (Perrin e Schlehuber-Caissier, 2016). Denominado, neste trabalho, pela sigla FDM-DS, o método aprende um difeomorfismo que mapeia um sistema dinâmico $\dot{x} = -x$ nas trajetórias das demonstrações, obtendo um sistema dinâmico não linear com garantia de estabilidade assintótica global ao alvo. O método funciona em dois passos: 1) dado um conjunto de demonstrações, o método cria uma sequência temporizada média dos pontos das trajetórias $\mathbf{Y} = (\mathsf{y}(i\Delta t))_{i\in\{0,\dots,N\}}$; 2) e em seguida é aplicado o algoritmo de correspondência, que é baseado em uma tradução localmente ponderada (do inglês, locally weighted translation), para construir um difeomorfismo Φ que mapeia $\mathbf{X} = \left(\frac{N-i}{N}\mathsf{y}(0)\right)_{i\in\{0,\dots,N\}}$ para $\mathbf{Y}$. Com a execução dos dois passos é obtido a função candidata de Lyapunov $x \mapsto \|\Phi^{-1}(x)\|$. O método foi comparado utilizando padrões de demonstrações simples em 2D com o método τ-SEDS com a função candidata de Lyapunov WSAQF. Os resultados mostraram que o método FDM-DS apresentou melhores resultados do que o método τ-SEDS, além de apresentar um algoritmo mais rápido na fase de aprendizado. Os autores também mostraram que o método é capaz de aprender com maior precisão um número maior de movimentos que o τ-SEDS. Mesmo obtendo melhores resultados, o método FDM-DS não foi avaliado em padrões de movimentos que apresentam diferentes comportamentos ao ser iniciando em regiões distintas do espaço de estados, apesar dos autores informarem que é possível estender o método para este propósito. Este tipo de padrão de movimento é aprendido pelos métodos SEDS, NIVF, CLF-DM e τ-SEDS.

Em 2017 foi publicado por Blocher et al. (2017) um método que utiliza a regressão de mistura gaussiana (GMR), com garantia de estabilidade do sistema dinâmico aprendido, utilizando a teoria de contração proposta por Lohmiller e Slotine (1998). Denominado de Contração GMR (C-GMR, do inglês

Contraction GMR), o método compartilha a mesma ideia do CLF-DM que utiliza um comando de estabilização para garantir a estabilidade do sistema dinâmico durante a fase de execução. O método funciona da seguinte forma: 1) um sistema dinâmico é estimado utilizando GMR; 2) um comando de estabilização e uma função de ativação são calculadas para o sistema dinâmico utilizando a teoria de contração; 3) durante a fase de execução o comando de estabilização é suavemente ativado e desativado usando a função de ativação. O uso da função de ativação garante uma boa reprodução dos movimentos aprendidos sem afetar a estabilidade do sistema. O método C-GMR foi comparado com o método CLF-DM e foi mostrado que ele é menos preciso. Em relação ao tempo de treinamento o C-GMR foi bem mais eficiente que o CLF-DM, o que torna o método interessante para aplicações de aprendizado incremental, onde novas demonstrações das tarefas são sempre fornecidas, o que pode ser inviável para o CLF-DM.

Todas as técnicas apresentadas até o momento foram compiladas na Tabela 2.1, onde são comparadas a estabilidade e a função de Lyapunov utilizada. A última coluna da tabela denominada "integrado" indica se a técnica tem a estabilidade garantida na fase de aprendizado ou não.

Tabela 2.1: Comparação dos métodos apresentados nesta seção.

Técnica	**Estabilidade**	**Função Lyapunov**	**Integrado**	**Ano**
DMP	Global	-	Sim	2002 - 2013
BM	Local	Específica	Sim	2010
SEDS	Global	Quadrática	Sim	2011
NIVF	Local	Quadrática	Sim	2013
CLF-DM	Global	WSAQF	Não	2012 - 2014
NIVF	Local	WSAQF, NILC	Sim	2013
τ-SEDS	Global	WSAQF, NILC	Sim	2015
FDM-DS	Global	Construída	Sim	2016
C-GMR	Global	-	Não	2017

Além dos trabalhos anteriores, um método que utiliza técnicas de identificação de sistemas usando modelos autoregressivos também foi proposto

para ensinar robôs móveis, com rodas, utilizando demonstrações (Nehmzow et al., 2006, 2007). O método codifica os sistemas dinâmicos utilizando o modelo não linear autorregressivo com média móvel e entradas exógenas (NARMAX[6]) polinomial ou o modelo linear (ARMAX). Diferentemente dos trabalhos anteriores, que modelam trajetórias de robôs, os métodos usando modelos NARMAX ou ARMAX constroem um mapeamento sensor-motor, onde as informações sensoriais das demonstrações são usadas para obter um modelo que é usado diretamente para controlar o robô. Os modelos aprendidos, que requerem uma quantidade considerável de dados e demonstrações para serem estimados, foram utilizados em tarefas reativas, como seguir uma parede, seguir um corredor, atravessar portas e no aprendizado de rotas. Vários trabalhos que utilizam o método proposto por Nehmzow et al. (2006) foram publicados explorando diferentes técnicas de treinamento, de escolha de sensores e na análise de estabilidade para a tarefa de atravessar a porta (Iglesias et al., 2008; Akanyeti et al., 2008, 2009, 2010; Nehmzow et al., 2010; Kerr et al., 2010; Gardiner et al., 2012).

2.3 Comentários

Como foi mostrado nas seções anteriores deste capítulo, é fundamental que os sistemas dinâmicos obtidos utilizando a programação por demonstração (PPD) que modelam trajetórias de robôs, possuam estabilidade assintótica global ou local ao alvo para que o sistema defina uma política de controle útil.

Sobre o uso do modelo NARMAX polinomial na PPD aplicadas em

[6]Sigla NARMAX originada do inglês *Nonlinear AutoRegressive Moving Average with eXogenous variables*

Robôs, nenhum artigo encontrado modela trajetória de robôs utilizando o espaço de estados e apresenta alguma garantia de estabilidade assintótica global ou local ao alvo. Desta forma, neste trabalho é proposto uma metodologia baseada em sistemas dinâmicos, identificados utilizando modelos não lineares autorregressivos NAR polinomiais, representados no espaço de estados e com garantia de estabilidade assintótica local ao alvo. No próximo capítulo são apresentas as ferramentas e técnicas utilizadas no desenvolvimento da metodologia.

Capítulo 3

Ferramentas e técnicas adotadas

Este capítulo apresenta algumas ferramentas e técnicas usadas ao longo do desenvolvimento deste trabalho. Na Seção 3.1 são apresentados alguns conceitos sobre sistemas dinâmicos e na Seção 3.2 é abordada a técnica de identificação de sistemas utilizando o modelo não linear autorregressivo com média móvel e entradas exógenas (NARMAX) polinomial.

3.1 Sistemas dinâmicos

Um *sistema* é um conjunto de componentes que interagem de forma definida. Diz-se que um sistema é *dinâmico* quando a saída no instante k depende de valores passados da saída e da entrada.

Os sistemas dinâmicos podem ser classificados como lineares ou não lineares. Os sistemas dinâmicos são lineares quando satisfazem o princípio da superposição[1], caso contrário, os sistemas são não lineares. Além da linearidade, os sistemas dinâmicos podem ser classificados como autônomos ou não autônomos. Os sistemas autônomos são aqueles cujas equações diferen-

[1]Princípio da superposição: Considere um sistema que ao ser excitado pela entrada $u_1(t)$ produz a saída $y_1(t)$, e quando excitada por $u_2(t)$ produz a saída $y_2(t)$. Se o sistema for excitado por $c_1u_1(t) + c_2u_2(t)$ e produzir $c_1y_1(t) + c_2y_2(t)$, sendo c_1 e c_2 constantes quaisquer, esse sistema satisfaz o princípio da superposição.

ciais ou de diferenças, que descrevem seus comportamentos, não dependem explicitamente do tempo e os não autônomos dependem do tempo.

Para representar matematicamente os sistemas dinâmicos são utilizadas equações diferenciais, no caso de sistemas contínuos, e equações de diferença, no caso de sistemas discretos[2]. Um sistema dinâmico autônomo discreto de ordem n pode ser representado por:

$$\mathbf{x}(k) = \mathbf{F}[\mathbf{x}(k-1)], \tag{3.1}$$

em que $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ e $\mathbf{F}(\cdot) = [f_1(\cdot), f_2(\cdot), \ldots, f_n(\cdot)]^T$ é um vetor n-dimensional de funções de $\mathbf{x}$.

3.1.1 Espaço de estados

O *espaço de estados*, ou *espaço de fases*, é um espaço onde um sistema é representado por n variáveis, conhecidas como variáveis de estado, e por n equações diferenciais, ou de diferenças, de primeira ordem. Os eixos do espaço de estados são formados pelas variáveis de estado e um estado $\mathbf{x}$ é representado por um ponto com coordenadas $(x_1(k), x_2(k), \ldots, x_n(k))$. O movimento do sistema é determinado exclusivamente pelo estado atual e pelas equações de evolução dos estados.

O sistema dinâmico da Equação (3.1), que está representado no espaço

[2]Normalmente se refere a sistemas dinâmicos de tempo discreto.

de fase, pode ser reescrito como:

$$\begin{aligned} x_1(k) &= f_1[x_1(k-1), x_2(k-1), \ldots, x_n(k-1)] \\ x_2(k) &= f_2[x_1(k-1), x_2(k-1), \ldots, x_n(k-1)] \\ &\vdots \quad \vdots \\ x_n(k) &= f_n[x_1(k-1), x_2(k-1), \ldots, x_n(k-1)], \end{aligned} \tag{3.2}$$

com $f_i : \mathbb{R}^n \mapsto \mathbb{R}$ com $i = 1, 2, \ldots, n$.

As variáveis independentes x_j são as variáveis de estado. A dimensão do espaço de estados é definido pelo número de variáveis de estado. O *campo vetorial*, que é o conjunto de todos os vetores que indicam as possíveis mudanças nos estados no espaço estados de um sistema dinâmico, é definido pelas funções f_i.

Um conjunto de soluções para o sistema da Equação (3.2), para $k = 0, 1, \ldots, m$, define uma sequência de pontos que representam uma *trajetória* no espaço de estados. Denomina-se *retrato de fase* o conjunto de trajetórias obtidas pela evolução temporal do sistema a partir de todas condições iniciais válidas dos estados $\mathbf{x}$.

3.1.2 Estabilidade de pontos fixos

Para o sistema da Equação (3.1) o estado estacionário pode ser obtido definindo-se $\mathbf{x}(k) = \mathbf{x}(k-1) = \bar{\mathbf{x}}$. Desta forma, as soluções de equilíbrio do sistema, ou seja, os pontos fixos do sistema, são soluções de:

$$\bar{\mathbf{x}} = \mathbf{F}[\bar{\mathbf{x}}]. \tag{3.3}$$

"O *teorema de Hartman-Grobman* para mapas garante que a estabilidade

de um ponto fixo P hiperbólico pode ser determinada analisando a versão linearizada do sistema" (Guckenheimer e Holmes, 1983).

A linearização de um sistema não linear pode ser obtida pelo cálculo de sua matriz Jacobiana. A matriz Jacobiana do sistema da Equação (3.1) é

$$\mathrm{D}F(x_1, x_2, \ldots, x_n) = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_n}{\partial x_1} & \cdots & \dfrac{\partial f_n}{\partial x_n} \end{bmatrix}. \tag{3.4}$$

Então, seja P um ponto fixo do sistema não linear da Equação (3.1) e $\mathrm{D}F$ a sua matriz Jacobiana calculada em P. Os autovalores λ_i $(i = 1, \ldots, n)$ de $\mathrm{D}F$ podem ser separados em três grupos da seguinte forma:

$$\begin{aligned} \lambda_i &\in \sigma_e \qquad \text{se} \qquad |\lambda| < 1, \\ \lambda_i &\in \sigma_i \qquad \text{se} \qquad |\lambda| > 1, \\ \lambda_i &\in \sigma_c \qquad \text{se} \qquad |\lambda| = 1, \end{aligned}$$

sendo que $\lambda_i \in \mathbb{C}$, σ_e o grupo "estável", σ_i o grupo "instável" e σ_c o grupo "central".

Um ponto fixo é *hiperbólico* se todos os autovalores λ_i pertencem aos grupos σ_e ou σ_i. Se algum autovalor pertencer ao grupo σ_c, o ponto fixo é *não hiperbólico*.

Utilizando o teorema de Hartman-Grobman pode-se classificar os pontos fixos hiperbólicos de acordo com a Tabela 3.1. Para pontos fixos não hiperbólicos a estabilidade pode ser verificada utilizando o método direto de Lyapunov ou a teoria da variedade central. Esses dois métodos não serão abordados neste texto, mas podem ser consultados em (Monteiro, 2011).

Tabela 3.1: Classificação de pontos fixos hiperbólicos para sistemas de n dimensões.

AutoValores	**Classificação**	**Estabilidade**
$\lambda_i \in \sigma_e,\ \forall i = 1, 2, 3, \ldots, n.$	Atrator	Assintoticamente Estável
$\lambda_i \in \sigma_i,\ \forall i = 1, 2, 3, \ldots, n.$	Repulsor	Instável
$\lambda_i \in \{\sigma_e \cup \sigma_i\},\ \forall i = 1, 2, 3, \ldots, n.$ tal que $\sigma_e \neq \varnothing$ e $\sigma_i \neq \varnothing$	Sela	Instável

Em relação aos pontos fixos classificados como atratores, todos possuem uma região no espaço de estados associada a eles chamada de *bacia de atração*. A bacia de atração de um ponto fixo atrator $\mathcal{A}$ é o conjunto de todos os pontos que ao serem evoluídos no tempo, segundo a Equação (3.1), para $k \to \infty$ convergem para $\mathcal{A}$. Diz-se que um ponto fixo atrator $\mathcal{A}$ possui *estabilidade assintótica global* se a sua bacia de atração abrange todo o espaço de estados, caso contrário, $\mathcal{A}$ possui *estabilidade assintótica local*.

3.2 Identificação de sistemas

A identificação de sistemas é uma "área do conhecimento que estuda e desenvolve técnicas e algoritmos para obter (identificar) modelos de sistemas dinâmicos a partir de dados gerados pelo próprio sistema" (Aguirre, 2007, p.689). Assim, pode-se dizer que a identificação de sistemas utiliza um modelo matemático para explicar de forma aproximada a relação de causa e efeito presente nos dados de um sistema, podendo o modelo representar partes do sistema ou ele completo.

A área de identificação de sistemas pode ser dividida em duas:

- *Identificação caixa preta* - onde pouco ou nenhum conhecimento prévio do sistema é necessário para descrever o seu comportamento; e

- *Identificação caixa cinza* - que utiliza alguma informação auxiliar, que não se encontra no conjunto de dados durante a identificação.

Nesta seção será apresentada a técnica de identificação caixa preta de sistemas dinâmicos utilizando como representação matemática o modelo NARMAX (do inglês, *Nonlinear AutoRegressive Moving Average with eXogenous variables*) polinomial.

3.2.1 Modelos NARMAX

Introduzidos por (Leontaritis e Billings, 1985a,b), os modelos NARMAX são capazes de representar uma ampla classe de sistemas não lineares, discretos e invariantes no tempo. Um modelo NARMAX tenta explicar o valor da saída $y(k)$ em função de valores prévios dos sinais de entrada e saída, e é representado por:

$$\begin{aligned} y(k) =& F[y(k-1),\ldots,y(k-n_y),u(k-\tau_d),u(k-\tau_d-1),\ldots,u(k-\tau_d-n_u),\ldots \\ & \ldots,e(k-1),\ldots,e(k-n_e)] + e(k), \end{aligned} \tag{3.5}$$

sendo que n_y, n_u e τ_d são os maiores atrasos em y, em u e o tempo morto, respectivamente; $e(k)$ é o ruído e n_e é o maior atraso no modelo de ruído; F é uma função não linear, que pode ser representada na forma polinomial. Outras alternativas de representação para F podem ser encontradas em (Aguirre e Letellier, 2009).

O modelo NARMAX polinomial para a Equação (3.5) é obtido expandindo-se F como um polinômio de grau ℓ, como apresentado em (Chen e Billings, 1989):

$$\begin{aligned} y(k) = \theta_0 + \sum_{i_1=1}^{n} \theta_{i_1} x_{i_1}(k) + \sum_{i_1=1}^{n} \sum_{i_2=i_1}^{n} \theta_{i_1 i_2} x_{i_1}(k) x_{i_2}(k) + \ldots + \\ \sum_{i_1=1}^{n} \ldots \sum_{i_l=i_{l-1}}^{n} \theta_{i_1 \ldots i_l} x_{i_1}(k) \ldots x_{i_l}(k) + e(k), \end{aligned} \tag{3.6}$$

sendo θ_i os parâmetros que devem ser estimados, $n = n_y + n_u + n_e$ e $x_1(k) = y(k-1), x_2(k) = y(k-2), \ldots, x_{n_y+1}(k) = u(k-1), \ldots, x_{n_y+n_u+1}(k) = e(k-1), \ldots, x_n(k) = e(k-n_e)$.

Neste trabalho é utilizada a representação polinomial para F. A vantagem de se utilizar os modelos polinomiais para representar um sistema dinâmico é que os modelos são lineares ou pseudolineares nos parâmetros, o que permite utilizar algoritmos de estimação de parâmetros da família dos mínimos quadrados (MQ).

3.2.2 Identificação de modelos lineares nos parâmetros

Assumindo um modelo NARX, ou seja, o modelo NARMAX sem os regressores de ruído, pode-se reescrever o modelo da Equação (3.6), na forma de erro de predição

$$y(k) = \psi^T(k-1)\hat{\theta} + \xi(k), \tag{3.7}$$

sendo que $\xi(k)$ é o resíduo no instante k e $\psi^T(k-1)$ é o vetor de regressores que pode incluir medições até o instante $(k-1)$, enfatizando o caráter preditivo do modelo.

Ao usar o modelo da Equação (3.7) ao longo de uma janela de dados de tamanho N, é obtida a equação matricial:

$$\mathbf{y} = \Psi\hat{\theta} + \boldsymbol{\xi}, \tag{3.8}$$

sendo que Ψ é a matriz de regressores das N observações, $\hat{\theta}$ é o vetor de parâmetros estimados e $\boldsymbol{\xi}$ é o vetor de erros cometidos pelo modelo ao se tentar explicar $\mathbf{y}$ por $\Psi\hat{\theta}$.

O vetor de parâmetros θ pode ser estimado minimizando a soma dos

quadrados dos erros dado pela seguinte função de custo:

$$J_{\text{MQ}} = \sum_{i=1}^{N} \|\boldsymbol{\xi}(i)\|^2 = \boldsymbol{\xi}^{\,T}\boldsymbol{\xi}. \tag{3.9}$$

A forma fechada, conhecida como método clássico de mínimos quadrados, é dada por:

$$\hat{\theta}_{\text{MQ}} = [\Psi^T\Psi]^{-1}\Psi^T\mathbf{y}. \tag{3.10}$$

O estimador MQ está sujeito a polarização induzida por ruído presente nos dados. Isto ocorre quando o erro que aparece na Equação (3.8) é auto-correlacionado e o modelo possui regressores da saída. Uma alternativa para evitar a polarização é utilizar o estimador estendido de mínimos quadrados (EMQ) apresentado no Apêndice A.

As resoluções da Equação (3.10) do estimador MQ e da Equação (A.3) do estimador EMQ podem apresentar problemas numéricos e afetar a estabilidade do algoritmo de mínimos quadrados caso as matrizes Ψ ou Ψ^* sejam mal condicionadas. Uma alternativa é a ortogonalização das matrizes, onde os métodos são conhecidos como mínimos quadrados ortogonais (Chen et al., 1989).

Identificação com restrições

Em alguns casos, como será visto posteriormente neste trabalho, é interessante realizar a estimação de parâmetros sujeita a um conjunto de restrições. Uma possibilidade, apresentada por Draper et al. (1998) para minimizar a função custo $J_{MQ} = \boldsymbol{\xi}^{\,T}\boldsymbol{\xi}$ sujeita a um conjunto de n_r restrições de igualdade sobre os parâmetros, da forma $\mathbf{c} = S\theta$, é resolver o seguinte problema

de otimização:

$$\hat{\theta}_{\text{MQR}} = \begin{array}{c} \arg\min \\ \theta : \mathbf{c} = S\theta, \end{array} \left[\boldsymbol{\xi}^{T} \boldsymbol{\xi} \right] \tag{3.11}$$

em que os valores de $\mathbf{c} \in \mathbb{R}^{n_r \times 1}$ e $S \in \mathbb{R}^{n_r \times n_\theta}$ são conhecidos. A solução da Equação (3.11) é dada por (Draper et al., 1998):

$$\hat{\boldsymbol{\theta}}_{\text{MQR}} = \hat{\boldsymbol{\theta}}_{\text{MQ}} - \mathcal{A}[S\mathcal{A}]^{-1}(S\hat{\boldsymbol{\theta}}_{\text{MQ}} - \mathbf{c}), \tag{3.12}$$

onde $\mathcal{A} = [\Psi_i^{\text{T}}\Psi_i]^{-1}S^{\text{T}}$ e $\hat{\theta}_{\text{MQ}}$ é a solução de mínimos quadrados convencional (Equação (3.10)). Um aspecto importante da solução da Equação (3.12) é que ela não é iterativa.

3.2.3 Seleção de Estrutura

Como visto na Equação (3.6), o número de termos possíveis, ou seja, o número de termos candidatos para o modelo, cresce muito com o aumento do grau de não linearidade l e dos atrasos máximos n_y, n_u e n_e. O número de termos candidatos para os modelos monovariáveis é dado pela expressão (Korenberg et al., 1988).

$$n_\theta = M + 1, \tag{3.13}$$

onde n_θ é o número total de termos do modelo e

$$M = \sum_{i=1}^{l} n_i, \qquad n_i = \frac{n_{i-1}(n_y + n_u + n_e + i - 1)}{i}, \qquad n_0 = 1.$$

Porém, nem todos os termos candidatos são importantes na representação do modelo. De acordo com Çinar (1995), a utilização de uma estrutura não compatível com a não linearidade dos dados pode afetar a qualidade do

modelo gerado e os esforços na identificação. Além disso, a sobreparametrização do modelo pode apresentar regimes dinâmicos espúrios (Aguirre e Billings, 1995). Logo, a seleção de estrutura de modelos consiste em escolher os termos mais relevantes para a composição do modelo.

Um critério amplamente utilizado na seleção de estrutura de modelos NARMAX polinomiais é a taxa de redução de erro (ERR do inglês, *error reduction ration*) (Billings et al., 1989). Esse critério, que é baseado na minimização do erro de predição de um passo à frente, avalia a importância dos termos do modelo de acordo com a sua capacidade de explicar a variância dos dados de saída.

Reescrevendo o modelo da Equação (3.7) como

$$\begin{aligned} y(k) &= \psi^T(k-1)\hat{\theta} + \xi(k) \\ &= \sum_{i=1}^{n_\theta} \hat{\theta}_i \psi_i(k-1) + \xi(k), \end{aligned} \tag{3.14}$$

onde n_θ é o número de parâmetros, pode-se, de forma análoga, representar o modelo na base ortogonal da seguinte forma:

$$y(k) = \sum_{i=1}^{n_\theta} \hat{g}_i w_i(k-1) + \xi(k), \tag{3.15}$$

sendo que $\hat{g}_i$ são os parâmetros estimados e w_i os regressores ortogonais sobre os dados.

Utilizando a Equação (3.15), define-se a variância do resíduo de modelagem $\xi(k)$ como:

$$\text{var}\{\xi(k)\} = \lim_{N\to\infty} \frac{1}{N}\left[\mathbf{y}^T\mathbf{y} - \sum_{i=1}^{N} g_i^2 w_i^T w_i\right], \tag{3.16}$$

sendo $\mathbf{y}$ o vetor contendo os dados de saída.

Caso nenhum termo seja adicionado ao modelo, ou seja, valor de $n_\theta = 0$, a variância de $\xi(k)$ será o valor quadrático da saída $y(k)$. Para cada i-ésimo termo adicionado ao modelo, a variância de $\xi(k)$ decresce de um fator $\frac{1}{N}(g_i^2 w_i^T w_i)$, onde w_i é o termo incluído e g_i o seu respectivo parâmetro. A redução no valor da variância pode ser normalizada com relação ao erro quadrático médio do sinal de saída, resultando na taxa de redução de erro para cada termo que é definida como:

$$[\text{ERR}]_i = \frac{(g_i^2 w_i^T w_i)}{\mathbf{y}^T \mathbf{y}}. \tag{3.17}$$

O critério ERR permite ordenar os termos candidatos por relevância, sendo que os candidatos com maior ERR possuem maior relevância, mas não define quantos termos devem compor o modelo. Existem várias formas de se tentar determinar o número "ideal" de termos necessários em um modelo, entre eles destacam-se os critérios de informação de Akaike (Akaike, 1974), o critério de informação de Bayes (Kashyap, 1977), entre outros.

De acordo com o critério de Akaike, o número de termos de um modelo deve minimizar a seguinte função de custo:

$$\text{AIC}(n_\theta) = N \ \ln(\text{var}\{\xi(k)\}) + 2n_\theta, \tag{3.18}$$

sendo N o número de dados, $\text{var}\{\xi(k)\}$ a variância dos resíduos e n_θ o número de termos do modelo.

O número de termos determinado a partir do critério de Akaike minimiza a variância dos resíduos de identificação partindo de uma estrutura previamente ajustada pelo ERR. Porém, não se pode afirmar que o número de termos escolhidos será capaz de reproduzir as propriedades dinâmicas do sistema original (Aguirre e Billings, 1994).

3.2.4 Modelo MIMO polinomial

Modelos MIMO são modelos com mais de uma variável de entrada e mais de uma variável de saída. O modelo NARMAX da Equação (3.5) pode ser generalizado para o caso multivariável, como (Billings et al., 1989):

$$\begin{aligned}\mathbf{y}(k) =& F^l[\mathbf{y}(k-1),\ldots,\mathbf{y}(k-n_y),\mathbf{u}(k-\tau_d),\ldots,\mathbf{u}(k-n_u),\ldots \\ & \ldots,\mathbf{e}(k-1),\ldots,\mathbf{e}(k-n_e)] + \mathbf{e}(k),\end{aligned} \tag{3.19}$$

sendo $F^l[\cdot]$ uma função vetorial polinomial de grau l e

$$\mathbf{y}(k) = \begin{bmatrix} y_1(k) \\ y_2(k) \\ \vdots \\ y_m(k) \end{bmatrix}, \quad \mathbf{u}(k) = \begin{bmatrix} u_1(k) \\ u_2(k) \\ \vdots \\ u_r(k) \end{bmatrix}, \quad \mathbf{e}(k) = \begin{bmatrix} e_1(k) \\ e_2(k) \\ \vdots \\ e_m(k) \end{bmatrix} \tag{3.20}$$

são os vetores de saída, entrada e ruído, respectivamente, m é o número de saídas e r o número de entradas. Como o modelo é multivariável, qualquer variável de qualquer subsistema pode ser utilizada para compor o modelo de qualquer subsistema.

O i-ésimo subsistema é representado por:

$$\begin{aligned}y_i(k) = \theta_0^i + \sum_{i_1=1}^{M_r} \theta_{i_1}^i x_{i_1}(k) + \sum_{i_1=1}^{M_r}\sum_{i_2=i_1}^{M_r} \theta_{i_1 i_2}^i x_{i_1}(k)x_{i_2}(k) + \ldots + \\ \sum_{i_1=1}^{M_r} \ldots \sum_{i_l=i_{l-1}}^{M_r} \theta_{i_1\ldots i_l}^i x_{i_1}(k)\ldots x_{i_l}(k) + e_i(k),\end{aligned} \tag{3.21}$$

com $i = 1,\ldots,m$, sendo que θ_i são os parâmetros a estimar, x_i os regressores, y_i a saída do i-ésimo subsistema, $e_i(k)$ é o erro da equação de regressão desse subsistema e $M_r = m(n_y + n_e) + r \times n_u$.

É importante salientar que cada saída $y_i(k)$ pode ter um conjunto diferente de regressores e atrasos máximos diferentes, e que o mesmo estimador do caso monovariável pode ser utilizado para identificar os parâmetros de cada subsistema.

Capítulo 4

Aprendizado de movimentos de alcance

Neste capítulo é apresentada a principal contribuição dessa tese, que é uma metodologia para aplicar a PPD de robôs no aprendizado de movimentos de alcance utilizando modelos não lineares autorregressivos (NAR) polinomiais para codificar trajetórias baseadas em sistemas dinâmicos, representados no espaço de estados e com garantia de estabilidade assintótica local ao alvo.

4.1 Definição do problema

O problema abordado neste trabalho é o aprendizado de movimentos de alcance de robôs a partir de um conjunto de demonstrações. Cada demonstração é uma trajetória no espaço de trabalho do robô. As trajetórias são obtidas através da execução do robô controlado por um sensor de força/movimento ou através de um *joystick* operado por um humano, onde as posições são coletadas em uma dada taxa de amostragem. Após o aprendizado, o robô deverá estar apto para reproduzir trajetórias similares às demonstrações sem interferência humana.

Em uma situação ideal, o movimento aprendido pelo robô deve ser robusto a perturbações temporais, causadas por atrasos de comunicação e pela

dinâmica dos controladores de baixo nível, e a perturbações espaciais ocasionadas por algum erro de atuação ou sensoriamento. Espacialmente, também é desejável que trajetórias similares às demonstrações sejam executadas pelo robô mesmo iniciando o movimento de pontos diferentes das demonstrações e mesmo com a alteração da posição do alvo durante a execução.

Seguindo as ideias de sucesso da literatura(Khansari-Zadeh e Billard, 2011; Lemme et al., 2013), o objetivo é aplicar a PPD codificando trajetórias utilizando sistemas dinâmicos não lineares representados no espaço de estados. Os modelos dinâmicos estimados devem reproduzir, na região dos dados de treinamento, trajetórias que sempre alcançam e param no alvo e que sejam similares às demonstrações. Por isso, o problema principal requer:

- Que o modelo dinâmico seja autônomo. Esta característica garante a robustez do modelo a perturbações temporais;

- Que o alvo seja um ponto fixo atrator e que a sua bacia de atração cubra todos os pontos das demonstrações. Desta forma, qualquer movimento inicializado próximo à região das demonstrações, ou seja, dentro da bacia de atração, irá terminar no alvo;

- Que a distância, por exemplo euclidiana, entre os pontos das demonstrações e a fronteira da bacia de atração do alvo seja a maior possível, o que aumenta a robustez espacial do modelo.

Neste trabalho, estes modelos serão denominados de modelos RM (do inglês *Reaching Motions*) e serão codificados utilizando modelos não lineares autorregressivos (NAR) polinomiais estimados utilizando as técnicas de identificação de sistemas apresentadas na Seção 3.2.

4.2 Aprendizado utilizando modelos NAR

O modelo NAR escolhido para codificar os movimentos tem o atraso máximo n_y de 1 e é representado no espaço de estados. A representação do sistema NAR é dada por:

$$\mathbf{y}(k) = F^{\ell}[\mathbf{y}(k-1)] + \mathbf{e}(k), \tag{4.1}$$

onde

$$\mathbf{y}(k) = [y_1(k)\ y_2(k)\ \ldots\ y_p(k)]^{\mathrm{T}}\ ,$$
$$F^{\ell}[\cdot] = \left[f_1^{\ell}[\cdot]\ f_2^{\ell}[\cdot]\ \ldots\ f_p^{\ell}[\cdot]\right]^{\mathrm{T}}\ ,$$

em que $y_i(k)$, $i = 1, 2, \ldots, p$ são as componentes de $\mathbf{y}(k) \in \mathbb{R}^p$ que é um vetor que representa o movimento do robô sem ambiguidade[1] no instante k, $F^{\ell}[\cdot]$ uma função vetorial polinomial com grau de não linearidade $\ell \in \mathbb{N}$ e $\mathbf{e}(k)$ é a parte dos dados que não foi explicada por $F^{\ell}[\cdot]$.

Apesar do estimador de mínimos quadrados para o modelo NAR estar sujeito à polarização, os experimentos iniciais realizados neste trabalho não apresentaram mudanças significativas nos resultados quando se utilizou a parcela de média móvel (MA), ou seja, o modelo NARMA com o estimador estendido de mínimos quadrados. Desta forma, optou-se por utilizar o modelo NAR, pois o esforço computacional é menor.

Para identificar os modelos, é utilizado um conjunto de n demonstrações $\mathbf{Y}_{(n)}$, que são interpretadas como trajetórias em um espaço de estados $\mathbb{R}^p$. Além disso, assume-se que $\mathbf{Y}_{(n)}$ são curvas integrais de um campo vetorial subjacente. O objetivo é encontrar modelos NAR que aproximem o campo

[1] A ambiguidade acontece quando a trajetória de um movimento contém a mesma coordenada em períodos distintos de tempos, ou seja, quando existem interseções na trajetória do movimento.

vetorial. Neste caso, $\mathbf{y}(k) = [y_1(k) \ \ldots \ y_p(k)]^{\mathrm{T}}$ (veja Equação (4.1)), onde $y_1(k), \ldots, y_p(k)$ são coordenadas da trajetória. Por isso, procura-se por modelos compostos por p equações de diferença de primeira ordem ($n_{y_1} = \cdots = n_{y_p} = 1$), para produzir $\hat{y}_1(k), \ldots, \hat{y}_p(k)$.

O conjunto de regressores candidatos das funções $F^{\ell}[\cdot]$ das variáveis $y_1(k), \ldots, y_p(k)$ são compostos por todas as possíveis combinações até o grau ℓ dessas variáveis mais um termo constante. Assumindo, sem perda de generalidade, que o alvo é a origem do espaço de estados, é possível garantir que ele seja um ponto fixo removendo o termo constante do conjunto de regressores candidatos (Aguirre e Mendes, 1996). Assim, todos os regressores do modelo dependerão de $y_1(k), \ldots, y_p(k)$. Os regressores de cada equação do modelo são automaticamente e independentemente escolhidos do conjunto de candidatos, usando o critério ERR descrito na Subseção 3.2.3. Esse procedimento permite que os termos em cada equação sejam diferentes. Isso é feito em duas etapas, primeiro os regressores são ranqueados na ordem descendente de relevância em termos do critério ERR (Billings et al., 1989); e depois, o número de termos de cada equação do modelo, $\dim(\theta_1) = n_{\theta_1}, \ldots, \dim(\theta_p) = n_{\theta_p}$, é escolhido utilizando o critério de Akaike (Akaike, 1974). Um típico modelo tem a forma geral:

$$\begin{aligned} y_1(k) &= F_1^{\ell}[y_1(k-1), \ \ldots, \ y_p(k-1)] + e_1(k) \\ y_2(k) &= F_2^{\ell}[y_1(k-1), \ \ldots, \ y_p(k-1)] + e_2(k) \\ &\cdots \\ y_p(k) &= F_p^{\ell}[y_1(k-1), \ \ldots, \ y_p(k-1)] + e_p(k). \end{aligned} \tag{4.2}$$

A análise do estado estacionário do modelo (4.2) tem um papel importante neste trabalho. Assumindo que o modelo é estável, no estado estacionário

tem-se que $y_1(k) = y_1(k-1) = \bar{y}_1, \ldots, y_p(k) = y_p(k-1) = \bar{y}_p$. Assim, desconsiderando o ruído, os pontos de equilíbrio do modelo (4.2) são dados por $(\bar{y}_1, \ldots, \bar{y}_p)$, que são as soluções para o conjunto de equações:

$$\begin{aligned} \bar{y}_1 &= F_1^\ell[\bar{y}_1, \ldots, \bar{y}_p], \\ \bar{y}_2 &= F_2^\ell[\bar{y}_1, \ldots, \bar{y}_p], \\ &\ldots \\ \bar{y}_p &= F_p^\ell[\bar{y}_1, \ldots, \bar{y}_p], \end{aligned} \tag{4.3}$$

e a estabilidade de cada ponto de fixo (ponto de equilíbrio), como descrito na Subseção 3.1.2, pode ser determinada pelos autovalores $\lambda_1, \ldots, \lambda_p$ da matriz Jacobiana

$$\mathrm{D}F(\boldsymbol{y}) = \begin{bmatrix} \dfrac{\partial F_1^\ell}{\partial y_1} & \cdots & \dfrac{\partial F_1^\ell}{\partial y_p} \\ \cdots & \ddots & \cdots \\ \dfrac{\partial F_p^\ell}{\partial y_1} & \cdots & \dfrac{\partial F_p^\ell}{\partial y_p} \end{bmatrix} \tag{4.4}$$

evoluída em cada ponto fixo.

Se um ponto fixo $\bar{\boldsymbol{y}}^*$ é um atrator do sistema, então ele possui uma bacia de atração. A bacia de atração de $\bar{\boldsymbol{y}}^*$ é o conjunto de todas as condições iniciais, tais que, quando o sistema parte de uma destas condições, o sistema se direciona para $\bar{\boldsymbol{y}}^*$.

4.2.1 Alocação de pontos fixos

Uma importante contribuição neste trabalho é a imposição de pontos fixos durante a estimação de parâmetros para alcançar alguma propriedade desejada no modelo. Seja $\bar{\boldsymbol{y}}^* = [y_1^*, y_2^* \ldots y_p^*]^{\mathrm{T}}$ as coordenadas de um ponto fixo.

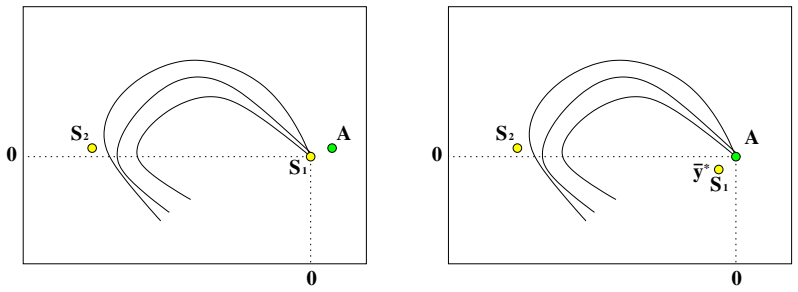
A fim de impor que o modelo da Equação (4.2) tenha esse equilíbrio, uma restrição na forma $\mathrm{c} = S\boldsymbol{\theta}$ com

$$\mathbf{c} = \begin{bmatrix} y_1^* \\ y_2^* \\ \vdots \\ y_p^* \end{bmatrix}, \qquad S = \begin{bmatrix} F_1^\ell[y_1^*, y_2^*, \ldots, y_p^*] \\ F_2^\ell[y_1^*, y_2^*, \ldots, y_p^*] \\ \vdots \\ F_p^\ell[y_1^*, y_2^*, \ldots, y_p^*] \end{bmatrix} \tag{4.5}$$

deve ser satisfeita. Isto pode ser alcançado utilizando o estimador MQR (Equação (3.12)). Lembre-se de que para impor o ponto fixo trivial $\bar{\boldsymbol{y}}^* = \mathbf{0}$ no modelo, basta remover o termo constante do conjunto de regressores candidatos.

Usando as técnicas padrões de identificação de sistemas é possível obter modelos na forma da Equação (4.2) que são capazes de produzir curvas integrais na região das demonstrações que se assemelham às trajetórias demonstradas. Estes modelos sempre possuem um ponto fixo na posição do alvo, mas isto não garante que o ponto sempre será um atrator. Isso acontece porque o estimador MQ não impõe nenhuma restrição quanto à estabilidade do ponto fixo. A Figura 4.1(a) mostra exemplos de pontos fixos de um modelo bidimensional M_1, onde o ponto A é um atrator e os pontos S_1 e S_2 são selas.

Uma das contribuições deste trabalho é investigar como modificar a localização de alguns pontos fixos de um modelo a fim de obter ou melhorar algumas das características desejadas e aprender movimentos de alcance. Nesta linha, chamamos a atenção para o seguinte *fato*: considere um modelo irrestrito M_1 com um ponto fixo em $\boldsymbol{y}_1^*$ que foi estimado a partir dos dados $\mathbf{Y}_{(n)}$. Os parâmetros podem ser reestimados dos mesmos dados usando o estimador MQR (Equação (3.12)) com uma restrição do tipo (Equação (4.5)) a fim de impor um novo ponto fixo em $\boldsymbol{y}^*$. O modelo restrito é referenciado



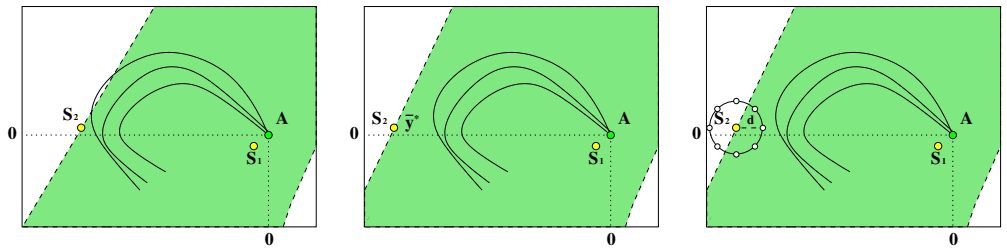
(a) Modelo sem restrições M_1. (b) Modelo restrito M_2.

Figura 4.1: Modelo bidimensional identificado utilizando o estimador MQ (a), e o estimador MQR (b). Em M_2 a *localização* de S_1 foi imposta por meio de uma estimação restrita, isso é $S_1 = \bar{\boldsymbol{y}}^* = [y_1^*, y_2^*]^{\mathrm{T}}$ é a nova posição de S_1, e a origem continua sendo um ponto fixo – porque o termo constante foi removido do conjunto de regressores candidatos. O tipo de estabilidade de S_1 e A não modifica, veja Conjectura 1.

como M_2. Agora a seguinte conjectura é apresentada:

Conjectura 1 [Tipo de estabilidade de ponto fixo mantido sob pequenas perturbações] *O ponto fixo* $\boldsymbol{y}^*$ *de* M_2 *será do mesmo tipo que o de* $\boldsymbol{y}_1^*$ *de* M_1 *para* $|\boldsymbol{y}_1^* - \boldsymbol{y}^*|$ *suficientemente pequeno.*

Uma justificativa da Conjectura 1 é a seguinte. O estimador MQR (Equação (3.12)) está na forma $\hat{\boldsymbol{\theta}}_{\mathrm{MQR}} = \hat{\boldsymbol{\theta}}_{\mathrm{MQ}} - \mathcal{A}[S\mathcal{A}]^{-1}\epsilon$, onde $\epsilon = S\hat{\boldsymbol{\theta}}_{\mathrm{MQ}} - \mathbf{c}$ é o "erro" pelo qual o vetor de parâmetros estimado irrestrito perde com a restrição. Se $\hat{\boldsymbol{\theta}}_{\mathrm{MQ}}$ satisfaz a restrição, então $\epsilon = 0$ e ambos os vetores estimados, restrito e irrestrito, serão iguais. Consequentemente $\lim_{\epsilon \to 0} \hat{\boldsymbol{\theta}}_{\mathrm{MQR}} = \hat{\boldsymbol{\theta}}_{\mathrm{MQ}}$. Assim, desde que ϵ não seja muito grande, M_2 e seus pontos fixos podem ser interpretados como uma versão perturbada de M_1 e seus pontos fixos. Como as restrições implementadas usando a Equação (4.5) referem-se à localização de pontos fixos, ϵ está diretamente relacionado a $|\boldsymbol{y}_1^* - \boldsymbol{y}^*|$. Então, para $|\boldsymbol{y}_1^* - \boldsymbol{y}^*|$ suficientemente pequeno o tipo dos pontos fixos de M_1 e M_2 será o mesmo, isto é, as selas permanecerão selas, os repulsores permanecerão repulsores e os atratores permanecerão atratores. Este foi o resultado obtido em um grande número de casos, e está esquematicamente ilustrado na Figura 4.1, onde os pontos amarelos S_1 e S_2 representam pontos de sela



(a) Modelo M_2 com a bacia de atração.

(b) Modelo M_3 obtido com a alocação do ponto S_2 pela estimação restrita

(c) Oito pontos distribuídos uniformemente em torno de S_2, com distância d.

Figura 4.2: Possível modelo bidimensional com a bacia de atração estimada na cor verde. A sela S_2 em (b) foi imposta à esquerda de sua contraparte em (a) a fim de aumentar a bacia de atração e incluir todos os dados de demonstração. M_3 é um modelo reaching motions.

e o ponto verde A é um atrator. Resultados estatísticos da aplicação da Conjectura 1 são apresentados na Capítulo 5.

A bacia de atração do alvo (atrator A) é mostrado na Figura 4.2(a) em verde e não inclui alguns pontos das demonstrações. A bacia de atração geralmente tem como limite a variedade estável de uma sela (ponto fixo). Isto sugere um procedimento muito simples para aumentar o tamanho da bacia: "afastar" a sela. De fato, o que é feito é impor um ponto fixo fora da bacia (para a esquerda na Figura 4.2(a)) como descrito anteriormente. Se a distância $|S_2 - \boldsymbol{y}^*|$ for pequena, com base na Conjectura 1, espera-se que ele permaneça uma sela e, como consequência, a bacia seja automaticamente aumentada nessa direção, como ilustrado na Figura 4.2(b).

A escolha do novo local para a sela S_2 é um pouco subjetiva. O ideal é que a posição seja distante (para a esquerda no caso da Figura 4.2(b)) de tal forma que pequenos erros de atuação e sensoriamento durante a execução do modelo ainda resultarão em trajetórias dentro da bacia de atração e, consequentemente, o robô será direcionado ao alvo. Por outro lado, se S_2 for colocado longe demais, então há a chance de a Conjectura 1 ser violada e ele não ser mais uma sela, resultando uma bacia de atração completamente

diferente. Um possível procedimento prático para estender ainda mais a bacia de atração é impor o ponto fixo S_2 em vários locais e escolher um baseado na bacia resultante. Uma sugestão simples é tomar m pontos igualmente localizados em torno de S_2 a uma distância d, como ilustrado na Figura 4.2(c).

Em síntese, a metodologia proposta nesta seção é: dado um conjunto de demonstrações, uma estrutura para o modelo é selecionada e seus parâmetros identificados, usando o estimador MQ padrão. Em seguida, aplicando o estimador com restrições MQR, o modelo é modificado para transformá-lo em um modelo de movimento de alcance (RM), isto é, um modelo com um atrator no alvo e uma bacia de atração que cobre todos os pontos das demonstrações. A próxima seção irá propor um conjunto de algoritmos que implementam tal metodologia.

4.3 Implementação Numérica

O Algoritmo 1 proposto a seguir é genérico e pode ser usado (i) para tornar o alvo um atrator, (ii) para aumentar a bacia de atração, e (iii) para otimizar outras características do modelo. Ele recebe como entrada as demonstrações $\mathbf{Y}_{(n)}$, o grau de não linearidade ℓ, o número máximo de iterações $MaxIt$, a distância d, o número de pontos a serem testados m, o modelo irrestrito e retorna um modelo NAR restrito. O Algoritmo 1 é utilizado para realizar mudanças no modelo irrestrito até que uma característica específica (por exemplo, o alvo atrator e o tamanho da bacia de atração) seja obtida ou otimizada.

O laço principal do Algoritmo 1 (linhas 3–16) funciona da seguinte forma: Inicialmente é verificado se o modelo corrente $Model$ possui a característica procurada pelo algoritmo, caso tenha, o processo é finalizado e o modelo é

Algoritmo 1: Algoritmo geral

```
   Input: Y_(n), ℓ, MaxIt, d, m, Model
 1 begin
 2   Best ← Model;     i ← 0 ;
 3   while i < MaxIt do
 4     if hasCharacteristic(Model) then
 5       return Model;
 6     end
 7     FP ← getFixedPoint(Model);
 8     M_aux ← imposeFP(Y_(n), ℓ, FP, d, m, Model);
 9     if acceptable(M_aux) then
10       Model ← M_aux;
11       Best ← bestModel(Best,M_aux);
12     else
13       change the distance d;
14     end
15     i ← i + 1 ;
16   end
17   return Best;
18 end
19 Function imposeFP(Y_(n), ℓ, FP, d, m, Model)
20   begin
21     choose k points in FP;
22     i ← 0 ;
23     foreach P_1 in k do
24       Dp ← EvenlyDistributed(P_1, d, m);
25       foreach P in Dp do
26         C ← MakeFPConstraint(P);
27         V[i] ← estimateModel(Model, Y_(n), ℓ, C);
28         i ← i + 1 ;
29       end
30     end
31     best ← bestModel(V);
32     return best;
33   end
```

retornado (linhas 4–6). Se o modelo não possui a característica procurada a iteração continua. Em seguida são obtidos todos os pontos fixos FP de $Model$ e é executada a função `imposeFP`, que retorna o melhor modelo M_{aux}

após aplicar a imposição de pontos fixos, utilizando os pontos de FP (linhas 7 e 8). Por fim, é verificado se M_{aux} é aceitável. Caso seja aceitável, o modelo M_{aux} vira o modelo corrente $Model$ e o melhor modelo até o momento é salvo em $Best$. Caso contrário, a distância d é modificada (linhas 9–14). Em relação ao funcionamento da função `imposeFP`, inicialmente ela seleciona,[2] do conjunto FP, os k pontos fixos onde serão aplicados a imposição de pontos fixos (linha 21). Para cada ponto de k são definidos m pontos distribuídos uniformemente ao seu redor e armazenados no conjunto Dp. Cada ponto P de Dp define um novo modelo que é obtido através da re-estimação do modelo com uma restrição que define um ponto fixo na posição P (linhas 23–29). Do conjunto de modelos obtidos ($k \times m$) é escolhido o melhor modelo que é retornado pela função (linhas 31 e 32).

Uma vez que o Algoritmo 1 tem finalidade geral, algumas funções devem ser definidas de acordo com a aplicação. A função mais importante é `bestModel` (linha 11) que define como os modelos serão avaliados e é responsável por guiar todo o processo de busca do algoritmo. Pode ser, por exemplo, uma função que calcula a precisão na execução de uma trajetória aprendida, como será detalhado na Subseção 4.3.1. A função `hasCharacteristic` (linha 5) é usada para verificar se o modelo tem uma característica esperada, por exemplo, o alvo atrator. Finalmente, a função `acceptable` (linha 9) é usada para verificar se o modelo possui as características mínimas para continuar evoluindo. Se o modelo não for aceitável, a distância d é modificada e a pesquisa continua.

As próximas subseções mostram como o Algoritmo 1 pode ser usado para

[2]Varias seleções podem ser feitas, por exemplo, todos os pontos fixos exceto o alvo, somente as selas, todos os pontos fixos dentro da bacia de atração e outras possibilidades. Neste trabalho é utilizado todos os pontos fixos existentes na região que está sendo avaliado o modelo, exceto o alvo.

alterar características específicas do modelo.

4.3.1 Obtendo um atrator no alvo

Em nossos experimentos, os modelos geralmente tinham um atrator na posição alvo ou, em casos raros, o alvo não era um atrator, mas tinha um ou mais atratores próximos da posição alvo. Essas duas situações acontecem porque as demonstrações utilizadas na estimação dos modelos sempre descrevem trajetórias que terminam em torno do mesmo alvo, definindo uma ou mais regiões atrativas próximas a esta região. Na situação em que o alvo não é um atrator, aplicando as restrições baseadas na Conjectura 1, é possível reestimar os parâmetros e obter um modelo com um atrator no alvo, conforme ilustrado esquematicamente na Figura 4.1.

Baseado no Algoritmo 1, é possível implementar uma função denominada `getTargetAttractor`, que é chamada para verificar se um modelo tem um atrator no alvo e, caso não possua, para modificar o modelo e obter essa característica. A ideia é obter m modelos com um ponto fixo imposto à distância d ao redor do alvo usando a função `imposeFP` e verificar se o melhor modelo obtido tem um atrator no alvo. Se nenhum modelo tiver um atrator no alvo, a distância d é aumentada. É importante lembrar que o alvo é sempre um ponto fixo.

Para avaliar e escolher o melhor modelo, é utilizado a função `bestModel`. Esta função analisa todos os modelos utilizando uma métrica que avalia a precisão na reprodução dos movimentos aprendidos. Uma métrica possível é chamada de varredura da área do erro (SEA, do inglês swept error area) (Khansari-Zadeh e Billard, 2014), que quantifica a precisão do modelo em duas dimensões. Esta métrica compara as trajetórias obtidas pelo modelo

com o conjunto de demonstrações $\mathbf{Y}_{(n)}$ e é calculada por

$$\varepsilon = \frac{1}{n}\sum_{m=1}^{n}\sum_{t=1}^{T^m-1}\mathcal{A}\left(\hat{\mathbf{Y}}_{(m,t)}, \hat{\mathbf{Y}}_{(m,t+1)}, \mathbf{Y}_{(m,t)}, \mathbf{Y}_{(m,t+1)}\right), \tag{4.6}$$

onde $\mathcal{A}\left(p^1,p^2,p^3,p^4\right)$ corresponde à área do quadrilátero gerado pelos quatro pontos p^1 até p^4, T^m é o número de pontos na m^{th} demonstração e $\hat{\mathbf{Y}}_{(m,t)}$ é o ponto t da trajetória $\hat{\mathbf{Y}}_{(m)}$ que começa no mesmo ponto de partida da demonstração $\mathbf{Y}_{(m)}$. A Figura 4.3 mostra um exemplo da SEA entre uma demonstração e uma trajetória estimada.

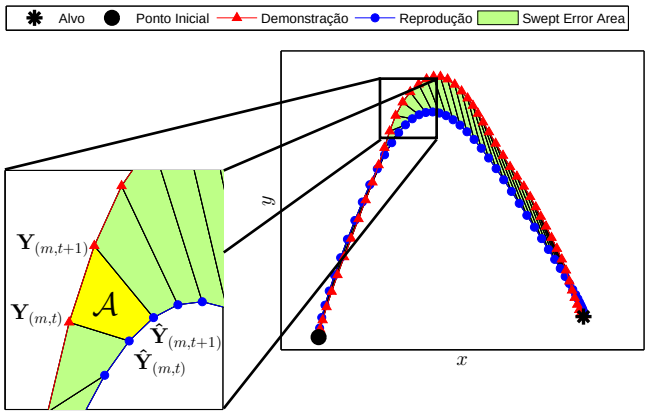


Figura 4.3: Ilustra a varredura da área do erro entre uma demonstração e uma trajetória estimada.

A princípio, não se pode garantir que todas as trajetórias que deixam os pontos iniciais atinjam o alvo, uma vez que alguns desses pontos podem não pertencer à bacia de atração do alvo. Portanto, pode-se usar um cálculo parcial da SEA para escolher o modelo, como segue:

1. determinar, para cada modelo, o número de trajetórias que alcançam o alvo, iniciando dos pontos iniciais das demonstrações;

2. escolha os modelos que possuem mais trajetórias que alcançam o alvo e calcule a SEA para essas trajetórias;

3. escolha o modelo que apresentar o menor valor de SEA.

Outras funções necessárias são `acceptable` e `hasCharacteristic`, que neste caso são similares, verificando se o modelo possui um atrator no alvo.

4.3.2 Analisando a bacia de atração do alvo

Como afirmado na Seção 4.1, procura-se modelos em que a bacia de atração do alvo contenha todos os pontos das demonstrações. Para calcular a bacia, a região R onde a bacia será estimada é definida. R deve representar o domínio esperado do sistema e conter todos os pontos das demonstrações. Como o número de pontos de R é infinito, optou-se por estimar a bacia de atração de forma discreta. Desta forma, dada uma região contínua R, define-se um mapa discreto M de R, que é composto por um conjunto de células regulares formando uma grade (veja a Figura 4.4(a)), e aplicamos o Algoritmo 2 para estimar a bacia de atração.

O Algoritmo 2 usa dois conjuntos, um para armazenar as células pertencentes à bacia de atração chamada $Basin$ e outro para armazenar as células de M já testadas, chamado $Visited$. Inicialmente, a célula com o alvo é colocada nos conjuntos $Basin$ e $Visited$ (linhas 2–3). Enquanto $Visited$ não contém todas as células de M, o modelo é iterado para cada célula C vizinha às células que estão em $Visited$, começando no ponto central de C até alcançar uma célula vizinha C_{next} (linhas 5–7). Se C_{next} não pertencer à bacia, então C é vinculado a C_{next}, indicando que uma iteração do modelo a partir de C atinge C_{next} (linhas 8 - 9). Se C_{next} pertence a $Basin$, então C e todas as células vinculadas a ele são colocadas em $Basin$ (linha 11–12). Todas as células vizinhas das células em $Visited$ são colocadas em $Visited$ (linha 15), expandindo o conjunto de células já verificadas e retornando para a linha 4. Depois de verificar todo mapa M, o algoritmo retorna a bacia de atração no conjunto $Basin$.

Algoritmo 2: Estimando a bacia de atração do alvo

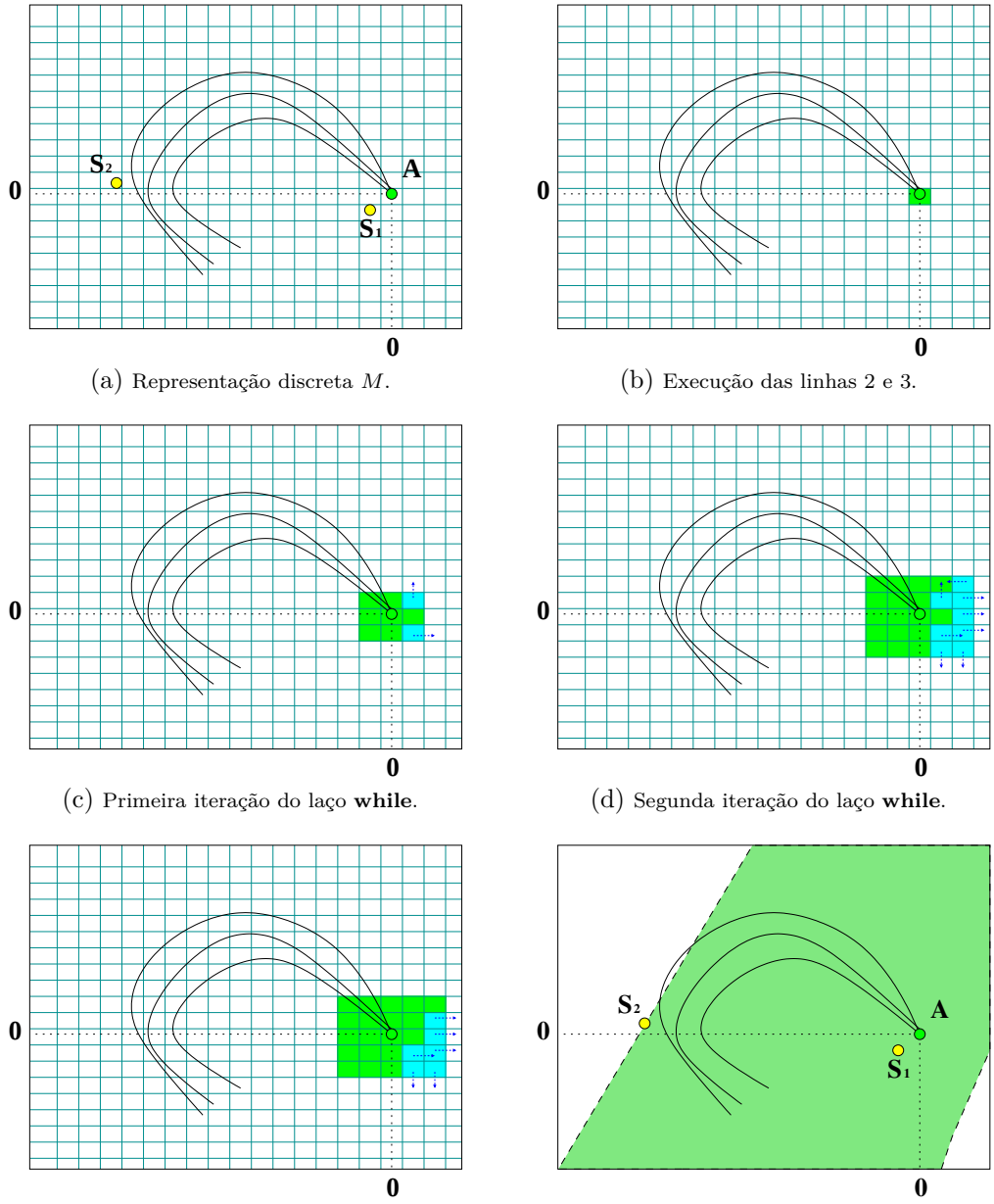
```
   Input: Model, map M
1  begin
2      Put target cell in Basin;
3      Put target cell in Visited;
4      while Visited does not have all cells of M do
5          foreach C cell in the neighborhood of Visited do
6              Put C in Aux;
7              Iterate the Model starting from C until reaches the next
                cell C_next;
8              if C_next ∉ Basin then
9                  link the cell C to the C_next;
10             else
11                 Put C in Basin;
12                 Put cells linked to C in Basin;
13             end
14         end
15         Put Aux in Visited
16     end
17     return Basin;
18 end
```

Uma observação importante é a possibilidade de ter um ponto fixo atrator na célula C. Nesse caso, a iteração do modelo na linha 7 provavelmente não alcançaria outra célula. Desta forma, a função na linha 7 nunca terminará se não for capaz de detectar tal situação. Na implementação utilizada neste trabalho é permitida apenas um número máximo de iterações em cada célula. Assim, se o número máximo de iterações for atingido, o algoritmo assume que existe um atrator na célula C e retorna $C_{next} = C$.

A Figura 4.4 ilustra o funcionamento do Algoritmo 2, onde as células coloridas de azul pertencem ao conjunto $Visited$, as células de cor verde pertencem aos conjuntos $Visited$ e $Basin$ e as setas pontilhadas em azul representam as células C que foram vinculadas as células C_{next}. Inicialmente a área de trabalho é discretizada em uma matriz M como mostra a Figura 4.4(a).

Em seguida o algoritmo adiciona a célula que contém o alvo nos conjuntos *Basin* e *Visited* (Figura 4.4(b)). O resultado da primeira iteração do laço **while** (linhas 4–16) é mostrado na Figura 4.4(c), onde nota-se que duas células foram vinculadas a células que ainda não foram avaliadas. A execução da segunda iteração do laço **while** pode ser visto nas Figuras 4.4(d) e 4.4(e). Na Figura 4.4(d) é mostrado a execução do laço **while** sem ter processado a linha 12, ou seja, não mostra a adição no conjunto *Basin* das células vinculadas a uma célula C que foi adicionada recentemente à bacia de atração. Nota-se que duas células na Figura 4.4(d) estão vinculadas a uma célula que foi adicionada à bacia, logo estas células também pertencem à bacia (linha 12) como é mostrado na Figura 4.4(e). Por fim, a estimativa da bacia de atração do alvo para a Figura 4.4(a) é apresentado na Figura 4.4(f).

Para aumentar a bacia de atração do alvo e transformar o modelo atual em um modelo RM, duas funções baseadas no Algoritmo 1, chamadas `getRM_Model1` e `getRM_Model2`, foram implementadas. A função `getRM_Model1` usa SEA parcial na função `bestModel` e implementa uma função `hasCharacteristic` que verifica se todas as trajetórias que começam dos pontos iniciais das demonstrações atingem o alvo. No entanto, como o modelo não reproduz perfeitamente as trajetórias, nem sempre o uso da SEA parcial será suficiente para garantir que todos os pontos das demonstrações estarão na bacia de atração. Assim, a função `getRM_Model2`, que utiliza a função `bestModel` para escolher modelos com o menor número de células que contém pontos das demonstrações e não pertencem à bacia de atração do alvo, também é necessária. Nesta função, `hasCharacteristic` verifica se é um modelo RM. Ambas, `getRM_Model1` e `getRM_Model2` utilizam a função `acceptable` que aceitam apenas modelos que possuem um atrator no alvo. Se nenhum modelo aceitável for encontrado, a distância d é reduzida



(a) Representação discreta M.

(b) Execução das linhas 2 e 3.

(c) Primeira iteração do laço **while**.

(d) Segunda iteração do laço **while**.

(e) Adicionando as células vinculadas ao conjunto $Basin$ (linha 12).

(f) Exemplo de uma bacia de atração estimada.

Figura 4.4: Funcionamento do Algoritmo 2. As células coloridas de azul pertencem ao conjunto $Visited$ e as células de cor verde pertencem aos conjuntos $Visited$ e $Basin$. As setas pontilhadas em azul representam as células C que foram vinculadas as células C_{next}.

e todos os pontos fixos, além do alvo, serão impostos em locais ligeiramente diferentes na tentativa de melhorar o modelo.

4.3.3 Melhorando um modelo RM

Para avaliar quantitativamente os modelos RM, em 2D, utiliza-se a SEA para medir a precisão da reprodução. Também é proposto neste trabalho uma métrica chamada distância-α, para avaliar a robustez da bacia de atração em relação às perturbações espaciais. Essa métrica é definida abaixo:

Definição 1 *A distância-α é a menor distância euclidiana entre as trajetórias das demonstrações e a borda da bacia de atração do alvo.*

A distância-α indica o maior desvio que pode ocorrer na trajetória antes dessa sair da bacia de atração. Portanto, quanto maior a distância-α, maior é a robustez em relação às perturbações espaciais. Métodos globalmente estáveis possuem uma distância-α infinita.

Neste trabalho, a distância-α é aproximada pela menor distância entre os pontos do centro das células que compõem a borda da bacia de atração e as células que contém as trajetórias das demonstrações em M.

Para otimizar os valores do SEA e distância-α, duas outras funções baseadas no Algoritmo 1 foram implementadas. As funções implementadas `optimizeAlphaDistance` e `optimizeAccuracy` permitem que todos os pontos fixos, exceto o alvo, mudem de localização, não utilizam a função `hasCharacteristic` e codificam a função `acceptable` de forma que somente modelos RM são permitidos. Caso nenhum modelo aceitável for encontrado, a distância d é reduzida. A função `optimizeAlphaDistance` usa a distância-α na função `bestModel` com a restrição de que o modelo escolhido não tenha um valor de SEA maior que o parâmetro *SEAMax*. Por outro lado, a função `optimizeAccuracy` usa o SEA na função `bestModel` com a restrição de que o modelo escolhido não tenha um valor de distância-α menor que o parâmetro *AlphaMin*.

4.3.4 Procedimento completo

Uma vez que cada função foi definida, um procedimento completo para obter um modelo de movimento de alcance (RM) é apresentado no Algoritmo 3, onde as funções são chamadas em sequência. O conjunto de parâmetros de entrada I é o mesmo do Algoritmo 1, isto é, $I = \{\mathbf{Y}_{(n)}, \ell, MaxIt, r, d, Model\}$. Este procedimento completo é denominado de NAR-RM.

Algoritmo 3: NAR-RM

```
1 Model ← estimateModel(Y_(n), ℓ);
2 Model← getTargetAttractor(I);
3 Model← getRM_Model1(I);
4 Model← getRM_Model2(I);
5 Model← optimizeAlphaDistance(I, SEAMax);
6 Model← optimizeAccuracy(I, AlphaMin);
```

O método NAR-RM proposto é validado no próximo capítulo utilizando uma biblioteca 2D de movimentos de caligrafia e também utilizando dois robôs reais, um robô móvel planar e um manipulador industrial que opera em 3D.

Capítulo 5

Experimentos e resultados

Neste capítulo são apresentados os experimentos e resultados obtidos com a aplicação do método NAR-RM proposto no Capítulo 4. Na Seção 5.1 são apresentados os experimentos realizados utilizando a biblioteca *LASA Handwriting Dataset*, composta por movimentos de caligrafia em 2D, e a comparação do método proposto com alguns dos métodos citados no Capítulo 2. O NAR-RM também é avaliado em um robô móvel terrestre não holonômico na Seção 5.2 e em um manipulador industrial na Seção 5.3. Por fim, na Seção 5.4, são discutidos os resultados obtidos.

5.1 Biblioteca LASA Handwriting Dataset

O método NAR-RM foi implementado usando o programa de cálculo numérico MATLAB[1] e executado em um computador Core I7 com 8GB de memória. O código está disponível em (NAR-RM, 2017). Inicialmente o método NAR-RM foi testado com a biblioteca *LASA Handwriting Dataset library*[2], também utilizada em (Khansari-Zadeh e Billard, 2011; Neumann et al., 2013; Lemme et al., 2013; Khansari-Zadeh e Billard, 2014; Neumann e Steil, 2015; Lemme et al., 2015). A biblioteca consiste de 30 padrões de movimentos de

[1]Disponível em: https://www.mathworks.com
[2]Disponível em: https://bitbucket.org/khansari/lasahandwritingdataset

Tabela 5.1: Porcentagem de pontos fixos que não mudam de tipo quando novos pontos são impostos usando a distância d.

d (mm)	0,01	0,05	0,1	0,2	0,5	1	2	4	6	8	10	Nº de pontos impostos
(%) inalterada; $\ell = 3$	100	100	100	99	98	98	97	93	86	80	75	744
(%) inalterada; $\ell = 4$	100	100	100	100	99	98	95	86	78	70	62	1136
(%) inalterada; $\ell = 5$	100	99	99	99	98	97	92	78	65	57	52	1576

caligrafia em 2D. Cada padrão de movimento possui 7 demonstrações com 1.000 pontos. Toda demonstração possui um ponto de partida diferente e o mesmo ponto final em $x = 0$ e $y = 0$.

Nos experimentos, o grau de não linearidade ℓ foi variado de 3 a 5 na construção dos modelos NAR, pois os modelos com valores de $\ell < 3$ não conseguem modelar alguns padrões e para valores de $\ell > 5$ o MATLAB apresenta uma maior incidência de matrizes mal condicionadas no processo de estimação dos parâmetros. Pela análise qualitativa dos padrões da biblioteca, verificou-se que os movimentos podem ser representados por menos pontos, indicando que o intervalo de amostragem é muito curto. Desta forma, com o objetivo de evitar um possível mal condicionamento das matrizes no processo de estimação, optou-se em decimar os dados a uma taxa de 20, resultando em 50 pontos cada. O ponto $(0, 0)$ foi adicionado ao final de todas as demonstrações decimadas, resultando em 51 pontos. Os pontos de equilíbrio são encontrados resolvendo-se simbolicamente o conjunto de equações não lineares da Equação (4.3) usando a função do MATLAB `solve`.

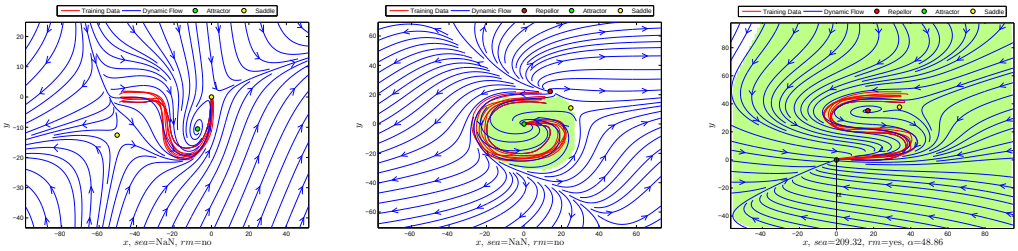
Para verificar numericamente a Conjectura 1, foram compiladas as porcentagem de pontos fixos que mantiveram seu tipo original, após a imposição de um ponto fixo a uma distância d. O experimento utilizou 11 distâncias diferentes (d) dos pontos fixos originais para os 90 modelos, 30 para cada valor de $\ell = 3, 4, 5$ (Tabela 5.1). O número total de pontos fixos foram 94, 143 e 198 para o grupo de modelos com o valor de $\ell = 3, 4, 5$, respectiva-

mente. Para cada ponto fixo, exceto o da origem, 8 pontos foram impostos para cada valor de d, como mostrado na Figura 4.2(c), resultando em um total de 38.016 pontos fixos impostos.

Avaliando a Tabela 5.1, nota-se que, quanto menor o valor de d, maior é a porcentagem de pontos fixos que não mudam de tipo, como esperado pela Conjectura 1. Além disso, para valores mais altos de ℓ, a porcentagem de pontos fixos que não mudam de tipo é menor, devido ao aumento dos graus de liberdade.

No restante desta seção, o NAR-RM foi aplicado usando uma região de avaliação R, definida por quatro vezes o tamanho da menor área retangular de dimensões dx e dy que contém as demonstrações. As trajetórias das demonstrações foram mantidas no centro desta região. No entanto, para uma melhor visualização, as figuras serão apresentadas com a metade da área de R. O mapa discreto M de R foi definido com 100 linhas e 100 colunas. Para impor os pontos fixos foi utilizado $m = 8$ pontos, como mostrado na Figura 4.2(c), e a distância d foi definida como $5 \times dist$, onde $dist$ é o maior valor entre a largura e comprimento de uma célula de M. Para os modelos da bilioteca LASA, $d \approx 7{,}5$ mm. Além disso, $MaxIt$ foi fixado com o valor de 15 em todas as funções, $SEAMax$ foi definido como 20% maior que o atual SEA e $AlphaMin$ com o valor 20% menor do que a atual distância-α.

Antes da aplicação do método, foram avaliados os 90 modelos iniciais obtidos utilizando o estimador de mínimos quadrados (MQ). O número de parâmetros escolhidos para representar os modelos variaram no intervalo de $[6, 18]$, $[17, 28]$ e [24,39] para $\ell = 3, 4$ e 5, respectivamente. Dos 90 modelos iniciais, 41 já são modelos RM. Dos 49 modelos que não são RM, 9 não apresentam um atrator no alvo (veja a Tabela 5.2). A Figura 5.1 mostra o resultado de 3 modelos com $\ell = 3$ obtidos pelo estimador MQ, onde



(a) Modelo sem atrator no alvo.

(b) Modelo com alguns pontos das trajetórias fora da bacia de atração

(c) Modelo RM.

Figura 5.1: Três modelos iniciais obtidos com $\ell = 3$ utilizando o estimador MQ. Execução do método NAR-RM. As figuras mostram o retrato da fase dos modelos identificados em azul, os pontos fixos em amarelo (sela), verde (atrator) e vermelho (repulsor) e a bacia de atração em verde. As trajetórias das demonstrações são mostradas como trajetórias vermelhas. Todos os eixos são mostrados em milímetros.

Tabela 5.2: Resultados obtidos utilizando o estimador MQ irrestrito.

	Total	$\ell = 3$	$\ell = 4$	$\ell = 5$
Número de modelos	90	30	30	30
Alvo não é atrator	9	2	3	4
Modelos RM	41	18	12	11

tem-se um modelo sem atrator no alvo na Figura 5.1(a), um modelo com alguns pontos das trajetórias fora da bacia de atração representada pela cor verde (Figura 5.1(b)) e um modelo desejado de movimento de alcance (RM) (Figura 5.1(c)).

Nos testes com os 9 modelos que inicialmente não tinham um ponto fixo atrator no alvo, a função `getTargetAttractor`, do método NAR-RM, foi capaz de encontrar modelos com um atrator no alvo, impondo pontos fixos com um pequeno valor de d (≈ 1 mm).

O método NAR-RM foi aplicado nos 90 modelos iniciais obtendo 90 modelos RM. Dos 41 modelos iniciais, que já eram modelos RM, a aplicação do NAR-RM melhorou o valor do SEA em 30 deles, a distância-α em 39 e simultaneamente em ambos os critérios em 28 modelos, como mostra a Tabela 5.3.

Os resultados típicos obtidos após a execução de cada função do método

Tabela 5.3: Numero de modelos (dos 41 modelos RM) com melhor desempenho após aplicar o método NAR-RM.

	Total	$\ell = 3$	$\ell = 4$	$\ell = 5$
Modelos RM	41	18	12	11
Melhorou SEA	30	14	7	9
Melhorou distância-α	39	17	12	10
Melhorou ambos	28	13	7	8

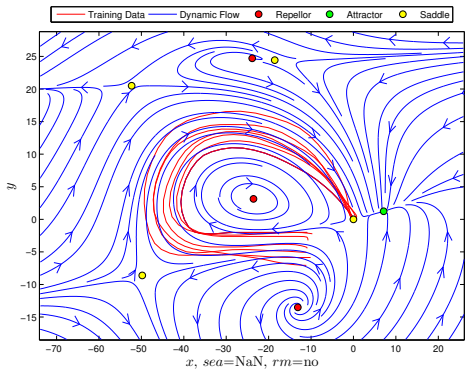
NAR-RM podem ser vistos na Figura 5.2, onde o modelo NAR obtido foi:

$$\begin{aligned} x(k) = \; & +0.983506\, x(k-1) + 0.096590\, y(k-1) \\ & -0.000078\, x(k-1)^3 + 0.005253\, y(k-1)^2 \\ & -0.000538\, y(k-1)^3 - 0.016513\, x(k-1)\; y(k-1) \\ & -0.000300\, x(k-1)^2\; y(k-1) - 0.004126\, x(k-1)^2 \end{aligned}$$

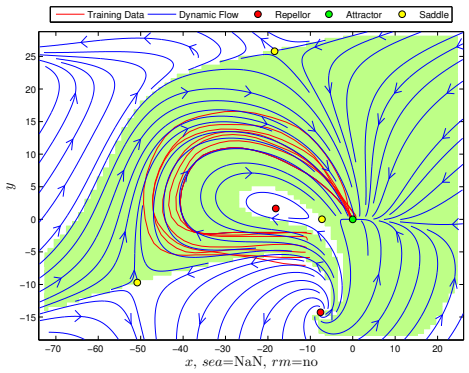
$$\begin{aligned} y(k) = \; & +0.779775\, y(k-1) - 0.000042\, x(k-1)^3 \\ & -0.015285\, x(k-1)\; y(k-1) - 0.002493\, y(k-1)^2 \\ & -0.000216\, x(k-1)^2\; y(k-1) - 0.004130\, x(k-1) \\ & -0.000102\, y(k-1)^3 - 0.001130\, x(k-1)^2 \\ & +0.000001\, x(k-1)\; y(k-1)^2\,. \end{aligned}$$

Para avaliar o efeito do grau de não linearidade nos modelos NAR-RM, foi elaborada a Tabela 5.4. Observe que o grau de não-linearidade $\ell = 5$ obteve o melhor SEA em 22 modelos, contra 7 de $\ell = 4$ e 1 de $\ell = 3$. Comparando o valor da distância-α, um comportamento oposto é observado em relação a ℓ, onde $\ell = 3$ apresentou a melhor distância-α em 18 padrões, contra $\ell = 4$ com 8 e $\ell = 5$ com 4. Então, utilizando os valores médios de SEA e distância-α, concluí-se que $\ell = 5$ é a melhor opção para a biblioteca LASA Handwriting Dataset.

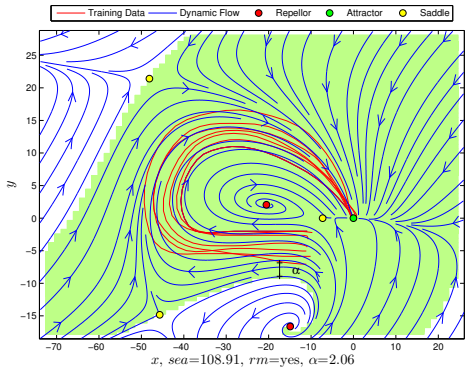
A Figura 5.3 apresenta os retratos de fase de 30 modelos com os melhores valores de SEA e que tem pelo menos 10 mm de distância-α, obtidos usando o NAR-RM. No lado esquerdo da Figura 5.3 são apresentadas as bacias de



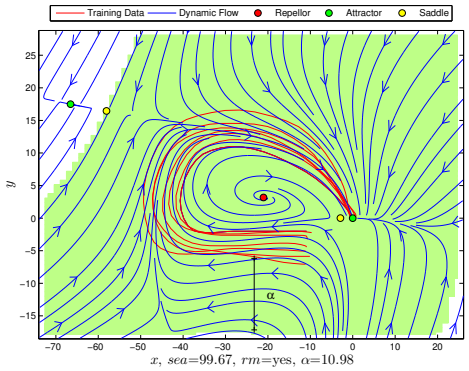
(a) Modelo obtido com o estimador MQ.



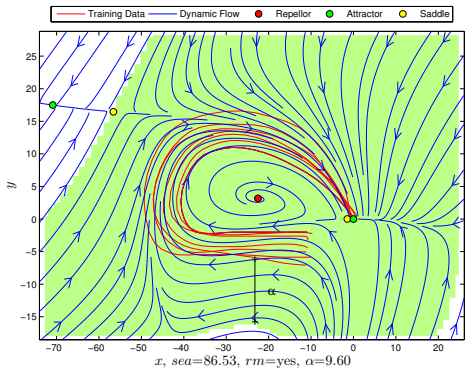
(b) Após a função `getTargetAttractor`.



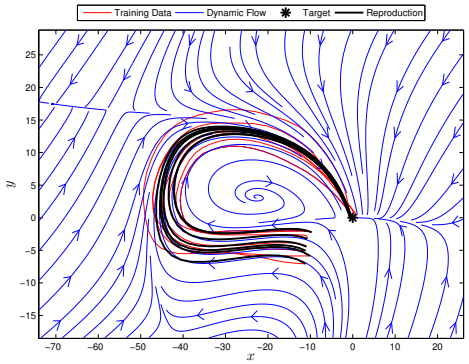
(c) Após as funções `getRM_Model` (1 e 2).



(d) Após a função `optimizeAlphaDistance`.



(e) Após a função `optimizeAccuracy`.



(f) Execução do modelo RM.

Figura 5.2: Execução do método NAR-RM. As figuras mostram o retrato da fase dos modelos identificados em azul, os pontos fixos em amarelo (sela), verde (atrator) e vermelho (repulsor) e a bacia de atração em verde. As trajetórias das demonstrações são mostradas como trajetórias vermelhas e os movimentos executados em preto. Todos os eixos são mostrados em milímetros.

atração dos alvos e no lado direito as trajetórias obtidas com a simulação dos modelos saindo dos pontos iniciais das demonstrações. Esses resultados

Tabela 5.4: Comparação entre os modelos NAR-RM obtidos.

	$\ell = 3$	$\ell = 4$	$\ell = 5$
Modelos com melhor SEA	1	7	22
Modelos com melhor distância-α	18	8	4
SEA médio dos modelos	195.14	120.52	101.57
Distância-α média dos modelos	10.24	7.01	7.27

são comparáveis (e melhores em vários casos) aos melhores resultados publicados na literatura, utilizando os mesmos dados (Khansari-Zadeh e Billard, 2011, 2014; Neumann e Steil, 2015; Lemme et al., 2013; Neumann et al., 2013). Uma comparação numérica com alguns desses métodos é realizada na próxima seção.

A generalização dos modelos obtidos é exemplificada na Figura 5.4, onde é mostrada a reprodução de quatro dos modelos da Figura 5.3, iniciando de pontos diferentes dos pontos iniciais das demonstrações.

5.1.1 Comparação com trabalhos anteriores

O desempenho do NAR-RM foi comparado com métodos do estado da arte usando o *benchmark*[3] para geração de movimentos de alcance para robôs (Lemme et al., 2015).

O *benchmark* considera quatro cenários, que incluem perturbações típicas que ocorrem no movimento do robô: (i) generalização, que é baseada na inicialização do modelo em diferentes pontos de partida; (ii) impulso discreto, onde o robô sofre um deslocamento repentino durante o movimento; (iii) impulso contínuo, onde o movimento é continuamente perturbado durante uma janela de tempo; e (iv) variação da posição do alvo, quando a posição alvo é modificada durante o movimento. Essas perturbações são ilustradas

[3]O benchmark está disponível em: https://www.amarsi-project.eu/benchmark-framework

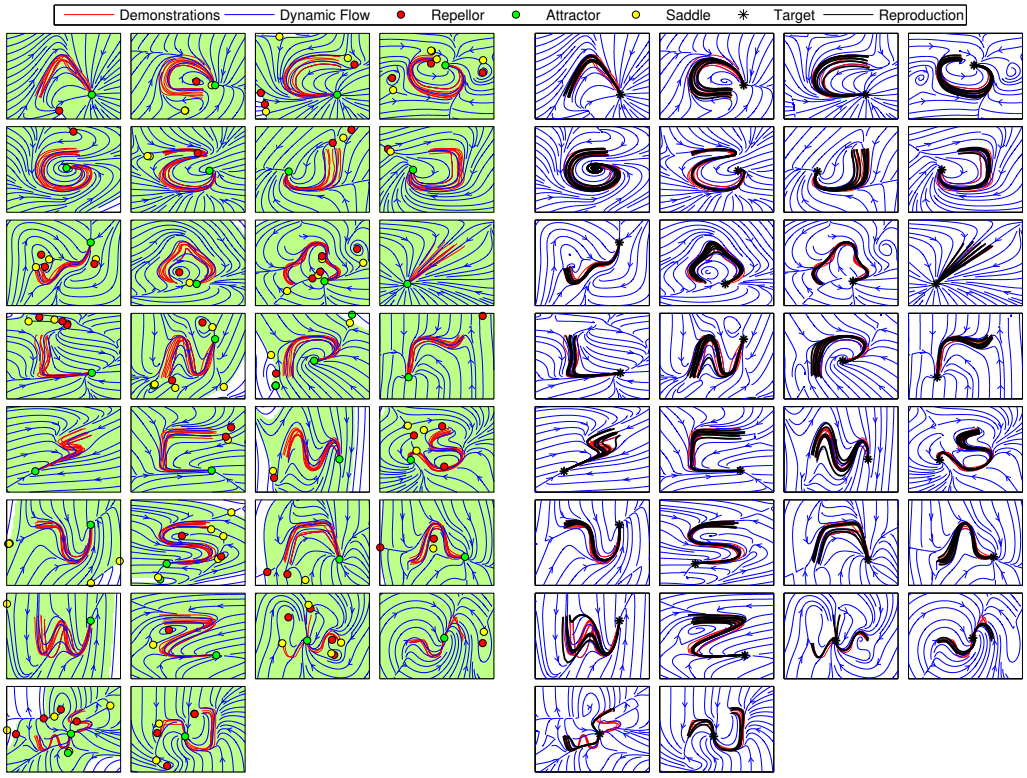
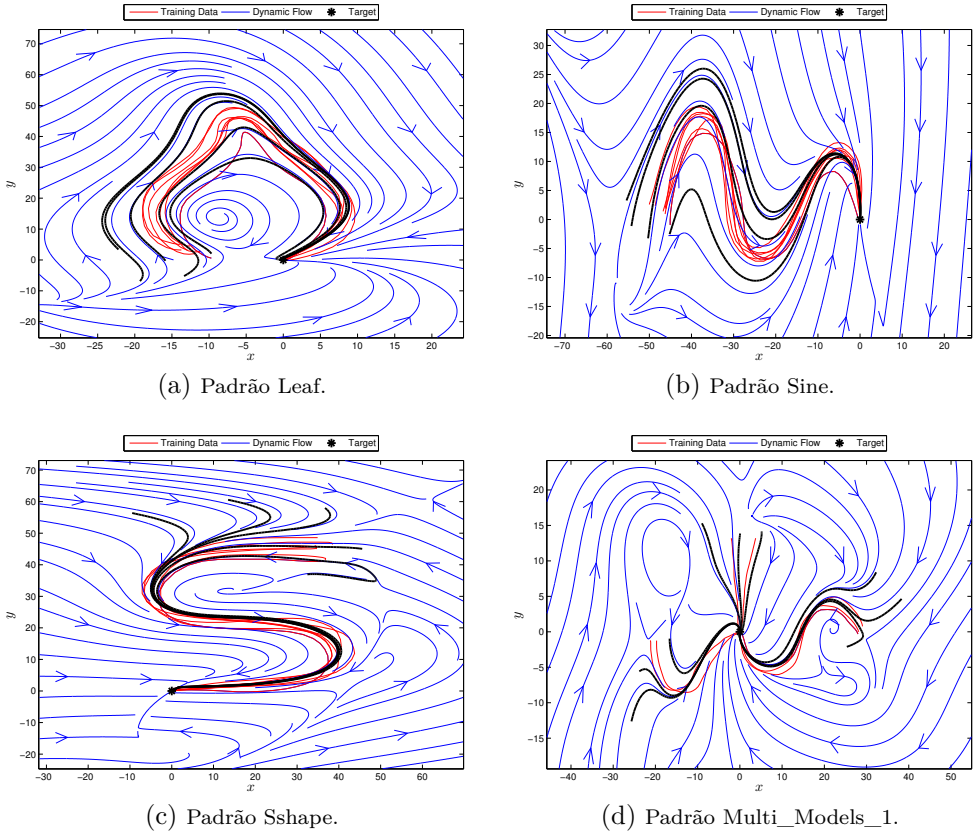


Figura 5.3: Os 30 modelos com os melhores valores de SEA e que tem pelo menos 10 mm de distância-α obtidos usando o NAR-RM. No lado esquerdo, a bacia de atração do alvo, os pontos fixos e as demonstrações são apresentadas. Do lado direito estão as reproduções obtidas pelos modelos com os mesmos pontos iniciais das demonstrações.

na (Figura 5.5).

Em cada cenário, algumas métricas são usadas para avaliar a evolução e o desempenho dos métodos. As métricas são divididas em nível geométrico, nível cinemático, nível geométrico & cinemático e modelo de software. No nível geométrico, a precisão do modelo em reproduzir as demonstrações é avaliada. No nível cinemático, o foco é no desempenho relacionado aos perfis de velocidade. Finalmente, nos níveis geométricos & cinemáticos, as leis do movimento, que são relevantes para o controlador de baixo nível, são avaliadas. A Tabela 5.5 mostra todas as métricas aplicadas no *benchmark*. Mais detalhes podem ser encontrados em (Lemme et al., 2015).



(a) Padrão Leaf.

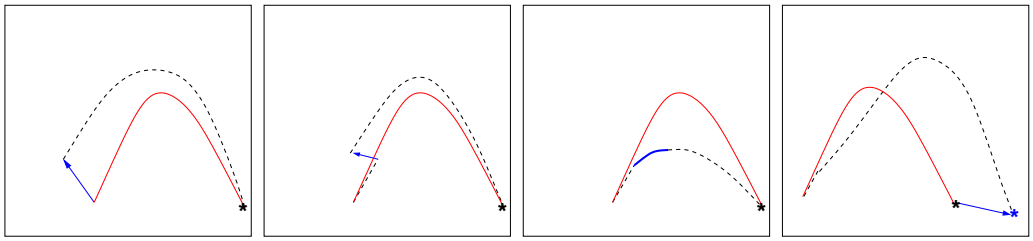
(b) Padrão Sine.

(c) Padrão Sshape.

(d) Padrão Multi_Models_1.

Figura 5.4: Reprodução de 4 modelos da Figura 5.3 iniciando de pontos diferentes dos pontos iniciais das demonstrações. As trajetórias das demonstrações são mostradas em vermelho e os movimentos reproduzidos em preto. Os eixos são mostrados em milímetros.

O benchmark foi aplicado usando a biblioteca LASA Handwriting Dataset. O método NAR-RM foi comparado com três métodos autônomos avaliados no benchmark: CLF-DM (Khansari-Zadeh e Billard, 2014); NIVF usando funções candidatas quadráticas de Lyapunov (NIVF-QLG) (Lemme et al., 2013) e com funções de Lyapunov aprendidas das demonstrações (NIVF-LL) (Neumann et al., 2013). Cada método foi implementado por seus próprios autores. O NAR-RM foi representado nos experimentos pelos modelos obtidos na Seção 5.1. Para cada padrão de movimento foi escolhido, entre os 3 modelos obtidos usando $\ell = 3, 4, 5$, o que apresentou o melhor valor de SEA e que tenha a distância-$\alpha \geq 10\,\text{mm}$ (veja Figura 5.3).



(a) Mudar a posição inicial. (b) Impulso discreto. (c) Impulso contínuo. (d) variação da posição do alvo.

Figura 5.5: Quatro diferentes cenários de referência usados no benchmark. A trajetória vermelha (linha contínua) é o movimento demonstrado e a trajetória preta (linha tracejada) é o movimento perturbado gerado na simulação. A seta azul indica a direção e a amplitude da perturbação aplicada.

Tabela 5.5: Diferentes métricas usadas pelo *benchmark* (Lemme et al., 2015).

Categoria	Propriedade	Medida	Rótulo usado nas figuras
Geométrico	caminho	RMSE	trajectory-position-error
	erro de posição do alvo	L^2-norm	target-position-error
Cinemático	perfil de velocidade	RMSE	trajectory-velocity-error
	perfil de velocidade	R^2	R2-speed
	erro da velocidade no alvo	L^2-norm	target-velocity-error
	duração do movimento	see Equation(6) in (Lemme et al., 2015)	normalizedFinalTime
Geométrico & Cinemático	trajetória	R^2	R2
	lei de potência	ΔR^2	PL-R2
	jerk mínimo	RMSD	mean-jerk
Modelo de software	tempo de processamento	ms	MeanComputationTime

O benchmark foi executado usando os mesmos parâmetros de (Lemme et al., 2015). Esses parâmetros definem 100 execuções para cada conjunto de 7 demonstrações dos 30 padrões de movimentos, resultando em 21.000 ($100 \times 30 \times 7$) trajetórias simuladas para cada um dos quatro cenários do benchmark. Isso resultou em 84.000 simulações para cada método. As métricas foram calculadas para cada trajetória simulada e os resultados obtidos nos 4 benchmarks foram compilados gerando um resultado global para cada métrica.

No nível geométrico, o benchmark usa a métrica trajectory-position-error que mede a raiz quadrada do erro quadrático médio (RMSE) entre a trajetória reproduzida e a trajetória da demonstração. A Figura 5.6 mostra que o NAR-RM obteve o segundo melhor valor da mediana, ficando 5% atrás do CLF-DM (veja Tabela 5.6). Outra métrica geométrica usada pelo benchmark

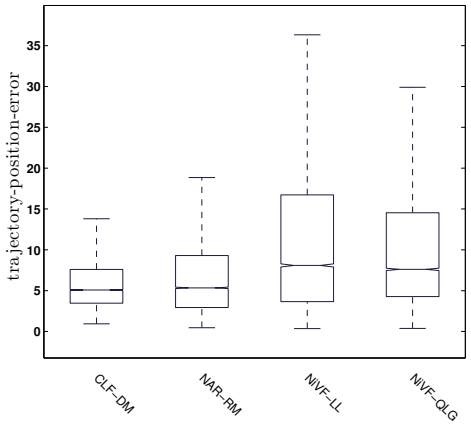
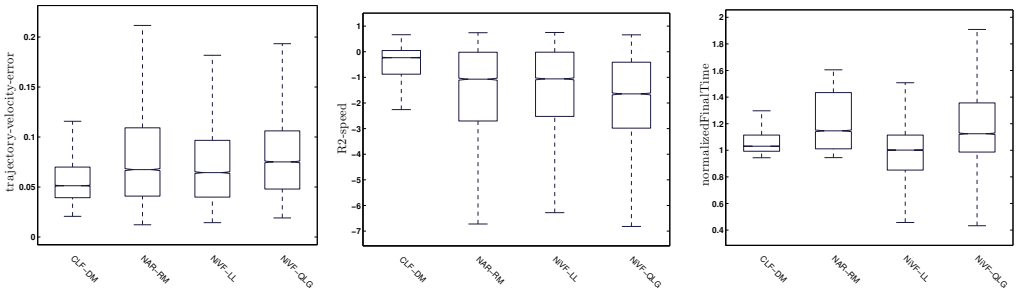


Figura 5.6: O desempenho do NAR-RM com os outros métodos para métrica trajectory-position-error do nível cinemático. Os resultados da métrica para os quatro benchmarks foram compilados e são mostrados juntos.

é a target-position-error, que mede a distância do último ponto registrado até o alvo. Como os valores obtidos para essa métrica nos quatro métodos testados tinham o valor da mediana menor ou igual a 1 mm, que é um *threshold* utilizado em (Lemme et al., 2015) para interromper alguns dos métodos, a métrica foi desconsiderada.

Quatro métricas cinemáticas foram avaliadas. A primeira é trajectory-velocity-error, que mede a raiz quadrada do erro quadrático médio (RMSE) entre a velocidade da reprodução e a velocidade da demonstração. Os resultados para esta métrica são mostrados na Figura 5.7(a). Observa-se que o CLF-DM apresentou o melhor valor da mediana, seguido pelo NIFV-LL. O NAR-RM obteve o terceiro melhor valor, próximo do NIVF-LL. A segunda métrica, chamada R2-speed, compara os perfis de velocidade das trajetórias demonstradas e reproduzidas com duração padronizada, usando o coeficiente de determinação, também chamado de R^2. A Figura 5.7(b) mostra os resultados da métrica R2-speed, onde o CLF-DM obteve o melhor valor de mediana, seguido por NIVF-LL e NAR-RM. Novamente, o valor da mediana de NAR-RM e NIVF-LL foram próximos. NormalizedFinalTime é a terceira



(a) Métrica trajectory-velocity-error.

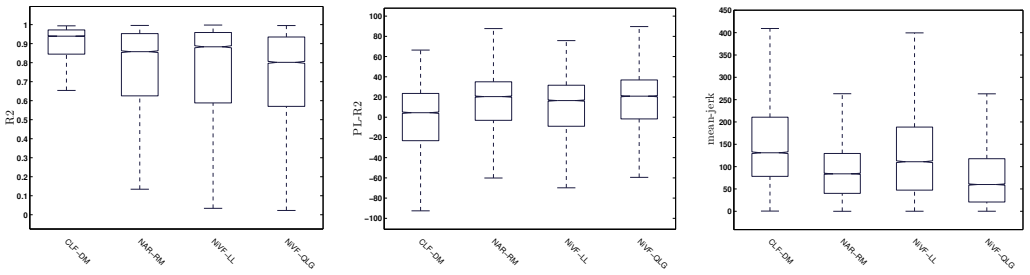
(b) Métrica R2-speed.

(c) Métrica normalizedFinalTime metric.

Figura 5.7: O desempenho do NAR-RM com os outros métodos para métricas de nível cinemático: trajectory-speed-errors, R2-speed, normalizedFinalTime. Os resultados de cada métrica para os quatro benchmarks foram compilados e são mostrados juntos.

métrica, ela compara a duração da reprodução da trajetória do movimento com a trajetória da demonstração correspondente. Nesta métrica, o melhor valor é o mais próximo de 1. Na Figura 5.7(c) é possível observar que o NIVF-LL apresentou o melhor valor da mediana e o NAR-RM o pior. Mesmo com o pior resultado, o valor obtido pelo NAR-RM foi apenas 14% pior que o melhor resultado (veja Tabela 5.6). A última métrica do nível cinemático, target-velocity-error mede a velocidade final ao atingir o alvo usando a norma L2. Os resultados obtidos para todos os métodos no benchmark tiveram o valor da mediana zero.

Em relação às leis de controle, isto é, nível geométrico & cinemático juntos, as métricas R2, PL-R2 e mean-jerk foram utilizadas. A métrica R2 foi usada para avaliar a precisão da trajetória, onde a forma da trajetória e o perfil de velocidade são avaliados simultaneamente. A Figura 5.8(a) mostra os resultados de R2, onde o CLF-DM obteve o melhor resultado, seguido por NIVF-LL e NAR-RM. Os valores das medianas obtidas pelos três métodos foram próximos (ver Tabela 5.6). PL-R2 é a lei de potência que usa a medida R2 para prever o perfil de velocidade das trajetórias. O resultado da métrica pode ser visto na Figura 5.8(b), onde o NIVF-QLG obteve o melhor



(a) Métrica R2. (b) Métrica PL-R2. (c) Métrica Mean-jerk.

Figura 5.8: O desempenho do NAR-RM com os outros métodos para métricas de nível cinemático & geométrico: R2, PL-R2 and mean-jerk. Os resultados de cada métrica para os quatro benchmarks foram compilados e são mostrados juntos.

desempenho seguido pelo NAR-RM. A última métrica, mean-jerk, usa a raiz da derivada quadrada média (RMSD) de uma trajetória e os resultados são mostrados na Figura 5.8(c). Para essa métrica, o NIVF-QLG apresentou o melhor resultado seguido pelo NAR-RM.

A Tabela 5.6 mostra os valores das medianas e a porcentagem em que cada valor é pior em relação ao melhor valor encontrado, representado pela notação w%, para os níveis geométrico, cinemático e cinemático & geométrico. Analisando a Tabela 5.6, vimos que o CLF-DM é o melhor método ao considerar métricas geométricas e cinemáticas, mas apresentou o pior desempenho em duas das três métricas do conjunto cinemático & geométrico. Os métodos NIVF-LL e NIVF-QLG mostraram baixo desempenho na métrica geométrica e um comportamento médio nas métricas cinemáticas. Quanto às métricas cinemáticas & geométricas, o NIVF-LL obteve um resultado médio, enquanto o NIFV-QLG foi o melhor método. Avaliando o NAR-RM, o método obteve um excelente resultado na métrica geométrica, um desempenho médio na métrica cinemática e um bom resultado nas métricas geométricas & cinemáticas. Analisando todo esse contexto, concluí-se que o NAR-RM, mesmo não sendo o melhor método em cada métrica individual avaliada, pode ser considerado a melhor alternativa quando se quer uma boa precisão nas

Tabela 5.6: Resultados das diferentes métricas usadas pelo benchmark (Lemme et al., 2015). A porcentagem em que cada valor é pior em relação ao melhor valor encontrado é representada na tabela pela notação w%. Os melhores valores são mostrados em negrito.

	Geométrico		Cinemático						Cinemático & Geométrico					
	trajectory-position-error		trajectory-velocity-error		R2-speed		normalized-FinalTime		R2		PL-R2		mean-jerk	
Method	mediana	w%	mediana	w%	mediana	w%	mediana	w%	mediana	w%	mediana	w%	mediana	w%
CLF-DM	**5,08**	**0**	**0,051**	**0**	**-0,23**	**0**	1,03	3	**0,94**	**0**	4,46	79	131,06	119
NAR-RM	5,34	5	0,067	31	-1,07	360	1,15	14	0,86	9	20,39	2	83,93	40
NIVF-LL	8,09	59	0,064	25	-1,06	355	**1,00**	**0**	0,88	6	16,48	21	111,02	85
NIVF-QLG	7,61	50	0,075	47	-1,65	606	1,12	12	0.80	15	**20,81**	**0**	**59,92**	**0**

trajetórias aprendidas (nível geométrico), e, ao mesmo tempo, necessita-se de alguma garantia de que as trajetórias possam ser seguidas por um controlador de baixo nível (nível cinemático & geométrico). Essas características não podem ser alcançadas simultaneamente pelo CLF-DM, que é muito bom em métricas geométricas e cinemáticas, mas exige muito do controlador, e o NIVF-QLG que demanda baixo esforço do controlador mas apresenta grandes erros geométricos e cinemáticos.

A métrica MeanComputationalTime estima a complexidade computacional do algoritmo gerador de movimento em fornecer o próximo estado desejado, com base no estado atual do movimento. Os métodos NIVF-LL e NIVF-QLG obtiveram o menor tempo médio com valor de 0,07 ms, CLF-DM com 0,23 ms e NAR-RM com 0,54 ms. O método NAR-RM apresentou o pior desempenho entre os métodos avaliados, o que provavelmente aconteceu porque, como o método não é global, é necessário fazer algumas verificações extras para avaliar se as trajetórias estão dentro da bacia de atração do alvo. Caso alguma trajetória não esteja dentro da bacia de atração do alvo, ela é paralisada, ou seja, o último ponto obtido ao sair da bacia é retornado como o próximo ponto na simulação do *benchmark*. No entanto, o tempo de cálculo necessário ainda é pequeno, principalmente se implementado em uma linguagem de programação mais eficiente que o MATLAB, indicando que o método pode ser usado em tempo real, como será mostrado ainda neste capítulo.

5.1.2 Outras análises

Até o momento, o tempo gasto para obter os modelos RM pelo método NAR-RM não foi discutido. Como os demais métodos apresentados neste trabalho, a fase de aprendizado é realizada *off-line*, sendo considerado mais importante o tempo computacional necessário para se obter uma nova posição na fase de execução do sistema. Mas, pensando em uma aplicação futura onde um usuário irá reprogramar o seu robô, o tempo de computação na fase de aprendizado deve ser levado em consideração.

O NAR-RM é um método iterativo em que vários modelos são estimados e avaliados a cada iteração do sistema. O tempo computacional em cada iteração depende do número de pontos fixos impostos, onde cada imposição gera um modelo, e do tempo gasto na estimação e avaliação de cada modelo. O tempo computacional para estimar os modelos está relacionado ao tamanho da matriz de regressores, ou seja, no nosso caso, depende do número de pontos que é utilizado para representar cada trajetória e do número de regressores utilizados nas funções do modelo. Na avaliação dos modelos, o tempo computacional está atrelado ao tempo do cálculo dos pontos fixos e da estimação da bacia de atração. Logo, o tempo computacional do NAR-RM depende do número de modelos avaliados em cada iteração, sendo que este valor está atrelado ao número de pontos fixos existentes no modelo antes da imposição dos pontos fixos, e ao tempo gasto na estimação e avaliação de cada modelo gerado.

Para analisar o impacto do tempo gasto na fase de aprendizado, outros experimentos foram realizados. O primeiro foi utilizando o método NAR-RM variando o número de pontos utilizados para representar cada trajetória das demonstrações. Para isso, foram aplicadas as taxas de decimação 1, 5, 10 e 20, obtendo trajetórias com 1.001, 201, 101 e 21 pontos respectivamente,

Tabela 5.7: Resultado da mediana do tempo gasto em minutos (t) e a mediana do número de modelos avaliados para os 30 padrões com as trajetórias sendo representadas com 21, 101, 201 e 1001 pontos.

Nº de pontos	21		101		201		1.001	
Medidas	t(min)	modelos	t(min)	modelos	t(min)	modelos	t(min)	modelos
$\ell = 3$	4,3	372,5	6,2	373,0	9,4	297,5	52,1	364,0
$\ell = 4$	8,8	638,5	14,4	692,5	21,3	757,5	95,03	903,0
$\ell = 5$	13,2	810,5	18,07	949,5	24,9	697,5	99,48	889,5

devido à adição do ponto (0,0) no final de cada trajetória. A distância utilizada na imposição foi $d = 4$, a matriz M foi definida com 70 linhas e 70 colunas, o número de iterações em cada função com o valor $MaxIt = 10$ e os demais parâmetros foram os mesmos utilizados na seção anterior.

A Tabela 5.7 apresenta a mediana do tempo gasto em minutos e a mediana do número de modelos avaliados para os 30 padrões com os valores de não linearidade $\ell = 3, 4, 5$ e com as trajetórias sendo representadas com 21, 101, 201 e 1001 pontos. Os resultados mostram que o aumento da entrada de dados impacta de forma significativa no aumento do tempo para se processar um padrão usando o método NAR-RM, como mostrado para $\ell = 3$, onde um padrão de movimento é calculado em 4,3 minutos com as trajetórias contendo 21 pontos e 52,1 minutos com 1001 pontos. Também verifica-se que o aumento do grau de não linearidade impacta no aumento do tempo. Isto ocorre devido ao aumento do número de pontos fixos nos modelos inciais com graus de não linearidade maiores, onde $\ell = 3, 4, 5$ apresentam 94, 143 e 198 pontos fixos. O aumento do número de pontos fixos, aumenta o número de modelos avaliados, como se pode ver na Tabela 5.7.

Um outro aspecto importante a ser verificado é se o aumento do número de pontos das trajetórias das demonstrações afeta as métricas utilizadas no *benchmark* (Lemme et al., 2015) aplicado na seção anterior. A escolha dos modelos que representaram as trajetórias com 21, 101, 201 e 1001 pontos, seguiram os mesmos critérios utilizados na seção anterior, ou seja, modelos en-

Tabela 5.8: Resultados das diferentes métricas usadas pelo benchmark (Lemme et al., 2015). A porcentagem em que cada valor é pior em relação ao melhor valor encontrado é representada na tabela pela notação w%. Os melhores valores são mostrados em negrito.

	Geométrico		Cinemático						Cinemático & Geométrico					
	trajectory-position-error		trajectory-velocity-error		R2-speed		normalized-FinalTime		R2		PL-R2		mean-jerk	
Entrada	mediana	% w	mediana	% w	mediana	% w	mediana	% w	mediana	% w	mediana	% w	mediana	% w
21	5,40	13	0,053	10	-0,29	142	**0,99**	**0**	0,93	2	**24,28**	**0**	**78,70**	**0**
101	4,94	3	0,051	6	-0,20	67	**0,99**	**0**	0,93	1	17,96	26	107,80	37
201	**4,78**	**0**	**0,048**	**0**	**-0,12**	**0**	**0,99**	**0**	**0,94**	**0**	16,42	32	101,70	29
1001	5,13	7	0,050	4	-0,20	67	**0,99**	**0**	0,93	1	18,65	23	102,00	30

tre $\ell = 3, 4, 5$ que possuem o melhor SEA e que tenha a distância-$\alpha \geq 10\,\text{mm}$.

Os resultados obtidos foram compilados na Tabela 5.8. Analisando os resultados da métrica trajectory-velocity-error do nível geométrico, pode-se concluir que a variação do tamanho da entrada afeta pouco a precisão em reproduzir as trajetórias aprendidas. No nível cinemático, somente a métrica R2-speed apresenta uma variação considerável nos resultados, onde os modelos com trajetórias contendo 201 pontos apresentaram o melhor resultado. Nas métricas do nível cinemático & geométrico, os modelos com trajetórias contendo 21 pontos apresentaram o desempenho superior nas métricas PL-R2 e mean-jerk e similar aos demais na métrica R2, destacando-os neste nível. Avaliando todos os resultados das Tabelas 5.7 e 5.8 conclui-se que a utilização de trajetórias contendo 21 pontos é a melhor opção, pois apresenta o menor tempo na fase de aprendizado dos padrões de movimento e por ter sido obtido um bom resultado nos níveis geométrico e cinemático e se destacado no nível cinemático & geométrico do *benchmark*.

Uma outra possibilidade para tentar reduzir o tempo computacional na fase de aprendizado dos movimentos é diminuir o número de modelos avaliados a cada iteração. Seguindo esta linha de raciocínio, foi implementada uma variação no método NAR-RM denominada NAR-RM2 (veja o Algoritmo 4). Diferente do método original, que a cada iteração de suas funções verifica todos os possíveis modelos gerados pelas imposições de pontos fixos ao redor

de todos os pontos de equilíbrio do modelo, o método NAR-RM2 monta uma lista de prioridade com os pontos fixos existentes no modelo, para definir a ordem em que eles serão avaliados (linha 8). A prioridade da lista deve ser definida de acordo com a característica que se quer obter ou melhorar do modelo. Um exemplo seria priorizar os pontos fixos sela quando se está ampliando a bacia de atração. Então, com a lista definida, o método aplica as imposições utilizando o primeiro ponto fixo da lista e avalia os modelos gerados. Caso o melhor modelo avaliado obtenha ou melhore a característica procurada pela função, o procedimento vai para a próxima iteração, caso contrário, outro ponto da lista é escolhido até que todos sejam avaliados (linhas 10–23). Se nenhum modelo avaliado melhorou a característica procurada, a distância d é modificada (linhas 24–26). Exemplificando, suponha que tem-se um modelo com 4 pontos fixos, onde um deles é o alvo, e chama-se a função `optimizeAlphaDistance` para ampliar a distância-α do modelo. Em uma iteração do método original, a função verificaria possíveis 24 (3×8) modelos após a imposição dos pontos fixos ($m = 8$). No método NAR-RM2, monta-se uma lista de prioridade com os 3 pontos fixos, por exemplo priorizando as selas, e verifica-se inicialmente os 8 possíveis modelos para o primeiro ponto fixo da lista. Caso o melhor modelo avaliado entre os oito tenha ampliado a distância-α, a função vai para a próxima iteração, caso contrário, o segundo ponto fixo da lista de prioridade é avaliado.

Na implementação do NAR-RM2 testada neste trabalho, foi definido para as funções `getRM_Model1`, `getRM_Model2` e `optimizeAlphaDistance`, que buscam ampliar a bacia de atração do alvo, o seguinte critério para montar a lista de prioridade que prioriza os pontos fixos selas: dado um modelo com seus pontos fixos já calculados, (1) separe todos os pontos fixos em dois grupos, um contendo as selas e o outro contendo os demais pontos fixos, exceto

Algoritmo 4: Algoritmo geral para o NAR-RM2

```
Input: Y_(n), ℓ, MaxIt, d, m, Model
begin
    if hasCharacteristic(Model) then
        return Model;
    end
    Best ← Model;    i ← 0 ;
    while i < MaxIt do
        FP ← getFixedPoint(Model);
        Build a priority queue Q using FP;
        improved ← False;
        while Q ≠ ∅ and improved = False do
            P ← dequeue(Q);
            M_aux ← imposeFP(Y_(n), ℓ, P, d, m, Model);
            if acceptable(M_aux) then
                if hasCharacteristic(M_aux) then
                    return M_aux;
                end
                Best ← bestModel(Best,M_aux);
                if Best = M_aux then
                    Model ← M_aux;
                    improved ← True;
                end
            end
        end
        if improved = False then
            change the distance d;
        end
        i ← i + 1 ;
    end
    return Best;
end
```

o alvo, (2) como todas as selas possuem a mesma prioridade e não se está utilizando outro critério para diferenciá-las[4], aplique um sorteio no grupo das sela para que todas sejam adicionadas na lista de prioridade, (3) utilize

[4]Um possível critério para diferenciar as selas poderia ser um histórico da última vez que o ponto fixo foi escolhido durante a execução da função, onde os pontos escolhidos mais recentemente tenham uma maior prioridade.

Tabela 5.9: Resultados de 3 execuções do método NAR-RM2 e uma do NAR-RM. São apresentados as medianas dos valores de SEA, distância-α, do tempo gasto em minutos (t) e do números de modelos avaliados para os 30 padrões de movimentos para os valores de $\ell = 3, 4, 5$

	$\ell = 3$				$\ell = 4$				$\ell = 5$			
	SEA	Dist-α	t(min)	Modelos	SEA	Dist-α	t(min)	Modelos	SEA	Dist-α	t(min)	Modelos
NAR-RM2 (1)	144,1	20,7	3,0	226,5	96,6	8,3	4,6	383,5	99,7	11,0	8,2	486,5
NAR-RM2 (2)	132,0	19,0	3,1	224,0	96,0	10,1	5,0	396,5	85,2	7,2	8,1	483,5
NAR-RM2 (3)	139,6	21,2	2,9	211,5	92,3	8,8	4,9	392,5	92,9	16,4	8,2	449,5
NAR-RM	141,1	24,1	4,3	372,5	96,0	8,4	8,8	638,5	87,2	15,2	13,2	810,5

o mesmo processo de sorteio para o segundo grupo. No fim do processo, os pontos fixos selas estarão no início da lista de prioridade. Para a função `optimizeAccuracy`, a lista de prioridade é montada de forma aleatória utilizando todos os pontos fixos do modelo, exceto o alvo, pois a mudança na posição de qualquer ponto fixo pode afetar o valor do SEA.

O método NAR-RM2 foi executado usando os mesmos parâmetros utilizados nesta seção utilizando trajetórias contendo 21 pontos. Como o método possui um processo aleatório e visando verificar a capacidade de obter bons modelos, 3 execuções foram feitas para os valores de $\ell = 3, 4, 5$. Os resultados dos experimentos foram coletados e são comparados ao método NAR-RM original, aplicando os mesmos parâmetros. A Tabela 5.9 apresenta as medianas dos valores de SEA, distância-α, do tempo gasto em minutos e do números de modelos analisados para os 30 padrões de movimentos utilizando os valores de $\ell = 3, 4, 5$.

Analisando a Tabela 5.9, observa-se que o NAR-RM2 obteve resultados similares aos valores de SEA e distância-α, avaliando menos modelos e com um tempo computacional inferior, tornando-se uma alternativa promissora ao método NAR-RM original.

5.2 Aplicação a um robô móvel

O NAR-RM foi usado para ensinar um robô móvel terrestre não holonômico a alcançar um alvo a partir de várias posições iniciais. Possíveis aplicações incluem a manipulação de objetos por deslocamento, onde um robô precisa alcançar um ponto específico do objeto seguindo um conjunto restrito de trajetórias que moveria o objeto na direção correta.

Assumindo que um robô é representado por uma configuração exata $\mathbf{q}$, o método proposto pode ser diretamente aplicado para controlar, por exemplo, um manipulador robótico com n graus de liberdade ou qualquer outro robô holonômico (Khansari-Zadeh e Billard, 2011; Neumann et al., 2013; Mühlig et al., 2009; Schaal et al., 2002). No caso de robôs não holonômicos, ou seja, com restrições de velocidades, um controlador de baixo nível baseado em linearização por realimentação de estados pode ser utilizado para que o robô consiga seguir um campo vetorial (Slotine e Li, 1991; Gonçalves et al., 2010; de Lima e Pereira, 2013).

Nesta seção assume-se um robô terrestre com restrição não holonômica e representado pelo seguinte modelo:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \text{sen}(\theta) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v \\ w \end{bmatrix}, \tag{5.1}$$

em que x, y são as coordenadas do robô no plano, θ é a sua orientação, v e w são as entradas da velocidade linear e angular do robô, respectivamente.

A linearização por realimentação de estados do modelo da Equação (5.1) é impossível, pois a velocidade angular não aparece explicitamente nas equações de $\dot{x}$ e $\dot{y}$ (Yun e Yamamoto, 1992). Para contornar este problema, so-

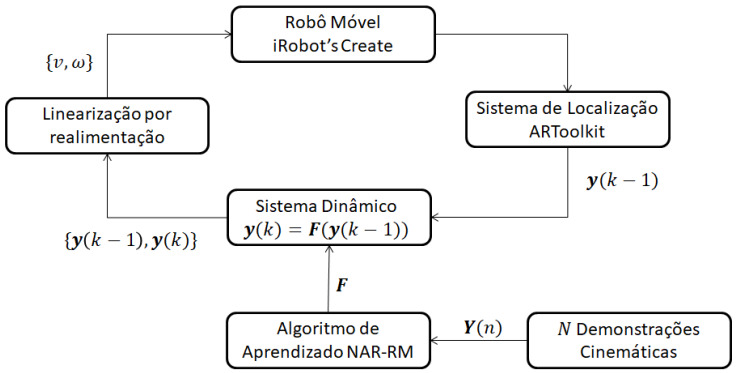


Figura 5.9: Esquema da arquitetura utilizada nos experimentos com o robô móvel.

mente a posição do robô $[x, y]^T$ é controlada utilizando um ponto d a frente do robô, onde o sistema de saída é definido como $[x_d, y_d]^T = [x + d\cos(\theta), y + d\,\text{sen}(\theta)]^T$.

Nos experimentos desta seção, foi utilizado um robô iRobot's Create equipado com um pequeno laptop e localizado por um conjunto de câmeras externas e o software ARToolkit[5]. A arquitetura utilizada pode ser vista na Figura 5.9.

Na primeira parte do experimento o robô foi controlado por um humano usando um controle com velocidade baixa e constante, enquanto os dados de posição $\mathbf{y} = [x, y]^{\mathrm{T}}$ eram salvos a uma taxa de 10 Hz. Após a aquisição dos dados, as trajetórias foram tratadas para que todas tivessem como alvo a origem. Em seguida o método NAR-RM é aplicado para se obter um modelo de movimento de alcance (RM). Depois que o modelo foi aprendido, um controlador baseado em linearização por realimentação foi utilizado para rastrear o campo vetorial resultante (d'Andréa Novel et al., 1995). Basicamente, usando o modelo NAR $F[x,y]$, é calculado um vetor $\mathbf{u} = F[x, y] - [x, y]^T$, para cada posição do robô $[x,y]$. Se a norma desse vetor for alterada para a velocidade desejada do robô, a sua velocidade linear e angular são definidas

[5]Disponível em: https://github.com/artoolkit

como:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \operatorname{sen}(\theta) \\ \frac{-\operatorname{sen}(\theta)}{d} & \frac{\cos(\theta)}{d} \end{bmatrix} \mathbf{u}\,, \tag{5.2}$$

onde d é a distância de um ponto de controle na frente do robô e θ é a orientação do robô.

Dois experimentos foram realizados. O primeiro experimento representa um cenário onde o robô deve atingir um ponto A no espaço a partir de vários pontos localizados a esquerda de A. Para o segundo experimento um movimento com formato de gancho foi escolhido. A Figura 5.10 mostra o robô durante o experimento. Um vídeo do experimentos pode ser encontrado em https://goo.gl/AqMLAH.

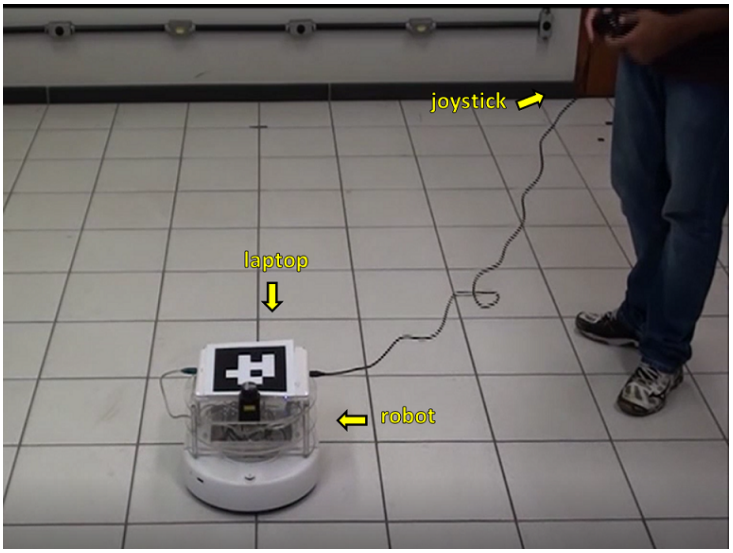
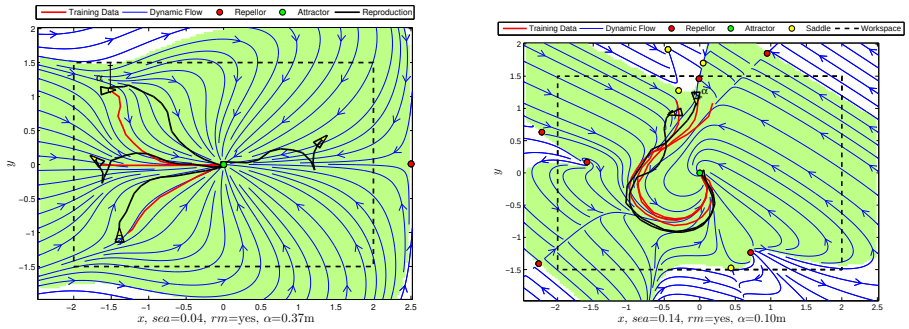


Figura 5.10: Robô móvel.

A Figura 5.11(a) mostra as trajetórias executadas pelo robô no retrato de fase do modelo estimado com o grau de não linearidade $\ell = 3$ para o primeiro experimento e a Figura 5.11(b) as trajetórias do segundo experimento cujo modelo foi obtido com $\ell = 5$. A orientação do robô é representada por um triângulo onde a frente do robô é identificada pela cor amarela. A linha



(a) Experimento 1. (b) Experimento 2.

Figura 5.11: Retrato de fase do modelo e as reproduções realizadas pelo robô móvel.

tracejada representa o espaço de trabalho do robô. Os resultados mostraram que o método NAR-RM pode ser aplicado para ensinar um robô móvel utilizando demonstrações.

5.3 Aplicação a um manipulador

O método NAR-RM também foi aplicado para ensinar movimentos de alcance a um manipulador robótico. Nestes experimentos, além de considerar trajetórias planares, também foram utilizadas trajetórias 3D representadas pela posição do atuador no espaço cartesiano, isto é, $\mathbf{y} = [x, y, z]^{\mathrm{T}}$.

Nos experimentos foi utilizado um manipulador Comau Smart Six equipado com um sensor que permite ao usuário guiar o robô em seu espaço de trabalho com as mãos (Figura 5.12). Os dados dos experimentos foram coletados a uma taxa de 10 Hz. A arquitetura utilizada nos experimentos pode ser vista na Figura 5.13, onde as configurações $\mathbf{q}$ do manipulador foram obtidas utilizando a cinemática inversa e as posições $\mathbf{y}$ aplicando a cinemática direta fornecidas pelo próprio sistema do manipulador.

No primeiro experimento, o manipulador é ensinado a descrever um padrão de trajetórias em um quadro branco. Na primeira parte do experimento



Figura 5.12: Manipulador industrial.

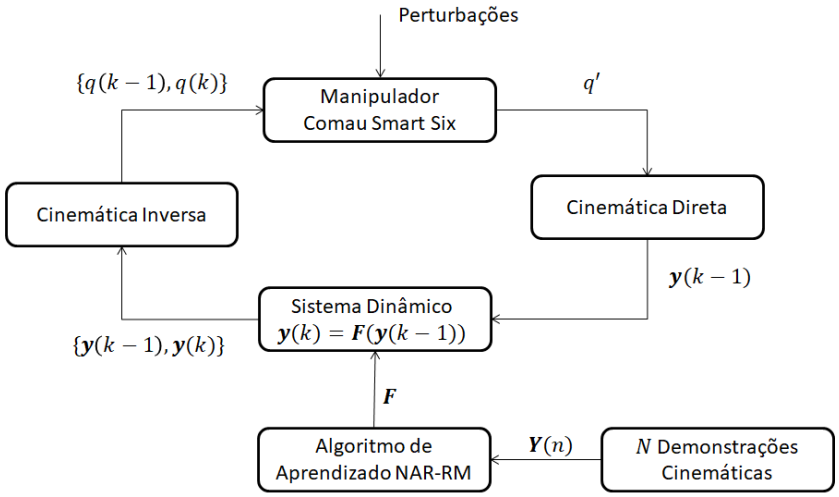
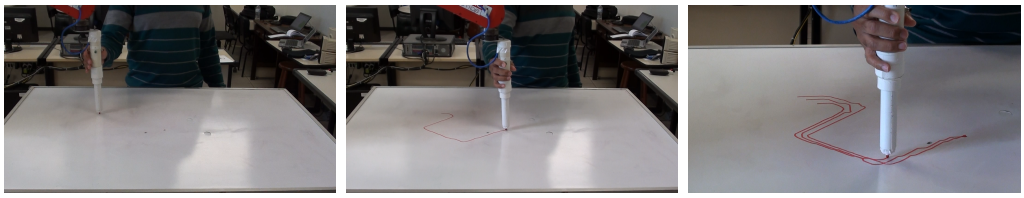


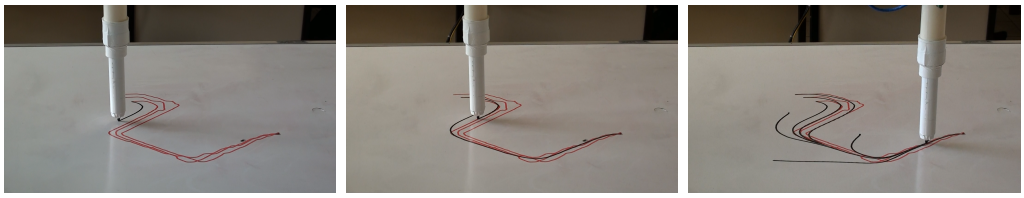
Figura 5.13: Esquema da arquitetura utilizada nos experimentos com o manipulador.

o robô foi controlado por um humano para reproduzir 3 trajetórias similares (veja Figura 5.14(a)). Cada trajetória foi representada por 51 pontos discretos e as posições foram obtidas utilizando a cinemática direta implementada pelo sistema do robô. Em seguida é aplicado o método NAR-RM com o grau de não linearidade $\ell = 4$ e por fim, o robô executa as trajetórias aprendidas saindo de pontos distintos do espaço de trabalho (veja Figura 5.14(b)).

No segundo experimento o robô foi controlado por um humano para re-



(a) Ensinando as trajetórias.



(b) Execução do modelo aprendido.

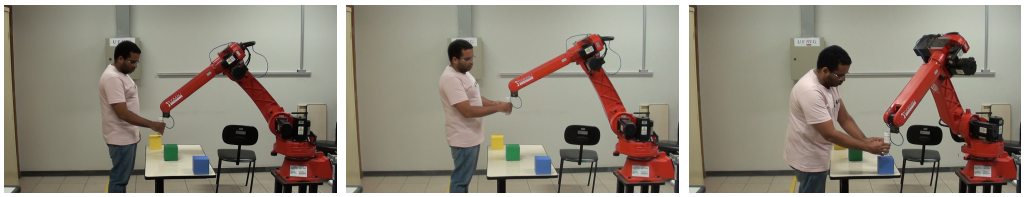
Figura 5.14: Primeiro experimento usando o manipulador industrial.

produzir 3 trajetórias similares à trajetória verde na Figura 5.16(a). Como no experimento anterior, cada trajetória foi representada por 51 pontos discretos e as posições também foram obtidas utilizando a cinemática direta implementada pelo sistema do robô. O modelo foi estimado com grau de não linearidade $\ell = 3$. Em vez do SEA, que foi definido apenas para 2D, o modelo em 3D foi aprendido usando a métrica RMSE para medir a precisão da reprodução de um conjunto de demonstrações $\mathbf{Y}_{(n)}$. RMSE é calculado por:

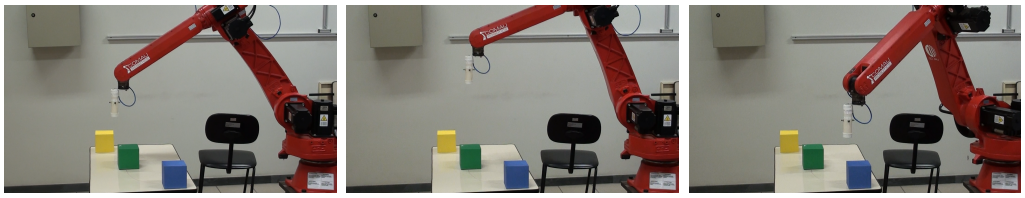
$$\text{RMSE} = \frac{\sqrt{\sum_{m=1}^{n}\sum_{t=1}^{T^m}\left(\mathbf{Y}_{(m,t)} - \hat{\mathbf{Y}}_{(m,t)}\right)^2}}{\sqrt{\sum_{m=1}^{n}\sum_{t=1}^{T^m}\left(\mathbf{Y}_{(m,t)} - \bar{\mathbf{Y}}_{(\mathbf{m})}\right)^2}},$$

onde T^m é o número de pontos na mth demonstração, $\bar{\mathbf{Y}}_{(\mathbf{m})}$ é o valor médio da trajetória m, $\hat{\mathbf{Y}}_{(m,t)}$ é o ponto t da trajetória $\hat{\mathbf{Y}}_{(m)}$ que inicia no mesmo ponto de partida da demonstração $\mathbf{Y}_{(m)}$ e é gerada pelo modelo.

Imagens das fases de treinamento e da execução do robô podem ser vistas nas Figuras 5.15(a) e 5.15(b), respectivamente. Para avaliar o modelo, algumas execuções foram realizadas inicializando o robô em partes distintas do espaço de trabalho, como mostrado na Figura 5.16(b). Um vídeo desse e

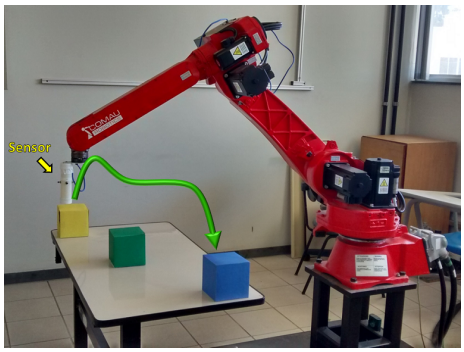


(a) Ensinando as trajetórias.

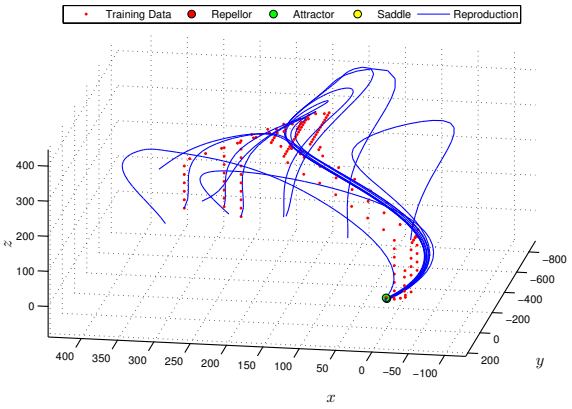


(b) Execução do modelo aprendido.

Figura 5.15: Segundo experimento usando o manipulador industrial.



(a) Configuração do segundo experimento.



(b) Trajetórias realizadas pelo manipulador industrial.

Figura 5.16: (a) Imagem contendo a configuração do segundo experimento e (b) o resultado das trajetórias inicializando o robô em partes distintas do espaço de trabalho.

de outros experimentos podem ser visto em https://goo.gl/AqMLAH.

5.4 Discussão

Os resultados das seções anteriores mostram que o NAR-RM pode ser usado para ensinar robôs por demonstrações. O método foi comparado com outros métodos do estado da arte usando um *benchmark*, onde foi mostrado que os

resultados obtidos são compatíveis com os anteriores e que o método pode ser considerado a melhor alternativa quando é necessária uma boa precisão nas trajetórias aprendidas e, ao mesmo tempo, um baixo esforço para o controlador de baixo nível.

Outra possível vantagem da abordagem proposta em relação aos métodos anteriores é a simplicidade e compactação do modelo obtido. Além disso, diferentemente de outros modelos (por exemplo, redes neurais), o modelo NAR pode se beneficiar da vasta literatura sobre identificação de sistemas que é capaz de usar a estrutura e os parâmetros do modelo para extrair algumas características do sistema (Aguirre e Mendes, 1996; Aguirre et al., 2002, 2005).

Em relação à garantia de estabilidade, até o momento não há um resultado teórico geral para garantir a estabilidade global dos sistemas não lineares. Até mesmo as técnicas que utilizam a teoria de Lyapunov, geralmente, dependem da busca pelo maior domínio de atração possível, dentro do qual o sistema é estável. Essa busca baseia-se em uma escolha pré-especificada, frequentemente subótima, de uma função candidata Lyapunov definida positiva, como em SEDS (Khansari-Zadeh e Billard, 2011) e em τ-SEDS (Neumann e Steil, 2015). Outra possibilidade é o uso de um sinal externo para garantir a estabilidade como em CLF-DM (Khansari-Zadeh e Billard, 2014). Outras abordagens, como BM (Khansari-Zadeh e Billard, 2010), o NIVF (Lemme et al., 2013) e agora, o NAR-RM, apresentam apenas garantias de estabilidade assintótica local ao alvo. Este fato não é uma desvantagem significativa, uma vez que a principal característica desejável do sistema aprendido é que o formato das trajetórias executadas se assemelhem às demonstrações. Isso não é garantido para todos os métodos estáveis globais, que podem gerar movimentos totalmente diferentes para pontos que estão longe do alvo. Por-

tanto, a maior vantagem dos métodos globais, que é uma boa robustez para as perturbações causada pela existência de um valor infinito de distância-α, pode ser igualada por distâncias-α razoavelmente grandes. Isto é confirmado pelos resultados numéricos apresentados anteriormente, que indicam que o NAR-RM pode lidar com perturbações típicas que ocorrem no movimento do robô e ainda ter boa reprodutibilidade de trajetória.

O método NAR-RM foi o mais lento se comparado aos métodos do estado da arte usando o benchmark. Como discutido anteriormente, isso acontece pois são realizados testes extras para garantir que o movimento esteja dentro da bacia alvo de atração. No entanto, o método ainda é bastante rápido e aplicável ao controle de sistemas reais, como ilustrado nas subseções anteriores usando o robô móvel e o manipulador. De fato, uma implementação em C++ da fase de execução do método em um computador Core i5 leva apenas 38 μs, em média, para ser executada.

Sobre o tempo computacional da fase de aprendizado, foi mostrado que ele depende do número de pontos utilizados para representar as trajetórias das demonstrações e também do grau de não linearidade do modelo. Uma proposta para melhorar o tempo computacional na fase de aprendizado do método original NAR-RM foi apresentada e denominada NAR-RM2, que se mostrou uma boa alternativa.

Uma desvantagem do método proposto é a estimação da bacia de atração numericamente, pois o custo computacional depende do número de células do mapa discreto M, utilizado para representar o espaço de atuação do robô. Neste ponto, o primeiro problema está na escolha entre precisão e menor tempo computacional, ou seja, quanto menor o número de células mais rápido é a estimação da bacia de atração, pois o Algoritmo 2 itera sobre todas as células de M. Por outro lado, quanto menor o número de células, menos

precisa é a bacia de atração. O segundo problema é o acréscimo significativo no tempo computacional da bacia de atração, com o aumento do número de dimensões. Isto ocorre porque o número de células cresce de forma exponencial com o aumento da dimensão. Por exemplo, se as dimensões de M forem representadas pelo mesmo número de células C, o número de células existentes no mapa M será dado por C^D, onde D é o número de dimensões. Logo, um modelo 2D e 3D com $C = 70$ terão um mapa M contendo 4.900 células e 343.000 células, respectivamente. Nos experimentos apresentados nesta tese, o maior tempo de computação foi gasto na estimação das bacias de atração dos modelos.

Capítulo 6

Considerações finais

Esta tese apresentou uma metodologia para aprender movimentos de alcance de robôs a partir de um conjunto de demonstrações com o uso de modelos não lineares autorregressivos (NAR) polinomiais. Como os modelos NAR, obtidos usando técnicas padrões de identificação de sistemas, nem sempre modelam adequadamente os movimentos de alcance, foi proposto nesta tese a ideia inovadora de modificar a localização de alguns pontos fixos do modelo com o objetivo de obter ou melhorar modelos de movimentos de alcance. Utilizando a metodologia proposta foi desenvolvido um algoritmo capaz de aprender movimentos de alcance denominado de NAR-RM.

Para avaliar a precisão dos movimentos aprendidos, foi utilizado a varredura da área do erro (SEA) e introduzido uma nova métrica, chamada distância-α, para estimar a qualidade da bacia de atração do alvo. Os resultados experimentais mostram que o método NAR-RM é capaz de identificar (aprender) sistemas dinâmicos que modelam movimentos de alcance com boa precisão e robustez a perturbações temporais e espaciais. NAR-RM foi comparado a outros métodos do estado da arte, e foi mostrado que o método proposto é uma boa alternativa aos métodos atuais quando é necessária uma boa precisão nas trajetórias aprendidas e, ao mesmo tempo, uma baixa demanda do controlador de baixo nível. Experimentos utilizando robôs mostraram a

aplicabilidade do método para controlar sistemas reais.

6.1 Propostas de continuidade

Uma linha de investigação interessante na continuação desta tese seria na proposição de novas estratégias para escolha dos pontos fixos. Como mostrado na Subseção 5.1.2, utilizando a variação do método proposto denominado NAR-RM2, é possível melhorar o tempo computacional do algoritmo e ainda manter bons resultados avaliando somente alguns pontos fixos a cada iteração.

Outra investigação, também focada na melhora do desempenho computacional do método NAR-RM, seria na proposição de novas estratégias para estimar a bacia de atração numericamente. No algoritmo proposto nesta tese, cada estimativa da bacia é feita do zero, ou seja, não é levada em conta a bacia atual do modelo antes da imposição dos pontos fixos. Como discutido na Subseção 4.2.1, o modelo após a imposição de um ponto fixo pode ser visto como uma versão perturbada do modelo original, se a distância utilizada na imposição não for grande. Desta forma, tentar estimar a bacia de atração, após a imposição de um ponto fixo, utilizando a bacia de atração do modelo original pode diminuir o tempo computacional do algoritmo.

Ainda com o foco na melhoria do método NAR-RM, uma investigação interessante seria a automatização da escolha do parâmetro de grau de não linearidade ℓ. Alguns modelos podem ser aprendidos utilizando baixos valores de ℓ, enquanto outros precisam de valores maiores. Como o tempo computacional do método também depende do valor de ℓ, a escolha automática pode diminuir o tempo gasto na fase de aprendizado de muitos modelos. Além disso, em uma futura aplicação do método por pessoas "leigas", não só o grau de não linearidade deve ser escolhido de forma automática, como

também os demais parâmetros do método.

A aplicação da metodologia proposta na robótica móvel também é uma linha de investigação interessante. Um dos desafios seria a inclusão de restrições não-holonômicas no processo de estimação dos modelos. Como apresentado na Seção 5.2, é possível utilizar o método NAR-RM na robótica móvel, mas como o processo de estimação não inclui as restrições não-holonômicas do robô, podem existir regiões dentro da bacia de atração em que o campo vetorial não atenda a estas restrições. Assim, caso o robô atinja estas regiões, ele não seguirá o campo vetorial, o que pode levá-lo para fora da bacia de atração. Outro desafio da robótica móvel é o ambiente dinâmico. Neste caso, os modelos aprendidos devem ser capazes de se adaptar ao ambiente em que o robô está inserido e ao surgimento de situações inesperadas, ou seja, as informações sensoriais também devem ser consideradas. Uma possibilidade de investigação seria a junção do método NAR-RM com o método proposto por Nehmzow et al. (2006, 2007) que constroem um mapeamento sensor-motor utilizando as informações sensoriais das demonstrações. O método proposto em (Nehmzow et al., 2006, 2007) foi apresentado no final da Subseção 2.2.1.

Outras linhas de investigação seriam:

- Como combinar dois ou mais modelos obtidos pelo NAR-RM para compor movimentos mais complexos;
- Como adaptar o método NAR-RM para aprendizado por reforço; e
- Como adaptar o método NAR-RM para modelar movimentos de alcance que demandam informações de tempo.

Referências Bibliográficas

Aguirre, L. A. (2007). *Introdução à identificação de sistemas–Técnicas lineares e não-lineares aplicadas a sistemas reais*. Editora UFMG, 3ª edition.

Aguirre, L. A. e Billings, S. A. (1994). Validating identified nonlinear models with chaotic dynamics. *International Journal of Bifurcation and Chaos*, 4(01):109–125.

Aguirre, L. A. e Billings, S. A. (1995). Dynamical effects of overparametrization in nonlinear models. *Physica D: Nonlinear Phenomena*, 80(1):26–40.

Aguirre, L. A., Coelho, M., e Corrêa, M. (2005). On the interpretation and practice of dynamical differences between hammerstein and wiener models. *IEE Proceedings, Part D — Control Theory and Applications*, 152(4):349–356.

Aguirre, L. A., Corrêa, M., e Cassini, C. (2002). Nonlinearities in narx polynomial models: representation and estimation. *IEE Proceedings, Part D — Control Theory and Applications*, 149(4):343–348.

Aguirre, L. A. e Letellier, C. (2009). Modeling nonlinear dynamics and chaos: a review. *Mathematical Problems in Engineering*.

Aguirre, L. A. e Mendes, E. M. A. M. (1996). Global nonlinear polynomial models: Structure, term clusters and fixed points. *International Journal of Bifurcation and Chaos*, 6(02):279–294.

Akaike, H. (1974). A new look at the statistical model identification. *Automatic Control, IEEE Transactions*, 19(6):716–723.

Akanyeti, O., Nehmzow, U., e Billings, S. A. (2008). Robot training using system identification. *Robotics and Autonomous Systems*, 56(12):1027–1041.

Akanyeti, O., Nehmzow, U., e Billings, S. A. (2009). Complex robot training tasks through bootstrapping system identification. Em *Robotics and Biomimetics ROBIO 2008. IEEE International Conference on*, pgs. 2168–2173.

Akanyeti, O., Rañó, I., Nehmzow, U., e Billings, S. A. (2010). An application of Lyapunov stability analysis to improve the performance of narmax models. *Robotics and Autonomous Systems*, 58(3):229–238.

Argall, B. D., Chernova, S., Veloso, M., e Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483.

Bautista-Ballester, J., Vergés-Llahí, J., e Puig, D. (2014). Programming by demonstration: A taxonomy of current relevant methods to teach and describe new skills to robots. Em *ROBOT2013: First Iberian Robotics Conference*, pgs. 287–300. Springer.

Billard, A., Calinon, S., Dillmann, R., e Schaal, S. (2008). Robot programming by demonstration. Em *Springer handbook of robotics*, pgs. 1371–1394. Springer.

Billings, S. A., Chen, S., e Korenberg, M. (1989). Identification of MIMO non-linear systems using a forward-regression orthogonal estimator. *International Journal of Control*, 49(6):2157–2189.

Blocher, C., Saveriano, M., e Lee, D. (2017). Learning stable dynamical systems using contraction theory. Em *Proceedings of the International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2017)*.

Broadbent, E., Stafford, R., e MacDonald, B. (2009). Acceptance of healthcare robots for the older population: review and future directions. *International Journal of Social Robotics*, 1(4):319–330.

Calinon, S., Guenter, F., e Billard, A. (2007). On learning, representing, and generalizing a task in a humanoid robot. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(2):286–298.

Calinon, S., Sauser, E. L., Billard, A., e Caldwell, D. G. (2010). Evaluation of a probabilistic approach to learn and reproduce gestures by imitation. Em *Robotics and Automation (ICRA), IEEE International Conference on*, pgs. 2671–2676.

Chatzis, S. P., Korkinof, D., e Demiris, Y. (2012). A nonparametric bayesian approach toward robot learning by demonstration. *Robotics and Autonomous Systems*, 60(6):789–802.

Chen, S. e Billings, S. A. (1989). Representations of non-linear systems: the narmax model. *International Journal of Control*, 49(3):1013–1032.

Chen, S., Billings, S. A., e Luo, W. (1989). Orthogonal least squares methods and their application to non-linear system identification. *International Journal of control*, 50(5):1873–1896.

Choset, H. M. (2005). *Principles of robot motion: theory, algorithms, and implementation*. MIT press.

Çinar, A. (1995). Nonlinear time series models for multivariable dynamic processes. *Chemometrics and intelligent laboratory systems*, 30(1):147–158.

Connolly, C. I., Burns, J. B., e Weiss, R. (1990). Path planning using Laplace's equation. Em *Robotics and Automation. IEEE International Conference on*, pgs. 2102–2106.

d'Andréa Novel, B., Campion, G., e Bastin, G. (1995). Control of nonholonomic wheeled mobile robots by state feedback linearization. *The International Journal of Robotics Research*, 14(6):543–559.

Dautenhahn, K. e Nehaniv, C. L. (2002). The agent-based perspective on imitation. Em *Imitation in animals and artifacts*, pgs. 1–40. MIT Press.

de Lima, D. A. e Pereira, G. A. S. (2013). Navigation of an autonomous car using vector fields and the dynamic window approach. *Journal of Control, Automation and Electrical Systems*, 24(1-2):106–116.

Demiris, J. e Hayes, G. (1996). *Imitative learning mechanisms in robots and humans*. University of Edinburgh, Department of Artificial Intelligence.

Dias, A., Emiliano, J., Silva Pereira, G. A. S., e Palhares, R. M. (2015). Longitudinal model identification and velocity control of an autonomous car. *Intelligent Transportation Systems, IEEE Transactions on*, 16(2):776–786.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

Draper, N. R., Smith, H., e Pownell, E. (1998). *Applied regression analysis*, volume 3. John Wiley and Sons, New York.

Gardiner, B., Coleman, S. A., McGinnity, T. M., e He, H. (2012). Robot control code generation by task demonstration in a dynamic environment. *Robotics and Autonomous Systems*, 60(12):1508–1519.

Gonçalves, V. M., Pimenta, L. C., Maia, C. A., Dutra, B. C., e Pereira, G. A. S. (2010). Vector fields for robot navigation along time-varying curves in-dimensions. *Robotics, IEEE Transactions on*, 26(4):647–659.

Guckenheimer, J. e Holmes, P. (1983). Local bifurcations. Em *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*, pgs. 117–165. Springer.

Hägele, M., Nilsson, K., e Pires, J. N. (2008). Industrial robotics. Em *Springer handbook of robotics*, pgs. 963–986. Springer.

Hart, P. E., Nilsson, N. J., e Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107.

Hayes, G. M. e Demiris, J. (1994). *A robot controller using learning by imitation*. University of Edinburgh, Department of Artificial Intelligence.

Hoffmann, H., Pastor, P., Park, D.-H., e Schaal, S. (2009). Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance. Em *Robotics and Automation. ICRA IEEE International Conference on*, pgs. 2587–2592.

Huang, G.-B., Zhu, Q.-Y., e Siew, C.-K. (2006). Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501.

Hwang, J.-H., Arkin, R. C., e Kwon, D.-S. (2003). Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control. Em *Intelligent Robots and Systems (IROS)*, volume 2, pgs. 1444–1449.

Iglesias, R., Nehmzow, U., e Billings, S. A. (2008). Model identification and model analysis in robot training. *Robotics and Autonomous Systems*, 56(12):1061–1067.

Ijspeert, A., Nakanishi, J., e Schaal, S. (2003). Learning control policies for movement imitation and movement recognition. Em *Neural information processing system*, volume 15, pgs. 1547–1554.

Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P., e Schaal, S. (2013). Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373.

Ijspeert, A. J., Nakanishi, J., e Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. Em *Robotics and Automation. ICRA IEEE International Conference on*, volume 2, pgs. 1398–1403.

Kashyap, R. L. (1977). A Bayesian comparison of different classes of dynamic models using empirical data. *Automatic Control, IEEE Transactions*, 22(5):715–727.

Kerr, D., Nehmzow, U., e Billings, S. A. (2010). Towards automated code generation for autonomous mobile robots. Em *3d Conference on Artificial General Intelligence (AGI-2010)*. Atlantis Press.

Khansari-Zadeh, S. M. (2012). *A dynamical system-based approach to modeling stable robot control policies via imitation learning*. PhD thesis, Citeseer.

Khansari-Zadeh, S. M. e Billard, A. (2010). BM: An iterative algorithm to learn stable non-linear dynamical systems with gaussian mixture models. Em *Robotics and Automation (ICRA), IEEE International Conference*, pgs. 2381–2388.

Khansari-Zadeh, S. M. e Billard, A. (2011). Learning stable nonlinear dynamical systems with gaussian mixture models. *Robotics, IEEE Transactions*, 27(5):943–957.

Khansari-Zadeh, S. M. e Billard, A. (2012). A dynamical system approach to realtime obstacle avoidance. *Autonomous Robots*, 32(4):433–454.

Khansari-Zadeh, S. M. e Billard, A. (2014). Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, 62(6):752–765.

Khansari-Zadeh, S. M., Kronander, K., e Billard, A. (2012). Learning to play minigolf: A dynamical system-based approach. *Advanced Robotics*, 26(17):1967–1993.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98.

Kim, S., Shukla, A., e Billard, A. (2014). Catching objects in flight. *Robotics, IEEE Transactions on*, 30(5):1049–1065.

Korenberg, M., Billings, S. A., Liu, Y., e McIlroy, P. (1988). Orthogonal parameter estimation algorithm for non-linear stochastic systems. *International Journal of Control*, 48(1):193–210.

Kronander, K., Khansari-Zadeh, M. S., e Billard, A. (2011). Learning to control planar hitting motions in a minigolf-like task. Em *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pgs. 710–717.

Kronander, K., Khansari-Zadeh, S. M., e Billard, A. (2015). Incremental motion learning with locally modulated dynamical systems. *Robotics and Autonomous Systems*, 70:52–62.

Kulić, D., Ott, C., Lee, D., Ishikawa, J., e Nakamura, Y. (2012). Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research*, 31(3):330–345.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press.

Lawitzky, M., Medina, J. R., Lee, D., e Hirche, S. (2012). Feedback motion planning and learning from demonstration in physical robotic assistance: differences and synergies. Em *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pgs. 3646–3652. IEEE.

Lee, D. e Nakamura, Y. (2006). Stochastic model of imitating a new observed motion based on the acquired motion primitives. Em *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, pgs. 4994–5000.

Lee, D., Ott, C., Nakamura, Y., e Hirzinger, G. (2011). Physical human robot interaction in imitation learning. Em *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pgs. 3439–3440. IEEE.

Lemme, A., Meirovitch, Y., Khansari-Zadeh, M., Flash, T., Billard, A., e Steil, J. J. (2015). Open-source benchmarking for learned reaching motion generation in robotics. *Paladyn, Journal of Behavioral Robotics*, 6(1).

Lemme, A., Neumann, K., Reinhart, R. F., e Steil, J. J. (2013). Neurally imprinted stable vector fields. Em *ESANN*. Citeseer.

Leontaritis, I. J. e Billings, S. A. (1985a). Input-output parametric models for non-linear systems part i: deterministic non-linear systems. *International journal of control*, 41(2):303–328.

Leontaritis, I. J. e Billings, S. A. (1985b). Input-output parametric models for non-linear systems part ii: stochastic non-linear systems. *International journal of control*, 41(2):329–344.

Lohmiller, W. e Slotine, J.-J. E. (1998). On contraction analysis for nonlinear systems. *Automatica*, 34(6):683–696.

Martins, M. F. e Demiris, Y. (2010). Learning multirobot joint action plans from simultaneous task execution demonstrations. Em *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pgs. 931–938. International Foundation for Autonomous Agents and Multiagent Systems.

McLachlan, G. e Peel, D. (2004). *Finite mixture models*. John Wiley & Sons.

Monteiro, L. H. A. (2011). *Sistemas dinâmicos*. Editora Livraria da Física, 3 edition.

Mühlig, M., Gienger, M., Hellbach, S., Steil, J. J., e Goerick, C. (2009). Task-level imitation learning using variance-based movement optimization. Em *Robotics and Automation (ICRA). IEEE International Conference*, pgs. 1177–1184.

Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., Schaal, S., e Kawato, M. (2004). Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47(2):79–91.

NAR-RM (2017). Learning reaching motions using NAR. https://bitbucket.org/coroufmg/nar-rm.

Narayanan, K. K., Posada, L. F., Hoffmann, F., e Bertram, T. (2013). Acquisition of behavioral dynamics for vision based mobile robot navigation from demonstrations. Em *Mechatronic systems*, number 1, pgs. 37–44.

Nehmzow, U., Akanyeti, O., e Billings, S. A. (2010). Towards modelling complex robot training tasks through system identification. *Robotics and Autonomous Systems*, 58(3):265–275.

Nehmzow, U., Akanyeti, O., Weinrich, C., Kyriacou, T., e Billings, S. A. (2007). Robot programming by demonstration through system identification. Em *Intelligent Robots and Systems. IROS. IEEE/RSJ International Conference on*, pgs. 801–806.

Nehmzow, U., Iglesias, R., Kyriacou, T., e Billings, S. A. (2006). Robot learning through task identification. *Robotics and Autonomous Systems*, 54(9):766–778.

Neumann, G., Daniel, C., Paraschos, A., Kupcsik, A., e Peters, J. (2014). Learning modular policies for robotics. *Frontiers in computational neuroscience*, 8.

Neumann, K., Lemme, A., e Steil, J. J. (2013). Neural learning of stable dynamical systems based on data-driven Lyapunov candidates. Em *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference*, pgs. 1216–1222.

Neumann, K. e Steil, J. J. (2015). Learning robot motions with stable dynamical systems under diffeomorphic transformations. *Robotics and Autonomous Systems*, 70:1–15.

Ott, C., Lee, D., e Nakamura, Y. (2008). Motion capture based human motion recognition and imitation by direct marker control. Em *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pgs. 399–405. IEEE.

Pastor, P., Hoffmann, H., Asfour, T., e Schaal, S. (2009). Learning and generalization of motor skills by learning from demonstration. Em *Robotics and Automation. ICRA IEEE International Conference on*, pgs. 763–768.

Pereira, G. A. S. e Chaimowicz, L. (2007). Robôs móveis. Em Aguirre, L. A., Silva, A. P. A. d., Campos, M. F. M., e Amaral, W. C. d., editores, *Enciclopédia de Automática: Controle e Automação*, volume 3, pgs. 369–386, São Paulo. Editora Blucher.

Perk, B. E. e Slotine, J. J. E. (2006). Motion primitives for robotic flight control. *arXiv preprint cs/0609140*.

Perrin, N. e Schlehuber-Caissier, P. (2016). Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems. *Systems & Control Letters*, 96:51–59.

Pimenta, L. C., Fonseca, A. R., Pereira, G. A. S., Mesquita, R. C., Silva, E. J., Caminhas, W. M., e Campos, M. F. M. (2006). Robot navigation based on electrostatic field computation. *IEEE Transactions on Magnetics MAG*, 42(4):1459.

Prassler, E., Munich, M. E., Pirjanian, P., e Kosuge, K. (2016). Domestic robotics. Em *Springer handbook of robotics*, pgs. 1729–1758. Springer.

Rimon, E. e Koditschek, D. E. (1992). Exact robot navigation using artificial potential functions. *Robotics and Automation, IEEE Transactions on*, 8(5):501–518.

Santos, R., Pereira, G. A. S., e Aguirre, L. A. (2018). Learning robot reaching motions by demonstration using nonlinear autoregressive models. *Robotics and Autonomous Systems*, 107:182–195.

Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242.

Schaal, S., Atkeson, C., e Vijayakumar, S. (2002). Scalable locally weighted statistical techniques for real time robot learning. *Applied Intelligence-Special issue on Scalable Robotic Applications of Neural Networks*, 17(1):49–60.

Schaal, S., Ijspeert, A., e Billard, A. (2003). Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 358(1431):537–547.

Schaal, S., Peters, J., Nakanishi, J., e Ijspeert, A. (2005). Learning movement primitives. Em *Robotics Research. The Eleventh International Symposium*, pgs. 561–572. Springer.

Seeger, M. (2004). Gaussian processes for machine learning. *International journal of neural systems*, 14(02):69–106.

Slotine, J. J. E. e Li, W. (1991). *Applied nonlinear control*. Prentice-Hall Englewood Cliffs, NJ.

Soh, H. e Demiris, Y. (2015). Learning assistance by demonstration: smart mobility with shared control and paired haptic controllers. *Journal of Human-Robot Interaction*, 4(3):76–100.

Tellex, S., Kollar, T., Dickerson, S., Walter, M. R., Banerjee, A. G., Teller, S. J., e Roy, N. (2011). Understanding natural language commands for robotic navigation and mobile manipulation. Em *AAAI*, volume 1, pg. 2.

Thrun, S., Burgard, W., e Fox, D. (2005). *Probabilistic robotics*. MIT press.

Tso, S. e Liu, K. (1996). Hidden Markov model for intelligent extraction of robot trajectory command from demonstrated trajectories. Em *Industrial Technology, 1996.(ICIT'96), Proceedings of The IEEE International Conference on*, pgs. 294–298.

Ude, A., Gams, A., Asfour, T., e Morimoto, J. (2010). Task-specific generalization of discrete and periodic dynamic movement primitives. *Robotics, IEEE Transactions on*, 26(5):800–815.

Valavanis, K. P., Hebert, T., Kolluru, R., e Tsourveloudis, N. (2000). Mobile robot navigation in 2-d dynamic environments using an electrostatic potential field. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 30(2):187–196.

Yang, H.-D., Park, A.-Y., e Lee, S.-W. (2007). Gesture spotting and recognition for human–robot interaction. *IEEE Transactions on robotics*, 23(2):256–270.

Yun, X. e Yamamoto, Y. (1992). On feedback linearization of mobile robots. Technical report, University of Pennsylvania, USA.

Zhao, Y., Xiong, R., Fang, L., e Dai, X. (2014). Generating a style-adaptive trajectory from multiple demonstrations. *International Journal of Advanced Robotic Systems*, volume 11.

Apêndice A

Estimador estendido de mínimos quadrados

O estimador estendido de mínimos quadrados modela os resíduos de identificação como um processo de média móvel (MA) da seguinte forma:

$$e(k) = c\nu(k-1) + \nu(k), \tag{A.1}$$

em que $\nu(k)$ é ruído branco. Desta forma, os termos $\nu(k-1)$ podem ser adicionados à matriz de regressores e os seus parâmetros c ao vetor de parâmetros do modelo da Equação (3.8) como apresentado a seguir:

$$\mathbf{y}^* = \Psi^*\theta^* + \boldsymbol{\xi}^*, \tag{A.2}$$

sendo que $\mathbf{y}^* = \mathbf{y}$, $\theta^* = [\theta \vdots c]^T$, $\boldsymbol{\xi}^* = [\nu(k) \dots \nu(k+N-1)]^T$ e

$$\Psi^* = \begin{bmatrix} & \vdots & \nu(k-1) \\ & \vdots & \nu(k) \\ \Psi & \vdots & \nu(k+1) \\ & \vdots & \vdots \\ & \vdots & \nu(k+N-2) \end{bmatrix}.$$

Nota-se que $\boldsymbol{\xi}^*$ é uma variável aleatória branca, de forma que $\boldsymbol{\xi}^*$ não está correlacionado com Ψ^* na Equação (A.2). Logo o estimador de mínimos quadrados estendido

$$\hat{\theta}^*_{\text{EMQ}} = [\Psi^{*T}\Psi^*]^{-1}\Psi^{*T}\mathbf{y} \tag{A.3}$$

não apresentaria polarização, ou seja, $E[\hat{\theta}^*] = [\theta^T c]^T$.

É importante salientar que a parte com média móvel (MA) é utilizada somente na fase de estimação de parâmetros do modelo NARX para reduzir a polarização, sendo que as predições são feitas utilizando somente a parte determinística do modelo (Chen et al., 1989). Outro detalhe importante é que para a Equação (A.3), é preciso estimar de forma iterativa os regressores $\nu(k-i)$, pois eles não são medidos. Um algoritmo iterativo pode ser encontrado em (Aguirre, 2007).